NAME → Ayush Goyal

Roll no → A-25 (AIML)

Subject → TOC

No → 8767593598

UNIT - 1

* History:

1930 → Alan Turing (Father of modern computer science (1912 - 1954)) [Turing Machines]

↳ Decidability

every | Problem

- Decidable (Solvable)
  - Tracable (P - problem)
    - ↳ Polinomial form
  - Intracable (NP - Problem)
    - NP - complete (convertible to polinomial)
    - NP - hard [1969 - Cook]
- Non-decidable (Non solvable)

* Mathematical Notation & Techniques :

⇨ {SETS}

↳ A group of objects. The objects in a set are called elements of set.

Ex. {1,2,3} , {1,.... 99}

Note :→ Set {1,3,3,3,55} ≡ Set {1,3,5}.

Set builder Notation

Ex. O = {x | x = odd +ve integer}

Some common sets

N = {x | Natural numbers} = {1,2....}

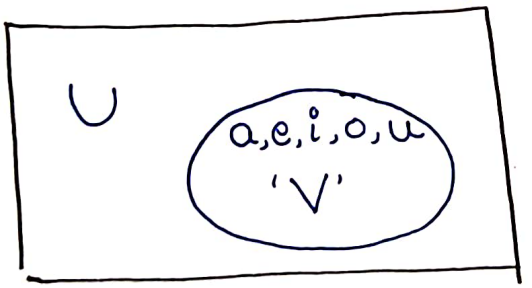
Z = {x | Integers} = {... -2,-1,0,1....}

Z = {x | Positive integers} = {1,2,3...}

R = {x | Real Numbers} ⇒ (All real nos.)

Set described by Venn diagram

Ex Draw 'V', set of vowels in english

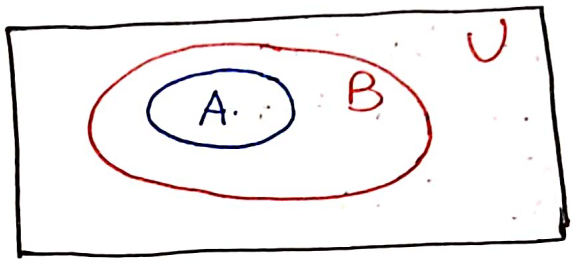


Note:→ A set with no elements is called empty set (ϕ)

- **Subset** → A is subset of B if B contains all the elements of A.
  ↳ Notation → $A \subseteq B$ [A is subset of B]

- If $A \neq B$ and A is B's subset, then A is called as a **proper subset**
  → $A \subset B$

Ex



- Set of all subsets is called as a **power set** P(S) or $\mathcal{P}(2^S)$

Ex $P(\{0,1,2\}) = \{\phi, \{0\}, \{1\}, \{2\}, \{0,1\}, \{0,2\}, \{1,2\}, \{0,1,2\}\}$

* Set operations

① Union $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$

② Intersection $A \cap B = \{x \mid x \in A \text{ and } x \in B\}$

③ Difference $A - B = \{x \mid x \in A \text{ and } x \notin B\}$

④ Complement $\overline{A} = \{x \mid x \in U \text{ and } x \notin A\}$

where U → "universal set"
⇓
contains all elements.

* Set Identities

(1) A] $A \cup \phi = A$

B] $A \cap U = A$

(2) A] $A \cup U = U$

B] $A \cap \phi = \phi$

(3) A] $A \cup A = A$

B] $A \cap A = A$

(4) $(\bar{A}) = A$

(5) A] $A \cup B = B \cup A$

B] $A \cap B = B \cap A$

(6) A] $A \cup (B \cup C) = (A \cup B) \cup C$

B] $A \cap (B \cap C) = (A \cap B) \cap C$

(7) A] $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$

B] $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$

(8) A] $\overline{A \cup B} = \bar{A} \cap \bar{B}$

B] $\overline{A \cap B} = \bar{A} \cup \bar{B}$

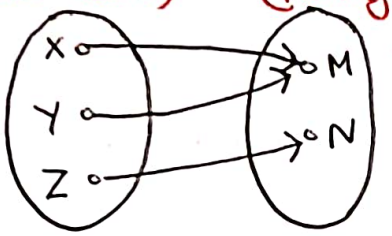
⇒ Functions & Relations

↳ A fⁿ f from set A to B is assignment of one element of B to each element of A.

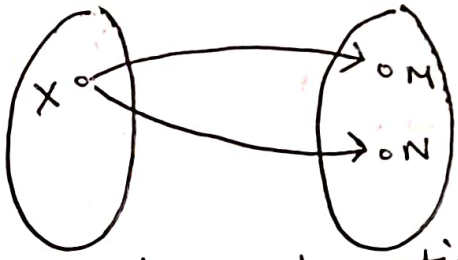
↳ $f: A \to B$ [f is a fⁿ from A to B]

↳ $f(a) = b$ [b is a unique element of B assigned by fⁿ 'f' to the element 'a' of A]

(domain) (Range)



A function



Not a function.

- A relation R from A to B is a subset of $A \times B$.

∴ R → A set of pairs

If $(a,b) \in R$; a has a relation R with b, denoted as $\underline{aRb}$

* Languages & Grammars

- Alphabets (Σ) → A set of symbols (non-empty)
- Sentences → Strings of symbols
- Language → A set of sentences of finite length.
- Grammar → A finite set of rules for defining a language.

⇨ Strings → A string/word is a finite sequence of symbols chosen from Σ.

→ Empty string ⇒ epsilon (ε) $\boxed{\therefore |w| = 0}$

→ Length of string ⇒ $|w|$
↳ No. of non-ε characters.

→ Σ^k ⇒ set of all strings of length k

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

then → $\Sigma^0 = \{\varepsilon\}$

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

$\Sigma^2 = \{00, 01, 10, 11\}$

$\Sigma^3 = \{000, 001, 010, 011 \cdots 111\}$

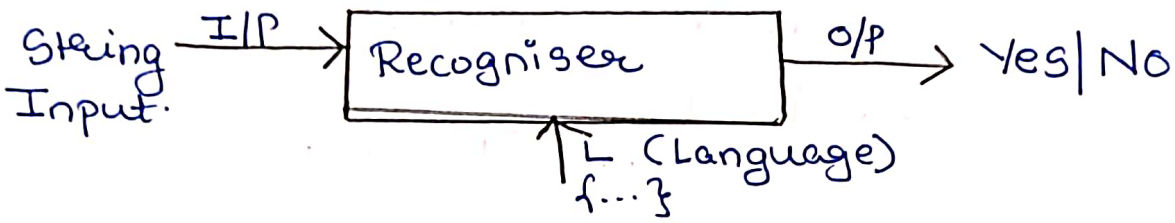
Set of all strings over given alphabets. ← $\Sigma^* = \{\Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \ldots\ldots \text{Infinite}\}$

→ [{yob} → Invalid string
{boy} → Valid string]

→ ϕ (phy) → Represents empty language, not even ε.

Note:→ L is a language over alphabet Σ only if $L \subseteq \Sigma^*$ → Doesn't include invalid words.
↳ Subset

▷ Recogniser problem:



▷ Generation problem:

Ex. Addition [2 + 3 = 5]
I/P O/P

→ Every language
1. → Syntax: Set of rules to write valid statements.
2. → Semantics: Must have meaning
↳ (Logical)

• Natural language (Eng, french) have very complex rules of syntax and not necessarily well defined.

* Formal Grammar

↳ Well written grammar with no ambiguity.
↳ Well defined set of rules for syntax

→ Lang. with formal grammar is called formal language.

⇓

Representation

⇨ $G = (V, T, P, S)$
(V: can be replaced; S: cannot be replaced)

where,

V → Set of non-terminal symbols [all capital]
T → Set of terminal symbols [All small]
P → Set of production rules.
S → start symbol.

Q.1] Find the language for the given production rules:→

(A) $S \to 0S1$
$S \to 01$

Step 1: Find V, T, P, S.
Step 2: Find valid strings using product^n rules.

Soln] $V = \{S\}$
$T = \{0, 1\}$
$P = \{S \to 0S1,\ S \to 01\}$
$S = S$

(1)	(2)	(3)
S	S	S
0S1	01	0S1
0011		00S11
		000111

$\therefore L = \{0^n, 1^n;\ n \geq 1\}$

ε → Not valid.

(B) $S \to aA$
$A \to aA \mid b$

Soln] $V = \{S, A\}$
$T = \{a, b\}$
$P = \{S \to aA$
$A \to aA$
$A \to b\}$
$S = S$

Valid strings →

S	S	S
aA	aA	aA
aaA	aaA	ab
aab	aaaA	
	aaab	

$\therefore L = \{a^n, b ; n \geq 1\}$

(C) $S \to 0A1$
$A \to A1 \mid a$

Soln] $V = \{S, A\}$
$T = \{0, 1, a\}$
$P = \{S \to 0A1$
$A \to A1$
$A \to a\}$
$S = S$

valid strings →

S	S	S
0A1	0A1	0A1
0a1	0A11	0A11
	0a11	0A111
		0a111

$\therefore L = \{0a1^n ; n \geq 1\}$

(D) $S \to Ab$
$A \to aAb \mid \varepsilon$

Soln] Valid strings →

S	S	S	S
Ab	Ab	Ab	Ab
aAbb	aAbb	aAbb	εb
aεbb	aaAbbb	aaAbbb	b
abb	aaεbbb	aaaAbbbb	
	aabbb	aaaεbbbb	
		aaabbbb	

$\therefore L = \{a^n . b^{n+1} ; n \geq 0\}$
↳ $\in$ is valid

(e) $S \to Aa$
$A \to B$
$B \to Aa \mid b$

Soln] $V = \{S, A, B\}$
$T = \{a, b\}$
$P \Rightarrow \{ S \to Aa$
$A \to B$
$B \to Aa \mid b \}$
$S = S$

Valid strings →

S	S	S
Aa	Aa	Aa
Ba	Ba	Aaa
ba	Aaa	Aaaa
	baa	baaa

$\therefore L = \{ b, a^n ; n \geq 1 \}$

(f) $S \to aA \mid \varepsilon$
$A \to bS$

Soln] $V = \{S, A\}$
$T = \{a, b, \varepsilon\}$
$P \to \{ S \to aA$
$S \to \varepsilon$
$A \to bS \}$
$S = S$

Valid strings →

S	S	S
ε	aA	aA
	abε	abS
		abaA
		abab

$\therefore L = \{ (ab)^n ; n \geq 0 \}$, ε is valid

(g) $S \to aSc \mid B$
$B \to bBc \mid \varepsilon$

Soln]

S	S	S	S	S	S
B	B	aSc	aSc	aSc	aSc
ε	bBc	aBc	aBc	aBc	aaScc
	bc	ac	abBcc	abBcc	aaBcc
			abcc	abbBccc	aacc.
				abbccc	

$\therefore L = \{ a^m b^n c^n c^m ; m, n \geq 0 \}$
$= \{ a^m b^n c^{n+m} ; m, n \geq 0 \}$

(h) $S \to aSbb$

$S \to \varepsilon$

Soln]

S	S	S
ε	$aSbb$	$aSbb$
	abb	$aaSbbbb$
		$aabbbb$

$\therefore L = \{a^n b^{2n} ; n \geq 0\}$

(i) $S \to aaA$

$A \to aAb$

$A \to ab$

Soln]

S	S	S
aaA	aaA	aaA
$aaab$	$aaaAb$	$aaaAb$
	$aaaabb$	$aaaaAbb$
		$aaaaabbb$

$\therefore L = \{a^{n+2} b^n ; n \geq 1\}$

(j) $S \to aSa \mid bSb \mid C \to L = \{wcw^R : w \in \{a,b\}^*\}$

(k) $S \to 0S1S1S \mid 1S1S0S \mid 1S0S1S \mid \in$

* The Chomsky Hierarchy

⇨ Types of Grammar (4 types)

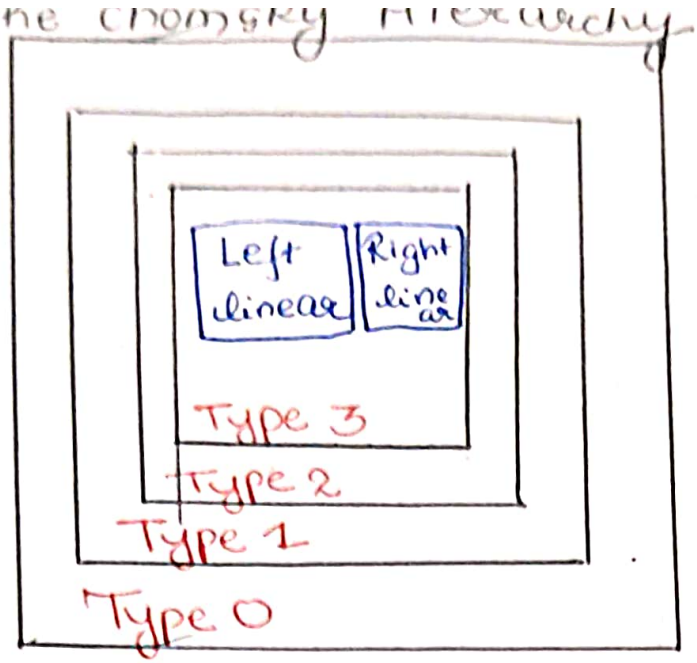
- **Type 0 → Unrestricted grammar**
  - ↳ No rules for writing production rules. Ex. $aAb \to ab$ (contracting)
  - ↳ There must be at least 1 variable on LHS of production.

- **Type 1 → Context sensitive grammar**
  - ↳ Non contracting
  - ↳ Length (left) ≤ l (right)
    Ex. $Aa \to ba$

- **Type 2 → Context free grammar**
  - ↳ l (LHS) = 1
    Ex. $A \to a$

- **Type 3 → Regular grammar**
  - ① ↳ l (LHS) = 1
  - ② ↳ RHS must have only 1 Non-terminal and either at 1st or last place

    Ex $A \to Aa$ ✓ (left linear grammar)
    $B \to bB$ ✓ (right linear grammar)
    $A \to aBa$ ✗



Note → The type of lang. is decided by the type of grammar used to write it
ex Type 2 lang is given by type 2 grammar

Q.2] Identify the type of grammar.

i) $S \to aBCD$
$B \to aAb$
$aD \to ad$
⇒ Type 3 ×
Type 2 ×
Type 1 ✓ [context sensitive]

ii) $S \to aBcD$
$B \to aAb$
$D \to d$
⇒ Type 3 ×
Type 2 ✓ [context free]

iii) $S \to aB$
$B \to aA$
$D \to d$
⇒ Type 3 ✓
[Regular right linear]

iv) $S \to Aa$
$A \to bB$
$B \to C$
⇒ Type 2 [context free]

Note → Type 3 must be either Right or left linear (only 1)

Tip → In exam, try to give the max possible type (type 2 > type 1) ex

↳ Recognisers of each type

Type 0 → Turing Machines
Type 1 → Linear Bounded Automata
Type 2 → Push down Automata
Type 3 → Finite automata

Q.3] Find the type & recognizer for the foll. grammars.

i) $S \to ACaB$
$Ca \to aaC$
$CBD \to DB$
$D \to abb$

⇒ Type 0 → unrestricted grammar
⇒ Turing Machines

ii) $S \to AAa$
$A \to C|Ba$
$Ba \to bca$

⇒ Type 1 → context sensitive grammar
⇒ Linear Bounded Automata

iii) $S \to ASB|d$
$A \to aA$

⇒ Type 2 → Context free
⇒ Push down automata

⇨ Equivalent Grammars

↳ Two grammars are equivalent if they generate the same strings / language.

Q.4] Are the given languages equivalent?

G1: $S \to aSb \mid \in$ G2: $S \to aAb \mid \in$, $A \to aAb \mid \in$

Soln]

⇒ The languages generated by these grammars is as follows:→

G1: S	S	S
aSb	aSb	∈
ab	aasbb	
	aabb	

$\Rightarrow L = \{a^n b^n ; n \geq 0\}$

G2: S	S	S	S
aAb	aAb	∈	aAb
ab	aaAbb		aaAbb
	aabb		aaaAbbb
			aaabbb

$\Rightarrow L = \{a^n b^n ; n \geq 0\}$

grammars

∴ Both ~~lang~~ generate the same lang:

thus $G(1) \equiv G(2)$ Hence Proved

⇨ Ambiguious grammar

↳ Same string can be derived in more than one way.

Q5] Is the given grammar ambiguious?

Soln] $S \to assb \mid bssa \mid \in$

S	S
assb	assb
abssasb	asb
abab	abssab
	abab

∴ we can conclude that this grammar is ambiguious grammar.

Hence Proved

Q.6] Check if the foll. strings are valid or not :→

(A) $E \to E+E$
$E \to E*E$
$E \to id$

String => id + id * id

Tip :→ 2 methods to solve.

Method I → Derivation method (Start from 'S' & derive to generate the string)

Step 1

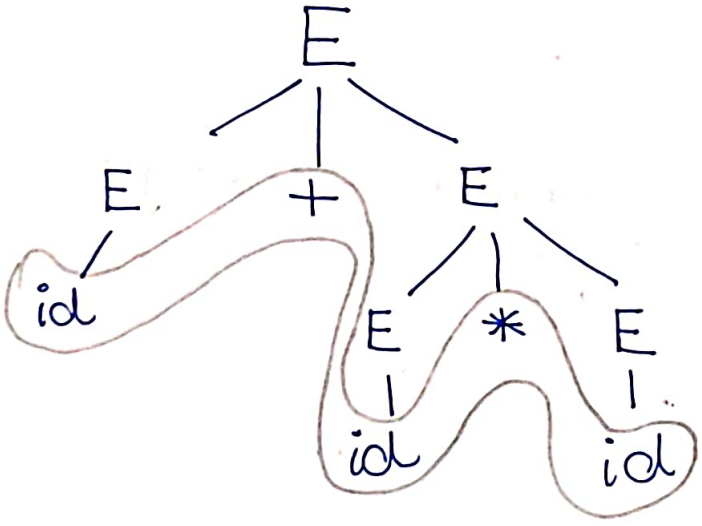
Leftmost derivation ⇓

E
$E + E$
$id + E$
$id + E * E$
$id + id * E$
$\Rightarrow$ id + id * id

Rightmost deri. ⇓

E
$E + E$
$E + E * E$
$E + E * id$
$E + id * id$
$\Rightarrow$ id + id * id

Step 2 → Draw Parse tree [Same for leftmost, Rightmost]



Method 2 → Reduction Method (Start with string & apply rules to get S)

Step I

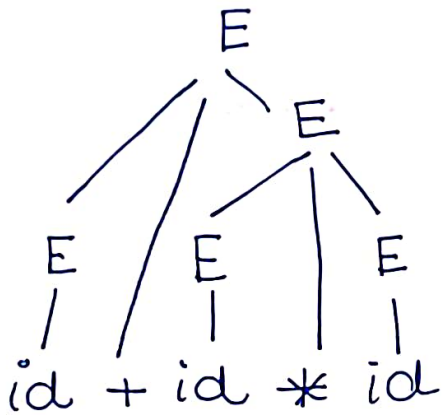
Leftmost reduct^n

id + id * id
E + id * id
E + E * id
E * id
E * E
→ E

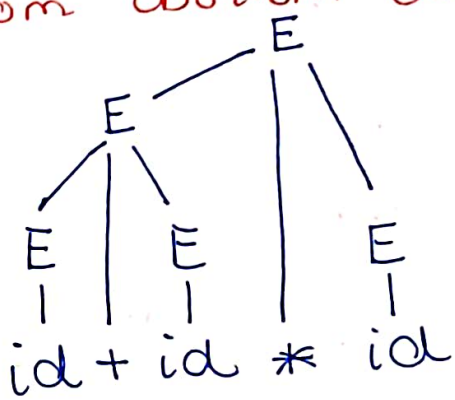
Rightmost Redn

id + id * id
id + id * E
id + E * E
id + E
E + E
→ E

Step II → Draw parse tree [Tree is generated from bottom to top]



Rightmost parse tree



Leftmost parse tree

(B) $S \to a \mid \wedge \mid (T)$
$T \to T,S \mid S$
String $\Rightarrow ((a,a),a)$

Soln] Derivation Method

Leftmost

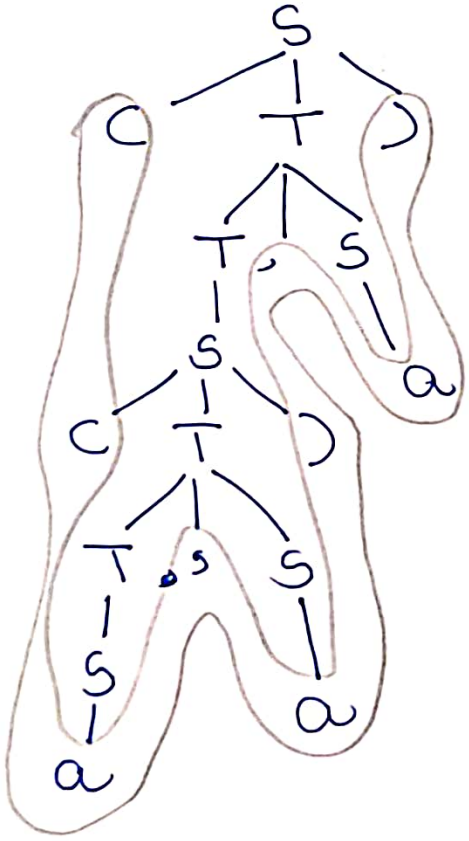
S
(T)
(T,S)
(S,S)
((T),S)
((T,S),S)
((S,S),S)
((a,S),S)
((a,a),S)
$\Rightarrow ((a,a),a)$

Rightmost

S
(T)
(T,S)
(T,a)
(S,a)
((T),a)
((T,S),a)
((T,a),a)
((S,a),a)
$\Rightarrow ((a,a),a)$

Now,

Parse tree



Reduction Method

Leftmost

((a,a),a)

((S,a),a)

((S,S),a)

((T,S),a)

((T),a)

(S,a)

(T,a)

(T,S)

(T)

⇒ S

Rightmost

((a,a),a)

((a,a),S)

((a,S),S)

((S,S),S)

((T,S),S)

((T),S)

(S,S)

(T,S)

~~(S,S)~~

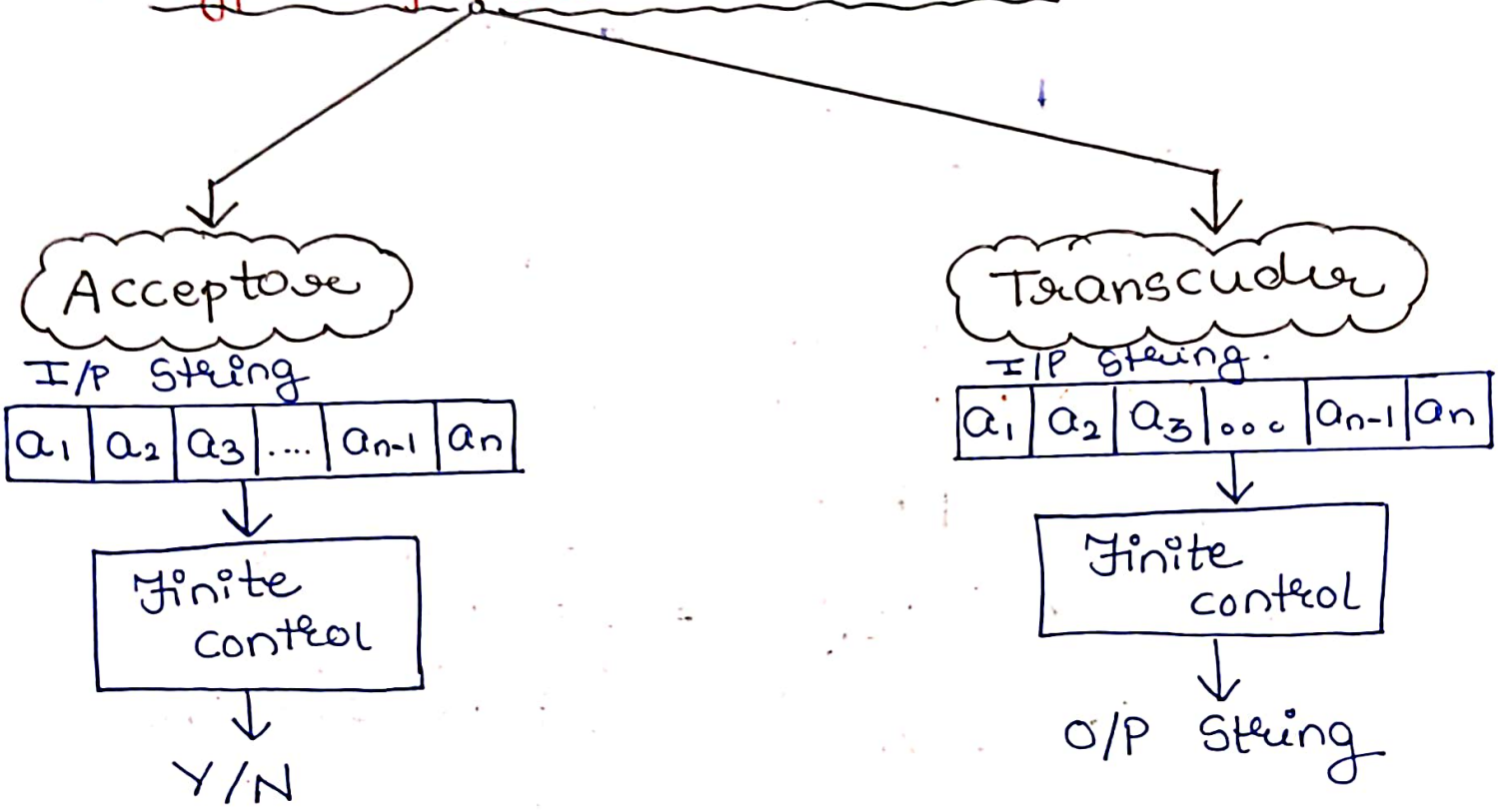
(T)

⇒ S

UNIT - II

FINITE AUTOMATA

⇨ Types of Finite Automata



⇨ Finite automata is used to recognise regular grammar's/language's valid strings

⇨ Defining / Representation of FA.

① Notations:

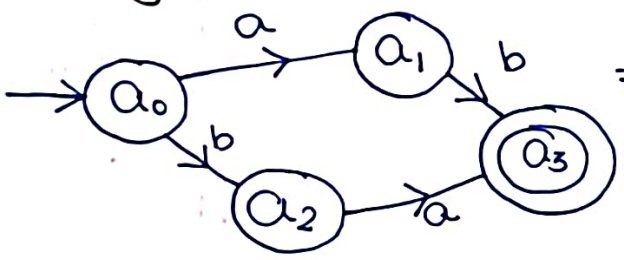
$M = \{Q, \Sigma, \delta, Q_0, F\}$

where,

- $Q \rightarrow$ set of states in FA
- $\Sigma \rightarrow$ set of I/P symbols
- $\delta \rightarrow$ Transition function (How states change)
- $Q_0 \rightarrow$ Initial state of FA.
- $F \rightarrow$ Set of final states.

② Using Graphs:

Ex



$Q = \{a_0, a_1, a_2, a_3\}$

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

$\delta \rightarrow$ (table needed)

$Q_0 = a_0$ (Incoming arrow)

$F = \{a_3\}$ (double circle)

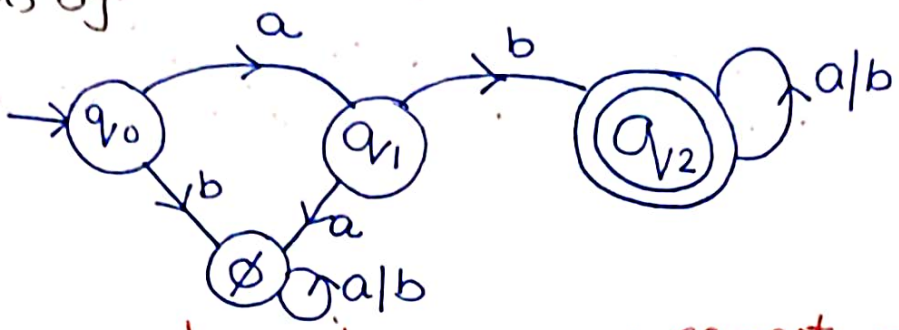
③ Transition table:

δ	a	b
$\rightarrow a_0$	a_1	a_2
a_1	ϕ	a_3
a_2	a_3	ϕ
$* a_3$	ϕ	ϕ

$L = \{ab, ba\}$

Q1] Design a finite automata to accept all strings starting with 'ab' and $\Sigma = \{a, b\}$

Soln]



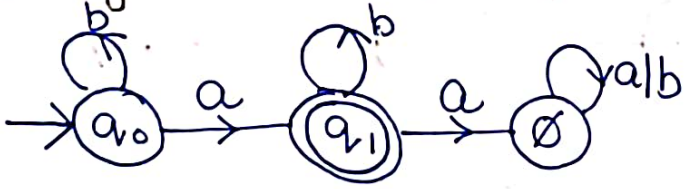
↳ Trap state → cannot come out of this state.

⇒ Transition table

δ	a	b
(initial) → q_0	q_1	$\emptyset$
q_1	$\emptyset$	q_2
(final) * q_2	q_2	q_2

Q2] Design a finite automata which accepts a set of string of 'a' & 'b' where all strings have exactly 1 'a'. $\Sigma = \{a, b\}$

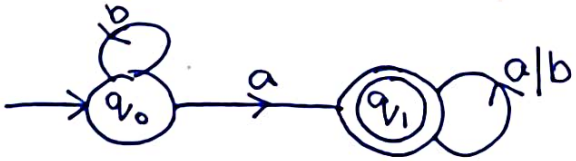
Soln]



Q3] All strings with atleast 1 a's

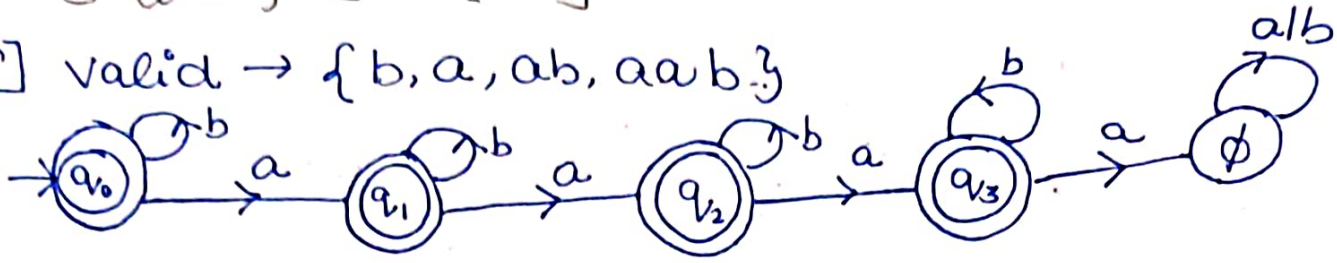
Soln] Step ① → Generate valid strings

{a, ab, ba, aab, baa ... }



Q4] All strings with no more than 3 a's ; $\Sigma = \{a, b\}$

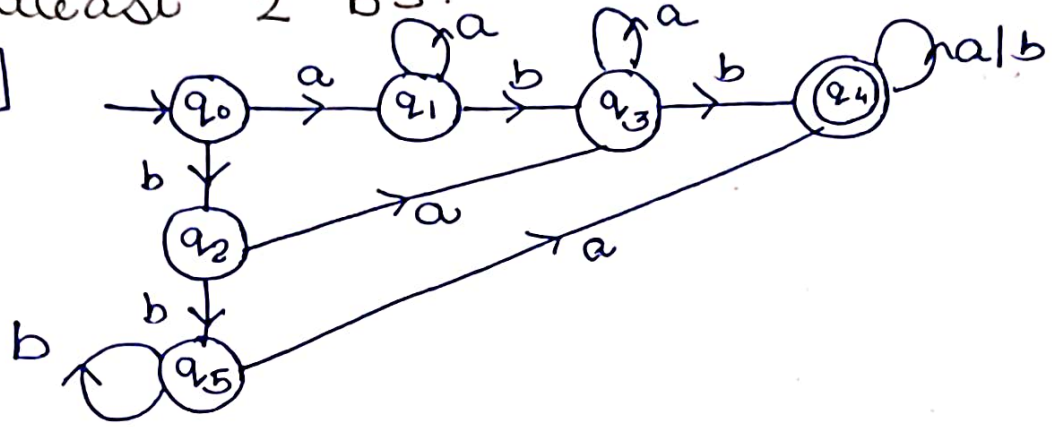
Soln] valid → {b, a, ab, aab}



• HW problems

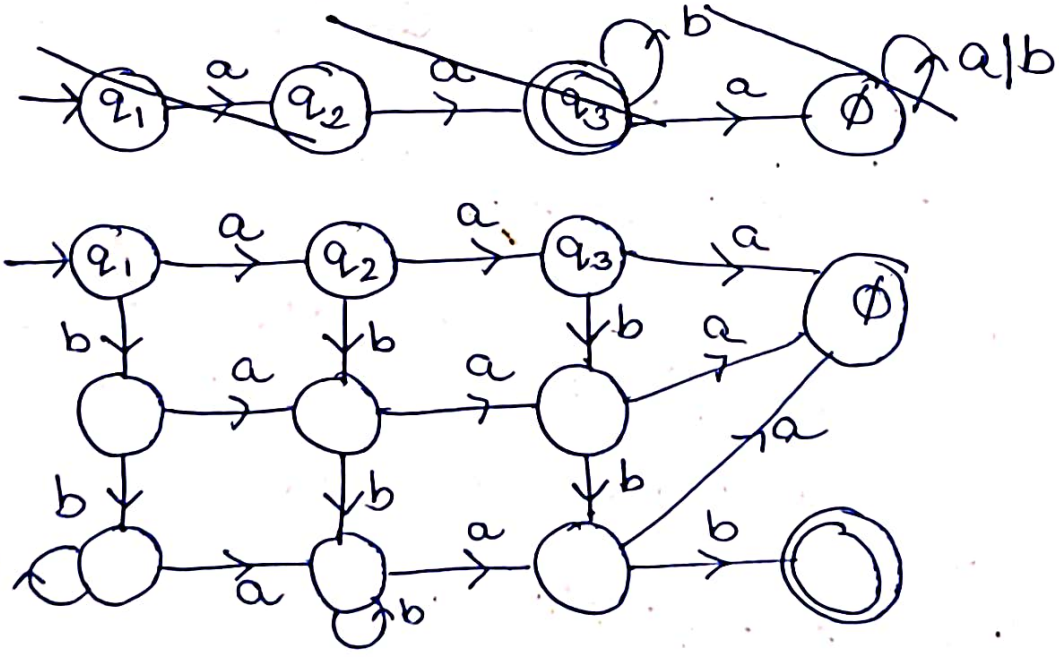
① All strings with atleast 1 a and atleast 2 b's.

Soln]



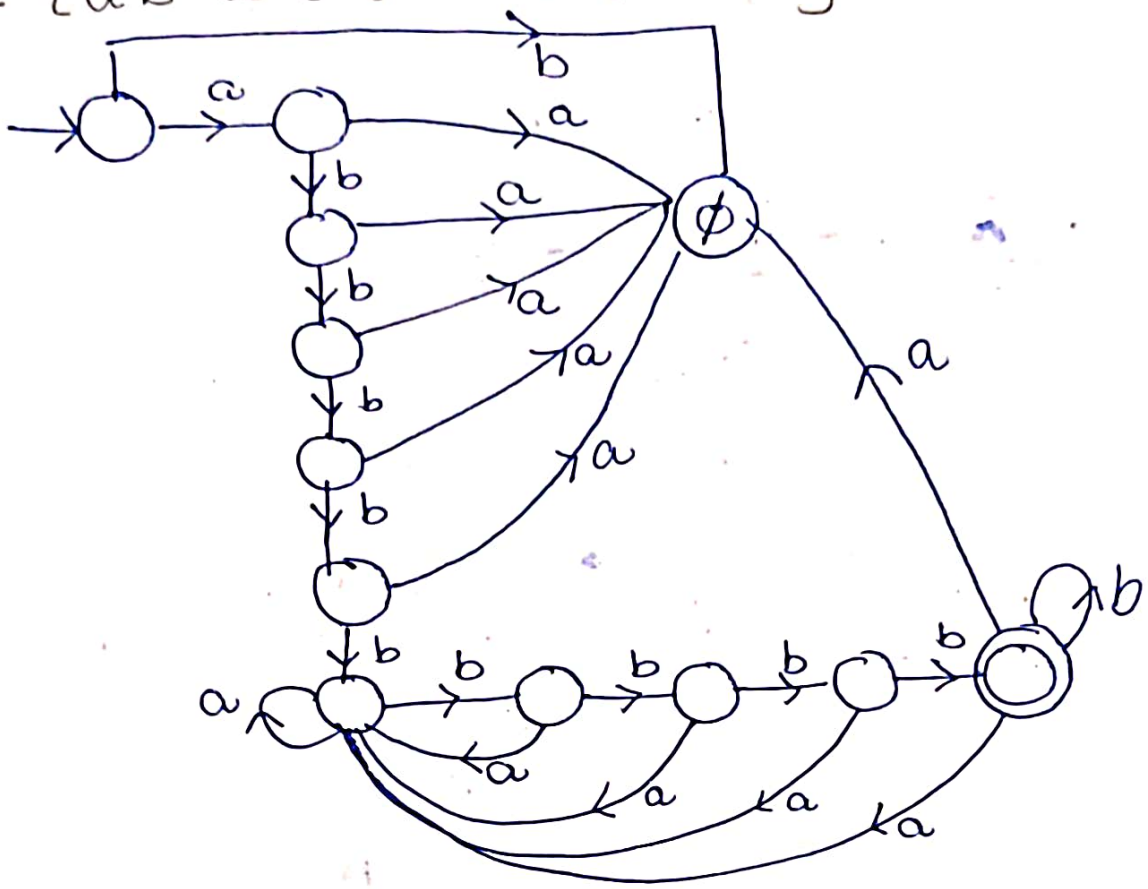
② All strings exactly 2as and more than 2 b's.

Soln]



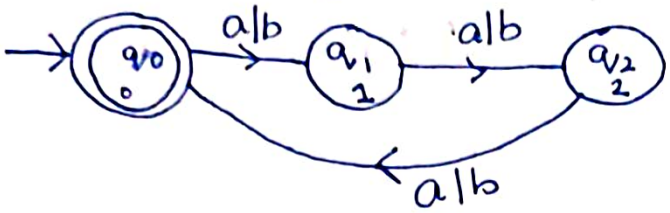
(3) $L = \{ a b^5 \omega b^4 ; \omega \in \{a,b\}^* \}$

Soln]



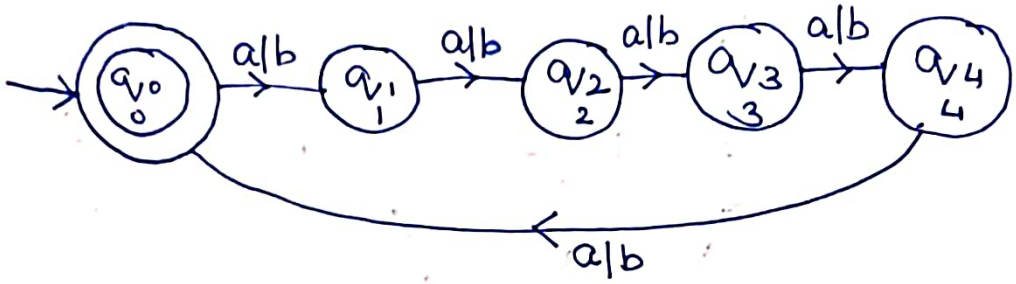
Q5] $L \rightarrow \{w : |w| \bmod 3 = 0\}$. $\Sigma = \{a, b\}$

Soln] valid strings $\rightarrow \{\varepsilon, aaa, bbb, aba, bab, aaaaaa \ldots\}$



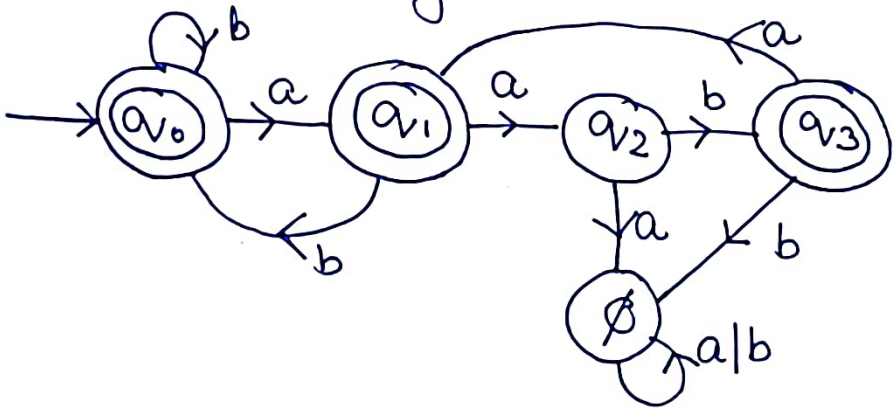
Q6] $L \rightarrow \{w : |w| \bmod 5 = 0\}$

Soln]



Q.7] Every string 'aa' is followed by a single 'b'.

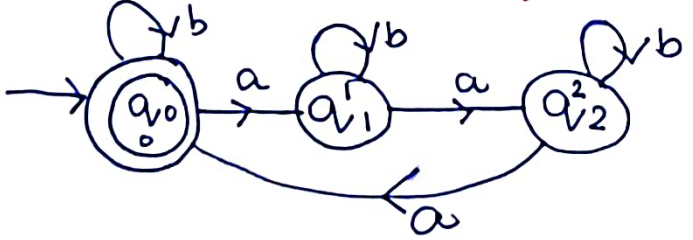
Soln] valid strings $\rightarrow \{aab, ababaab, bbbaab\}$



Q8] $L = \{w : Na(w) \bmod 3 = 0\}$

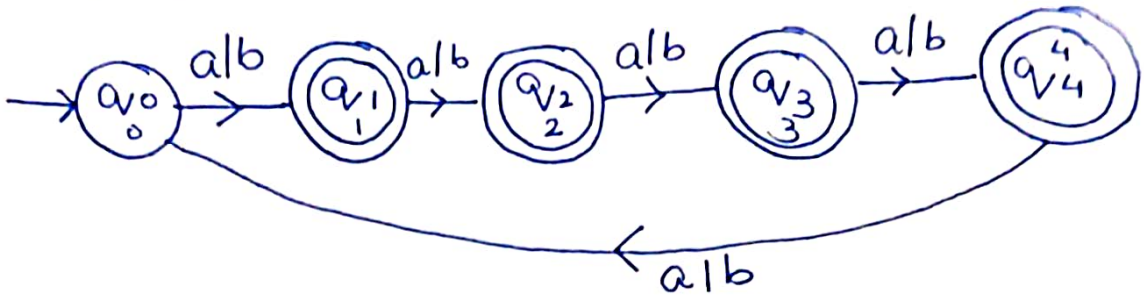
$Na(w)$ → No. of 'a' in 'w'.

Soln]



Q.9] $L = \{w; |w| \bmod 5 \neq 0\}$

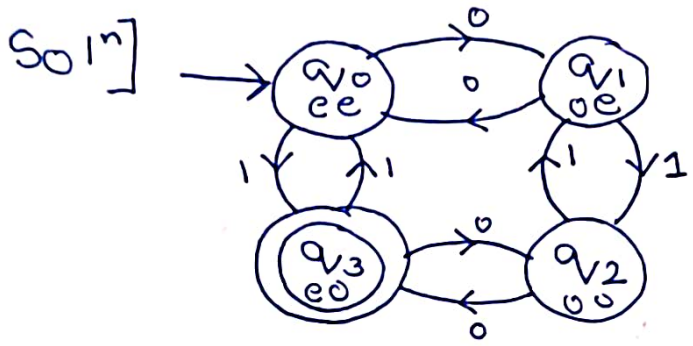
Soln]



Q.10] Even number of 0s and odd 1s.

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

Soln]

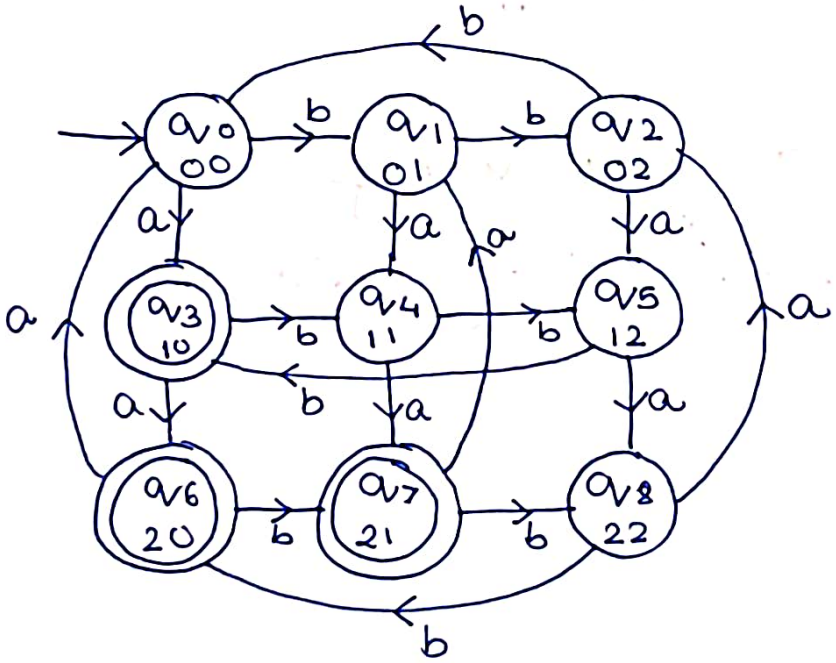


0s	1s
e	e
e	o
o	e
o	o

✱ Q.11] $L = \{w : Na(w) \bmod 3 > Nb(w) \bmod 3\}$

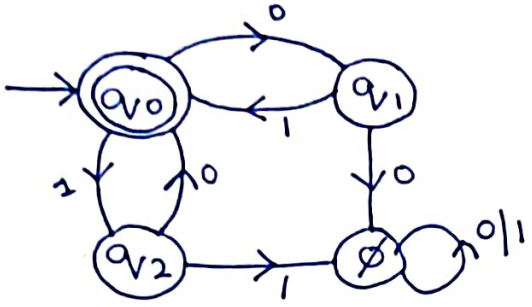
Soln] Total possible combinations of mod of a, b:

a	b	
0	0	
0	1	
0	2	
1	0	(v)
1	1	
1	2	
2	0	(valid)
2	1	(v)
2	2	



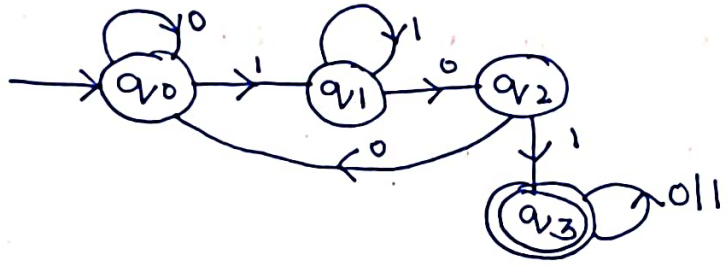
Q.12] $\Sigma \rightarrow \{0,1\}$. Equal no. of 0s and 1s and no prefix of string should contain 2 or more 0s than 1s and two or more 1s than 0s.

Soln] valid strings $\{\epsilon, 01, 10, 1010, 0110, 1001 \cdots\}$



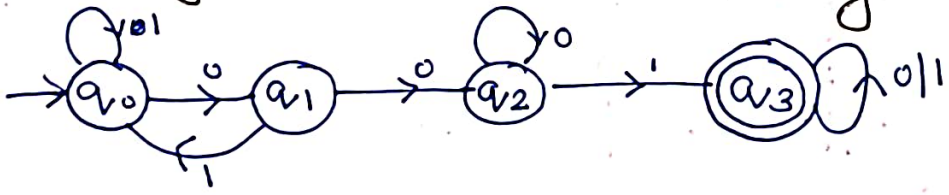
Q13] All strings over 0 & 1 which contain {101} as a substring.

Soln]



Q.14] {001} as a substring.

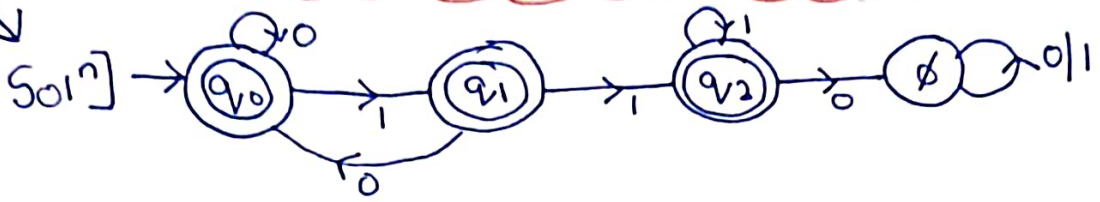
Soln]



Q.15] Design DFA which do not contain {110} as a substring.

Tip:→ 1st design DFA which contains the given string, and then swap the final and non final states.

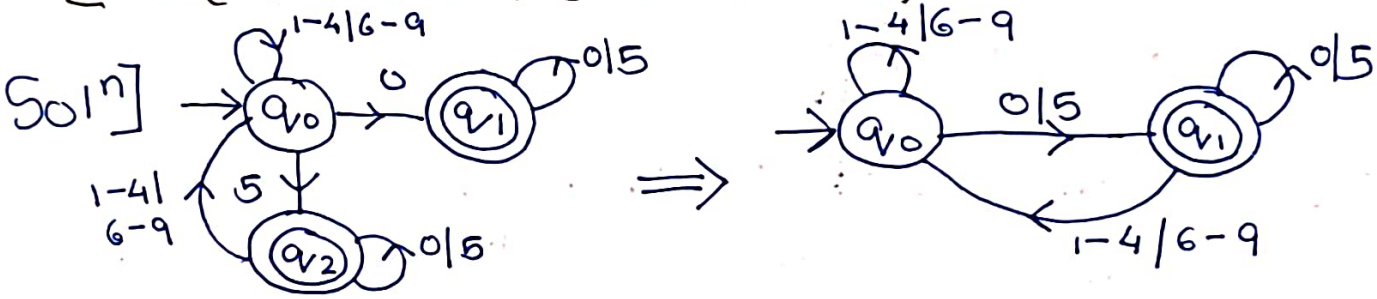
Soln]



Q16] Design finite automata, which will take decimal no. as input and all strings divisible by 5 are valid.

$\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

Soln]



Tip:→ No. of states = No. of all possible remainders

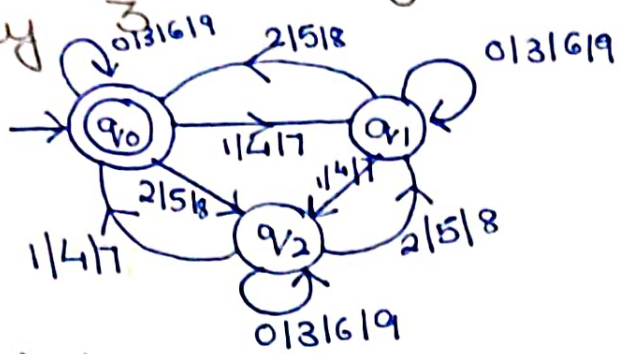
HW.

Q.17] $\Sigma = \{0-9\}$. String valid if divisible by 3.

Soln] Next page.

Q.17] $\Sigma = \{0-9\}$. String valid if divisible by 3

Soln]



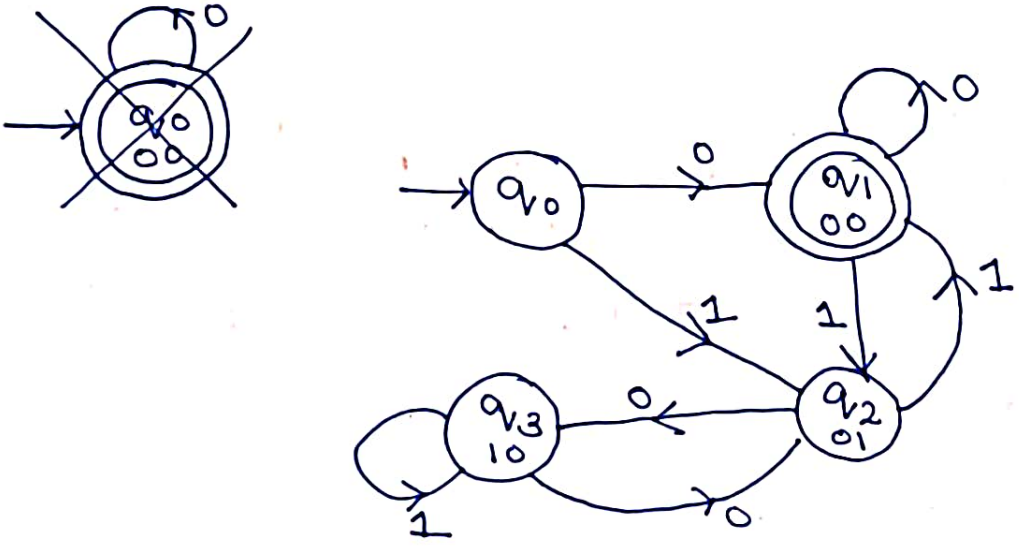
Now, ↳ Transform this such that Σ is not valid.

Soln]

Q.18] $\Sigma = \{0,1\}$ check if binary string is divisible by 11. (↳3)

Soln] Remainders ⇒ 00, 01, 10 [3]

∴ we will have 3 states.

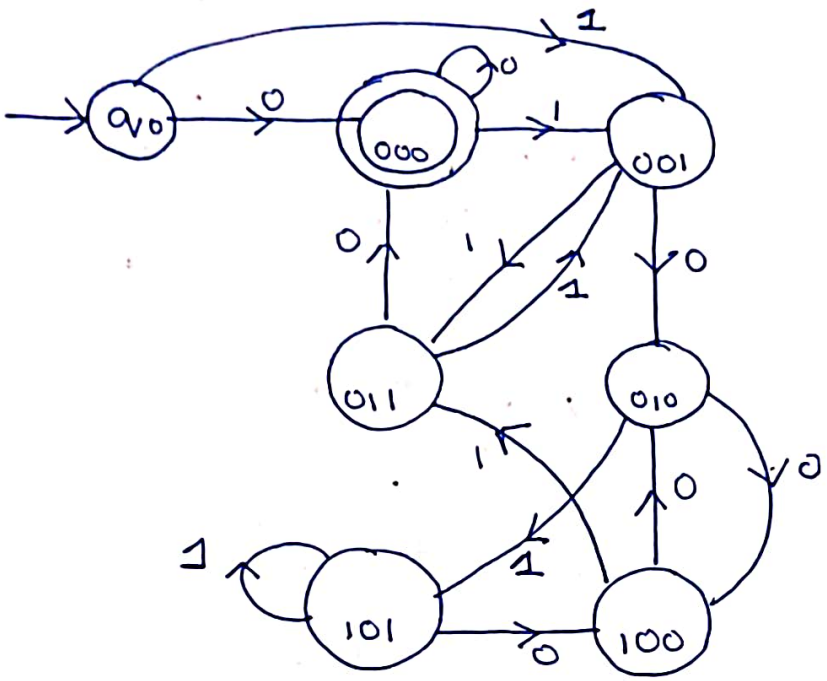


Q.19] $\Sigma = \{0,1\}$ check if string divisible by '110'

Soln] 6 remainders (0 to 5) + 1 for Epsilon

∴ Total 7 states.

ε
000
001
010
011
100
101

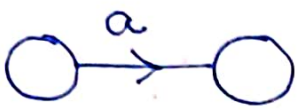


Finite Automata

Types

DFA

Deterministic Finite Automata



↳ Only one transition from a state on an input symbol

↳ No transition is not possible

↳ No 'ε' transition.

↳ Only one path from initial to final for a valid string.

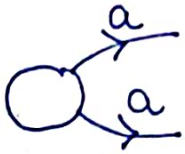
↳ Complex design

↳ Efficient

↳ Exactly one path.

NFA

Non-deterministic finite automata



↳ 'N' no. of transitn (0, 1, 2 --- n) for same input.

↳ No transition is possible

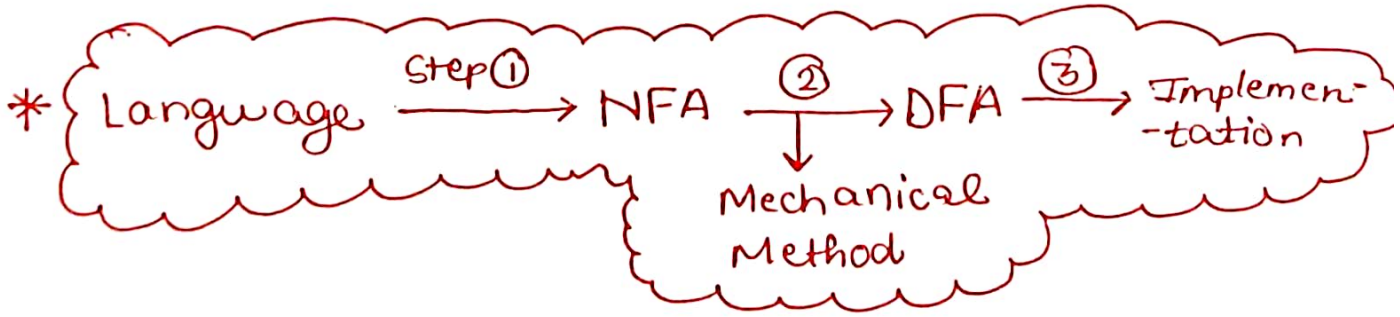
↳ 'ε' transition possible

↳ Multiple paths for a valid string.

↳ Simple design

↳ Ineffitient

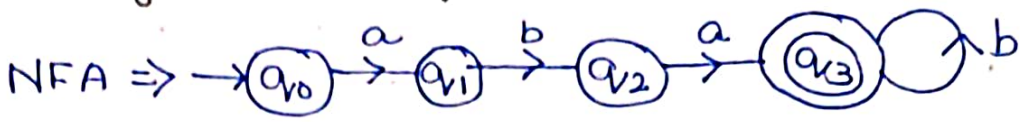
↳ Trial & error method.



Designing NFA

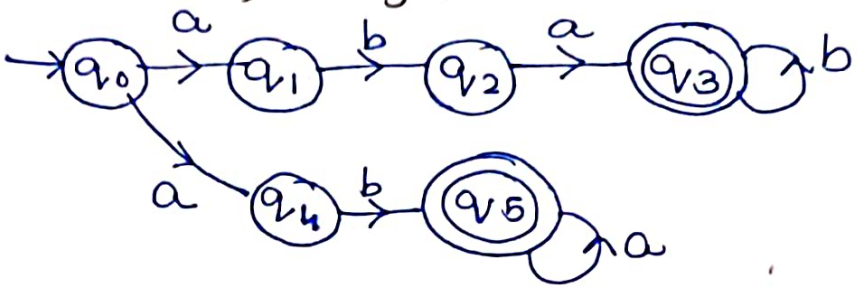
Q.20] Design NFA for (i) $L = \{abab^n ; n \geq 0\}$

Sol^n] NFA ⇒



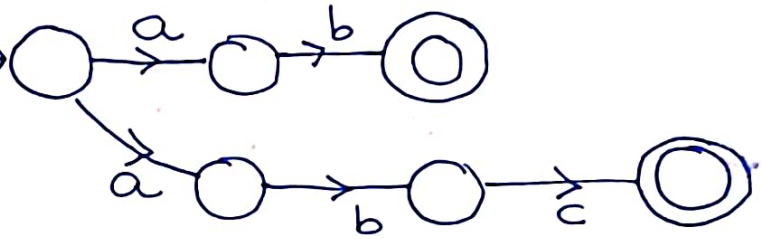
(ii) $L = \{abab^n ; n \geq 0\} \cup \{aba^m ; m \geq 0\}$

Sol^n]



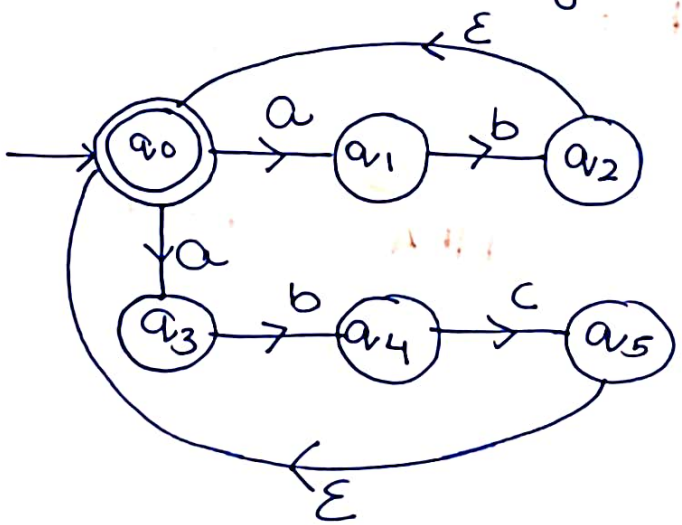
(iii) $L = \{ab, abc\}$

Sol^n]



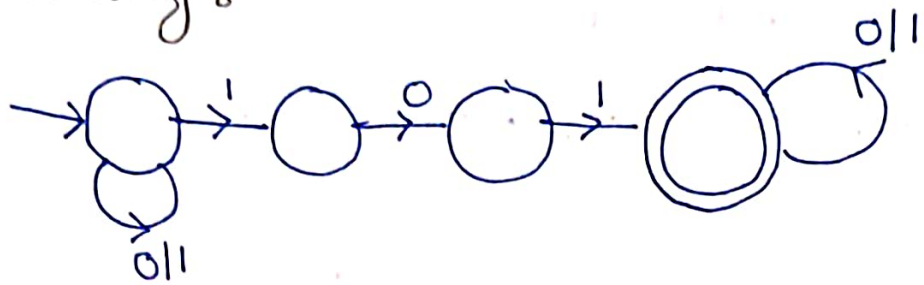
(iv) $L = \{ab, abc\}^*$

Sol^n] valid strings → {ε, ab, abc, abab, abcabc ...}



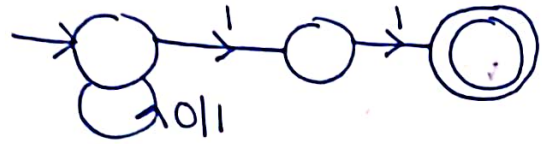
(V) String of {0,1} containing 101 as a substring.

Soln]



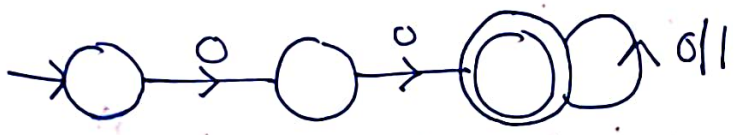
(VI) $\Sigma = \{0, 1\}$ ends with (1 0 1)

Soln]

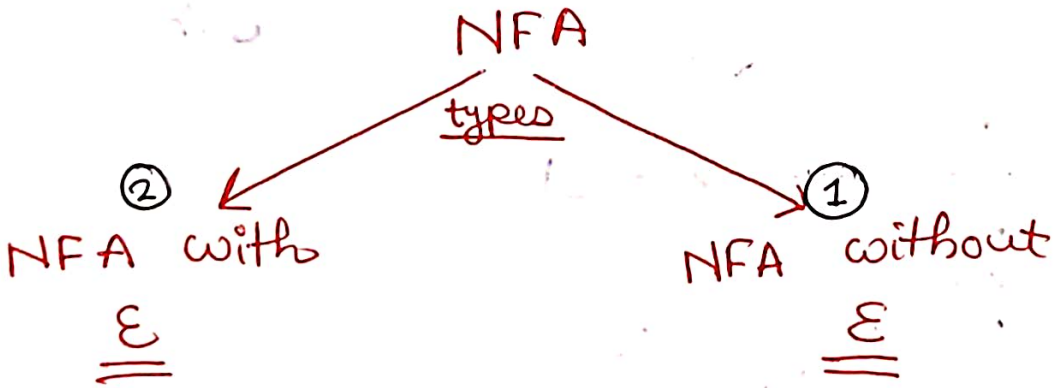


(VII) $\Sigma = \{0, 1\}$ starts with 0,0

Soln]



⇨ **Conversion from NFA to DFA**



① NFA without ε

→ NFA is defined as:→

$$M = \{Q, \Sigma, \delta, Q_0, F\}$$

where,

$Q \rightarrow$ set of states in NFA

$\Sigma \rightarrow$ set of I/P symbols.

$\delta \rightarrow$ Transition funct^n of NFA

$\delta : Q \times (\Sigma \cup \epsilon) \rightarrow Q$

$Q_0 \rightarrow$ Initial state of NFA

$F \rightarrow$ set of final states in NFA.

[equivalent DFA for any NFA]

→ (DFA) is defined as:→

$$M' = \{Q', \Sigma, \delta', Q_0', F'\}$$

where,

$Q' \rightarrow$ set of states in DFA

$\Sigma \rightarrow$ set of I/P symbols.

$\delta' \rightarrow$ Transition f^n in DFA.

$Q_0' \rightarrow$ Initial state of DFA.

$F' \rightarrow$ set of final states in DFA.

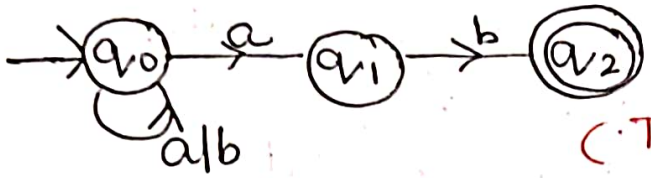
∴ For NFA without ε.

① $Q_0' = Q_0$

② $\delta'(Q, \Sigma) = \delta(Q, \Sigma)$

Note → NFA & DFA are equivalent if →
↳ All strings valid in NFA are also valid in DFA.
↳ All strings invalid in NFA are invalid in DFA.

Q.21] Find DFA for this NFA :→



(This is transition diagram)

Soln] Step ① Draw transition table (NFA)

δ	a	b
→ q_0	$q_0 q_1$	q_0
q_1	–	q_2
* q_2	–	–

No. of column = No. of symbols

No. of rows = No. of states

Now, $Q_0' = Q_0 = q_0$

Step ② Transition table (DFA)

(Naming)

δ'	a (one single state)	b
Q_0' →	$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
Q_1'	$\{q_0 q_1\}$	$\{q_0 q_1\}$	$\{q_0 q_2\}$
Q_2' *	$\{q_0 q_2$	$\{q_0 q_1\}$	$\{q_0\}$

Note → Final states in DFA will be all the states which have final states in NFA

// calculating states manually.

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

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

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

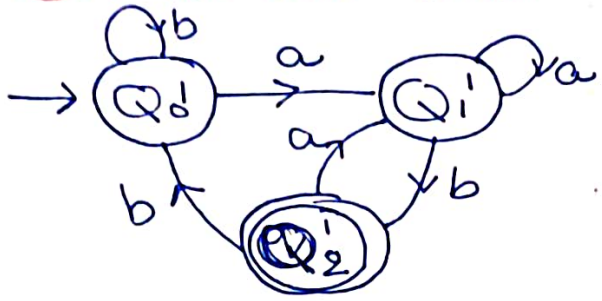
$= \delta(q_0, a) \cup \delta(q_1, a)$

$= \{q_0 q_1\} \cup \phi = \{q_0 q_1\}$

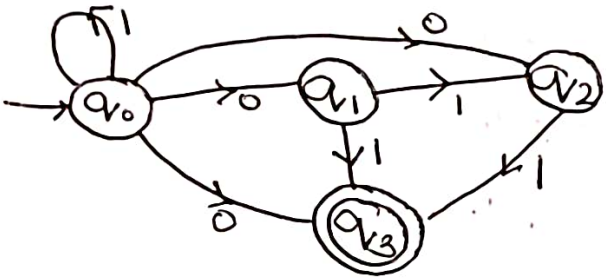
$\delta'(\{q_0 q_1\}, b) = \delta(q_0, b) \cup \delta(q_1, b)$

$= \{q_0\} \cup \{q_2\} = \{q_0 q_2\}$

Step (3): Draw the transition diagram (DFA)



HW.
Q.22]



find DFA.

Soln] Transition table ~~DFA~~ NFA, DFA

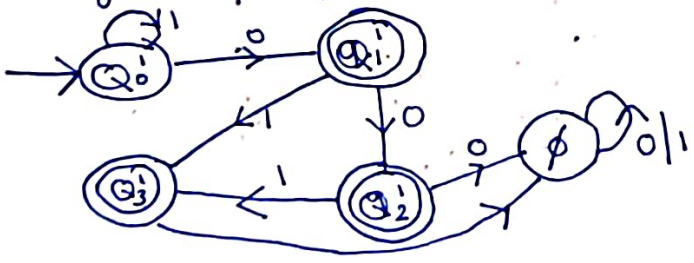
NFA

δ	0	1
$\rightarrow q_0$	$\{q_1 q_2 q_3\}$	$\{q_0\}$
q_1	$\{q_2 q_3\}$	—
q_2	—	$\{q_3\}$
$*q_3$	—	—

DFA

δ_1	0	1
$\rightarrow \{q_0\}$ Q_0'	$\{q_1 q_2 q_3\}$	$\{q_0\}$
$* \{q_1 q_2 q_3\}$ Q_1'	$\{q_2 q_3\}$	$\{q_3\}$
$* \{q_2 q_3\}$ Q_2'	—	$\{q_3\}$
$* \{q_3\}$ Q_3'	—	—

diagram ⇒



HW

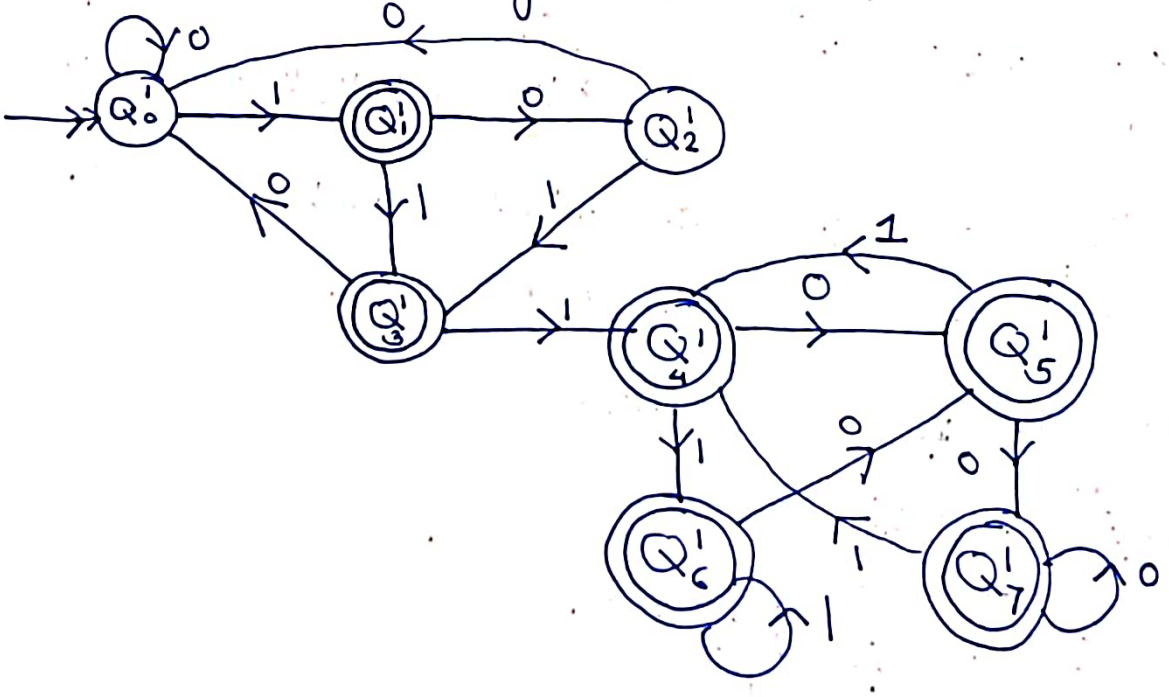
Q.2]

δ	0	1
$\rightarrow p$	p	pq
$*q$	r	r
r	—	s
$*s$	s	s

Soln]

δ_1	0	1
$Q_0' \rightarrow \{p\}$	$\{p\}$	$\{pq\}$
$Q_1' * \{pq\}$	$\{pr\}$	$\{pqr\}$
$Q_2' \{pr\}$	$\{p\}$	$\{pqs\}$
$Q_3' * \{pqr\}$	$\{p\}$	$\{pqs\}$
$Q_4' * \{pqs\}$	$\{prs\}$	$\{pqrs\}$
$Q_5' * \{prs\}$	$\{ps\}$	$\{pqs\}$
$Q_6' * \{pqrs\}$	$\{prs\}$	$\{pqrs\}$
$Q_7' * \{ps\}$	$\{ps\}$	$\{pqs\}$

→ Transition diagram:



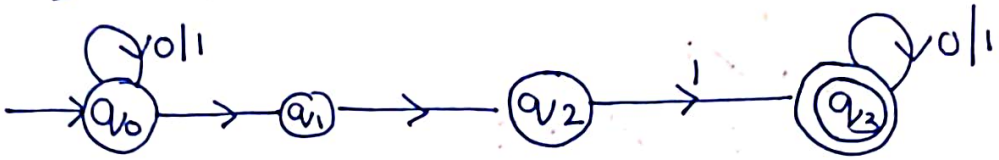
⇨ Optimization of DFA

Note:→ ① Only non-final & non-final or final - final states can merge.

② We can merge states with same transition.

Q.24] For any string containing 101 as a substring (optimized DFA)

Sol^n] Step ①: draw NFA



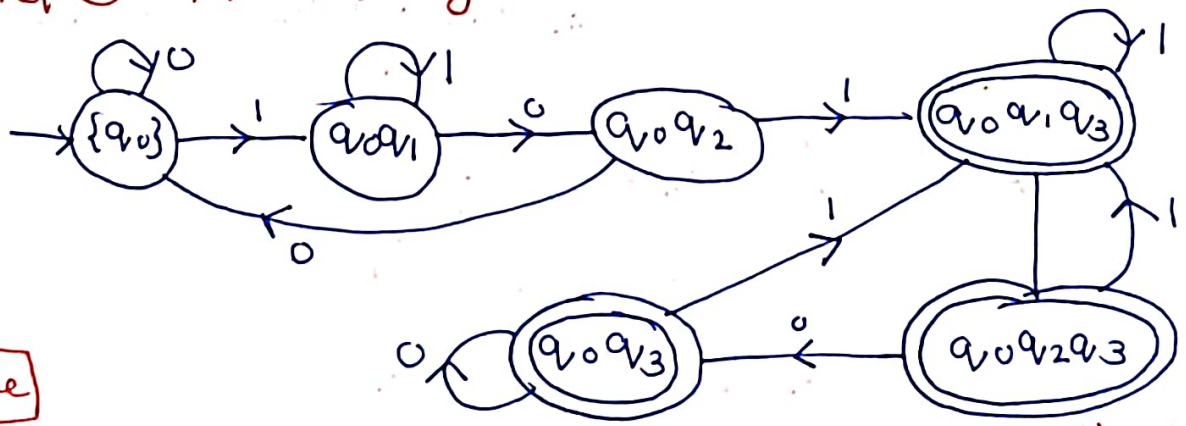
Step ②: Transition tables.

δ	0	1
→q_0	q_0	q_0q_1
q_1	q_2	—
q_2	—	q_3
*q_3	q_3	q_3

⇒

δ'	0	1
→{q_0}	{q_0}	{q_0q_1}
{q_0q_1}	{q_0q_2}	{q_0q_1}
{q_0q_2}	{q_0}	{$q_0q_1q_3$}
* {$q_0q_1q_3$}	{$q_0q_2q_3$}	{$q_0q_1q_3$}
* {$q_0q_2q_3$}	{q_0q_3}	{$q_0q_1q_3$}
* {q_0q_3}	{q_0q_3}	{$q_0q_1q_3$}

Step ③: DFA diagram:



Note

↳ We can make even more effitient DFA

Step ④ : Give names to the transition states.

S1	0	1
→ Q_0	Q_0'	Q_1'
Q_1'	Q_2'	Q_1'
Q_2'	Q_0'	Q_{345}'
* Q_{345}'	Q_{345}'	Q_{345}'
Q_{45}'	Q_{45}'	Q_3'
Q_5'	Q_{45}'	Q_3'

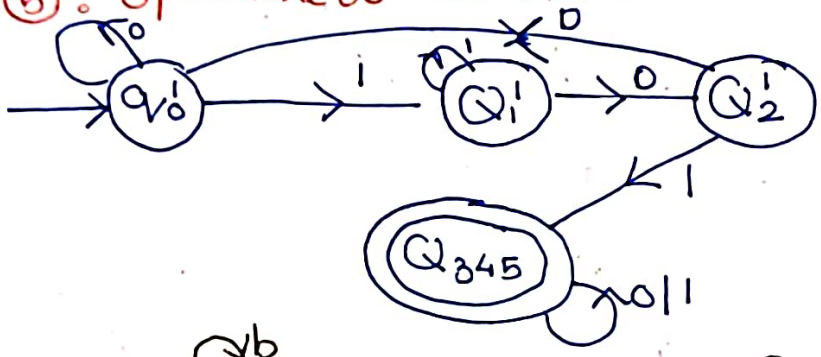
Repeat this till no merge possible

We can merge these two states

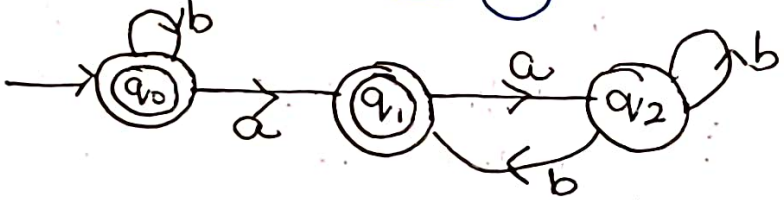
⇓

all Q_{45}', Q_5' become Q_{45}'

Step ⑤ : optimized DFA :→



Q.25]



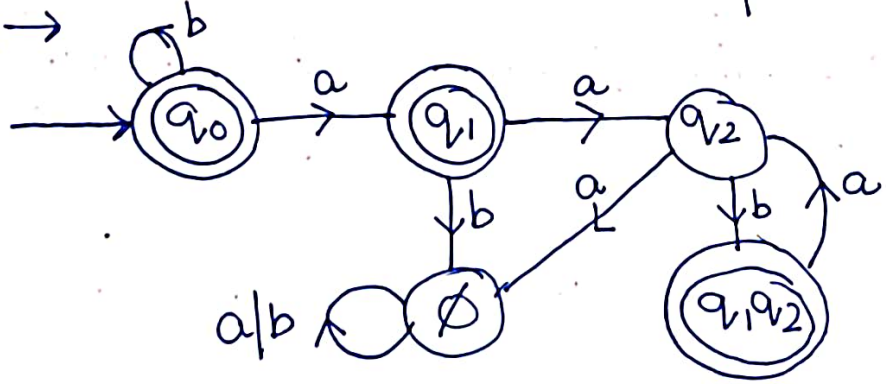
Soln]

S	a	b
→* q_0	q_1	q_0
* q_1	q_2	—
q_2	—	q_1q_2

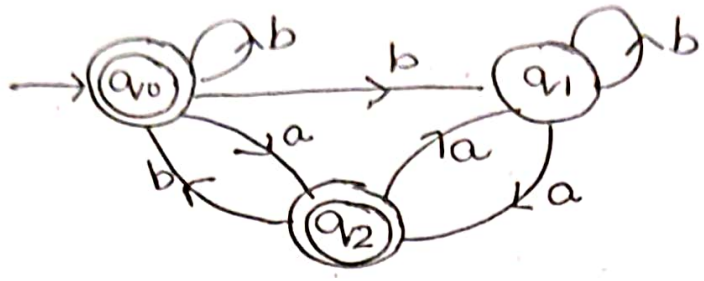
⇒

S'	a	b
→*{q_0}	{q_1}	{q_0}
*{q_1}	{q_2}	—
{q_2}	—	{q_1q_2}
*{q_1q_2}	{q_2}	{q_1q_2}

DFA :→



Q.26]



Soln]

δ	a	b
→* q_0	q_2	q_0q_1
q_1	q_2	q_1
* q_2	q_1	q_2

⇒

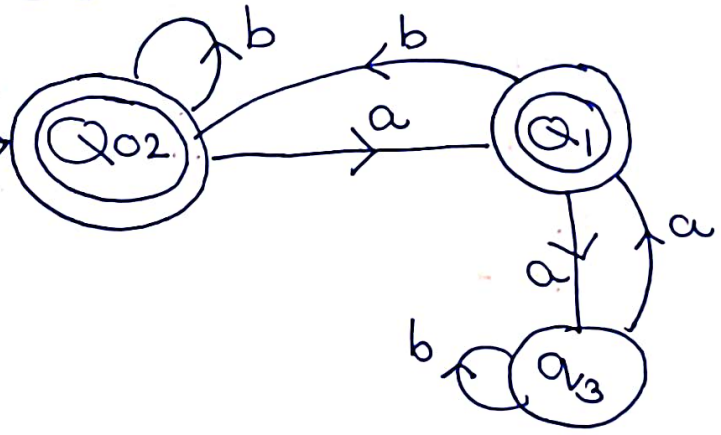
δ'	a	b
→* {q_0}	{q_2}	{q_0q_1}
* {q_2}	{q_1}	{q_2}
* {q_0q_1}	{q_2}	{q_0q_1}
{q_1}	{q_2}	{q_1}

Changing Names.

δ'	a	b
* Q_{02}	Q_1	Q_{02}
* Q_1	Q_3	Q_{02}
* Q_2	Q_1	Q_2
Q_3	Q_1	Q_3

→ (only 1 state is deleted).

DFA ⇒



Q.27]

δ	0	1
→ p	pq	p
q	r	r
r	s	–
* s	s	s

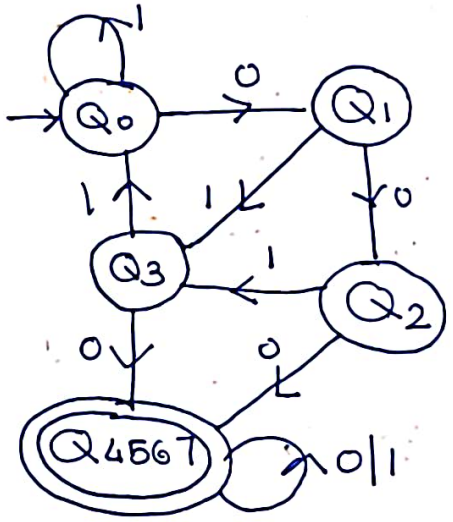
δ'	0	1
{p}	{pq}	{p}
⇒ {pq}	{pqr}	{pr}
{pr}	{pqs}	{p}
* {pqrs}	{pqrs}	{prs}
* {pqs}	{pqrs}	{prs}
* {prs}	{pqs}	{ps}
* {ps}	{pqs}	{ps}

changing Names

δ'	0	1
→ Q_0	Q_1	Q_0
Q_1	Q_2	Q_3
Q_2	Q_{4567}	Q_3
Q_3	Q_{4567}	Q_0
* Q_{4567}	Q_{4567}	Q_{67}
* ~~Q_5~~	~~Q_4~~	~~Q_{67}~~
* ~~Q_{67}~~	~~Q_{45}~~	~~Q_{67}~~
* ~~Q_7~~	~~Q_5~~	~~Q_7~~

Q_{45}

⇒



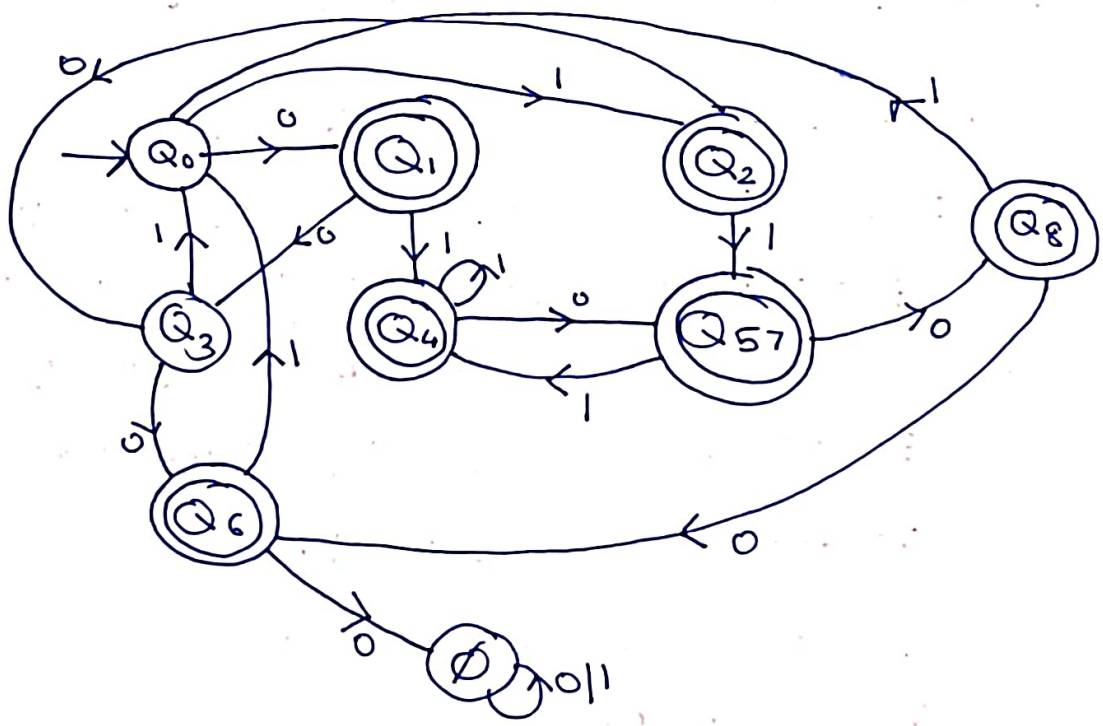
Q_{45}	Q_{45}	Q_{67}
~~Q_5~~	~~Q_4~~	~~Q_{67}~~
Q_{67}	Q_{45}	Q_{67}
~~Q_7~~	~~Q_5~~	~~Q_7~~

Q.28]

δ	0	1
$\to p$	qs	q
$*q$	r	qr
r	s	p
s	–	p

Soln]

δ'	0	1
$\{p\}_0$	$\{qs\}_1$	$\{q\}_2$
$\{qs\}_1$	$\{r\}_{38}$	$\{qrp\}_4$
$\{p\}_2$	$\{r\}_{38}$	$\{qr\}_{57}$
$\{rs\}_3$	$\{s\}_6$	$\{p\}_0$
$\{qrp\}_4$	$\{qsr\}_{57}$	$\{qrp\}_4$
$\{qsr\}_5$	$\{sr\}_{38}$	$\{qrp\}_4$
$\{s\}_6$	–	$\{p\}_0$
~~$\{qsr\}_7$~~	~~$\{rs\}_8$~~	~~$\{qrp\}_4$~~
~~$\{rs\}_8$~~	~~$\{s\}_6$~~	~~$\{p\}_0$~~



(2) NFA with ϵ to DFA

METHOD 1

ϵ - closure $(q_0) = \{$

Q_0' = ϵ - closure (Q_0)

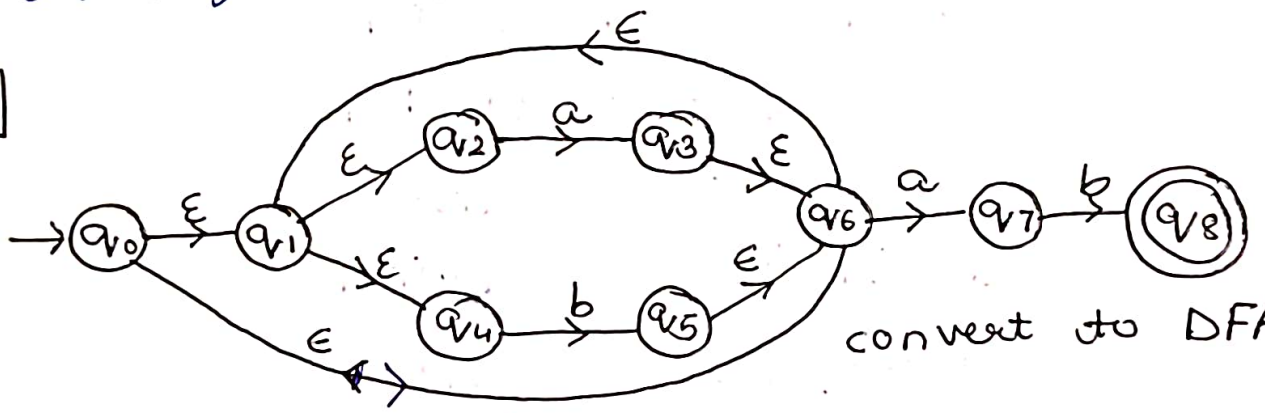
Q_0' → Initial state of DFA

Q_0 → Initial state of NFA

▷▷ Transition :→

$$\delta'(Q', \Sigma) = \epsilon\text{-closure}(\delta(Q', \Sigma))$$

Q.29]



convert to DFA.

Soln] $Q_0' = \epsilon$ - closure $(q_0) = \{q_0, q_1, q_2, q_4, q_6\}$

$\delta'(Q_0', a) = \epsilon$ - closure $(\delta(Q_0', a))$

$= \epsilon$ - closure $(\delta(q_0, q_1, q_2, q_4, q_6), a)$

$\Rightarrow \epsilon$ - closure $(\delta(q_0, a) \cup \delta(q_1, a) \cup \delta(q_2, a) \cup (q_4, a), \cup \delta(q_6, a))$

$\Rightarrow \epsilon$ - closure $(\phi \cup \phi \cup q_3 \cup \phi \cup q_7)$

$\Rightarrow \epsilon$ - closure (q_3, q_7)

$\Rightarrow \epsilon$ - closure $(q_3) \cup \epsilon$ - closure (q_7)

$\Rightarrow \{q_3, q_6, q_1, q_2, q_4\} \cup \{q_7\}$

$\therefore$ $\delta'(Q_0', a) \Rightarrow \{q_1, q_2, q_3, q_4, q_6, q_7\} \rightarrow Q_1'$

Now,

$\delta'(Q_0', b) = \epsilon\text{-closure}(\delta(Q_0', b))$

$= \epsilon\text{-closure}(\delta(q_0, q_1, q_2, q_4, q_6), b)$

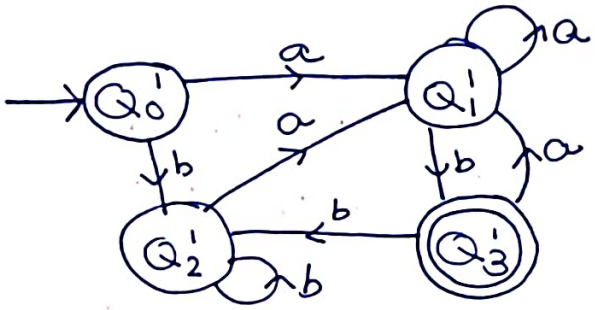
$\Rightarrow \epsilon\text{-closure}(\delta(q_0, b) \cup \delta(q_1, b) \cup \delta(q_2 b) \cup \delta(q_4, b) \cup \delta(q_6, b))$

$\Rightarrow \epsilon\text{-closure}(\phi \cup \phi \cup \phi \cup q_5 \cup \phi)$

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

$\delta'(Q_0', b) \Rightarrow \{q_5, q_6, q_1, q_2, q_4\} \Rightarrow Q_2'$

DFA



→ Ans.

$\delta'(Q_1', a) \Rightarrow \epsilon\text{-closure}(\delta(Q_1', a))$

$\Rightarrow \epsilon\text{-closure}(\delta(q_1, q_2, q_3\ q_4\ q_5\ q_7), a)$

$\Rightarrow \epsilon\text{-closure}(\delta(q_1, a) \cup \delta(q_2, a) \cup \delta(q_3, a) \cup \delta(q_4, a) \cup \delta(q_6, a) \cup \delta(q_7, a))$

$\Rightarrow \epsilon\text{-closure}(\phi \cup q_3 \cup \phi \cup \phi \cup q_7 \cup \phi)$

$\Rightarrow \epsilon\text{-closure}(q_3, q_7)$

$\Rightarrow \epsilon\text{-closure}(q_3) \cup \epsilon\text{-closure}(q_7)$

$\Rightarrow \{q_3, q_6, q_1, q_2, q_4, q_7\} \cup \{q_7\}$

$\Rightarrow Q_1'$

$\delta'(Q_1', b) = \epsilon\text{-closure}(\delta(q_1, b) \cup \delta(q_2, b) \cup \delta(q_3, b) \cup \delta(q_4, b) \cup \delta(q_6, b) \cup \delta(q_7, b)$

$\Rightarrow \epsilon\text{-closure}(\phi \cup \phi \cup \phi \cup q_5 \cup \phi \cup q_8)$

$\Rightarrow \epsilon\text{-closure}(q_5, q_8)$

$\Rightarrow \epsilon\text{-closure}(q_5) \cup \epsilon\text{-closure}(q_8)$

$\Rightarrow (q_5, q_6, q_1, q_2, q_4) \cup (q_8)$

$\delta'(Q_1', b) \Rightarrow (q_5, q_6, q_1, q_2, q_4, q_8) \Rightarrow Q_3'$.

$\delta'(Q_2', a) = \epsilon\text{-closure}(\phi \cup q_7, \phi, q_3, \phi)$

$\Rightarrow \epsilon\text{-closure}(q_7, q_3)$

$\Rightarrow \epsilon\text{-closure}(q_7) \cup \epsilon\text{-closure}(q_3)$

$\delta'(Q_2', a) = Q_1'$

$\delta'(Q_2', b) = \epsilon\text{-closure}(\phi \cup \phi \cup q_5 \cup \phi \cup \phi)$

$\delta'(Q_2', b) = Q_2'$

$\delta'(Q_3', a) = \epsilon\text{-closure}(\phi \cup q_7 \cup \phi \cup q_3 \cup \phi \cup \phi)$

$\delta'(Q_3', a) = Q_1'$

$\delta'(Q_3', b) = \epsilon\text{-closure}(\phi \cup \phi \cup \phi \cup \phi \cup q_5 \cup \phi)$

$\delta'(Q_3', b) = Q_2'$

Q 30]

Soln] $Q_0' = \epsilon$ - closure (q_0)

$Q_0' = \{q_0, q_1, q_2\}$

$\delta'(Q_0', 0) = \epsilon$ - closure $(\delta(Q_0', 0))$

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

$= \epsilon$ - closure $(q_0 \cup \phi \cup \phi)$

$= \epsilon$ - closure (q_0)

$\boxed{\delta'(Q_0', 0) = Q_0'}$

$\delta'(Q_0', 1) = \epsilon$ - clos $(\phi \cup q_1 \cup \phi)$

$= \epsilon$ - closure (q_1)

$\boxed{\delta'(Q_0', 1) = \{q_1, q_2\}} \Rightarrow Q_1'$

$\delta'(Q_0', 2) = \epsilon$ - closure $(\phi \cup \phi \cup q_2)$

$= \epsilon$ - closure (q_2)

$\boxed{\delta'(Q_0', 2) = \{q_2\}} \Rightarrow Q_2'$

$\delta'(Q_1', 0) = \epsilon$ - closure $(Q \cup \phi)$

$= \epsilon$ - closure (ϕ)

$\boxed{\delta'(Q_1', 0) = \phi}$

$\delta'(Q_1', 1) = \epsilon$ - closure $(q_2 \cup \phi)$

$= \epsilon$ - closure (ϕ) (q_2)

$\boxed{\delta'(Q_1', 1) = Q_1'}$

$\delta'(Q_1', 2) = \epsilon\text{-closure}(\phi \cup q_2)$

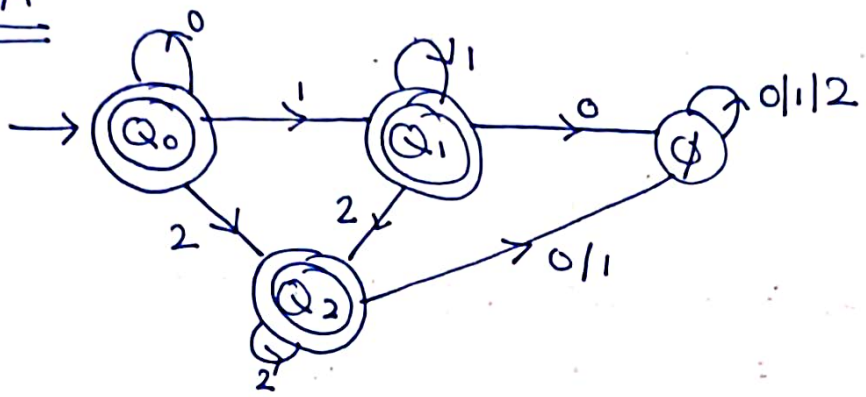
$\delta'(Q_1', 2) = Q_2'$

$\delta'(Q_2', 0) = \phi$

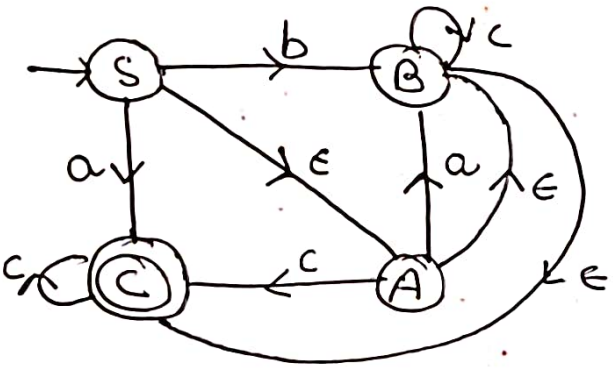
$\delta(Q_2', 1) = \phi$

$\delta(Q_2', 2) = Q_2'$

∴ DFA



Q.31]



Sol^n] $Q_0' = \epsilon\text{-closure}(S)$

$\phi_0' = \{S, A, B, C\}$

$\delta'(Q_0', a) = \epsilon\text{-closure}(\delta(S,a) \cup \delta(A,a) \cup \delta(B,a) \cup \delta(C,a))$

$\Rightarrow \epsilon\text{-closure}(C \cup B \cup \phi \cup \phi)$

$\Rightarrow \epsilon\text{-closure}(B) \cup \epsilon\text{-closure}(C)$

$\Rightarrow \{B, C\} \cup \{C\}$

$\delta'(Q_0', a) = \{B, C\} = Q_1'$

$\delta'(Q_0', b) = \epsilon\text{-closure}(B \cup \phi \cup \phi \cup \phi)$

$\delta'(Q_0', b) = \{B, C\} = Q_1'$

$\delta'(Q_0', c) = \epsilon\text{-closure}(\phi \cup C \cup B \cup C)$

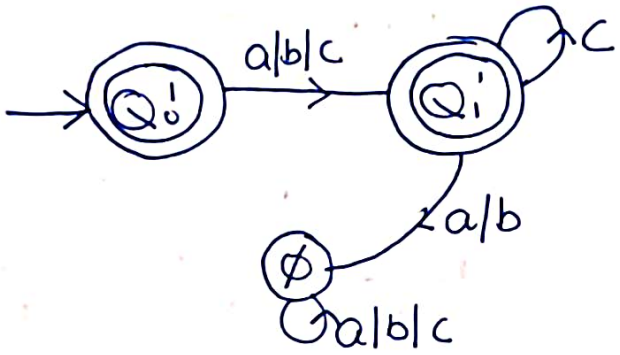
$\delta'(Q_0', c) = \phi = Q_1'$

$\delta'(Q_1', a) = \phi$

$\delta'(Q_1', b) = \phi$

$\delta'(Q_2', c) = \phi \; Q_1'$

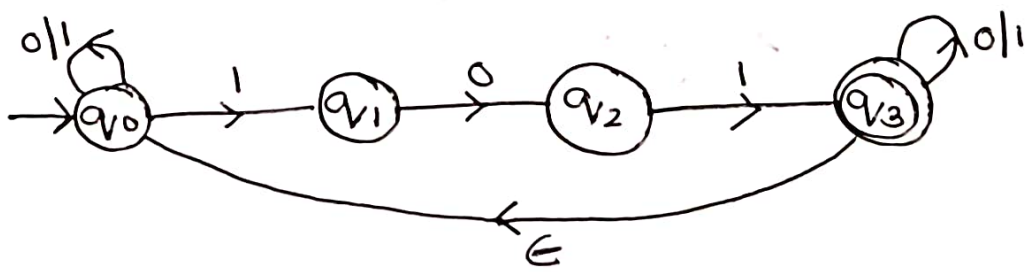
∴ DFA



Q.32]

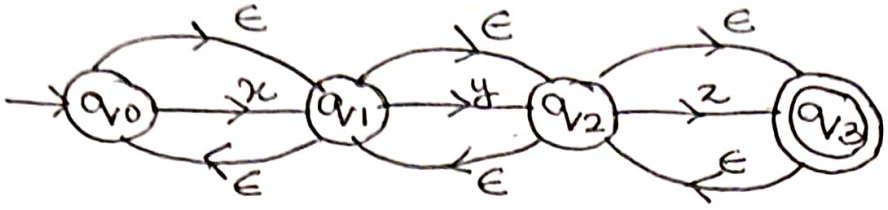
δ	ε	0	1	2
→ q_0	q_1	q_0	q_1	q_1
q_1	q_2	ϕ	q_1	q_2
* q_2	ϕ	ϕ	ϕ	q_2

Q.33]



METHOD 2 → convert NFA (with ε) to DFA using state table method.

Q.34]



Soln] **Step 1:** Make the NFA transition table

δ	x	y	z	ε
→ q_0	q_1	—	—	q_1
q_1	—	q_2	—	$q_0 q_2$
q_2	—	—	q_3	$q_1 q_3$
* q_3	—	—	—	q_2

[where can we travel using only ε]

Step 2: Make the ε-closure table.

	ε - closure
q_0	$\{q_0 q_1 q_2 q_3\}$
q_1	$\{q_0 q_1 q_2 q_3\}$
q_2	$\{q_0 q_1 q_2 q_3\}$
q_3	$\{q_0 q_1 q_2 q_3\}$

→ Take that, apply in the NFA, then the output is again put in ε-clos

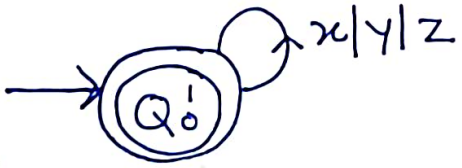
(New transition only through DFA)

Note → The 1st state of DFA will be the 1st transition of ε-closure.

Step 3: Make DFA table

δ'	x	y	z
→ * $\{q_0 q_1 q_2 q_3\}$	$\{q_0 q_1 q_2 q_3\}$	$\{q_0 q_1 q_2 q_3\}$	$\{q_0 q_1 q_2 q_3\}$

Step 4: draw NFA



Q.35]

Soln]

δ	0	1	ϵ
→ q_0	q_1	q_1	—
* q_1	q_0q_2	q_1	q_2
q_2	—	q_1	—

⇒

	ϵ - closure
q_0	$\{q_0\}$
q_1	$\{q_1, q_2\}$
q_2	$\{q_2\}$

Tip → change names of states in DFA table

Now, DFA table

δ'	0	1
→ $\{q_0\}$	$\{q_1q_2\}$	$\{q_1q_2\}$
* $\{q_1q_2\}$	$\{q_0q_2\}$	$\{q_1q_2\}$
$\{q_0q_2\}$	$\{q_1q_2\}$	$\{q_1q_2\}$

δ'	0	1
→ Q'_2	Q'_1	Q'_1
⇒ * Q'_1	Q'_2	Q'_1
~~Q'_{20}~~	~~Q'_1~~	~~Q'_1~~

⇒

Q.36]

Soln]

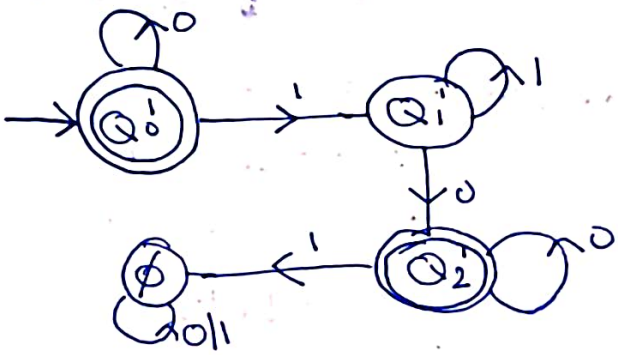
δ	0	1	ϵ
→ q_0	q_0	q_1	q_2
q_1	q_2	q_1	—
* q_2	q_2	—	—

⇒

	ϵ - closure
q_0	$\{q_0q_2\}$
q_1	$\{q_1\}$
q_2	$\{q_2\}$

δ'	0	1
Q'_0 → $\{q_0q_2\}$	Q'_2 $\{q_0q_2\}$	$\{q_1\}_{Q'_1}$
* $\{q_0q_2\}$		
Q'_1 $\{q_1\}$	$\{q_2\}_{Q'_2}$	$\{q_1\}_{Q'_1}$
* $\{q_2\}$	$\{q_2\}_{Q'_2}$	$\emptyset$
Q'_2		

⇒



* Method to convert NFA with ∈ into NFA without ∈

$M = (Q, \Sigma, \delta, Q_0, F) \rightarrow$ NFA with ∈

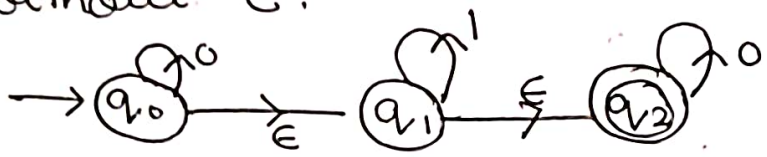
$M_1 = \{Q, \Sigma, \delta_1, Q_0, F_1\} \rightarrow$ NFA without ∈.

* $F_1 = F \cup Q_0$ if ∈- closure of Q_0 contains final state.

$F_1 = F$ (otherwise)

* $\delta_1(Q, \Sigma) = \in$- closure $(\delta(\in$-closure $(Q), \Sigma))$

Q.37] convert NFA with ∈ into NFA without ∈.



Sol^n] $M = (\{q_0 q_1 q_2\}, \{0,1\}, \delta, q_0, \{q_2\})$

$M_1 = (\{q_0 q_1 q_2\}, \{0,1\}, \delta_1, q_0, \{q_0, q_2\})$

Now, ∈ - closure $(q_0) = \{q_0, q_1, q_2\}$

$\Rightarrow \delta_1(q_0, 0) = \in$ closure $(\delta(\in$-closure $(q_0), 0))$

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

$= \in$ - closure $(q_0 q_2)$

$= \{q_0 q_1 q_2\}$

$\Rightarrow \delta_1(q_0, 1) = \in$- closure $(\delta(\in$-closure $(q_0), 1))$

$= \in$ - closure $(\delta(q_0 q_1 q_2), 1))$

$= \in$ - closure (q_1)

$= \{q_1 q_2\}$

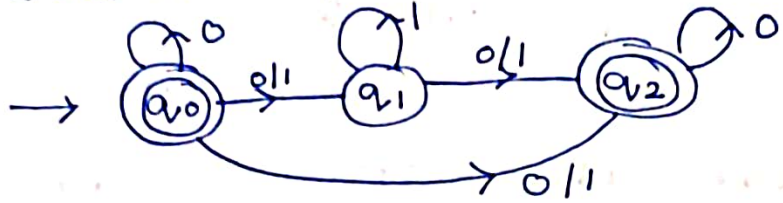
$\delta_1(q_1, 0) = \epsilon\text{-closure}(\delta(q_2), 0))$

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

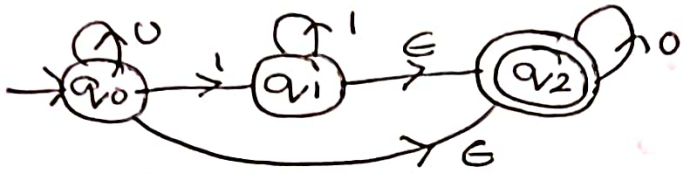
$\delta_1(q_1, 1) = \{q_1, q_2\}$

$\delta_1(q_2, 0) = \{q_2\}$

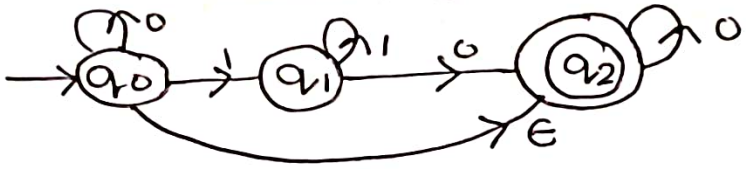
$\delta_1(q_2, 1) = \phi$



Q. 38]



Q. 39]



Optimization of Finite Automata

Marking Algorithm (Method 1)

- **Optimized FA :→** FA with min no. of states and min. transitions.

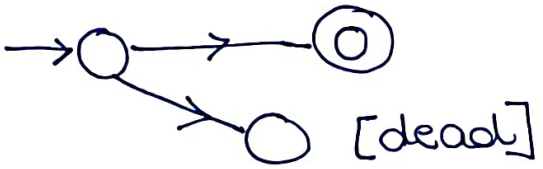
  Why? →
  1. With every state → Space increases
  2. With every transitn → time increases

- **Steps of ~~Marking algo~~ optimizing FA :**

Step 1 : Removal of dead states
Step 2 : Removal of unreachable states
Step 3 : Merging of similar states.

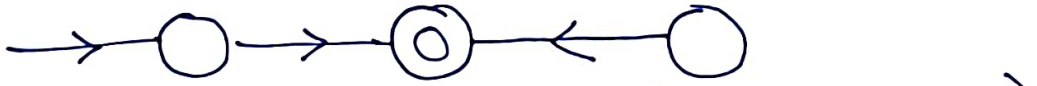
▷ Dead States

↳ All trap states
↳ No path to final states
↳



▷ Unreachable states

↳ No path from initial state
↳



- **Marking Algo** → Used to find similar states in FA.
  ↳ 3 step process.

⊙ Steps of Marking Algorithm:

Step 1: To get feasible state pairs:

Conditions:
- → Both final or both non final.
- → Similar transition.
- → $\delta(P, \Sigma) = \phi$ || $\delta(P, \Sigma) \neq \phi$
  $\delta(q, \epsilon) = \phi$ || $\delta(q, \epsilon) \neq \phi$

Step 2: Get feasible state pair table

Step 3: Marking of feasible state pairs.

Q. How to do Marking?

Ans) Rows of feasible state pair table will be marked if entry for any input character →

① ↳ Is not among feasible state pair (It should be pair i.e., {BB} is not allowed)

② ↳ It is already marked.

Q.40]

δ	a	b	c
A	B	E	F
B	C	–	H
C	A	–	H
D	F	–	E
* E	E	F	G
F	D	–	E
G	B	H	D
* H	H	D	A

Soln]

Step ①: Get feasible state pair

- FF ⇒ {EH}
- NF-NF ⇒ { {AG}, {BC} {BD} {BF}, {CD} {CF}, {DF} }

Step ②: Get feasible state pair table.

2nd \ 1st	a	b	c
AG	B	EH	DF
✓ BC	AC	–	HH
✓ BD	CF	–	EH
✓ BF	CD	–	EH
✓ CD	AF	–	EH
✓ CF	AD	–	EH
DF	DF	–	EE
* EH	EH	FD	AG

Tip: In round 1 ↳ अगर नहीं है तो Mark करना।

⇒ {AG, DF, EH} are unmarked; this means: → A ≅ G, D ≅ F, E ≅ H

Step ③: Final table after merging similar states (optimized final automata)

δ	a	b	c
→ AG	B	EH	DF
B	C	–	EH
C	AG	–	EH
DF	DF	–	EH
* EH	EH	DF	AG

Q.41]

S	0	1	a
A	B	C	—
B	F	D	C
C	E	D	B
D	F	—	—
E	—	D	—
* F	—	—	—

Soln]

Step① ⇒ {BC}

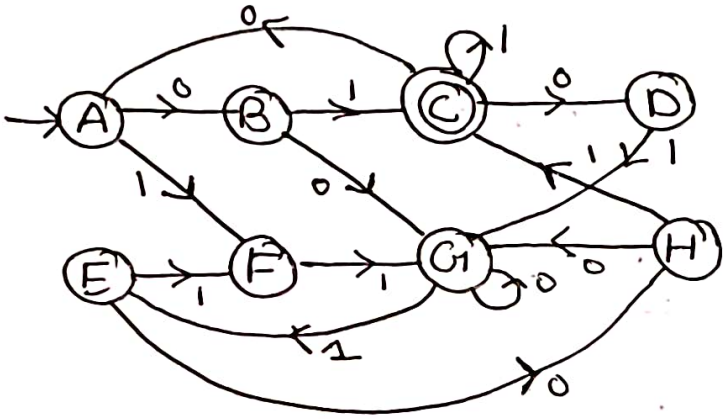
Step②:

	0	1	a
✓ BC	EF	DD	BC

Step③: [NO merging] → [No uncommon]

Conclusion: This FA is already optimized.

Q.42]



Soln]

	S	0	1
→	A	B	F
	B	G	C
*	C	AD	C
	D	—	G
	E	H	F
	F	—	G
	G	G	E
	H	G	C

① Pairs →

{ AB, AE, AG, AH,
BE, BG, BH,
DF,
EG, EH
GH }

→ Desirable state pair table:

	0	1
→ AB	BG	CF
AE	BH	FF
✓ AG	BG	EF
✓ AH	BG	CF
✓ BE	GH	CF
✓ BG	GG	CE
BH	GG	CC
DF	—	GG
✓ EG	GH	EF
✓ EH	GH	CF
✓ GH	GG	CE

AE, BH, DF are unmarked

⇓

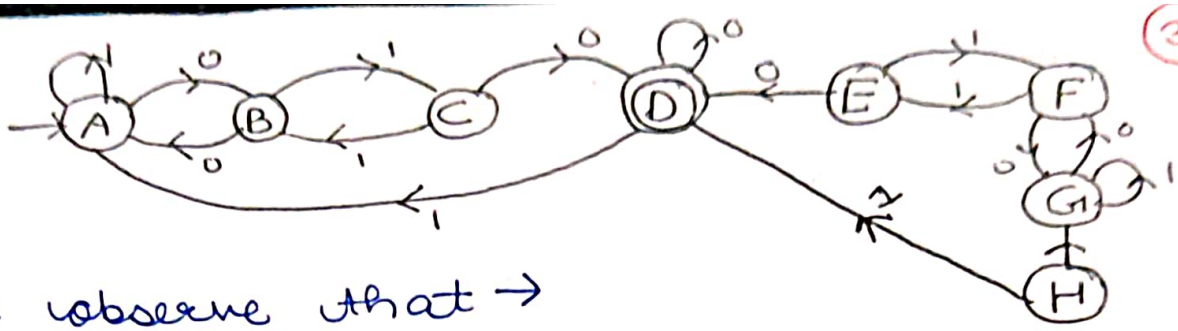
∴ [A ≅ E, B ≅ H, D ≅ F]

∴ optimized F. automata:

δ	0	1
AE	BH	F
BH	G	C
* C	AD	C
DF	—	G
G	G	AE

Note: In Step (1) remove dead and unreachable states first; then apply Marking Algo.

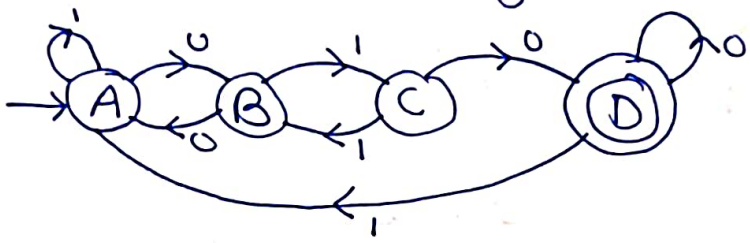
Q.43]



Sol?] We observe that →

E, F, G, H are unreachable states. Hence, we can remove those directly.

∴ we can directly solve for:



◁▷ Partitioning Algorithm

Q.44]

	δ	0	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

Sol?] Initially → All states were in 1 group.

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

Step① Divide the groups on the basis of final & non-final.

$G_1 = \{q_0, q_1, q_3, q_4, q_5, q_6, q_7\}$; $G_2 = \{q_2\}$

Step② Divide further on the basis of transitions.

① G_1

q_0q_1	0	1	→ Group
q_0	q_1 (I)	q_5 (I)	
q_1	q_6 (I)	q_2 (II)	

$\{q_0 \neq q_1\}$

↳ Transition on all symbols must go in same group otherwise they were dissimilar.

(ii)

q_0q_3	0	1
q_0	q_1 (I)	q_5
q_3	q_2 (II)	q_0

$[q_0 \neq q_3]$

(iii)

q_0q_4	0	1
q_0	q_1 (I)	q_5 (I)
q_4	q_7 (I)	q_5 (I)

$[q_0 \equiv q_4]$ Similar

(iv)

q_0q_5	0	1
q_0	q_1 (I)	q_5
q_5	q_2 (II)	q_6

$[q_0 \neq q_5]$

(v)

q_0q_6	0	1
q_0	q_1	q_5
q_6	q_6	q_4

$[q_0 \equiv q_6]$

(vi)

q_0q_7	0	1
q_0	q_1	q_5
q_7	q_6	q_2 (II)

$[q_0 \neq q_7]$

$\therefore$

Next Iteration $\Rightarrow$ $\{q_0, q_4, q_6\}$ (I) $\{q_1, q_3, q_5, q_7\}$ (II) $\{q_2\}$ (III)

Step (3) Further divide all groups with more than 2 elements

q_1q_3	0	1
q_1	q_6	q_2
q_3	q_2	q_6

$[q_1 \neq q_3]$

q_1q_5	0	1
q_1	q_6 (I)	q_2
q_5	q_2 (III)	q_6

$[q_1 \neq q_5]$

q_1q_7	0	1
q_1	q_6	q_2
q_7	q_6	q_2

$[q_1 \equiv q_7]$

$\Rightarrow \{q_0\ q_6\ q_4\}\ \{q_1\ q_7\}\ \{q_3\ q_5\}\ \{q_2\}$
(I) (II) (III) (IV)

q_0q_4	0	1
q_0	q_1	q_5
q_4	q_7	q_5

$[q_0 \equiv q_4]$

q_0q_6	0	1
q_0	q_1	q_5
q_6	q_6	q_4

$[q_0 \not\equiv q_6]$

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

↳ No further division possible.

$\therefore [q_6 \equiv q_4]\ [q_1 \equiv q_7]\ [q_3 \equiv q_5]$

∴ optimite table of finite automata:

δ	0	1
$\rightarrow q_{04}$	q_{17}	q_{35}
q_{17}	q_6	q_2
$* q_2$	q_{04}	q_2
q_{35}	q_2	q_6
q_6	q_6	q_{04}

Q.45]

δ	a	b
→ q_0	q_1	q_0
q_1	q_0	q_2
q_2	q_3	q_1
* q_3	q_3	q_0
q_4	q_3	q_5
q_5	q_6	q_4
q_6	q_5	q_6
q_7	q_6	q_3

Soln]

Step ①:

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

$\Downarrow$

$\{q_0 q_1 q_2 q_4 q_5 q_6 q_7\}$ (I) $\{q_3\}$ (II)

Step ② Transitions from q_0.

q_0q_1	a	b
q_0	q_1	q_0
q_1	q_0	q_2

$[q_0 \equiv q_1]$

q_0q_2	
q_0	q_1 (I) q_0
q_2	q_3 (II) q_1

$[q_0 \neq q_2]$

q_0q_4	a	b
q_0	q_1	q_0
q_4	q_3	q_5

$[q_0 \neq q_4]$

q_0q_5	a	b
q_0	q_1	q_0
q_5	q_6	q_4

$[q_0 \equiv q_5]$

q_0q_6	a	b
q_0	q_1	q_0
q_6	q_5	q_6

$[q_0 \equiv q_6]$

q_0q_7	a	b
q_0	q_1	q_0
q_7	q_6	q_3

$[q_0 \neq q_7]$

$\Rightarrow \{q_0, q_1, q_5, q_6\}$ (I) $\{q_2\ q_4\ q_7\}$ (II) $\{q_3\}$ (III)

q_1q_0	a	b
q_1	q_0	q_2 (II)
q_0	q_1	q_0 (I)

$[q_1 \neq q_0]$

q_1q_5	a	b
q_1	q_0	q_2
q_5	q_6	q_4

$[q_1 \equiv q_5]$

q_1q_6	a	b
q_1	q_0	q_2
q_6	q_5	q_6

$[q_1 \neq q_6]$

$\Rightarrow \{q_1\ q_5\}\ \{q_0 q_6\}\ \{q_2 q_4\ q_7\}\ \{q_3\}$
(I) (II) (III) (IV)

q_2q_4	a	b
q_2	q_3	q_1
q_4	q_3	q_5

$[q_2 \equiv q_4]$

q_2q_7	a	b
q_2	q_3	q_1
q_7	q_6	q_3

$[q_2 \neq q_7]$

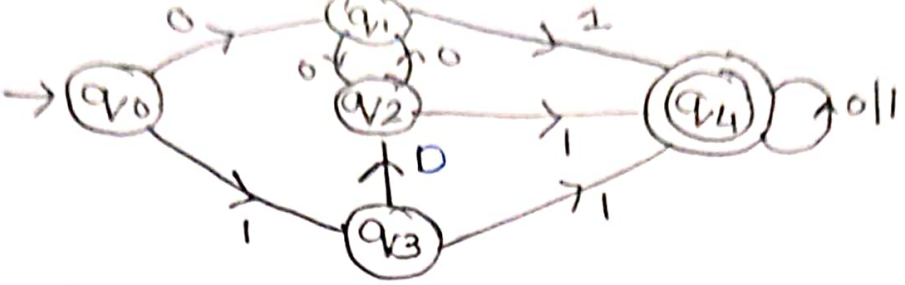
$\Rightarrow \{q_1q_5\}\ \{q_0q_6\}\ \{q_2q_4\}\ \{q_7\}\ \{q_3\}$

$\therefore [q_1 \equiv q_5]\ [q_0 \equiv q_6]\ [q_2 \equiv q_4]$

↳ optimised finite automata :→

S'	a	b
$\rightarrow q_{06}$	q_{15}	q_{06}
q_{15}	q_{06}	q_{24}
q_{24}	q_3	q_{15}
* q_3	q_3	q_{06}
q_7	q_{06}	q_3

Q 46]



Soln]

δ	0	1
$\to q_0$	q_1	q_3
q_1	q_2	q_4
q_2	q_1	q_4
q_3	q_2	q_4
$*q_4$	q_4	q_4

$\Rightarrow \{q_0\ q_1\ q_2\ q_3\}\ \{q_4\}$

→ checking transitions

q_0q_1	0	1
q_0	q_1	q_3
q_1	q_2	q_4

$[q_0 \neq q_1]$

q_0q_2	0	1
q_0	q_1	q_3
q_2	q_1	q_4

$[q_0 \neq q_2]$

q_0q_3	0	1
q_0	1	3
q_3	2	4

$[q_0 \neq q_3]$

$\Rightarrow \{q_0\}^{(I)}\ \{q_1\ q_2\ q_3\}^{(II)}\ \{q_4\}^{(III)}$

q_1q_2	0	1
q_1	q_2	q_4
q_2	q_1	q_4

$[q_1 \equiv q_2]$

q_1q_3	0	1
q_1	q_2	q_4
q_3	q_2	q_4

$[q_1 \equiv q_3]$

$\Rightarrow \{q_0\}\ \{q_1\ q_2\ q_3\}\ \{q_4\}$

↓

No more possible transitions

$[q_1 \equiv q_2 \equiv q_3]$

∴ Transition table →

δ'	0	1
$\to q_0$	q_{123}	q_{123}
q_{123}	q_{123}	q_4
$*q_4$	q_4	q_4

⇨ Finite Automata with output

Transducer

- Moore M/C
- Mealy M/C

* Moore M/C

$M = (Q, \Sigma, \Delta, \delta, \lambda, Q_0)$

where,

- $Q \Rightarrow$ Set of states
- $\Sigma \Rightarrow$ Set of input symbols.
- $\Delta \Rightarrow$ Set of output symbols
- $\delta \Rightarrow$ Transition function
- $\lambda \Rightarrow$ Output functions
- $Q_0 \Rightarrow$ Initial state

$$[\lambda : Q \to \Delta]$$

Note → No final state (Every state has an output)

↳ Whenever moore machine reads input of size 'n', it generates an output string of $(n+1)$ size.

↳ Every state has an output.

↳ Take care of $\in$.

Q4] Discuss this Moore Machine

Present state	Next state		output
	0	1	
$\rightarrow q_0$	q_3	q_1	0
q_1	q_1	q_2	1
q_2	q_2	q_3	0
q_3	q_3	q_0	0

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

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

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

$\delta \rightarrow$ (table)

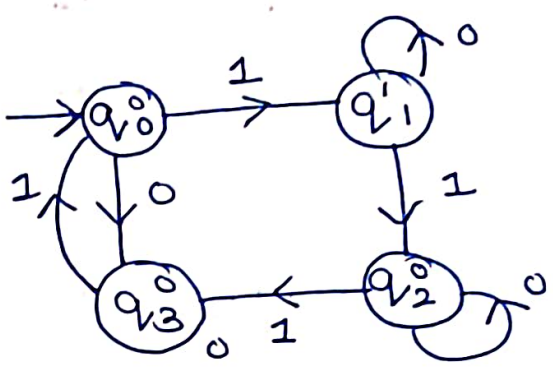
$\lambda \Rightarrow q_0 \rightarrow 0$

$q_1 \rightarrow 1$

$q_2 \rightarrow 0$

$q_3 \rightarrow 0$

$Q_0 = q_0$



Ex String I/P $\Rightarrow$ 0111

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

output 0 0 0 1 0

$\therefore$ O/P string $\Rightarrow$ '00010'

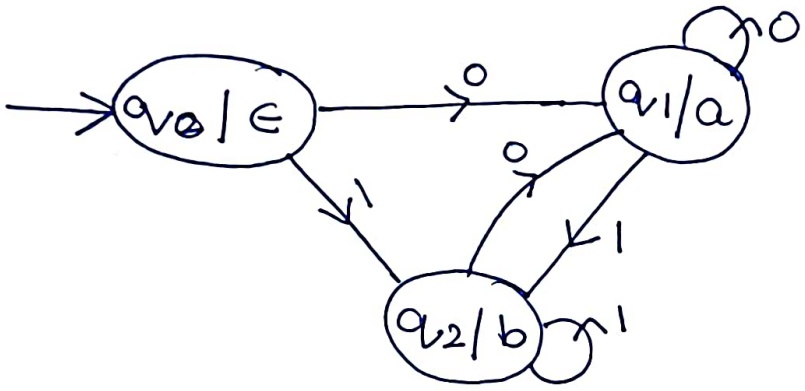
Q.48] Design a moore machine to convert strings of 0/1 to string of a & b

every $0 \rightarrow a$

every $1 \rightarrow b$

Soln]

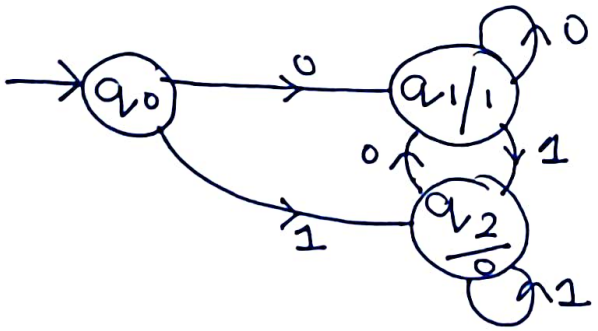
Present state	Next State		output
	0	1	
q_0	q_2	q_1	$\in$
q_1	q_0	q_2	a
q_2	q_0	q_1	b



Q.49] Find 1s complement of binary no.

Soln]

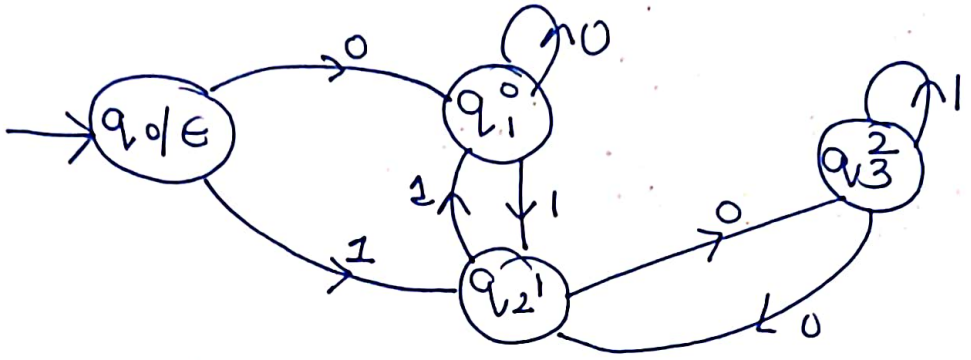
Present	Next State		O/P
	0	1	
q_0	q_1	q_2	G
q_1	q_1	q_2	1
q_2	q_1	q_2	0



Q.50] I/P is a binary no. and o/p generate remainders of 3 (mod 3)

Soln]

Present	Next state		Output
	0	1	
q_0	q_1	q_2	0
q_1	q_1	q_2	1
q_2	q_3	q_1	2
q_3	q_2	q_3	



Sample I/P ⇒ 01001

O/P ⇒ 01210.

---

* Mealy M/C

$M = (Q, \Sigma, \Delta, \delta, \lambda, Q_0)$

$[\lambda : Q \times \Sigma \to \Delta]$

→ Output function is based on current state & next input symbol.

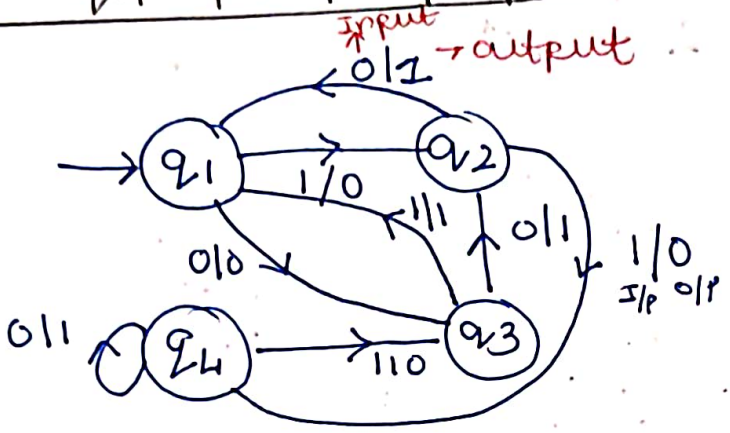
- Rest all things same.

Q. 5] Discuss this Mealy Machine.

Present state	Next state			
	0		1	
	S	O/P	S	O/P
→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

note: in Mealy M.C.
↓
no. of symbols in I/P = no. of symbols in O/P.

Soln]



Ip string => 0 0 1 1
State => $q_1 \to q_3 \to q_2 \to q_4$
O/P str => 0 1 0 0

Note → In Mealy Mc, number of symbols in input = no. of symbols in output

Q.52] Generate 1s complement using Mealy M/C.

Soln]

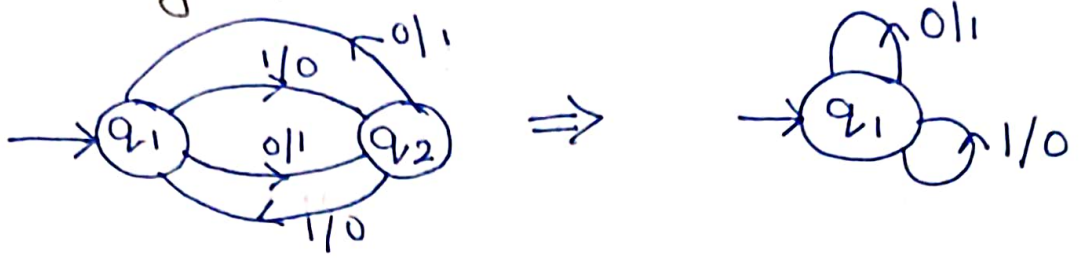
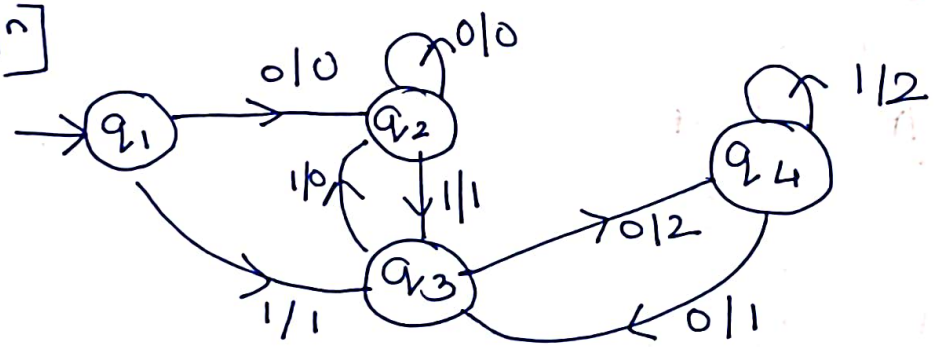


Table ⇒

PS	Next state			
	0		1	
	NS	O/P	NS	O/P
q_1	q_1	1	q_1	0

Q53] Display mod 3 for binary

Soln]



⇨ Mealy M/C to Moore M/C

Step 1] Determine no. of outputs associated with every state of Mealy M/C in next state of transition table.

Step 2] Split the states having more than one o/p into no. of states equal to no. of o/p associated with it.

Step 3] Get the transition table for new set of states

Step 4] Adjust the table for Moore M/C.

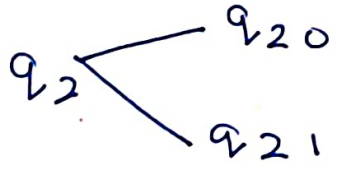
Q.53]

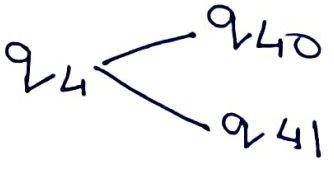
PS	Next State			
	a=0		a=1	
	S	o/p	S	o/p
→q_1	q_3	0	q_2	0
q_2	q_1	1	q_4	0
q_3	q_2	1	q_1	1
q_4	q_4	1	q_3	0

Soln] Step 1]

State	o/p	→ No. of o/p associated
q_1	1	(1)
q_2	1, 0	(2)
q_3	0	(1)
q_4	1, 0	(2)

Step 2] Split q_2 & q_4





Note → Split into 2 states as it had 2 outputs.

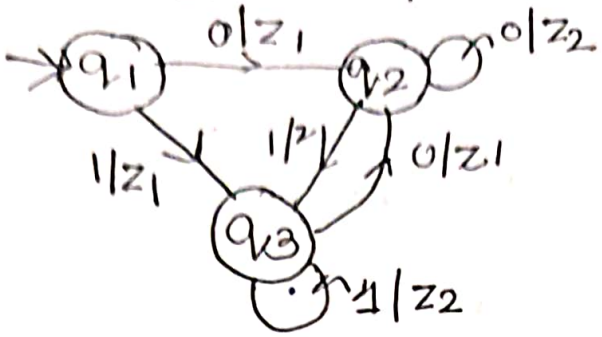
Step 3] Get new transition table for the increased states.

PS	Next State			
	a=0		a=1	
	S	O/P	S	O/P
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

Step 4] Moore table ⇩

PS	Next State		O/P
	a=0	a=1	
q_1	q_3	q_{20}	1
q_{20}	q_1	q_{40}	0
q_{21}	q_1	q_{40}	1
q_3	q_2	q_1	0
q_{40}	q_{41}	q_3	0
q_{41}	q_{41}	q_3	1

Q.54] Find Moore M/C



Sol^n]

PS	Next state			
	a=0		a=1	
	S	O/P	S	O/P
q_1	q_2	z_1	q_3	z_1
q_2	q_2	z_2	q_3	z_1
q_3	q_2	z_1	q_3	z_2

$q_1 \rightarrow \in$

$q_2 \rightarrow z_1, z_2$

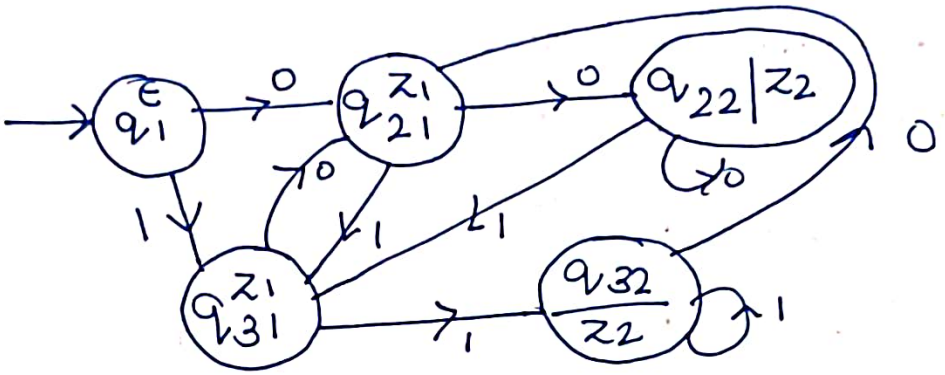
$q_3 \rightarrow z_1, z_2$

$q_2 \rightarrow q_{21}, q_{22}$

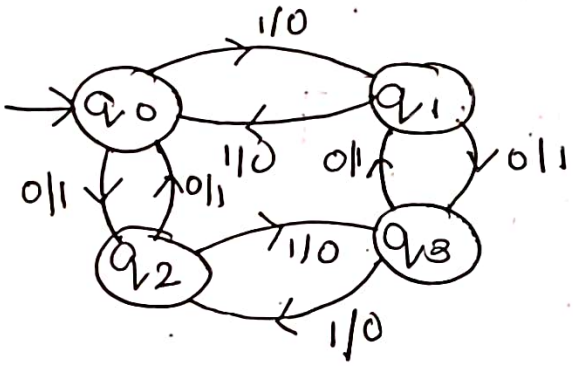
$q_3 \rightarrow q_{31}, q_{32}$

PS	Next state			
	a=0		a=1	
	S	O/P	S	O/P
q_1	q_{21}	z_1	q_{31}	z_1
q_{21}	q_{22}	z_2	q_{31}	z_1
q_{22}	q_{22}	z_2	q_{31}	z_1
q_{31}	q_{21}	z_1	q_{32}	z_2
q_{32}	q_{21}	z_1	q_{32}	z_2

PS	Next State		O/P
	a=0	a=1	
q_1	q_{21}	q_{31}	ϵ
q_{21}	q_{22}	q_{31}	z_1
q_{22}	q_{22}	q_{32}	z_2
q_{31}	q_{21}	q_{32}	z_1
q_{32}	q_{21}	q_{32}	z_2



Q 55]



Soln]

PS	Next State			
	a=0		a=1	
	S	O/P	S	O/P
q_0	q_3	1	q_1	0
q_1	q_2	1	q_0	0
q_2	q_1	1	q_3	0
q_3	q_0	1	q_2	0

$q_0 \rightarrow 0,1$: q_{00}, q_{01}

$q_1 \rightarrow 0,1$: q_{10}, q_{11}

$q_2 \rightarrow 0,1$: q_{20}, q_{21}

$q_3 \rightarrow 0,1$: q_{30}, q_{31}

PS	Next State			
	a=0		a=1	
	S	O/P	S	O/P
q_{00}	q_{31}	1	q_{10}	0
q_{01}	q_{31}	1	q_{10}	0
q_{10}	q_{21}	1	q_{00}	0
q_{11}	q_{21}	1	q_{00}	0
q_{20}	q_{11}	1	q_{30}	0
q_{21}	q_{11}	1	q_{30}	0
q_{30}	q_{01}	1	q_{20}	0
q_{31}	q_{01}	1	q_{20}	0

PS	Next State		O/P
	a=0	a=1	
q_{00}	q_{31}	q_{10}	0
q_{01}	q_{31}	q_{10}	1
q_{10}	q_{21}	q_{00}	0
q_{11}	q_{21}	q_{00}	1
q_{20}	q_{11}	q_{30}	0
q_{21}	q_{11}	q_{30}	1
q_{30}	q_{01}	q_{20}	0
q_{31}	q_{01}	q_{20}	1

⇨ Moore to Mealy M/C

Step 1] Define O/P fn of Mealy MC $\lambda'(q,a) = \lambda(\delta(q,a))$

where, $\lambda \rightarrow$ O/P fn of Moore MC. & δ is the transition fn of Moore MC.

Step 2] → All transition remain same.

Q 56]

PS	Next State		O/P
	a=0	a=1	
→q_0	q_3	q_1	0
q_1	q_1	q_2	1
q_2	q_2	q_3	0
q_3	q_3	q_0	0

⇒ $q_0 = 0$, $q_1 = 1$, $q_2 = 0$, $q_3 = 0$

Soln]

PS	Next State			
	a=0		a=1	
	S	O/P	S	O/P
q_0	q_3	0	q_1	1
q_1	q_1	1	q_2	0
q_2	q_2	0	q_3	0
q_3	q_3	0	q_0	0

Q.57] Get equivalent mealy M/C

PS	Next state		O/P
	a=0	a=1	
q_1	q_4	q_2	1
q_2	q_1	q_2	0
q_3	q_1	q_5	1
q_4	q_3	q_1	0
q_5	q_6	q_4	0
q_6	q_6	q_4	1

Soln]

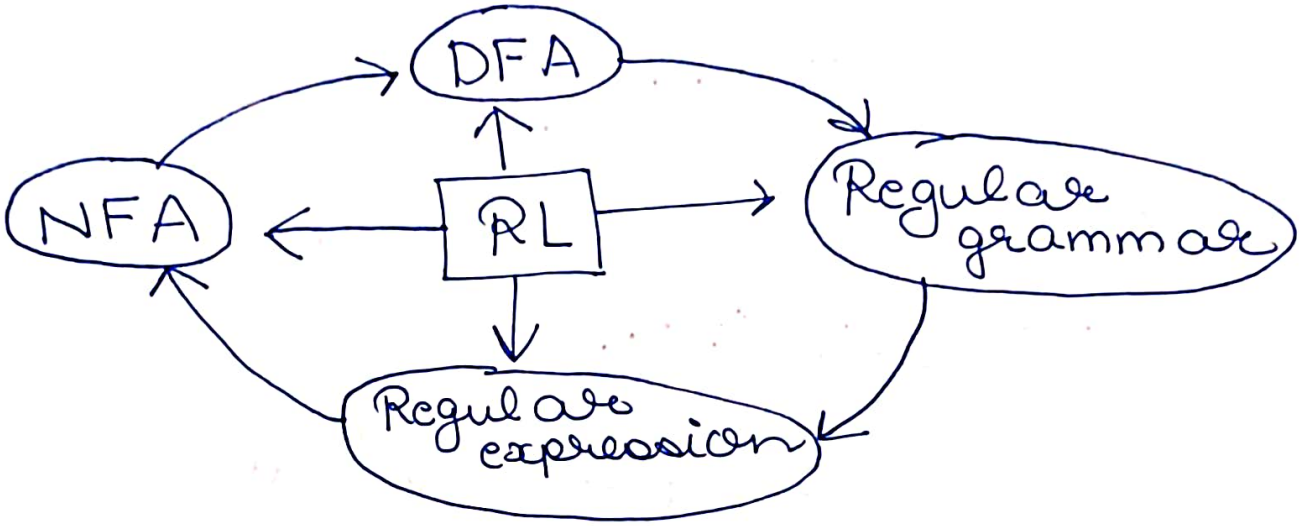
PS	Next State			
	a=0		a=1	
	S	O/P	S	O/P
q_1	q_4	0	q_2	0
q_2	q_1	1	q_2	0
q_3	q_1	1	q_5	0
q_4	q_3	1	q_1	1
q_5	q_6	1	q_4	0
q_6	q_6	1	q_4	0

$q_1 = 1$
$q_2 = 0$
$q_3 = 1$
$q_4 = 0$
$q_5 = 0$
$q_6 = 1$

UNIT-3

REGULAR EXPRESSION

▷ Representation of Regular Language



▷ Regular Expression

Operand → Σ , ∈

operater → . (concatination operater) / (union (OR)) + (Positive closure) * (clean closure)

Examples:

RE	RG
∈	→ {∈}
a	→ {a}
a.b	→ {ab}
a/b	→ {a, b}
a^+ (one or more occurances of a)	→ {a, aa, aaa.....}
a^* (0 or more occurances of a)	→ {∈, a, aa, aaa.....}

$a^* = a^+ \cup \{\in\}$

Q.1] $(a/b)^*abb$ give regular language

Soln] $(a/b)^*.a.b.b \Rightarrow \{abb, aabb, babb, ababb \ldots\}$

Q.2] Write regular expression for all string on $\Sigma = \{a, b, c\}$ with exactly 1 a.

Soln] RL = {a, ab, ac, abc, ba, ca, bca ... }

$RE = (b/c)^*.a.(b/c)^*$

Q.3] $\Sigma = \{0, 1\}$ all string ends with 01

Soln] $RE = \{(0/1)^*.0.1\}$

Q.4] $\Sigma = \{a, b, c\}$ representing all strings with no more than 3 a.

Soln] RE $\Rightarrow$ 0 as / 1 as / 2 as / 3 as $\rightarrow$ Valid

$\hookrightarrow \{(b/c)^* / (b/c)^*.a.(b/c)^* /$
$(b/c)^*.a.(b/c)^*.a.(b/c)^* /$
$(b/c)^*.a.(b/c)^*.a.(b/c)^*.a.(b/c)^*.\}$

Note:

- ↳ LHS of '/' cannot travel to RHS
- ↳ '/' means → 1 बार में कोई एक string
- ↳ You can take out common.

Q5] All strings that contains atleast 1 occurence of each symbol $\Sigma = \{a, b, c\}$

Soln] RE ⇒

$\{(a|b|c)^*.a.(a|b|c)^*.b.(a|b|c)^*.c(a|b|c)^* /$
$(a|b|c)^*.a.(a|b|c)^*.c.(a|b|c)^*.b(a|b|c)^* /$
$(a|b|c)^*.b.(a|b|c)^*.a.(a|b|c)^*.c.(a|b|c)^* /$
$(a|b|c)^*.b.(a|b|c)^*.c.(a|b|c)^*.a.(a|b|c)^* /$
$(a|b|c)^*.c.(a|b|c)^*.a.(a|b|c)^*.b.(a|b|c)^* /$
$(a|b|c)^*.c.(a|b|c)^*.b.(a|b|c)^*.a.(a|b|c)^*\}$

Q.6] $\Sigma = \{0, 1\}$ all strings not ending with (01)

Soln] ending with 00 / end with 10 / end with 11 / ϵ / 1 / 0

RE ⇒ $\{(0|1)^*.00 / (0|1)^*.1.0 / (0|1)^*.1.1 / \epsilon / 0 / 1$

$\Rightarrow \{(0|1)^* (00|10|11) / \epsilon / 0 / 1\}$

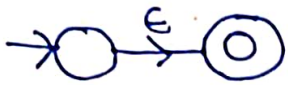
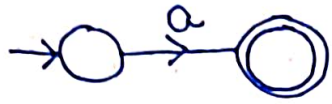
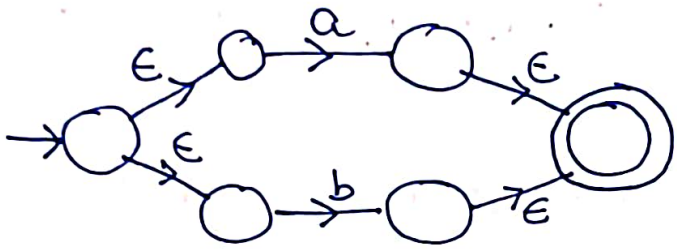
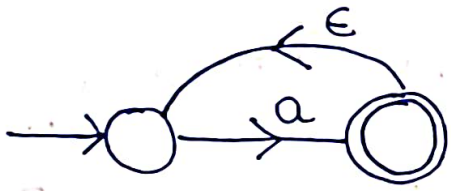
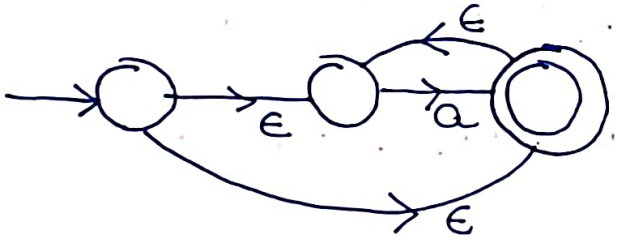
Q7] $\Sigma = \{0, 1\}$ all strings with even 0

Soln] $(1^* 0 1^* 0 1^*)^* / 1^*$

Q.8] $\Sigma = \{a, b, c\}$ $L = \{w : |w| \bmod 3 = 0\}$

Soln] $\left((a|b|c).(a|b|c).(a|b|c)\right)^*$

▷ Regular expression to NFA

RE	NFA	(Valid strings)
ϵ		$\{\epsilon\}$
a		$\{a\}$
$a \cdot b$		$\{ab\}$
$a \mid b$		$\{a, b\}$
a^+		$\{a, aa, aaa, \ldots\}$
a^*		$\{\epsilon, a, aa, aaa, \ldots\}$

Q.9] Draw NFA for $(a/b)^*$.

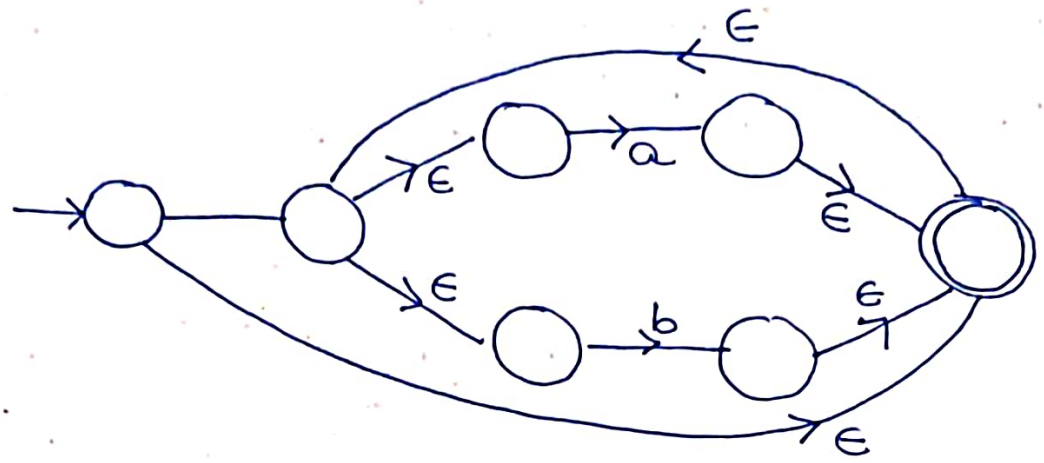
Step ① 1st draw (a/b)

Step ② Take '*' in account.

METHOD 1

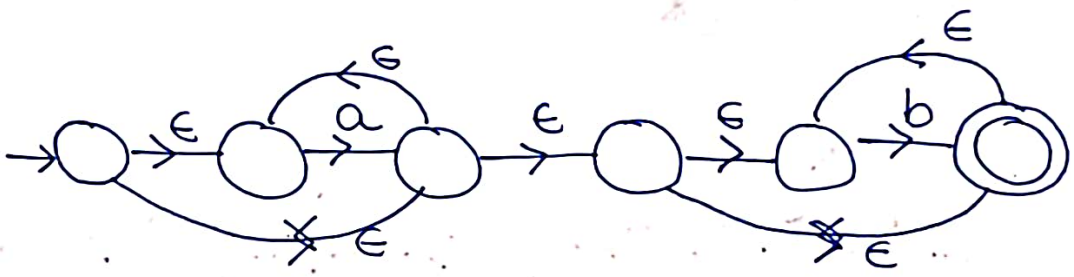
conjunction Method

Soln]



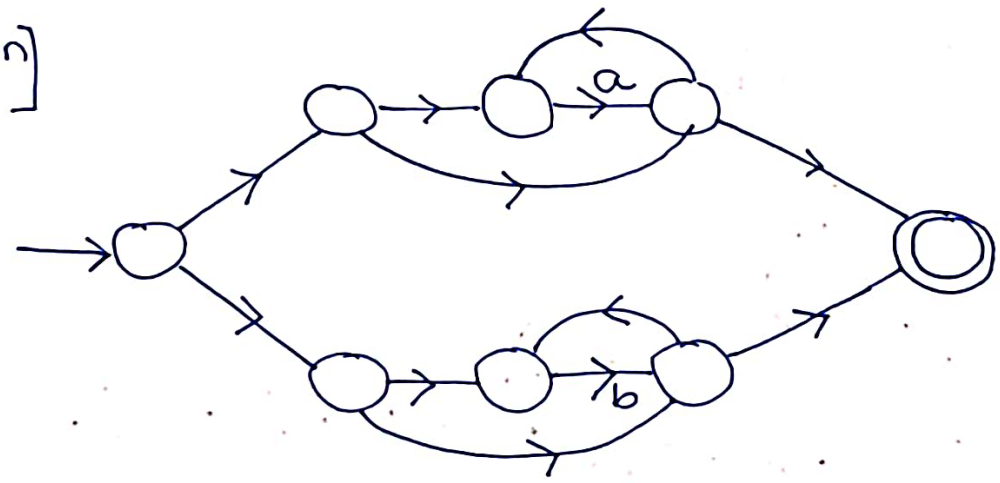
Q.10] $a^*.b^*$

Soln]



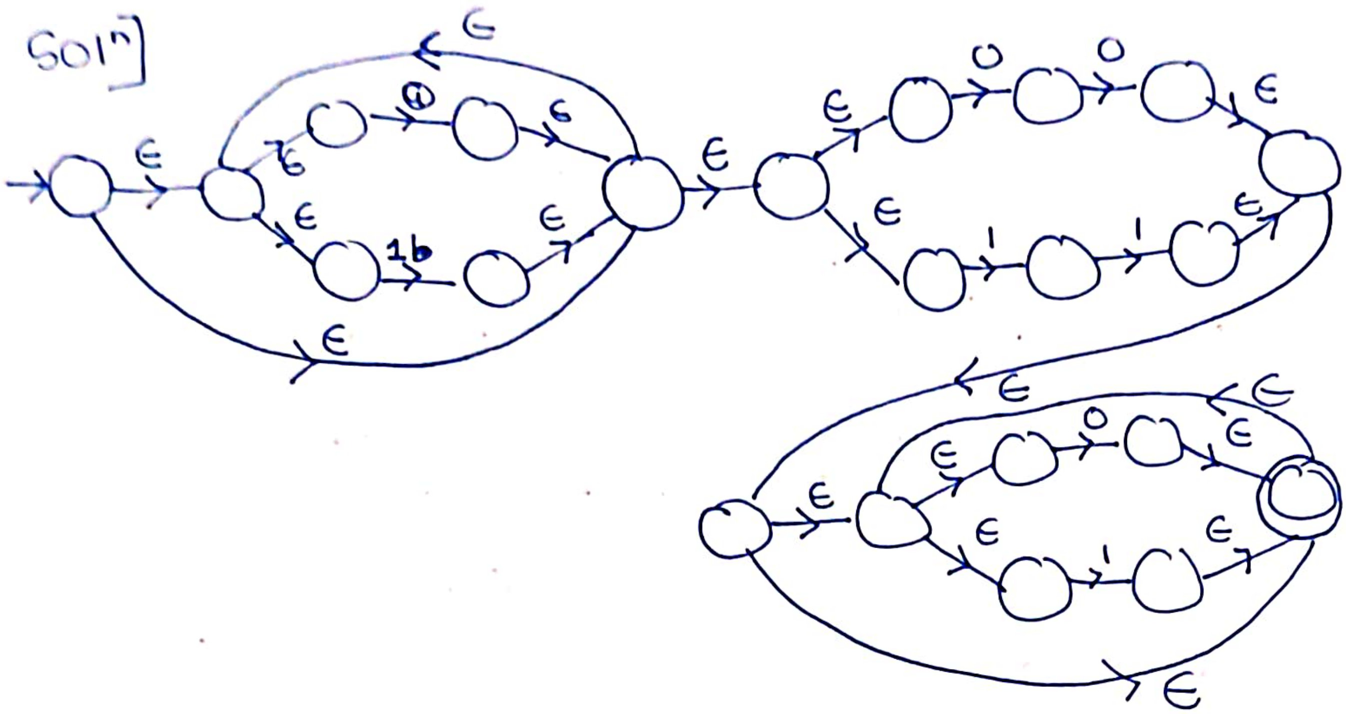
Q.11] a^*/b^*

Soln]



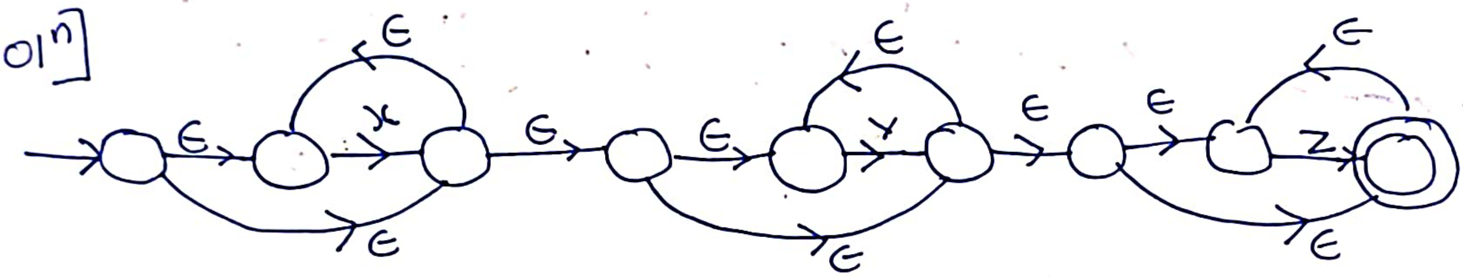
Q.12] RE = $(0/1)^*$ $(00/11)$ $(0/1)^*$ (draw separately)
① ② ③

Soln]



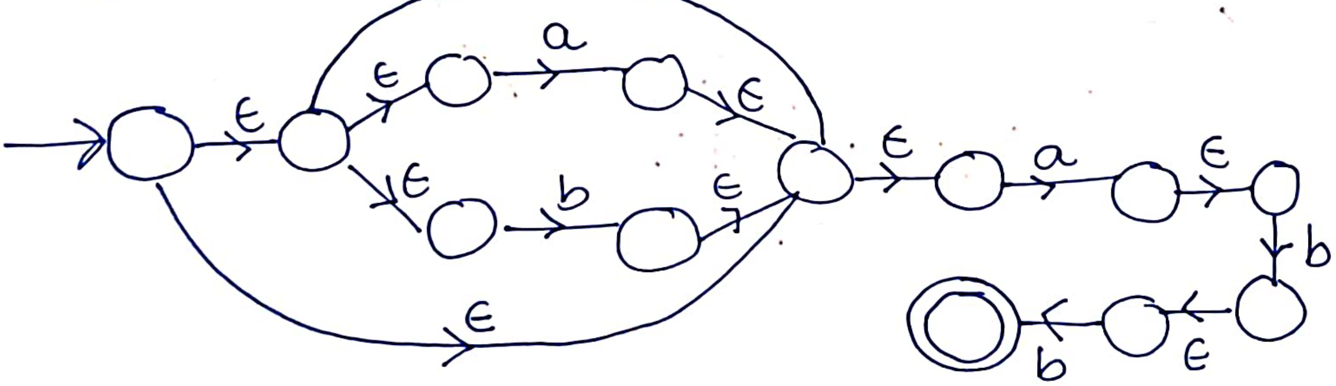
Q.13] $x^* y^* z^*$

Soln]



Q.14] $(a/b)^*abb$

Soln]



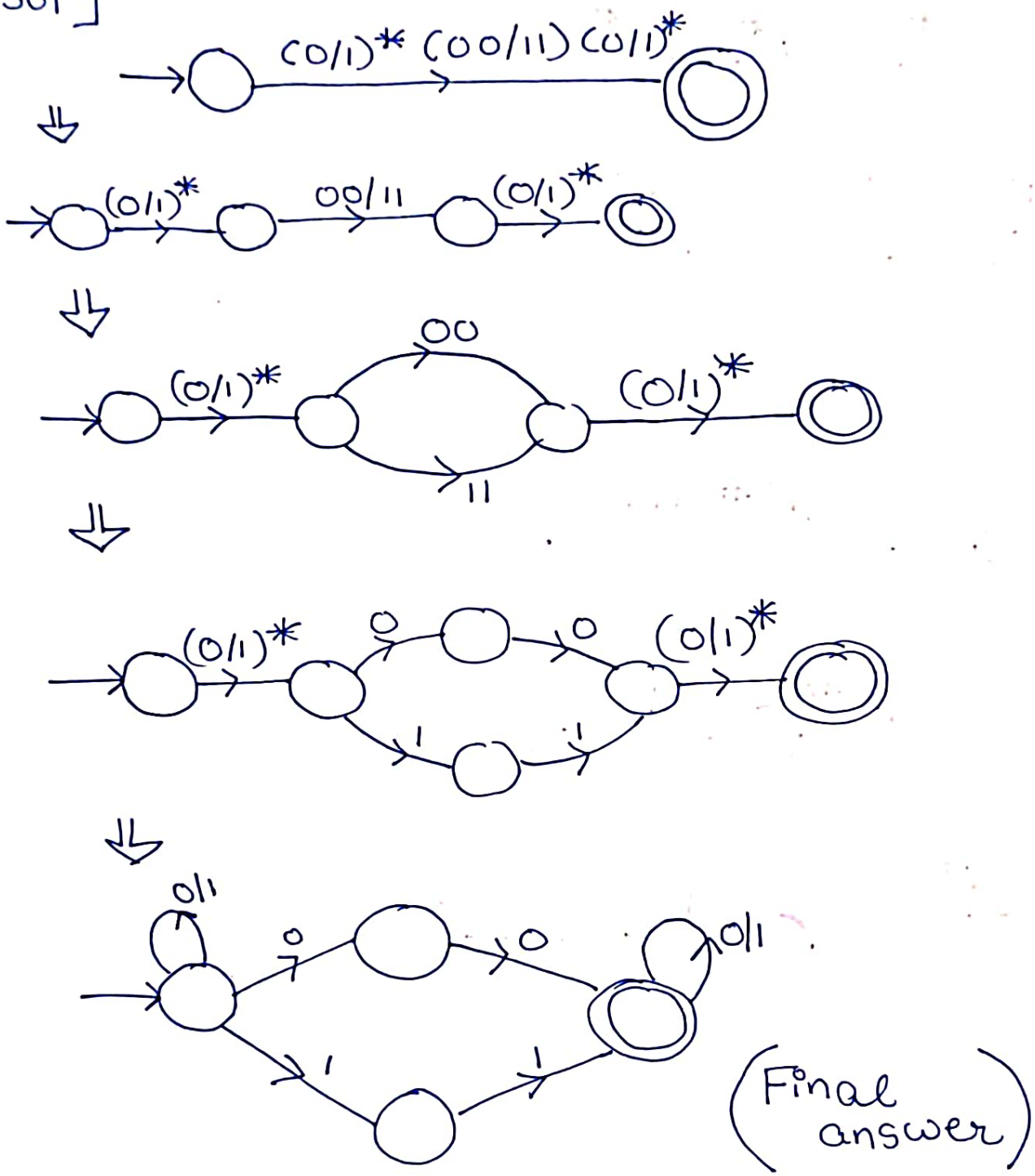
METHOD 2 ⇒ Divisive Method

Example : RE = $(0|1)^*(00|11)(0|1)^*$

↳ 1st draw NFA for complete operator ; then divide repeatedly.

↳ Less no of states ; Less "ϵ transitions"

Soln]



(Final answer)

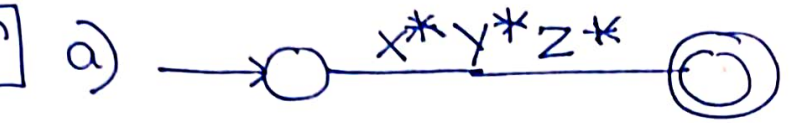
Q.15] Solve using Divisive method.

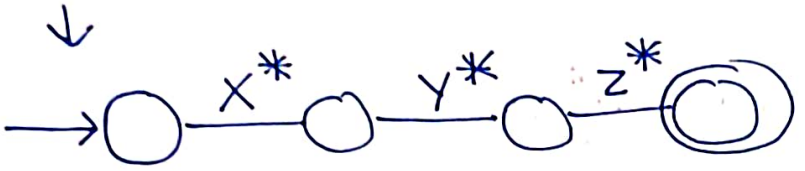
a) $x^* y^* z^*$

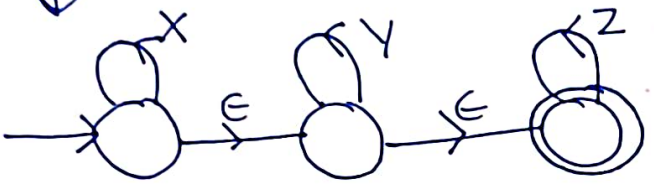
b) $(a/b)^* abb$

c) $(a/b)^* b(a/bb)^*$

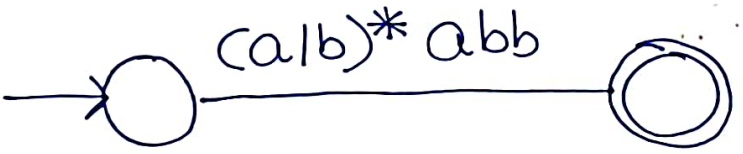
Soln] a)

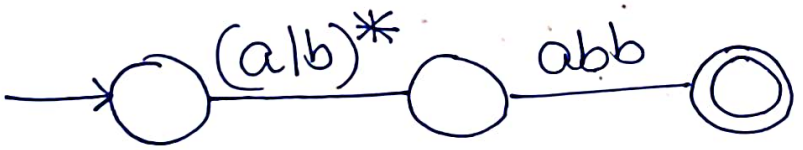


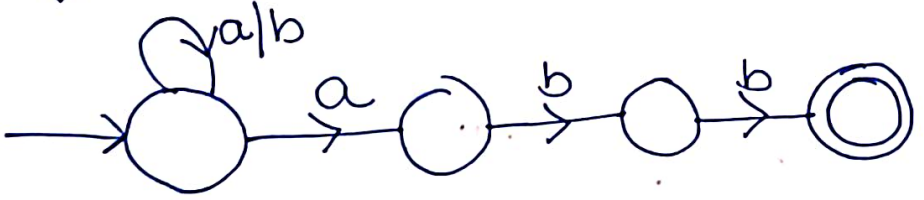




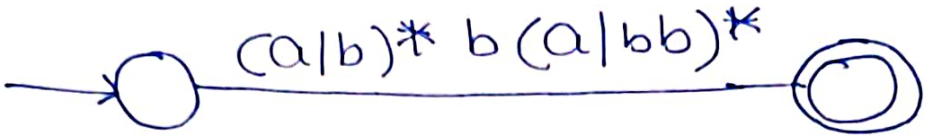
b]

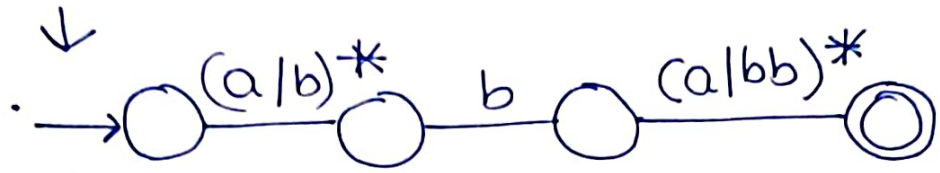


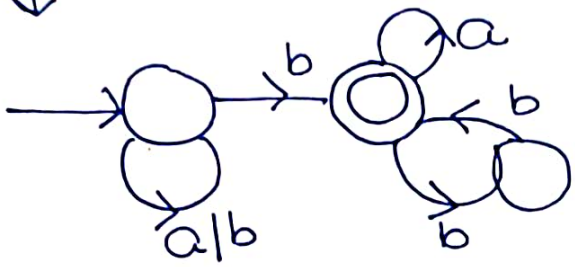




c] $(a|b)^* b (a|bb)^*$







▷ Identities of Regular Expression

NOTES: '+' means (OR) operation

① $\phi + R = R$

② $\phi \cdot R = R \cdot \phi = R$

③ $\epsilon R = R\epsilon = R$

④ $\epsilon^* = \epsilon$

⑤ $R + R = R$

⑥ $R^* R^* = R^*$

⑦ $RR^* = R^* R$

⑧ $(R^*)^* = R^*$

⑨ $\epsilon + RR^* = R^* = \epsilon + R^* R$

⑩ $(PQ)^* P = P(QP)^*$

⑪ $(P+Q)^* = (P^* Q^*)^* = (P^* + Q^*)^*$

⑫ $(P+Q)R = PR + QR$

⑬ $R(P+Q) = RP + RQ$

⑭ $Q + RP = QP^*$ ⇒ **ARDEN'S THEOREM**

↳ Let P & Q be two regular exp. over Σ and if $P \neq \epsilon$ then the equation $R = Q + RP$ has a unique soln $R = QP^*$

↳ Proof → Substitute R in $R = Q + RP$ 'n' times we get → $Q(\epsilon + P^1 + P^2 \ldots P^n)$

$\Rightarrow Q P^*$

Q.16] Simplify: $RE = \epsilon + 1^*(011)^*(1^*(011)^*)^*$

Soln] Let $P = 1^*(011)^*$

$RE = \epsilon + PP^*$ [Identity 9]

$= P^*$

$= (1^*(011)^*)^*$ [by Identity 11]

$= \underline{(1 + 011)^*}$

Q.17] Prove that:

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

Soln] Let $P = (1 + 00^*1)$

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

LHS: ~~PPQQ~~ $P + PQ^*Q$ (Identity 12)

$\Rightarrow P(\epsilon + Q^*Q)$

$\Rightarrow P(Q^*)$ (Identity 9)

Now, solving $P \Rightarrow (1 + 00^*1)$

$\Rightarrow 1(\epsilon + 00^*)$ (by 12)

$\Rightarrow 10^*$ (by 9)

$\Rightarrow 0^*1$ (by 7)

Now, we have

$\Rightarrow P(Q^*)$

$\Rightarrow 0^*1(0 + 10^*1)^*$

LHS = RHS Hence Proved

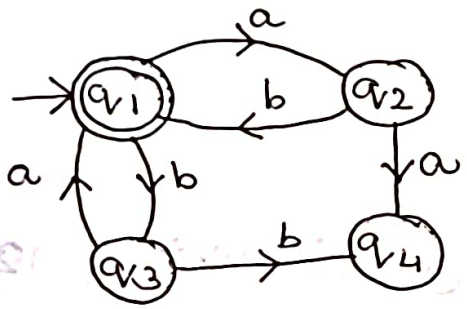
◁▷ Finite Automata to RE

↳ 2 step process.

Step ①: Write equation for every state of FA i.e., how many edges are coming into the state.

Step ②: Solve for final state by using identities and Arden's theorem.
Value of final state is the regular expression.

Q.18]



$\rightarrow q_1 = q_2b + q_3a + \epsilon$

$q_2 = q_1a$

$q_3 = q_1b$

$q_4 = q_2a + q_3b$

Now, Solving for final state (q_1)

$q_1 \Rightarrow q_1ab + q_1ba + \epsilon$

$q_1 = q_1(ab + ba) + \epsilon$ by (12)

$= \epsilon(ab+ba)^*$ by Arden's theorem

$= (ab+ba)^*$

↳ Required regular expression.

Note → Add ϵ for initial state

Q. 19]

Soln]

$\rightarrow q_1 = q_1a + q_2b + \epsilon$

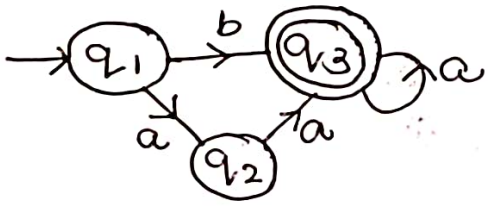
$q_2 = q_1a + q_3a + q_2b$

$* \; q_3 = q_2a$

$q_3 = q_2a$

$= (q_1a + q_3a + a + q_2b)\,a$

Q. 20]



Soln] $\rightarrow q_1 = \epsilon$

$q_2 = q_1a$

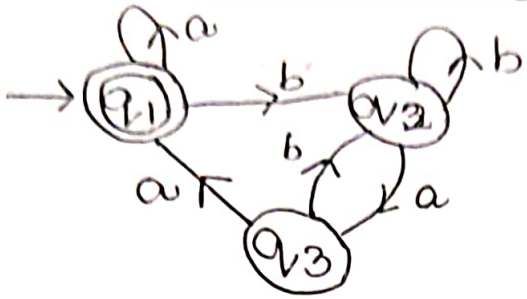
$* \; q_3 = q_1b + q_2a + q_3a$

$q_3 = \epsilon b + q_1aa + q_3a$

$= b + aa + q_3a$

$R = Q + RP = QP^*$

$\therefore \; q_3 = (b + aa)a^*$

Q.2] 

Soln] $q_1 = q_1a + q_3a + \in$

$q_2 = q_1b + q_2b + q_3b$

$q_3 = q_2a$

$q_2 = q_1b + q_2b + q_2ab$

$q_2 = q_1b + q_2(b+ab)$ $\quad R = Q + RP = QP^*$

$q_2 = q_1b(b+ab)^*$

$q_1 = q_1a + q_2aa + \in$

$= q_1a + q_1b(b+ab)^*aa + \in$

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

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

⇨ FA to Regular Grammar

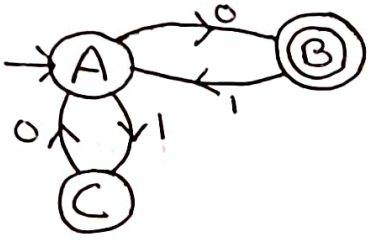
Step ①: Write equation for every state of finite automata

Step ②: Every outgoing transition will have a term in equation

Step ③: Add ∈ for final state.

Step ④: Start symbol of grammar is the symbol for initial state of finite automata

Q.22]



Soln]

$A = 0B \mid 1C$

$B = 1A \mid \in$

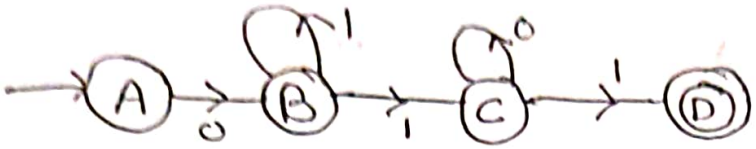
$C = 0A$

$\Rightarrow$

$A \rightarrow 0B \mid 1C$

$B \rightarrow 1A \mid \in$

$C \rightarrow 0A$

Q.23] 

Soln] → A = 0B
B = 1C/1B
C = 1D/0C
D = ∈

⇒ A → 0B
B → 1C/1B
C → 0C/1D
D → ∈

⇨ Right linear Grammar to Left linear Grammar.

Step ①: Obtain RE from Grammar
Step ②: Reverse this RE
Step ③: Obtain Right linear grammar from reversed RE. (RE → FA → RLG)
Step ④: Reverse right side of every product

Q.24] S → 01B/0
B ⇒ 1B/11

Soln] S = 01B/0
B = 1B/11
= 1B + 11 (by Arden's theorem)
B = 1* 11 — ①

Put ① in S, we get

S = 011* 11/0 (this is RE)

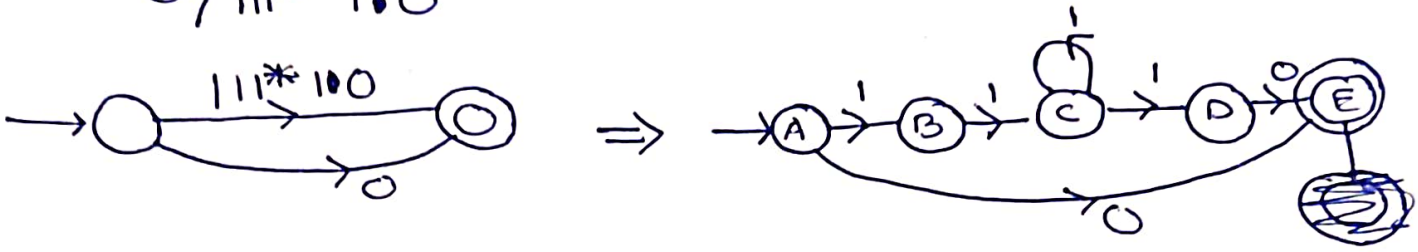
∴ RE = 011* 11/0

Now, reversing the regular expression:

Reversed (RE) = ~~0/1*110~~ 0/111*10

1st converting Reversed (RE) into FA.

0/111*100



FA

Now, getting RLG from FA.

$\rightarrow A \rightarrow 1B/0E$
$B \rightarrow 1C$
$C \rightarrow 1C/1D$
$D \rightarrow 0E$
$* E \rightarrow \epsilon$

Reversing ⇒

$\rightarrow A \rightarrow B1/E0$
$B \rightarrow C1$
$C \rightarrow C1/D1$
$D \rightarrow E0$
$* E \rightarrow \epsilon$

↳ final Ans. (Left linear grammar)

Q.25] $\rightarrow A \rightarrow gB$
$B \rightarrow aB/gC/g$
$C \rightarrow gB$

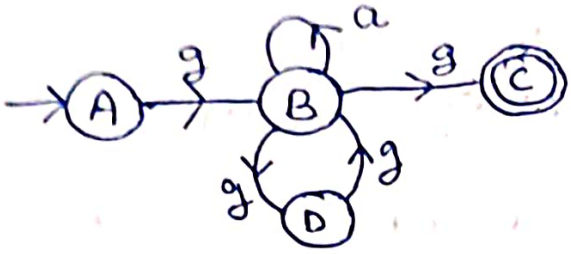
Sol^n] $A = gB$

$B = aB/gC/g \Rightarrow B = aB + gC + g$
$= aB + ggB + g$
$= (a+gg)B + g$ (ardens thm)
$= (a+gg)^* g$

↳ Put this in A

$A = g(a/gg)^* g$

FA ⇒



$A \to gB$
$B \to aB / gD / gC$
$C \to \epsilon$
$D \to gB$

Now, Reversing ⇒ $\to A \to Bg$
$B \to Ba / Dg / Cg$
$C \to \epsilon$
$D \to Bg$

⇨ Left Linear Grammar to Right Linear Grammar

Step ①: Reverse the RHS of every production
Step ②: Obtain RE
Step ③: Reverse RE
Step ④: Obtain RLG

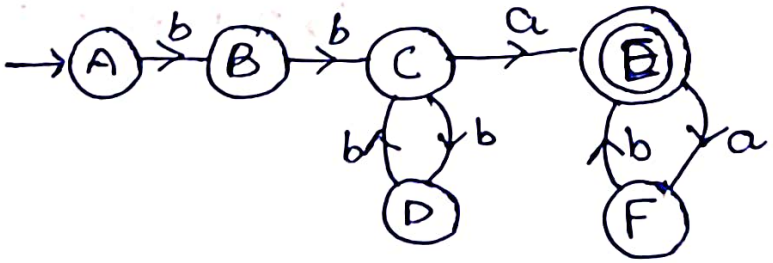
Q.26] $S \to Sab / Aa$
$A \to Abb / bb$

Soln] $S \to baS / aA$
$A \to bbA / bb \Rightarrow A = (bb)^* bb$

$\therefore S = baS / a(bb)^* bb$

RE ⇒ $S = (ba)^* a(bb)^* bb$

reversed (RE) = $bb(bb)^* a(ab)^*$



$A \to bB$
$B \to bc$
$C \to bD/aE$
$D \to bC$
$E \to aF/\epsilon$
$F \to bE$

↳ Answer

Q.27] $S \to Ag$
$A \to Aa/Bg/g$
$B \to Ag$

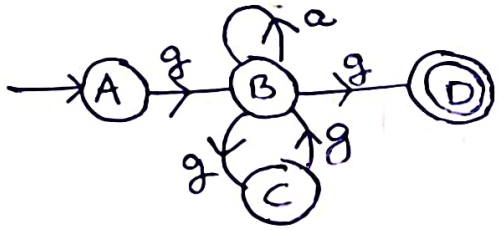
Soln] $S \to gA$
$A \to aA/gB/g$
$B \to gA$

$\Rightarrow A = aA/ggA/g$
$= (a/gg)A/g = (a/gg)^*g$

$S = g(a/gg)^*g$

Reversed (S) $= g(gg/a)^*g$

FA $\Rightarrow$



$A \to gB$
$B \to aB/gC/gD$
$C \to gB$
$D \to \epsilon$

↳ Answer
↳ Required Right linear grammar

CONTEXT FREE GRAMMAR

↳ More powerful than regular grammar.
↳ Regular grammar is a subset of context free grammar.
↳ Grammar of all programming languages in context free gramma

Q.28] Design context free grammar for all strings from a to b which is a palindrome.

Soln] $G_1 = (V, T, P, S)$

where,

{S} V → Set of NT
{a, b} T → Set of Terminal
P → Set of production rules
{S} S → Start symbol.

→ find values to these parameters to get the ans.

Now,

$S \rightarrow aSa \mid bSb \mid a \mid b \mid \in$ → production rules

Q.29] Design CFG for language

$L = \{ 0^m 1^n 0^{m+n} ; m, n \geq 1 \}$

Soln]

$S \rightarrow OAO$
$A \rightarrow OAO \mid 1BO$
$B \rightarrow 1BO \mid \in$

Q.30] $L = \{a^m b^n : m \neq n\}$

Soln] $S \to aSb \mid Aa \mid bB$

$A \to Aa \mid \epsilon$

$B \to bB \mid \epsilon$

Q.31] $L = \{a^p b^q c^m : q = p+m\}$

Soln] $L = \{a^p b^{p+m} c^m\}$.

$S \to AB$

$A \to aAb \mid \epsilon$

$B \to bBc \mid \epsilon$

Q.32] Write CFG for following lang:→

(a) $L = a^n b^n : n \geq 0$

(b) $L = a^n b^n : n \geq 1$

(c) $L = a^n b^{n+2} : n \geq 0$

(d) $L = a^{2n} b^n : n \geq 0$

(e) $L = a^m b^n : m > n, n \geq 0$

Soln] a) $S \to aSb \mid \epsilon$

b) $S \to aSb \mid ab$

c) $S \to aSb \mid bb$

d) $S \to aaSb \mid \epsilon$

e) $S \to ASb \mid aA\epsilon$

$A \to aA \mid a\epsilon$

$[S_1 = A]$

⇨ Optimization of CFGs

↳ A symbol in context free grammar is useful if it derives 2 strings of terminal.

② It is used in derivation of atleast 1 string of grammar.

↳ Symbols that doesnot satisfy 1st condition are known as useless symbols and which does not satisfy 2nd condition are known as unreachable symbols.

↳ All terminal symbols are useful.

→ NT are useful if RHS of that symbol contains only terminal or any combination of terminal are useful non terminal.

↳ Starting symbol is always reachable. All symbols on RHS of reachable are also reachable.

Q 33] optimize the CFG

$V = \{a, b, c, d, A, X, S\}$

✓ $S \rightarrow aB \mid bX$

✓ $A \rightarrow BAd \mid \cancel{BX} \mid a$ → Terminal → ∴ A useful

$XB \rightarrow aSB \mid bB\cancel{X}$

✓ $X \rightarrow SBD \mid aBX \mid ad$ → Terminal → ∴ X useful

Soln] Remove all rules with B.

↳ $S \rightarrow bX$

↳ $XA \rightarrow bSX \mid a$ → unreachable.

↳ $X \rightarrow ad$

∴ $\boxed{\begin{array}{l} S \rightarrow bX \\ X \rightarrow ad \end{array}}$ → Answer.

Q.34] ✓ $S \rightarrow AC \mid \cancel{SB}$

✓ $A \rightarrow bASC \mid a$

$\times B \rightarrow aSB \mid BbC$

✓ $C \rightarrow \cancel{BC} \mid ad$

Soln] B → useless.. ⇒ ✓ $S \rightarrow AC$
✓ $A \rightarrow bASC \mid a$
✓ $C \rightarrow ad$

} All are reachable

Q.35] ✓ $A \rightarrow xyz \mid Xyzz$

$X \rightarrow Xz \mid xYx$

$Y \rightarrow yYy \mid XZ$

✓ $Z \rightarrow Zy \mid z$

⇒ $A \rightarrow xyz$

$Z \rightarrow Zy \mid z$ → unreachable

$\boxed{A \rightarrow xyz}$

Note: do not write directly, write complete statements

⇨ Removal of ε-production

Ex. $S \to aA$
$A \to b|\epsilon$

① → Replace all productions rules with ε that contain ε.

② Add the new production rules into the previous productions.

Soln] Replace A by ε ⇒ $S \to aA$
$\to a\epsilon$
$S \to a$

∴ we get ⇒ $\boxed{\begin{array}{l} S \to aA/a \\ A \to b \end{array}}$

Q.36] $S \to ABAC$
$A \to aA|\epsilon$
$B \to bB|\epsilon$
$C \to c$

Soln] Replace all A & B on RHS.
(2 ε productions)

→ Replacing As

$S \to BAC$
$S \to ABC$
$S \to BC$
$A \to a$

Replacing B's

$S \to AAC$
$B \to bB$

→ All new productions ↓ Add in original grammar

↳ All new productions
↳ add in original grammar

$S \to ABAC/BAC/ABC/BC$
$A \to aA/a$
$B \to bB/\epsilon$
$C \to c$

→ Now, replace all B's in new grammar generated.

$S \to AAC \mid AC \mid AC \mid C$

$B \to b$

↓ New grammar

$S \to ABAC / BAC / ABC / BC / AAC / AC / C$

$A \to aA \mid a$

$B \to bB \mid b$

$C \to c$

Q.37] $S \to AaA$

$A \to Sb \mid bCG \mid \epsilon$

$C \to cc \mid abb$

Soln] $S \to aA \mid Aa \mid a$

$\Rightarrow S \to AaA \mid aA \mid Aa \mid a$

$A \to Sb \mid bCG \mid$

$C \to cc / abb$

Q.38] $S \to ABaC$

$A \to BC$

$B \to b \mid \epsilon$

$C \to D \mid \epsilon$

$D \to d$

Soln] for B ↴

$S \to AaC$ } Add to original

$A \to C$

we get →

$S \to ABaC / AaC$

$A \to BC \mid C$

$B \to b$

$C \to D \mid \epsilon$

$D \to d$

Now, for C ↴

$S \to ABa / Aa$ } add.

$A \to B \mid \epsilon$

→

$S \to ABaC / AaC / ABa / Aa$

$A \to B \mid C \mid B \mid \epsilon$

$B \to b$

$C \to D$

$D \to d$

Now, removing for A; we get

$S \to ABaC / AaC / ABa / Aa / BaC / aC / Ba / a$

$A \to BC / C / B$

$B \to b$

$C \to D$

$D \to d$ | final ans.

⇨ Removal of unit production.

↳ Ex $A \to B$ (Single non terminal on both LHS & RHS)

↳ Only increases the no. of steps to check a string.

⊙ 39] $S \to \tilde{A}B$

$A \to a$

$B \to \tilde{C} / b$

$C \to D$

$D \to E$

$E \to a$

Soln] 3 unique production

⇓

$\therefore S \to AB$

$A \to a$

$B \to a / b$

$C \to a$

$D \to a$ X

$E \to a$

$\Rightarrow S \to AB$

$A \to a$

$B \to a / b$

↳ Remove of left Recursion

$A \to A\alpha \mid \beta$ (51)

↳ $A \to \beta A'$

$A' \to \alpha A' \mid \epsilon$

↳ Remember

Q.40] $S \to aBDh$

$B \to \underset{A}{B}\underset{\alpha}{b} \mid \underset{\beta}{c}$ → left recursion

$D \to EF$

$E \to g \mid \epsilon$

$F \to f \mid \epsilon$

∴ $B \to Bb \mid c$ is replaced by $B \to cB'$

$B' \to bB' \mid \epsilon$

Q.41] $S \to (L) \mid a$

$L \to L,S \mid S$ → left recursion

Sol] $L \to aSL'$

$L' \to ,SL' \mid \epsilon$ } Just replace this

Q.42] $A \to \underset{\alpha}{ABC} \mid \underset{\alpha_2}{AC} \mid \underset{\alpha_3}{ACB} \mid \underset{B_1}{CD} \mid \underset{B_2}{D}$

$B \to b \mid \epsilon$

$C \to c \mid \epsilon$

$D \to d \mid \epsilon$

⇩

General formula →

$A \to A\alpha_1 \mid A\alpha_2 \mid A\alpha_3 \mid \dots \mid A\alpha_n \mid B_1 \mid B_2 \mid \dots \mid B_m$

⇩

$A \to B_1A' \mid B_2A' \mid B_3A' \mid \dots \mid B_mA'$

$A' \to \alpha_1A' \mid \alpha_2A' \mid \dots \mid \alpha_nA' \mid \epsilon$

Ans] ⇒ $A \to CDA' \mid DA'$

$A' \to BCA' \mid CA' \mid CBA' \mid \epsilon$

$B \to b \mid \epsilon$

$C \to c \mid \epsilon$ $D \to d \mid \epsilon$.

⇨ CHOMSKY NORMAL FORM (CNF)

↳ In given grammar, every production can either have 2 non terminal or exactly 1 terminal symbol. ex ① $A \to BC$, $A \to c$

② Example of CNF: $S \to AB \mid a$, $A \to a$, $B \to BA \mid SA \mid b$

Q43] convert into CNF

$S \to ABa$

$A \to aab$

$B \to Ac$

Soln] Start taking additionals productions.

$Xa \to a$

$Xb \to b$

$Xc \to c$

$\Rightarrow$

$S \to ABXa$

$A \to XaXaXb$

$B \to AXc$ ✓

$Y \to BXa$

$Z \to XaXb$

$\Rightarrow$

$S \to AY$

$A \to XaZ$

$B \to AXc$

$Y \to BXa$

$Z \to XaXb$

$Xa \to a$

$Xb \to b$

$Xc \to c$

→ CNF form

Q 44] $S \to aSb \mid bb$

Soln] $X_b \to b$, $X_a \to a$ $\Rightarrow$ ~~$S \to X_a S X_b$~~

$S \to X_a \underline{S X_b} \mid X_b X_b$

$Y \to S X_b$

$\Rightarrow$ $S \to X_a Y \mid X_b X_b$, $Y \to S X_b$, $X_b \to b$, $X_a \to a$ → CNF final answer.

GREIBACH NORMAL FORM (GNF)

(non terminal)

↳ $S \to aX$ $\quad X \to V^*$ (V - set of NT)

~~Q.45~~ $S \to aB \mid bBC$, $B \to a$, $C \to cB \mid c$ } already in GNF

↳ Start with terminal $ contain any number of non-terminal symbols.

Q.45] • $S \to AB$ Convert to GNF.

$A \to aA \mid bB \mid b$

$B \to b$

Soln] Replace all A's in S.

↳ $S \to aAB \mid bBB \mid bB$

$A \to aA \mid bB \mid b$

$B \to b$

Q.46] $S \to abSa \mid aa$

Soln] $X_a \to a$, $X_b \to b$ $\Rightarrow$ $S \to aX_bSX_a \mid aX_a$, $X_a \to a$, $X_b \to b$

Q.47] $S \to abAB$
$A \to bAB \mid \epsilon$
$B \to BAa \mid A \mid \epsilon$

get CNF & GNF both.

Note → Before converting CNF/GNF, 1st remove all ϵ-production.

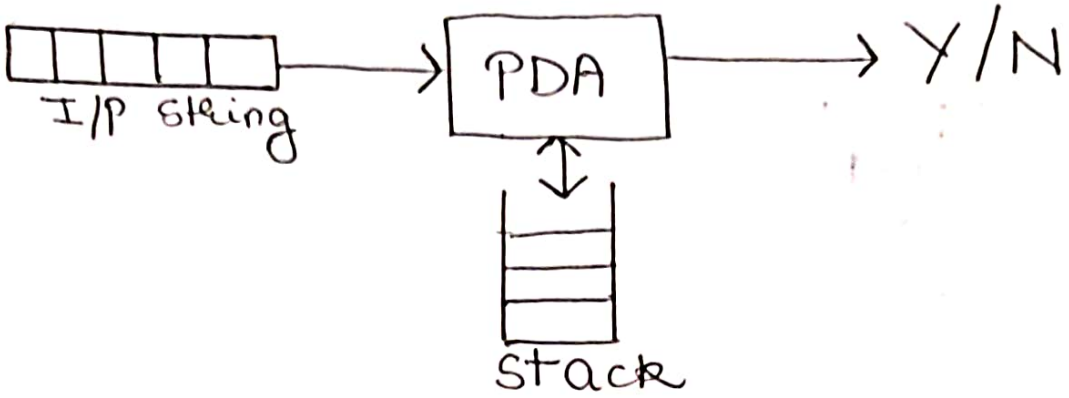
Soln] Step① Removing All ϵ.

Pushdown Automata

2Q 5M each

UNIT - 4

PUSHDOWN AUTOMATA



- Finite Automata → No memory
- Pushdown Automata has memory (stack)

⇨ Mathematically, Pushdown Automata is defined by 7 tuples ↓

$M = (Q, \Sigma, \Gamma, \delta, Q_0, Z_0, F)$

where,

Q → Finite set of states in PDA.

Σ → Finite set of i/p symbols.

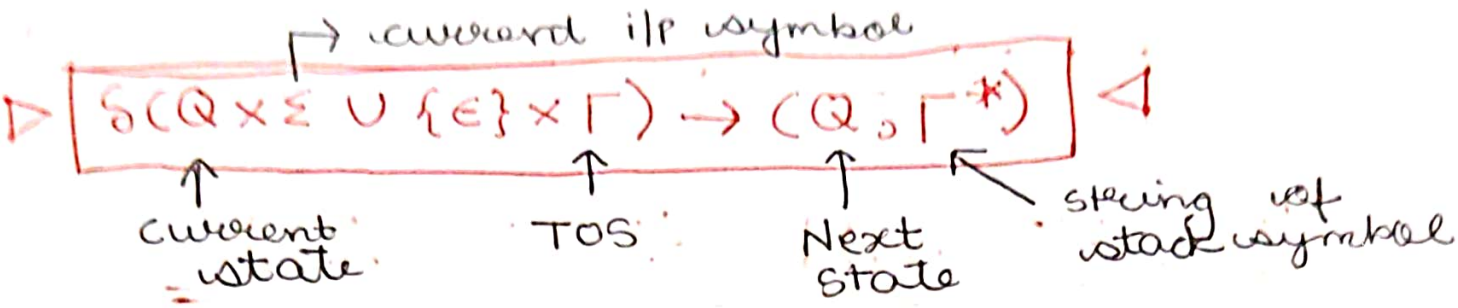
new Γ → Finite set of stack symbols.

δ → Transition function.

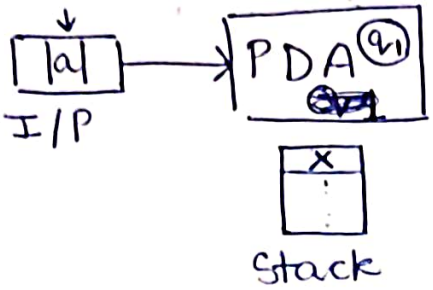
Q_0 → Initial state of PDA

new Z_0 → Start symbol of stack.

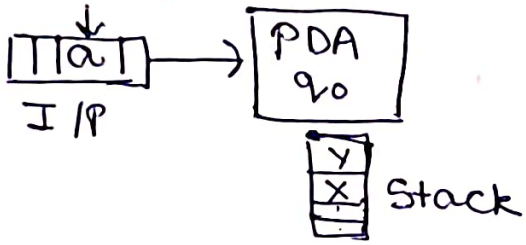
F → set of final states of PDA.



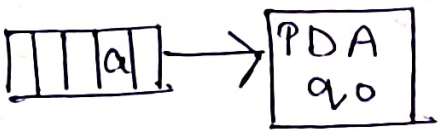
Ex ① $\delta(q_0, a, X) \rightarrow (q_1, X)$



② $\delta(q_0, a, X) \rightarrow (q_0, YX)$ → called as (Push transition)



③ $\delta(q_0, a, X) \rightarrow (q_0, \in)$ → operation (pop, transition)
↳ X is popped.



empty stack.

④ $\delta(q_0, a, X) \rightarrow (q_0, X)$ → (No operation)

⑤ $\delta(q_0, a, X) \rightarrow (q_0, Y)$ → (Replacing TOS)

Note → You can pop Max. 1 symbol.
You can push any no. of symbols

Q.1] $L = \{ w c w^R / w \in (0/1)^* \}$

Draw PDA. ($w \rightarrow$ any string of 0 & 1)

($w^R \rightarrow$ Reverse of w)

Soln] Valid strings → 0101 c 1010,
0011 c 1100

Logic → ① Push every symbol before 'c' into the stack with state q_0.

② After getting 'c' in input change state to q_1

③ In q_1 pop one symbol from the stack at a time if it matches the next i/p symbol.

④ If TOS contains Z_0 at the end, declare string as valid.

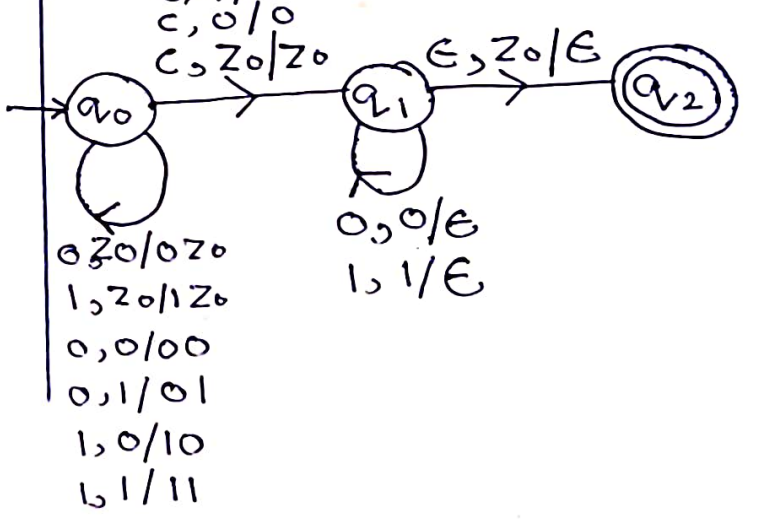
Now, Transitions ⇒

$\delta(q_0, 0, Z_0) \rightarrow (q_0, 0Z_0)$
$\delta(q_0, 1, Z_0) \rightarrow (q_0, 1Z_0)$
$\delta(q_0, 0, 0) \rightarrow (q_0, 00)$
$\delta(q_0, 0, 1) \rightarrow (q_0, 01)$
$\delta(q_0, 1, 0) \rightarrow (q_0, 10)$
$\delta(q_0, 1, 1) \rightarrow (q_0, 11)$

$\delta(q_0, c, Z_0) \rightarrow (q_1, Z_0)$
$\delta(q_0, c, 0) \rightarrow (q_1, 0)$
$\delta(q_0, c, 1) \rightarrow (q_1, 1)$

$\delta(q_1, 0, 0) \rightarrow (q_1, \epsilon)$
$\delta(q_1, 1, 1) \rightarrow (q_1, \epsilon)$
$\delta(q_1, \epsilon, Z_0) \rightarrow (q_2, \epsilon)$

* diagram



Q2] $L = \{a^n b^m c^{n+m} / n, m \geq 1\}$

Soln] Logic: (1) Till a is i/p symbol, push 'X' in stack.

(2) Till b is i/p symbol, change state & push 'X' in Stack.

(3) As soon as 'C' is in i/p symbol, start poping the stack till the stack is empty, after changing the state.

(4) If TOS contains 'Z0' declare string was valid.

∴ Transitions ↴

$\delta(q_0, a, Z_0) \rightarrow (q_0, XZ_0)$

$\delta(q_0, a, X) \rightarrow (q_0, XX)$

$\delta(q_0, b, X) \rightarrow (q_1, XX)$

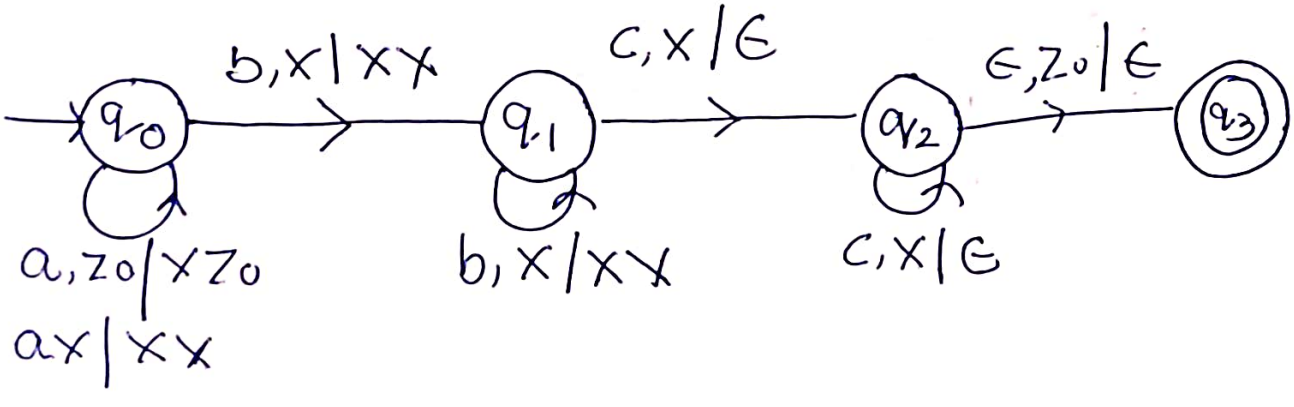
$\delta(q_1, b, X) \rightarrow (q_1, XX)$

$\delta(q_1, c, X) \rightarrow (q_2, \epsilon)$

$\delta(q_2, c, X) \rightarrow (q_2, \epsilon)$

$\delta(q_2, \epsilon, Z_0) \rightarrow (q_3, \epsilon)$

diagram ↴



Q.3] $L \rightarrow \{a^n b^{2n} / n \geq 1\}$

Soln] Logic: ① Till 'a' is i/p symbol, push 'XX' into the stack.

② If i/p symbol is b, change state and pop X for every 'b' in i/p. symbol.

③ If ip is ϵ and TOS contains Z_0, declare string as valid.

Now

Transitions:

$\delta(q_0, a, Z_0) = (q_0, XXZ_0)$

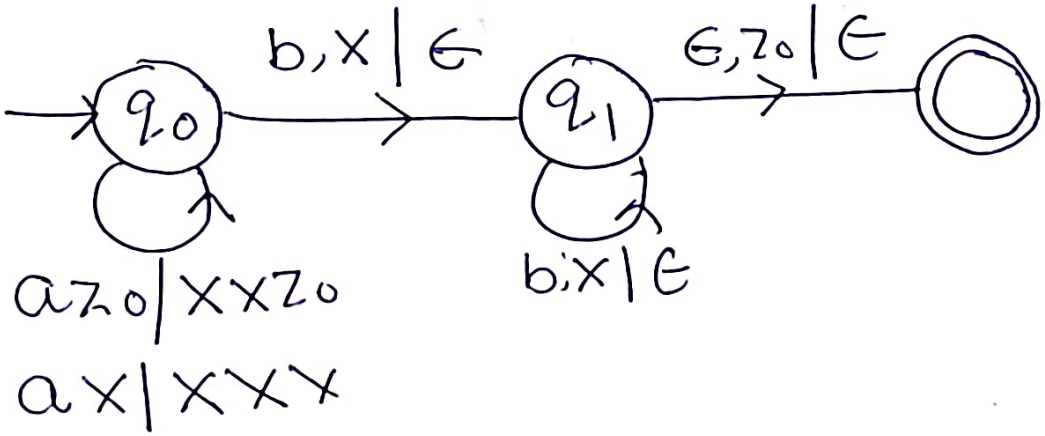
$\delta(q_0, a, X) = (q_0, XXX)$

$\delta(q_0, b, X) = (q_1, \epsilon)$

$\delta(q_1, b, X) = (q_1, \epsilon)$

$\delta(q_1, \epsilon, Z_0) = (q_2, \epsilon)$

Now, diagram:



Instantaneous Descriptions of PDA (IDs)

(current state of PDA ; Remaining strings in I/P Buster ; content of stack)

'$\vdash$' symbol means 'Application of a transition'

ex $L = a^n b^{2n} | n \geq 1$

String $\Rightarrow$ aabbbb

$\rightarrow (q_0, aabbbb, z_0) \vdash (q_0, abbbb, XXz_0)$

$\vdash (q_0, bbbb, XXXXz_0) \vdash (q_1, bbb, XXXz_0)$

$\vdash (q_1, bb, XXz_0) \vdash (q_1, b, Xz_0)$

$\vdash (q_1, \epsilon, z_0) \vdash (q_2, \epsilon, \epsilon)$

↳ These are the instantaneous descript^n of PDA or IDs of PDA.

Types of PDA

2 types

- Accepts string by final state P_F
- Always has final states.
- $P_F = (Q, \Sigma, \Gamma, \delta, Q_0, Z_0, F)$
- Last element of stack can be anything
- $(Q_0, w, z_0) \vdash^* (Q_F, \epsilon, \gamma)$ — Imp

- Accepts string by empty stack P_N
- doesnot have final state.
- $P_N = (Q, \Sigma, \Gamma, \delta, Q_0, Z_0)$
- Last symbol of stack $\equiv \epsilon$.
- $(Q_0, w, z_0) \vdash^* (Q, \epsilon, \epsilon)$ — Imp

[redacted]

Q.4] $\{L = ww^R / w \in (0+1)^*\}$ ① Draw P_N → Non deterministic PDA.

② Also draw IDs for string 001100

Sol] ① $\delta(q_0, 0, z_0) = (q_0, 0z_0)$

$\delta(q_0, 1, z_0) = (q_0, 1z_0)$

$\delta(q_0, 0, 0) = \{(q_1, \epsilon), (q_0, 00)\}$

$\delta(q_0, 0, 1) = (q_0, 01)$

$\delta(q_0, 1, 0) = (q_0, 10)$

$\delta(q_0, 1, 1) = \{(q_\phi, \epsilon), (q_0, 11)\}$

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

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

$\delta(q_1, \epsilon, z_0) = (q_1, \epsilon)$

② String ⇒ 001100

No transition

$(q_0, 001100, z_0) \vdash (q_0, 01100, 0z_0) \vdash (q_1, 1100, z_0)$ ✗

$\vdash (q_0, 1100, 00z_0)$

$\vdash (q_0, 100, 100z_0) \vdash (q_0, 00, 1100z_0)$

$\vdash (q_1, 00, 00z_0)$ $\vdash (q_0, 0, 01100z_0)$ ✗ ✗

$\vdash (q_1, 0, 0z_0)$

$\vdash (q_1, \epsilon, z_0)$

$\vdash (q_1, \epsilon, \epsilon)$

∴ the string is valid

Note → Non deterministic PDA cannot always be converted into deterministic PDA

→ Deterministic PDA cannot always be designed for context free language/grammar.

Q.5] $L = \{a^n, b^m c^{m-n} / m, n > 1 \text{ and } m > n\}$

Q.6] $L = \{a^n b^m a^m b^n / m, n \geq 1\}$

Q.5 Soln] Logic: ① for every a, push a in the stack.

② If i/p is b, change state and start poping.

③ If i/p is b & stack TOS is Z_0, start pushing a into the stack

④ If i/p is c, change state & start poping.

⑤ In end if Z_0 is TOS then string is valid.

Transitions:

$(q_0, a, Z_0) \rightarrow (q_0, aZ_0)$

$(q_0, a, a) \rightarrow (q_0, aa)$

$(q_0, b, a) \rightarrow (q_1, \epsilon)$

$(q_1, b, a) \rightarrow (q_1, \epsilon)$

$(q_1, b, Z_0) \rightarrow (q_2, aZ_0)$

$(q_2, b, a) \rightarrow (q_2, aa)$

$(q_2, c, a) \rightarrow (q_3, \epsilon)$

$(q_3, c, a) \rightarrow (q_3, \epsilon)$

$(q_3, \epsilon, Z_0) \rightarrow (q_4, \epsilon)$

Q.6 Soln] $L = \{a^n b^m a^m b^n / m, n \geq 1\}$

Logic: ① for each a in i/p, push a in stack.

② If i/p = b, change state and push b in stack.

③ If i/p is a, change state and pop b till all i/p of a are exhaust

④ If i/p is b, change state and pop all a's.

⑤ If ip is ϵ $ TOS is Z_0, final state is reached.

Transitions:

$(q_0, a\ Z_0) \rightarrow (q_0, aZ_0)$

$(q_0, a\ a) \rightarrow (q_0, aa)$

$(q_0, b\ a) \rightarrow (q_1, ba)$

$(q_1, b, b) \rightarrow (q_1, bb)$

$(q_1, a, b) \rightarrow (q_2, \epsilon)$

$(q_2, a, b) \rightarrow (q_2, \epsilon)$

$(q_2, b, a) \rightarrow (q_3, \epsilon)$

$(q_3, b, a) \rightarrow (q_3, \epsilon)$

$(q_3, \epsilon, Z_0) \rightarrow (q_4, \epsilon)$

P_n to P_f.

$P_n = (Q, \Sigma, \Gamma, \delta_N, Q_0, Z_0)$

$P_F = (Q \cup \{P_0, P_f\}, \Sigma, \Gamma \cup \{X_0\}, \delta_F, P_0, X_0, P_f)$

3 Rules

① $\delta_F(P_0, \epsilon, X_0) = (q_0, Z_0 X_0)$

② $\delta_F(q, a, \gamma) = \delta_N(q, a, \gamma)$ for all
- q in Q
- a in $\Sigma \cup \{\epsilon\}$
- γ in Γ

③ $\delta_F(q, \epsilon, X_0) = (P_f, \epsilon)$ for all q in Q

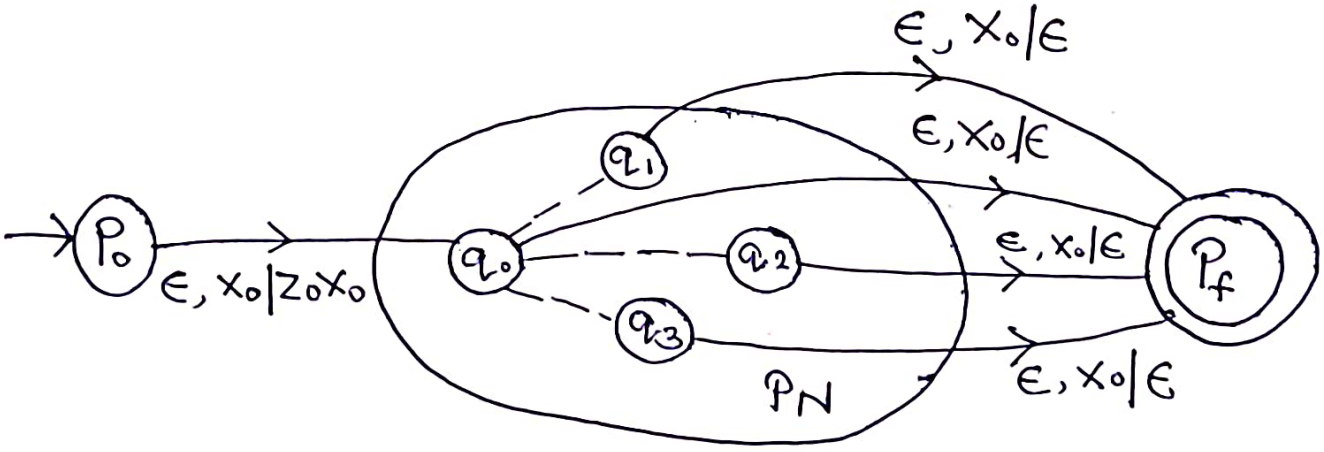


diagram to convert P_N into P_F.

Q.7] Design a P_N which accepts all strings of 0s & 1s where no. of 0 = no. of 1s. Also draw eq. P_F

Soln] Logic: ① Push the 1st symbol in TOS.

② If TOS == symbol, push in stack.

③ If next symbol not equal to TOS, then pop.

④ If TOS contains z_0 and buffer is empty - the string is valid.

Transitions:

$\delta(q_0, 0, z_0) = (q_0, 0z_0)$

$\delta(q_0, 1, z_0) = (q_0, 1z_0)$

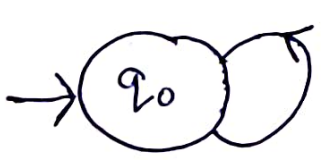
$\delta(q_0, 0, 0) = (q_0, 00)$

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

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

$\delta(q_0, 1, 1) = (q_0, 11)$

$\delta(q_0, \epsilon, z_0) = (q_0, \epsilon)$



0, z0 | 0z0
1, z0 | 1z0
0, 0 | 00
0, 1 | ϵ
1, 0 | ϵ
1, 1 | 11
ϵ, z0 | ϵ

This is P_n

now, equivalent P_f →

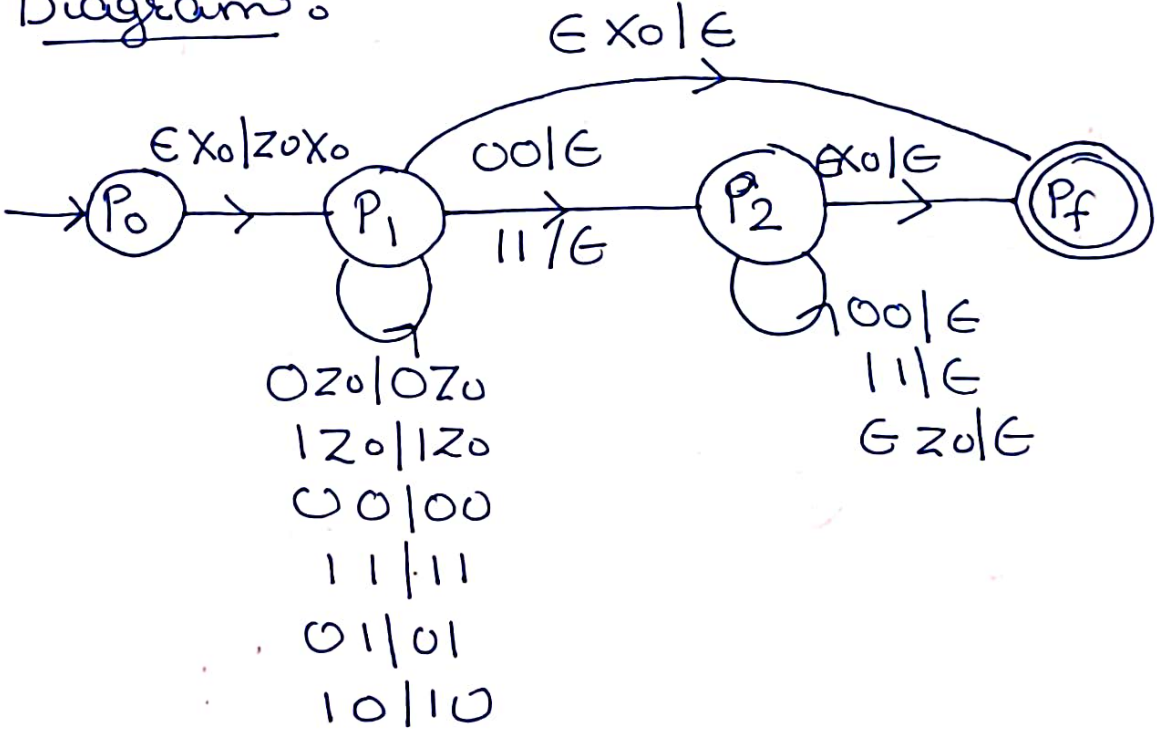
0,Z0|0Z0 0,1|ϵ
1,Z0|1Z0 1,1|11
0,0|00 ϵ,Z0|ϵ
1,0|ϵ

Q 8] Draw P_f for ↓

$L = (WW^R / W \in (0|1)^*)$

Soln] Transitions on page (59)

Diagram:



0Z0|0Z0
1Z0|1Z0
00|00
11|11
01|01
10|10

00|ϵ
11|ϵ
ϵZ0|ϵ

⇨ P_F to P_N

$P_F = (Q, \Sigma, \delta_F, \Gamma, Q_0, Z_0, F)$

$P_N = (Q \cup \{P_0, P_f\}, \Sigma, F \cup \{X_0\}, \delta_N, P_0, X_0)$

▷ 3 Rules ◁

① $\delta_N(P_0, \epsilon, X_0) = (q_0, Z_0 X_0)$

② $\delta_N(q, a, \gamma) = \delta_F(q, a, \gamma)$ for all ↓

- q in Q
- a in Σ
- γ in Γ

③ $\delta_N(q_F, \epsilon, \Gamma) = (P, \epsilon)$

$\delta_N(P, \epsilon, \Gamma) = (P, \epsilon)$

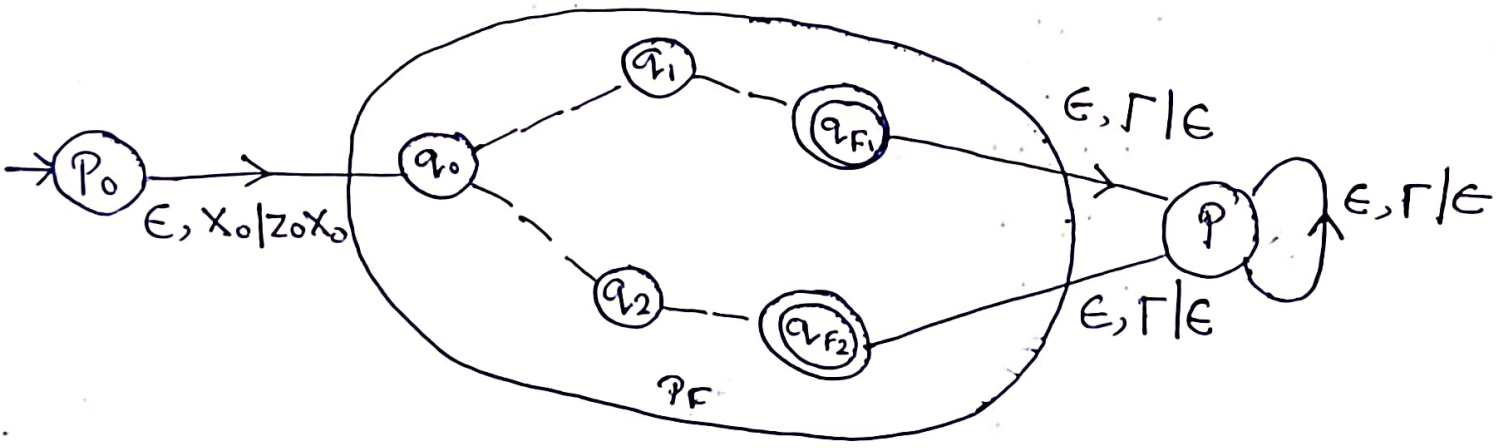


diagram to convert P_F to P_N

Q.9] $P_F = (\{q_0, q_1\}, \{0,1\}, \{X, Y, Z\}, \delta, q_0, Z, \{q_1\})$ convert into P_N.

1) $\delta(q_0, 0, Z) = (q_1, Z)$
2) $\delta(q_0, 1, Z) = (q_0, XZ)$
3) $\delta(q_0, 0, X) = (q_0, \epsilon)$
4) $\delta(q_1, 1, X) = (q_0, XX)$
5) $\delta(q_1, 0, Z) = (q_1, YZ)$
6) $\delta(q_1, 1, Z) = (q_0, Z)$
7) $\delta(q_1, 0, Y) = (q_1, YY)$
8) $\delta(q_1, 0, Y) = (q_1, \epsilon)$

Soln] $P_N = (\{P_0, q_0, q_1, P\}, \{0,1\}, \{X_0, X, Y, Z\}, \delta_N, P_0, X_0)$

$\delta_N(P_0, \epsilon, X_0) = (q_0, ZX_0)$
$\delta_N(q_0, 0, Z) = (q_1, Z)$
$\delta_N(q_0, 1, Z) = (q_0, XZ)$
$\delta_N(q_0, 0, X) = (q_0, \epsilon)$
$\delta_N(q_1, 1, X) = (q_0, XX)$
$\delta_N(q_1, 0, X) = (q_1, YZ)$
$\delta_N(q_1, 1, Z) = (q_0, Z)$
$\delta_N(q_1, 0, Y) = (q_1, YY)$
$\delta_N(q_1, 0, Y) = (q_1, \epsilon)$

$\delta_N(q_1, \epsilon, X_0) = (P, \epsilon)$
$\delta_N(q, \epsilon, X) = (P, \epsilon)$
$\delta_N(q, \epsilon, Y) = (P, \epsilon)$
$\delta_N(q, \epsilon, Z) = (p, \epsilon)$

$\delta_N(P, \epsilon, X_0) = (P, \epsilon)$
$\delta_N(P, \epsilon, X) = (P, \epsilon)$
$\delta_N(P, \epsilon, Y) = (P, \epsilon)$
$\delta_N(P, \epsilon, Z) = (P, \epsilon)$

Total 17 transitions

⇒ CFG to PDA

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

$$P_N = (\{q\}, T, TUV, \delta, q, S)$$

① For every NT symbol in A:

$$\delta(q, \epsilon, A) = (q, B) \mid A \to B \text{ in } P$$

② For every Terminal symbol "a":

$$\delta(q, a, a) = (q, \epsilon)$$

Q.10] $E \to E+E \mid E*E \mid (E) \mid I$

$I \to Ia \mid Ib \mid I0 \mid I1 \mid a \mid b$

Note → Whenever we convert grammar into PDA, PDA will have 1 state

Sol^n] $P_N = (\{q\}, \{+, *, (,), a, b, 0, 1\},$

$\{+, *, (,), a, b, 0, 1, E, I\}, \delta, q, E)$

↑ start symbol of stack

↳ Start state

Now, Adding transitions:

→ from rule ①

$$\delta(q, \epsilon, E) = \{(q, E*E)\ (q, E+E)\ (q, (E))\ (q, I)\}$$

$$\delta(q, \epsilon, I) = \{(q, Ia)\ (q, Ib)\ (q, I0)\ (q, I1)\ (q, a)\ (q, b)\}$$

→ from rule (2)

$\delta(q, +, +) = (q, \epsilon)$

$\delta(q, *, *) = (q, *\epsilon)$

$\delta(q, (, () = (q, \epsilon)$

$\delta(q,),)) = (q, \epsilon)$

$\delta(q, a, a) = (q, \epsilon)$

$\delta(q, b, b) = (q, \epsilon)$

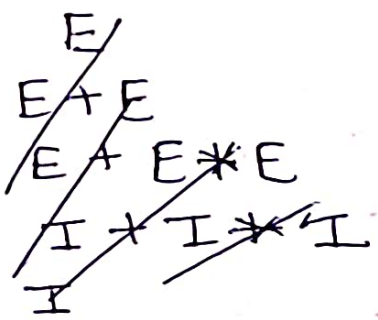
$\delta(q, 0, 0) = (q, \epsilon)$

$\delta(q, 1, 1) = (q, \epsilon)$

↳ Checking/verifying some solution :→

Ex a1+b1*b0

∴ by leftmost derivation :



E
E+E
I+E
I1+E
a1+E
a1+E*E
a1+I*E
a1+I1*E
a1+b1*I
a1+b1*I0
a1+b1*b0 → Req. string

Now, from PDA, we get ↴

(Next page)

$(q, a1+b1*b0, E) \vdash (q, a1+b1*b0, E+E)$

$\vdash (q, a1+b1*b0, I+E) \vdash (q, a1+b1*b0, I1+E)$

$\vdash (q, a1+b1*b0, a1+E) \xrightarrow{\text{Pop (a,a)}} \vdash (q, a1+b1*b0, a1+E*E)$

$\vdash (q, a1+b1*b0, a1+I*E) \vdash (q, a1+b1*b0, a1+I1*E)$

$\vdash (q, a1+b1*b0, a1+b1*E) \vdash (q, a1+b1*b0, a1+b1*I)$

$\vdash (q, a1+b1*b0, a1+b1*I0) \vdash (q, a1+b1*b0, a1+b1*b0)$

$\vdash (q, \epsilon)$ ✗ Again

Note → pop as soon as you get a T symbol

Buffer	State	
a1+b1 * b0	E	
a1+b1 * b0	TOS → E+E	
a1+b1 * b0	I+E	
a1+b1 * b0	I1+E	
a1+b1 * b0	a1+E	← Terminal symbol ∴ (POP)
1+b1 * b0	1+E	
+b1 * b0	+E	
b1 * b0	E	
b1 * b0	E*E	
b1 * b0	I1*E	
b1 * b0	b1*E	
1 * b0	1*E	
* b0	*E	
b0	E	
b0	I0	
0	b0	
ϵ	0	
	ϵ	≈ Valid string

At any pt. if TOS is terminal & does not match to the buffer, then PDA will declare the string invalid

Ex
a 1
Invalid

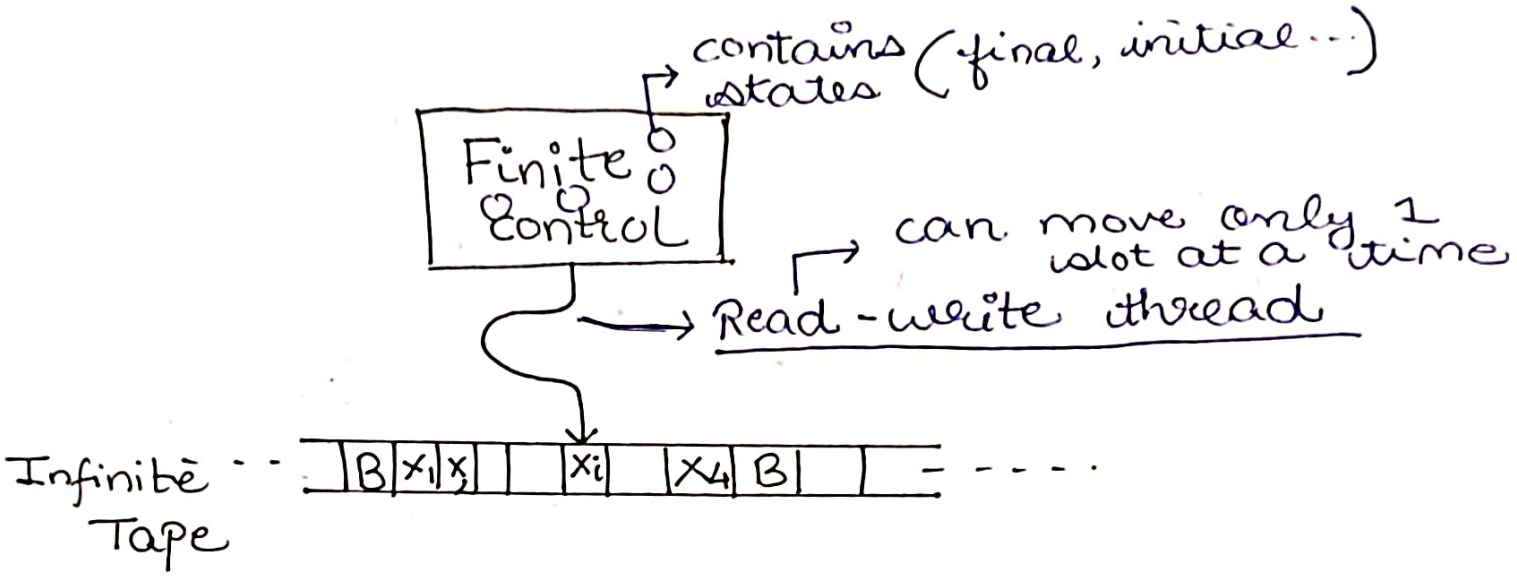
Q.11] Design PDA for the foll. and also check for validity of the given string.

Q.11.1] $S \rightarrow aSa \mid bSb \mid a$ (ababa)

Q.11.2] $S \rightarrow aAA$
$A \rightarrow aS \mid bS \mid a$ (ababa)

UNIT - 5

TURING MACHINE



▷ Mathematical definition:

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

$Q \rightarrow$ Finite set of states in TM.

$\Sigma \rightarrow$ I/P symbols.

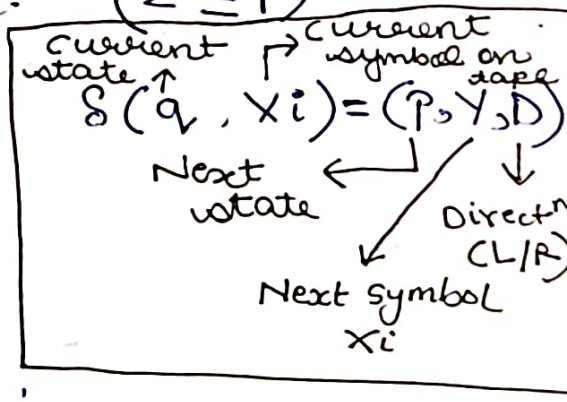
$\Gamma \rightarrow$ Set of tape symbol. $(\Sigma \subseteq \Gamma)$

$\delta \rightarrow$ Transition function

$q_0 \rightarrow$ Initial state of TM.

$B \rightarrow$ Blank symbol.

$F \rightarrow$ Set of final states.



Q.1] @ Design a turing machine which accepts lang. consisting of all palindromes of 0 and 1.

Soln] Transition table

δ	0	1	X (final)	Y (Halt)	B (for ∈)
→ q_0	(q_1, X, R)	(q_2, Y, R)	(q_f, X, H)	(q_f, Y, H)	(q_5, B, H)
q_1	$(q_1, 0, R)$	$(q_2, 1, R)$	(q_3, X, L)	(q_3, Y, L)	(q_3, B, L)
q_2	$(q_2, 0, R)$	$(q_2, 1, R)$	(q_4, X, L)	(q_4, Y, L)	(q_4, B, L)
q_3	(q_5, X, L)	—	(q_f, X, H) ↳ for odd string	—	—
q_4	—	(q_5, Y, L)	—	(q_f, Y, H)	—
q_5	$(q_5, 0, L)$	$(q_5, 1, L)$	(q_0, X, R)	(q_0, Y, R)	— ↳ Invalid

(b) Write all ID for 00110

↳ $q_0 00110 \vdash X q_1 0110 \vdash X0 q_1 110 \vdash X01 q_1 10$

$\vdash X011 q_1 0 \vdash X0110 q_1 \vdash X01100 q_1 B$

$\vdash X0110 q_3 0B \vdash X011 q_5 0XB \vdash X01 q_5 10XB$

$\vdash X0 q_5 110X \vdash X q_5 0110X \vdash q_5 X0110X$

$\vdash X q_0 0110X \vdash XX q_1 110X \vdash XX1 q_1 10X$

$\vdash XX11 q_1 0X \vdash XX110 q_1 X \vdash XX11 q_3 0X \vdash$

$XX1 q_5 1XX \vdash XX q_5 11XX \vdash X q_5 X11XX$

$\vdash XX q_0 11XX \vdash XXY q_2 1XX \vdash XXY1 q_2 XX$

$\vdash XXY q_4 1XX \vdash XX q_5 YYXX \vdash XX Y q_0 YXX$

$\vdash XXY q_f YXX$

Q.2] $L = \{ 0^n 1^n 2^n \mid n \geq 1 \}$

⇨ Addition, subtraction and Multiplicatn using Turing Machines

Note → for subtraction ⇒

$m \dot{-} n$ → $m-n$ if $m > n$

→ 0 otherwise.

∴ subtraction cannot be negative.

PDA to CFG:

$P_N = (Q, \Sigma, \Gamma, \delta, q_0, Z_0)$

$G_1 = (V, \Sigma, P, S)$

where,

① $V = [p \times q]$ — stack symbol; states of PDA

② 2 Rules for 'P'

(i) For all states P of PDA

$S \rightarrow [q_0 Z_0 P]$

(ii) $\delta(q, a, X) = (r, Y_1 Y_2 \dots Y_k)$

$[qX r_k] \rightarrow a[rY_1 r_1][r_1 Y_2 r_2] \dots [r_{k-1} Y_k r_k]$

for every state r_k in PDA.

Q.3] $M = (\{q_0, q_1\}, \{0, 1\}, \{Z_0, X\}, \delta, q_0, Z_0)$

1) $\delta(q_0, 1, Z_0) = (q_0, XZ_0)$
2) $\delta(q_0, 1, X) = (q_0, XX)$
3) $\delta(q_0, 0, X) = (q_1, X)$
4) $\delta(q_0, \epsilon, Z_0) = (q_0, X)$
5) $\delta(q_1, 1, X) = (q_1, X)$
6) $\delta(q_1, 0, Z_0) = (q_0, Z_0)$

Soln] Step① → finding all non-terminal CFG1.

(next page)

1. $[q_0 z_0 q_0]$ A (All possible combinations)
2. $[q_0 X q_0]$ B (total & non terminal
3. $[q_0 z_0 q_1]$ C symbols)
4. $[q_0 X q_1]$ D
5. $[q_1 z_0 q_0]$ E
6. $[q_1 X q_0]$ F
7. $[q_1 z_0 q_1]$ G
8. $[q_1 X q_1]$ H

Step ② → Production rules:

1) $S \rightarrow [q_0 z_0 P]$

$S \rightarrow [q_0 z_0 q_0]$

$S \rightarrow [q_0 z_0 q_1]$

2) (i) $\delta(q_0, 1, z_0) = (q_0, X z_0)$

$[q_0 z_0 q_0] = 1\,[q_0 X q_0]\,[q_0 z_0 q_0]$

$[q_0 z_0 q_0] = 1\,[q_0 X q_1]\,[q_1 z_0 q_0]$

$[q_0 z_0 q_1] = 1\,[q_0 X q_0]\,[q_0 z_0 q_1]$

$[q_0 z_0 q_1] = 1\,[q_0 X q_1]\,[q_1 z_0 q_1]$

(ii) $\delta(q_0, 1, X) = (q_0, XX)$

$[q_0 X q_0] = 1\,[q_0 X q_0]\,[q_0 X q_0]$

$[q_0 X q_0] = 1\,[q_0 X q_1]\,[q_1 X q_0]$

$[q_0 X q_1] = 1\,[q_0 X q_0]\,[q_0 X q_1]$

$[q_0 X q_1] = 1\,[q_0 X q_1]\,[q_1 X q_1]$

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

$[q_0 X q_0] = 0 [q_1 X q_0]$

$[q_0 X q_1] = 0 [q_1 X q_1]$

iv) $\delta(q_0, \epsilon, Z_0) = (q_0, X)$

$[q_0 Z_0 q_0] = \epsilon [q_0 X q_0]$ (No need to write ϵ)

$[q_0 Z_0 q_1] = [q_0 X q_1]$

v) $\delta(q_1, 1, X) = (q_1, X)$

$[q_1, X, q_0] = 1 [q_1 X q_0]$

$[q_1 X q_1] = 1 [q_1 X q_1]$

vi) $\delta(q_1, 0, Z_0) = (q_0 Z_0)$

$[q_1 Z_0 q_0] = 0 [q_0 Z_0 q_0]$

$[q_1 Z_0 q_1] = 0 [q_0 Z_0 q_1]$

Step ③ Assign name to every non terminal.

Now, $S \rightarrow A$

$S \rightarrow C$

$A \rightarrow 1BA \mid 1DE$

$C \rightarrow 1BC \mid 1DG$

Q.4] $M = (\{q_0, q_1\}, \{0,1\}, \{0,1\}, \delta, q_0, z_0)$

$\delta(q_0 \in z_0) = (q_1 \in)$

$\delta(q_0\ 0\ z_0) = (q_0, 0z_0)$

$\delta(q_0, 0, 0) = (q_0, 00)$

$\delta(q_0, 1, 0) = (q_0, 10)$

$\delta(q_0, 1, 1) = (q_0, 11)$

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

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

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

$\delta(q_1 \in z_0) = (q_1 \in)$

Pumping Lemma for Regular Sets

↳ used to prove that given lang is not regular.

↳ Based on pigeon-hole principle

Theorem: For every string w in L with constant 'n' ; $|w| >= n$

We can break w into 3 strings, $w = xyz$, such that:

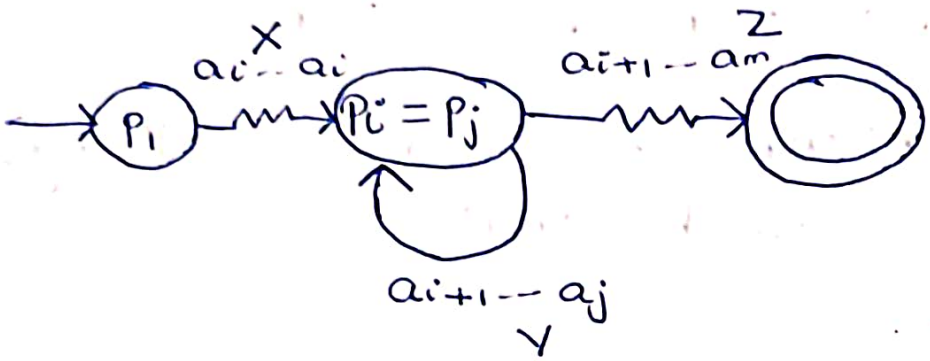
- $|y| > 0$
- $|xy| <= n$
- for all $k >= 0$, string $xy^k z$ is also in L.

Note → Pumping lemma only holds true for regular languages

L is a regular language (Proof)

M is FA for L & n is no. of states in M

Consider a string $w = a_1 a_2 a_3 \ldots a_m$ where $m > n$



Proof: Let L be a regular lang.

then L must satisfy pumping lemma

$L = a^i b^i \quad i \geq 0$

Consider a string of leng where

$w = a^n b^n$ where n is PL constant

$\therefore w = XYZ \quad |Y| > 0 \quad |XY| \leq n$ aaaa... bbbb

▸ Suppose $y = a^k$

$XY^2Z = a^{n+k} b^n \rightarrow$ This string is not in language L.

$\therefore$ It is not holding pumping Lemma.

$\therefore$ L is not regular.

Q.5] Lang L is palindrome over 0 & 1. Check if it is regular or not over pumping lemma.

Soln] Let L be a regular lang.

Let n be pumping lemma constant.

Now,

consider a string of lang L.

$$w = 0^n 1 0^n$$

As per pumping lemma, w can be divided into:

$$w = XYZ \quad \text{where } |Y| > 0, \; |XY| \leq n$$

Let $Y = 0^k$

$\therefore XY^2Z = 0^{n+k} 1 0^n$

Above string does not belong to lang. L.

$\therefore$ Lang L is not holding PL.

$\therefore$ L is not a regular language.

Q.6] $L = a^i b^j c^{2i} \quad i \geq 1$

Soln] Let L is a regular language

Let n be pumping lemma constant

Now,

consider string of lang. L.

$$w = a^n b^j c^{2n}$$

as per PL, w can be ÷ into

$$w = xyz \quad \text{where} \quad |y| > 0$$
$$|xy| \leq n$$

Let $y = a^k$

$$\therefore xy^2z = a^{k+n} b^j c^{2n}$$

above string does not belong to lang L.

Lang L is not holding PL.

$\therefore$ L is not a regular language.

Q.7] $L = a^m b^n c^{m+n}$; $m, n \geq 1$

Soln] ~~Let~~ Let L is regular lang.

let i be PL const^n.

Now,

consider string of lang L.

$$W = a^i b^n c^{i+n}$$

acc. per PL, W can be ÷ into

$W = XYZ$ where $|Y| > 0$

$|XY| \leq n$

let $Y = a^k$

$\therefore XY^2Z = a^{k+i} b^n c^{i+n}$

Above string does not belong to lang L

$\therefore$ L not holding PL.

$\therefore$ L is not a regular language.

ACKERMAN

1) $A(0, y) = y+1$

2) $A(x+1, 0) = A(x, 1)$

3) $A(x+1, y+1) = A(x, A(x+1, y))$