

(6 to 10 marks)

Theory of Computation (6 hrs).

- 50% [•] Finite Automata and Regular Language. [3 hrs]
- 20% [•] Pushdown automata and Context free language
- 0% [•] Linear bounded automata and Context sensitive language.
- 30% [•] Turing machine & Recursive Enumerable language.
- [•] Undecidability more question
- [•] Complexity theory. new language

Books:-

- (1). A, L, C } & ullman
- (2). A, L, C } & Peter Linz.
- (3). T.C.S } John. Mardin.
- (4). For theory
K.L.P. Kurva.

Definitions :- It is the mathematical study of computing machines and their capabilities.

(Q) It is the study of Automata theory and Formal languages.

↓
(C, C++, Java)

m/c ↔ language.

{ lang → program →
 ke o.s →
 m/c
 (laptop) }

English
machine.

not solved
by computer

• Ambiguity CFG No soln
 $z^n + y^n = z^m$
 $n \geq m$
x = y? no soln
to solve
using
meth algos
program

Alphabet: — (Σ)

Finite non empty set of symbols.

Ex: {a, b, c, d}.

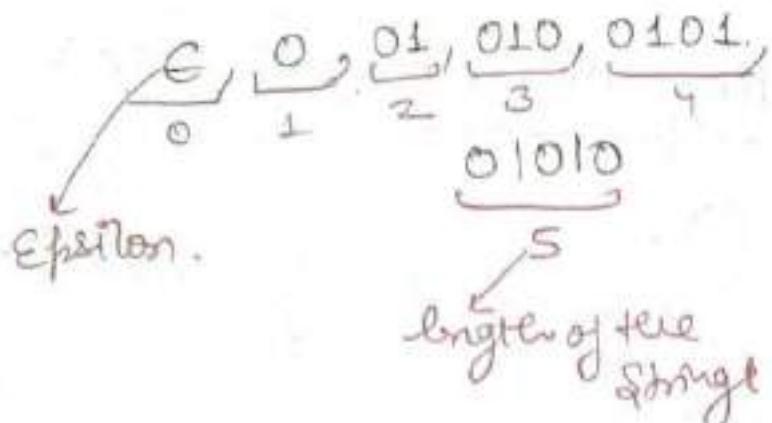
$\Sigma = \{1, 2, 3, 4\}$.

$\Sigma = \{3T, 3T, \dots, 3T\}$.

~~$\Sigma = \{\}$~~ . {set should be non-empty}.

String: — finite sequence of symbols.
over the given alphabets.

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



how many time
we are repeating
called length
of the string.

Language:- Any set of strings over the given Σ .

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

- ✓ $L = \{0, 1, 00, 10\}$ → finite lang.
- ✓ $L = \{0, 1, 00, 01, 10, 11, \dots\}$ → infinite lang.

✓ $L = \{\}$ {language because any set may be empty or non-empty are finite languages}.

Empty lang (string) over length zero

✓ $L = \{\epsilon\}$ → finite lang
{string with length zero}

$L = \{01\}$ → finite lang.

$L = \{e, 0, 1, 00, 01, 10, 11, \dots\}$ → it shows complete language all the combination of Σ^* $\{0, 1\}$

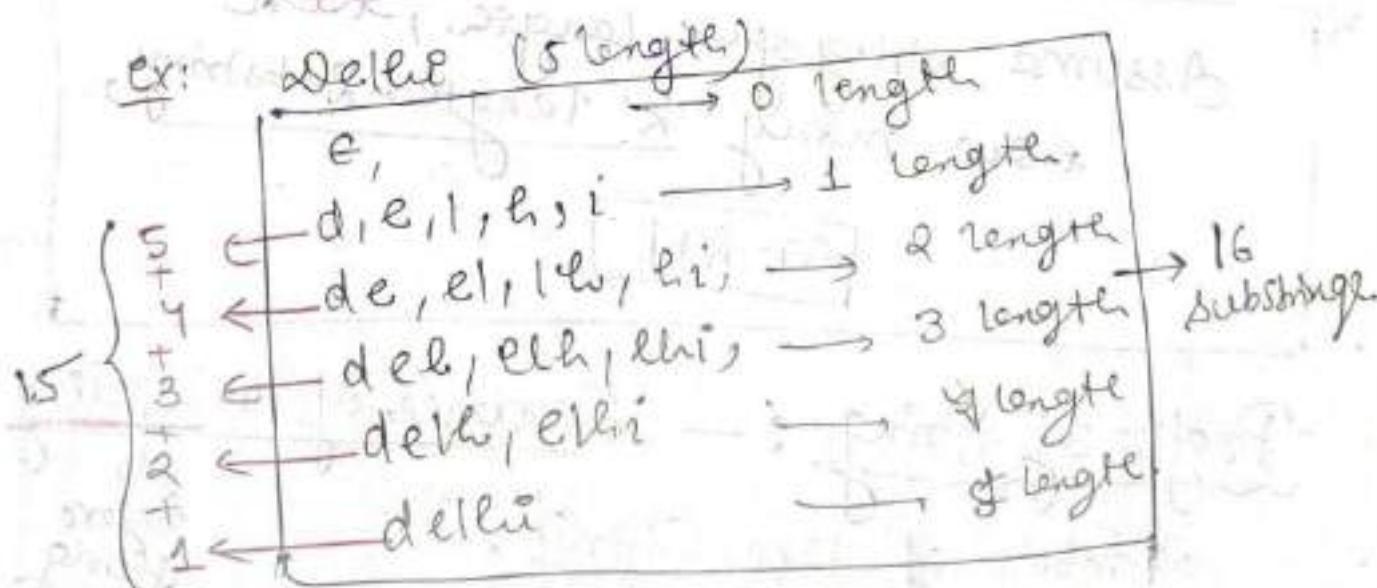
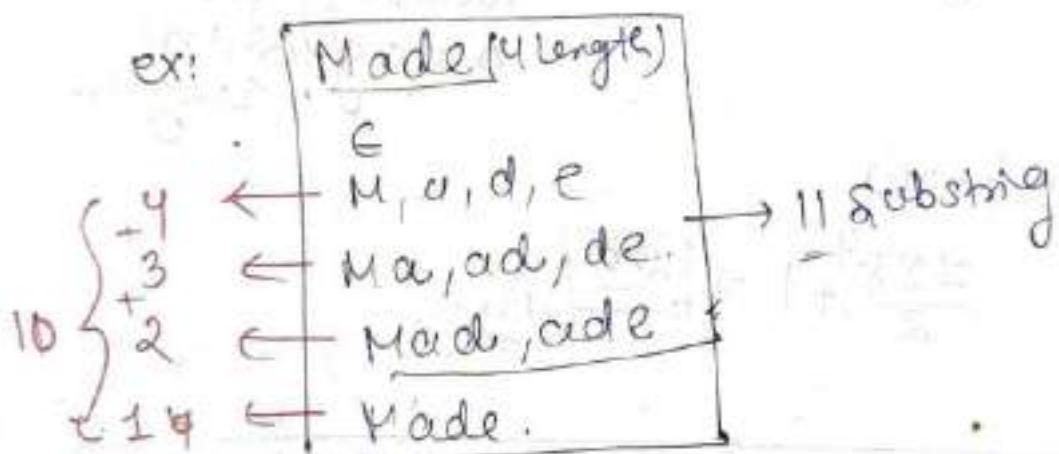
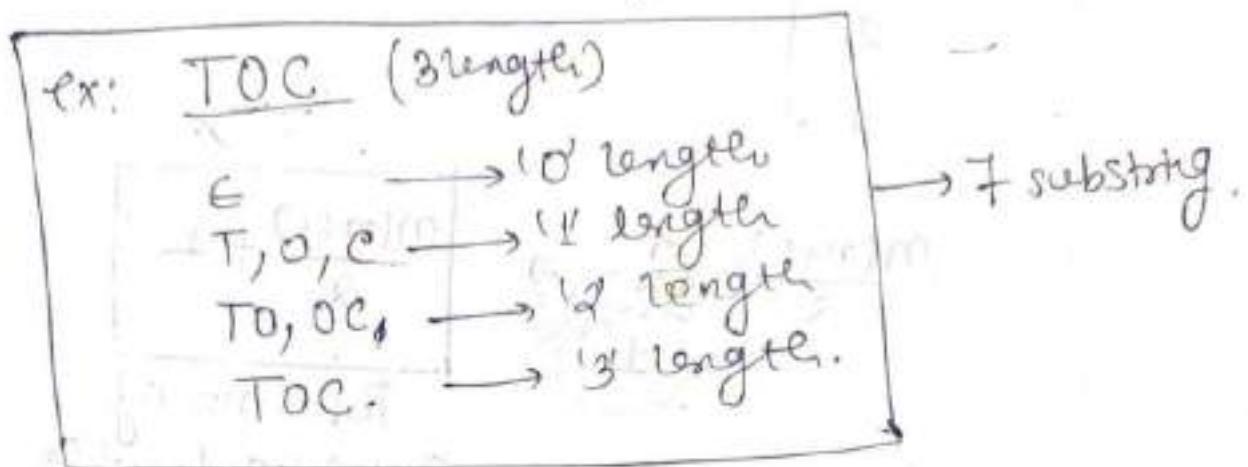
o all length string.

all 2 length string.

(4)

Substring:—
Substrings

Consecutive Sequence of Symbols over
the given String.



* abc ... m

$$\left. \begin{matrix} n \\ | \\ + \\ 3 \\ + \\ 2 \\ + \\ 1 \end{matrix} \right\} \Rightarrow \frac{n(n+1)}{2}$$

No. of substrings of different lengths (non-zero) can be formed from a character string of length n is \underline{n}^2

$$\frac{n(n+1)}{2} + 1 \quad \text{for } 1$$

$$\boxed{\frac{n(n+1)}{2} + 1}$$

Total no. of Substring possible over given n , length.

Ex: We know

$$\frac{7 \times 8}{2} + 1 \Rightarrow \boxed{29} \text{ Ans}$$

* Assume string of m length, then how many k length substring,

$$\boxed{m-k+1}$$

Prefix of String : — Sequence of leading symbols of the string before string.

(5)

Suffix of string:- Sequence of trailing symbols of the string.

Ex. of prefix:-

<u>TOC</u>
C
T
TO
TOC.

<u>Made</u>
e
M
Ma
Mad
Made.

<u>Delli</u>
e
D
de
Del
Delli
DELLI

*

abc.....n (String)



(n+1) no. of prefix.

Ex. of suffix:-

<u>TOC</u>
C
OC
TO
TOC

<u>Made</u>
e
de
ade
Made

<u>Delli</u>
i
lli
DELLI
DELLI
DELLI

A

abc.....n (String)



(n+1) no. of suffix

Proper prefix :- Exclude the original string from the prefix.

ex: TOC :- proper prefix.

E

T

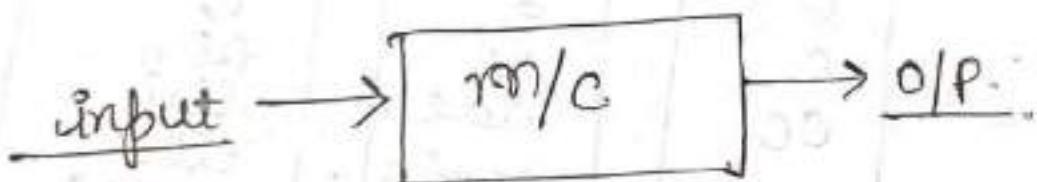
TO

~~TOC~~ (Excluded)

Made :- E, M, Ma, Mad.

Delli :- e, d, de, del, dell

Automata :- Automata is a machine that takes set of inputs & produces output by processing the input automatically. {without human interaction}



Formal language :- It is collection of strings where the strings are formed based on some conditions. {meaning of the strings not important}.

(6)

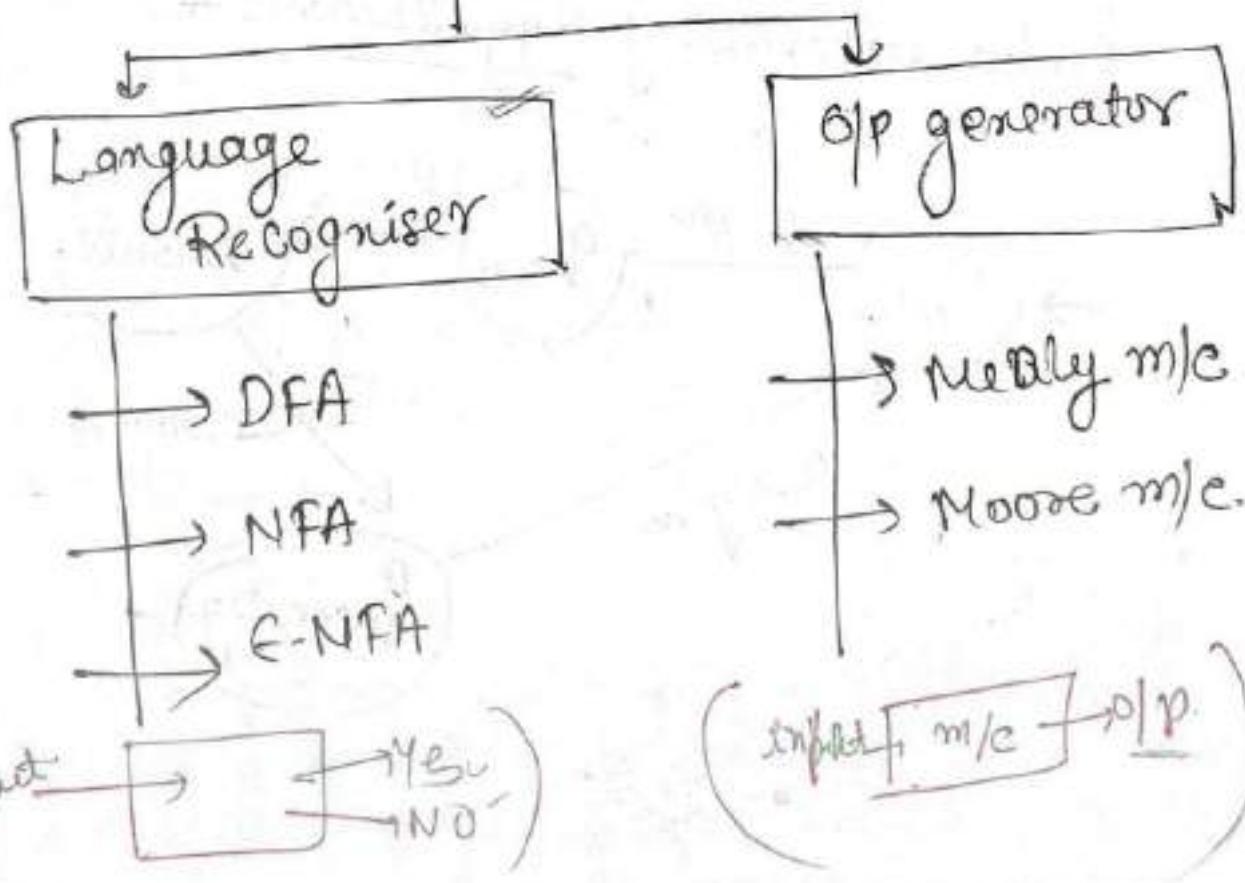
Ex: $L_1 = \{a^n b^m \mid n, m \geq 1\} = \{ab, \underline{aab}, \underline{abb}, \dots\}$
 $L_2 = \{a^n b^n \mid n \geq 1\} = \{ab, aabb, \dots\}$
 $L = \{a^n b^n c^n \mid n \geq 1\}.$

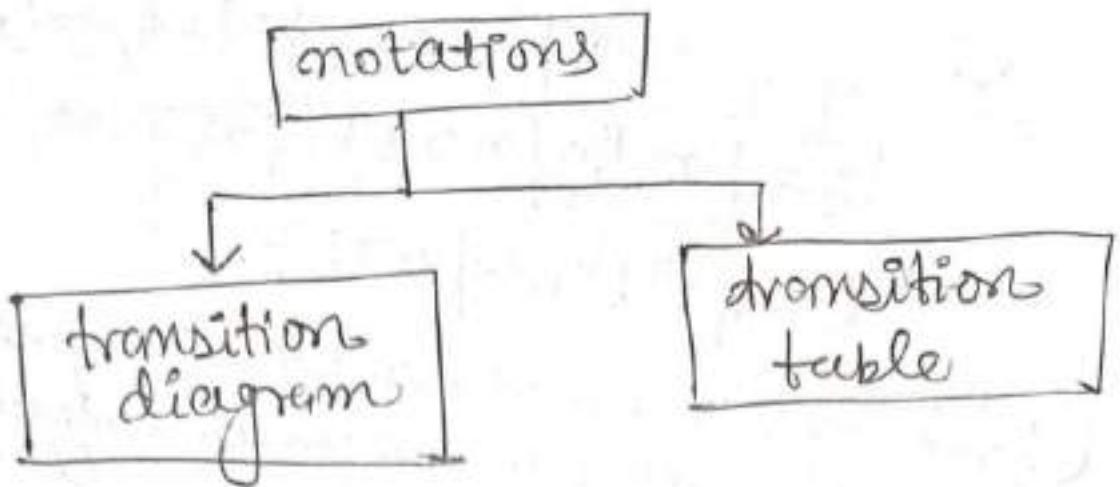
(^{formal}
language is a mathematical representation
of our original language)

Formal Finite Automata (with 3 methods)

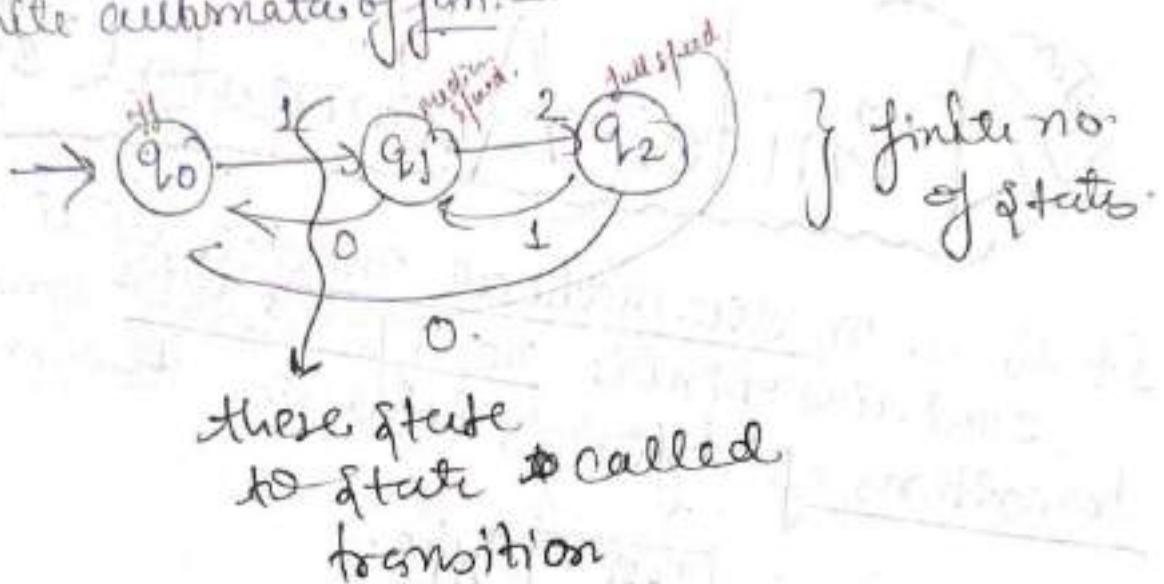
It is a mathematical model which contains finite no. of states and transitions defined between them.

F.A. Type.

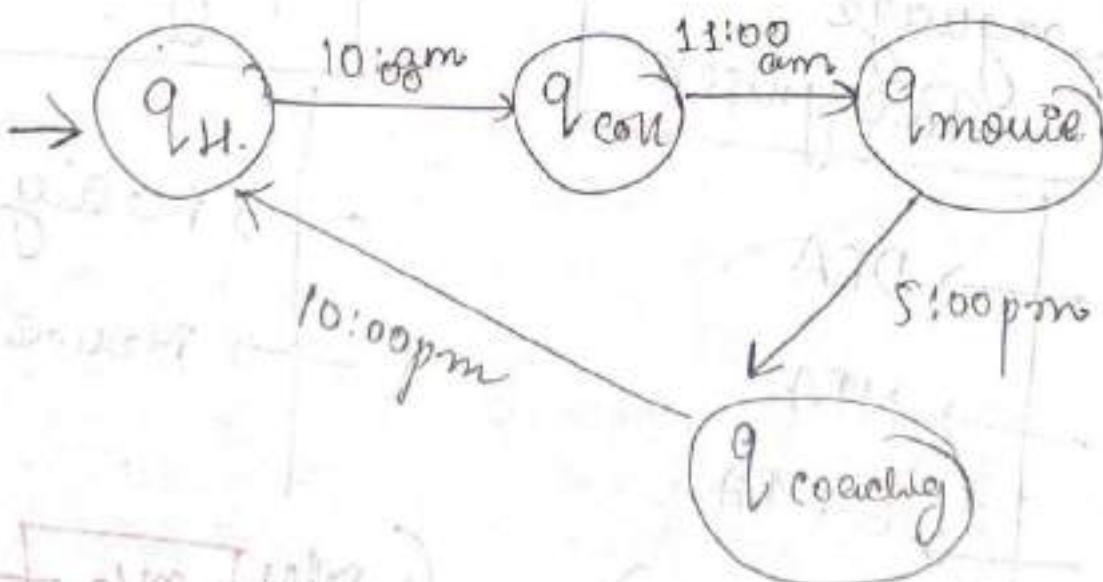




Finite automata of fm:-

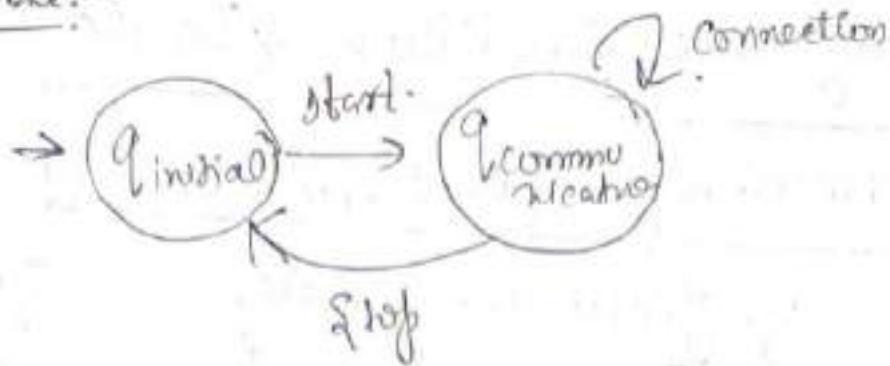


Finite automata of Engg. Student:-

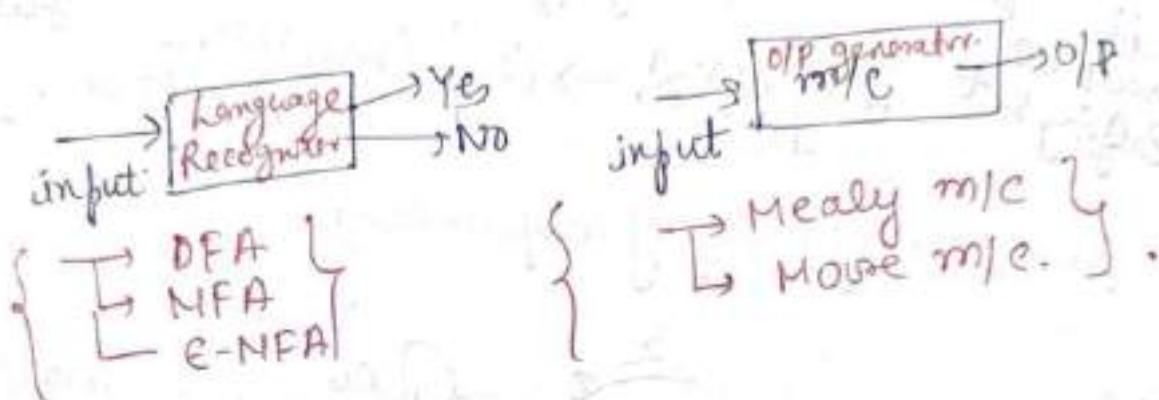


FA of mobile:-

(7)



Two ways to represent of the F.A.:-



- language recognizer finite automata gets used in the design of compilers.
- O/P generator, finite automata is used in Circuit Design Areas.

~~or~~  DFA (Deterministic Finite Automata).
means it shows exact nature.

It is a F.A. in which from each and every state on input symbol every

exactly one transition should exist.

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

Q : finite no. of states

Σ : input alphabet

q_0 : Initial state

F : Set of final states

In
only

DFA

these

final
states

should

have
been
given
that
automata
is DFA

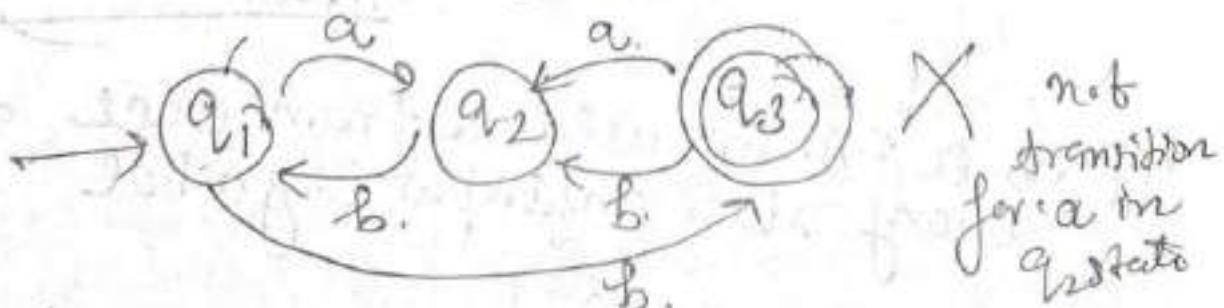
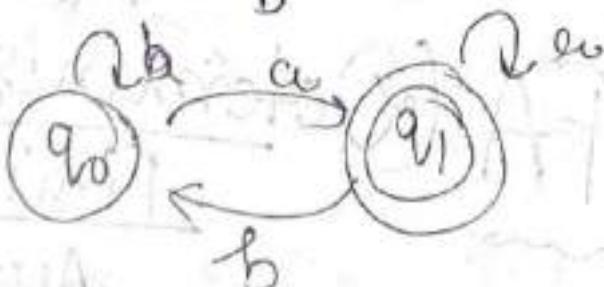
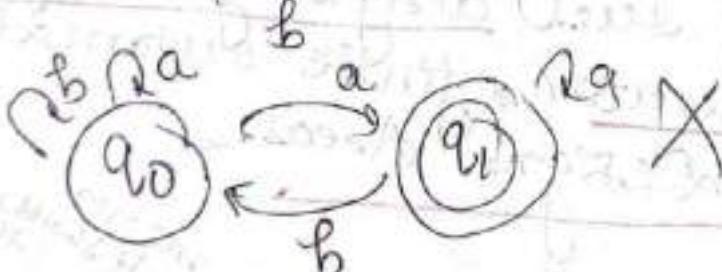
it
may be
empty set
so $\delta : [Q \times \Sigma \rightarrow Q]$ transition
function.

(DFA: - heart of theory of computation)

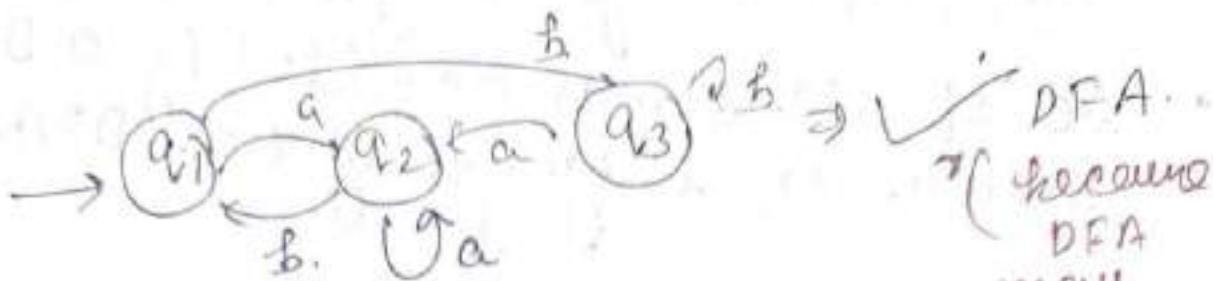
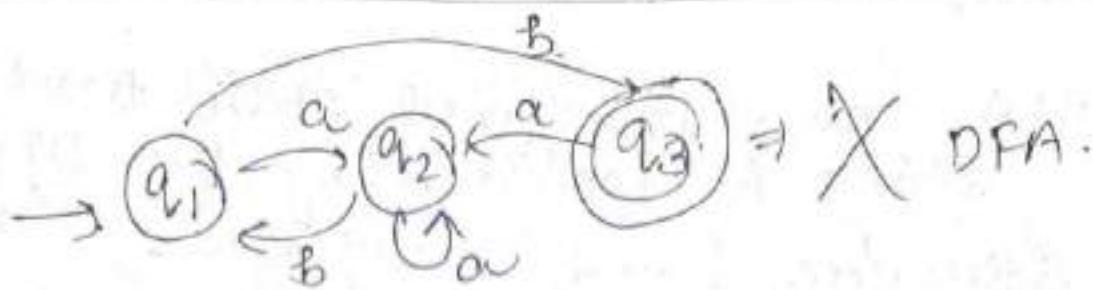
ex:



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

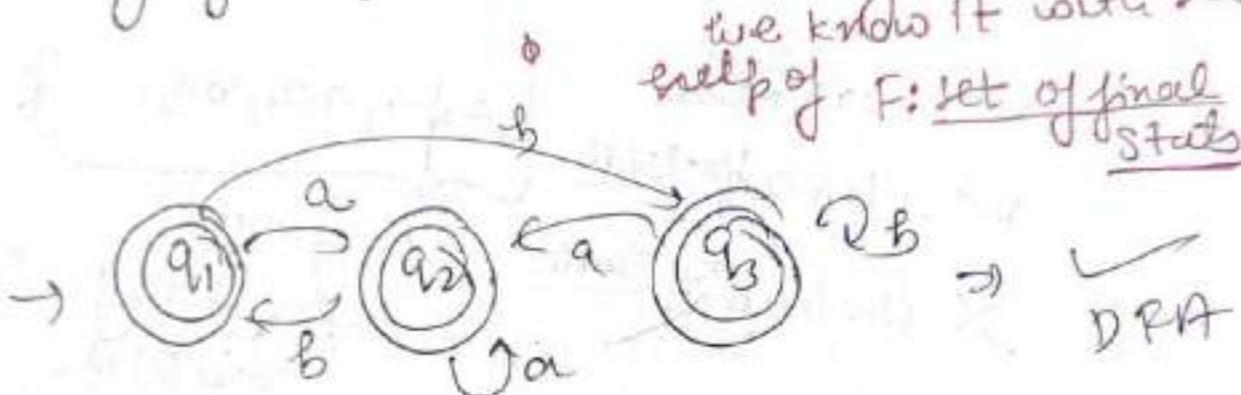


(8)



DFA without final states accepts empty language only.

(because DFA may contain any no. of final states it may be zero also we know it with the help of F: set of final states)



one transition should
 ≥ 1 \times ^{not} should
 < 1 \times not showed.

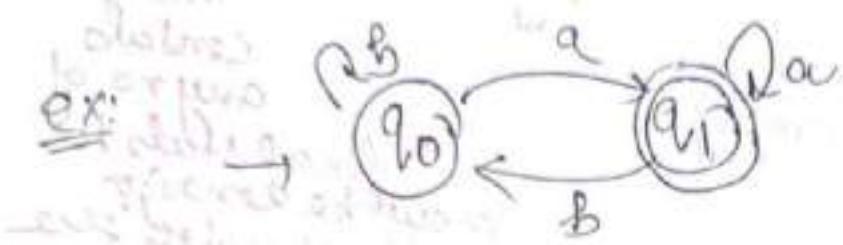
→ that's why it is Lang. recognizer.

Acceptance of String in DFA:-

By reading the complete string from left to right, end of string if the

DFA ends in final state then string is accepted by the DFA.
 Otherwise string is rejected. reject needed.

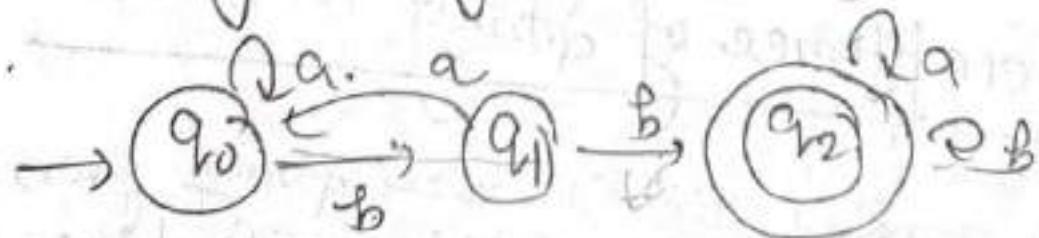
- Set of all strings accepted by a DFA known as language of the DFA.



$\checkmark ab\overset{\text{final state}}{a}$
 $\checkmark abb\overset{\text{final state}}{a}$
 $\times abb\overset{\text{not Fstate}}{b}x$

$L = \{a, ab, ba, \dots\}$
 language accepted by this DFA.
 set of all strings ended with 'a'.

Q. Identify language accepted by the following DFA:-



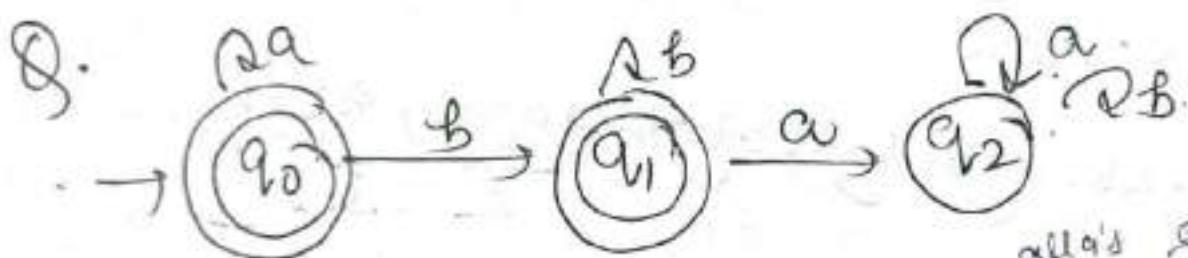
- (q)
- X(a) Starting with bb
 X(b) ending with bbt
 ✓(c) atleast 2bb
 (d) none.
- aabb x.
bbax
means only string starts with bb.

a bba
bb a

babababab
(it is accepted by automata but not by atleast 2bb)

~~the correct answer is~~ —

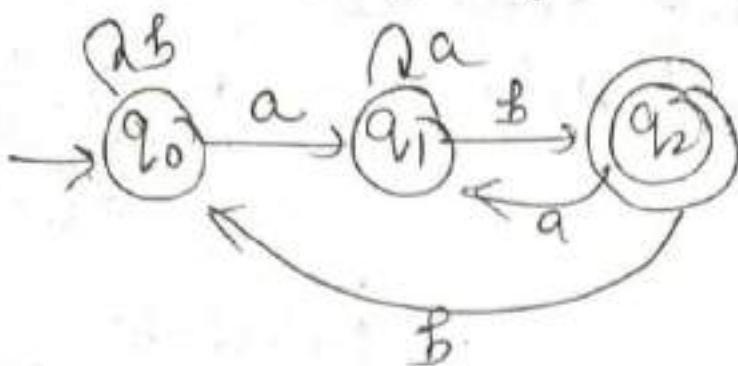
questioning bb



- X(a) $L = \{a^n b^m \mid n, m \geq 0\}$ → {e, $\overbrace{aa\dots}^{\text{all a's}}, \overbrace{bb\dots}^{\text{all b's}}$, ab...}
- X(b) $L = \{a^n b^m \mid n \geq 0, m \geq 1\}$ → min string is b not e
- X(c) $L = \{a^n b^n \mid n \geq 0\}$. → e possible but not single 'a' possible
 {e, ab, ...}
- (d) None.

* The initial state is final, e is accepted by DFA.

Q. Identify language of this DFA,



baba
babab

(a) starting with ab. abbabX.

b abX

(b) ending with ab

abb ab

(c) substring ab abX

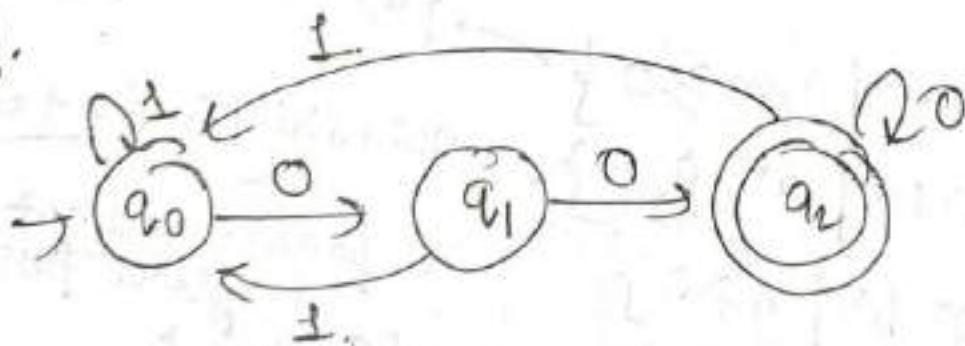
baabaa

(d) none!

bababab

$\Sigma = \{ab, aab, bab, abab, aaaaab, bbaab\}$

Q.



(a) starting with 0 100X

100101

(b) Ending with 0 → all ending with 0 and 00 are accepted

101

(c) substring 00

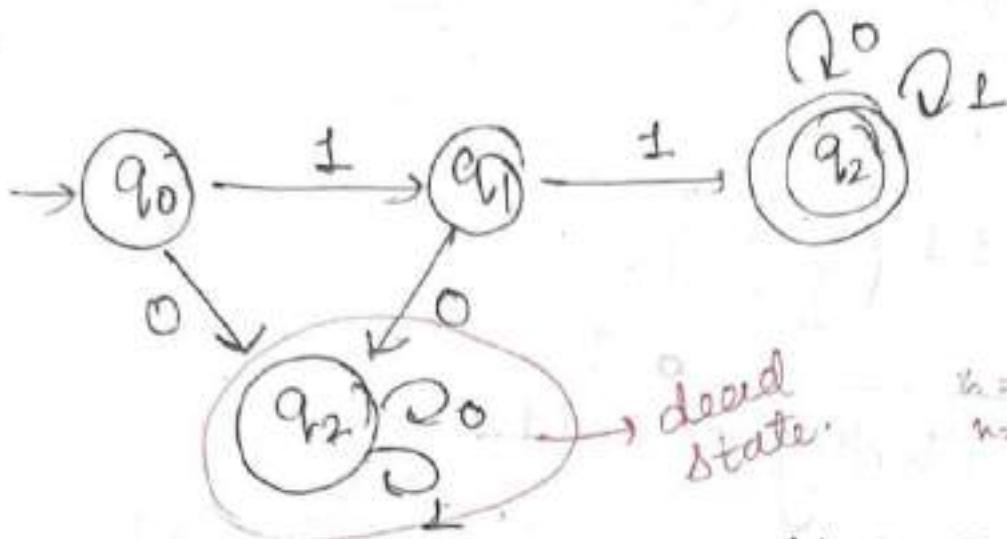
{0, 10, 00, X, X} not accepted by automata

(d) none

(C) \rightarrow 001 X.

S-S (10)

Q.



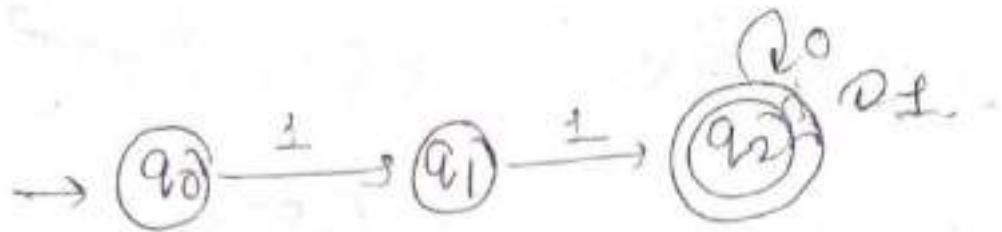
How many n length strings are accepted?

Sol:

- (a) $\{2^n \mid n \geq 0\}$
- (b) $\{2^{n-1} \mid n \geq 1\}$
- (c) $\{2^{n-2} \mid n \geq 2\}$
- (d) none.

* Dead state :- Any non final state having self loops on all input symbols known as dead state.

So, remove 1st the dead state -



$\frac{1}{1} \left(\begin{array}{c} \text{only} \\ 0's \\ \text{only} \\ 1's \end{array} \right)$

Starting with 11.

min string accepted: $\frac{n=2}{11} \rightarrow 1$

$n=3$,

11 0/1 }ⁿ⁼²

$n=4$

11 00
 01
 10
 11 } $\rightarrow 4$

$n=8$

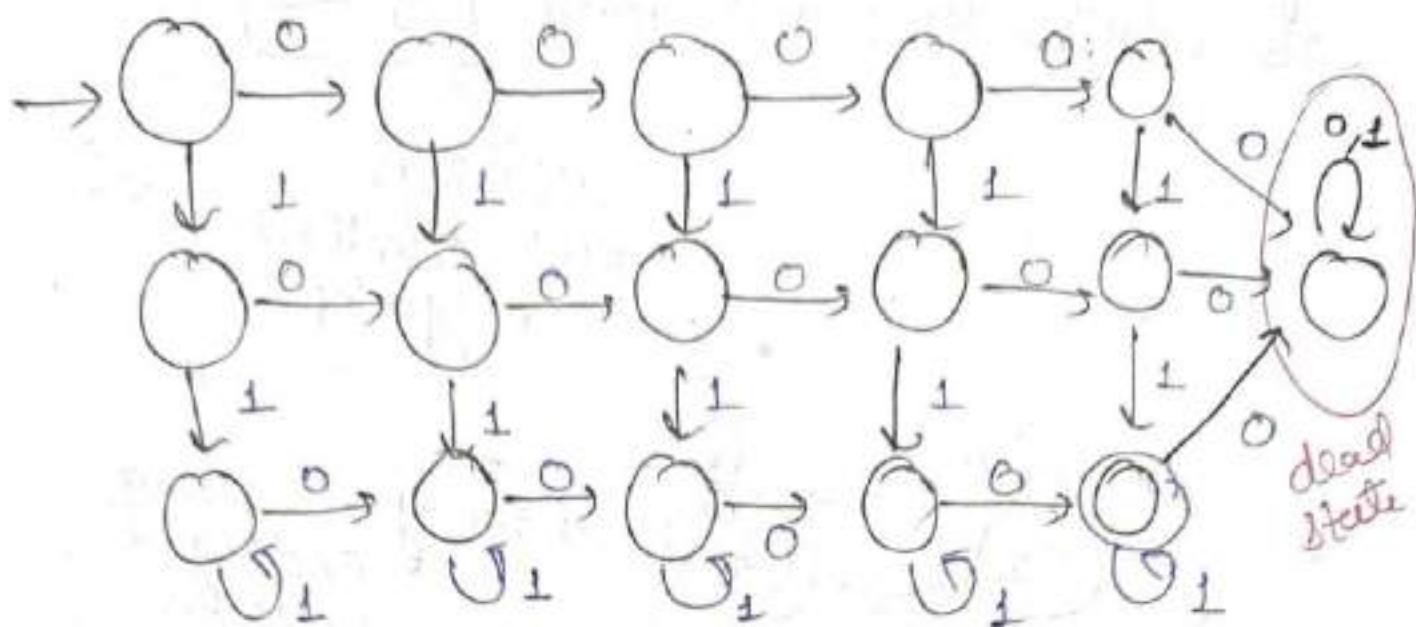
$n=5$ 11 011
 010
 111 } $\rightarrow 8$

2^{n-2}

see - 2 is for
starting with
11.

Q.

:-)



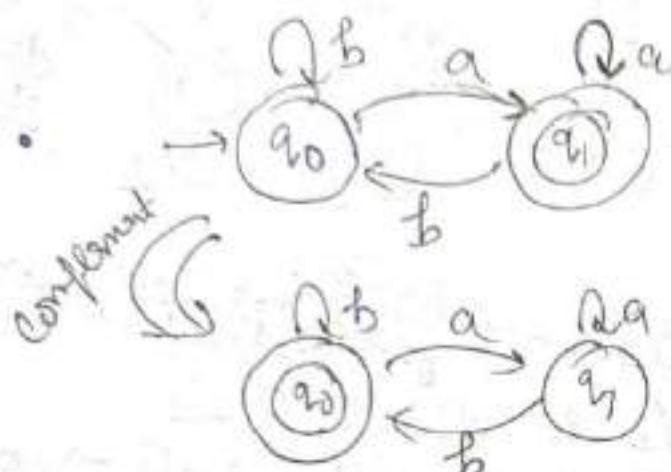
- (a) At least 4 0's → ~~00000~~ not accepted by going to dead state automata.
- (b) At least 4 0's & atleast 2 1's
- (c) Exactly 4 0's & exactly 2 1's → ~~00001111~~
- (d) Exactly 4 0's & atleast 2 1's.

{ \Rightarrow Qn } Complementing a D.F.A. :-

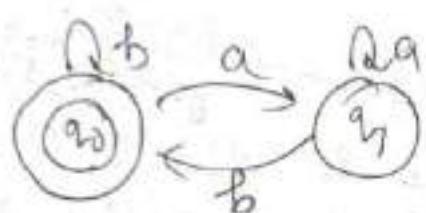
- Interchanging final and non-final state of given DFA we can get complemented DFA.
- Complimentation is possible only for DFA.

- and not possible for NFA and ENFA.
- If a language accepted by a DFA is 'L' then its complemented DFA accepts $\Sigma^* - L$

$\Sigma^* - L$ →
all strings
with complete
language.
only
DFA



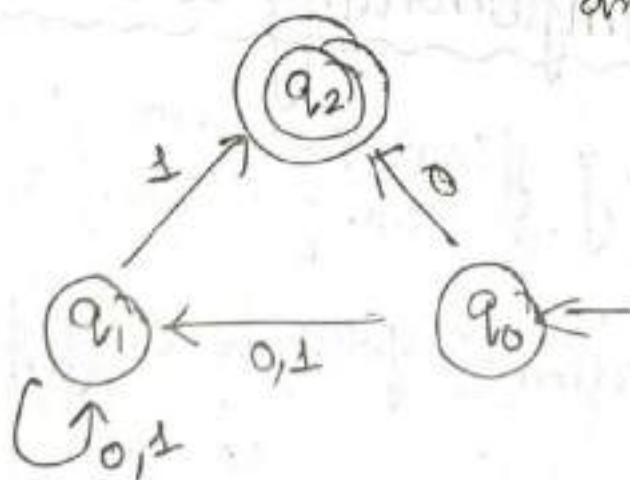
⇒ set of all string
ending
with 'a'
{ it does not accept }



⇒ set of all string
not ending
with 'a'

$(\Sigma^* - L)$
{ it also accept 'b' }

- ~~SATE~~. Construct the following automata
and language
accepted by
it is 'L'.

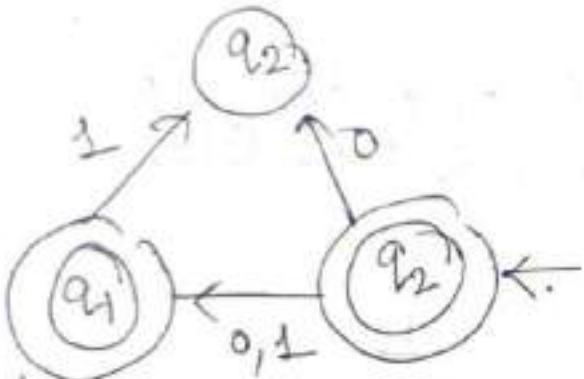


Interchange final & non-final state

of given automata then the language accepted by new automata is L_1 .
 Ques. Which of the following is true about L & L_1 .

- (a). $L_1 = L$
- (b). $\Sigma^* - L$
- (c). $\Sigma^* = L_1$
- (d). None

Soln: (a) $L_1 \neq L \Rightarrow L_1$ accept ϵ
 but L not \times

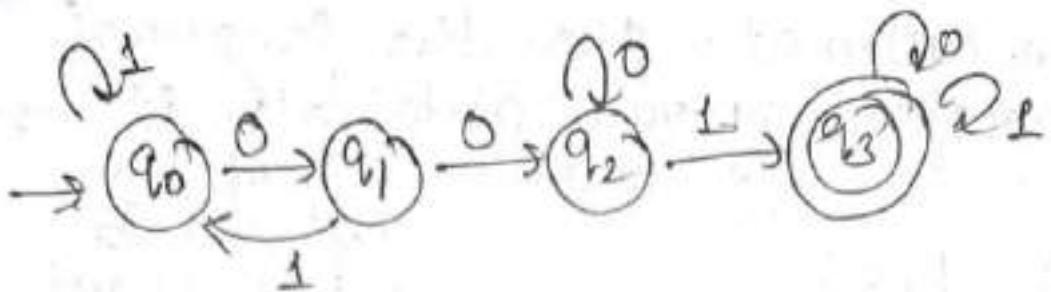


(b) $\Sigma^* - L \Rightarrow$ it work only for DFA
~~accepts~~
 not work here because the given automata is not DFA.

(c) It is not DFA because it is mention that final is non-final.
 not given complemented.

(d) $\{\epsilon, 0, 1, 00, 01, 10, 11, \dots\}$
 complete language accepted by the automata L_1 .

Q. Consider the following DFA's,



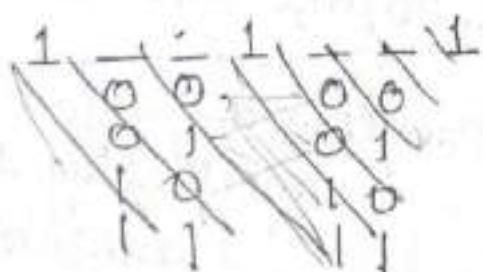
(a) what is the language accepted by this DFA?—

- ①. String starting with 001
- ②. String ending with 001
- ③. Substring 001
- ④. None.

(b) Let S: denotes 7 bit binary string in which 1st bit, 4th bit and last bits are 1.

1 — — 1 — — 1

then how many no. of strings are accepted by above DFA from S.

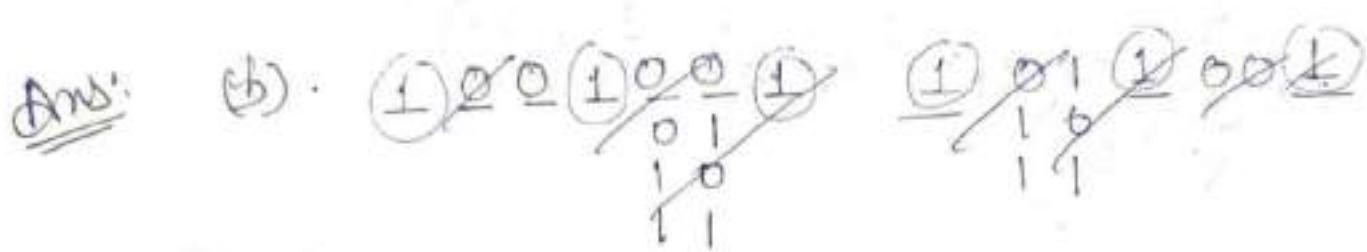


- ①. 16
- ③. 8
- ⑤. 9
- ②. 7

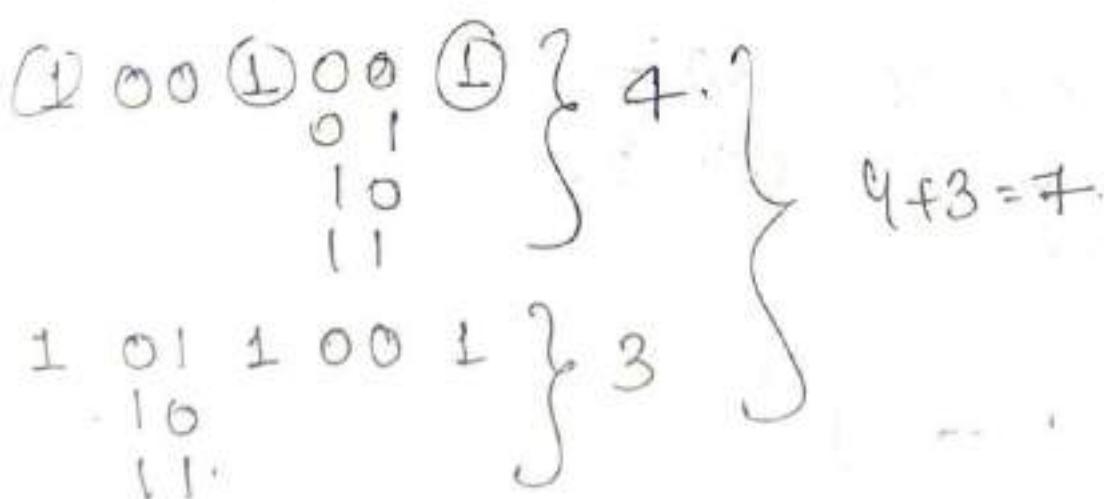
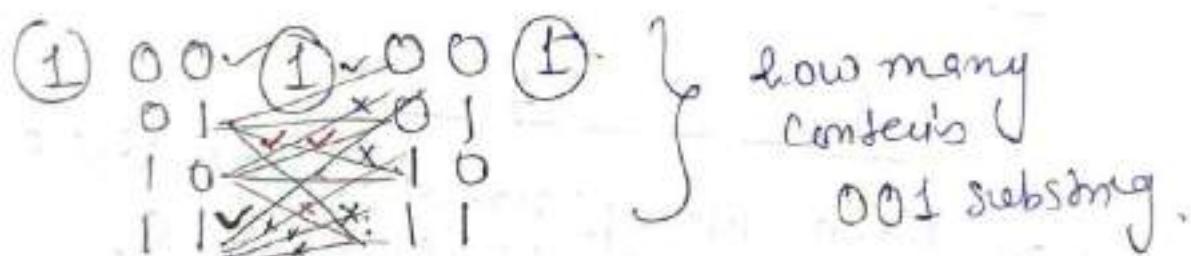
(C) Complement the above DFA then
 how many no. of strings the
 complemented DFA will accept
 from S.

(13)

- ① 16 ③ 8
- ② 4 ④ 9.



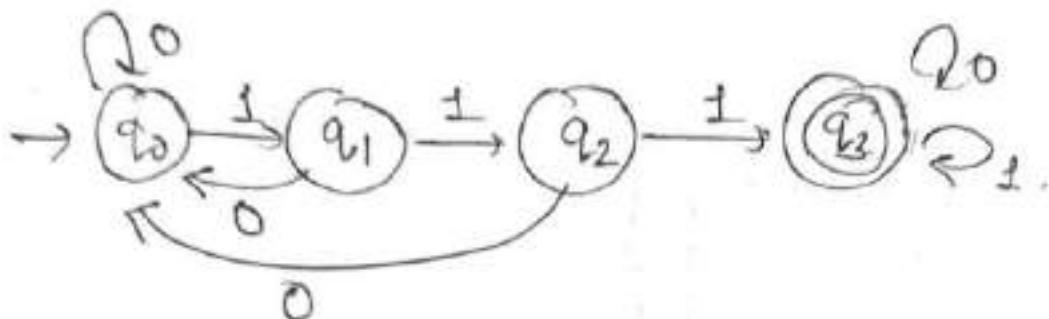
Total string = 16.



$$\begin{array}{r} 101 \quad 1 \quad 00 \quad 1 \\ 101 \quad 1 \quad 01 \quad 1 \end{array}$$

Ans(C) : $\sum^* - L \Rightarrow 16 - 8 = 7 = 9.$

Q. Construct the following DFA



- ①. Let S denotes a 6-bit binary string in which 1st bit & 4th bits are 1.

$$\underline{1} \quad \underline{\quad} \quad \underline{\quad} \quad \underline{\quad} \quad \underline{\quad}$$

then how many no. of strings the above DFA accepts from S .

- (a) 16 (b) 7
- (c) 8 (d) 9.

$$\begin{array}{r} 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad - \\ \underline{0 \quad 0 \quad 1} \\ 0 \quad 1 \quad 0 \\ 0 \quad 1 \quad 1 \\ \vdots \end{array}$$

⑧ Complement the above DFA then how many no. of strings the complemented DFA accepts from S

(19)

- (A) 16
- (B) 7
- ~~(C) 8~~
- (D) 9.

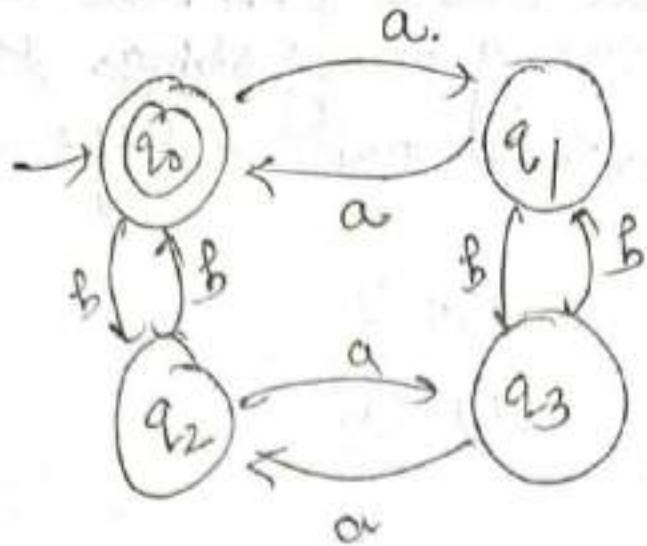
Ans:

①	1	-	-	1	-
	x	0	0	0	
	x	0	0	1	1
	x	0	1	0	
		0	1	1	
		1	0	0	
		1	0	1	
		1	1	0	
		1	1	1	

⇒ (8)

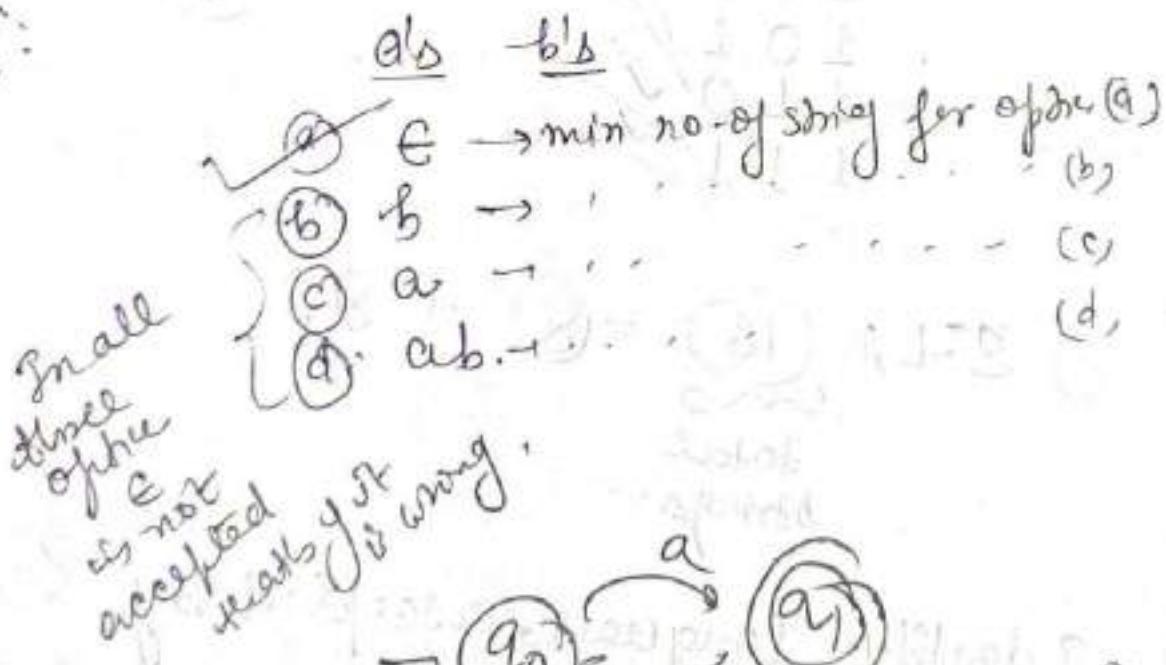
② $\Sigma^* - L = 16 - 8 \Rightarrow 8$
 total strings.

Q. Identify language accepted by following DFA:-

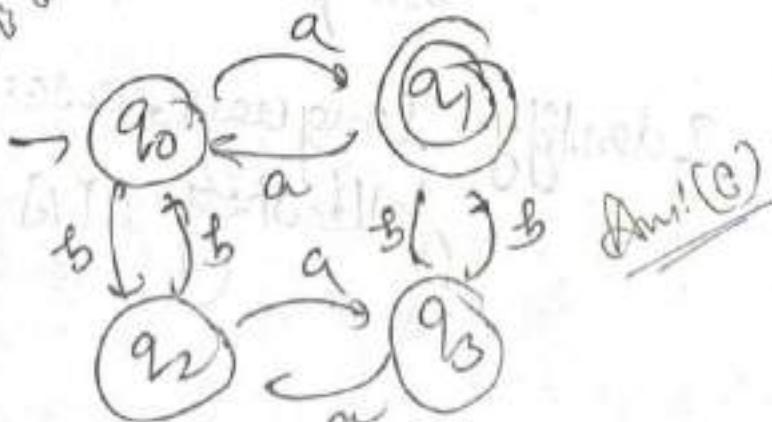


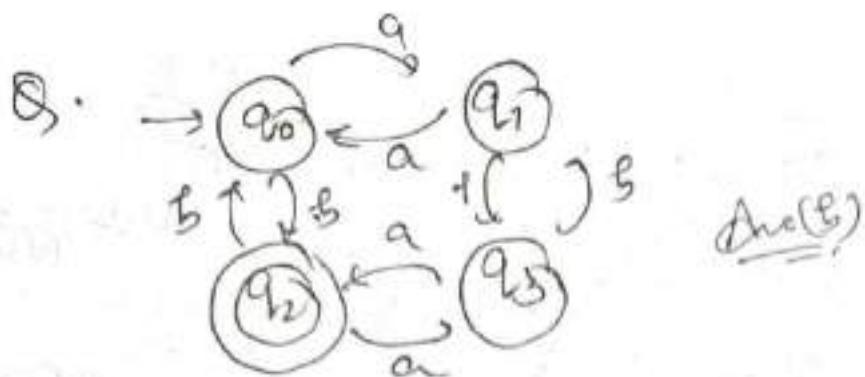
- (a) even no. of 'a's & even no. of 'b's
 (b) even no. of 'a's & odd no. of 'b's.
 (c) odd no. of 'a's & even no. of 'b's.
 (d) odd no. of 'a's & odd no. of 'b's.

Dif:



Q.

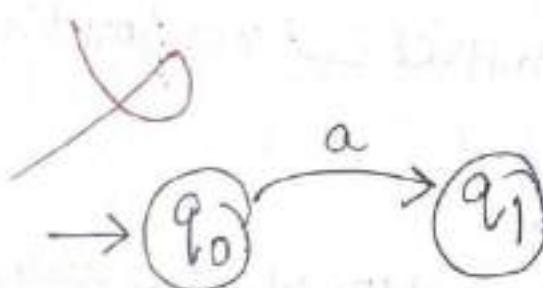




Language to DFA Construction:-

- Q. Construct the DFA for the language
 $L = \{a^n b^n | n \geq 1\}$.

\Rightarrow :
 $n=1 \Rightarrow ab$. equal no. of a's & b's
 $n=2 \Rightarrow aabb$.
 $n=3 \Rightarrow aaabbb$

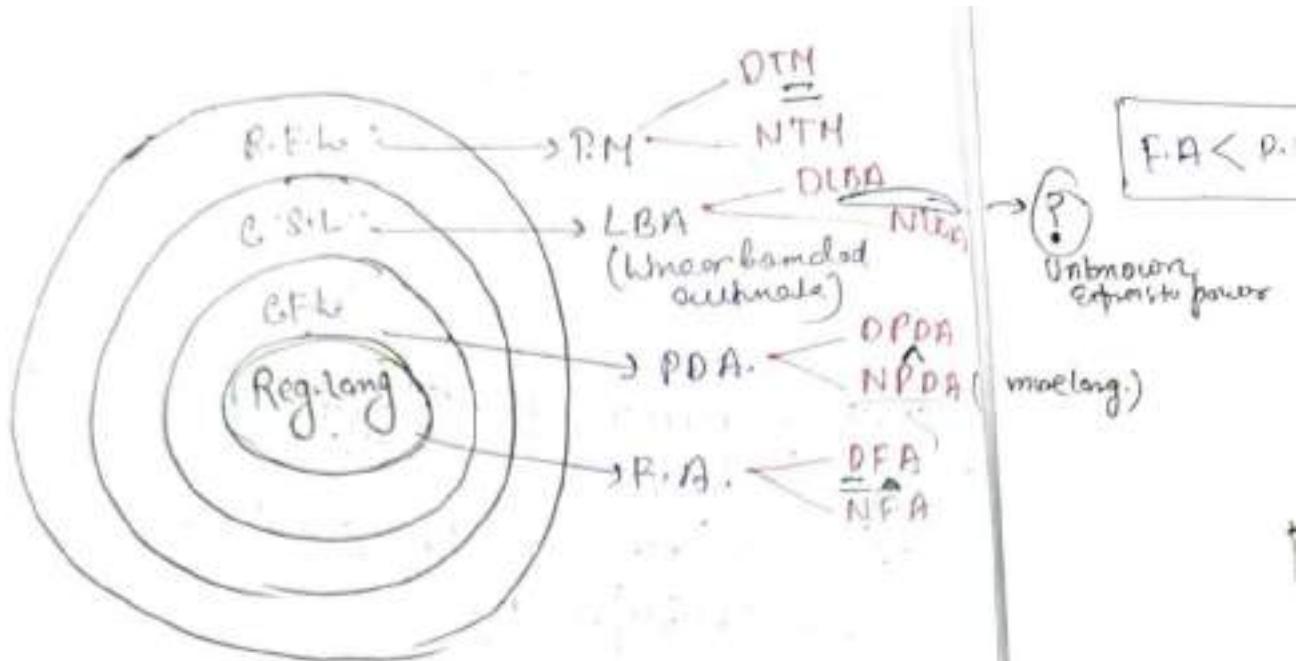


* DFA is not possible

This is Undecidable problem not possible for computers.

→ Always first check DFA is possible or not.

Phomsky hierarchy:-



✓ GFL but not Regular.

✓ R.E.L but not CFL.

✗ Regular but not CSL. (not possible)

DFA

- Due to
- same
- it is

- If the

C

- NPDA is

cm

Expressive power :— No. of languages accepted by a particular machine is known as expressive power of that machine.

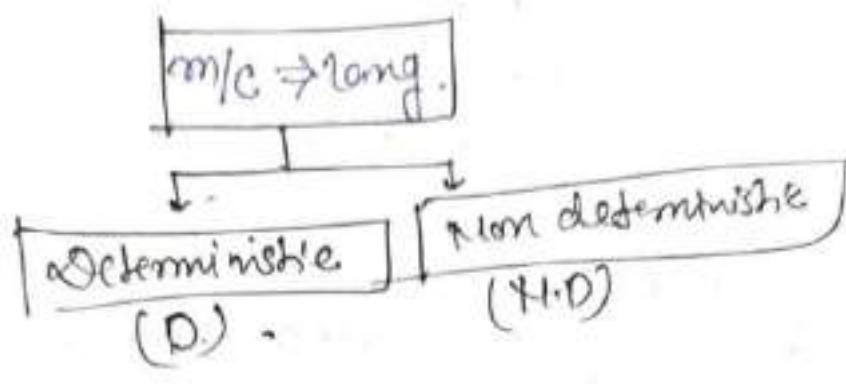
$$\boxed{\text{Reg-lang} \subset \text{CFL} \subset \text{C.S.L} \subset \text{R.E.L.}}$$

$F.A < P.D.A < L.B.A < T.M.$

→ ?
Unknown expressive power

most powerful machine.

more lang.)



$DFA = NFA$ { same power }

all the lang. that accept DFA.
must be accepted by NFA.

- The expressive power of DFA and NFA is same. hence if a language is accepted by NFA if and only if it is accepted by DFA.
- If the expressive powers are same every NFA will be converted into DFA.
(Every nondeterministic converted into deterministic)
- NPDA is more powerful than DPDA hence, no. of language accepted by NPDA is more.

- Hence every NPDA cannot be converted into DPDA!
- The expressive power of DTM and NTM is same.
Hence, every NTM will be converted into DTM.
- The expressive power of DLBA & NDLBA is unknown.(?).

Default:-

If we have given the ^{auto language} by default we construct \Rightarrow

Default FA :- DFA

Default PDA:- NPDA

Default TM:- DTM

The default finite automata is DFA,
the default PDA is NPDA, the
default TM is DTM.

(If expressive power is same we always
go for deterministic one).

~~Ans~~ Let D_f and D_p are no. of
language accepted by

DFA and DPA respectively.

(17)

Let N_f and N_p are no. of language accepted by NFA & NPDA respectively then which of the following is true:-

X ①. $D_f = N_f$
 $D_p = N_p$

X ②. $D_f \supset N_f$ X
 $D_p \subset N_p$



X ③. $D_f = N_f$
 $D_p \supset N_p$

④. none

$$D_f \supset D_p$$

$$N_p \supset D_p$$

Q. Which of the following statement is false:-

- ①. Every NFA can be converted into DFA.
②. Every NTM can be converted into DTM.
③. Every NPDA can be converted into DPDA.
④. None.

Why FA is needed ? because there is also T.M ;

With T.M no need to do accept FA there is not that we can't

need F.A but due to complexity we need T.M.

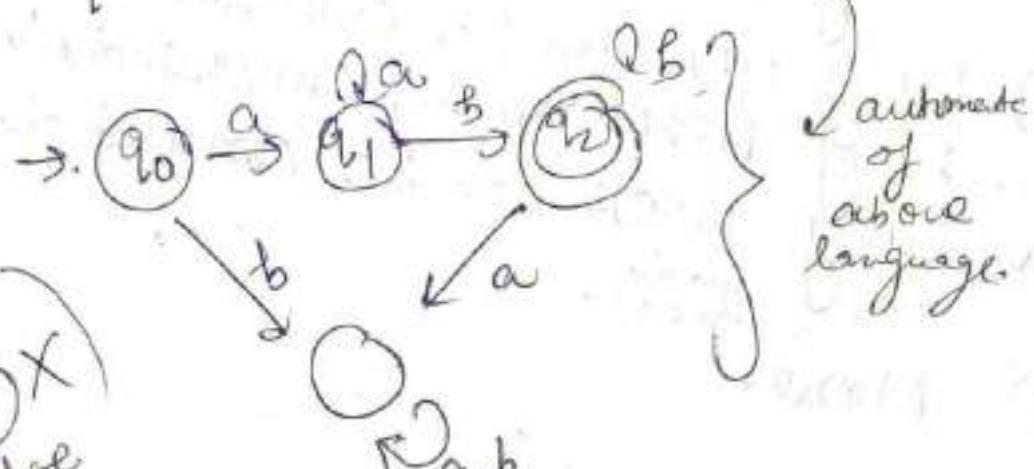
→ For regular-lang why we use FA
not TM because TM is included. FA
but FA is more simple.

$L_1 = \{a^n b^m \mid n, m \geq 1\} \rightarrow$ there is no comparison

$L_2 = \{a^n b^n \mid n \geq 1\} \rightarrow$ comparison
1 compⁿ

$L_3 = \{a^n b^n c^n \mid n \geq 1\} \rightarrow$ multiple X FA
2 compⁿ comparisons

FA + not possible → {ab, aab, aabb, aaaaab, ... } ~~DPDA~~



- In FA reading only possible not semantically.

$L = \{a^n b^m \mid n > m\}$ F.A X.

(18)
2/2
Ans

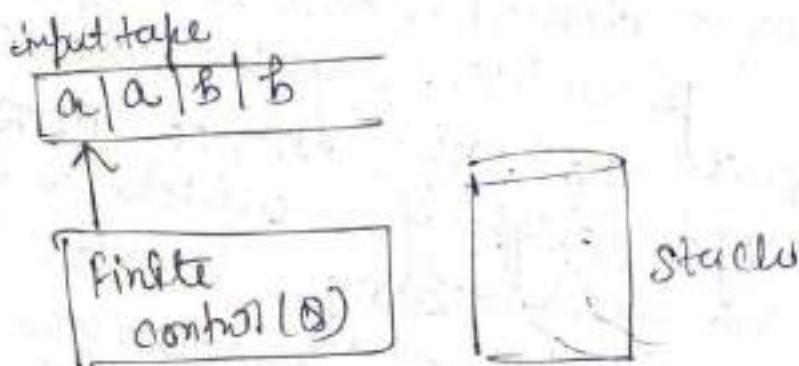
$L = \{a^n b^m \mid n \neq m\}$ F.A X.

- Finite automata can accept the language which don't have comparision between symbols.

- F.A fails to accept languages which are having comparisons between symbols.

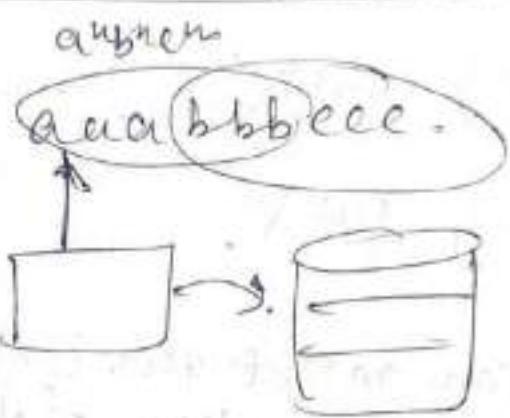
Ex: $\{a^n b^n \mid n \geq 1\}$
 $\{a^n b^n c^n \mid n \geq 1\}$.

There is no membrane in F.A so, for rememberance we add stack so, one DFA with stack called PDA.



$$\text{F.A + Stack} = \text{P.D.A.}$$

One stack handles one comparision if there is multiple comparision PDA fails.



Hence, these type of language accepted
by T.M.

- $L_1 = \{a^n b^m | n, m \geq 1\}$ F.A. ✓
- $L_2 = \{a^n b^n | n \geq 1\}$ F.A. ✗ PDA ✓
- $L_3 = \{a^n b^n c^n | n \geq 1\}$ F.A. ✗ PDA ✗ T.M. ✓

→ Pushdown automata can ~~accept~~ accept languages which requires one memory element (stack) only. Hence PDA fails to accept languages which requires more than one memory element. To accept such type of languages Turing machine will be constructed.

- Q. $L = \{a^n b^m c^n m^n | n, m \geq 1\}$ (no. of ab equals to c)
 equal. F.A. ✗ P.D.A. ✓ only one configuration.

Q. $L = \{a^n b^m \mid n \neq m\}.$

F.A \times .

P.D.A \checkmark (only one composition)

Q. $L = \{a^{2m} b^{3n} \mid n, m \geq 1\}.$

F.A \checkmark (no composition).

Q.

Consider the following three languages

$L_1, L_2 \text{ & } L_3$

$$L_1 = \{a^p b^q \mid p, q \geq 1\}$$

$$L_2 = \{a^p b^q \mid p = q\}$$

$$L_3 = \{a^p b^q c^r \mid p = q = r\}$$

which of the following is true \Rightarrow

(a) L_1, L_2 are accepted by finite automata.

(b) L_1, L_2, L_3 are accepted by PDA.

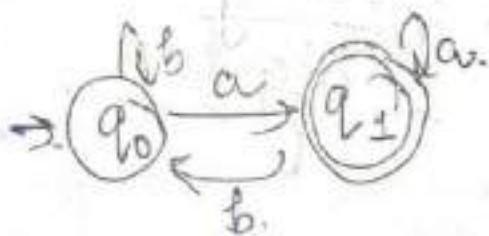
\checkmark (c) L_1, L_2, L_3 are accepted by TM.

(d) None of these.

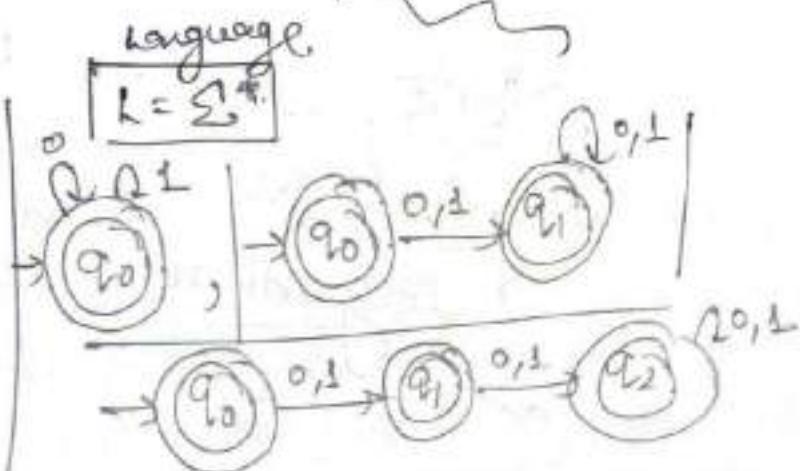
ex. of from one
very many DFA
exist



DFA \leftrightarrow Lang.



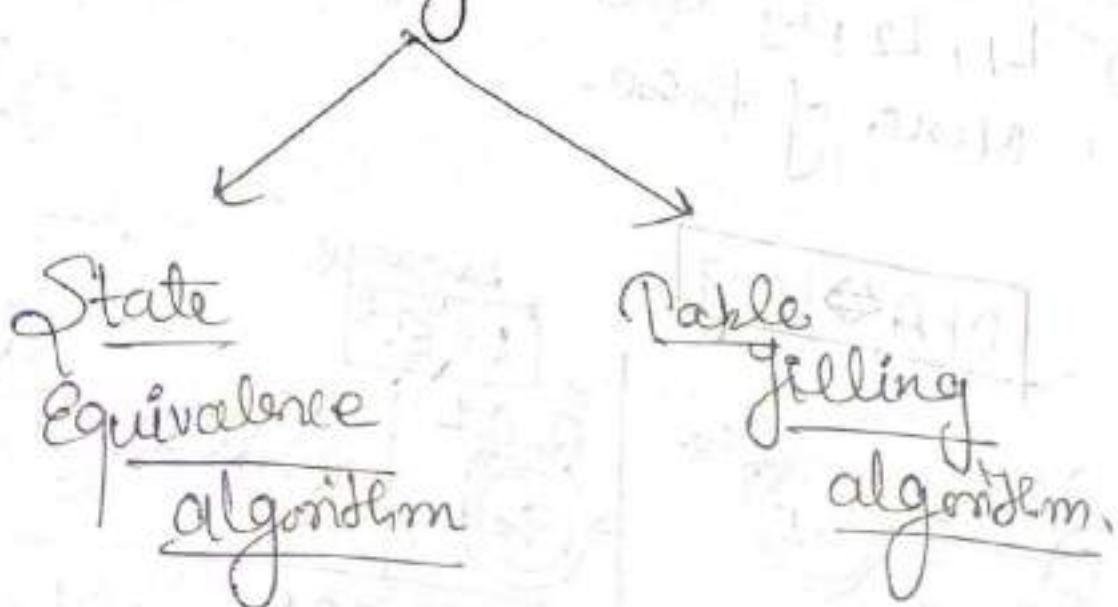
Lang:- Ending with 'a'



- Under one language, many no. of DFA exist
we prefer always min. no. of state DFA.
- But under one DFA only one lang. exist

Minimization of DFA :-

- Minimization is possible only for DFA, not possible for NFA and \in NFA.
- For one language many no. of DFAs is possible but the minimal state DFA is unique.
- Any DFA is converted into its equivalent minimal DFA by using following minimization algorithm,



State Equivalence algo →

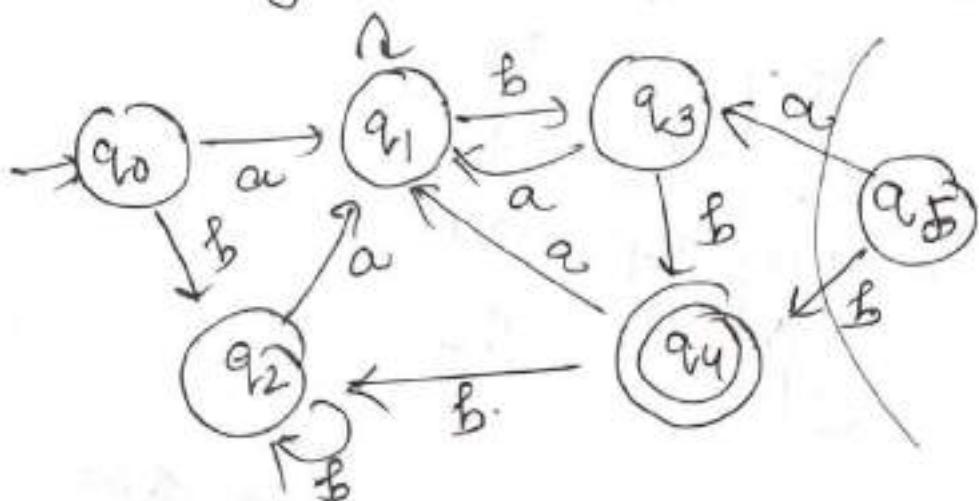
(20)

$$\begin{array}{c} (q_1, x) \xrightarrow{F} NF \\ (q_2, x) \xrightarrow{F} NF \end{array} \quad \left. \begin{array}{l} \text{equivalent} \\ \text{state} \end{array} \right\}$$

$$\begin{array}{c} (q_1, x) \xrightarrow{F} NF \\ (q_2, x) \xrightarrow{NF} \end{array} \quad \left. \begin{array}{l} \text{not equivalent state} \\ \end{array} \right\}$$

- Two states are equivalent for all $x \in \Sigma^*$ both are going to either final or non-final otherwise they are said to be non-equivalent.

Q. Minimize the following DFA:-



Soln:

Total 6 states,
there is possibility of minimization
mean
DFA with less no. of states

Minimization possible for DFA only
so, it is DFA.

1st remove inaccessible state.

Inaccessible state:-

- The state which is not reachable from the initial known as Inaccessible state.
- Inaccessible states should be removed from DFA.

remove q_5

	a	b
$\rightarrow q_0$	q_1	q_2
q_1	q_1	q_3
q_2	q_1	q_2
q_3	q_1	q_4
q_4	q_1	q_2

1st step:- separate all nonfinals in final.

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

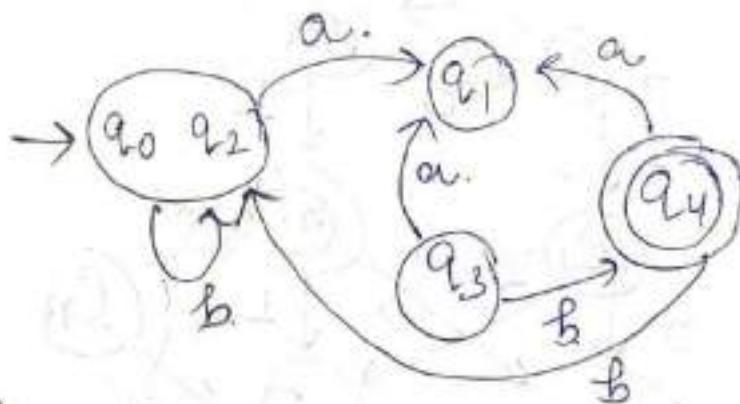
2nd step:- $\{q_0, q_1, q_2\} \{q_3\} \{q_4\}$

(2)

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

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

q_0, q_2
are
equal
merge as
one state

2nd method.

	a	b
q_0	q_1	q_2
q_2	q_1	q_2
q_{23}	q_1	q_3
q_3	q_1	q_4
q_4	q_1	q_2

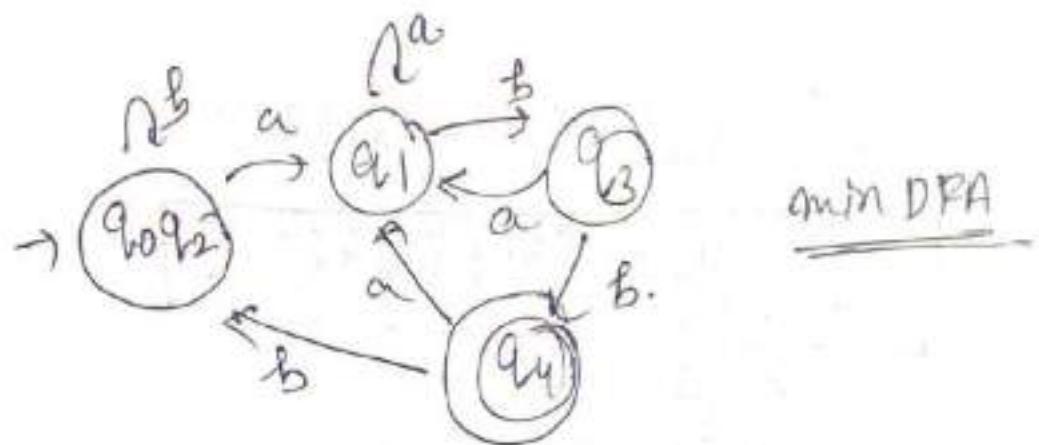
	a	b
q_{02}	q_1, q_2	q_1, q_2
q_1	q_1, q_3	q_1, q_3
q_3	q_1, q_4	q_1, q_4
q_4	q_1	q_2

→ final.
reduce

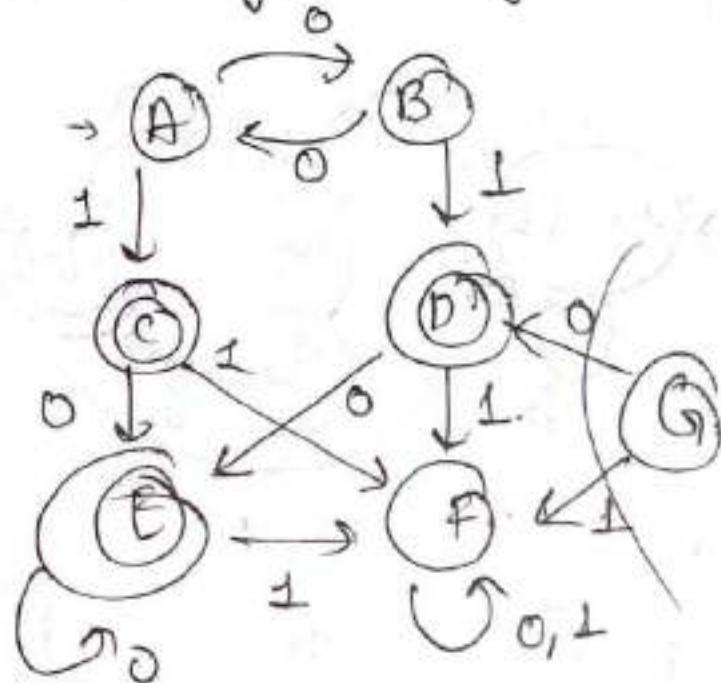
$$\{q_{02}, q_1, q_3\} \{q_4\}$$

$$\{q_{02}, q_1\} \{q_3\} \{q_4\}$$

$$\{q_{02}\} \{q_1\} \{q_3\} \{q_4\}$$



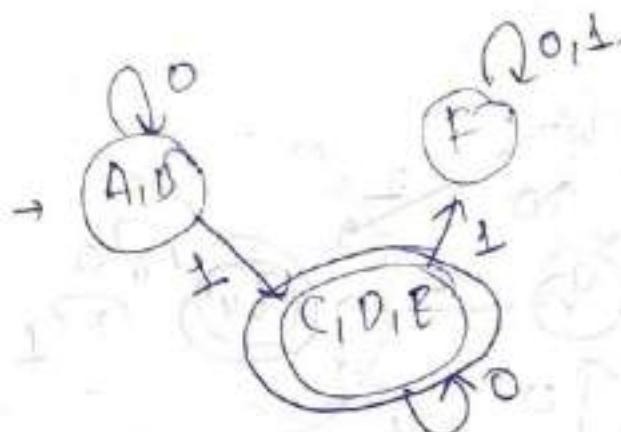
Q) Minimize the following DFA,



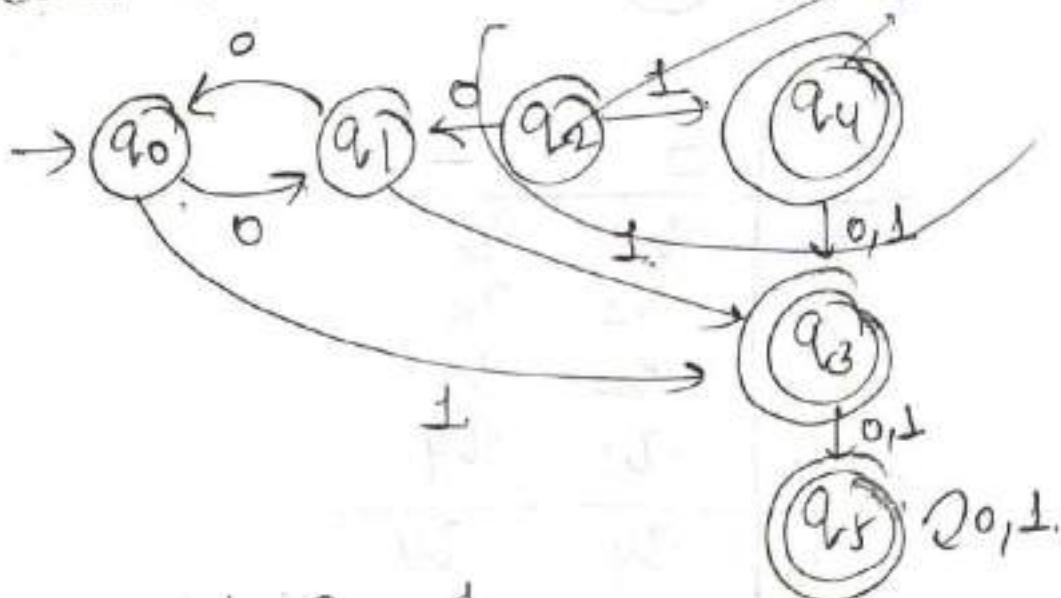
	0	1
A	B	C
B	A	D.
C	E	F
D	E	F
E	E	F
F	F	F

$\{A, B, F\} \quad \{C, D, E\}$

$\boxed{\{A, B\} \text{ ff } \{C, D, E\}.}$

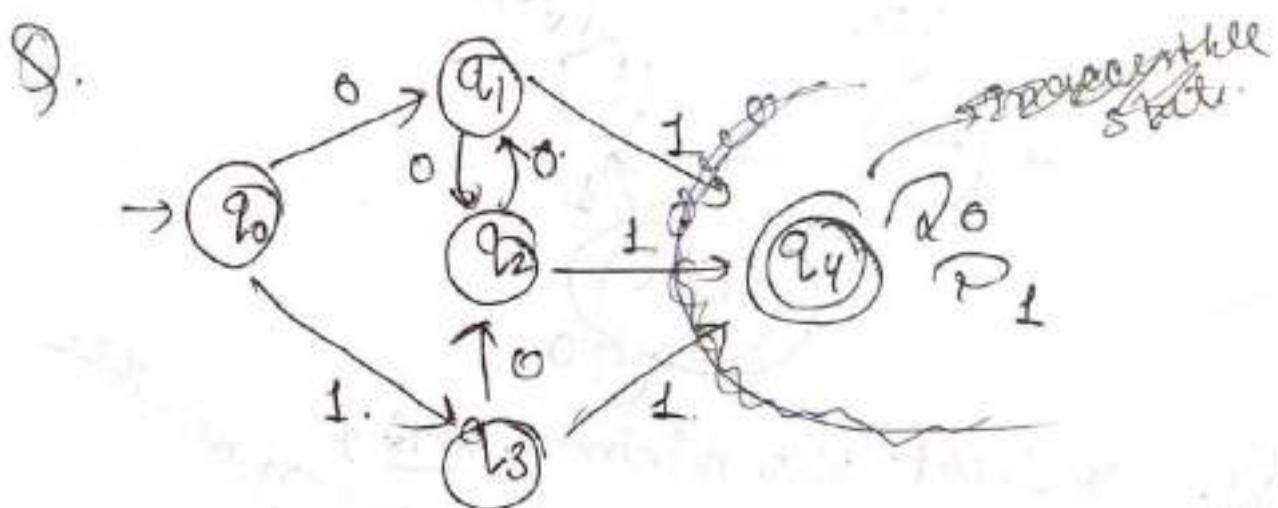
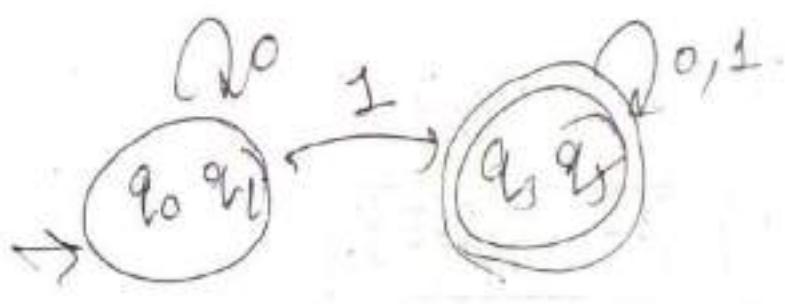


Q. Construct the minimal DFA, traceable



	0	1
q_0	q_1	q_3
q_1	q_0	q_3
q_3	q_5	q_5
q_5	q_8	q_5

$\boxed{\{q_0, q_1\} \{q_3, q_5\}}$



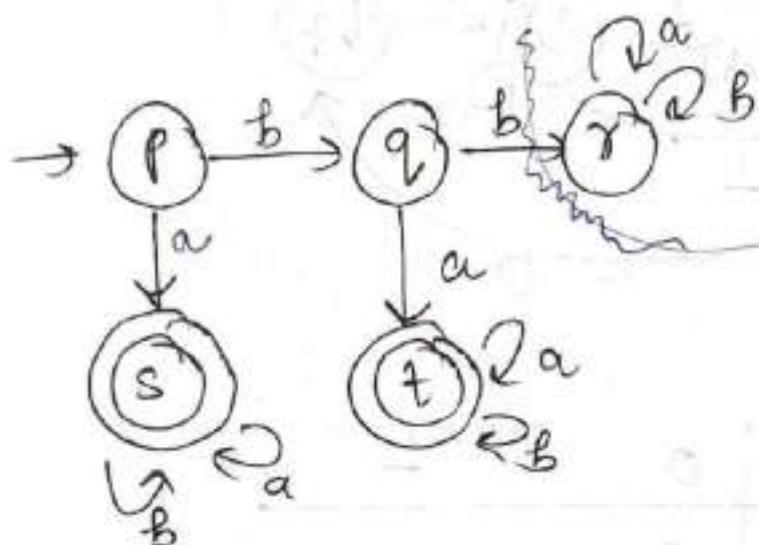
	0	1
q0	q_1	q_3
q_1	q_2	q_4
q_3	q_2	q_4
q_2	q_1	q_4
q_4	q_4	q_4

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

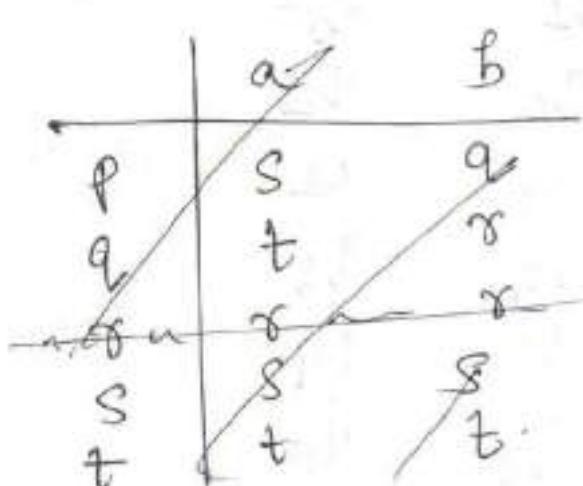
$\boxed{\{q_0\} \{q_1, q_3, q_2\} \{q_4\}}$ g states -

q_1, q_2, q_3 are equivalent state.

Q. Consider the minimal DFA for the following DFA:-

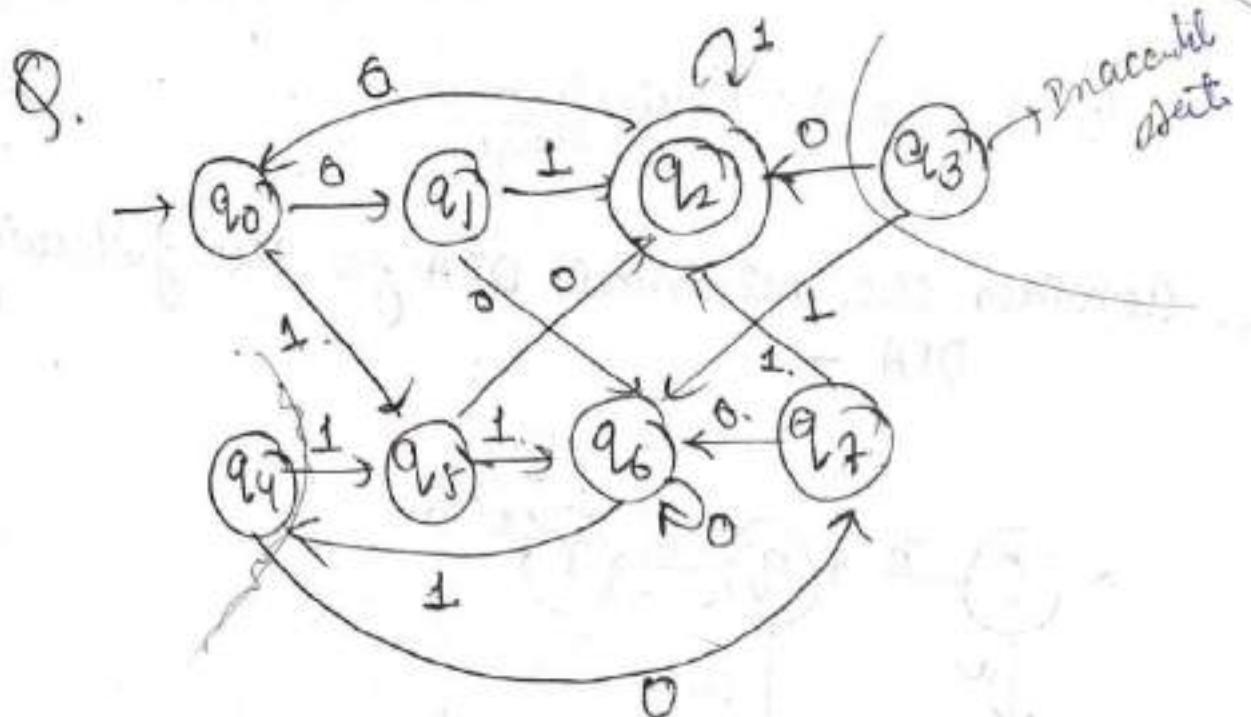


Solⁿ:



j	a	b
p	s	t, q
q	t	r
r	r	r
s	s	s
t	t	t

$\{p, q\} \{s, t\}, \{p, q, r\} \{s, t\}$
 $\{r\} \{p\} \{q, r\} \{s, t\}, \boxed{\{p\} \{q, r, r\} \{s, t\}}$



q_0^n

	0	1
$\rightarrow q_0$	q_1	q_5
q_1	q_6	q_2
q_2	q_0	q_2
q_5	q_2	q_6
q_6	q_6	q_4
q_4	q_7	q_5
q_7	q_6	q_2

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

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

$\{q_0 \underline{q_6}\} \{q_6\} \{q_1 \underline{q_5}\} \{q_2\} \{q_7\} \{q_2\}$

Q.

	0	1
q_{04}	q_{17}	q_5
q_{17}	q_6	q_2
q_5	q_2	q_6
q_6	q_6	q_4
q_2	q_0	q_2

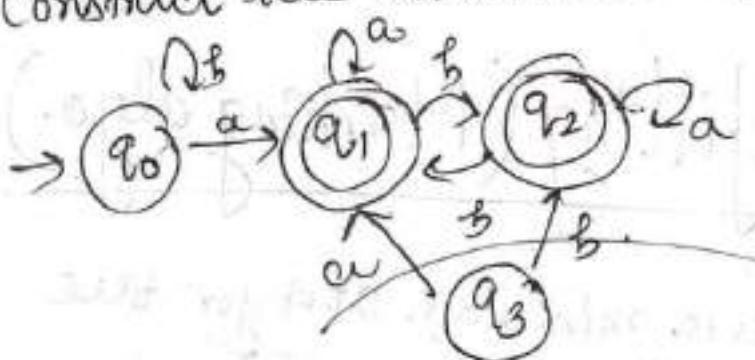
(24)

$\{q_{04}, q_{17}, q_5, q_6\} \cup \{q_2\}$.

$\{q_{04}, q_{17}, q_6\} \cup \{q_5\} \cup \{q_2\}$.

$\boxed{\{q_{04}\} \cup \{q_{17}\} \cup \{q_6\} \cup \{q_5\} \cup \{q_2\}}$

Q. Construct the minimal DFA,

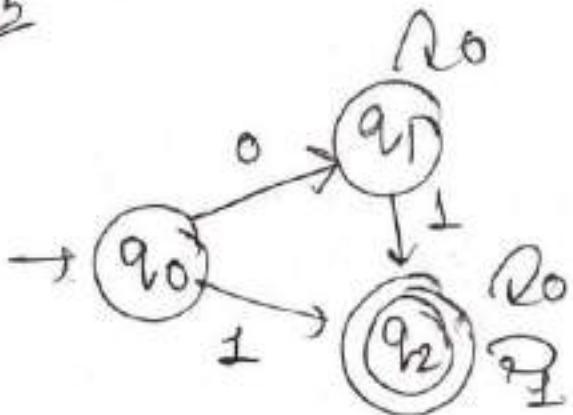


	a	b
q_0	q_1	q_0
q_1	q_1	q_2
q_2	q_2	q_1

$\boxed{\{q_0\} \cup \{q_1\} \cup \{q_2\}}$

Equivalent stat.

Ques. GATE 2013

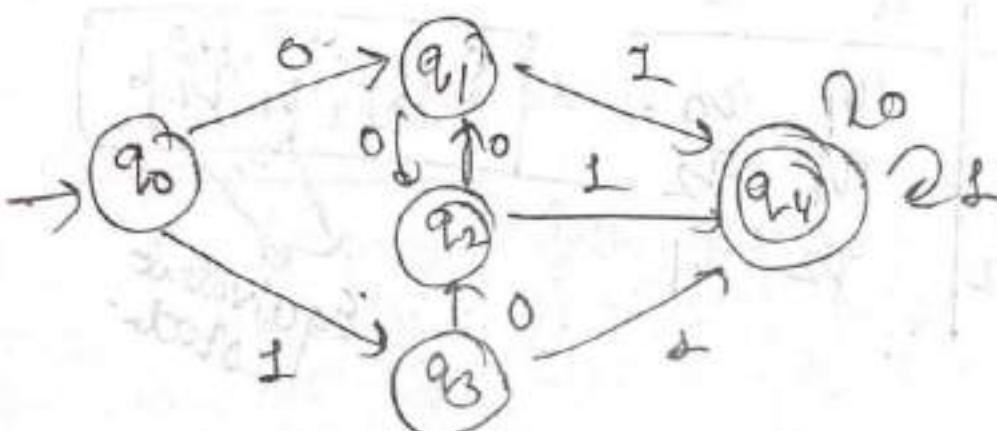


	0	1
q0	q1	q2
q1	q1	q2
q2	q2	q2

$\{q_0, q_1\} \cup \{q_2\}$ 2 states.

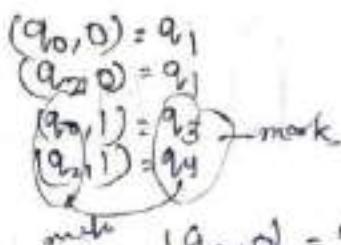
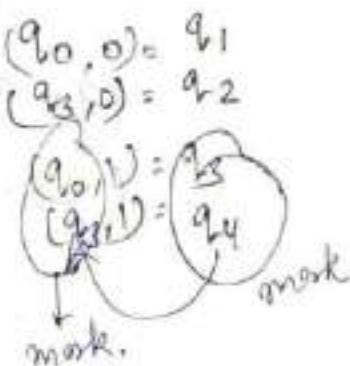
Ques. Table filling (Marking algo.)

Construct the minimal DFA for the following :-



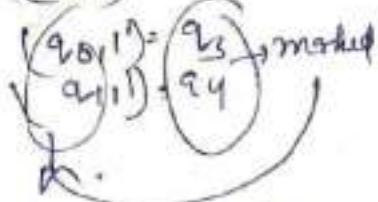
q_1	X.			
q_2	X	✓		
q_3	X	✓	✓	
q_4	X	X	X	X

first mark
all final & nonfinal state



$$(q_0, 0) = q_1$$

$$(q_1, 0) = q_2$$



so mark

$$\left. \begin{array}{l} (q_2, 0) = q_1 \\ (q_1, 0) = q_2 \\ (q_2, 1) = q_4 \\ (q_1, 1) = q_4 \end{array} \right\} \text{not mark.}$$

$$\left. \begin{array}{l} (q_3, 0) = q_2 \\ (q_2, 0) = q_1 \\ (q_3, 1) = q_4 \text{ mark.} \\ (q_2, 1) = q_4 \end{array} \right\} \text{not mark.}$$

$$\left. \begin{array}{l} (q_1, 0) = q_3 \\ (q_3, 0) = q_2 \\ (q_1, 1) = q_4 \\ (q_3, 1) = q_4 \end{array} \right\} \text{not mark.}$$

The strings that are not marked
are equivalent state.

$$(q_1, q_2) \xrightarrow{\text{merge}} (q_1, q_2, q_3).$$

(or)

$$\begin{array}{c} (\underline{q_1 \ q_2}) \\ (\underline{q_2 \ q_3}) \end{array} \rightarrow \begin{array}{c} (\underline{\underline{q_1 \ q_2}} \ q_3) \\ \text{equal} \\ \text{state} \end{array}$$

DFA Construction

Q. Construct the minimal DFA for the language.

$$L = \left\{ a^n b^m \mid n \neq m \right\}_{n,m \geq 1}$$

Solⁿ: DFA not possible because of comparison.

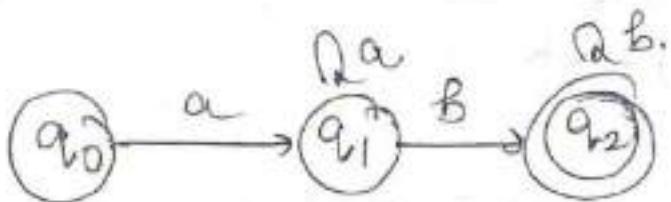
Q. Construct the minimal DFA for L

$$L = \left\{ a^n b^m \mid n, m \geq 1 \right\}$$

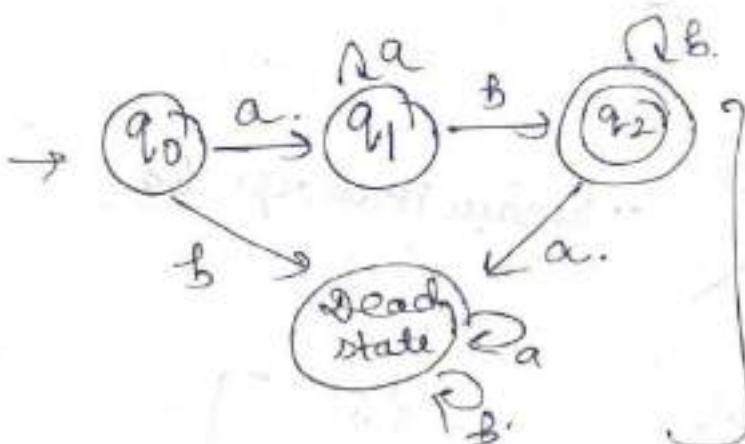
Sol^m: $L = \{ ab, aab, abb, aabb, aaabb, abbbb, aabab, \dots \}$

(26)

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



→ It is not deterministic to make it deterministic we add dead state.



it is minimal DFA it is clear by applying minimization.

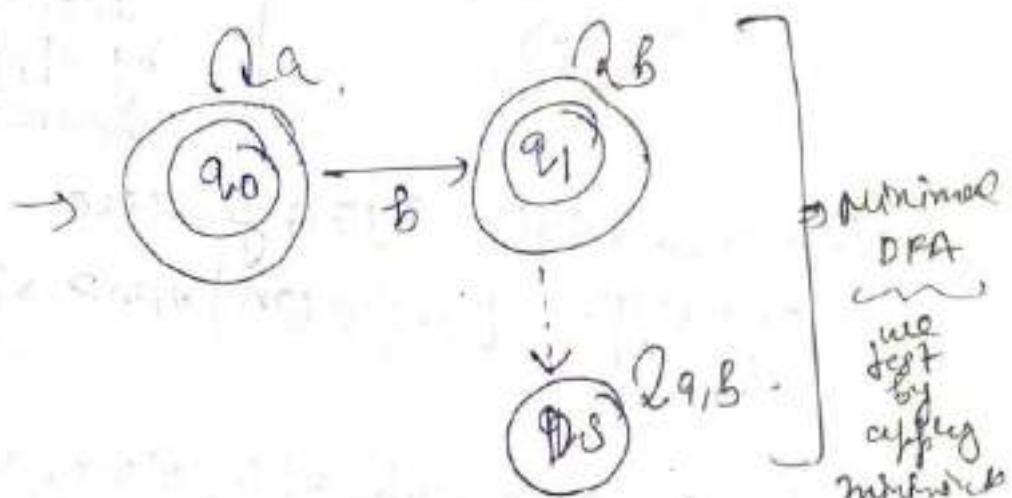
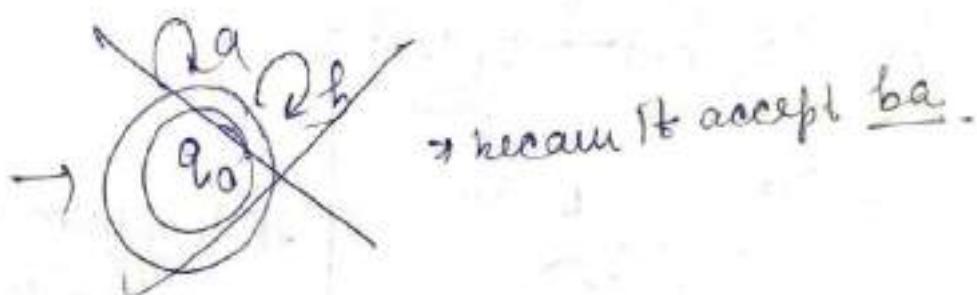
Q: Construct the minimal DFA for the language, $L = \{a^n b^m \mid n, m \geq 0\}$

Solⁿ: $\rightarrow L = \{\epsilon, q_1 b, ab, aqb, baa, aqb, aaaa, \dots\}$



$$L = \left\{ \begin{array}{l} c \\ a, aa, aab \dots \\ b, bb, bbb \dots \\ ab, abb, abb \dots \end{array} \right.$$

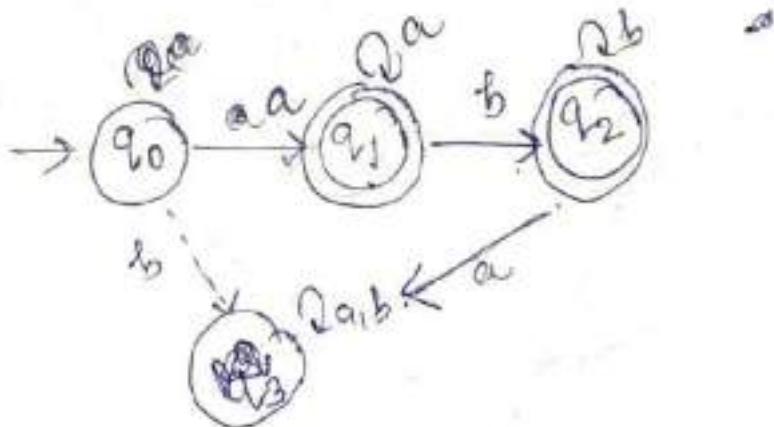
$\left. \begin{array}{l} ba \\ bab \\ aba \end{array} \right\}$ not possible.



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

$$L = \{a, ab, aab, abbb, aabb, \dots, @\}$$

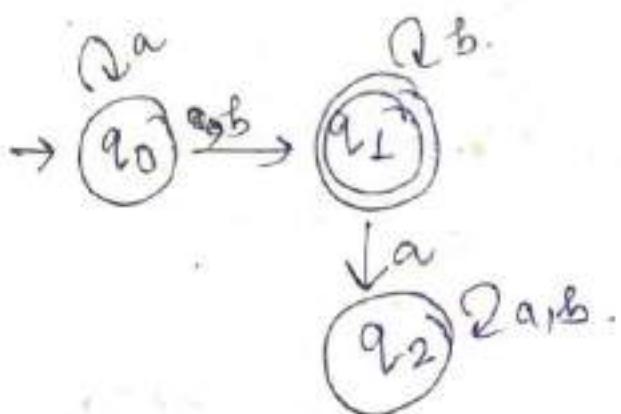
ba^x. ab^x. ^{not allowed}



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

$L = \{b, bb, bbb, \dots,$
 $\quad ab, aab, aaab, abbb, abb, \dots\}$

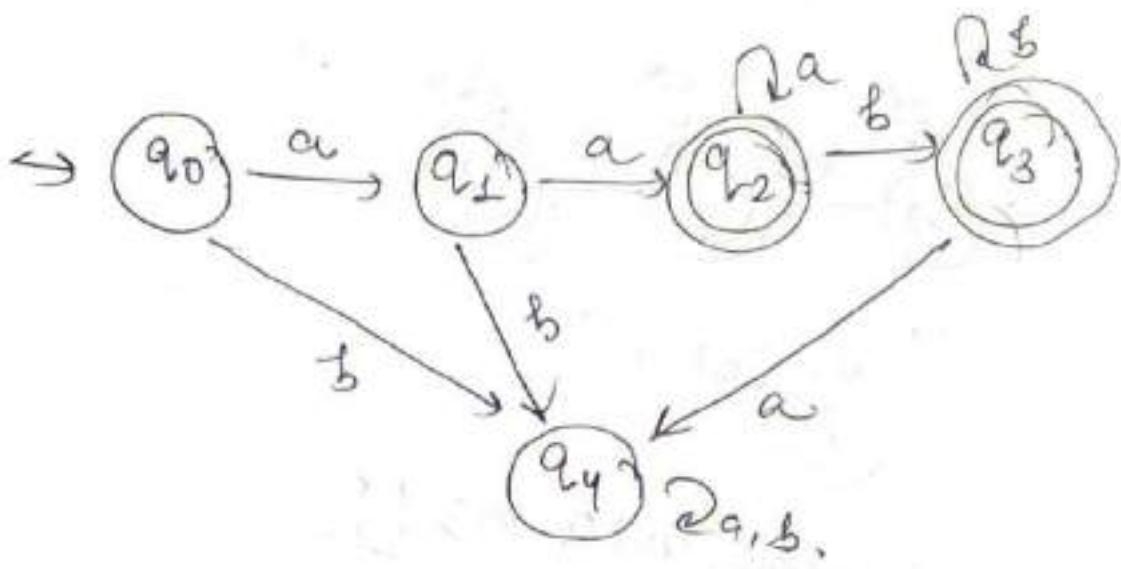
$a \times$
 $ab \times$.



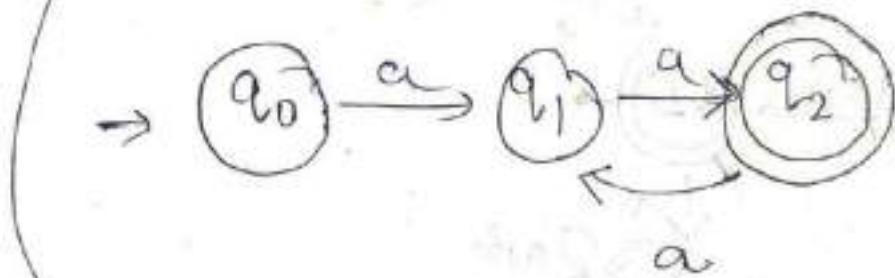
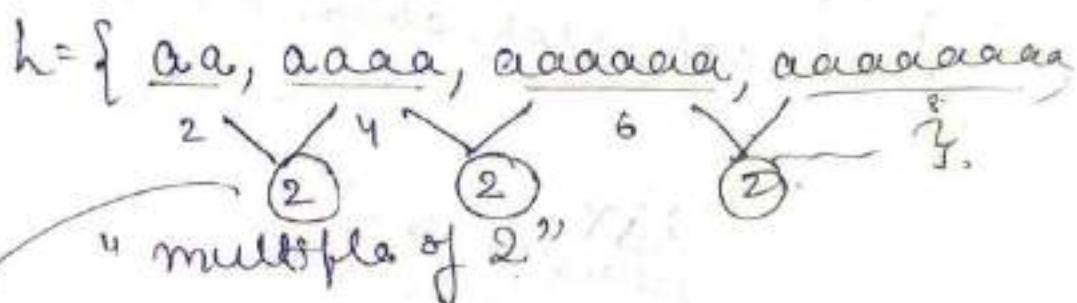
Q. $L = \{a^n b^m \mid n \geq 2, m \geq 0\}$

$L = \{aa, aaa, aaaa, \dots,$
 $\quad aab, aabb, aakbb, \dots,$
 $\quad aab, aaab, aaaaab, \dots\}$

$b \times$
 $ab \times$



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

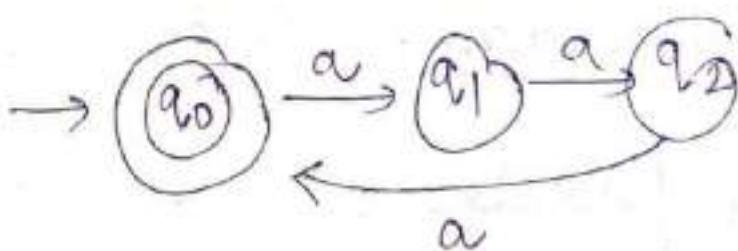
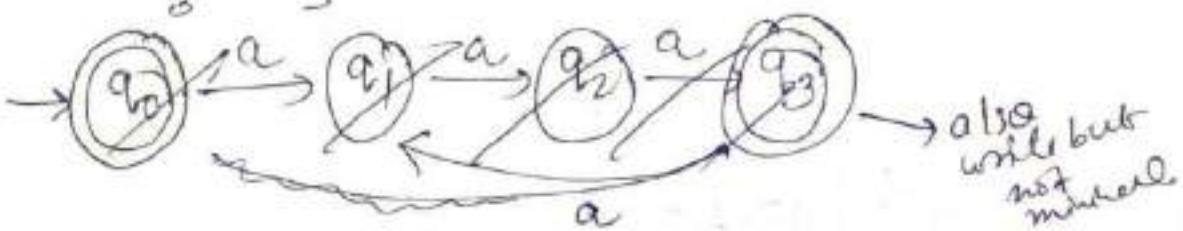


there is a common diff. of '2' so, we take '2' length cycle.

Q. $L = \{a^n \mid n \bmod 3 = 0\}$.

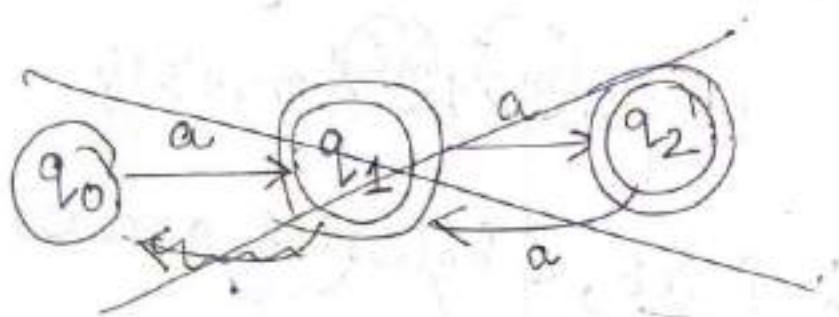
Let $n = 0, 3, 6, 9, \dots$

$$L = \{ \epsilon, \underbrace{aaa}_{3}, \underbrace{aaaaaa}_{3}, \dots \}$$



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

$L = \{ \underbrace{a}_{1}, \underbrace{aa}_{2}, \underbrace{aaaa}_{4}, \underbrace{aaaaaa}_{8}, \dots \}$



DFA not possible for this.

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

$L = \{ a, \underbrace{aaaa}_{4}, \underbrace{aaaaaaaa}_{16}, \dots \}$

Not
possible

Q. $L = \{a^n! \mid n \geq 1\}$.

$L = \{a, aa, aaaa, \dots\}$

Not possible.

- Any language symbol power not in arithmetic progression (seen DFA not possible to construct)

Q. Construct minimal DFA for,

$$L = \{a^{m^2} b^{n^3} \mid m, n \geq 1\}$$

$L = \{ab, aabb, aabbabb, aaaaab, \dots\}$

symbol
powers not
in AP.
so, not
possible DFA.

(29)

$$h = \{ a^{nk} \mid k \geq 1 \\ n \text{ is positive const} \}.$$

How many no. of states are there in
 $L = \{ a^n \mid n \in \text{positive const} \}$ minimal DFA to accept $\text{sol. } L$.

- (a) $k+1$
- (b) 2^{k+1}
- (c) 2^{nk}
- (d) None

Sol: ~~$L = \{ a^n, a^{nk}, a^{3n}, a^{4n}, \dots \}$~~

when $n=1$:

$$L = \{ aa, aaa, aaaa, \dots \}$$

$n=2$

$$L = \{ a^2, a^4, a^8, a^{16}, \dots \}$$

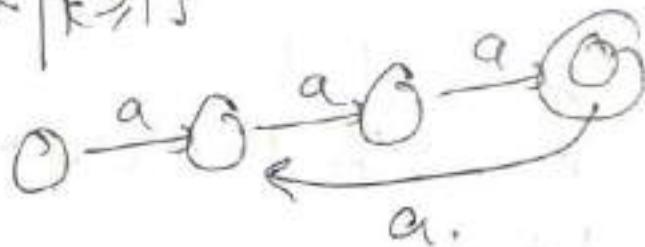
Sol: $\begin{array}{l} (m=1) \\ L = \{ a^k \mid k \geq 1 \} = \text{Q} \xrightarrow{a} \text{Q}^2 \xrightarrow{a} \dots \\ a, aa, aaa, \dots \end{array}$

$m=2$

$$L = \{ a^{2^k} \mid k \geq 1 \} = \text{Q} \xrightarrow{a} \text{Q} \xrightarrow{a} \text{Q} \xrightarrow{a} \dots$$

(m=3)

$$L = \{ a^{3k} \mid k \geq 1 \}$$



No. of states depends on n not on
 k so, we reject 1st & 2nd
option,

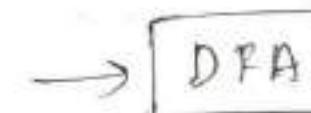
In above analysis, we
have

$(n+1)$ state.

If $n=3$ then the state is 4.
So, answer (d) which is $(n+1)$,

Q. Construct the minimal DFA that accept all
binary no.'s divisible by 4.

Ans:

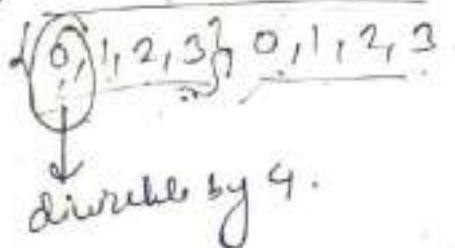


\cup_2	000✓	1000✓
	001X	1001X
	010	1010X
	011X	
	100✓	
	101X	
	110X	
	111X	

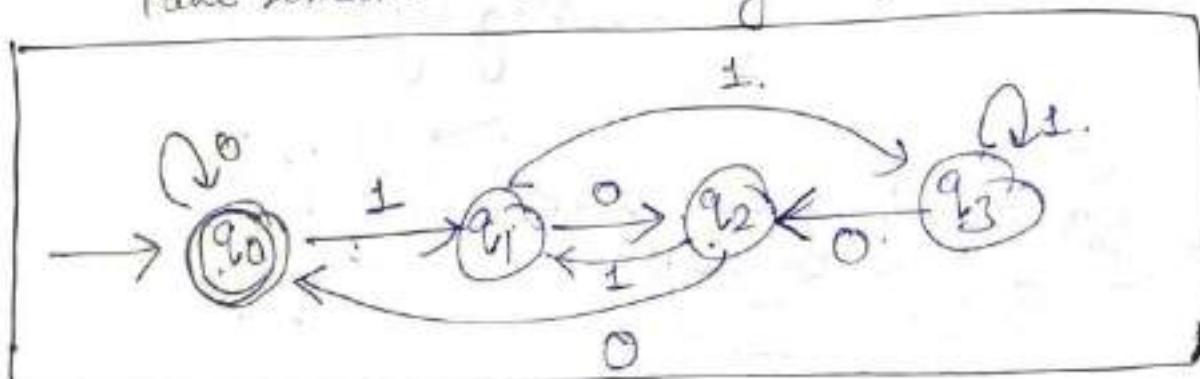
4) $0, 1, 2, 3, 4, 5, 6, 7, 8$

\rightarrow Laplace (1, 2, 3)
 \rightarrow Matrix.

(30)



Take remainders as a state of DFA,



000
 0001
 00010
 0011
 100
 101
 110
 111
 :

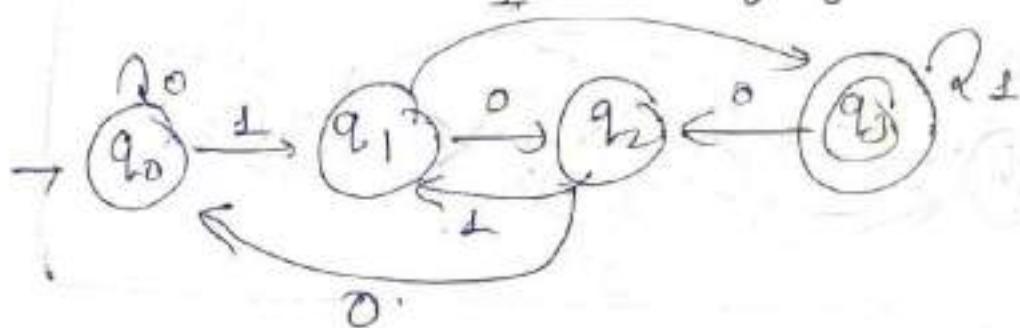
Until the system become deterministic we give transition.

	0	1	$\{q_0\} \{q_1, q_2, q_3\}$
q_0	q_0	q_1	$\{q_0\} \{q_1, q_2, q_3\}$
q_1	q_2	q_3	$\{q_0\} \{q_1, q_3\} \{q_2\}$
q_2	q_0	q_1	
q_3	q_2	q_1	

minimum state.

Q. construct the minimal DFA that accept
 all binary no's
 congruent to $3 \bmod 4$.
 means
 equal.

Ans: $()_2 = 3 \bmod 4$.
 dividing by 4 remainder is 3.



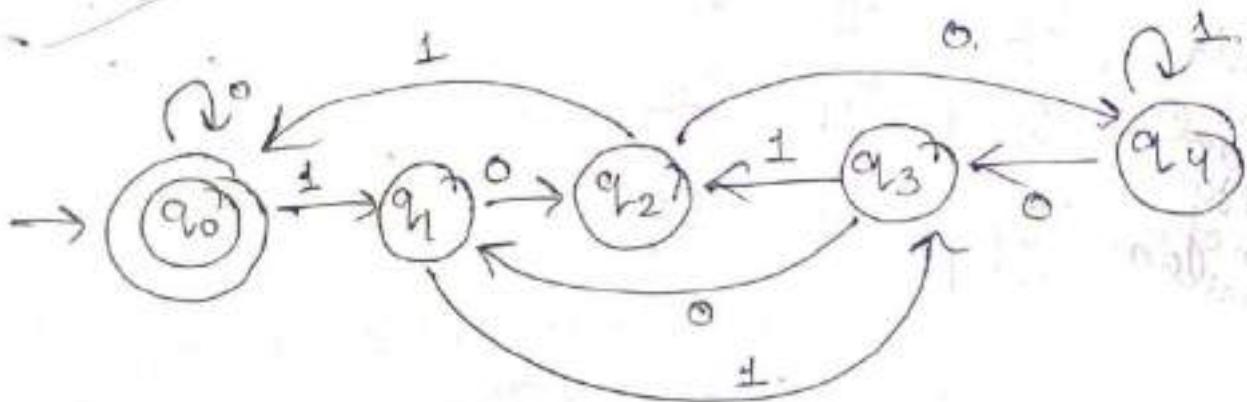
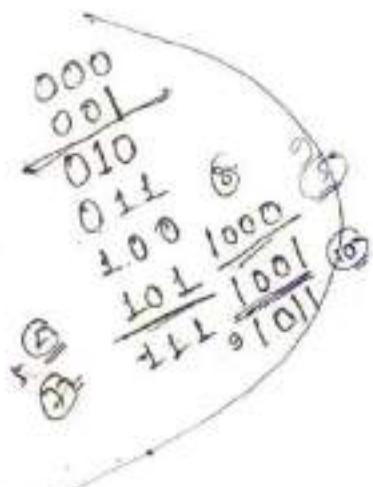
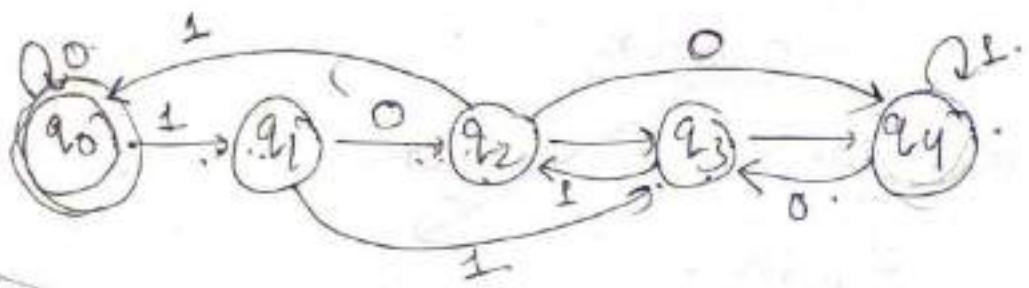
	0	1
q_0	q_0 q_1	
q_1	q_2 q_3	
q_2	q_0 q_1	
q_3	q_2 q_3	

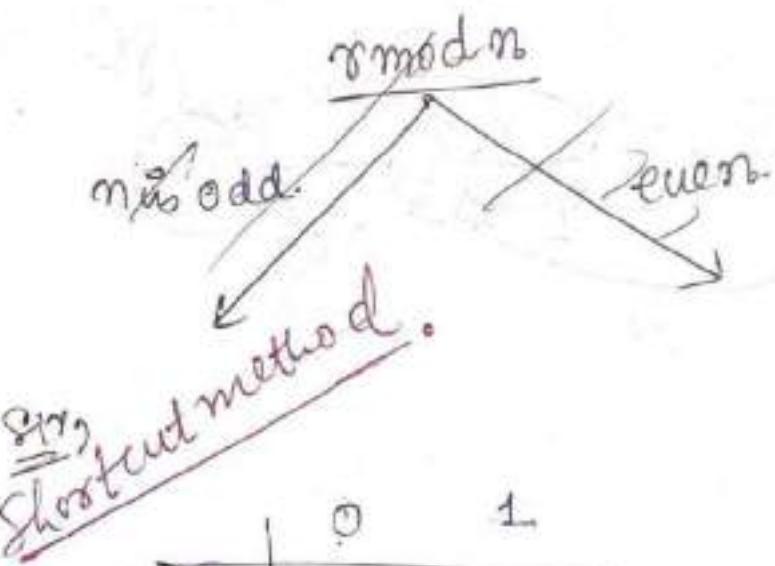
$\{q_0 q_1 q_2\} \{q_3\}$
$\{q_0 q_2\} q_3 \{q_3\}$

Q. construct the minimal DFA that accept
 all binary no's divisible by 5.

Ans: remainder $\rightarrow 0, 1, 2, 3, 4$

(31)





	0	1
$\rightarrow q_0$	q_0	q_1
q_1	q_2	q_3
q_2	q_4	q_0
q_3	q_1	q_2
stats one equal $1/20$ no. of remainder	q_4	q_3

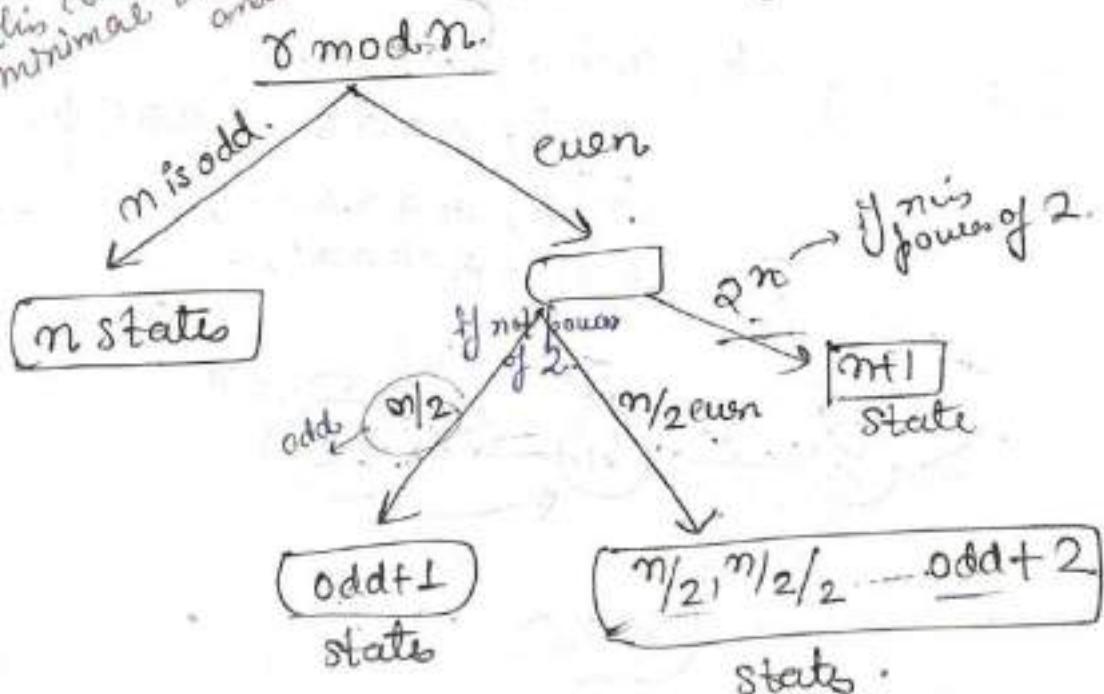
Q. Binary no. divisible by 7.

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

(7)
01

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

* In concept DFA is minimal answers. (only for binary no. 2).



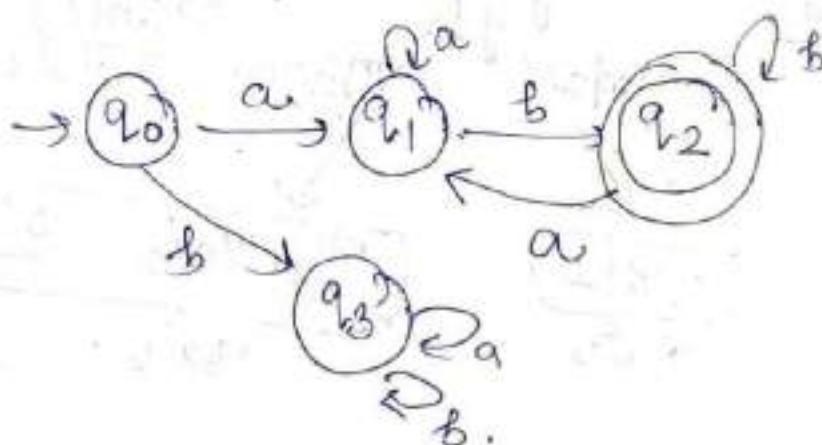
Q. Construct the minimal DFA
that accept all base 3 numbers
which are divisible by 4.

sol: $(\cdot)_3 \rightarrow$ divisible by 4.

	0,	1,	2,	
$\rightarrow q_0$	q_0	q_1	q_2	$\{q_0\} \{q_1 q_2 q_3\}$
q_1	q_3	q_0	q_1	$\overbrace{q_0}^1 \overbrace{q_1}^2 \overbrace{q_2}^0 \overbrace{q_3}^1$
q_2	q_2	q_3	q_0	
q_3	q_1	q_2	q_3	

Q. Construct the minimal DFA, that accept all strings of a's & b's, where each string is starting with a 'a' & ending with b.

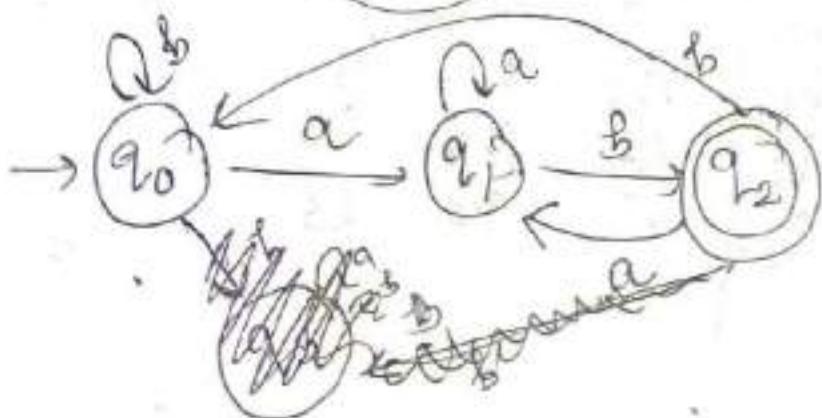
Soln: $L = \{ ab, aabb, \text{ (ahhb), } \dots, \text{ (aab), } \text{ (aaab), } \dots, \text{ (aabb), } \text{ (aaabb), } \dots, \text{ (abab), } \text{ (aabab), } \dots \}$



Total 4 states

Q. Each string ending with ab.

$L = \{ ab, aab, aaaa, \text{ (bab), } \dots, \text{ (abab), } \text{ (abaaaab), } \dots, \text{ (ahbab), } \dots \}$



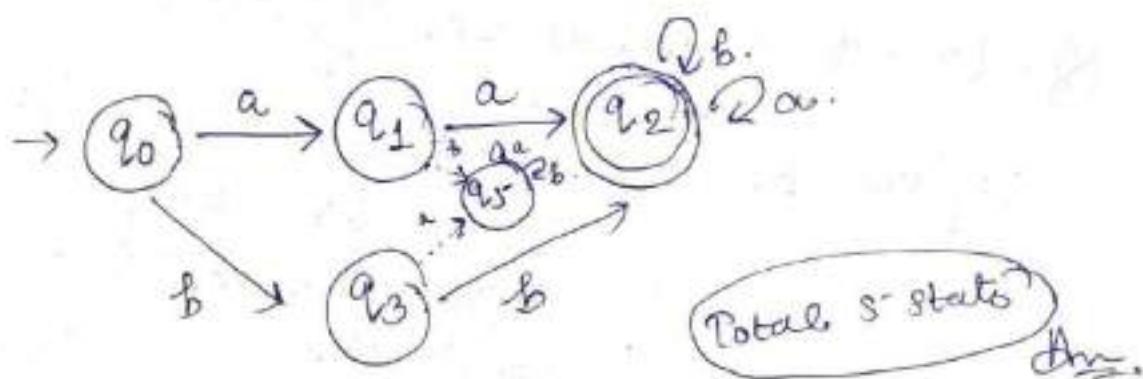
Ques

Q. 1st two symbols are same

(33)

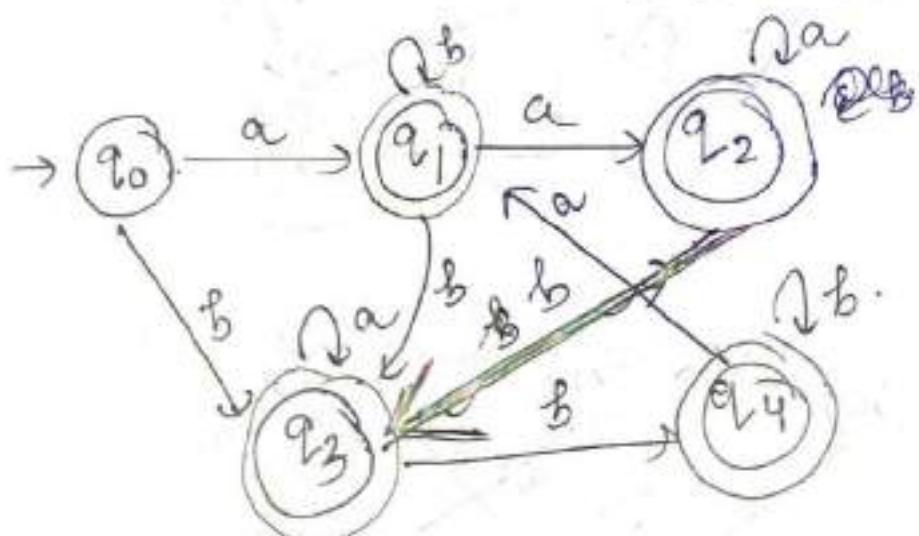
a, b, \dots in every string,
 $L = \{ aab, bba, aaabbab, \dots \}$

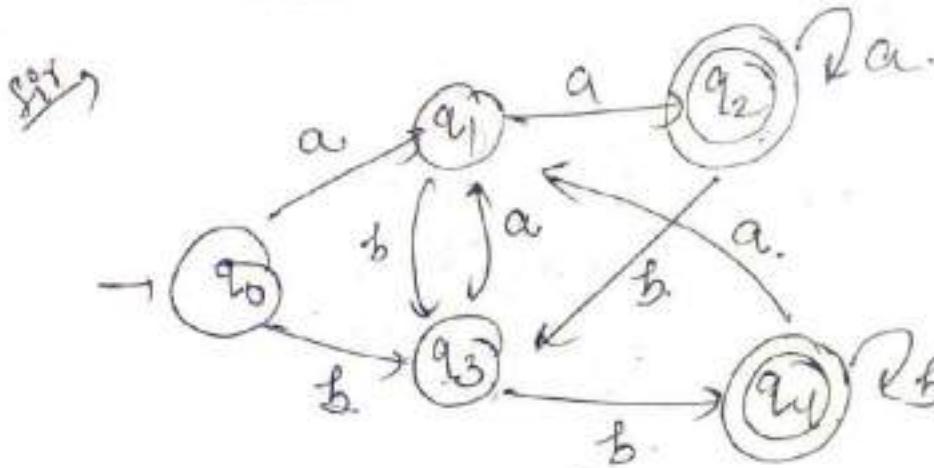
}



Q. Last two symbols are same.

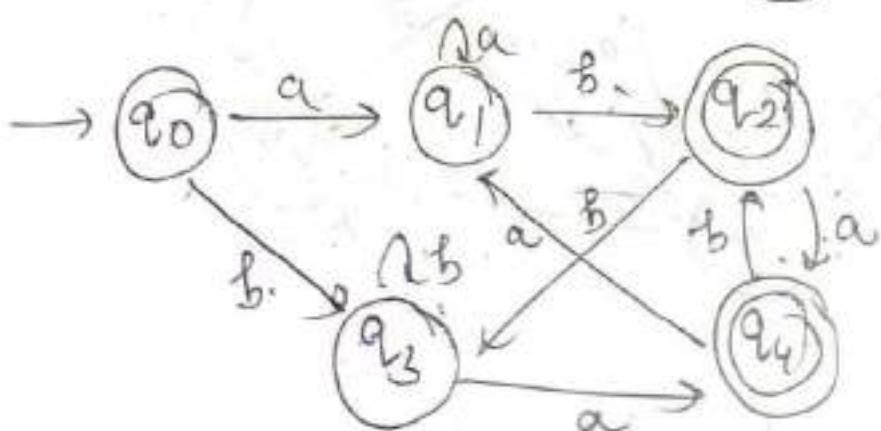
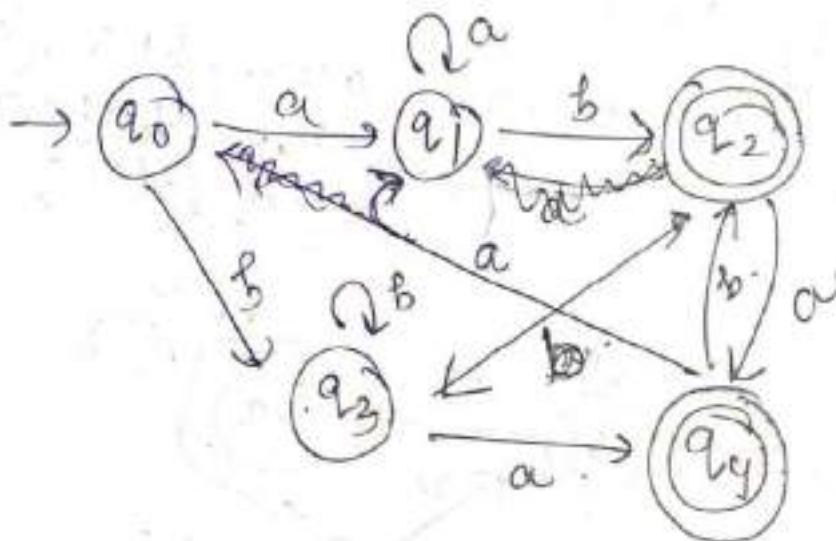
$L = \{ aa, bb, aaa, bbb, abab, baba, aabb, bbba \}$





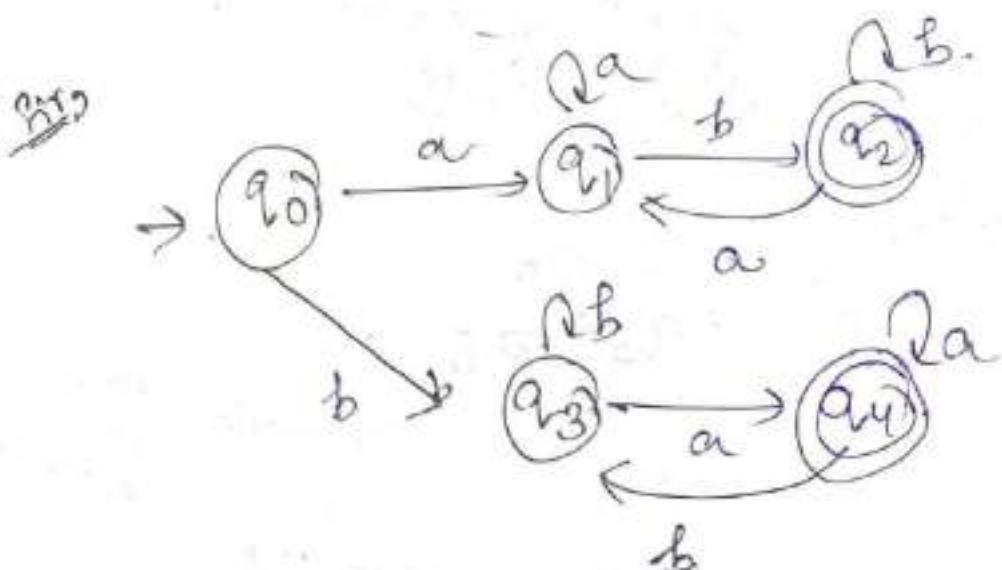
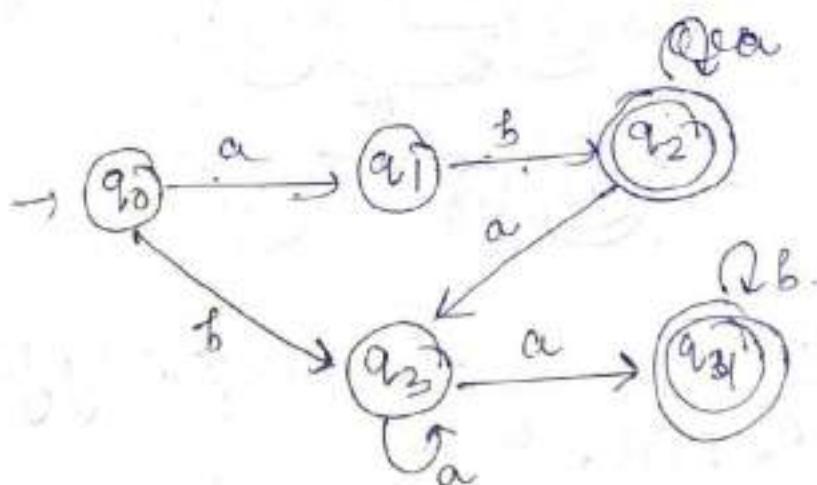
Q. Last two symbols are different.

$L = \{ \text{aaab}, \text{abab}, \text{aab}, \text{aaabb}, \text{bab}, \text{abb}, \text{abab}, \text{abbb}, \text{baab}, \text{akkkba}, \text{aaaaab} \}$



Q. Each string starting & ending with diff. symbol.

$L = \{ab, ba, ababa, abbba,$
 $ababab, abbabab,$
 $\dots \}$



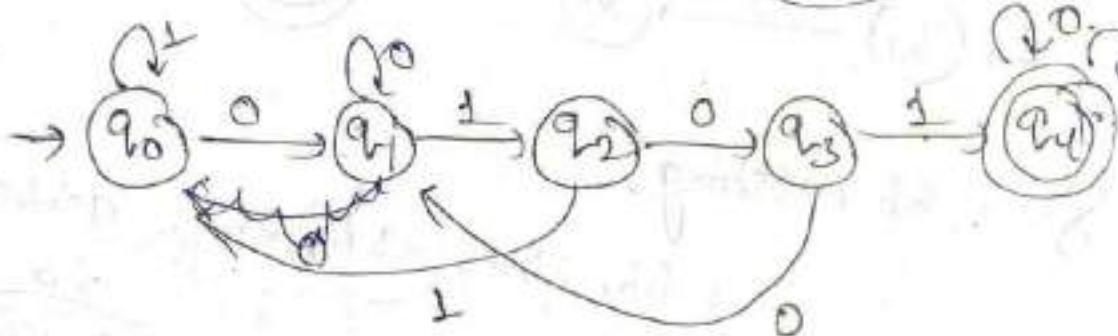
Q. Starting & Ending with same symbol.

* The minimal DFA that accept in length substring requires $(n+1)$ states.

Q. 0101 as substring

$h = \{ 0101, 01010, 0\underline{10}0101, \underline{01011}, 10101,$
 $\underline{01010101}, \underline{0100101}, \underline{0110101} \}$

0101



Q. Find it, 'a' substring.

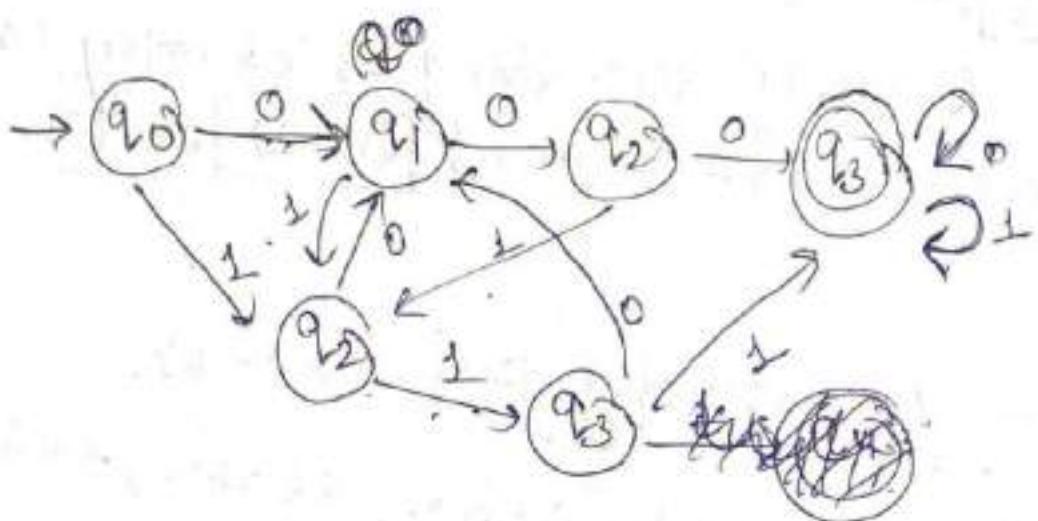
3 bits.

④ $\begin{matrix} 000 \\ 111 \end{matrix} \rightarrow \text{accept}$

$h =$

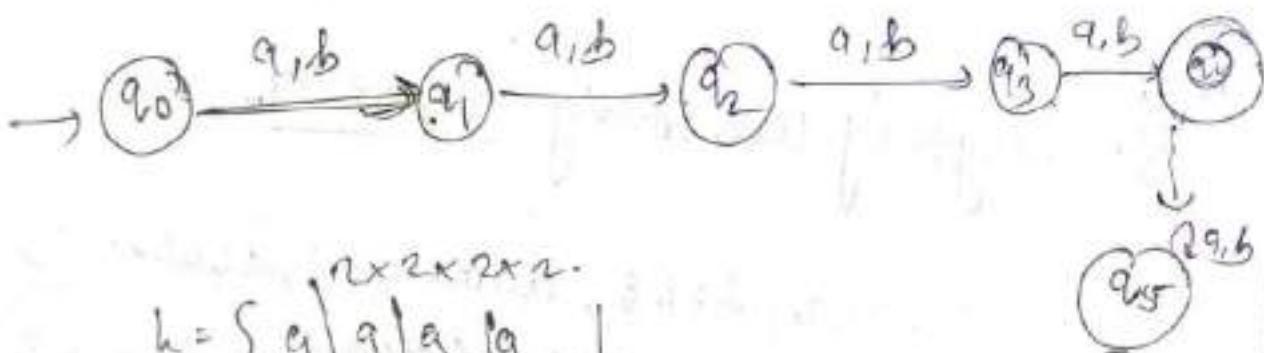
$\{ 0000, \underline{1000}, 0111, 1111, 00111, 11000, \underline{\dots} \}$

(36)



Q. Construct the minimal DFA,
that accept all strings of a 's & b 's
where every length string is exactly
4.

Solⁿ: $L = \{ \text{aaaa, bbbb, abab, aaab, bbba, } \\ \text{babab, } \dots \}$
Total
16 strings.



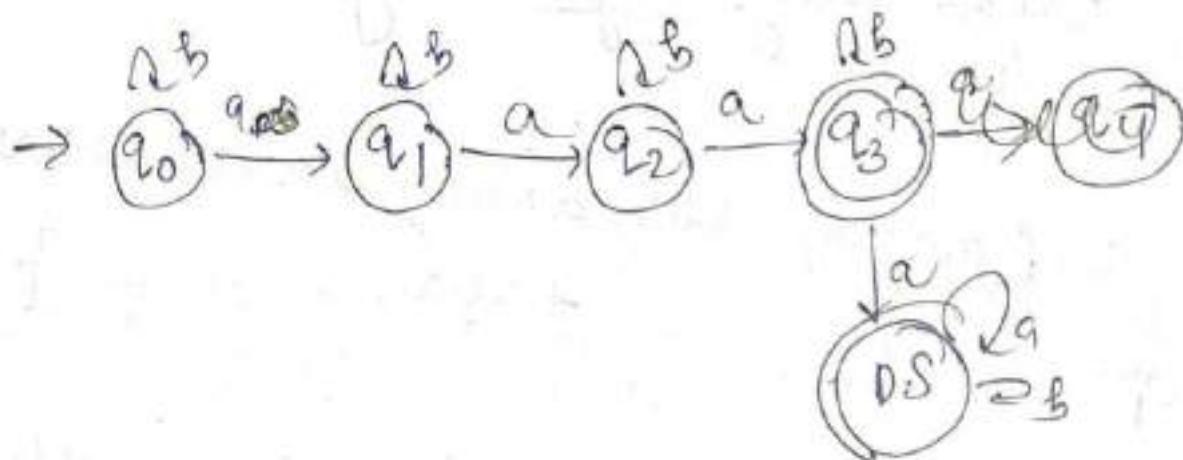
$$L = \{ \begin{array}{|c|c|c|c|} \hline & a & a & a \\ \hline & b & b & b \\ \hline a & a & a & a \\ \hline b & b & b & b \\ \hline a & a & b & b \\ \hline b & b & a & a \\ \hline \end{array} \}$$

Note:-

Length of the string is exactly m ,
minimal DFA requires $(m+2)$ states.

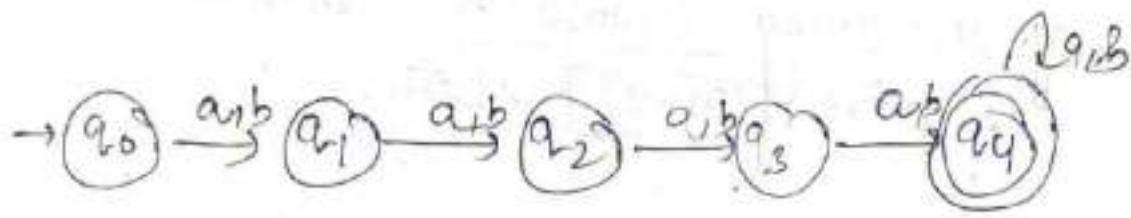
Q. No. of ds exactly three. $\Sigma = \{a, b\}$.

$L = \{aaa, aab, baa, aba, bab, bababa, \dots\}$.



Q. Length of the string at least 4.

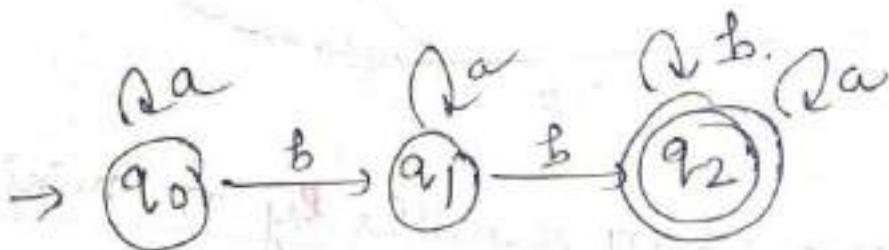
$L = \{aaaa, bbbb, aaaa\dots, bbbb\dots, abab, baba\dots\}$



* Length of the string at least 'm' minimal DFA requires $(m+1)$ states.

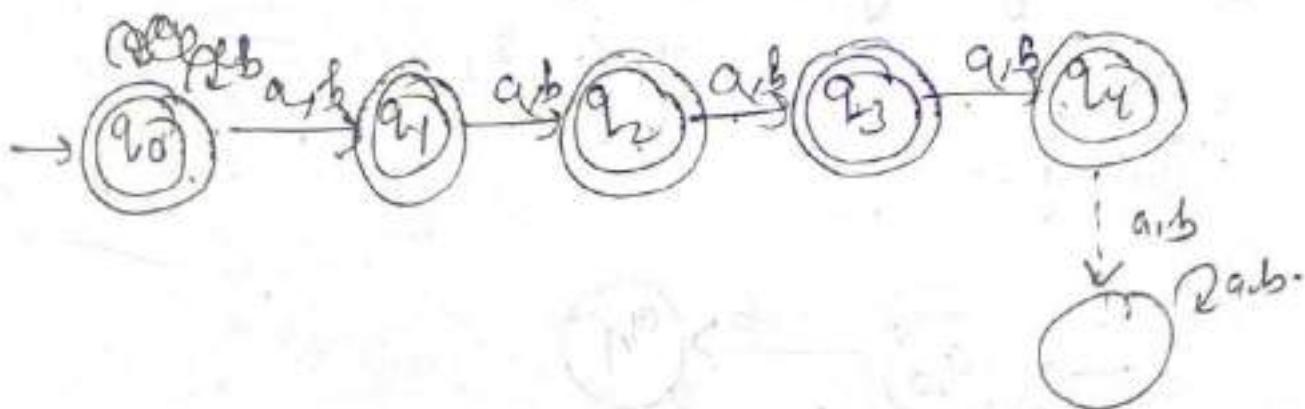
Q. At least 2 b's. $\Sigma = \{a, b\}$

$L = \{bb, abb, aabb, a...abb, \frac{abb...b}{bab} \}$



Q. At most 4 length string.

$L = \{\epsilon, a, b, ab, ba, aab, aaa, bab, aaa, bbbb\}$

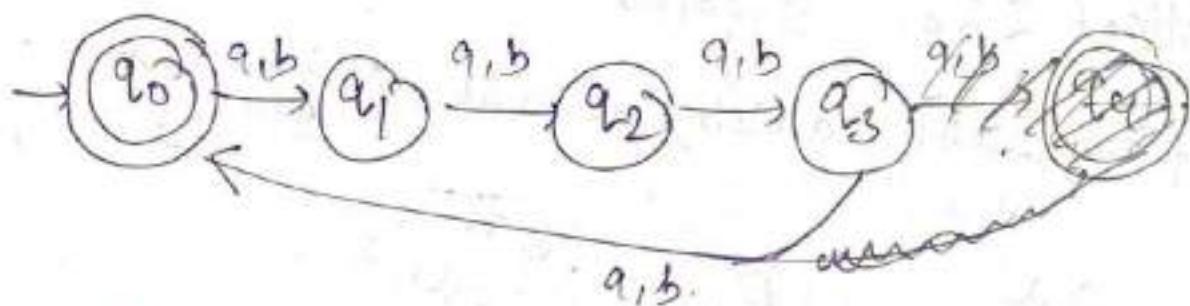


$m \geq 2$

Note:-
 Length of the string at most ' m ', minimal DFA requires $\lceil \frac{m+2}{2} \rceil$ states.

Q. Length of the string divisible by 4.

$$0, 4, 8, 12, 16, \dots$$



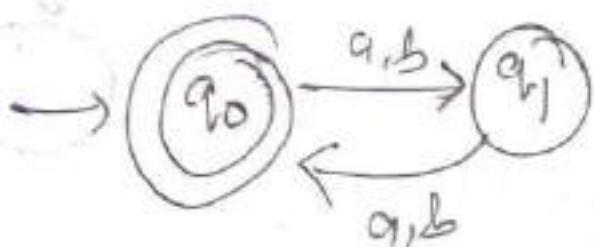
Note:-

Length of the string divisible by 'n' minimal DFA requires ' n ' states.

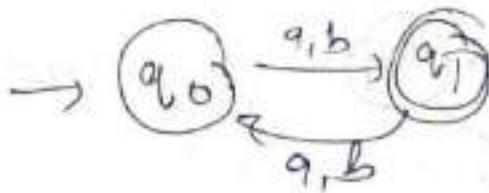
Q. Length of the string even.

even
means
divisible by 2

$$0, 2, 4, 6, 8, \dots$$



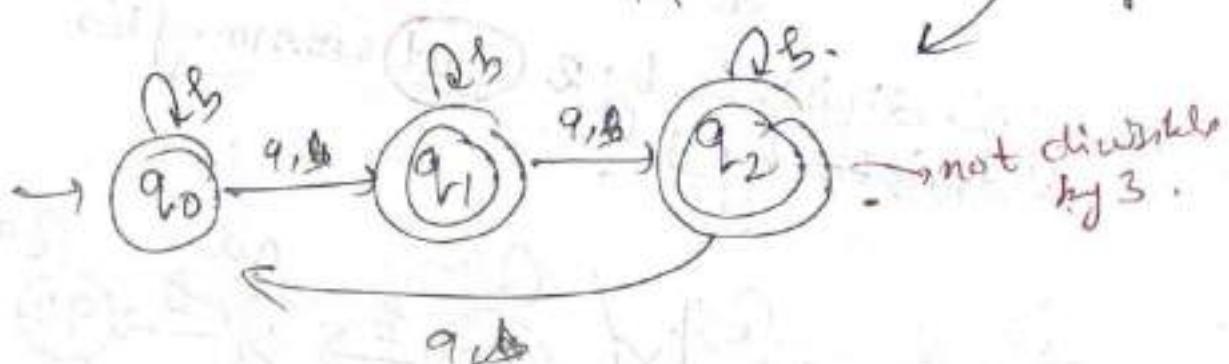
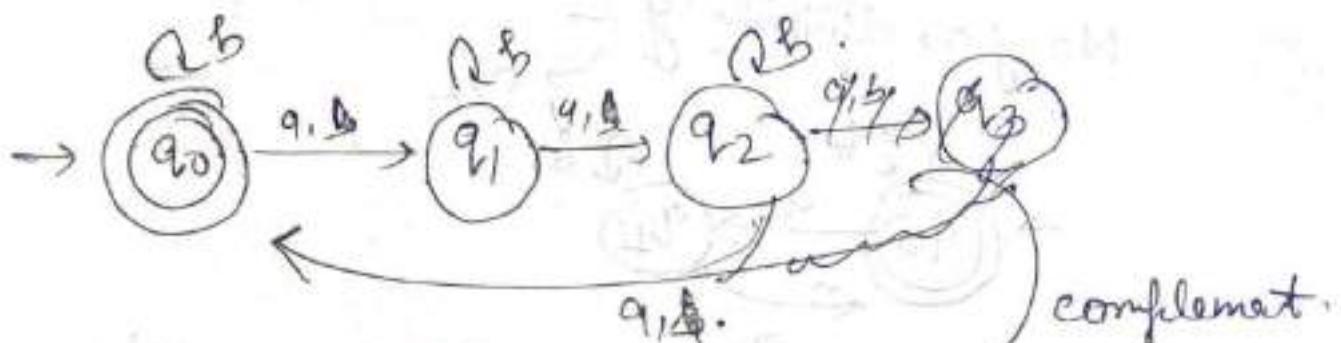
length of the string odd) -
complement of above.



Q. No. of a's not divisible by 3 $\Rightarrow \Sigma = \{a, b\}$

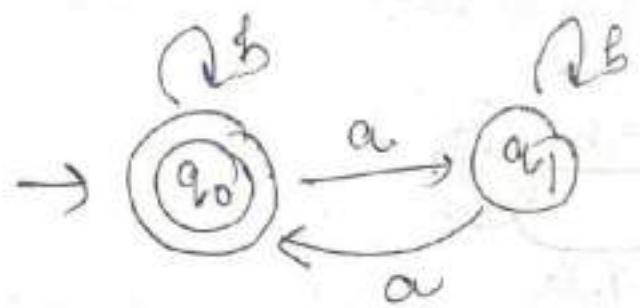
soln. No. of a's divisible by 3 \rightarrow

0, 3, 6, 9, 12, -



Q. No. of a's even -

0, 2, 4, 6, 8, ...

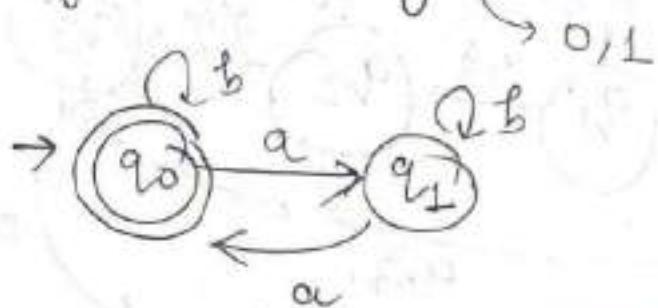


ϵ, aa, b, bab

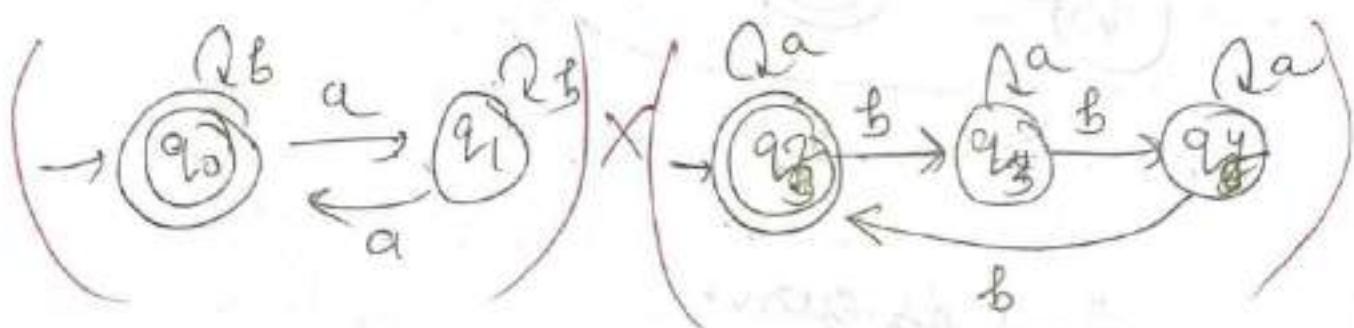
- Q. Construct the minimal DFA that accept all strings of a's & b's where no. of a's divisible by 2

sol^m

No of ab divisible by 2



- Q. No. of ab divisible by 2 and no. of bb divisible by 3.



a's divisible by 2

b's divisible by 3

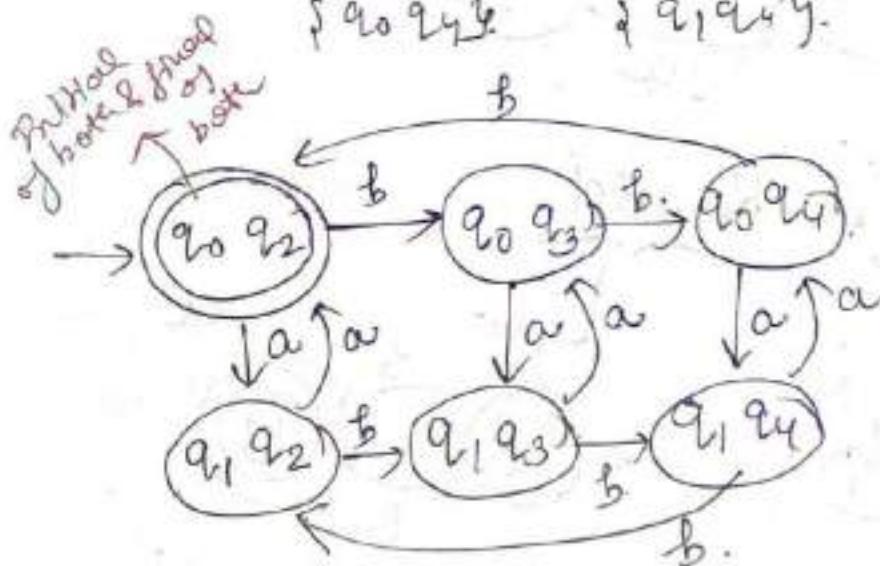
59

Hence we said and so, we use cross-product.

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

$$\begin{array}{ll} \{q_0 q_2\} & \{q_1 q_3\} \\ \{q_0 q_3\} & \{q_1 q_4\} \\ \{q_0 q_4\} & \{q_1 q_2\} \end{array}$$

- Make Initial which is initial of both of above.

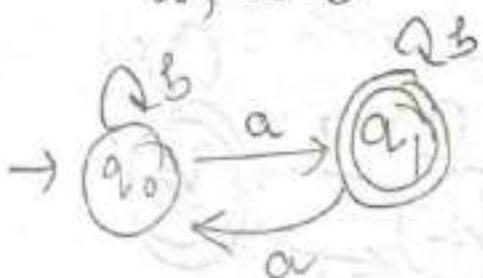


- Make final in which all are final of both of above.

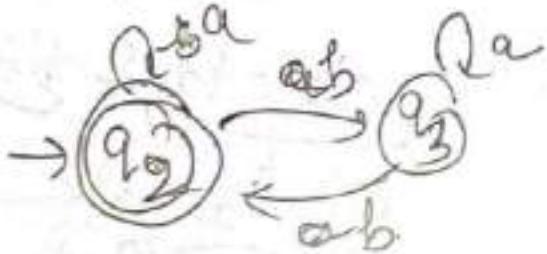
$$\begin{array}{l|l} (q_0, a) = q_1 & (q_0, b) = q_0 \\ (q_1, a) = q_2 & (q_0, b) = q_3 \\ \hline \end{array}$$

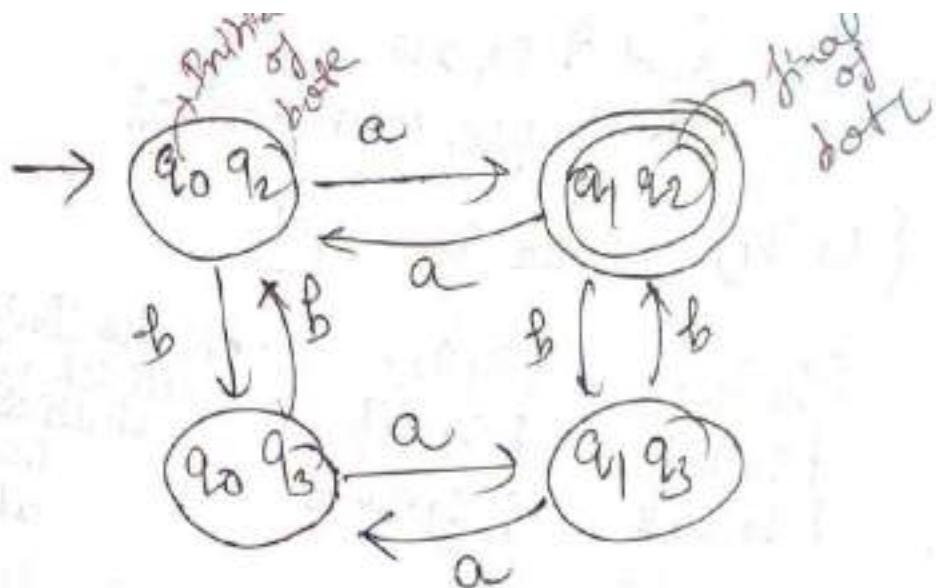
Q. No. of ab odd (and) no. of bb even.

so, No. of ab odd :-
a, aaa

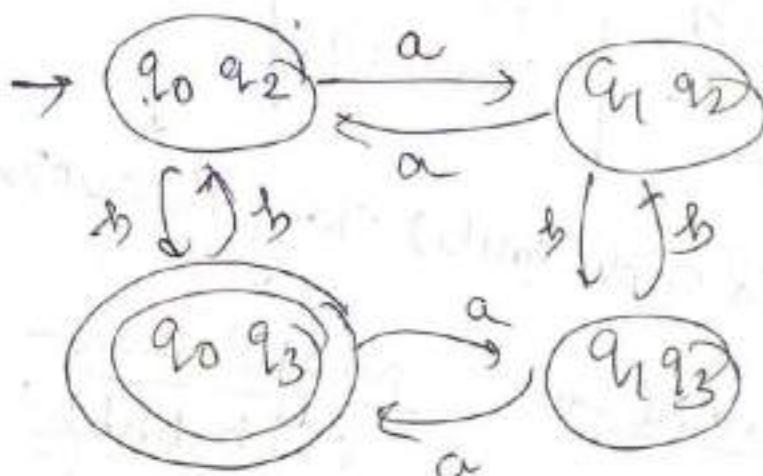
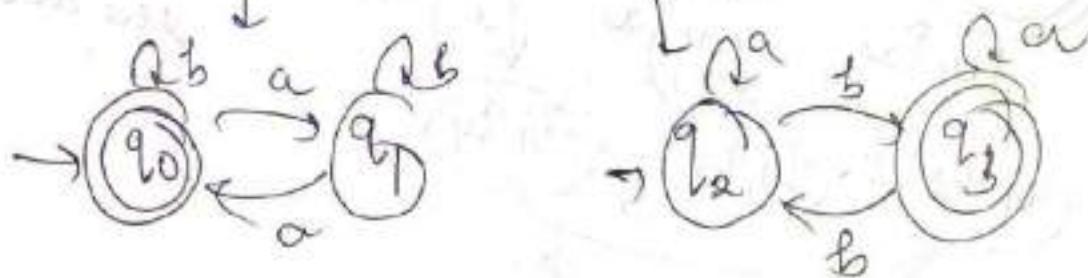


No. of bb even -
b, bb, bbbb

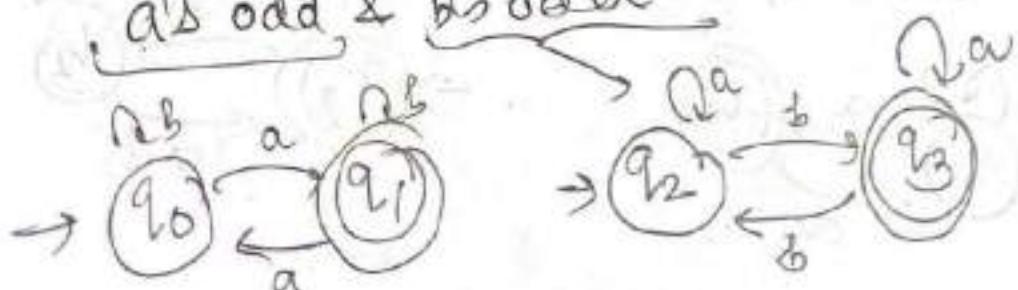




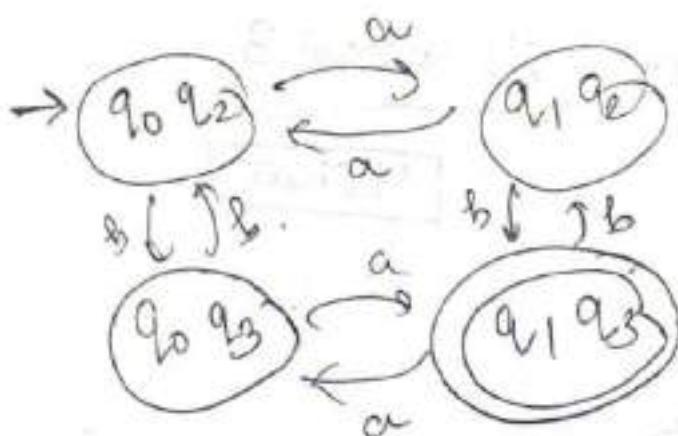
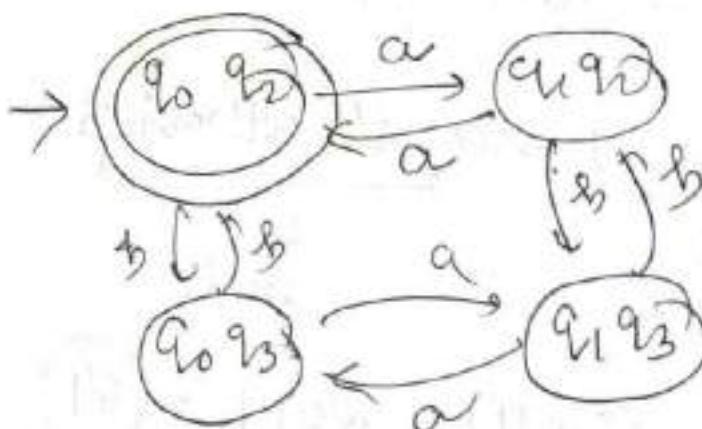
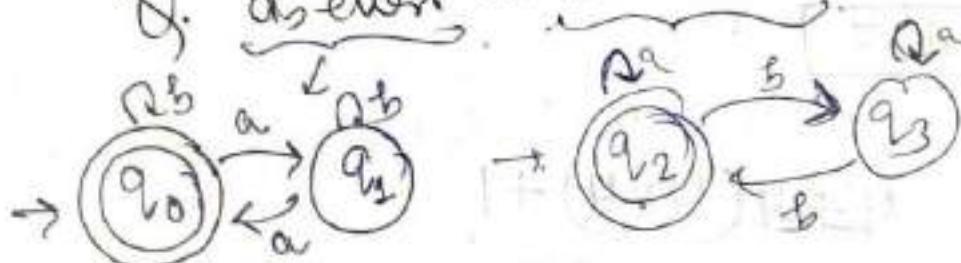
Q. a 's even & b 's odd.



Q. a 's odd & b 's odd



(4b)

Q. odd even & b even:Q. No. of states,⑤ a's odd and b's atleast 2 $\Rightarrow 6$ ⑥ a b atleast 3 and b's atleast 4 $\Rightarrow 20$ ⑦ a b divisible by 4 and b's atleast 2,

$$4 \cdot 4^3 = 64 \Rightarrow (12-1) \cdot 64 = 112$$

④ q's exactly 3 and b's atleast 3

It have

5 state

with one
dead state

so remove

the dead state

4 state

5 4

4 state

$$\Rightarrow \cancel{4 \times 4} (4 \times 4) + 1$$

$$\Rightarrow 16 + 1 = 17$$

⑤

a'b atmost 4 and b'b atleast 3

5
state

4
state

$$(5+4) = 20 + 1 = 21$$

⑥

a'b atmost 3 and b'b exactly 4

4

*

5

$$1 (20+1) = 21$$

Q. 0's exactly 4 and $\underbrace{\text{at least } 2 \text{ 1's}}$. (41)

$$5 * 3 = 15 + 1$$

= 16.

- When we have (and), we ~~product~~ apply cross product of both. so, it produce always minimal DFA.

No. of ab divisible by 6 (and) no. of ~~bs~~
~~not divisible by 3~~

16 states

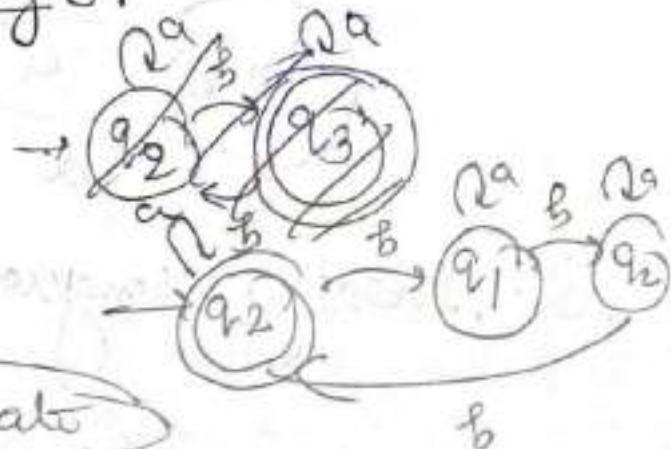
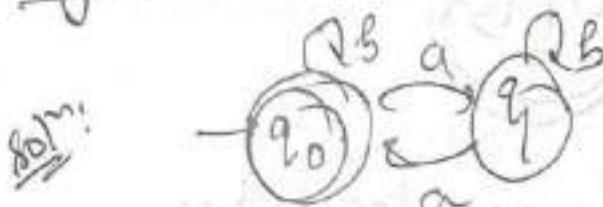
6 states

$$\Rightarrow 8 * 6 = 48.$$

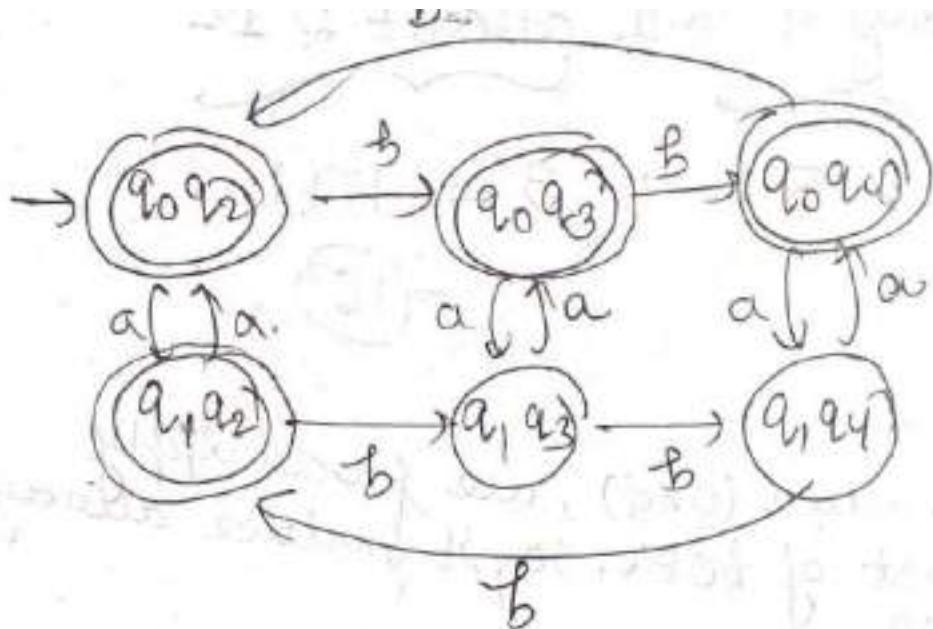
8 state
(because

of complement
no change
in state).

- Q. Construct the minimal DFA that accept all strings of ab & bb, no. of ab divisible by 2 ~~bb divisible by 3~~.

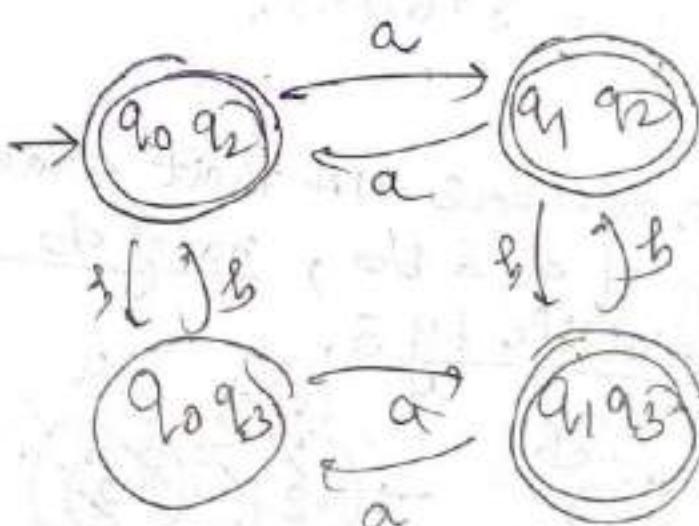
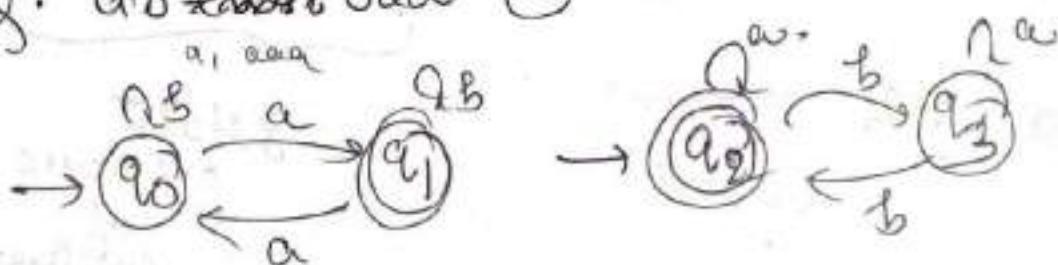


Total 6 State

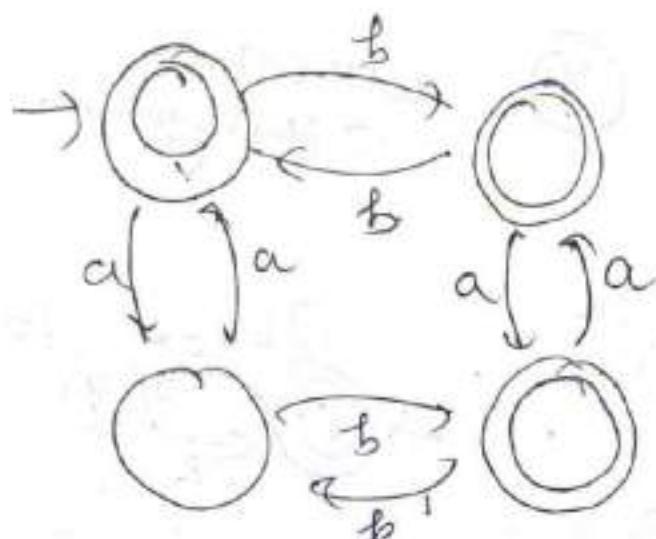


- If any one is final so make it final not necessary both the states be final.

Q. also even odd or b's even.



Q. Identify language accepted by given DFA.

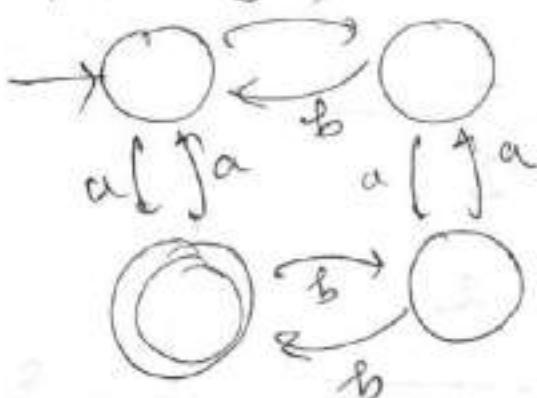


(42)

a even	b odd
a b	✓
a b	odd
a b b	even

- (a) ab even bb even
- (b) ab even bb odd
- (c) ab odd bb even
- (d) ab odd bb odd.

complement of above:-



ab odd & bb even

its complement

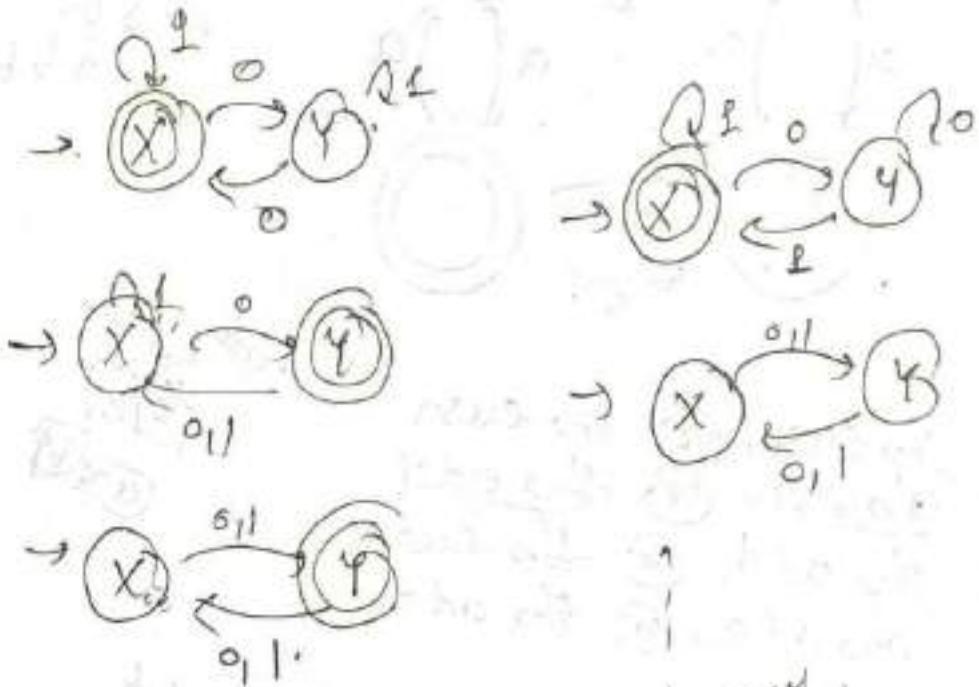
ab even & bb odd

Q. How many max. no. of DFAs are possible
with two states x and y
over the input alphabet $\Sigma = \{0, 1\}$.

89m



with 2 state
2 input symbol.



many

	0	1
→ x	2	2
y	2	2

total $\rightarrow (16)$.

(0, 218, 9018).

when initial state is x,

- | | |
|--|--------------------------------------|
| | $\Rightarrow 16$ (both are nonfinal) |
| | $\Rightarrow 16$ (x is final) |
| | $\Rightarrow 16$ (y is final) |

$\rightarrow \textcircled{X} \quad \textcircled{Y} \Rightarrow 16 \quad \{ \text{both or final}\}$

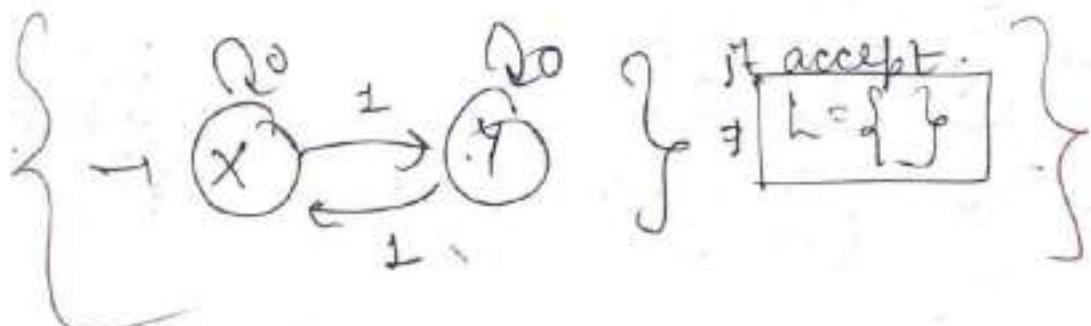
(43)

$$16 \times 4 = 64$$

$64 \rightarrow$ also for when Y is initial

So,

$$\boxed{\text{Total} = 64 + 64 = 128}$$



Q. How many max. no. of F.A. possible. when
3 states x, y, z with input alphabet a, b .

Sol:

		a	b
2	x	3	3
2	y	3	3
2	z	3	3
Total		$9 \times 3 \times 9 = 729$	

$$\frac{81}{9} \\ 9^2$$

~~$m \times m$~~
 ~~$2^n \times 2^n$~~
 ~~2^n~~

when X is initial

$$\rightarrow \textcircled{X} \textcircled{Y} \textcircled{Z} \Rightarrow 729$$

$$\rightarrow \textcircled{X} \textcircled{Y} \textcircled{Z} \Rightarrow 329$$

$$\rightarrow \textcircled{X} \textcircled{Y} \textcircled{Z} \Rightarrow 729$$

$$\rightarrow \textcircled{X} \textcircled{Y} \textcircled{Z} \Rightarrow 229.$$

$$\rightarrow \textcircled{Y} \textcircled{Y} \textcircled{Z}$$

$$\rightarrow \textcircled{X} \textcircled{Y} \textcircled{Z}$$

$$\rightarrow \textcircled{X} \textcircled{Y} \textcircled{Z}$$

$$\rightarrow \textcircled{X} \textcircled{Y} \textcircled{Z} - \frac{5832}{5832}$$

$$\begin{array}{r} 329 \\ 2 \times 3 \\ \hline 5832 \end{array} 7$$

when all X, Y, Z are initial one by one.

In this
1st case when
 X is initial,
2nd case when
 Y is initial,
3rd case when
 Z is initial

- Q. How many max. no. of DFA's possible with m^n states over the input alphabet contains m symbols.

(49)

	1	2	3	\dots	m
--	---	---	---	---------	-----

$2/q_1$

$n \ n \ n \ \dots \ n$

$2/q_2$

$n \ n \ n \ \dots$

\vdots

\vdots

$2/q_m$

$n \ n \ n \ \dots \ -$

(2^n)

$(\cancel{q_n}^{m^n})$

$n \times 2^m$

$q_1 \rightarrow 2$

$q_2 \rightarrow 2$

$q_m \rightarrow 2$

$m \times 2^n + 2^n m^n$

$\boxed{m \times 2^n + n^m}$

Q. How many max. no. of DFAs are possible with 100 states x and y over the input alphabet $\Sigma = \{0, 1\}$ that accept empty language only.



2^{100}
without
final

$00 \rightarrow x$

y

$\rightarrow 16$

x

y

$\rightarrow 16$

$\frac{1}{32}$ (but it is not answer)

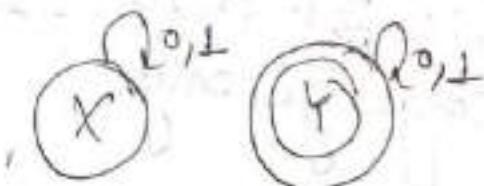
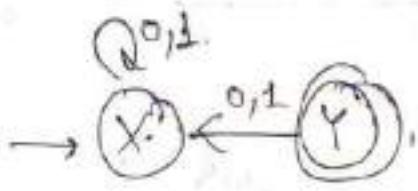
Note ①

If there is no final state in tree DFA,
the DFA will accept empty language
only.

Note ②

In any DFA if all states are final
then DFA accept complete language
only.

~~left
final~~



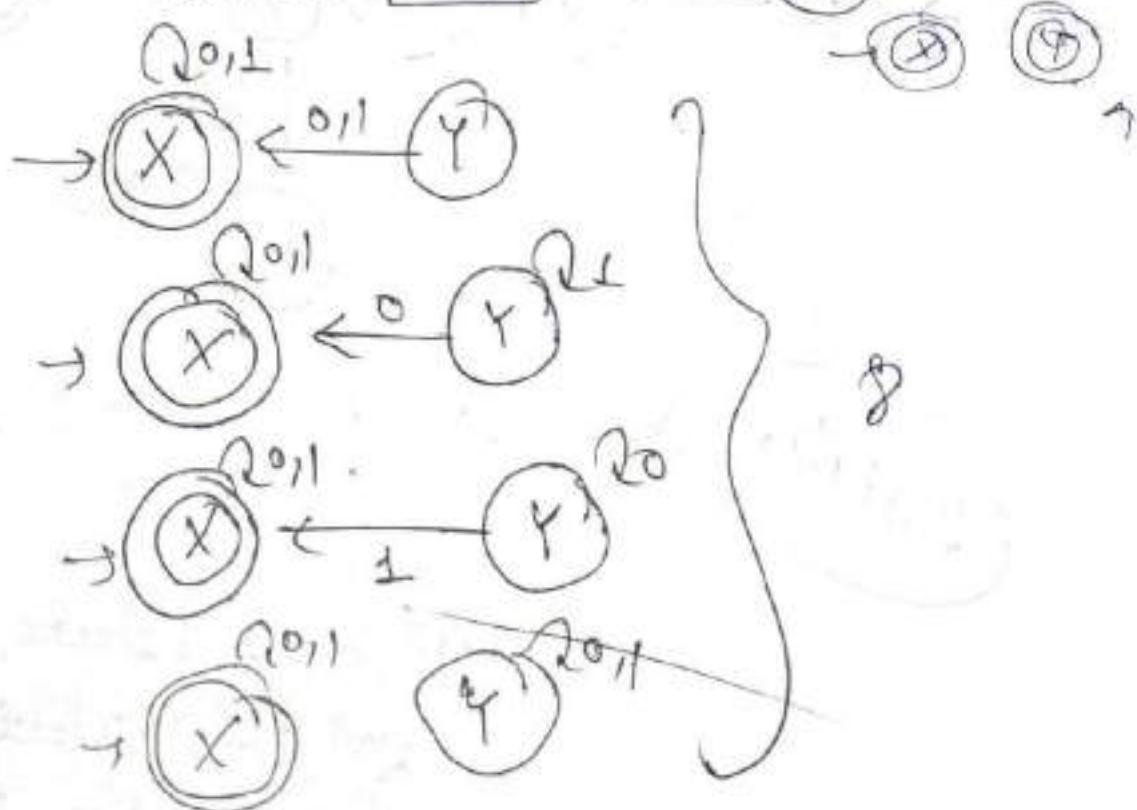
$$\boxed{\text{Total} = 3 \cdot 2 + 8 = 40}$$

Q. How many max. no. of DFA are possible
 with 200 states randomly.
 over the input alphabet 0, 1,
 that accept non-empty language. (45)

Sol: $128 - 40$
 $\rightarrow \boxed{88}$ ~~A~~

Q. How many max. no. of DFAs possible
 with 2 states n and y over the
 input alphabet 0, 1 that accept
complte language only.

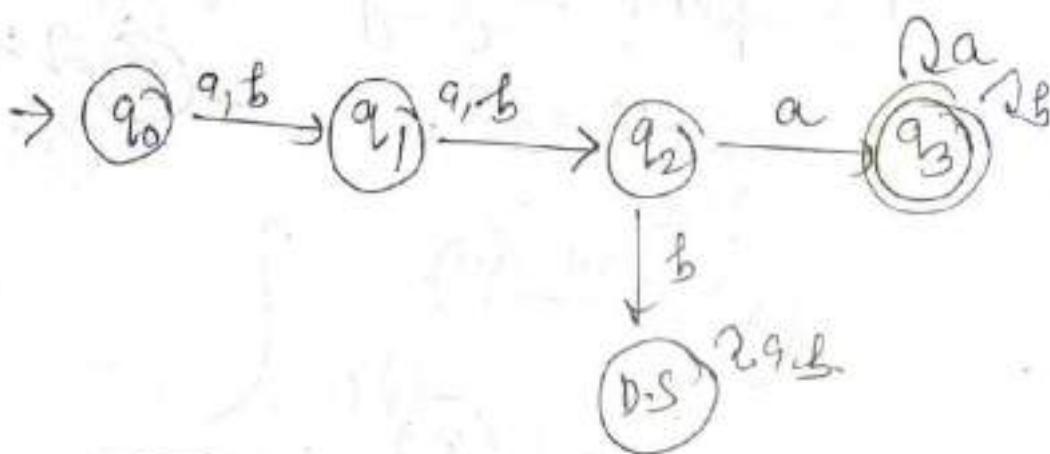
Sol: $16 + 16 = 32$



$32 + 8 = \boxed{40}$ ~~A~~

Q. Construct the minimal DFA that accept all strings of ab & b's where 3rd input symbol is a while reading the string from left hand side.

Sol: $L = \{ bba, aab, bb\bar{aaa}a, \bar{bab}a, \bar{ba}a, abab, baaba, baababa, abaabb\bar{a}a. \dots \}$



5 states

$3^{\text{rd}} a \rightarrow 5 \text{ state}$

$2^{\text{nd}} a \rightarrow 4 \text{ stat}$

$\frac{1^{\text{st}} a \rightarrow 6 \text{ stat}}{n \rightarrow n+2}$

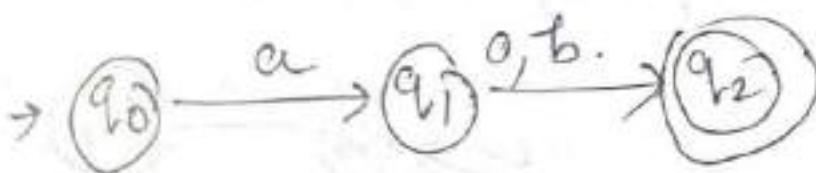
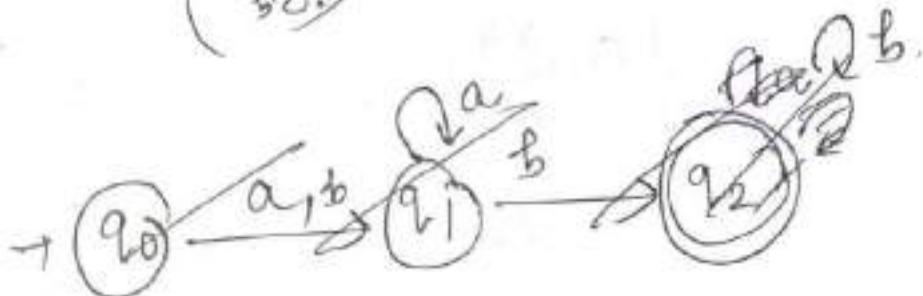
Note :

The minimal DFA that accept particular
 $(m+1)$ input symbol from left hand side
 requires $(m+2)$ strings.

Q. Construct the minimal DFA that accept
 all string of a & b's where 2nd input
 symbol is a while reading string
 from right side

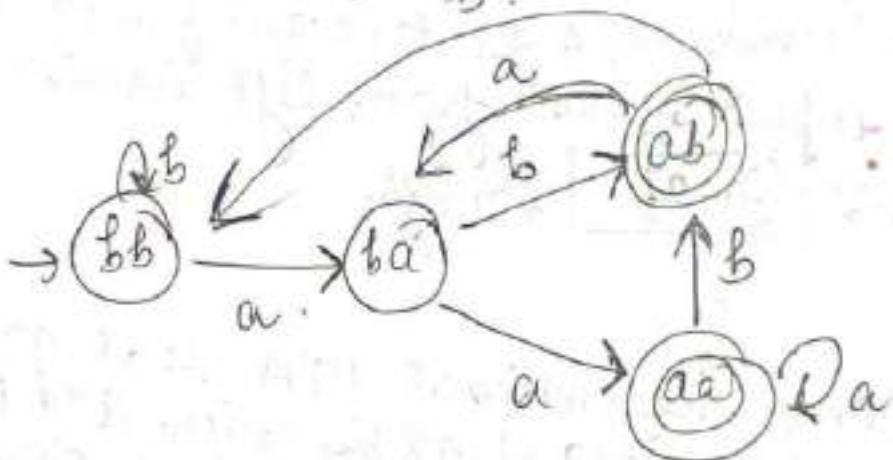
Soln. $L = \{ ab, aa, aab, aba, bab, \dots \}$
 $L = \{ aa, ab, aab, aba, bab, ababababa, \dots \}$

(als)
 (sol.) $\xrightarrow{a, a/b}$



aaaab ✓

aaaabb ✗ . b.



ab
aabb

aab
aab

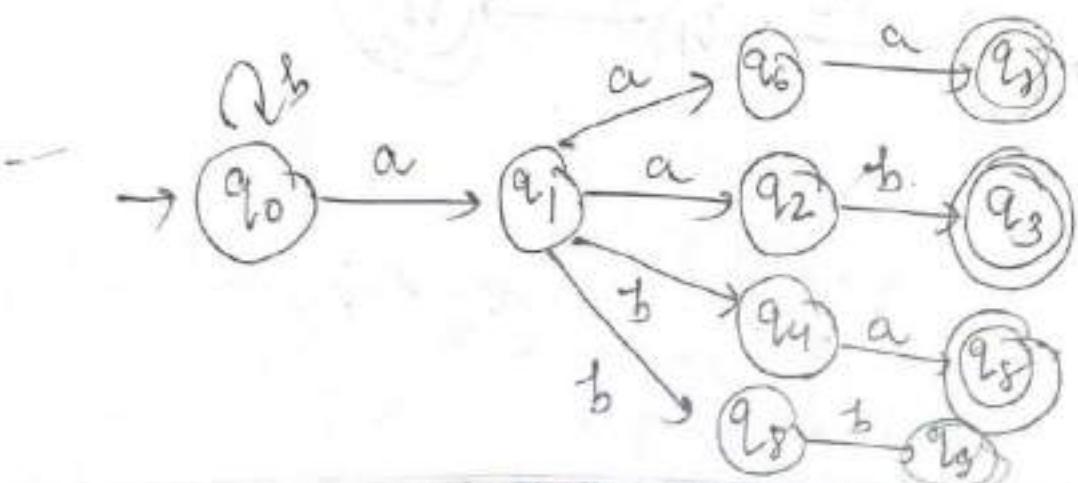
aaa
bab

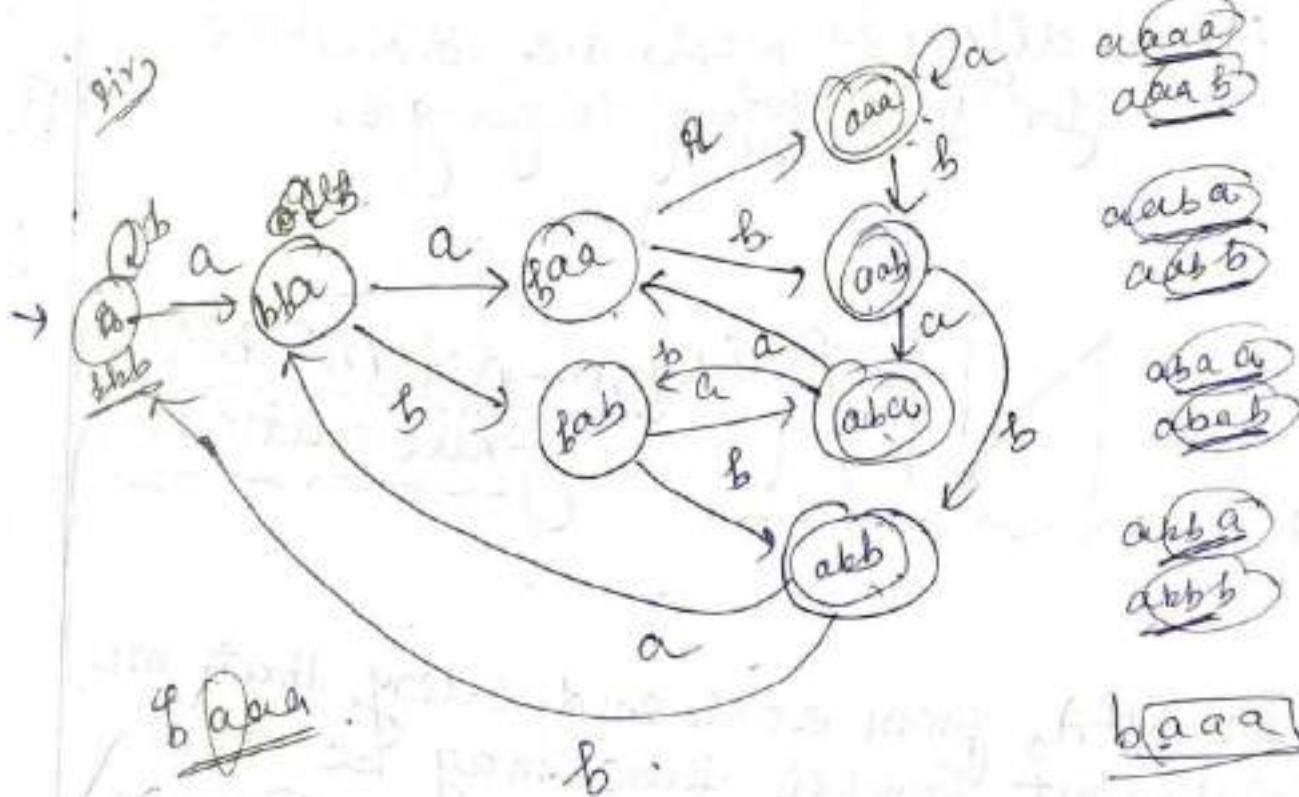
Q. 3rd input symbol is a from right hand side?

Sol: $\frac{aabb}{aab} \quad \frac{aab}{bab}$

$\begin{matrix} ab \\ ab \\ ba \\ bb \end{matrix}$

L = { aaa, aab, aab, aabb, ... }





R.H.S :-

Input symbol $2 \rightarrow 4$
 $3 \rightarrow 8$
 $4 \rightarrow 16$
 $5 \rightarrow 32$
 $n \rightarrow 2^n$ states

Note :

- The minimal DFA that accept all strings of $a^m b^m$ where m th input symbol from R.H.S requires 2^n states.
- For any language the construction of

DFA is difficult NFA's are constructed
for such type of languages.

NFA (Non-deterministic finite automata)

- In NFA, from each and every state on every input symbol there may be zero no. of transitions or one transition or more than one transition may exist. NFA is formally defined as,

$$\text{NFA formal def} = (Q, \Sigma, \delta, q_0, F)$$

Q : finite no. of states

Σ : input alphabet

δ : transition function.

q_0 : initial state.

F : set of final state.

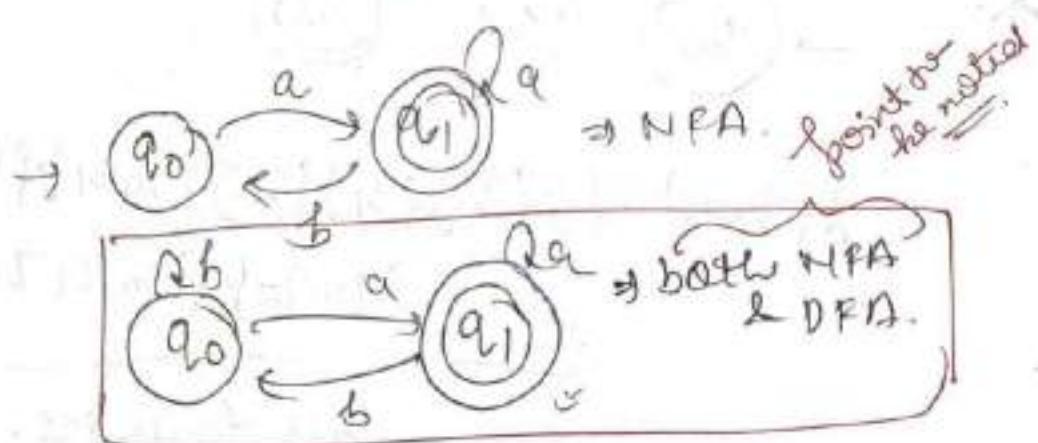
$$\delta: Q \times \Sigma \rightarrow P(Q) \text{ or } 2^Q$$

- ① Construction of NFA is easier compare to DFA.

- ✓ (1). Language detection is easier in DFA compare to NFA.
- (2). Complementation not possible for NFA.
- (3). Minimization not possible for NFA.
- ✓ (4). In NFA, even for valid strings also there may be non-final states possible to reach.

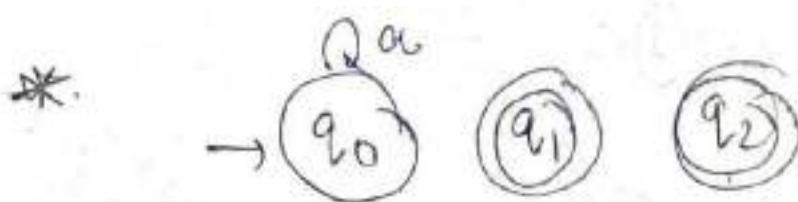
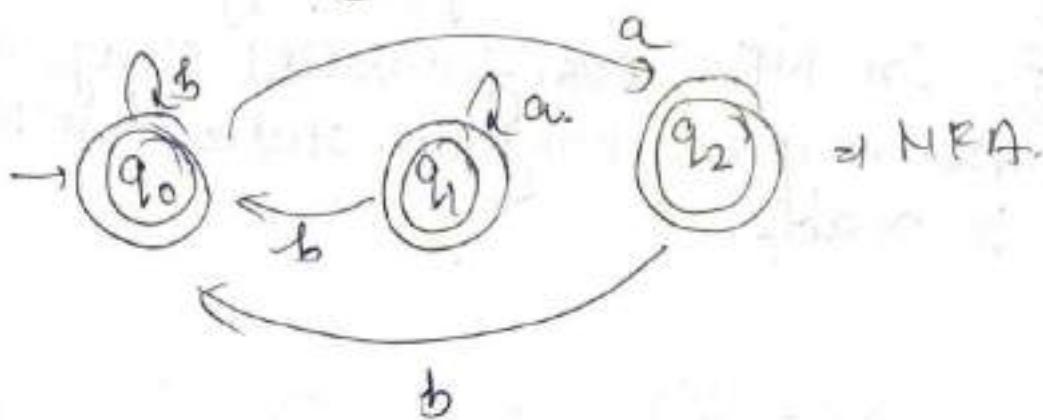
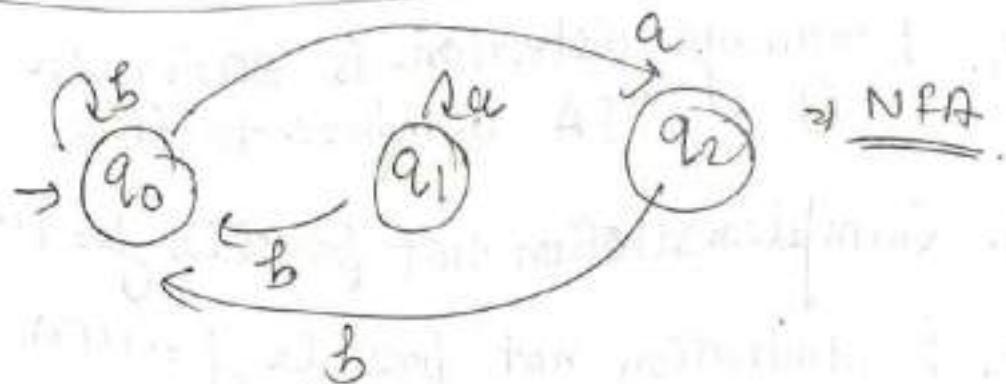
DFA: $Q \times \Sigma \rightarrow \{1\}$

NFA: $Q \times \Sigma \rightarrow \{0, 1, \emptyset\}$



- Every DFA is NFA but every NFA need not be DFA.

(Every deterministic is non-deterministic but every non-deterministic need not be deterministic)

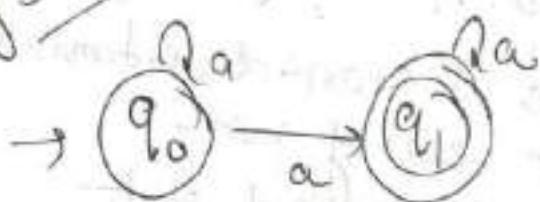


$$\delta(q_0, a) = \{ \{q_0\}, \{q_1\}, \{q_2\}, \{q_0, q_1, q_2\} \}$$

{ } { } { } { } .

all four set.

~~Affiliation of state point.~~



$$(q_0, a) = q_0 \} \text{ non}$$

$$(q_0, a) = q_1 \} \text{ deterministic.}$$

ex final accept string is accepted.



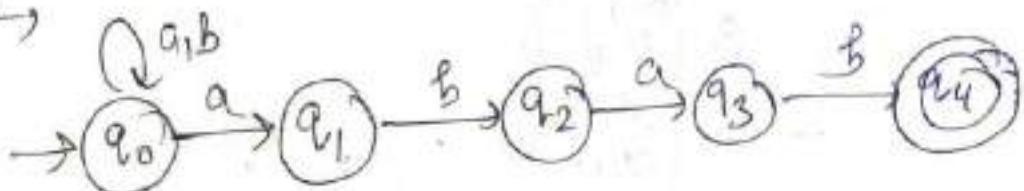
Q. Construct the NFA that accept all strings of a 's & b 's, where each string ending with abab.

Sol:

$\{abab\} \cup \{babab\} \cup \dots$

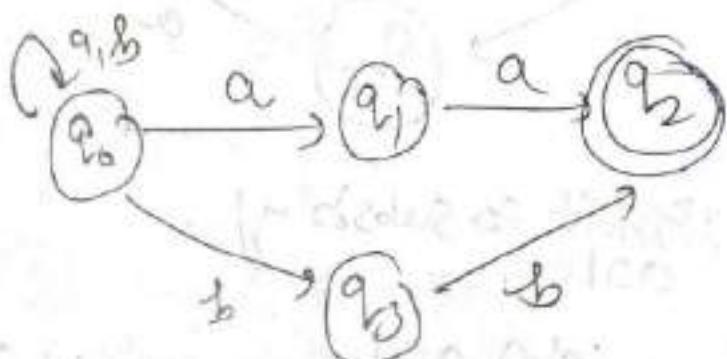
DFA, $L = \{abab, aabab, babab, \dots\}$.

NFA,



Q. last two symbols are same.

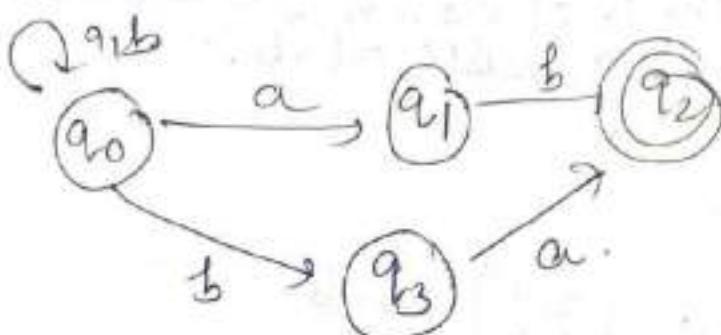
$\{ab\} \cup \{aa\} \cup \{bb\}$



Q. last two symbols are different.

$$\begin{pmatrix} a \\ b \end{pmatrix} ab$$

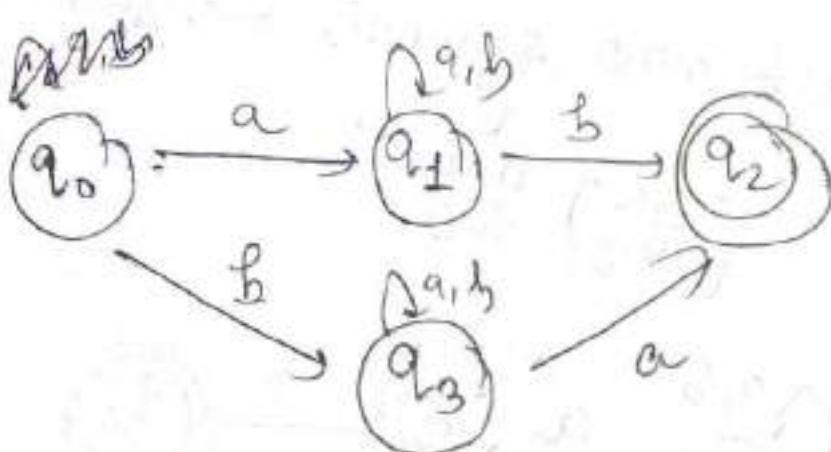
$$\begin{pmatrix} b \\ a \end{pmatrix} ba$$



Q. starting & ending w/ diff symbol.

$$a \begin{pmatrix} a \\ b \end{pmatrix} b$$

$$b \begin{pmatrix} a \\ b \end{pmatrix} a$$



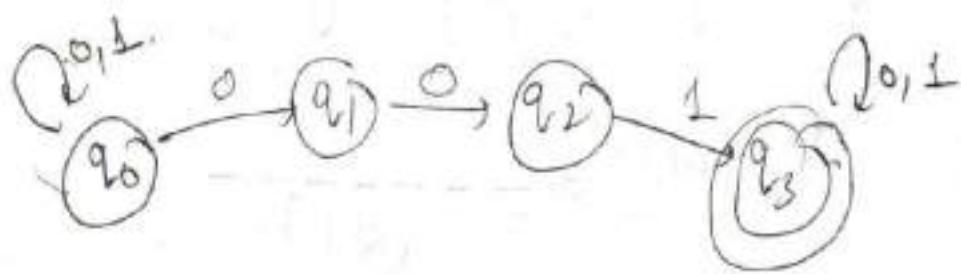
Q. ~~abit as subseq~~
001

$$\begin{pmatrix} a \\ b \end{pmatrix} 001$$

$$001 \begin{pmatrix} a \\ b \end{pmatrix}$$

$$(a) 001 \begin{pmatrix} a \\ b \end{pmatrix} .$$

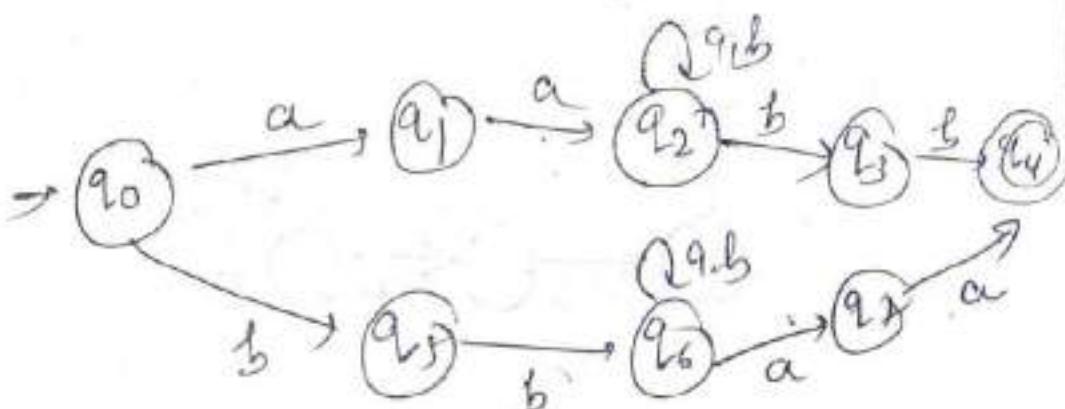
(51)



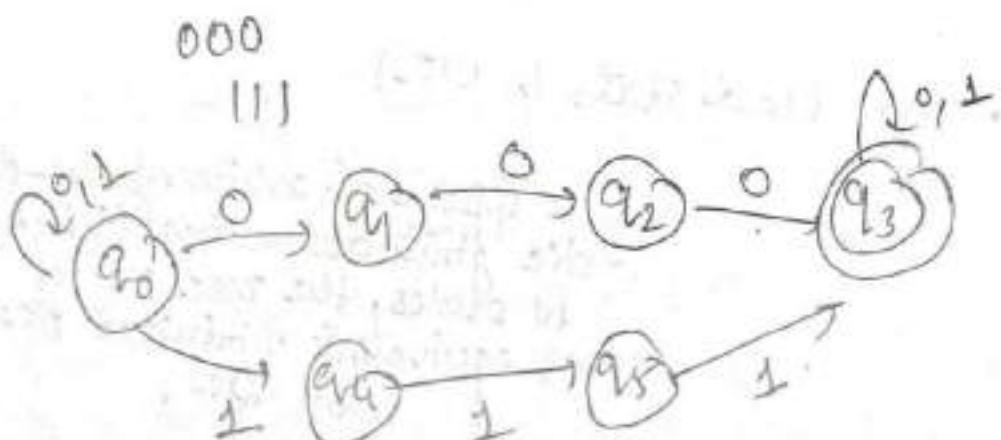
Q. Starting with aa then it should end with bb.

~~Q. Starting bb then should end with aa.~~

aa (q_1) bb bb (q_1) aa



Q. Primitve substring.

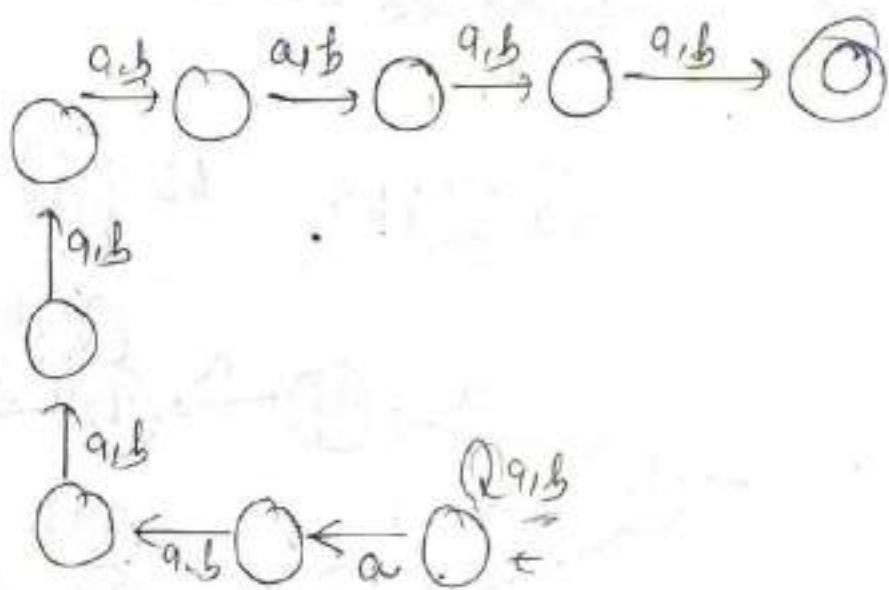


Q. get input symbol from R.H.S is a.

Ans:

$$(a/b) \xrightarrow{a} \overbrace{\quad}^{(2^f)}$$

(ld)



DFA: - 2^n
NFA \rightarrow (n+1)

Stack for R.H.S.

- No dead states in NFA.

Given an arbitrary non-deterministic finite automaton (NFA) with N states, the max. no. of states in an equivalent minimized DFA is at least 2^N .

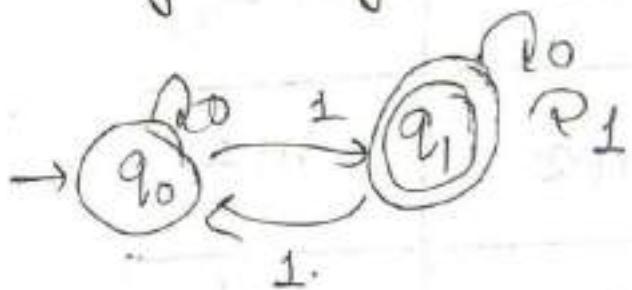
	<u>BORAS</u>	<u>DFA</u>	<u>NFA</u>
①. $ w =n$		$n+2$	$n+1$
②. $ w \leq n$		$n+2$	$n+1$
③. $ w \geq n$		$n+1$	$n+1$
④. $ w \bmod n = 0$		n	n
⑤. n^{th} input L.H.S a		$n+2$	$n+1$
⑥. n^{th} input R.H.S a		2^n	$n+1$

Construction from NFA to DFA :-

$\text{NFA} \Rightarrow \text{DFA}$ (Subset construction algorithm)

it does not give minimal DFA.

Q. Construct DFA equivalent to following NFA.



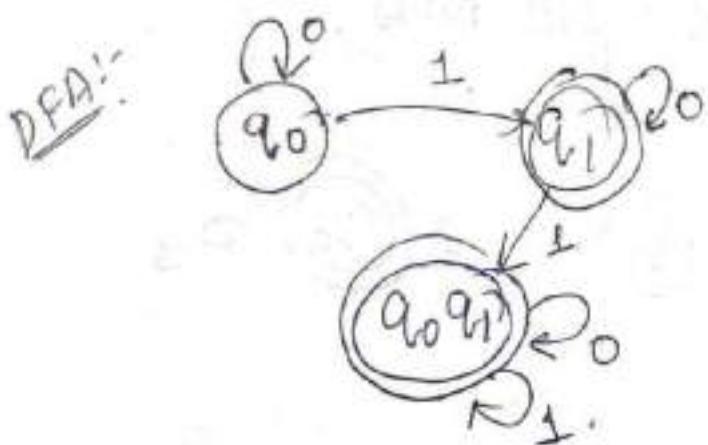
NFA:

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

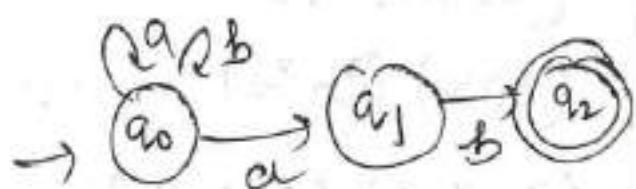
DFA:

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

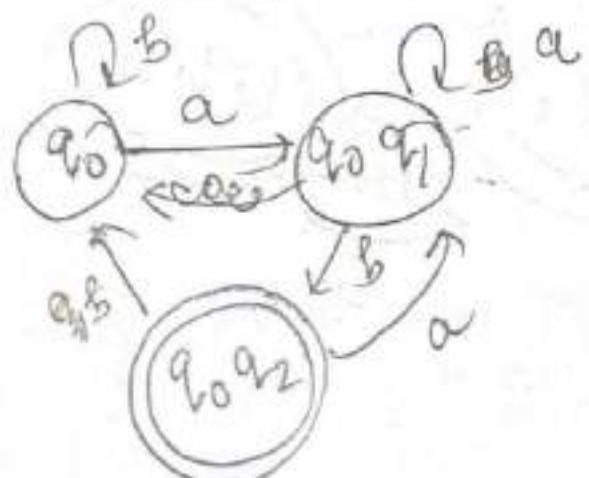
whenever
we get
it final.



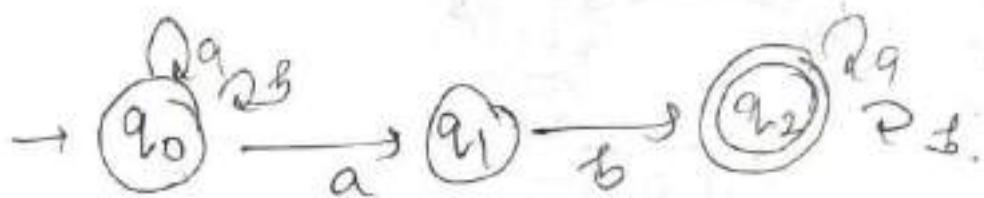
Q. Construct the DFA



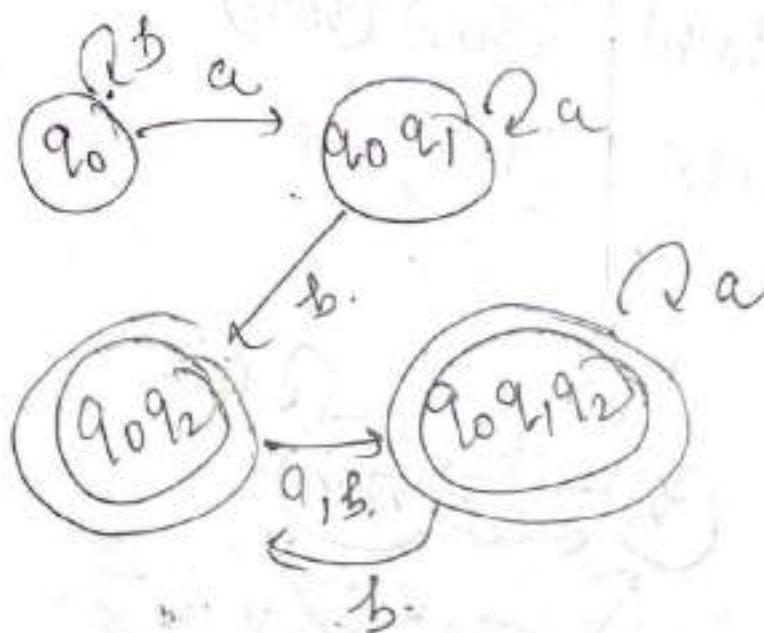
	a	b
q0		(q0q1) q0
(q0q1)		(q0q1) (q0q2)
(q0q2)		(q0q1) q0



Q. Construct the minimal DFA for the foll NFA,



	a	b
q_0	$(q_0 q_1)$	q_0
$(q_0 q_1)$	$(q_0 q_1)$	$(q_0 q_2)$
$(q_0 q_2)$	$(q_0 q_1 q_2)$	$(q_0 q_1 q_2)$
$(q_0 q_1 q_2)$	$(q_0 q_1 q_2)$	$(q_0 q_2)$



(54)

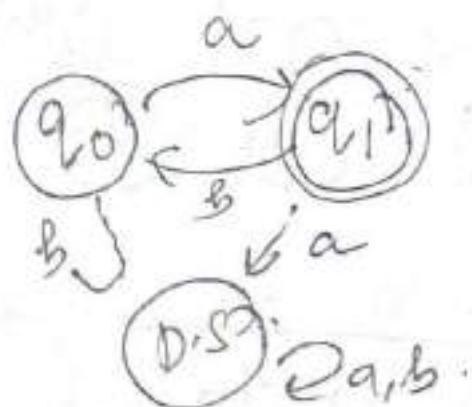
$$\{q_0q_1, (q_0q_1)\} \cap \{(q_0q_1), (q_0q_1q_2)\}.$$

$$\boxed{\{q_0\} \cup \{q_0q_1\} \cup \{(q_0q_1), (q_0q_1q_2)\}}.$$

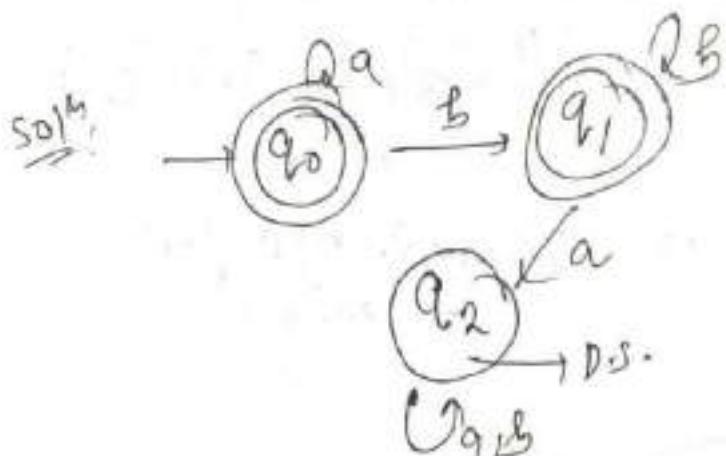
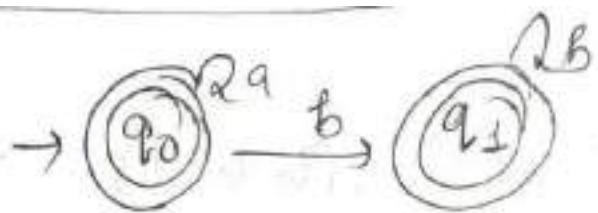
Q. Construct the minimal DFA for
NFA,



	a	b
q0	q1	-
q1	-	q0

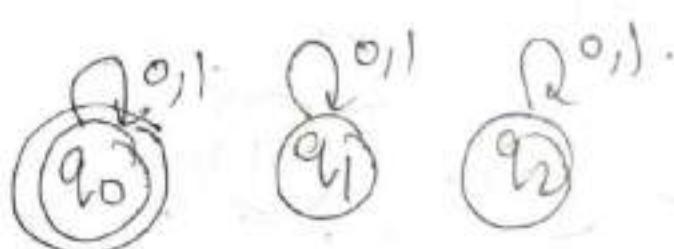


Q. Construct the minimal DFA,



Q. Construct the minimal DFA, for NFA

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



	0	1
q_0	q_0	q_0



NFA \Rightarrow DFA

(55)

$$2 \rightarrow 3$$

$$3 \rightarrow 3$$

$$3 \rightarrow 4$$

$$2 \rightarrow 3$$

$$3 \rightarrow 0^1$$

$$n \rightarrow 2^n \text{ states}$$

$$\Rightarrow \{1, \dots, 2^n\}$$

$$\begin{array}{c} n \rightarrow \text{NFA} \\ \downarrow \\ \text{DFA} \quad \frac{1}{2^n} \end{array}$$

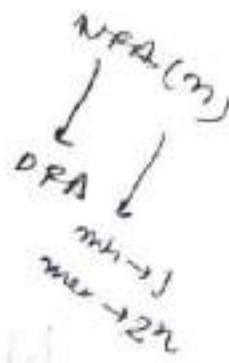
~~V.P.M.~~

Note

while converting from NFA to DFA max. no. of states possible in DFA for n state
NFA $\Rightarrow \{1 \text{ to } 2^n\}$.

- The time complexity of the algorithm is
 - $O(1)$ in best case.
 - $O(2^n)$ in worst case.

ϵ -NFA ϵ -NFA :-



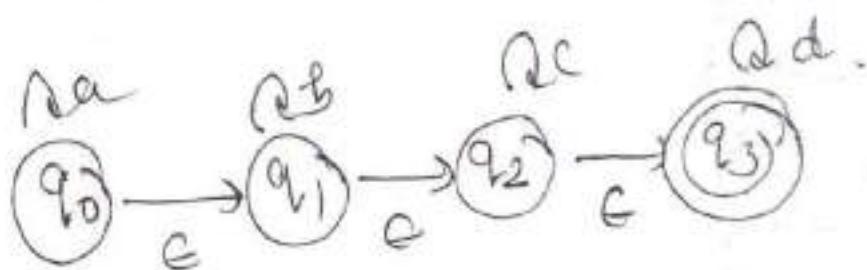
DFA
~~diffficult~~ ||

X NFA

||
εNFA

here not diff. to construction
new algo. exist.

$$L = \{a^n b^m c^k d^l \mid n, m, k, l \geq 0\}$$



Note:

NFA with ϵ -transition known as ϵ -NFA:

Formal definition \rightarrow

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

$$\boxed{\delta: Q \times \Sigma \cup \{\epsilon\} \rightarrow 2^Q}$$

* ϵ -NFA

\downarrow

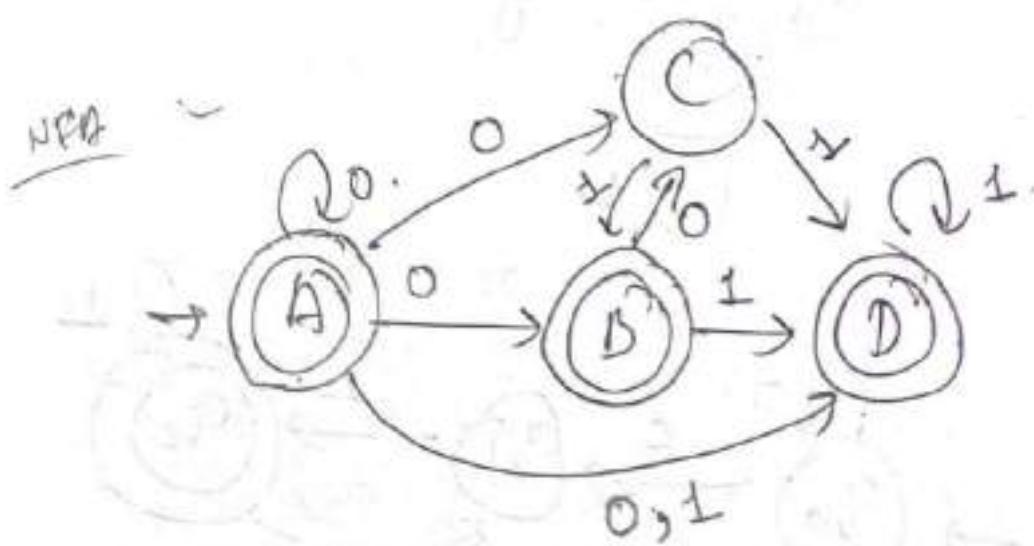
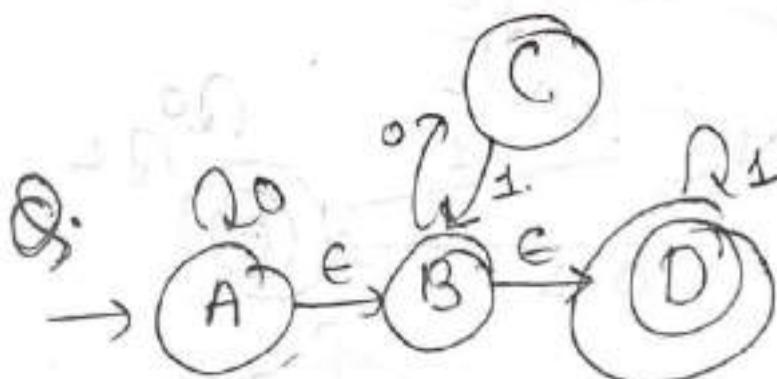
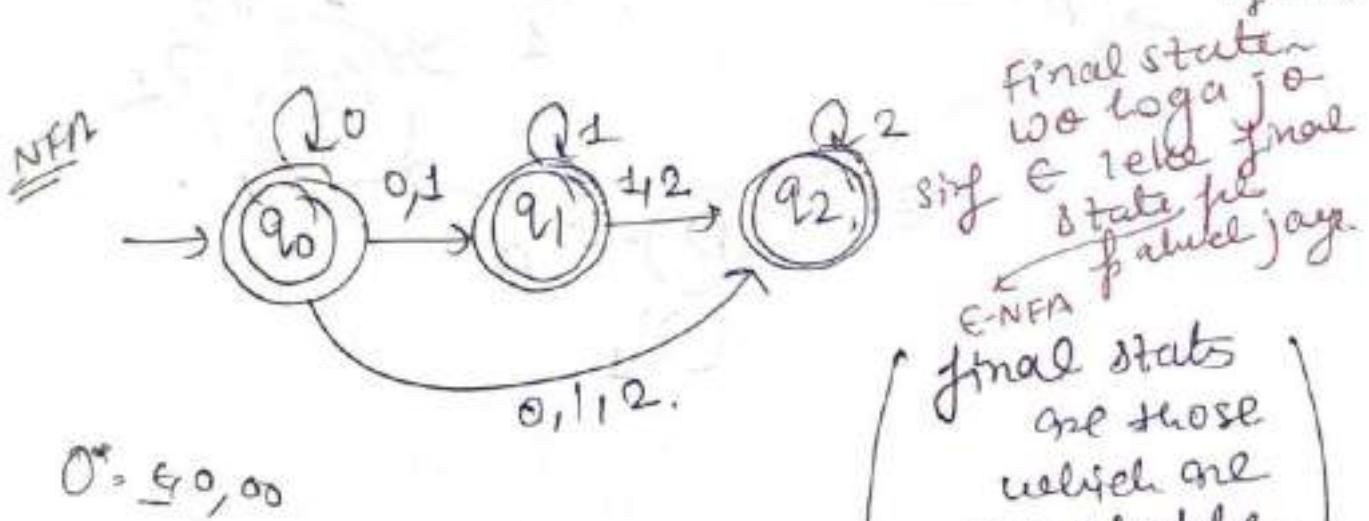
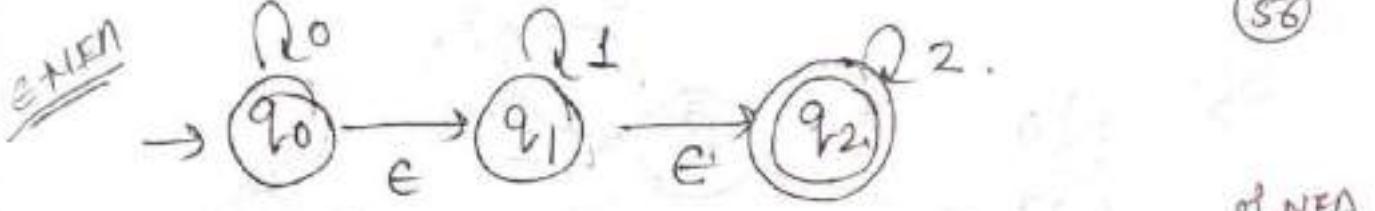
NFA

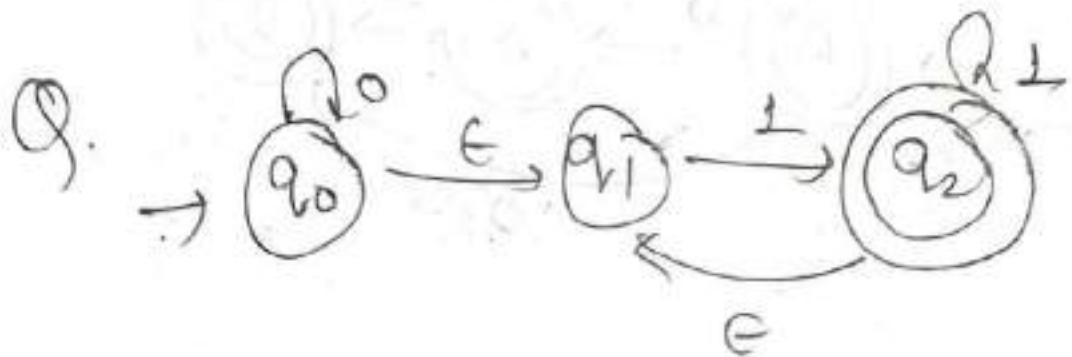
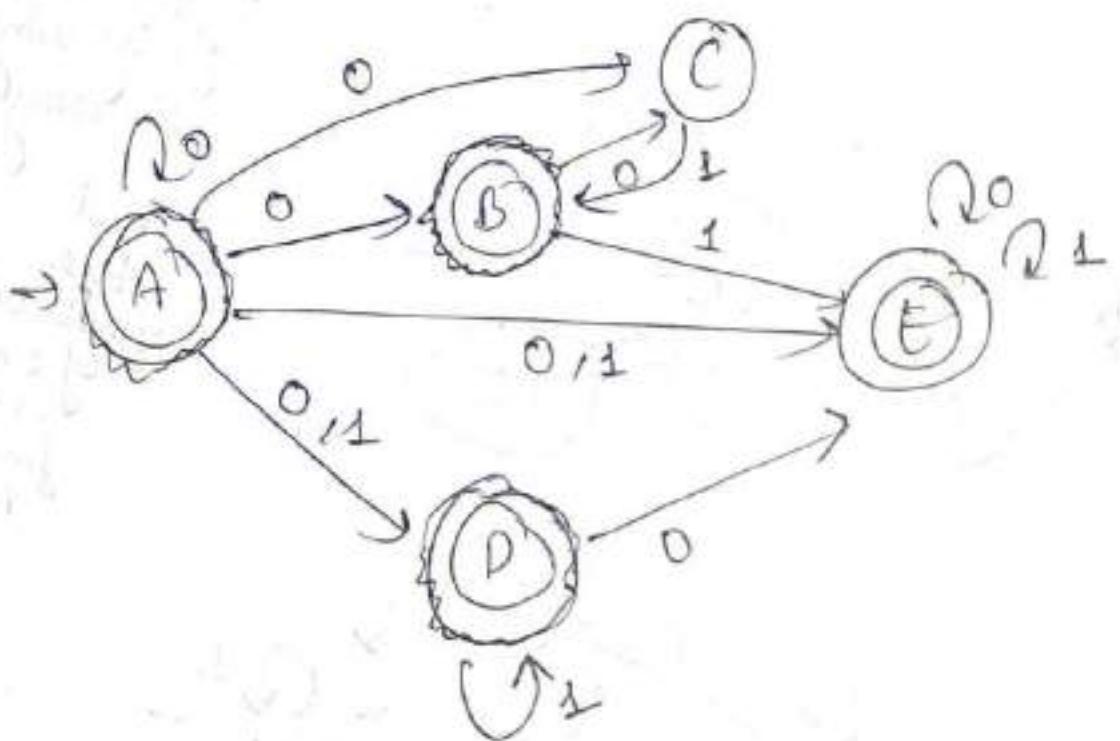
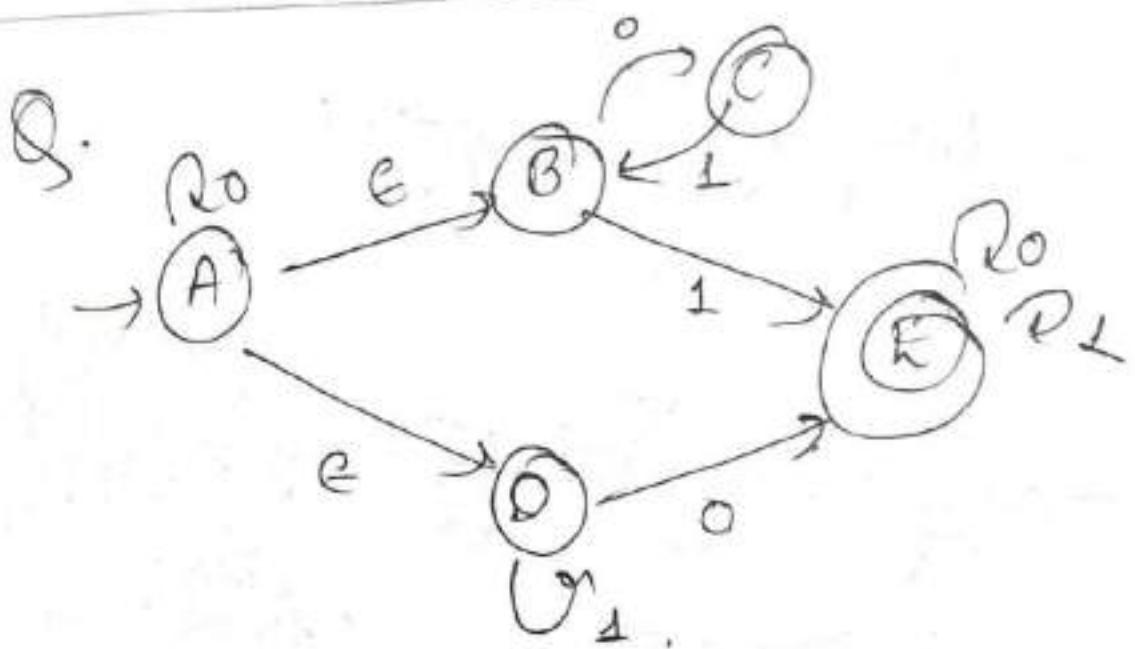
\downarrow

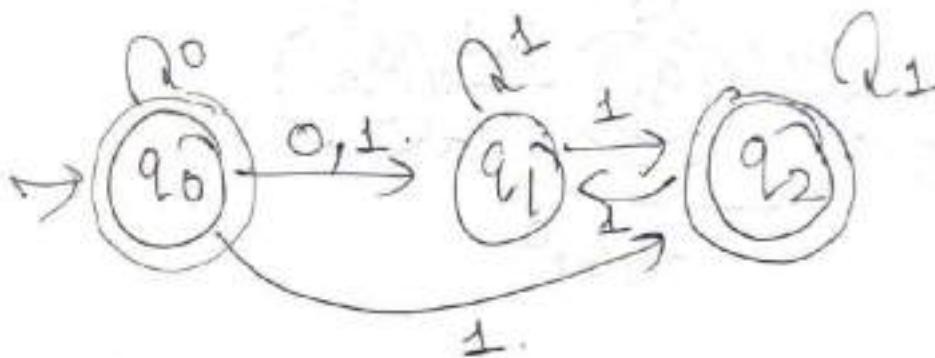
DFA.

NFA with ϵ \rightarrow without ϵ $\xrightarrow{\text{NFA}}$

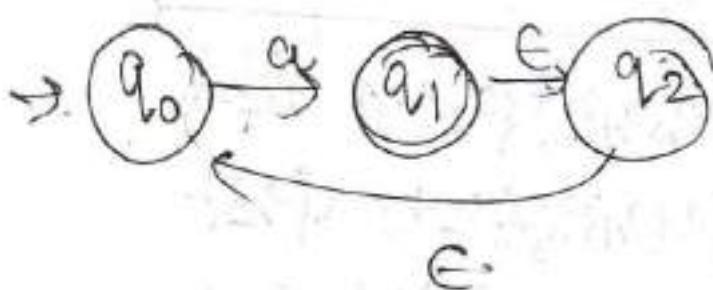
Q. Construct the NFA equivalent to
following ϵ -NFA



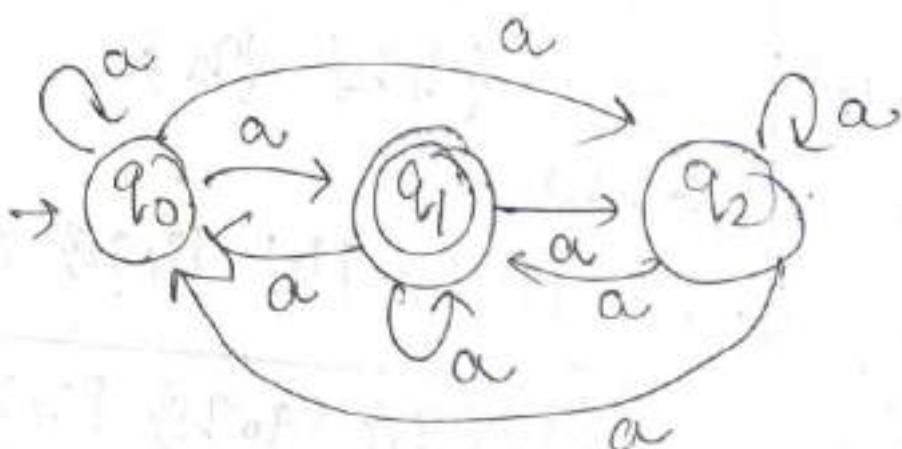




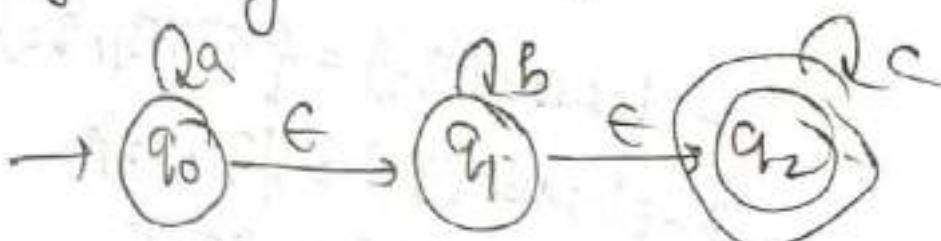
Q.

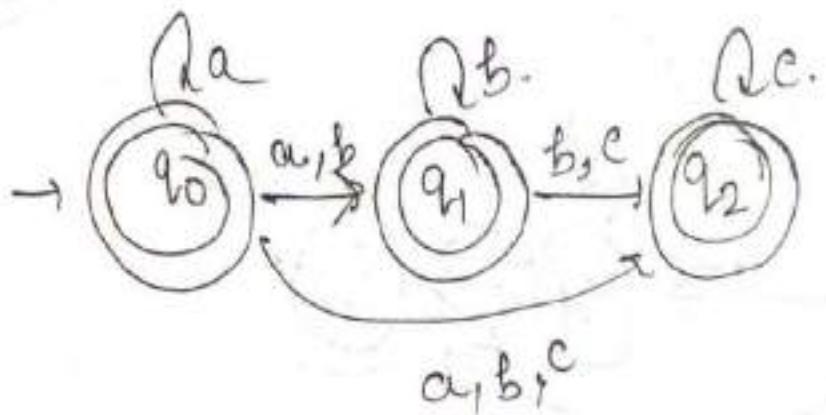


E.

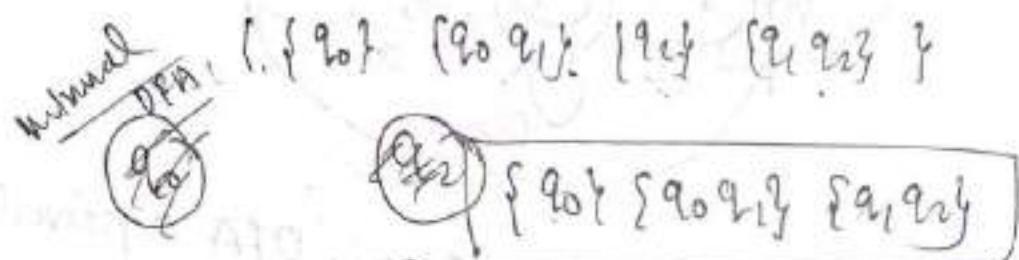


Q. Construct the minimal DFA equivalent
to following e-NFA,





	a	b	c
$\rightarrow q_0$	$\{q_0 q_1\}$, $\{q_2 q_1\}$	$\{q_2\}$	
$\{q_1\}$	$\{q_0 q_1\}$, $\{q_2 q_1\}$	$\{q_2\}$	
$\{q_2\}$		$\{q_1 q_2\}$, $\{q_2\}$	
$\{q_1, q_2\}$		$\{q_1 q_2\}$, $\{q_2\}$	
$\{q_0 q_1 q_2\}$	$\{q_0 q_1 q_2\}$		



$$\text{S}^*, \quad \epsilon\text{-closure}(q_0) = \{q_0, q_1, q_2\}.$$

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

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

ϵ -closure of a state is starting from the state by reading ϵ all possible reachable state.

The initial state of DFA is ϵ -closure of initial state of NFA.

NFA	a	b	c
q_0	$\{q_0 q_1 q_2\}$	$\{q_1 q_2\}$	q_2
q_1	-	$\{q_1 q_2\}$	q_2
q_2	-	-	q_2

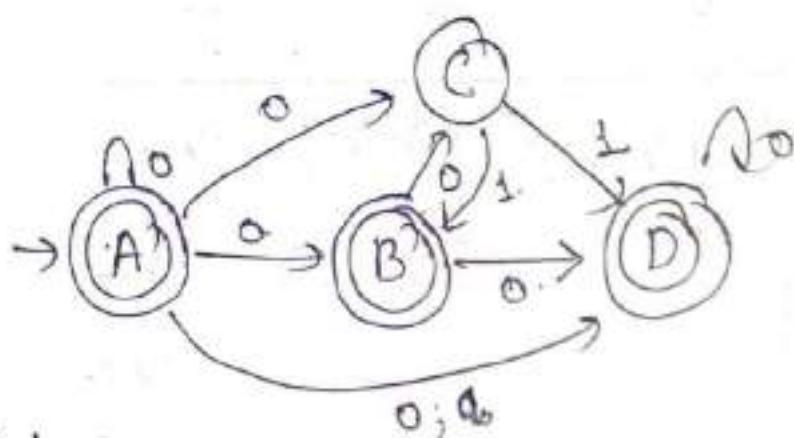
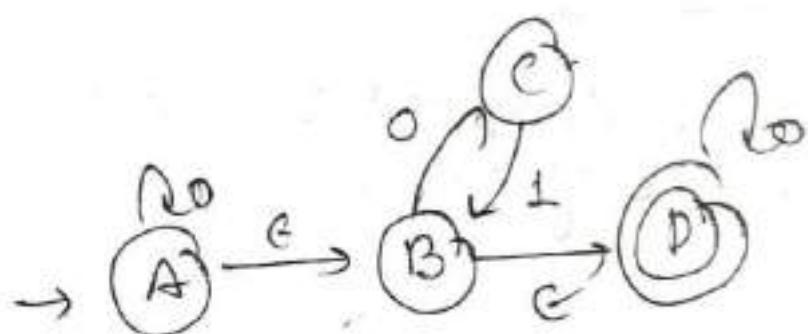
DFA	a	b	c
$[q_0 q_1 q_2]$	$[q_0 q_1 q_2]$	$[q_1 q_2]$	q_2
$[q_1 q_2]$	D.S	$[q_1 q_2]$	q_2
q_2	D.S	D.S	q_2
D.S	D.S	D.S	D.S

Perform minimization,

Initial states: $\{D.S\} \{[q_0 q_1 q_2], [q_1 q_2], [q_2]\}$

$\{q_1 q_2\} \{q_2\} \{q_0 q_1 q_2\} \{D.S\}$

Q. Construct the minimal DFA equivalent to following e-NFA



A	1	0	-
B	0	1	-
C	-	0	1
D	1	-	-

e-closure of (A) = A B ~~C D~~
e-closure of (B) = C D
e-closure of (D) = D.

DFA table:-

	0	1	0	1	
A	{ABCDEF}	{ABCDEF}	D.S.	ABD [ACDF]	9d.
{ABCDEF}	{ABCDEF}	{B,D}	FBCD	ABCDEF	D
{BD}	{CD}	D.S.	D	D	9f
{CD}	{D}	{B,D}			
{D}	{D}	D.S.			
D.S.	D.S.	D.S.			

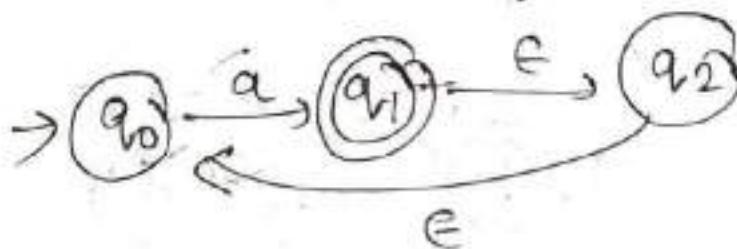
$\{A, ABCD, BD, CD, D\} \setminus \{B.S.Y\}$

$\{A, BD, D\} \setminus \{ABCD, CD\} \setminus \{B.S.Y\}$

$\{A, BD\} \setminus \{D\} \setminus \{ABCD, CD\} \setminus \{B.S.Y\}$

$\boxed{\{A, BD\} \setminus \{D\} \setminus \{ABCD, CD\} \setminus \{D\} \setminus \{B.S.Y\}}$

Q. Construct the following ϵ -NFA,



which
of the
two
folks is
complement of
lang accepted
by given
 ϵ -NFA

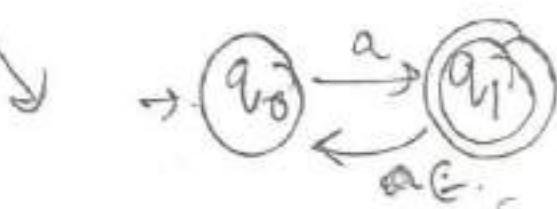
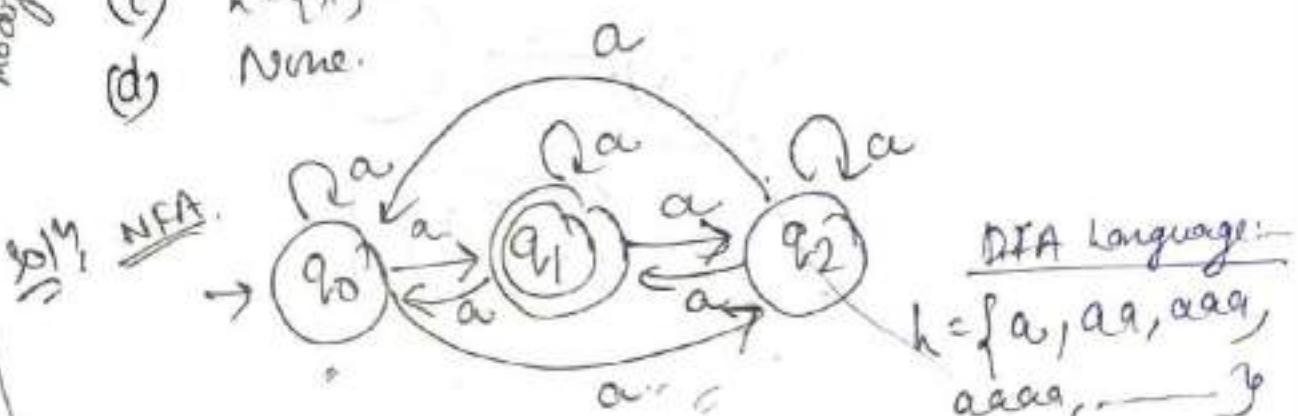
(a) $L = \{\}$

(b) $L = \{ε\}$

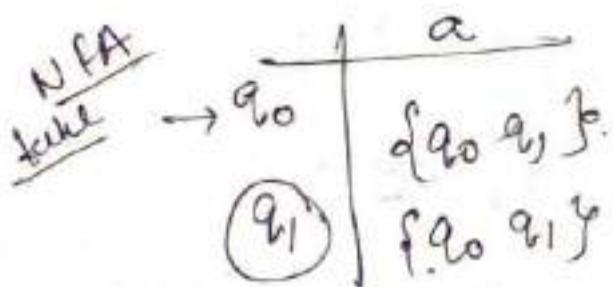
(c) $L = \{q_1\}$

(d) None.

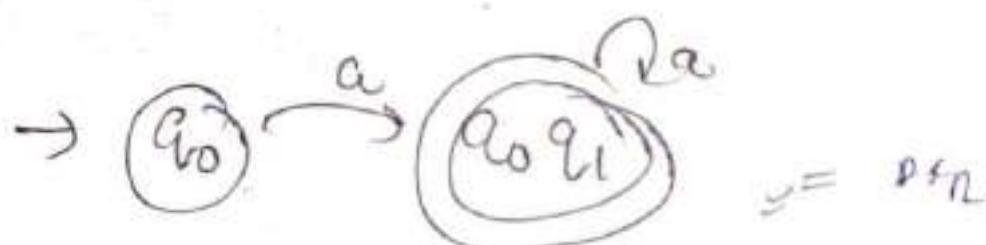
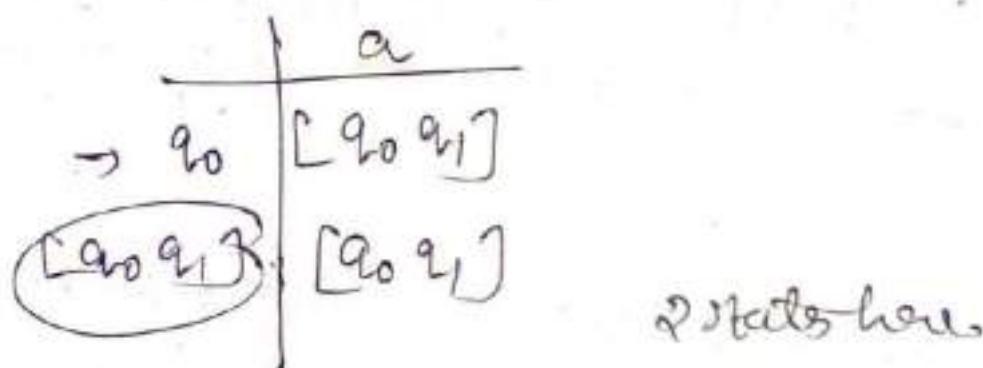
modified it as:



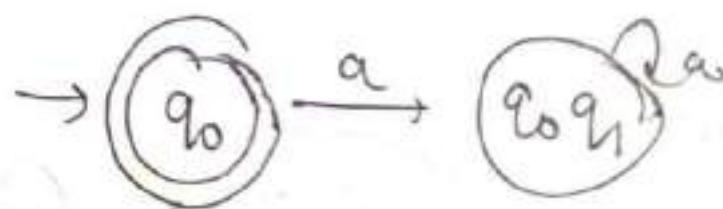
$\boxed{\Sigma^* - L = \{q_1, \epsilon\}}$



ϵ -closure of $q_0 = \{q_0\}$.



↓ Complement



$$L = \{\emptyset\}$$

Decision properties of F.A.

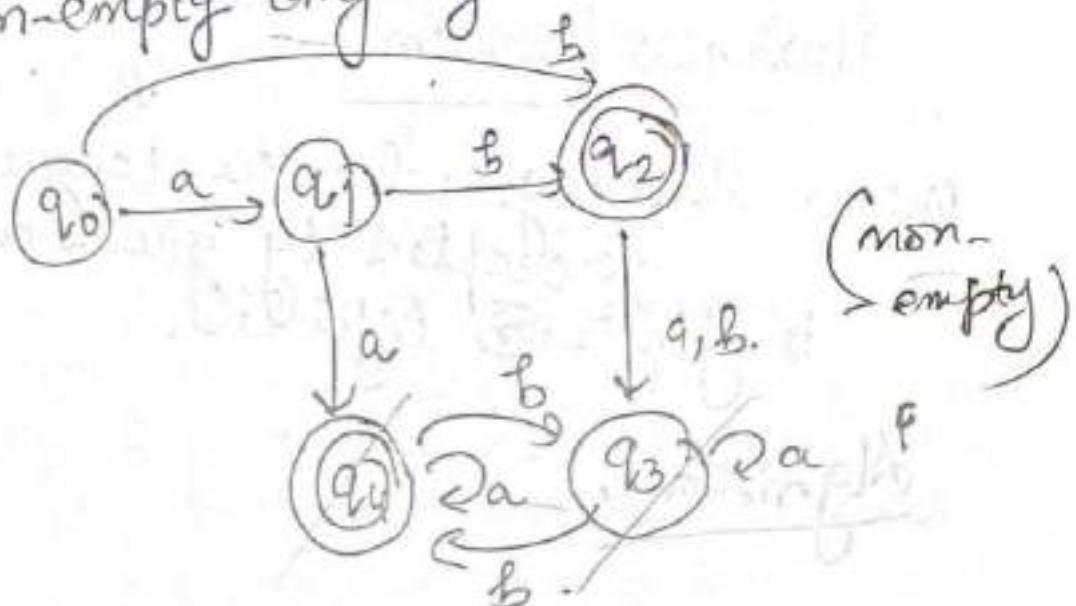
(If we able to say yes @ no → so decidable problem)

Decidable :-

A problem is said to be decidable there exist algorithm to solve that problem.
The following are some of the decidable problem of finite automata :-

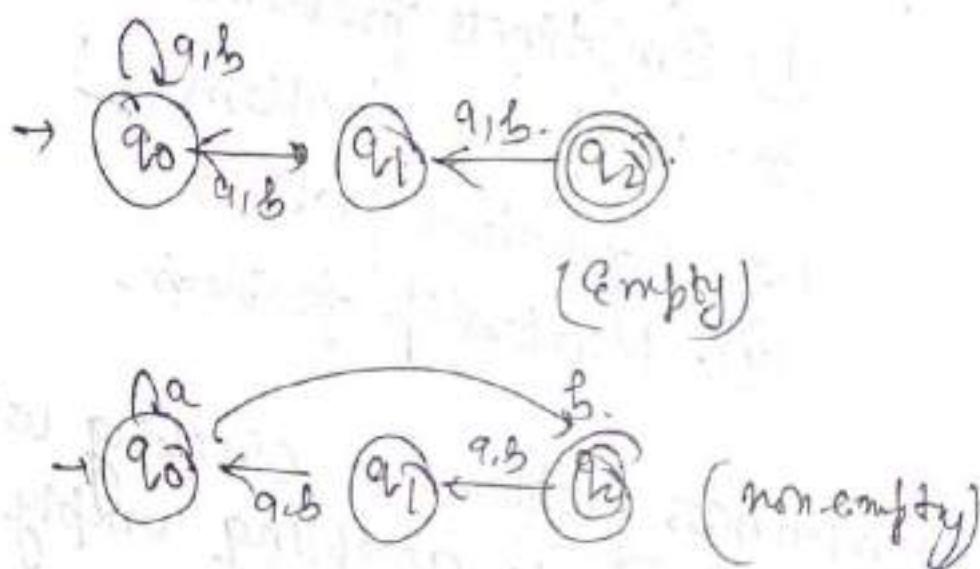
- ① Emptiness problem
- ② finiteness problem
- ③. Equivalence problem
- ④. Membership problem.

Emptiness problem :— Checking whether finite automata accepting empty language or non-empty language.



Algorithm :-

- ①. Remove all Inaccessible state of automata.
- ②. If the resulting automata if there exist atleast one final state then the automata accepts non-empty language otherwise empty.

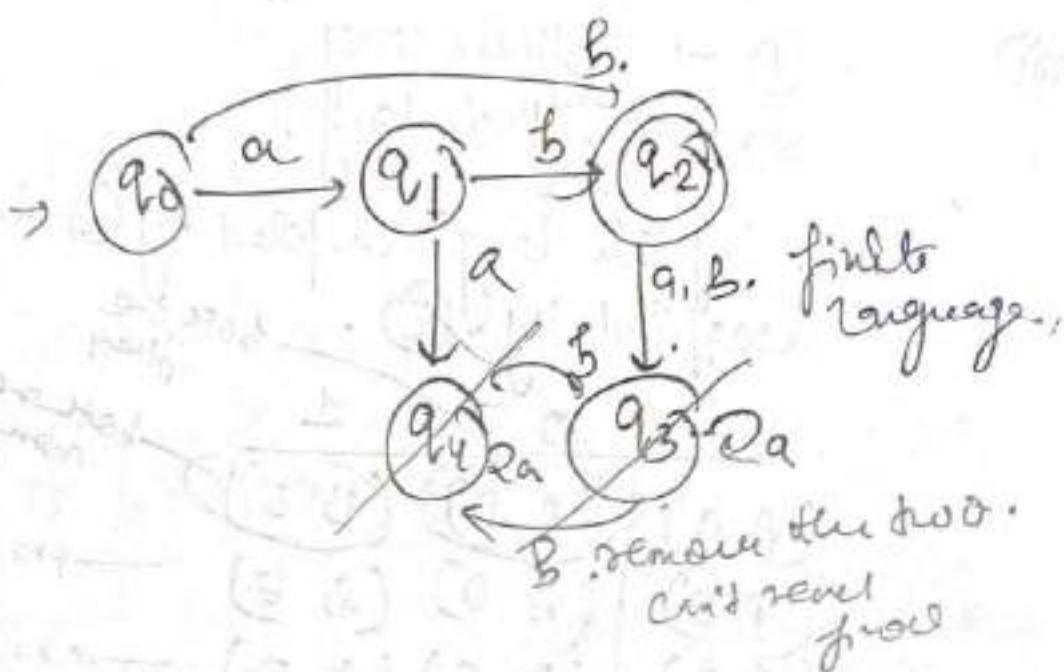
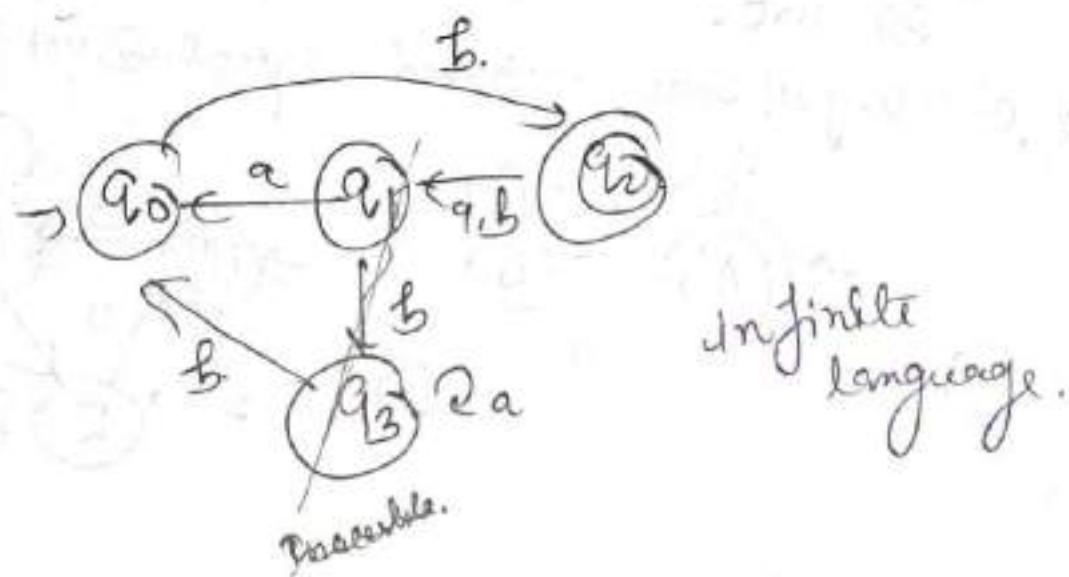


Finiteness problem:-

means checking whether language accepted by given automata is finite or infinite.

Algorithm :-

- ①. Remove all inaccessable states.
- ②. Remove the state from which we cannot reach final.
- ③. In the resultant automata if there exist atleast one loop or cycle that automata accepts infinite language otherwise finite language.



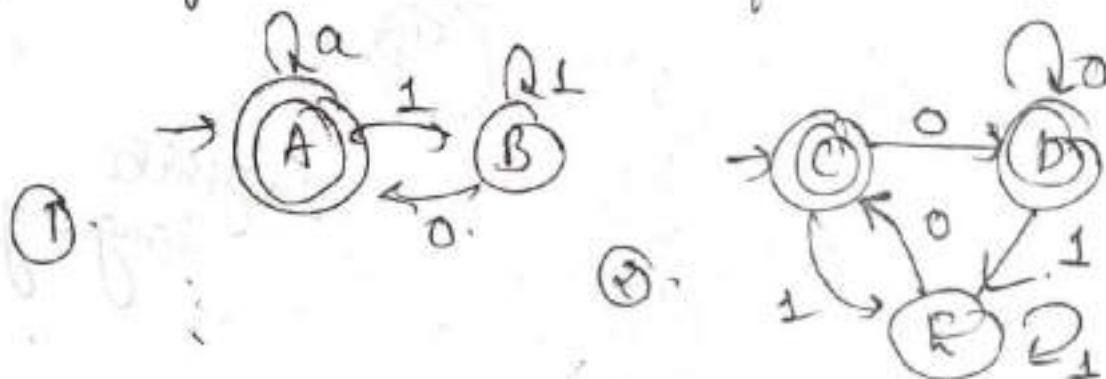
Equivalence problem:-

→ Checking whether lang. accepted by given two finite automata are equal or not.

Comparison table algorithm:-

exist to check given two automata equal or not.

Q. check foll two automata equal or not?



Sol:

① → Infinile lang.

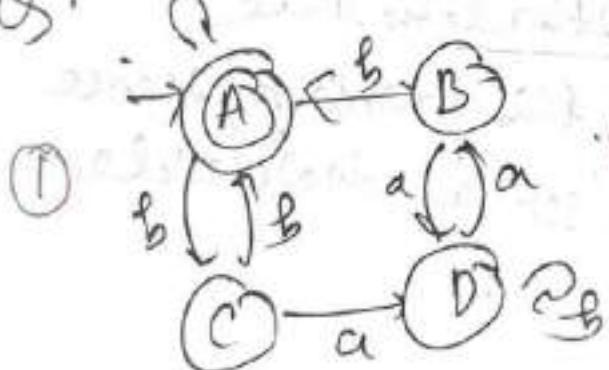
② → Infinile lang.

all infinite lang accepted by ① is also accepted by ②.

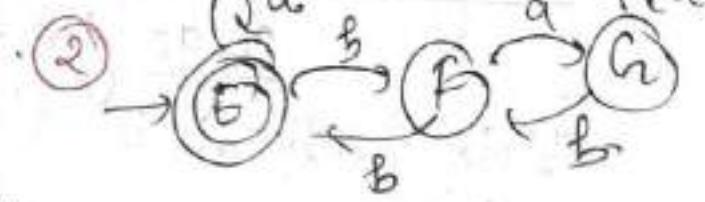
	0	1	
(A C)	(A D) (B E)		both are final so no problem
(A D)	(A D) (B E)		both are nonfinal so no problem
(B E)	(A C) (B E)		no problem

So, $\textcircled{1} \equiv \textcircled{2}$ if because there is no pair of final & nonfinal mean there is all equivalent state.

Q.



(I. nfinite)



(infinite)

2nd method

babb
string is
accepted
by $\textcircled{1}$ &
rejected
by $\textcircled{2}$.

	a	b	
(A E)	(A E)	(C F)	
(C F)	(D, G)	(A E)	
(D, G)	(B G)	(D, F)	
(A E) (B G)	(D G)	(A F)	→ problem.

not Equal ~~A~~

Membership problem:-

- means checking whether given string is member of given automata or not.
↓
{ means accepted @ not }

String acceptance method is the algo for this problem, hence this problem is also an decidable problem.

V.V.8m ✓ Regular Expressions : —

R.L \leftrightarrow R.E
DFA

- The simplest way of representing a regular language known as regular expression.
- For every regular language regular expression is possible & from

every regular expression a regular language⁽⁶⁴⁾
can be generated.

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

\downarrow

$(a+b)^*$ [Regular Lang.]

(if there is DFA possible for lang then we find
reg. expression.)

$$L = \{a^n b^n \mid n \geq 1\} \Rightarrow X \text{ not possible.}$$

$C = a + b.$

operands.

operator

\Rightarrow Union

\Rightarrow Concatenation

$\ast \Rightarrow$ Kleene closure

- For every regular language we can construct one equivalent regular expression by taking symbols of input alphabet as operands. and by using following three operators \rightarrow .

$\vdash \Rightarrow$ Union of
 $\cdot \Rightarrow$ Concatenation of
 $* \Rightarrow$ Kleene closure

- Regular expression is possible only for the language for which DFA possible.
(for regular language only).

Regular mean DFA possible.

language Regular

$$\left. \begin{array}{l}
 \textcircled{1} \quad L = \{\emptyset\} \rightarrow \emptyset \\
 \textcircled{2} \quad L = \{\epsilon\} \rightarrow \epsilon \\
 \textcircled{3} \quad L = \{a\} \rightarrow a
 \end{array} \right\} \begin{array}{l}
 \text{Basic R.E} \\
 \text{because we can't} \\
 \text{use any operator}
 \end{array}$$

$$\textcircled{4} \quad L = \{a, b\}^* = a + b \Rightarrow \text{either } a \text{ or } b \\
 \text{but not both}$$

$0 \xrightarrow{*} 0 - 0$

$$\textcircled{5} \quad L = \{ab\}^* = a \cdot b \Rightarrow \text{both } a \text{ & } b \text{ not leave any one of them.}$$

$- 0 \xrightarrow{a} 0 \xrightarrow{b} 0$

⑥ $L = \{ \epsilon, a, aa, aaa, \dots \}$

(65)

$$= \boxed{a^*} = (\epsilon + a + aa + aaa + \dots)$$

⑦ $L = \{ a, aa, aaa, \dots \}$

(95)

$$= \boxed{a^+} \text{ (positive closure)}$$

\downarrow
(a^* without ϵ)

⑧ $L = \{ a, b, aa, ab, ba, bb, \dots \}$

$$\textcircled{O} \xrightarrow{a,b} \textcircled{Q} \xrightarrow{a,b} \boxed{(a+b)^+}$$

Identities of R.E.: \rightarrow

①. $R + \phi = R = \phi + R.$ (ϕ acts as zero)

②. $R \cdot \phi = \phi \cdot R = \phi.$

(ϵ acts as 1)

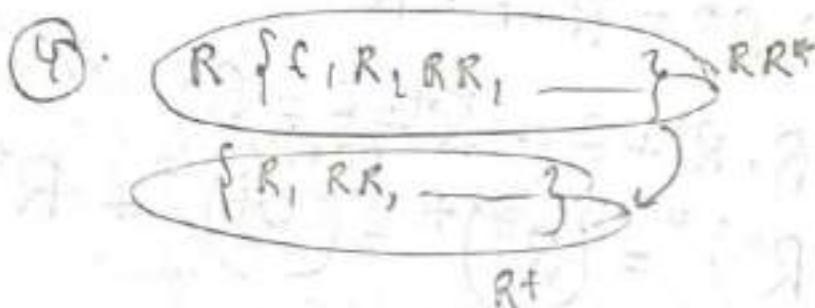
③. $R \cdot \epsilon = \epsilon \cdot R = R.$

④. $R \cdot R^* = R^* \cdot R = R^+$

⑤. $R \cdot R^* + \epsilon = R^* + \epsilon = R^* \Rightarrow (R^* + \epsilon = R^*)$

⑥. $(R^*)^* = (R^*)^+ = (R^+)^* = R^{**} \in \{\epsilon, R, R^*, R^+\}$

- 2/16/2021
7. $E^* = E$
8. $E^+ = E$
9. $\phi^* = E$
10. $\phi^+ = \phi$
11. $(a+b)^* \neq (ab)^*$ it does not contain ba .
12. $(a+b)^* \neq a^* b^*$ it contains ba .
13. $(a+b)^* \neq a^* + b^*$ not contain ab .
14. $ab \neq ba$
15. $(a+b) = (b+a)$
16. $a(ba)^* = (ab)^* a$
17. $(a+b)^* = (a^* + b^*)^*$
 $= (a^* + b)^*$
 $= (a + b^*)^*$
 $\vdash (a^* b^*)^* \rightarrow \cancel{a^* b^*} \quad \begin{array}{l} a^* b^* \\ \cancel{a^* b^*} \\ b^* a^* \end{array}$
 $\vdash a^* (b^* a^*)^*$



- for one regular language many Regular Expressions possible. (66)

→ Priority is given to the simpler

is one that have less no. of operators.

So, $(a+b)^*$ have highest priority because of Kleene closure

Q. construct the two foll. regular expression
P & Q:-

$$P \& Q = [(01)^* + 1]^*$$

$$Q = (01)^* (1(01)^*)^*$$

which of the following is true \Rightarrow

Soln: (a) ~~(01)*~~ P = Q

(b) P ⊂ Q

(c) P > Q.

(d) none

Soln: P = [(01) + 1]^*

Q = [01 + 1]^*

P as $(a^* + b)^*$
Q as $a^* (b a^*)^*$

Q. Consider the following 4 regular expressions:-

(I). $(0+\epsilon)(00)^*$

(a) (I) & (II)

(II). $0(00)^*$

(b) (II) & (IV)

(III). 0^*

(c) (I) & (III)

(IV). $(00)^*$

(d) (I) & (IV).

& which of these are equal.

Soln: (I). $0(00)^* + (0+\epsilon)(00)^* = \begin{cases} \text{Even} \\ \text{odd} \end{cases}$
 \Rightarrow even \oplus even = all even.

(II). $0 \boxed{(00)^*} \rightarrow \text{odd zero}$

(III). $0^* \rightarrow \text{all zeros}$

(IV). $(00)^* \rightarrow \text{even zeros}$

Q. Which of the following regular expression generates set of all strings of 0's & 1's not containing 100 as substring.

X (I). $0^* \boxed{(1+0)^*}$ work it out
not 100

X (II). $0^* 1010^* \rightarrow 100$

X (III). $0^* 1^* 0^* \rightarrow \epsilon$ not possible it does not contain 0100 but not all string of 0's

✓ (IV). $0^* (10+1)^*$

Q. Which of the following regular expressions are equal:—
(67)
 not.

- (a) $1(01)^*$ and $(10)^* 1$
- (b) $(XX)^* X$ and $X (XX)^*$
- (c) $(ab)^*$ and $a^* b^*$ but only a & only b possible
- (d) X^+ and XX^* but not only a & only b possible

Q. Consider the following 3 regular expressions:—

$$R_1 = 11 (0+1)^*$$

$$R_2 = (0+1)^* 11$$

$$R_3 = 11 (0+1)^* 11 + 111 + 11.$$

which of the foll is true:—

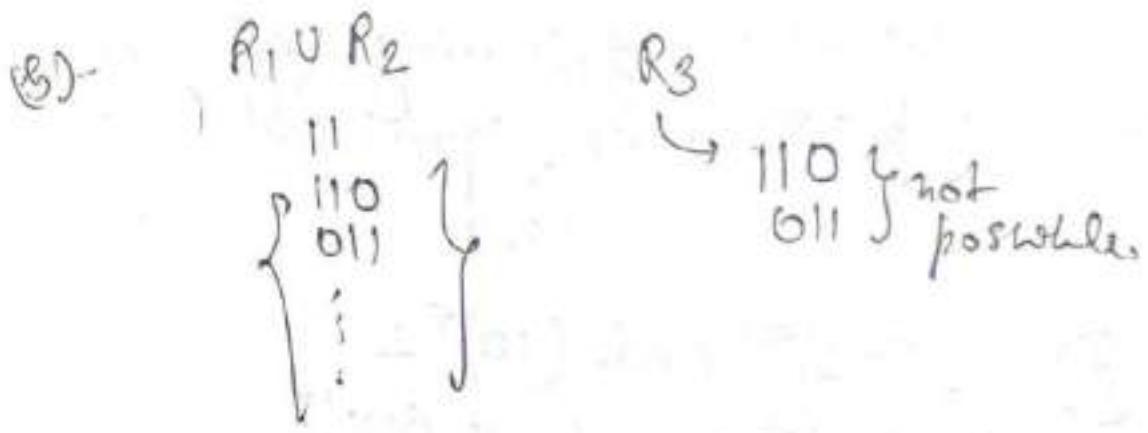
(a). $R_1 = R_2 = R_3$

(b). $R_1 \cup R_2 = R_3$

(c). $R_1 \subseteq R_3$

(d). $R_1 \cap R_2 = R_3$

$$\begin{array}{c} R_1 \neq R_3 \\ \downarrow \quad \downarrow \\ 110 \neq 110 \end{array}$$



(C) R_1 is bigger set than R_3 .

$$R_3 \subseteq R_1$$

Q. Consider the following 3 regular expressions r, s, t. Which of the following is true.

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

$$s = 11^*0 : \{10, 110, \dots\}$$

$$t = 1^*0 : \{0, 10, 110, \dots\}$$

(A) $r = s = t$

(B) $r \subset s$ and $s \subset t$

(C) $s \subset r$ and $s \subset t$.

(D) none.

Soln:

$$\underline{s \subset r} \quad s \subset t$$

(68)

Q. $L = b^* (a \cdot \phi \cdot b + ab + a \cdot \phi \cdot b^*) (b + \phi)^*$

- (a) exactly one a
 (b) atleast one a
 (c) atmost one a
 (d) None

~~b*~~ $b^* (\phi + ab + \frac{\phi}{ab}) (b)^*$

$$\begin{aligned} & b^* (ab + ab^*) b^* \\ & b^* (a(b + b^*)) b^* \\ & \downarrow \\ & b^* \end{aligned}$$

$$b^* (a b^*) b^*$$

$$\boxed{bab^*}$$

Q. Consider the following three regular expressions:-

①. $(a + ba)^* (b + \epsilon)$ subset of $(a+ba)^*(b+\epsilon)$ zero or it

②. $(a^* (ba)^\epsilon)^* (b + \epsilon) + a^* (b + \epsilon)$ $\frac{a^* (ba)^\epsilon}{(a+ba)^* (b+\epsilon)}$ + $\underbrace{(ba)^\epsilon}_{(b+ba)^*} (b + \epsilon)$

③. $(a + ba)(a + ba)^* (b + \epsilon)$ subset of $(a+ba)^*$ zero or it

which of the following are equal. —

- (a) ① & ② (b) ① & ③ (c) ② & ③ (d) None

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

$$\textcircled{1} \rightarrow \cancel{(a+ba)^*} (b+\epsilon) + a^*(b+\epsilon) + (ba)^* \cancel{(b+\epsilon)}$$

$$(b+\epsilon) \left\{ (a^*(ba)^*)^* + a^*(b+\epsilon) + (ba)^* \right\},$$

1st method

\textcircled{1} $\rightarrow \epsilon$ possible

\textcircled{2} $\rightarrow \epsilon$ possible

\textcircled{3} $\rightarrow \epsilon$ not possible

Q. Consider the following regular expression

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

The regular expr. equivalent to

\textcircled{1}. $01^*(0+1)^*$

\textcircled{2}. 0^*1^*

\textcircled{3}. $(0+1)^*$

\textcircled{4}. None

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

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

remove ϵ part $(0+1)^* = \{ \epsilon, 0, 1, 00, 01, 10, 11 \}$

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

Q. Which of the foll. sets are equal: -

(69)

①. $0^*(0+1)$

②. $\{0, 00, 000, \dots\} \cup \{1, 01, 001, 0001, \dots\}$

③. $\{0^n \mid n \geq 1\} \cup \{0^{n+1} \mid n \geq 0\}$.

④. $(0+1)0^*$.

1, 2, 3

Ans:

Construction of Regular expressions:-

Q. Construct the R.E. for

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

Soln. Not possible

$$\textcircled{2}. \quad L = \{a^n b^m \mid n, m \geq 0\}.$$

Any no. of a's & any no. of b's including 0

$$a^* b^*$$

$$\textcircled{3}. \quad L = \{(a^n)(b^m) \mid n, m \geq 1\}$$

$$a^+ b^+$$

$$\textcircled{4}. \quad L = \{a^n b^m \mid \begin{matrix} n \geq 0 \\ m \geq 1 \end{matrix}\}$$

$$a^+ b^+$$

$$\textcircled{5}. \quad L = \{a^n b^m \mid \begin{matrix} n \geq 2 \\ m \geq 3 \end{matrix}\}$$

$$aa^+ bbb^+$$

$$\textcircled{6}. \quad L = \{a^n b^m \mid n \neq m\}$$

Not possible.

$$\textcircled{7}. \quad L = \{a^n b^m \mid (n+m) \text{ is odd}\}$$

~~Not possible~~. Regular expression
possible.

even odd

odd even

$$\boxed{(aa)^* b (bb)^* + a (aa)^* (bb)^*}$$

⑧. $L = \{a^n b^m \mid (n+m) \text{ is even}\}$

even even
odd odd.

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

~~8m~~ ⑨. $L = \{a^n b^m \mid n \geq m \text{ and } n \leq m\}$

$L = \{\} \rightarrow \emptyset$ Empty language because
there no string possible like that

~~2011-12~~ $L \neq \emptyset$ because here
not say $n = m$
because here
we have 0 no. of ab &
0 no. of bb.

⑩. $L = \{a^n b^m \mid n \geq m \text{ and } n \leq m\}$

$L = \{ \epsilon, ab, aabb, aaabb, \dots \}$

$= \{a^n b^n \mid n \geq 0\} \times$ not possible

⑪. $L = \{a^n b^m \mid n \geq m \text{ and } n \leq m\}$

\Rightarrow Not possible comparison

~~Q12~~ find the R.E that generate all the substrings of the string TOC.

Substring:-

ϵ	$\frac{n(n+1)}{2} + 1$
T, O, C	$\frac{n(n+1)}{2}$
TO, OC	$\Rightarrow \frac{3 \times 4}{2} + 1$
TOC	$\Rightarrow 7$

~~Q13~~ [finite language] so, its regular express is \rightarrow

$$\boxed{\{ \epsilon + T + O + C + TO + OC + TOC \}},$$

~~Q13~~. Find the R.E that generates all odd length palindromes strings over the input alphabet $S = \{a, g\}$

~~Ans:~~ $\{ a, aaa, aaaaa, \dots \}$

~~a(ba)~~ $\boxed{a(a^2)^k}$

~~Q14~~ Find the R.E that generates set of all even length palindromes strings of English language

Ex: Noon

English alphabet (a, b, \dots, z)
 Input alphabet (a, b)

Not possible

- Q8. Find the regular expression that generates even length palindromes over the typed alphabet $S = \{a, b\}$.

Sol. $L = \{aa, bb, aaaa, abba, baab, bbbb, \dots\}$

Not possible

Palindrome languages over one symbol are regular more than one symbol palindrome languages are non-regular. Hence, regular expression not possible.

- Q9. Find the R.E for odd length palindrome strings of Hindi language.

wuw

abba

Not possible.

madam

comparison needs y also
not possible.

$$L = \{ \underline{w} \underline{w^R} \mid w \in \{a, b\}^* \}$$

(even length palindromes)

a b b b a

$$L = \{ \underline{w} \underline{c} \underline{w^R} \mid w \in \{a, b\}^* \}$$

a b a c a b a

(odd length palindromes)

for both R.E not possible due to comparison.

Q12 Find the R.E. for the language

$$L = \{ \underline{w} \underline{x} \underline{w^R} \mid w \in \{a, b\}^* \}$$

x $\in \{a, b\}^*$

Sol

(ab)abba, abbabba, —

②

when

$$\omega = \epsilon,$$

$$L = \{aa, ab, ba, bb, \dots\}$$

$ab \times ba \rightarrow$ belong to the
 $ba \times ab \rightarrow$ complete language

therefore every answer is complete lang.

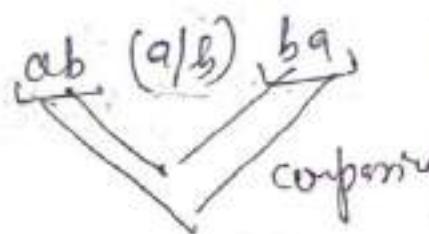
~~Ans~~

$R.E. : - (a+b)^*$

~~Ans~~

(18) $L = \{ \omega \times \omega^R \mid \begin{array}{l} \omega \in \{a, b\}^* \\ X \in \{a, b\}^* \end{array} \}$

$$\omega = \epsilon, \quad \underline{a} \oplus \underline{b}.$$



Not possible

(19) $L = \{ \omega \times \omega^R \mid \begin{array}{l} \omega \in \{a, b\}^* \\ X \in \{a, b\}^{*k} \end{array} \}$

$$\underbrace{a(a+b)^k a}_a + \underbrace{b(a+b)^k b}_b$$

$a(a+b)^k a + b(a+b)^k b$

⑩. $L = \{ \omega x \omega^R \mid \omega, x \in \{a, b\}^* \}$

~~abba~~

a a

[abca]aba[aaa]

not palindrome.

[ba agababb'ab]

reg. expression —

starting with a &

ending with a

(c)

starting with b & ending with b

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

Q. Find the R.E

$$L = \{ \underbrace{\omega \omega}_{\text{copied}} \mid \omega \in \{a, b\}^* \}$$

sol— $L = \{ \epsilon, aa, bb, abab, abbabb, \dots \}$

Not possible due to comparison.

Q. Write R.E for this,

$$L = \{wxw \mid w \in \{a, b\}^*\}$$

Soln: $L = \{abxab\}$
Not possible

Q. $L = \{wxw \mid w \in \{a, b\}^*, x \in \{a, b\}^*\}$.

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

Q. $L = \{w(xw) \mid w, x \in \{a, b\}^*\}$
 longer so R.E. possible

w x w

$$\cancel{ab} / \cancel{ab} ab \quad \boxed{(a+b)^*}$$

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

Q. $h = \{1, 2, 4, 8, \dots, 2^n, \dots\}$
 all these numbers in binary
 are

103

(DPRCCEG)

01, 10, 100, 1000, —————
0 0 0 0 0 1 0 —————

0° ± 0°

145

$$8. h = \{1, 2, 4, 8, \dots, 2^n, \dots\}$$

all these no's are in unary

105

1
1 1
1 1 1 1

3

Not possible

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

not in
A-B
DFA not possible
R-E not possible

74

Q. Find the R.E. that generates all strings of 0's & 1's where each string starting & ending with diff. symbol.

| sol:

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

Q. Starting & ending with same symbol.

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

Q. 3rd input symbol is 1 while reading the string from L.H.S.

10m Q. (0+1) : $(0+1)(0+1) 1 (0+1)^*$

Q. Length of the string exactly 4.

$$(0+1)(0+1)(0+1)(0+1)$$

Q. How many states are there in minimal DFA that accept the regular expression \rightarrow

$(a+b+c)(a+b+c) \dots$ $(n-2)$ times.

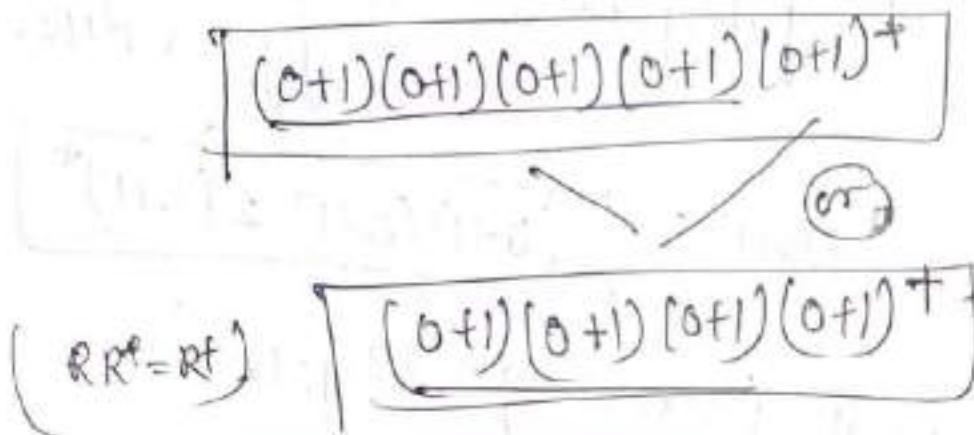
- (A) n
(B) n^2
(C) $n+2$
(D) none.

Sol^m

for exactly $n-1$ states $\rightarrow (n+2)$ states

$(n-2) \rightarrow n$ states

Q. length of the string atleast 4.



* length of string atleast n ,

$(0+1)(0+1)(0+1) \dots$ n times $(0+1)^*$.

or

$(0+1)(0+1)(0+1) \dots$ $(n-1)$ times $(0+1)^+$.

Q. How many states are there in minimal DFA
that accept the foll.
reg. expression,

(75)

n^{th}
 n^{th}

$$(a+b+c)(a+b+c) \dots (n-2) \text{ times } (a+b+c)^+$$

(a) n

(b) $n+1$

(c) $n-1$

(d) None.

Sol:

(a) at least $(n-1)$ length $\rightarrow n+1$
 $\rightarrow [n \text{ states}]$

Q. Length of the string atmost S .

$$(0+1+\epsilon) \cdot (0+1+\epsilon) \cdot (0+1+\epsilon) \cdot (0+1+\epsilon) \cdot (0+1+\epsilon)$$

$(0+1+\epsilon)^n$ times \rightarrow atmost n length string.

DFA: - $(n+2)$ states
 NFA: - $(n+1)$ states

Q. Each string contains exactly 4 0s
 $\Sigma = \{0, 1\}$,

$$1^* 0^* 1^* 0^* 1^* 0^* 1^*$$

Q. atmost 3 0s.

~~1* 0* 1* 0* 1*~~

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

gate 2003

each string contains atleast 2 0s

$$1^* 0^+ 1^* 0^+ 1^+$$

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

Q. length of the string divisible by 4,

6, 4, 8, 12, ...

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

Q. length of the string even.

$\frac{1}{2}$
divisible by 2.

$$[(0+1)^2]^*$$

Q. length of the string odd

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

$$\left\{ [(0+1)^n]^* \rightarrow \text{divisible by } n \right. \\ \left. \begin{array}{l} \text{DFA} \rightarrow n \text{ states} \\ \text{NFA} \rightarrow n \text{ states} \end{array} \right\}$$

Q. the string is starting with zero total
length is odd \cap string starting with
total length is even

$$0[(0+1)^2]^*$$

$$+ \frac{1}{2} [(0+1)^2]^*(0+1)$$

Q. odd zeros followed by even 1's

$$[00]^* 0 [11]^*$$

Q. How many no. of states are there for minimal DFA that accept,

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

卷之三

$$\frac{20}{2} = 10 \text{ sets}$$

Q. No. of zeros divisible by 3. $\Sigma (0,1)$

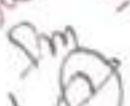
0, 000, 6 —

$$1^{\alpha} (001)^{\beta} \underline{0(00)}^{\gamma} (011)^{\delta} 1^{\epsilon}$$

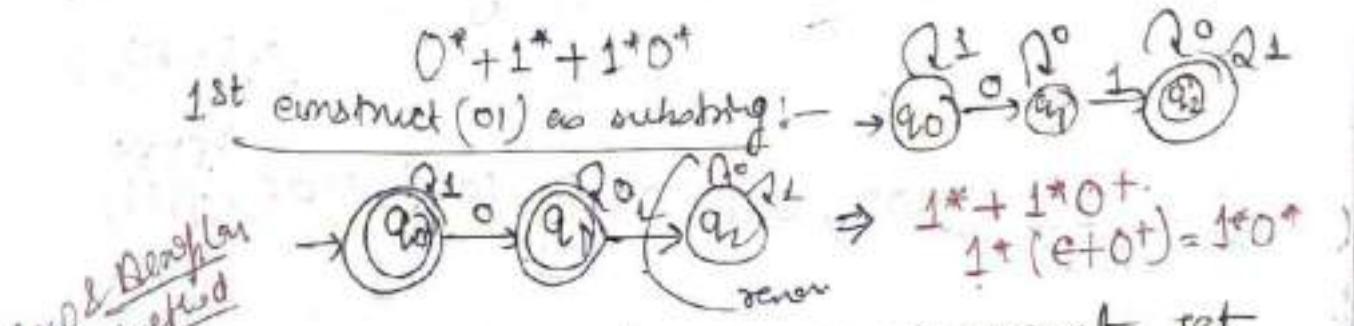
$$(101^k 01^k 01^k)^q + 1$$

↓ single I
not come so
add it

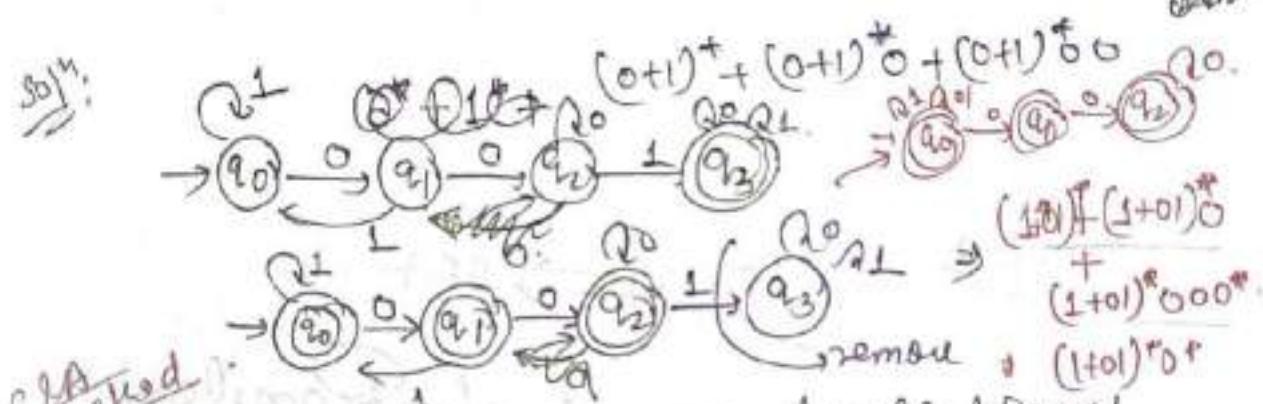
No. 8 Aeroplane
method

 Find the R.E. that generates set of all strings of 0's & 1's not containing 01 as substring.

$$\text{Soln: } (0+1)^* - \frac{01}{(0+1)^* 0 1 (0+1)^*} = (0+1)^* 0 + (0+1)^*$$



Q. Find the regular expression that generates set of all strings of 0s & 1s that containing 001 as substring.



Q. Find the R.E. that generates all binary no.s. not divisible by 3.

Soln:

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

1

Q.A
solved

Q. Find the R.E that generates all binary
nos divisible by 3.

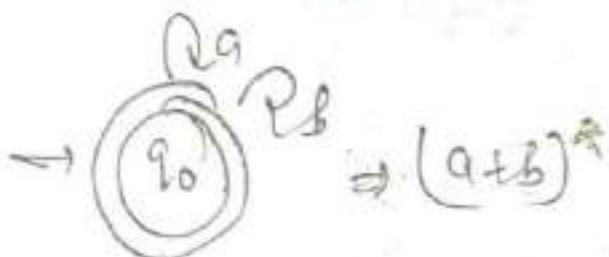
Solⁿ.

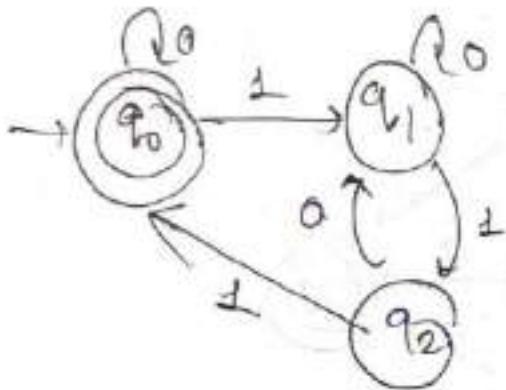
000, 011, 0110, 1001, 1100, 1111,
1

No. 0

+ R.E \leftrightarrow DFA

Finite Automata to Regular
Expression Construction :

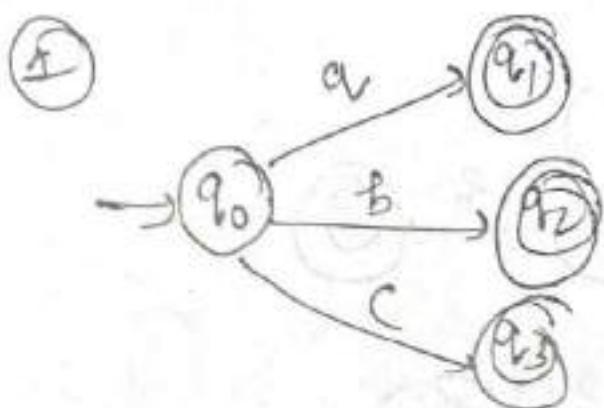
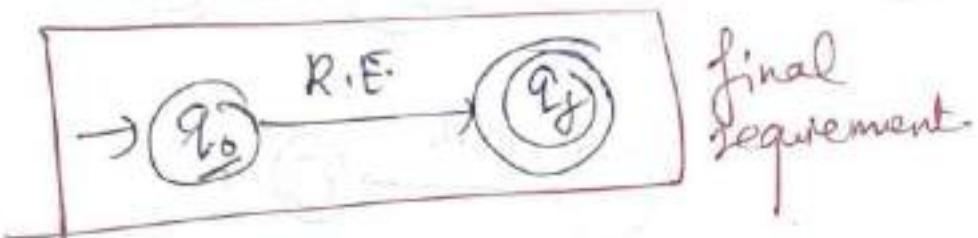


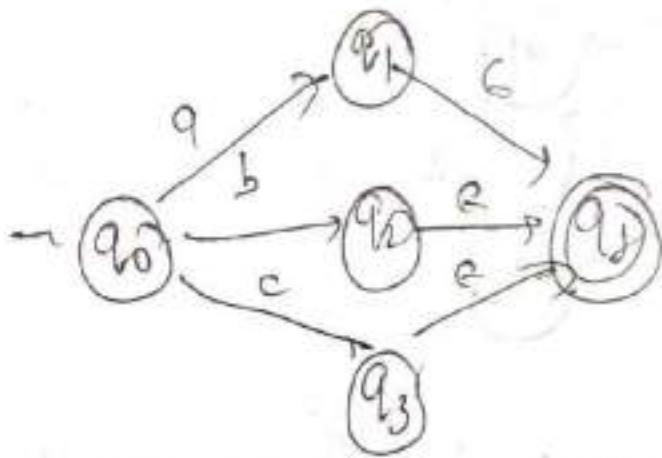


$\textcircled{1}$ Arden's method } 2 method to
 $\textcircled{2}$ R_{ij}^F method. } find R.E.
 If e-NFA given we can't construct the R.E.
 so, we go for state elimination method.

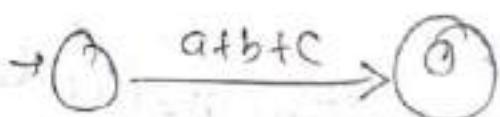
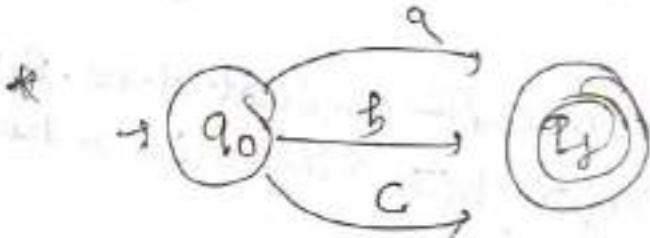
State Elimination method \Rightarrow

F.A \Rightarrow R.E.

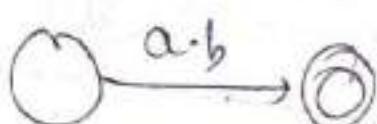




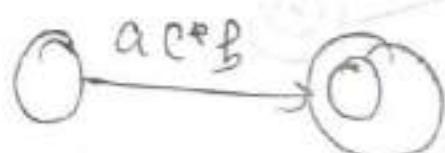
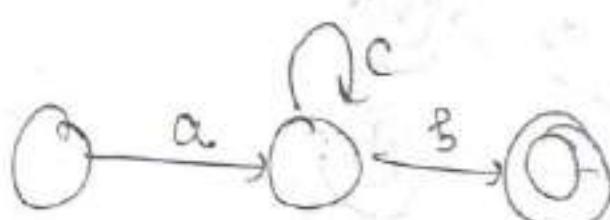
②



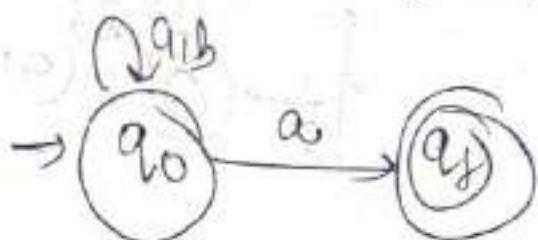
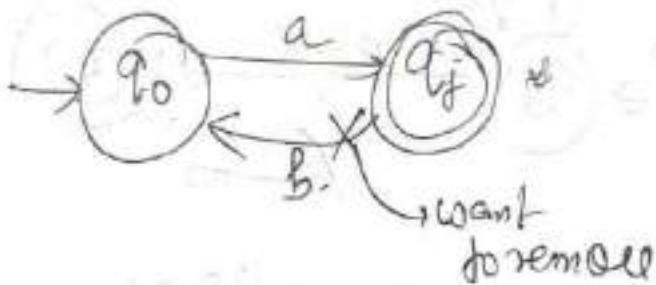
③



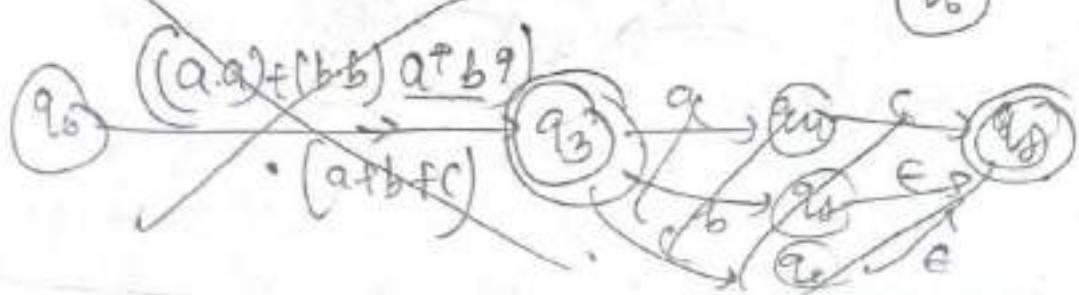
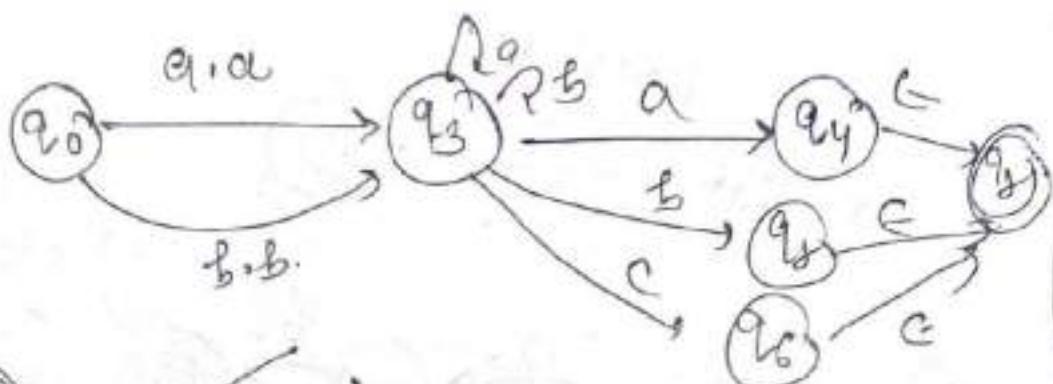
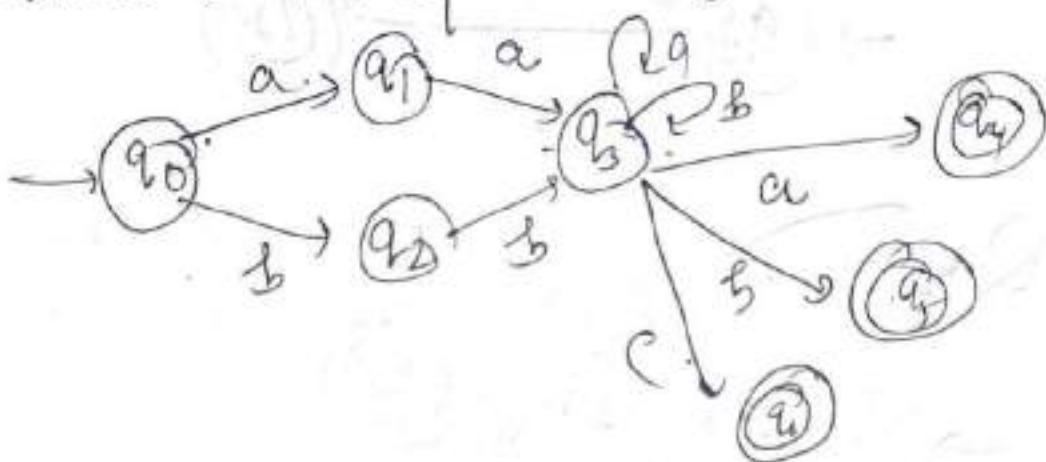
④

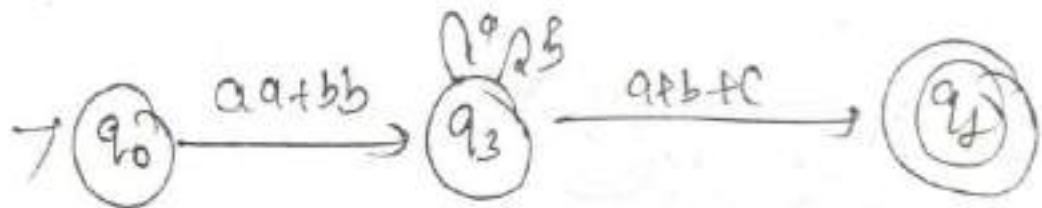


(5)

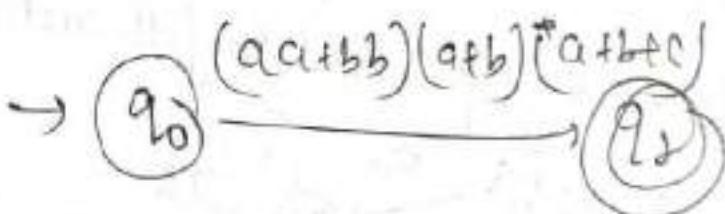


Q. Construct the R.E. equivalent to foll automata,

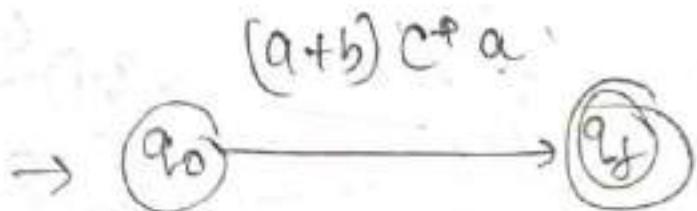
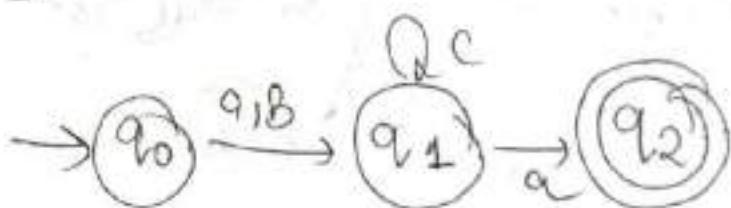




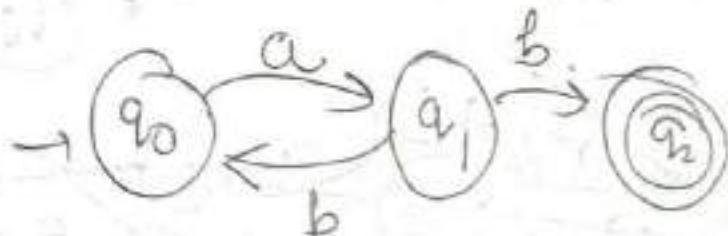
$$\{ - \overset{ab}{\circ} = (a+b)^*$$



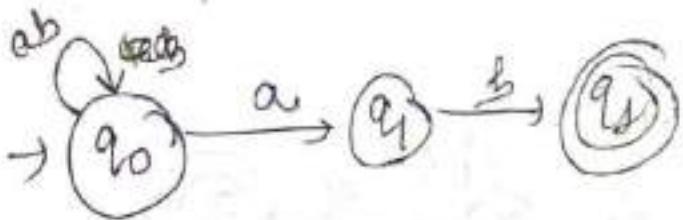
Q.



Q.

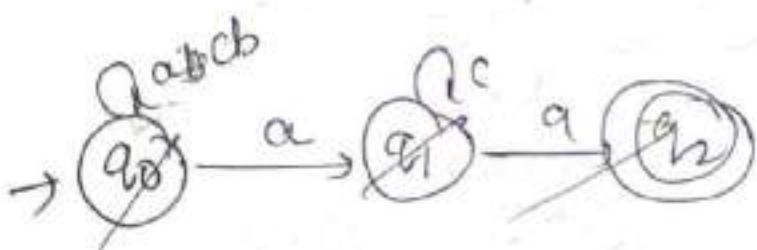
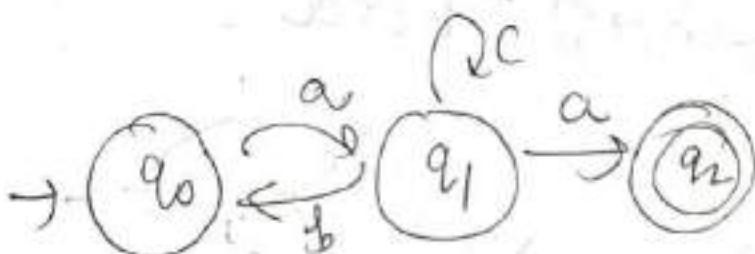


~~(ab)*ab
a+b+b
→ a+b~~

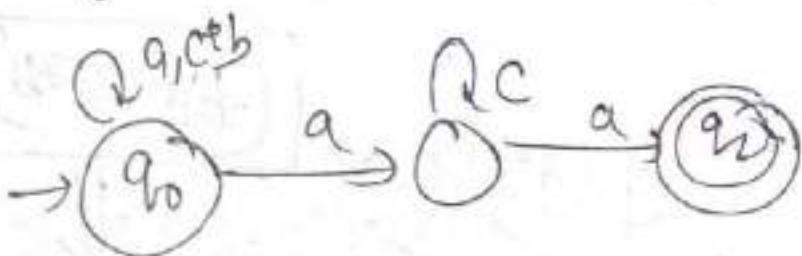


$(ab)^*ab.$

Q.

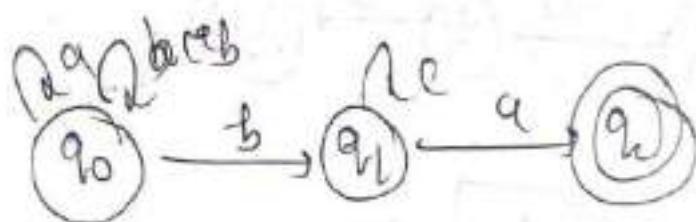
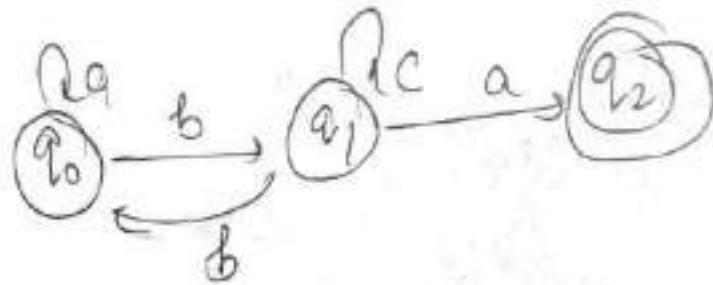


$(ab)^*ac$

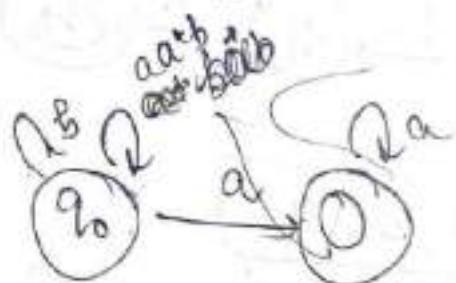
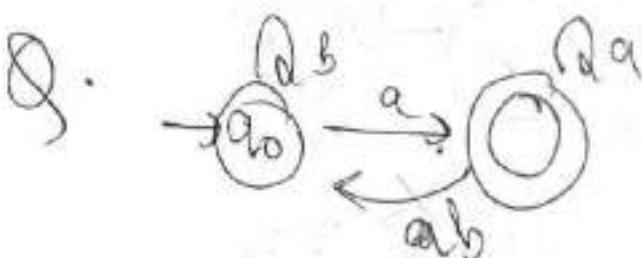


$(ac^*)^*ac$

Q.



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

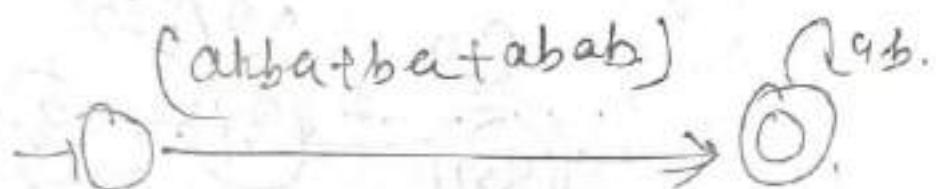
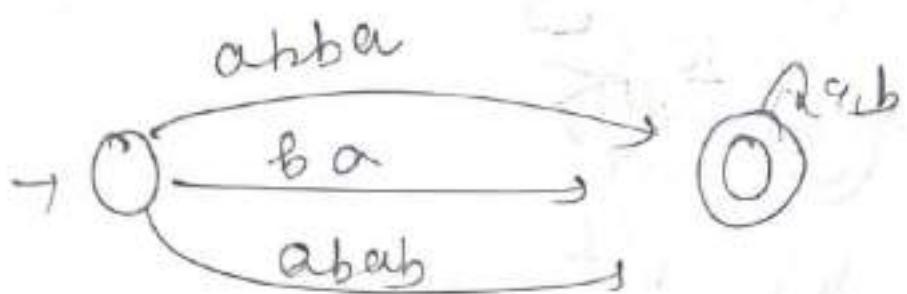
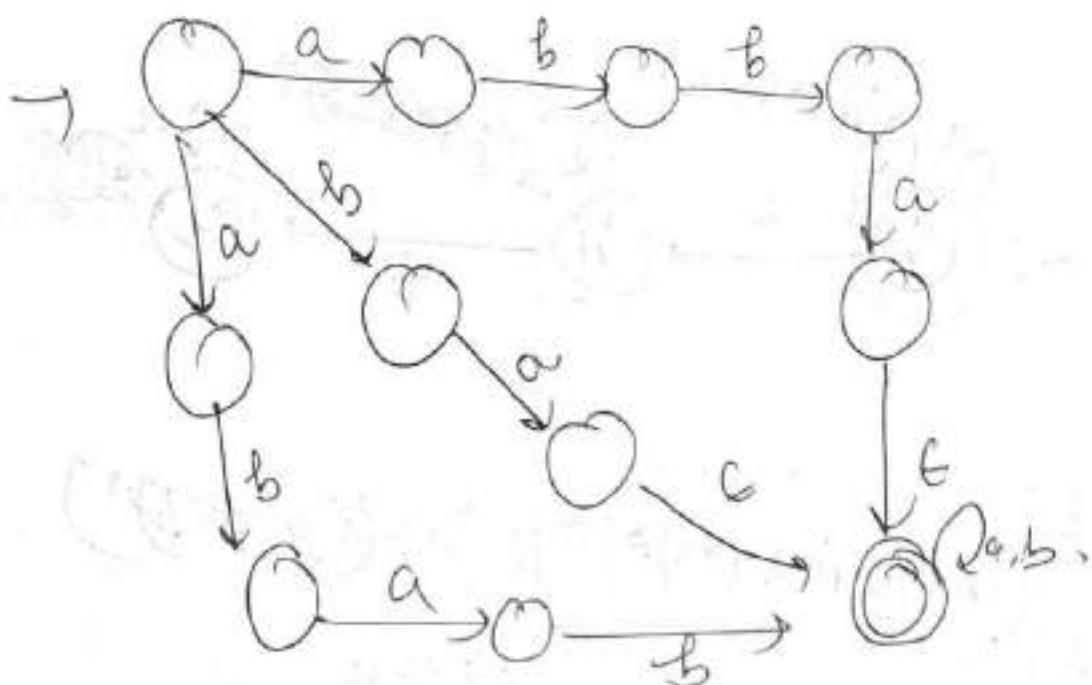


$$\begin{aligned}
 & [(b+a^*)^* a a^*] \quad [(b+a^* a^*)^* a a^*] \\
 & \downarrow \\
 & [b+a^*)^* a^* \\
 & = (a+b)^* a \\
 & = [b^* (a+b)^*]^* a a^*
 \end{aligned}$$

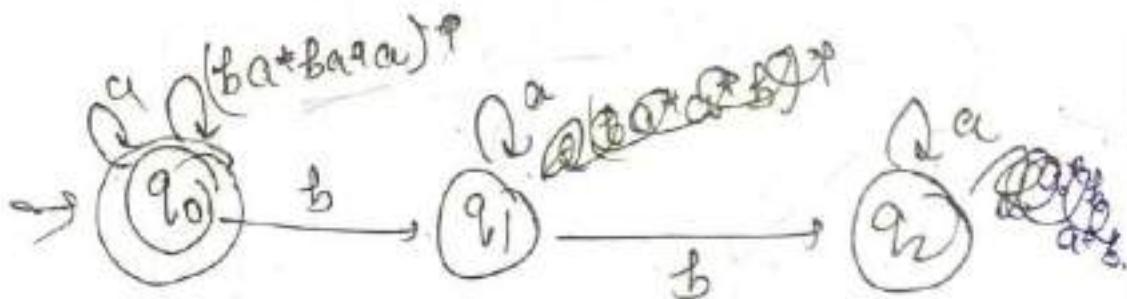
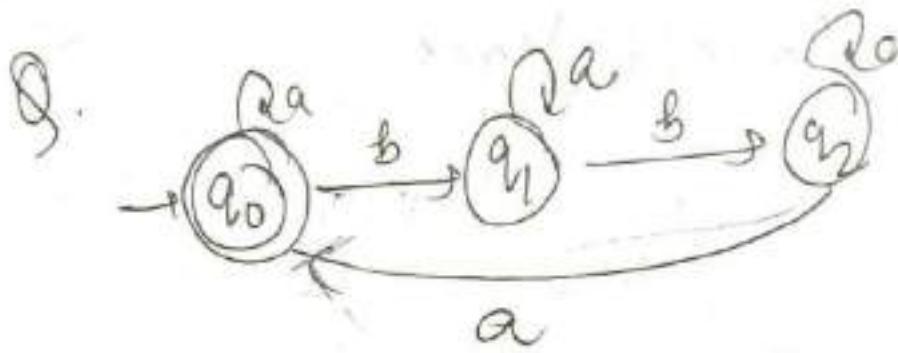
$$= b^* (aa + bb^*) + aat$$

(81)

Q.

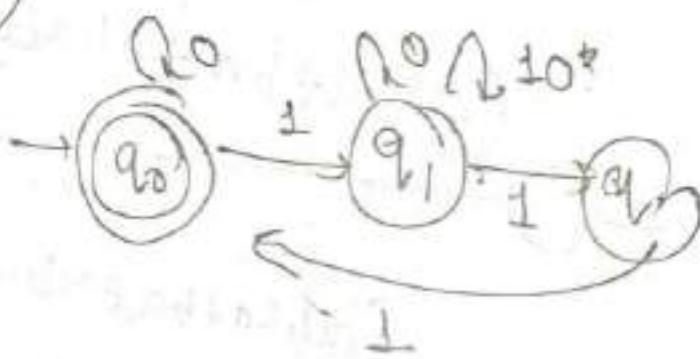
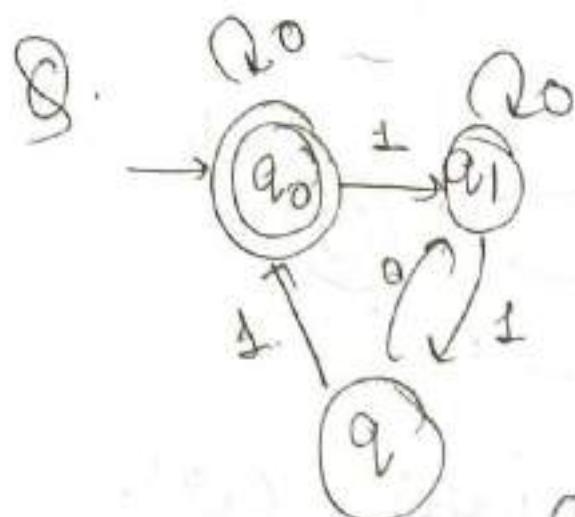


$(abb+bat+abab)(atb)^*$

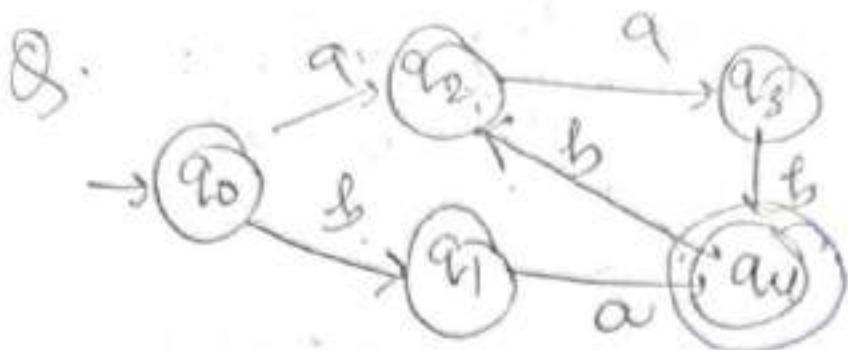


$$(a + (ba+baa)^*)^* \neq (a + (b+a+b^2b)^*)^*$$

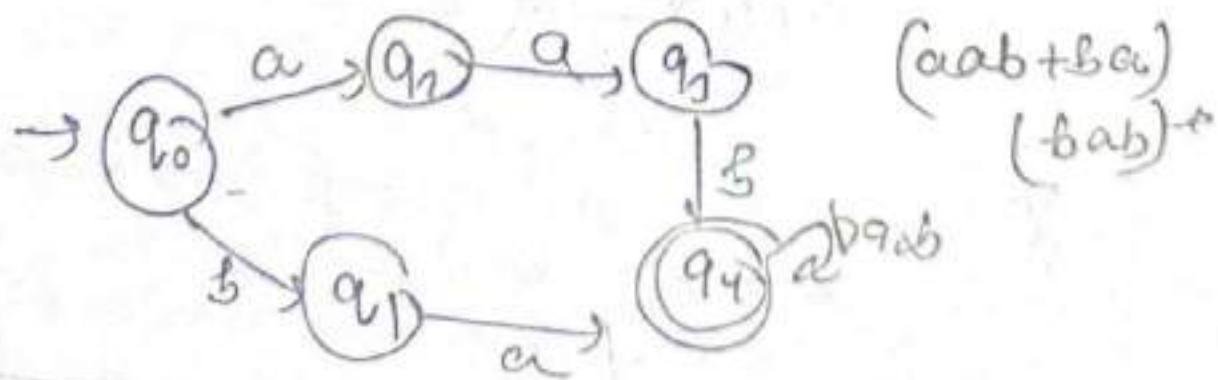
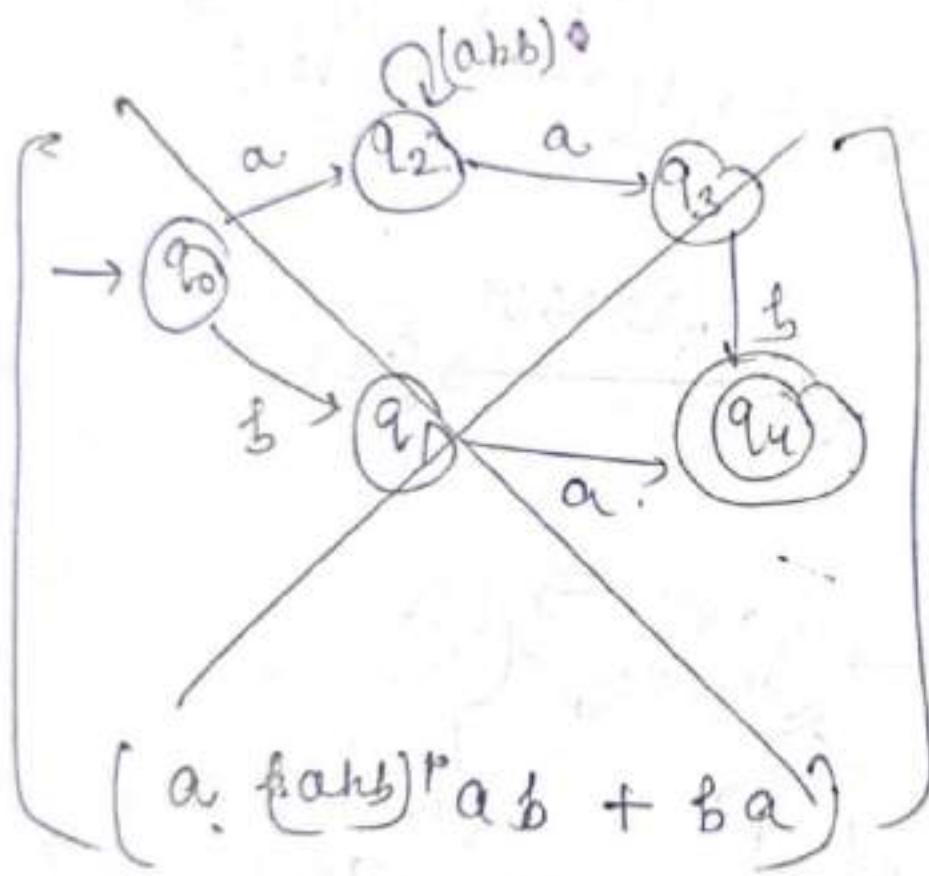
$\cancel{a^*}$

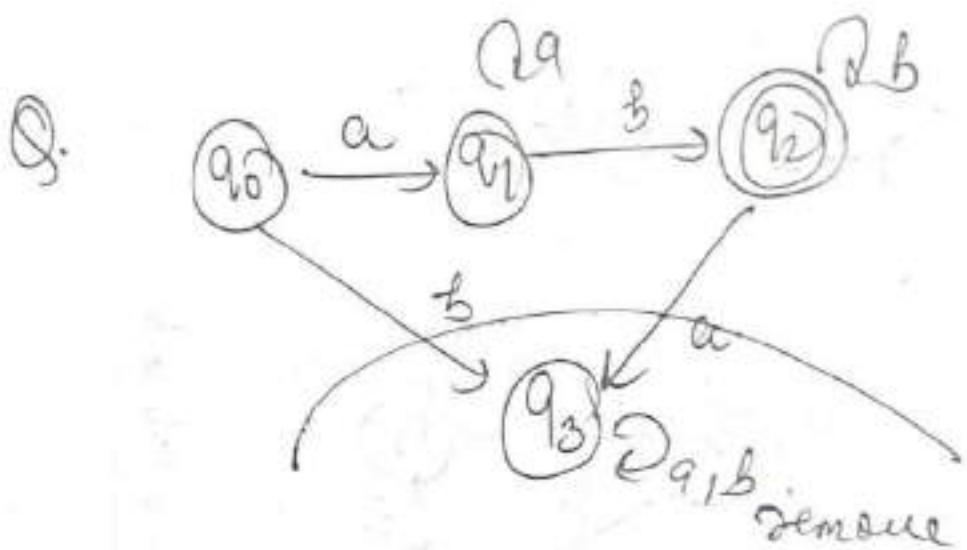


$$(0 + L(0+10^*)^*1 \perp)^*$$

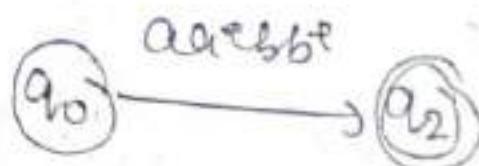
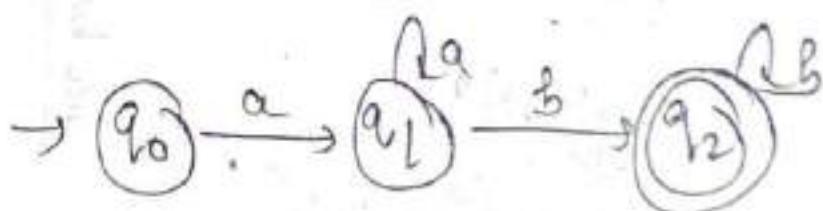


$$\overline{babab + aabbab}$$

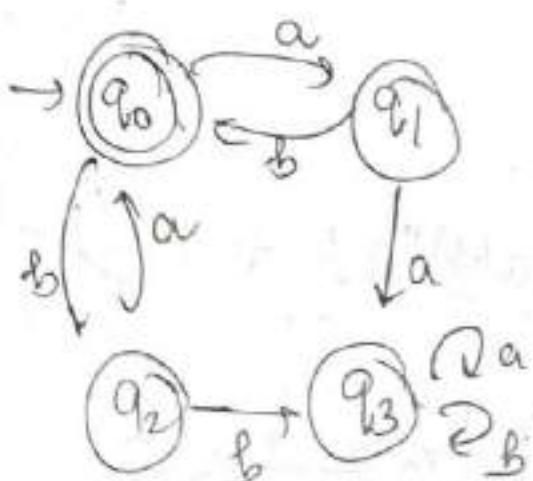




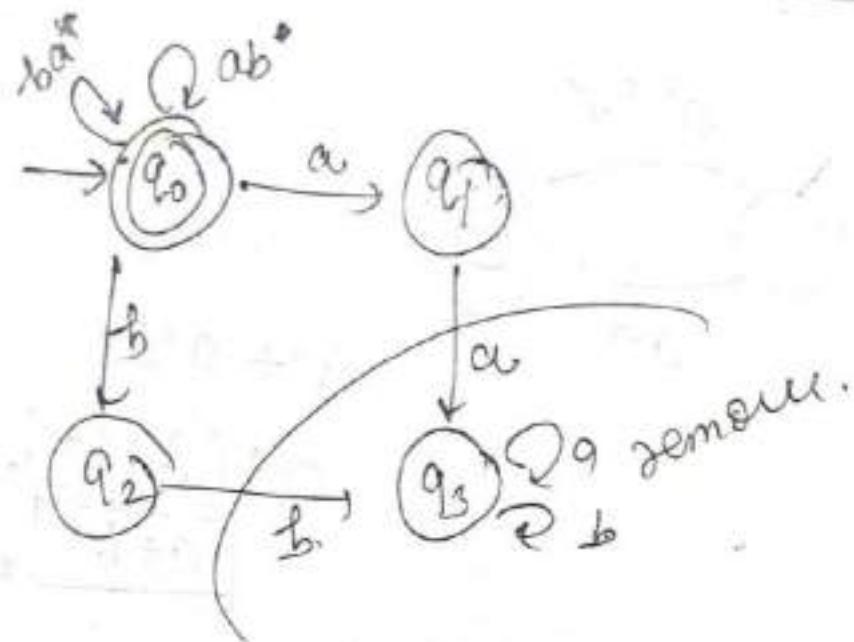
Soluⁿ:



Q.

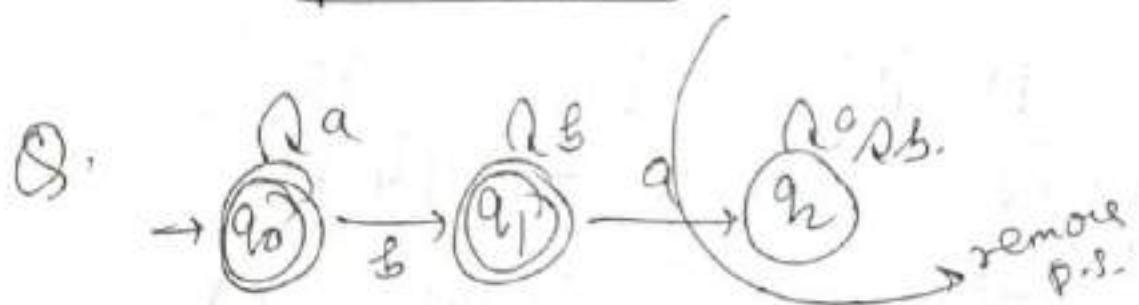


(2)



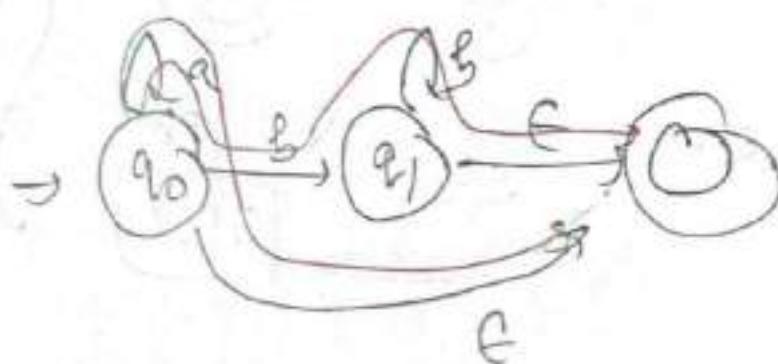
$$\begin{aligned}
 & (\text{ab} + \text{ba})^* \cap a(a+b)^* \\
 & (a(a+b)^*) \cap (a+b)^*
 \end{aligned}$$

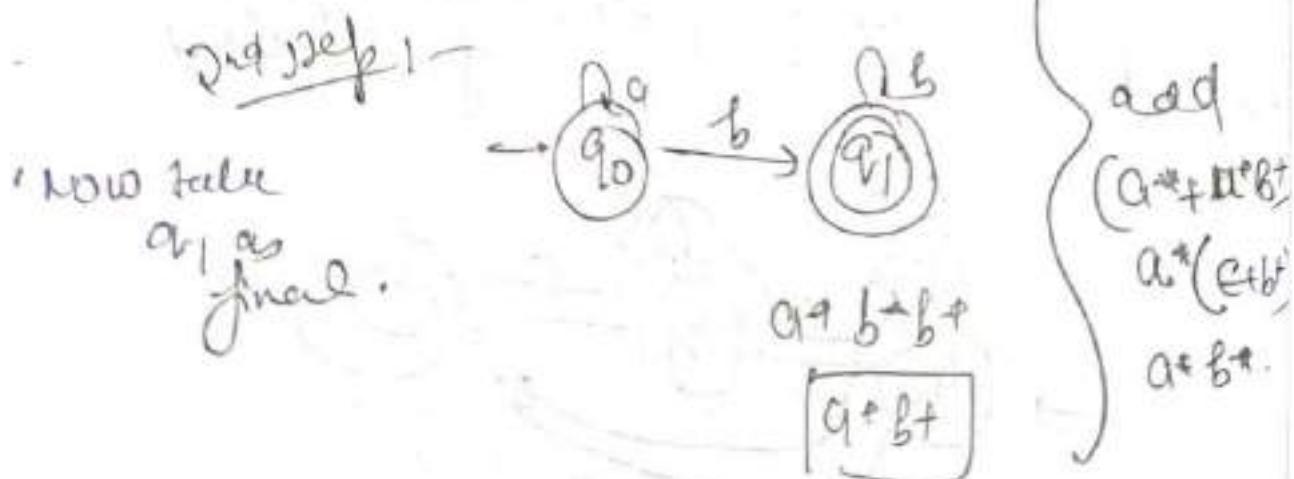
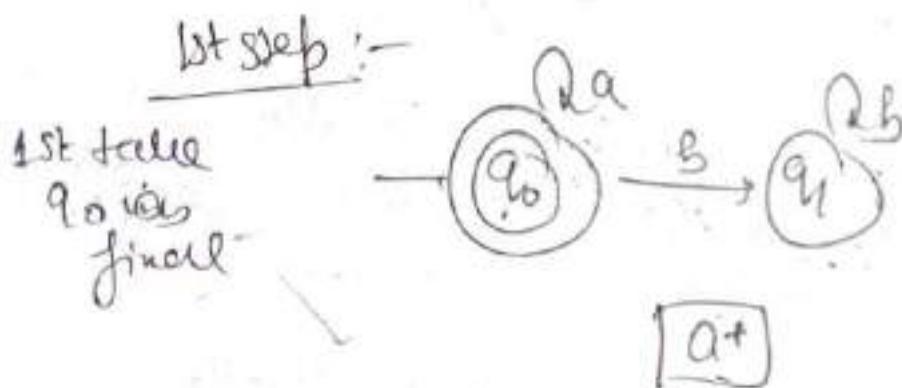
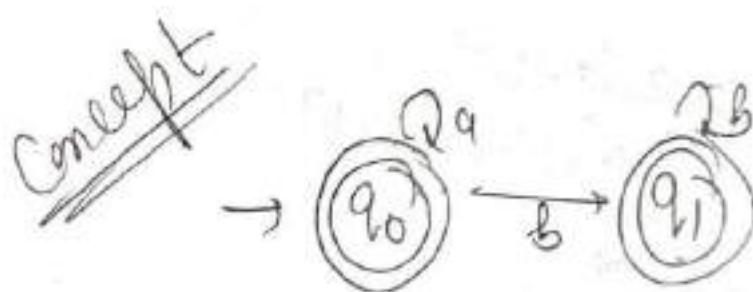
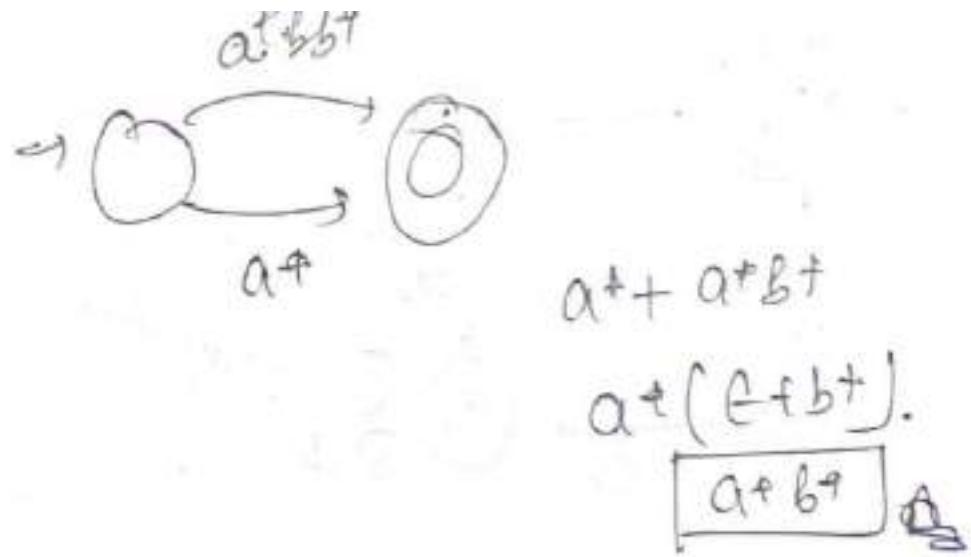
$$\boxed{(ab+ba)^*}$$

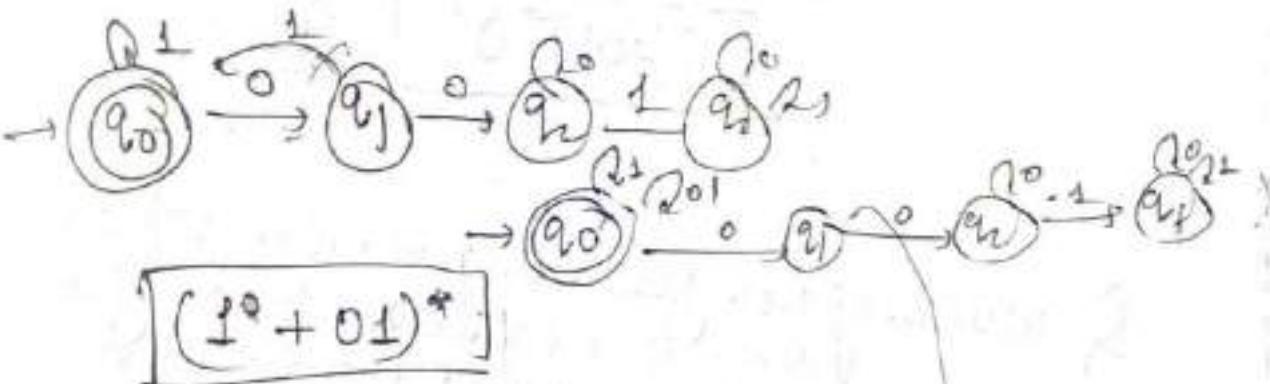
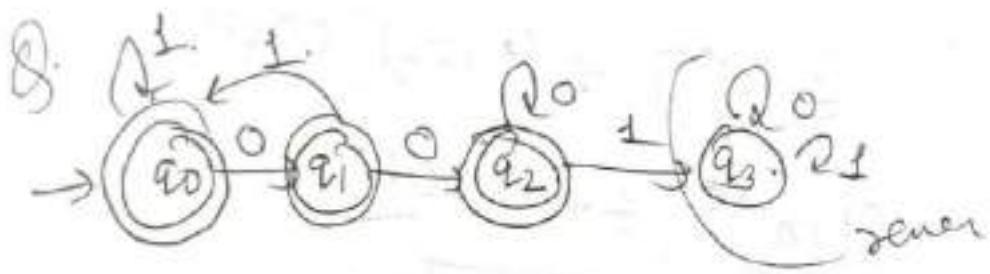


S.Y.I.

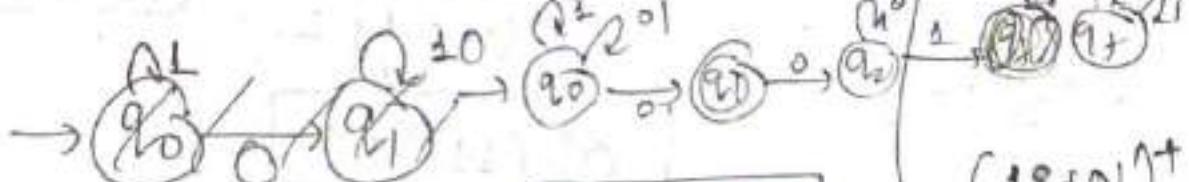
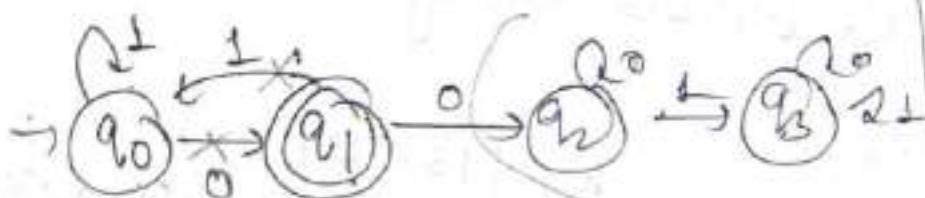
Q 1st make it to single
final state,







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

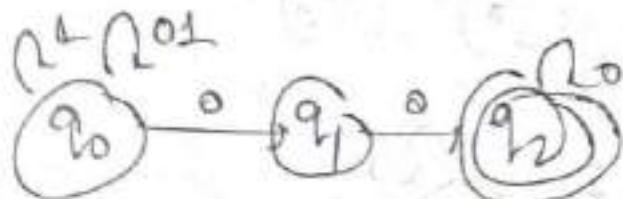
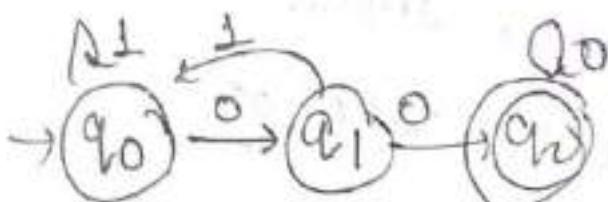


$$(1+01)^* 0$$

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

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

$$+ (1+01)^* 00t$$



$$(1+01)^* 000t$$

$$+ (1+01)^* 00t$$

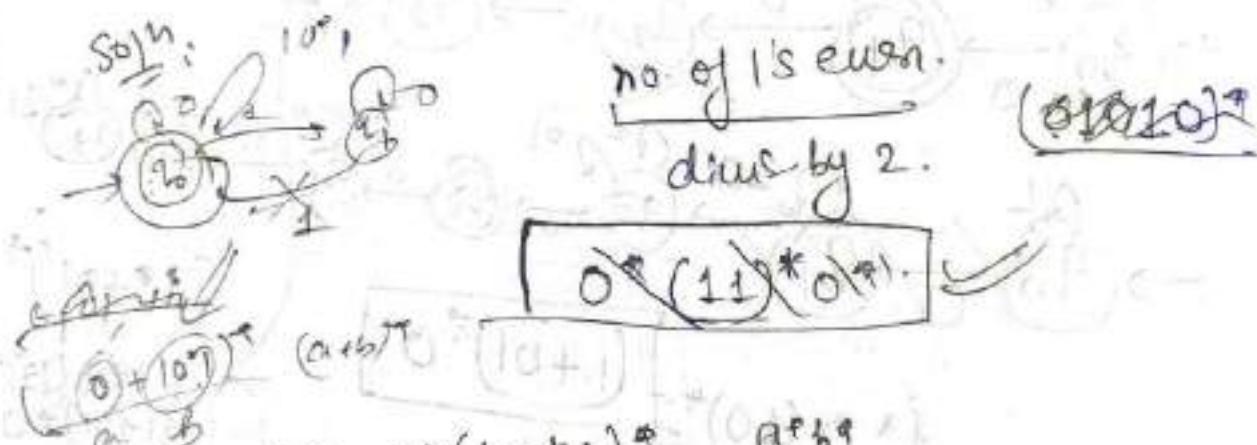
$$(1+01)^* + (1+01)^* 0 + (1+01)^* 0000^*$$

$$(1+01)^* [e+0+00^*]$$

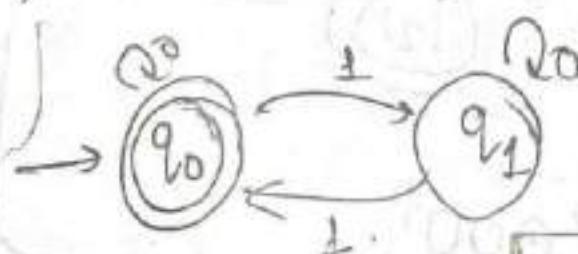
$(1+01)^* 0^*$

Ans.

Q. Which of the following regular expression generates set of all strings of ab where no. of 1's is one even.



- (a) $0^* (10^{+1})^* a^* b^*$
- (b) $0^* (10^{+1})^* 0^* a^* b^*$
- (c) $0^* (10^{+1} 0^*)^* a^* (b a^*)^*$
- (d) $0^* 10^* 10^* (a^* b)^*$



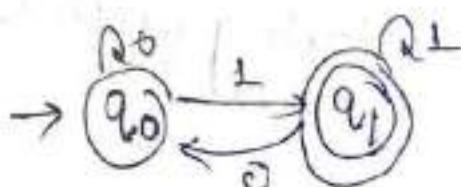
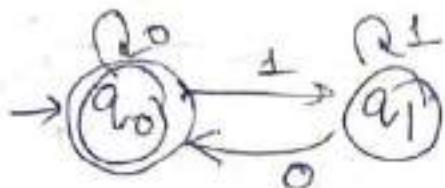
$(0 + 10^{+1})^*$

Q. Find the R.E. that generates all binary nos
not divisible by 2. (85)

S/ All binary nos divisible by 2;
remainders 0, 1

0, 0000, —
0010, 0100, 1000, 1010, —

8/21

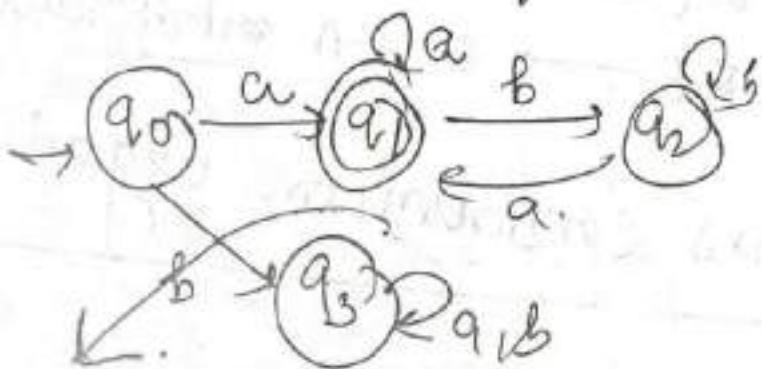


$$(0+1)^* 1^*$$

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

manually.

Q. Which of the following regular expression
is not belongs to the



remove

$$a(a+bba^*)^*$$

* a

- ✓ ①. $a a^* (b b^* a)^*$
 ✗ ②. $a(a+b b^+ a)^*$
 ✗ ③. $a (a b)^* a + a$ (manually)
 ✗ ④. $a(a+a a^+ b b^+ a)^*$
no use
 $\underbrace{a}_{\alpha_1} (a+b b^+ a)^*$

The automata is starting with a & ending with a.

Q.

Regular Expression to FA.

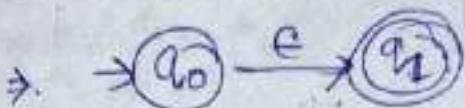
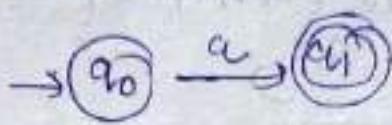
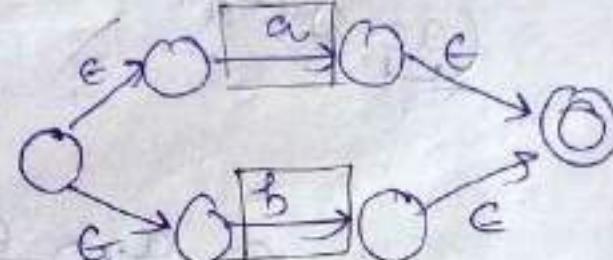
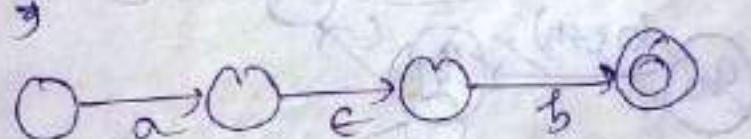
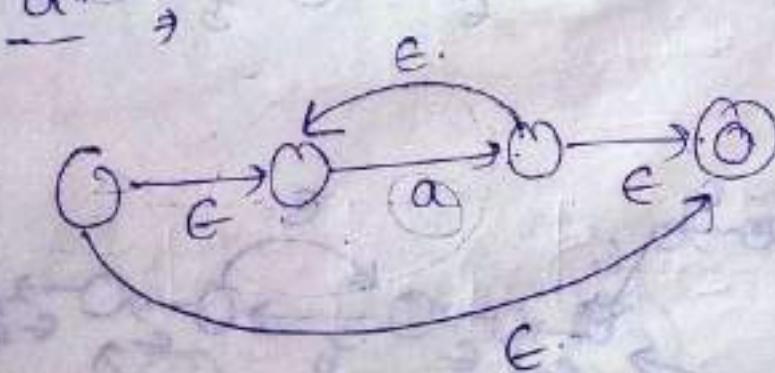
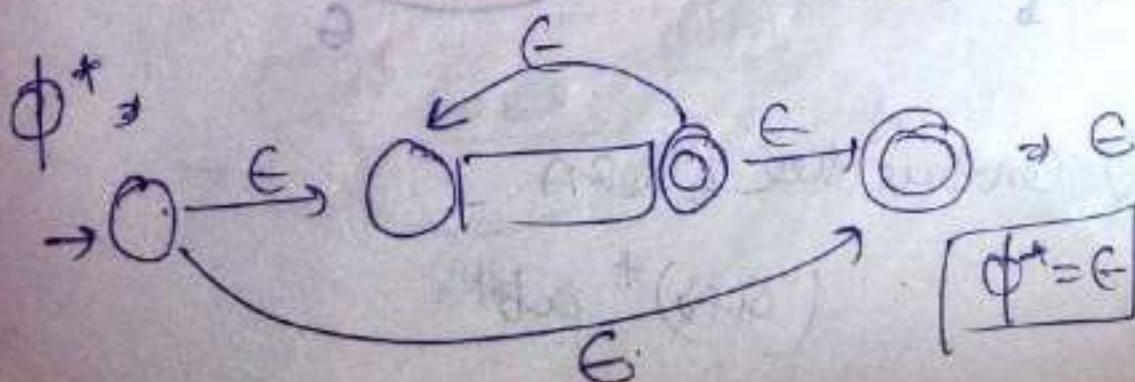
R.E \rightarrow DFA difficult

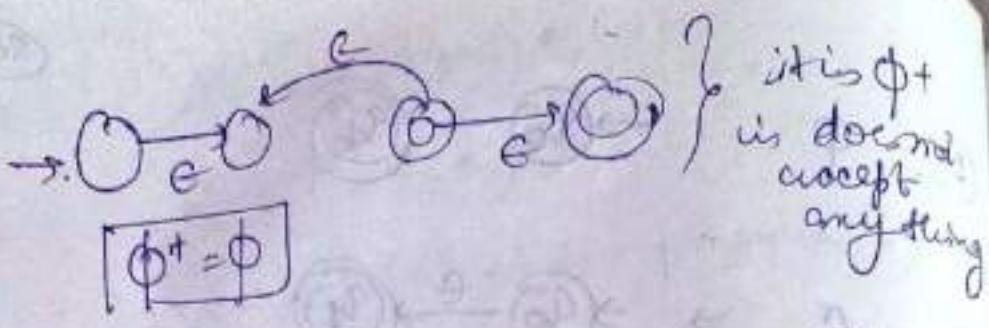
R.E \rightarrow NFA ,

R.E \rightarrow ϵ -NFA not difficult

Thomson's construction algorithm:-

(Not Difficult) ϵ -NFA from R.E \Rightarrow

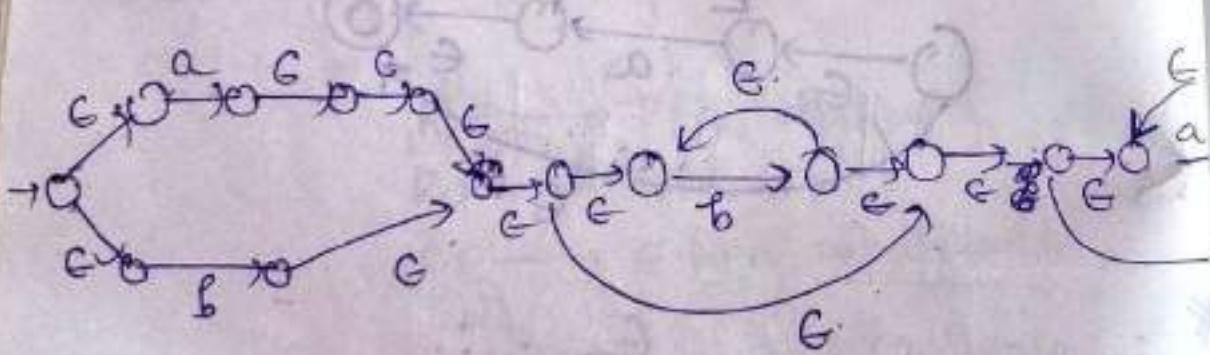
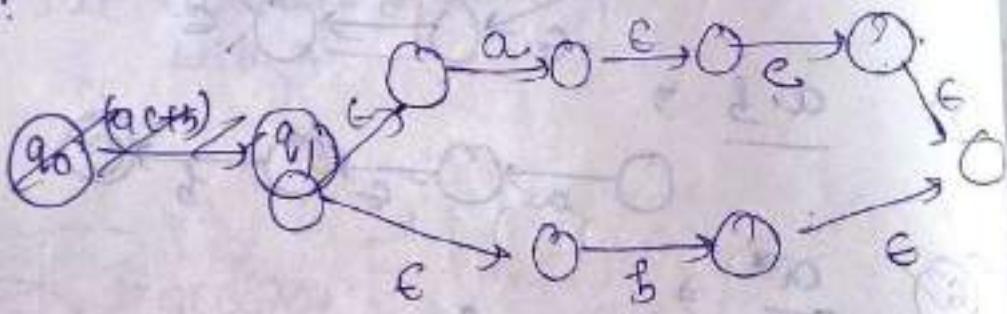
①. ϕ ②. ϵ ③. a ④. $a+b$ ⑤. $a \cdot b$ ⑥. a^*  ϕ^* 



Q. Construct the ϵ -NFA equivalent to following R.E.,

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

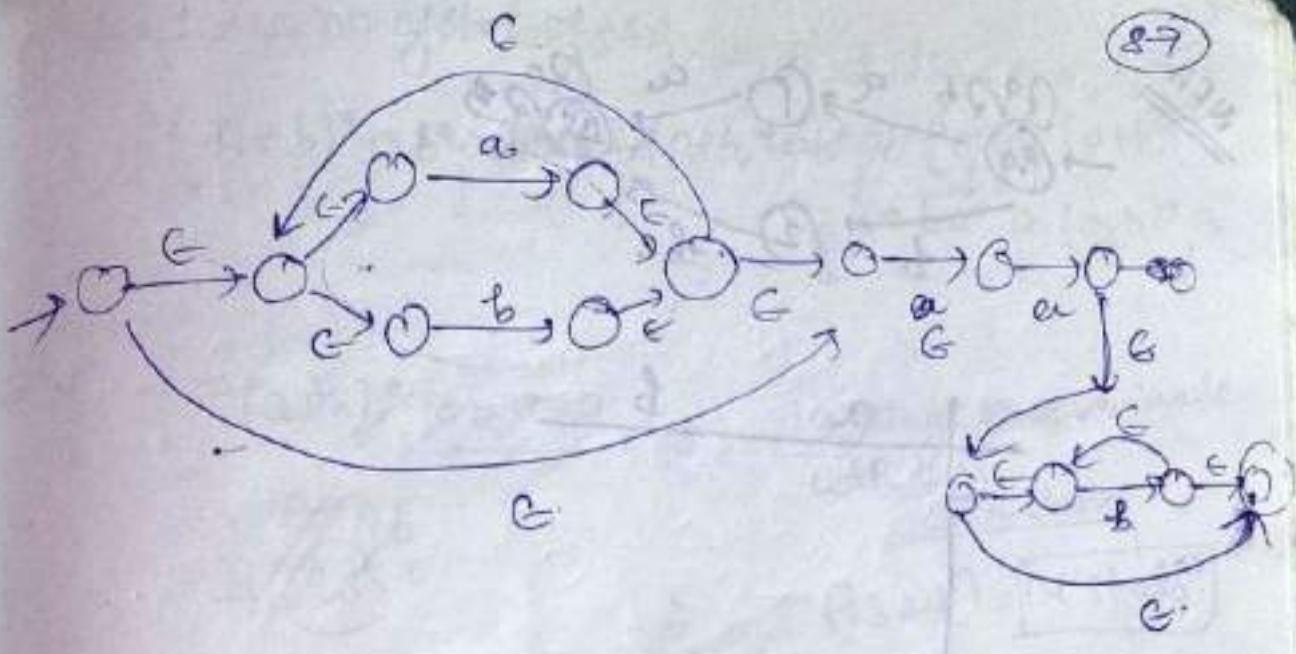
Soln.



Q. Construct the ENFA

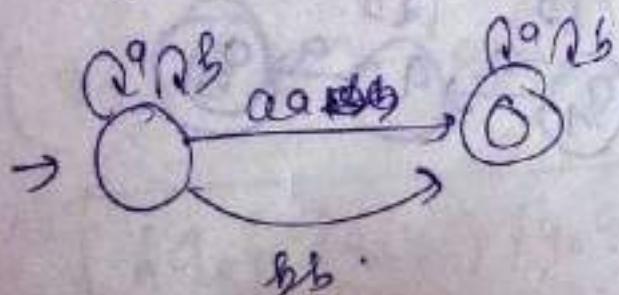
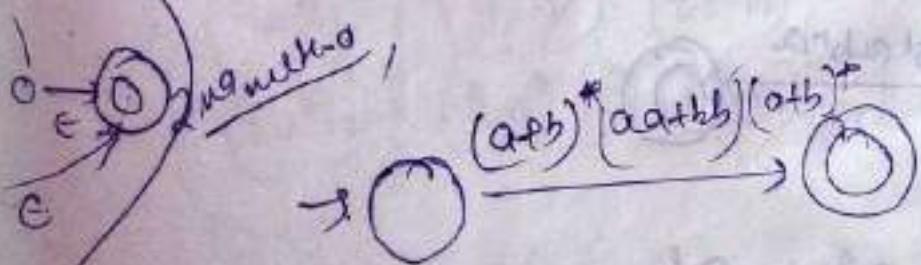
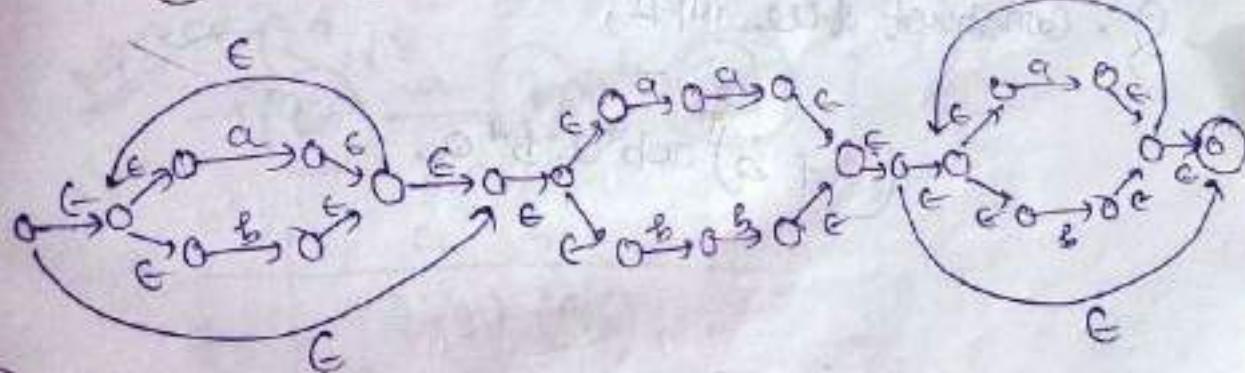
$$\boxed{\phi = \phi}$$

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

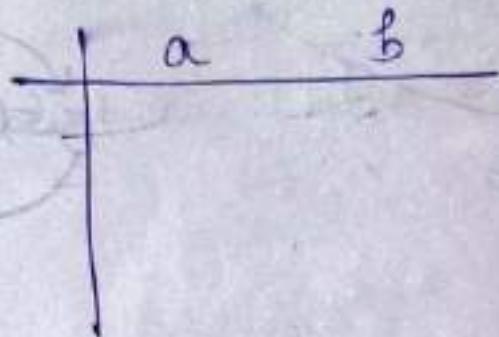
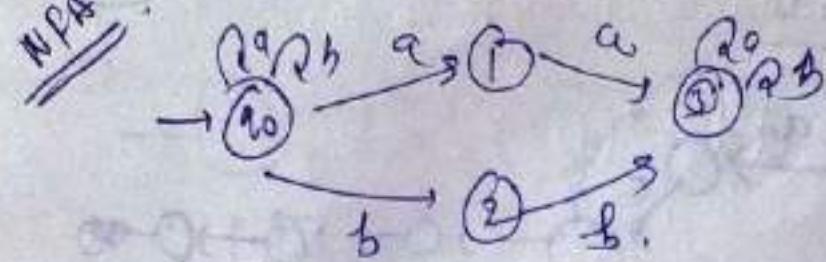


Q Construct NFA,

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

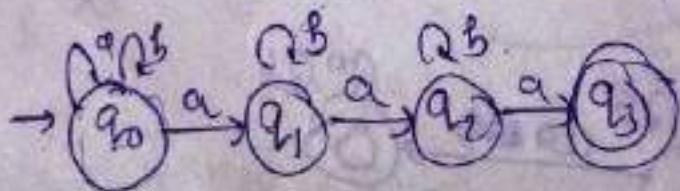
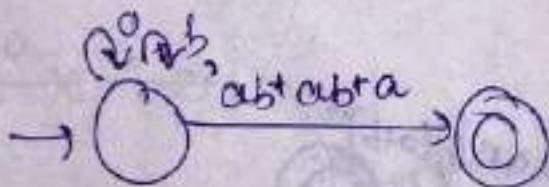


NFA



Q. Construct free NFA,

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



Q. find the no. of states of NFA,

$$(a+b)^* ab^* ab^* a b (a+b)^* ab a b^* (a+b)^* \\ \downarrow \quad \downarrow \\ f a^* b a^* a (a+b)^* a$$

~~32~~

$$(a+b)^* \cancel{ab^*} \cancel{ab^*} a$$

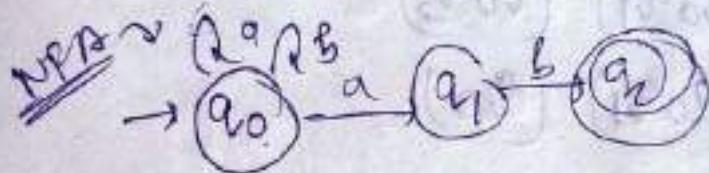


13 single variable
without clear close

$$(13+1) = 14 \text{ states}$$

Q. Construct the minimal DFA,

$$(a+b)^* ab.$$

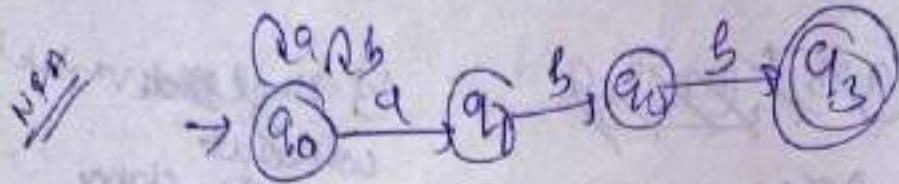


	a	b
q0	(q0, q1)	(q0)
(q0, q1)	(q0, q1)	(q0, q2)
(q0, q2)	(q0, q1)	(q0)

(q0, q1) (q0, q2)

$\boxed{\{q_0\} \{q_0q_1\} \{q_0q_2\}}$

Q. Construct the minimal DFA,
 $(a+b)^* a b$.



	a	b
q_0	$(q_0 q_1)$	q_0
$q_0 q_1$	$(q_0 q_1)$	$(q_0 q_2)$
$q_0 q_2$	$(q_0 q_1)$	$(q_0 q_3)$
$q_0 q_3$	$(q_0 q_1)$	(q_0)

$(q_0, \underline{q_0 q_1}, q_0 q_2) (q_0 q_3)$.

$(q_0, q_0 q_1) (q_0 q_2) (q_0 q_3)$.

$\boxed{\{q_0\} \{q_0 q_1\} \{q_0 q_2\} \{q_0 q_3\}}$

A

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

89

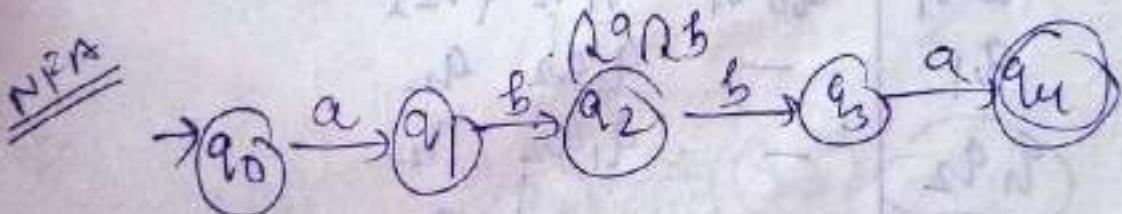
$$(a+b)^* ab \rightarrow \frac{\text{min. DFA}}{3}$$

$$(a+b)^* a+b \rightarrow 4$$

$(a+b)^* abcd \dots n \rightarrow (m+1)$ state
ending with
 m length.

- Q. Construct the minimal DFA, that accept all strings of ab & bb, each string starting with ab & ending with ba.

$$\begin{matrix} \text{aa} & \emptyset \\ ab(a+b)^* ba \end{matrix}$$



	a	b
--	---	---

$q_0 \quad q_1 \quad -$

$q_1 \quad - \quad q_2$

$q_2 \quad q_2 \quad (q_2 q_3)$

$(q_2 q_3) \quad (q_2 q_4) \quad (q_2 q_3)$

$(q_2 q_4) \quad (q_2) \quad (q_2 q_3)$

$q_5 \quad DS \quad DS$

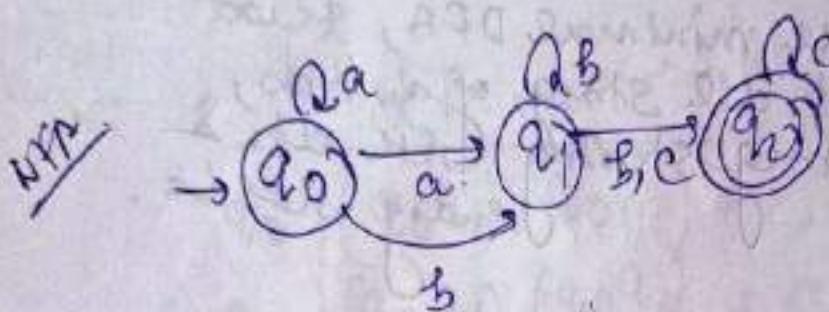
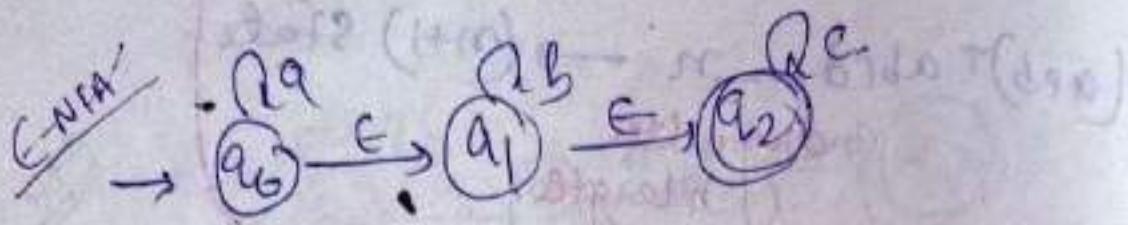
$(q_0 q_1 q_2 q_3)^*$

$(q_2 q_4)$

3 staty

Q. Construct minimal DFA, $a^* b^* c^*$

$a^* b^* c^* \leftarrow d^*(d+b)$



	a	b	c
q_0	$q_0 q_1$	q_1	-
$q_0 q_1$	$q_0 q_1$	$q_1 q_2$	$\{q_2\}$
q_1	-	$q_1 q_2$	q_2
q_2	-	$q_1 q_2$	q_2
q_3	-	-	q_2

$(q_0 \quad q_0 q_1 \quad q_1) \quad (q_1 \quad q_2 \quad q_2)$. D.S.

$\Rightarrow (q_0) \quad (q_0 q_1 \quad q_2) \quad (q_0 q_2 \quad q_2)$. D.S.

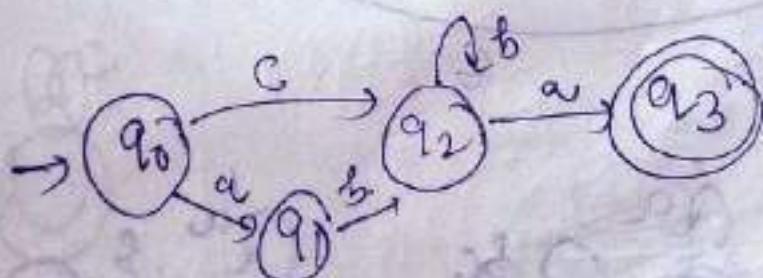
(q_0)

$$\begin{aligned}
 & \underbrace{a^k b^k}_{2} \rightarrow 3 \\
 & \underbrace{a^k b^k c^k}_{3} \rightarrow 4. \quad (\text{no. of symbols} + 1) \\
 & \underbrace{a^k b^k c^k d^k}_{4} \rightarrow 5 \quad \text{States} \\
 & \underbrace{a^k b^k c^k d^k e^k}_{5} \rightarrow 27 \\
 & \underbrace{a^k b^k c^k d^k e^k f^k}_{6} \rightarrow 11. \\
 & \underbrace{\dots}_{10 \text{ symbols}}
 \end{aligned}$$

Q. Construct the minimal DFA,

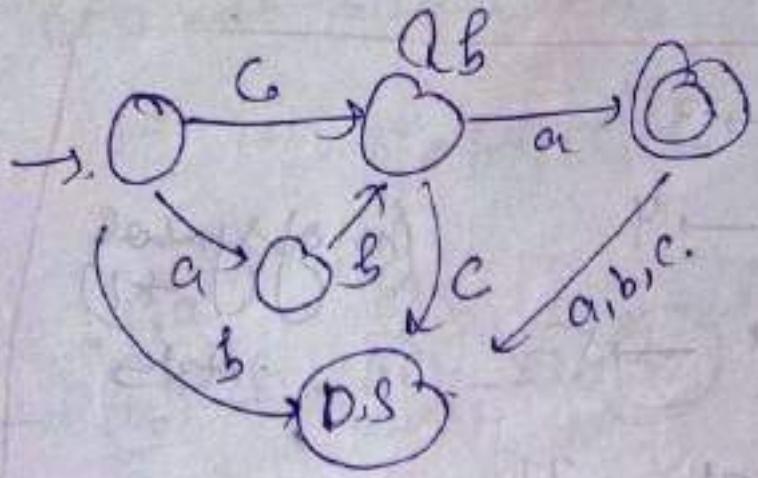
$$L = \{(c+ab)^* b^* a\}$$

so,



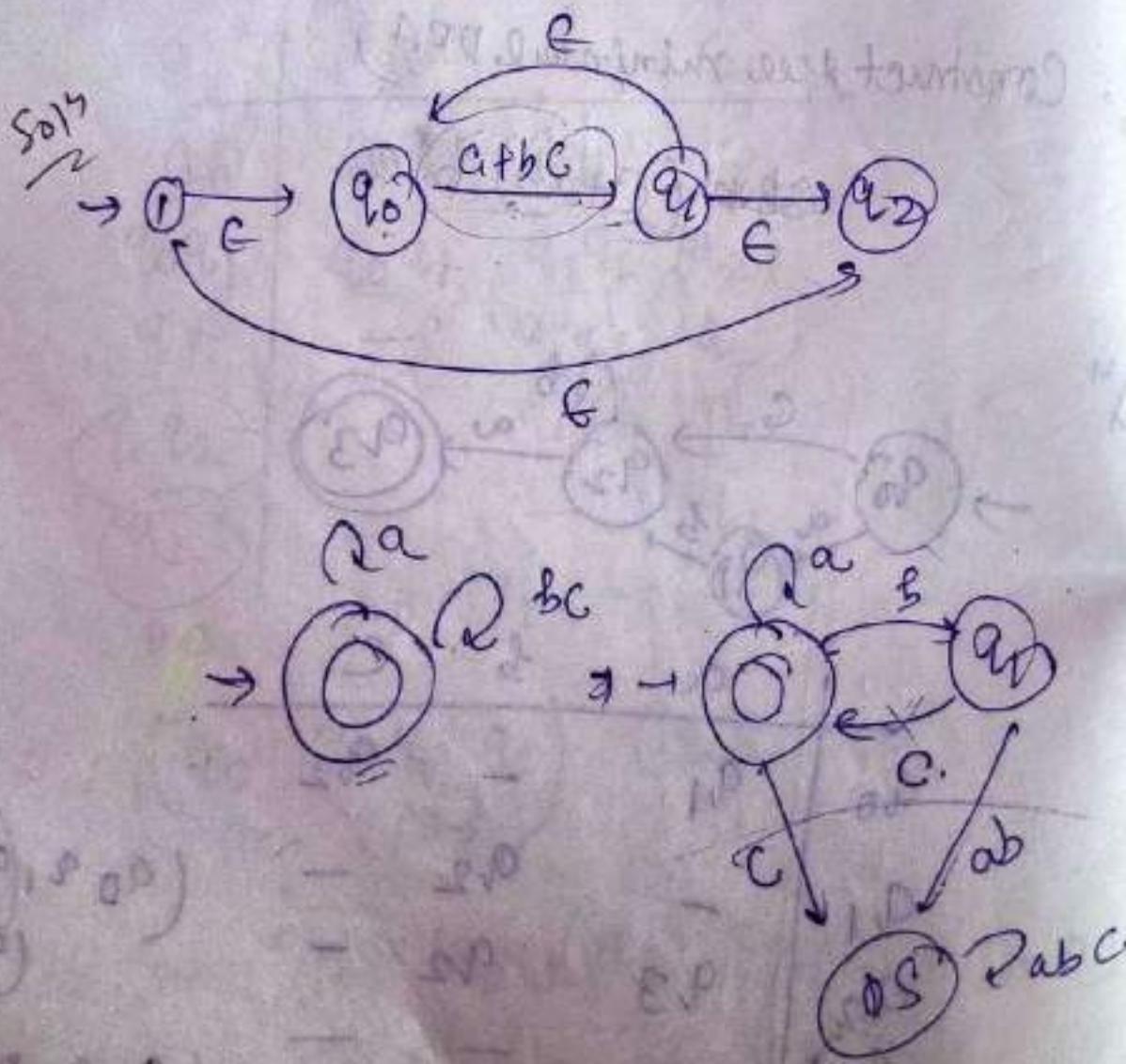
	a	b	c
q0	q1	-	q2
q1	-	q2	-
q2	q3	q2	-
q3	-	-	-
D.S.	-	-	-

$(q_0, q_1, q_2) \text{ D.S.}$
 (q_3)
 $(q_0, q_1, q_2) \text{ D.S.}$
 (q_3)
 $(q_0, q_1) \text{ D.S.}$
 $(q_2, q_3) \text{ D.S.}$



Q. construct the minimal DFA,

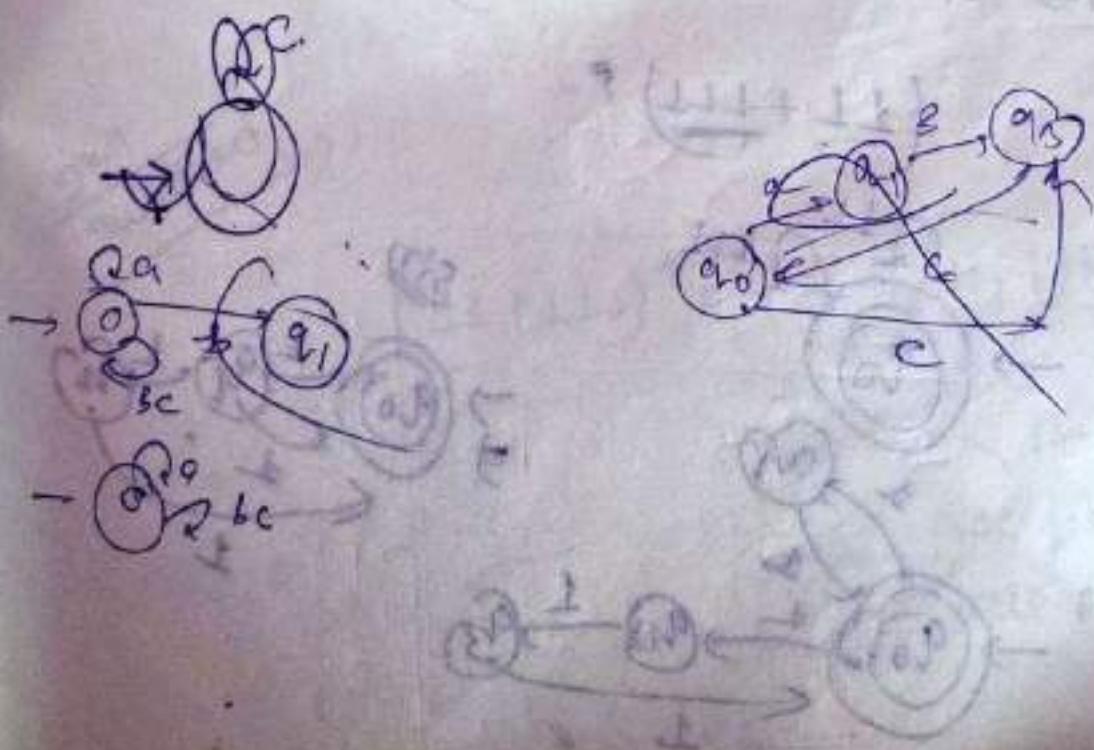
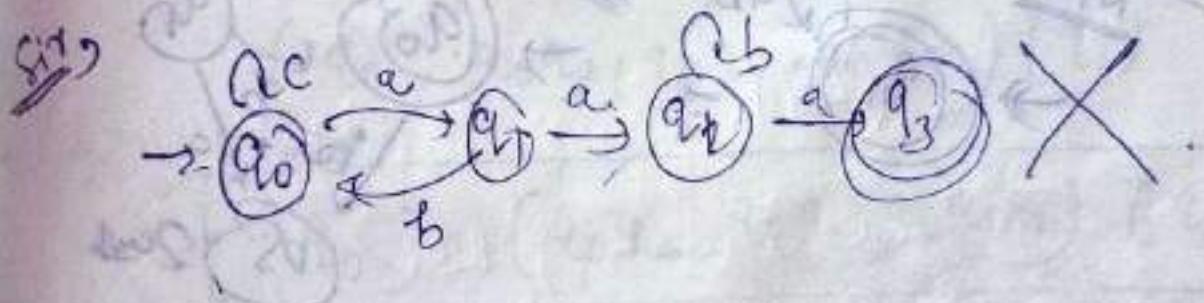
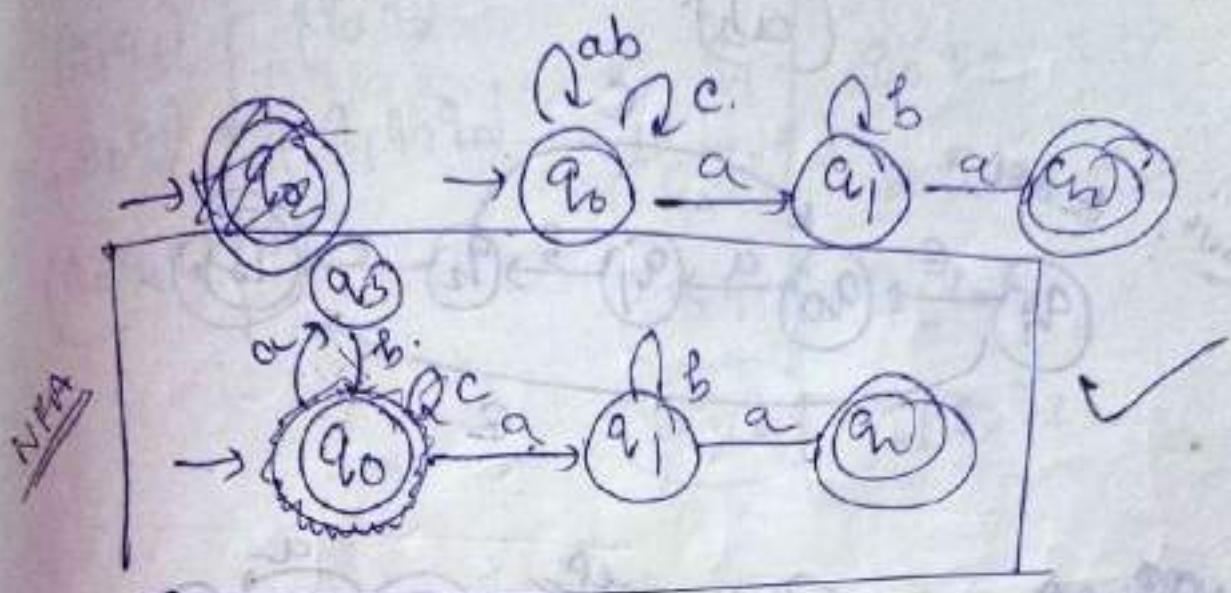
$$(a+b+c)^*$$



(91)

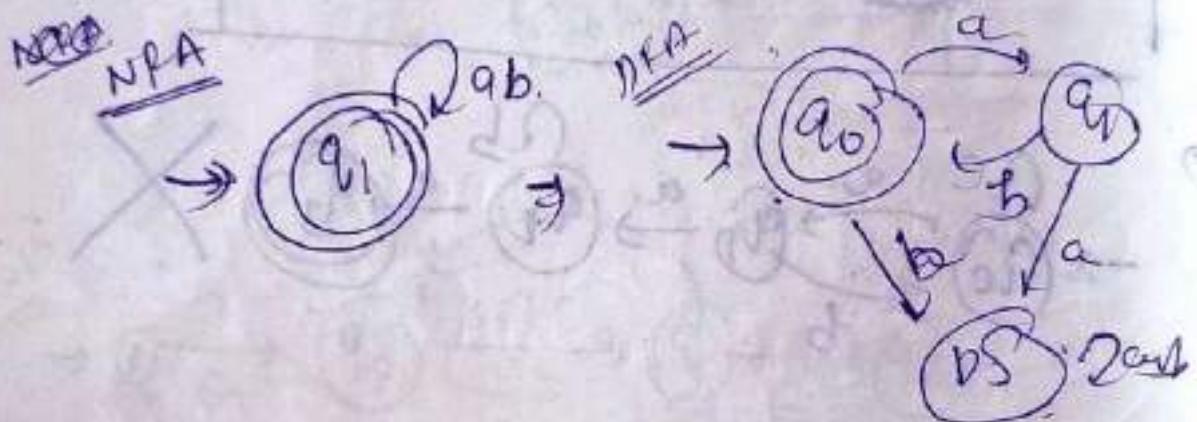
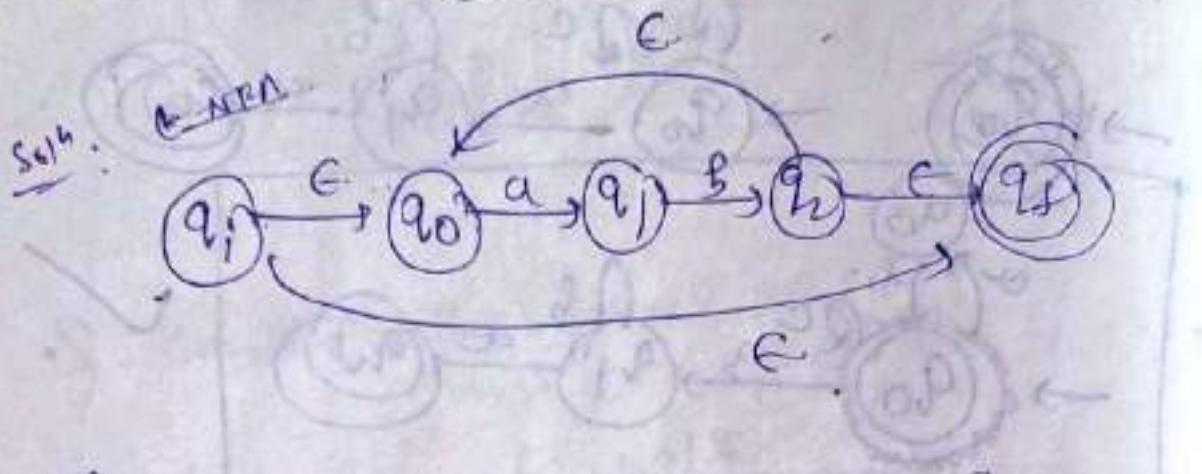
$$Q: \underline{(ab+c)^*ab^*a}$$

minimal DFA = ?



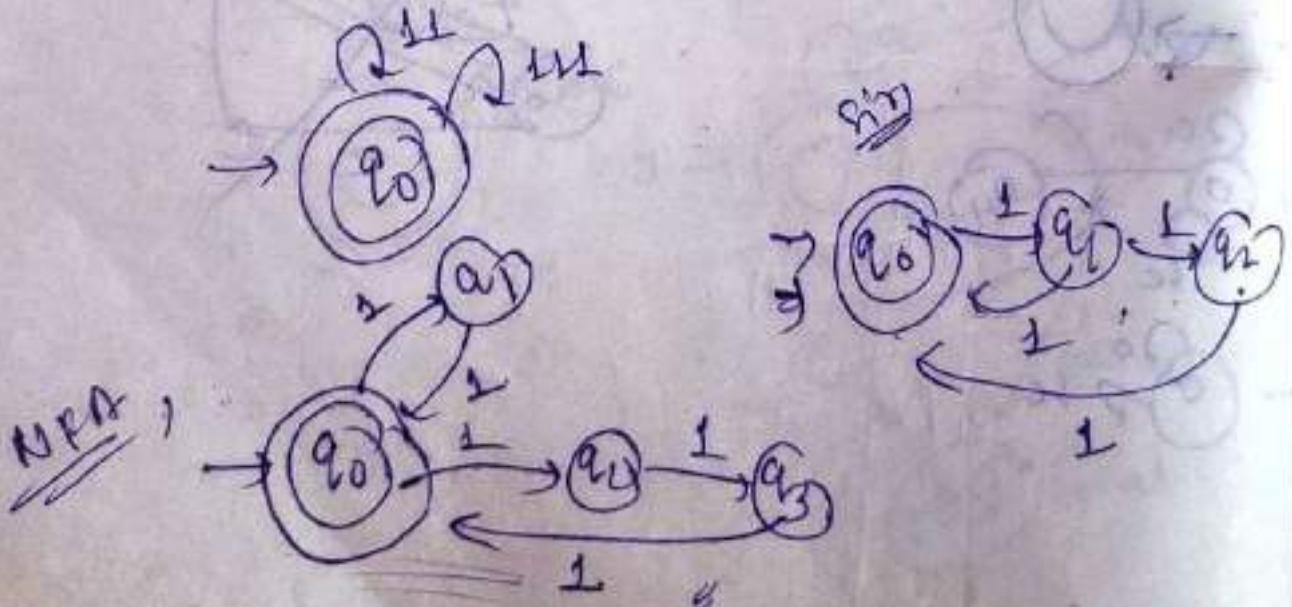
Q. Construct the minimal DFA,

$$(ab)^*$$



Q. Construct the minimal DFA,

$$(11 + 111)^*$$



(92)

		1
		2
1	1	
q_0	($q_1 q_2$)	q_0
($q_1 q_2$)	($q_0 q_3$)	q_1
($q_0 q_3$)	($q_1 q_2 q_0$)	$q_0 q_2$
($q_0 q_1 q_3$)		$q_1 q_0$
		$q_1 q_0 q_2$
		$q_1 q_0 q_2$

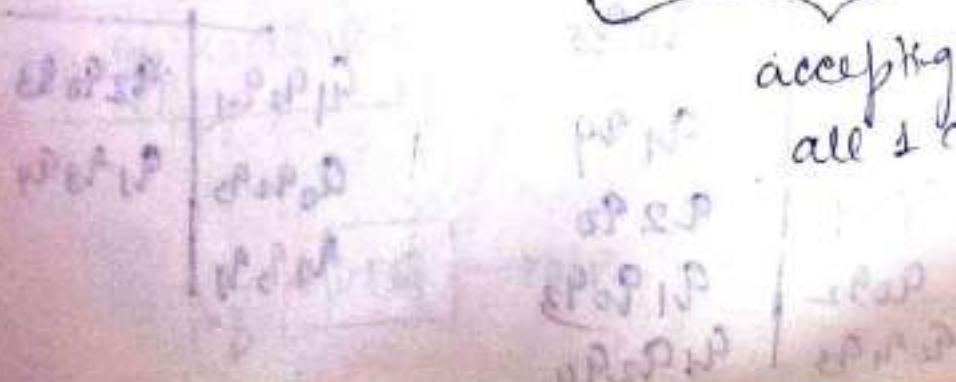
$(q_0, \underline{q_0 q_2}, \underline{q_0 q_1}, \underline{q_0 q_3}) (q_1)$

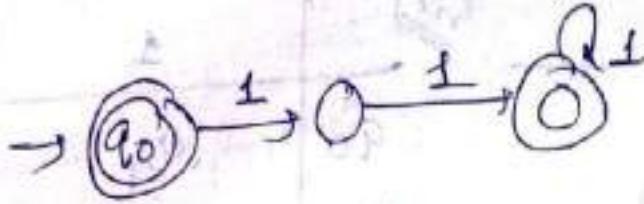
$\boxed{(q_0) (q_0 q_2, q_0 q_1, q_0 q_3) (q_1)}$

~~2nd method~~

$(11+111)^t = \{ \underline{\epsilon}, \underline{11}, \underline{111}, \underline{1111}, \underline{11111}, \text{etc.} \}$

accepting single 1
all 1 coming.



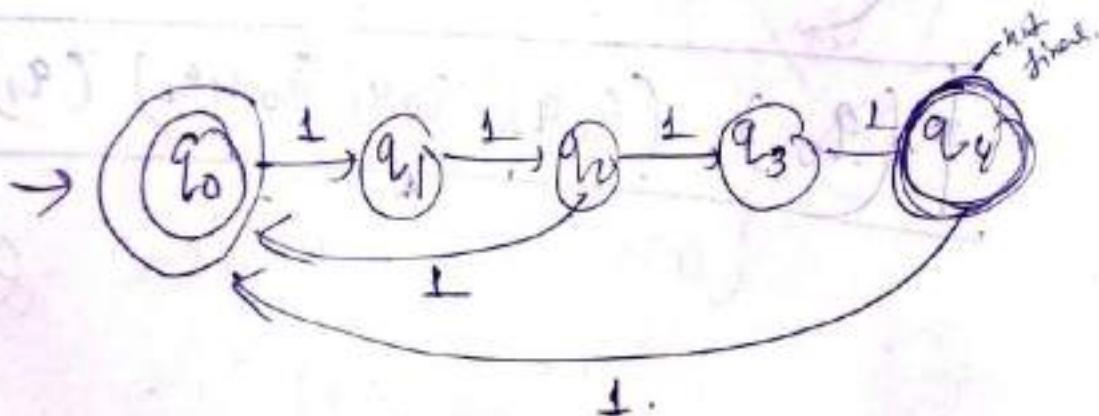


~~Gate 2006~~ Q. $L = (\underline{111} + \underline{1111})^*$. How many no. of states in minimal DFA that accept L.

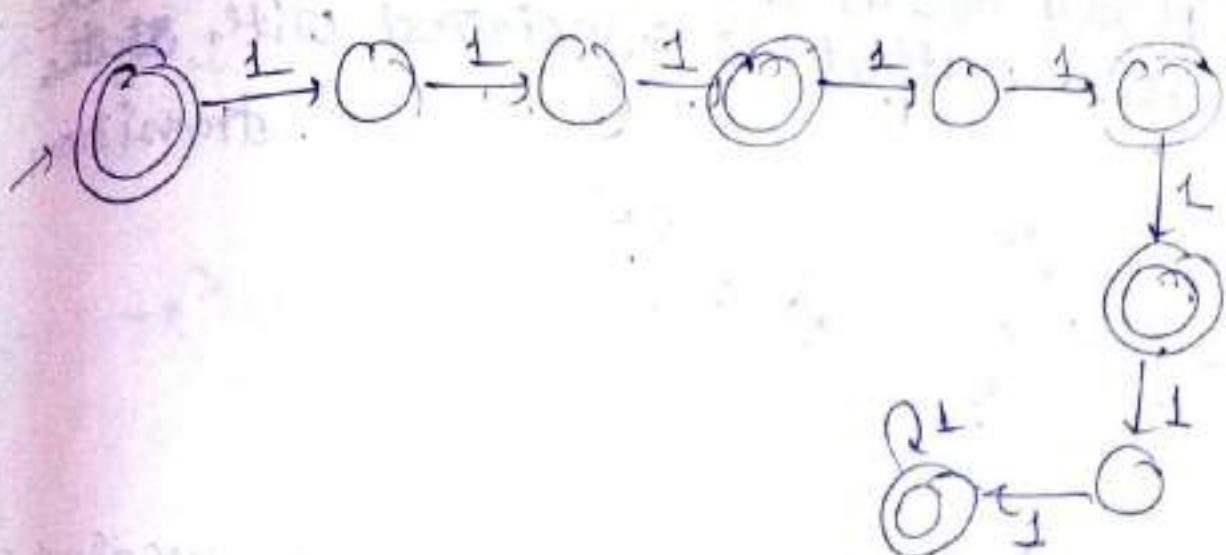
$L = \{\epsilon, 111, 1111, 11111, \dots\}$

$\begin{matrix} 1 & 3 & 5 & 6 \\ \swarrow & \downarrow & \downarrow & \searrow \\ 1 & 2 & 7 & 8, 9, 10, 11, 12, 13, 14, \dots \end{matrix}$

 all comes after 8
 \Rightarrow accept 1, 2, 7 all possible.



	1		
q_0	q_1	$(q_1 q_2)$	$\cancel{(q_1 q_2 q_3)}$
q_1	q_2		
q_2	$q_0 q_3$		
$q_0 q_3$	$q_1 q_4$	$q_1 q_2 q_4$	$q_2 q_0 q_3$
$q_1 q_4$	$q_2 q_0$	$q_0 q_2 q_3$	$q_1 q_3 q_4$
$q_0 q_2$	$q_1 q_0 q_3$	$q_1 q_3 q_1$	
$q_0 q_1 q_3$	$q_1 q_2 q_4$		



Total 9 states in minimal DFA.

Finite automata as output

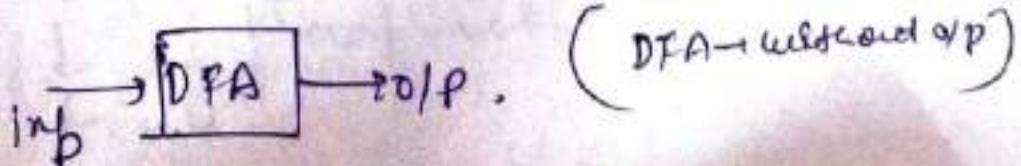
Generator :-

or

(F.A with O/P)

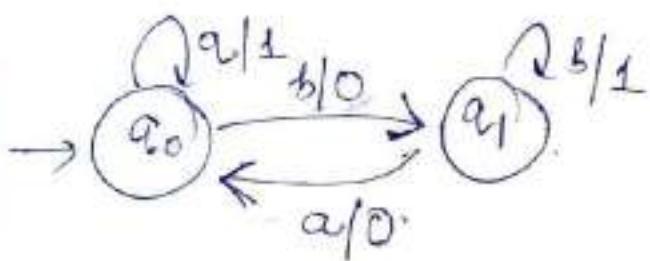
Mealy Machine

Moore m/c



Mealy M/C

① Mealy M/e is a mathematical model in which output is associated with ~~states~~ transitions.



Formal definition: — (for both mealy & moore)

$$(Q, \Sigma, q_0, \Delta, \delta, \chi)$$

Q : finite no. of states

Σ : input alphabet

q_0 : initial state

Δ : O/P alphabet

δ : transition function

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

λ : O/P function (~~now the O/P depends~~)

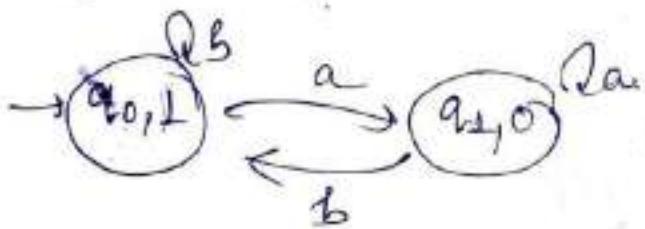
Mealy: $\lambda: Q \times \Sigma \rightarrow \Delta$ output depends on transitions

Moore: $\lambda: Q \rightarrow \Delta$

Moore m/c

(94)

- Moore m/c is a mathematical model, in which output is associated with states.

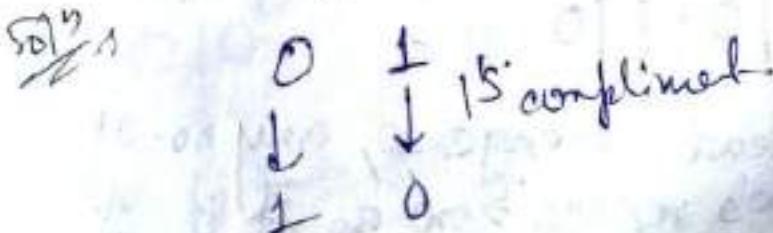


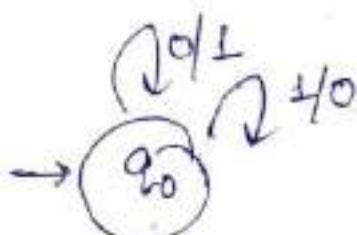
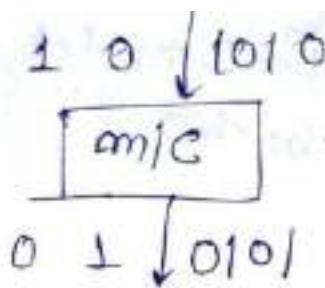
- Both mealy and moore m/c's are deterministic in nature. Hence, on each & every state on every input symbol exactly one transition should exist.

- There is no final in mealy & moore m/c's because they are not language recognizers.
- Mealy & Moore m/c's are practically used in the circuit designing areas.

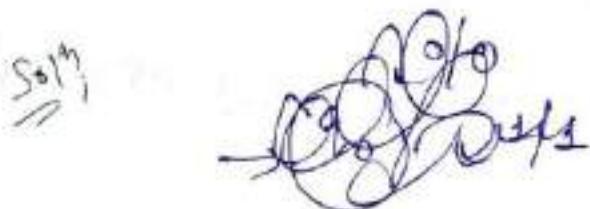
Mealy m/c :-

- Construct the mealy m/c, that represents the behaviour of 1's compliment circuit of binary no.





Q2 Consider the newly m/c treat represent the behavior of 2's complement of binary no.



Assume that we are reading the string from LSB to MSB & carry's discarded.

2's complement \rightarrow

(1) 0|0|00

$$\begin{array}{r} 0|0|00 \\ \downarrow \quad \downarrow \quad \downarrow \\ 101100 \end{array}$$

(2) 0|00|00|0

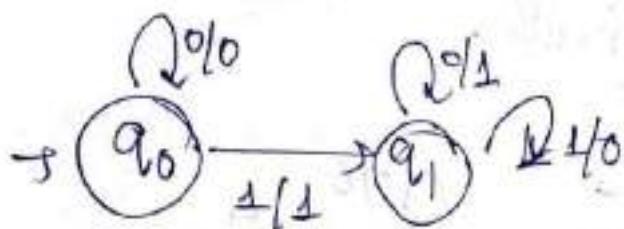
|0|1|0|1|1|0

LSB se read karenge so, any no. of zeros replace zero as it is, 1st one

Ans 4

(95)

from LSB replace it by 1 after that
complement these bits.



at 1 behavior is change
that's why we change the state.

Q.3 Construct the mealy m/c that represent
the behavior of binary incrementer (it increments
(it increments the given binary no by 1).
(same code)

Soln:

$$\begin{array}{r}
 01010 \\
 + 0101 \\
 \hline
 01101
 \end{array}$$

000 input

↓ output

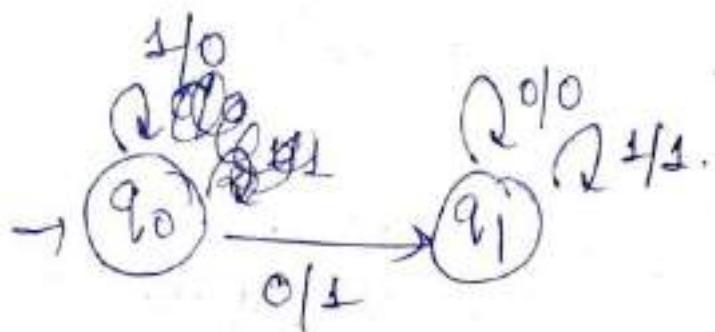
001
as it is) Any no of 1's
make it as zero.

010

↓
011 → 100 → 101

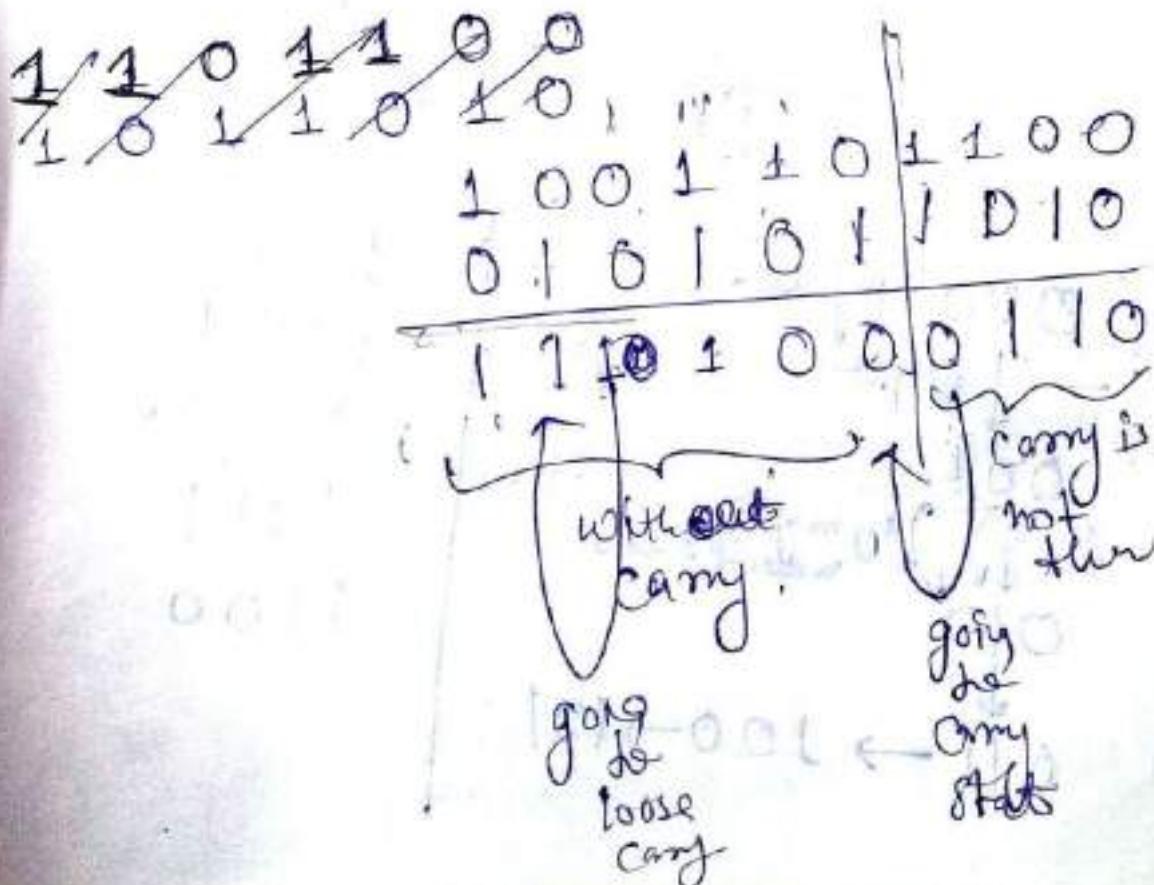
$$\begin{array}{r}
 1000 \\
 - 1001 \\
 \hline
 1010 \\
 - 1011 \\
 \hline
 1100
 \end{array}$$

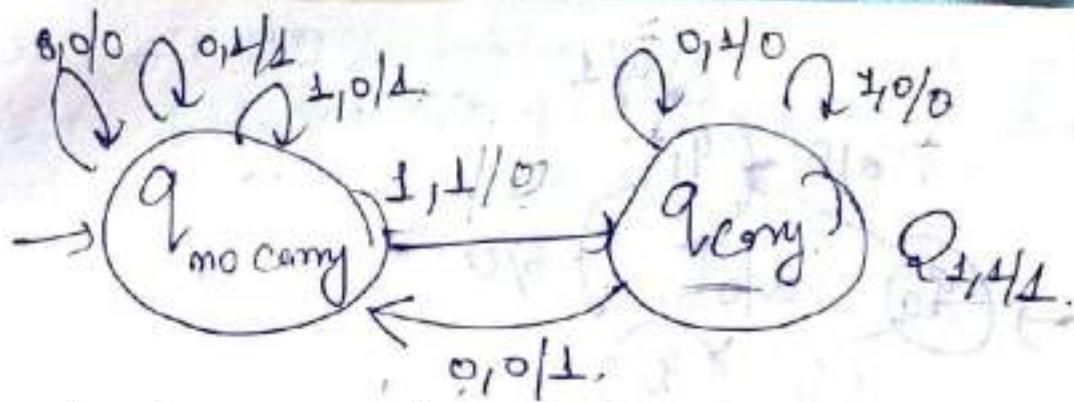
- while reading tree string from LSB
replace 0's by 1's as 0's & 1's
zero as 1 after treat remaining
bits will same.



Q.4 Construct the mealy m/c, that
represents the behavior of binary
adder. (adding

1000





(97)

Q5 construct the mealy m/c, treat take cell string of above bits as input & produce
- 1 as 0/1, 2 as the last
of each last sync

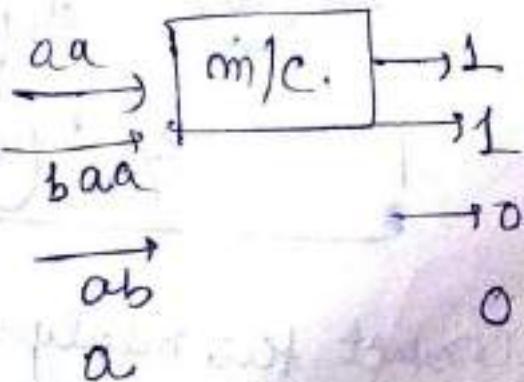
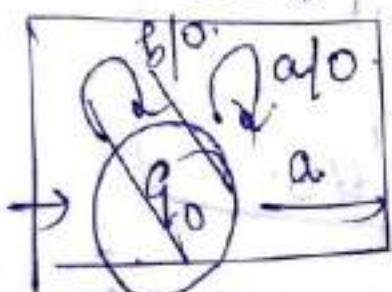


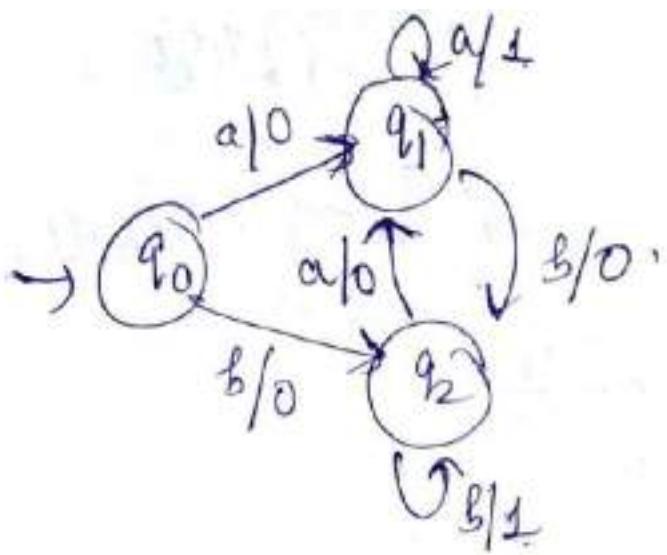
000 \rightarrow symbol
of repetitive
same, also
producing 1
as 0/1 follows
0
0/1

abbb | aq } 1 as 0/1.
abbb | bb }

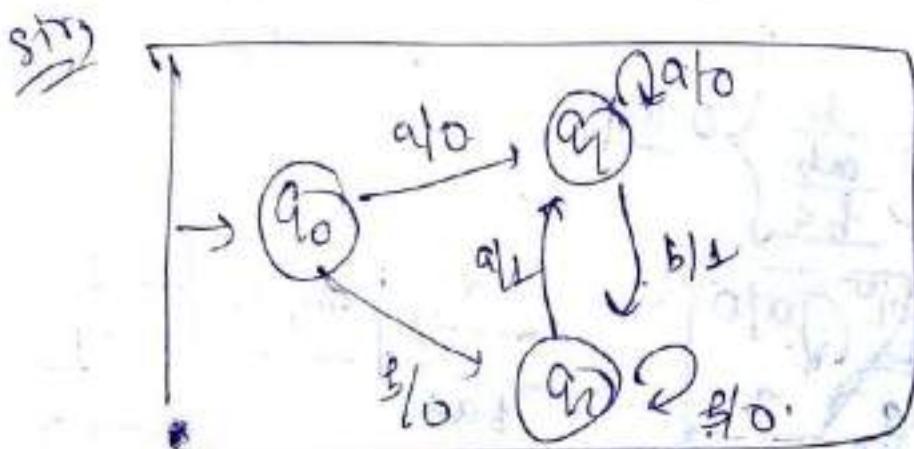
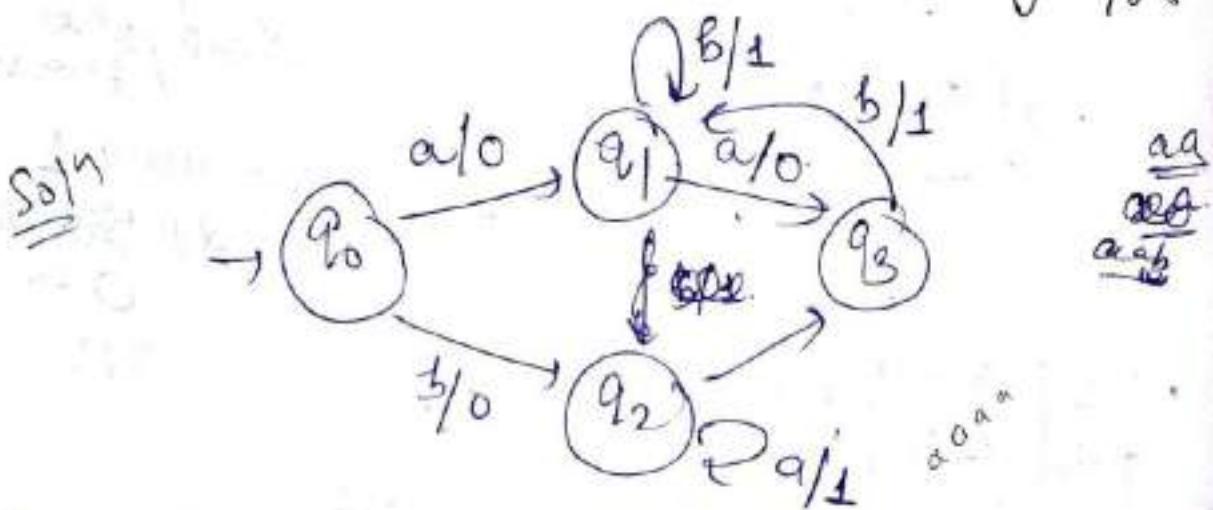
q
b
ab
bs

aa
bb





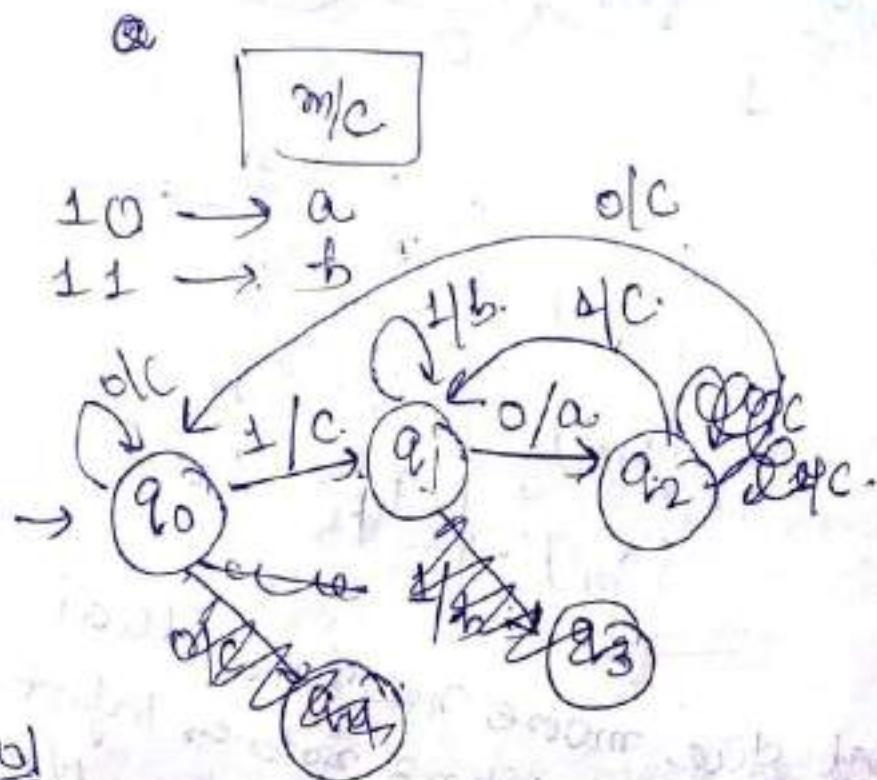
Q.6 The last two symbols are different producing
1 as D/P. other producing 0 as O/P.



Q.7 Convert the mealy m/c, that takes all binary strings as input, and produces 0 as output, the input is

~~Q. 1~~ Ending with 10 & producing b as output \rightarrow input 99
 \rightarrow ending with 11 otherwise producing C as output.

~~Q. 2~~



Q. 3

100

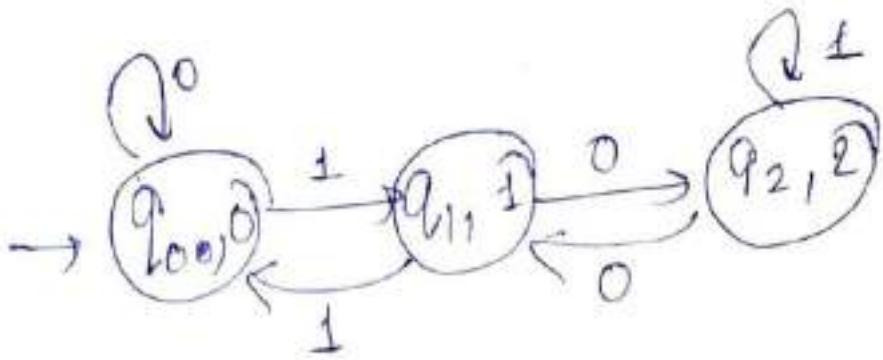
Moore m/c

- Q. Construct the moore m/c, that takes all binary no.'s as input, and produces residue, modulo 3 as o/p, remainders

~~Q. 3~~:

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

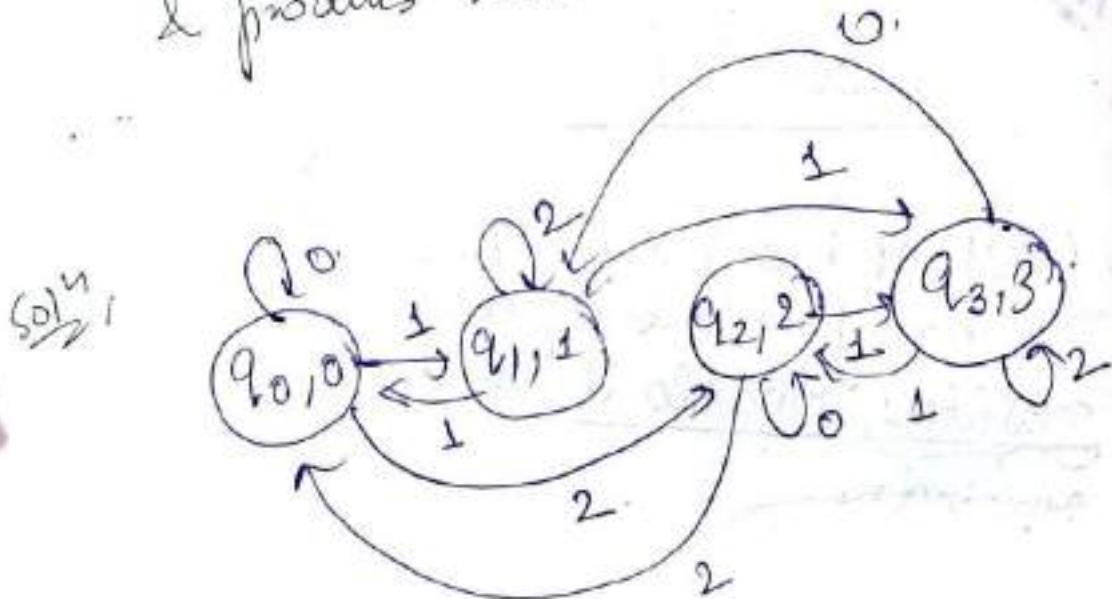
$$O/P \in \{0, 1, 2\}$$



$\begin{array}{l} 000 \\ 001 \\ 0010 \\ 00011 \\ 00100 \\ 00101 \end{array}$

	0	1
$(q_0, 0)$	q_0	q_1
$(q_1, 1)$	q_2	q_0
$(q_2, 2)$	q_1	q_2

Q: Construct the moore machine that takes all base 3 nos as input & produces residue modulo 4 as O/P.



$$\Sigma = \{0, 1, 2\}$$

$$\Delta = \{0, 1, 2\}$$

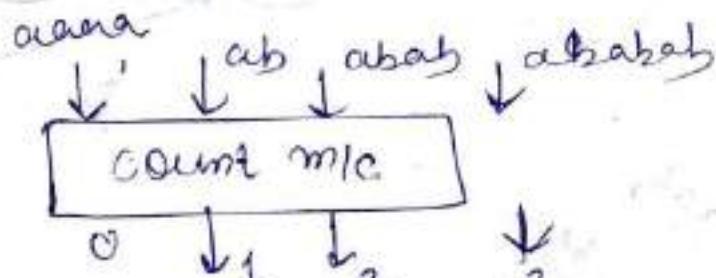
	0	1	2
$(q_0, 0)$	q_0	q_1	q_2
$(q_1, 1)$	q_3	q_0	q_1
$(q_2, 2)$	q_2	q_3	q_0
$(q_3, 3)$	q_1	q_2	q_3

Notes:-

The move that takes all bases m no.'s as input & produces residue modulo n as output requires m states.

- Q. Construct the move machine that takes all strings of ab & ba as input and count total no. of ab string present in the input.

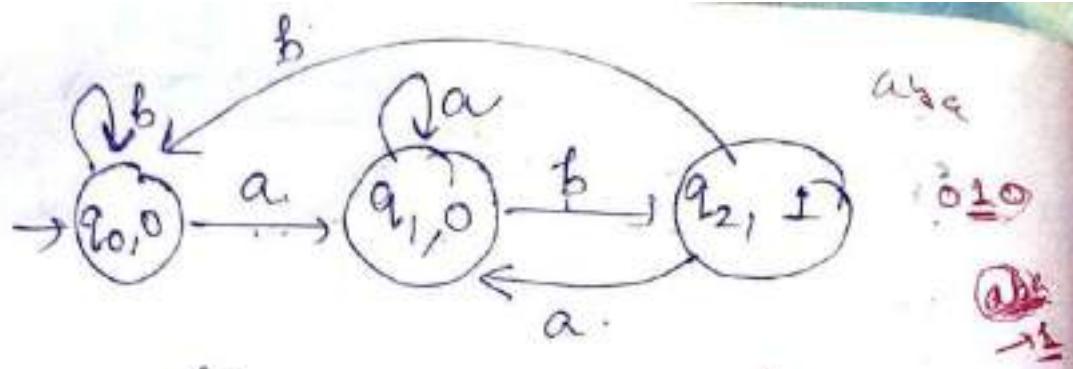
SOL:



1 11 111 —

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

$$\text{output alphabet} = \{0, 1\}$$



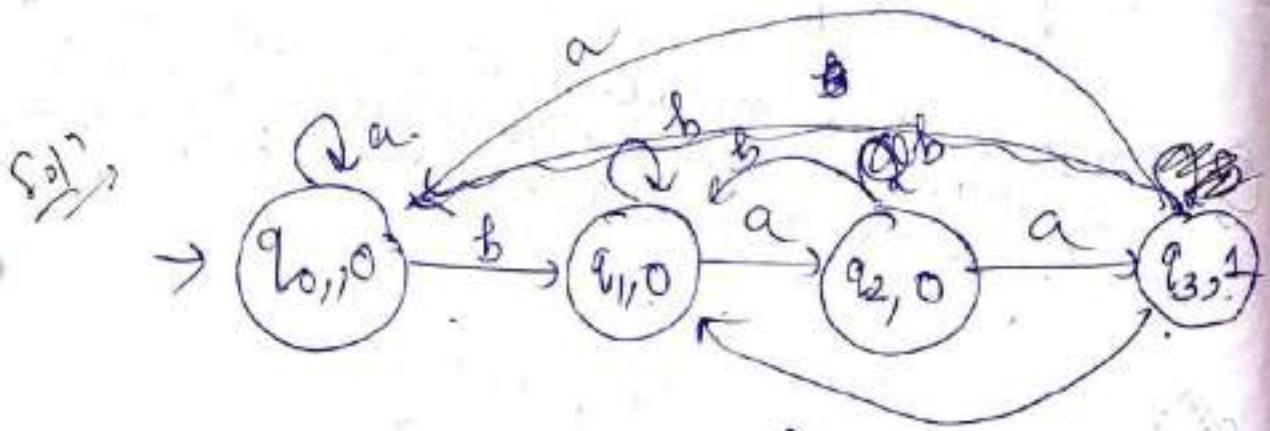
~~baa~~ 0101

001 → ~~010~~
010
0101

0101

1 01P
aab 1
abab 11
abbbbab 11

Q. Construct the moore machine that takes all strings of ab & bb as input and Count total no. of baa strings.

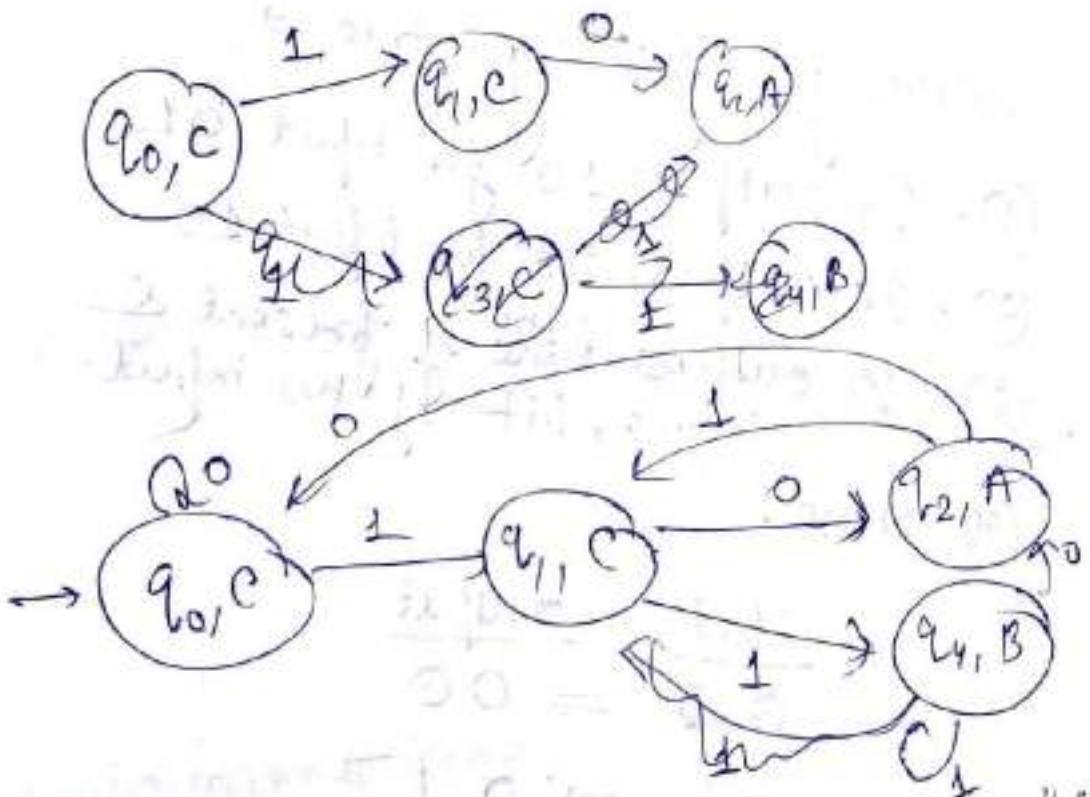


baa
aab ~~baab~~
baab ~~baab~~
baab ~~baab~~

Q. Construct the moore machine that takes all binary strings as input & produces A as output if the input is ending with 10 and producing B as output if input is

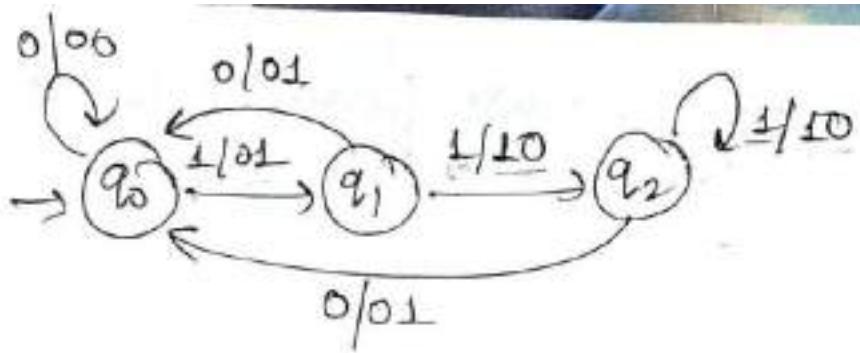
ending with 11, otherwise produce C as output (103)

~~solution~~



* Mealy machine is more comfortable than Moore.

Q. Construct the following finite state machine.



which of the following is true \rightarrow

- (1) It outputs 10 for input 01
- (2) It outputs 00 for input 10
- ~~(3)~~ It outputs sum of present & previous bit by the input
- (4) None,

<u>Input</u>	<u>Output</u>
0 0	\Rightarrow 0 0

0 1 \Rightarrow 0 1

1 0 \Rightarrow 0 1

1 1 \Rightarrow 1 0

Q. Construct the following finite state machine.

\vdots	$x=0$	$x=1$
A	B, 0	D, 0
B	C, 1	B, 1
C	D, 1	B, 0
D	C, 0	B, 1

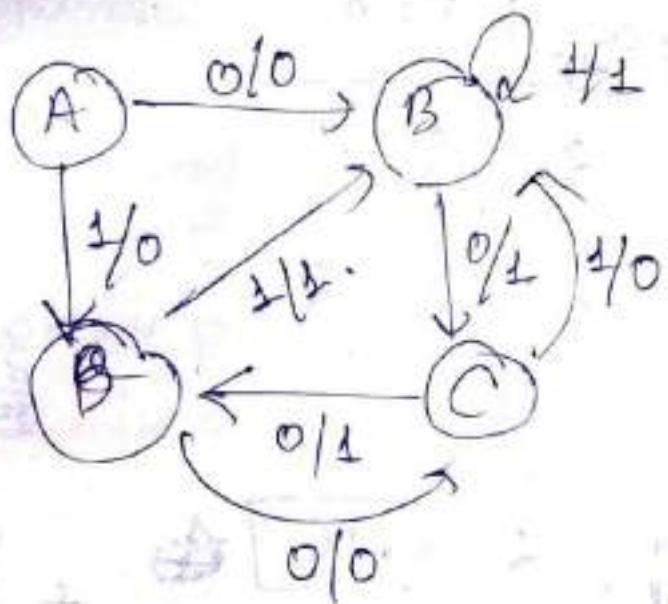
If $x=0$
Initial

state is
unknown

then the
shortest

input
beginning

to reach the state is



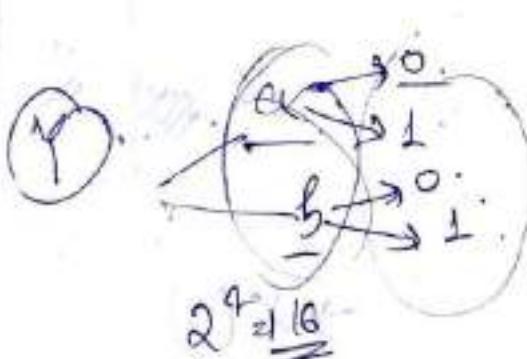
(105)

- (a) 01
- ~~(b)~~ 10
- (c) 101
- (d) 110

Q. How many max. no. of mealy machines are possible with 2 states X and Y over the input alphabet $\{a, b\}$
 $D = \{0, 1\}$

~~Q17~~

X



$$2^4 = 16$$

~~(1) X~~

~~(1) Y (0) Y~~

~~(1) Y (0) Y~~

	$x = a$	$x = b$
x	2 2	2 2
y	2 2	2 2

$$\frac{16}{32}$$

when x is initial, $2^8 = \boxed{256}$

when y is initial, $\boxed{256} + \boxed{512}$

Q. How many max. no. of move machines
are possible, with 2 states X &
 Y $\Sigma = \{a, b\}$ $\Delta = \{0, 1\}$

	ilp _a	ilp _b	0/p
$x, (\cancel{y})$	2	2	2
$y, (\cancel{x})$	2	2	2

$$\Rightarrow (2)^6 \rightarrow \boxed{64}$$

$\rightarrow X(0)$

$\rightarrow X(1)$

$Y(0) \cdot Y(1)$

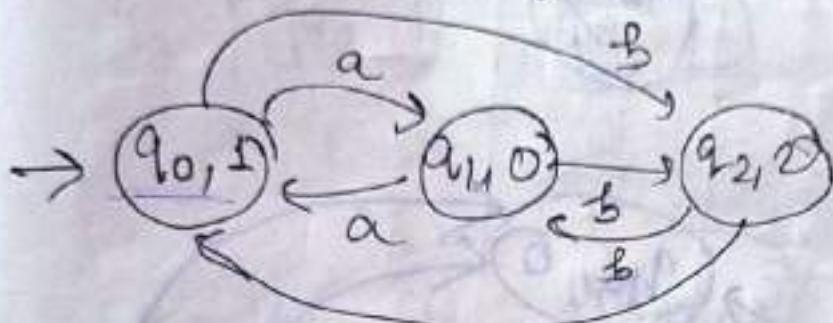
$Y(0) Y(1)$

Y is initial. $\Rightarrow 64$

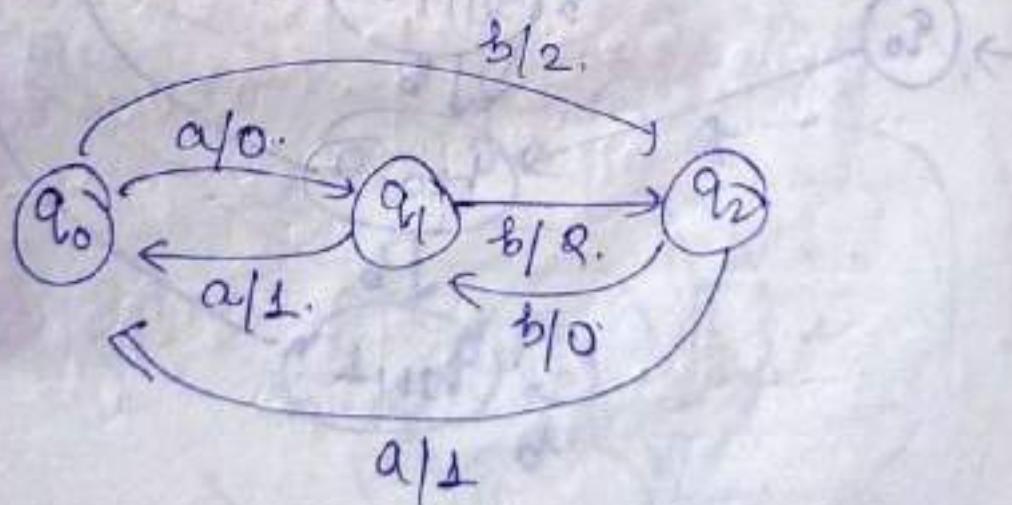
(107)

$$\text{Total} = 128 \quad \underline{\underline{A}}$$

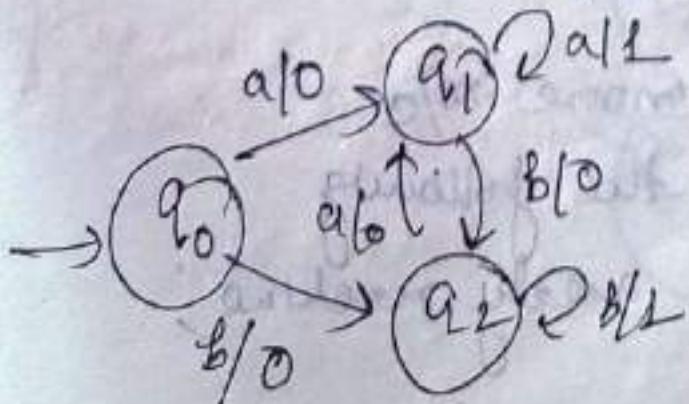
- Q. Construct tree mealy m/c, equivalent to following moore m/c.



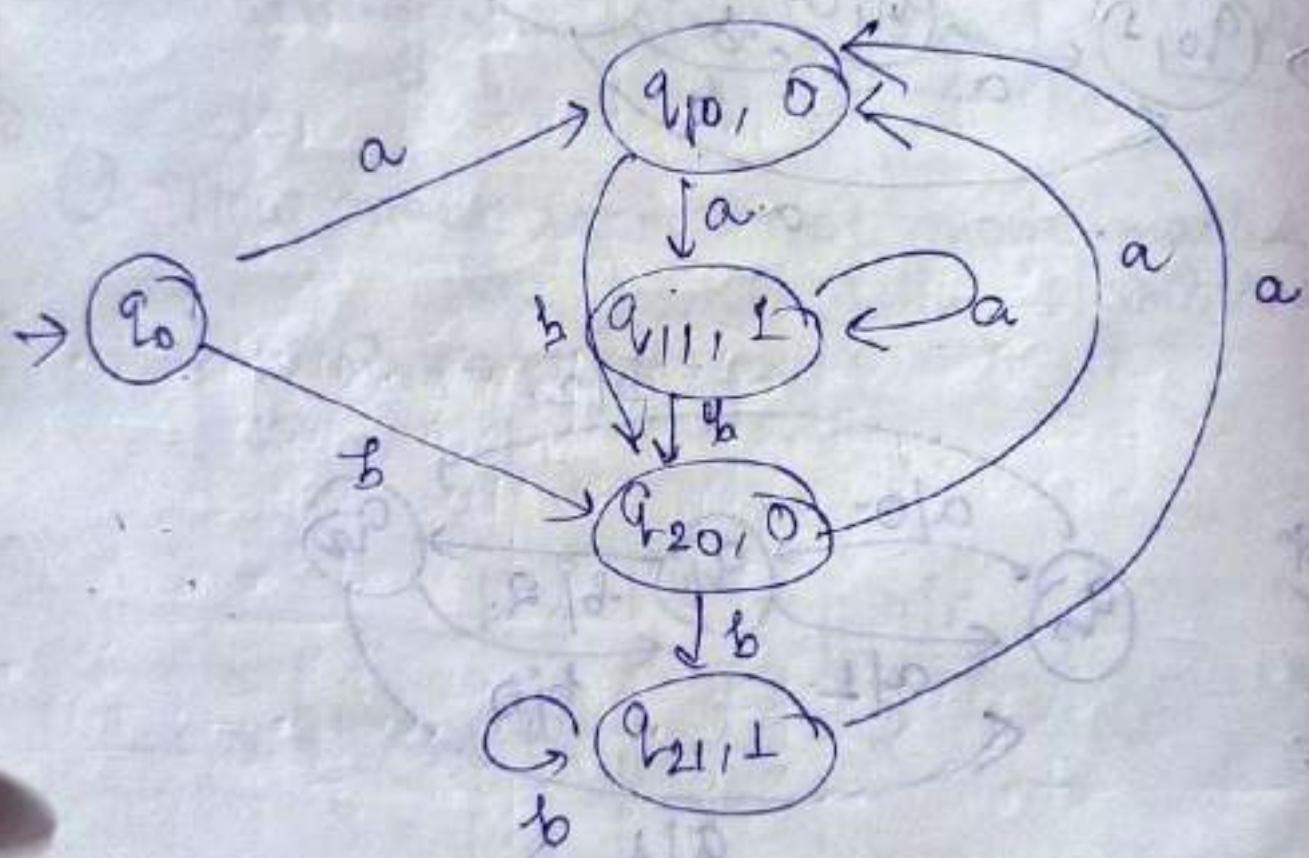
Sol^M



Q.



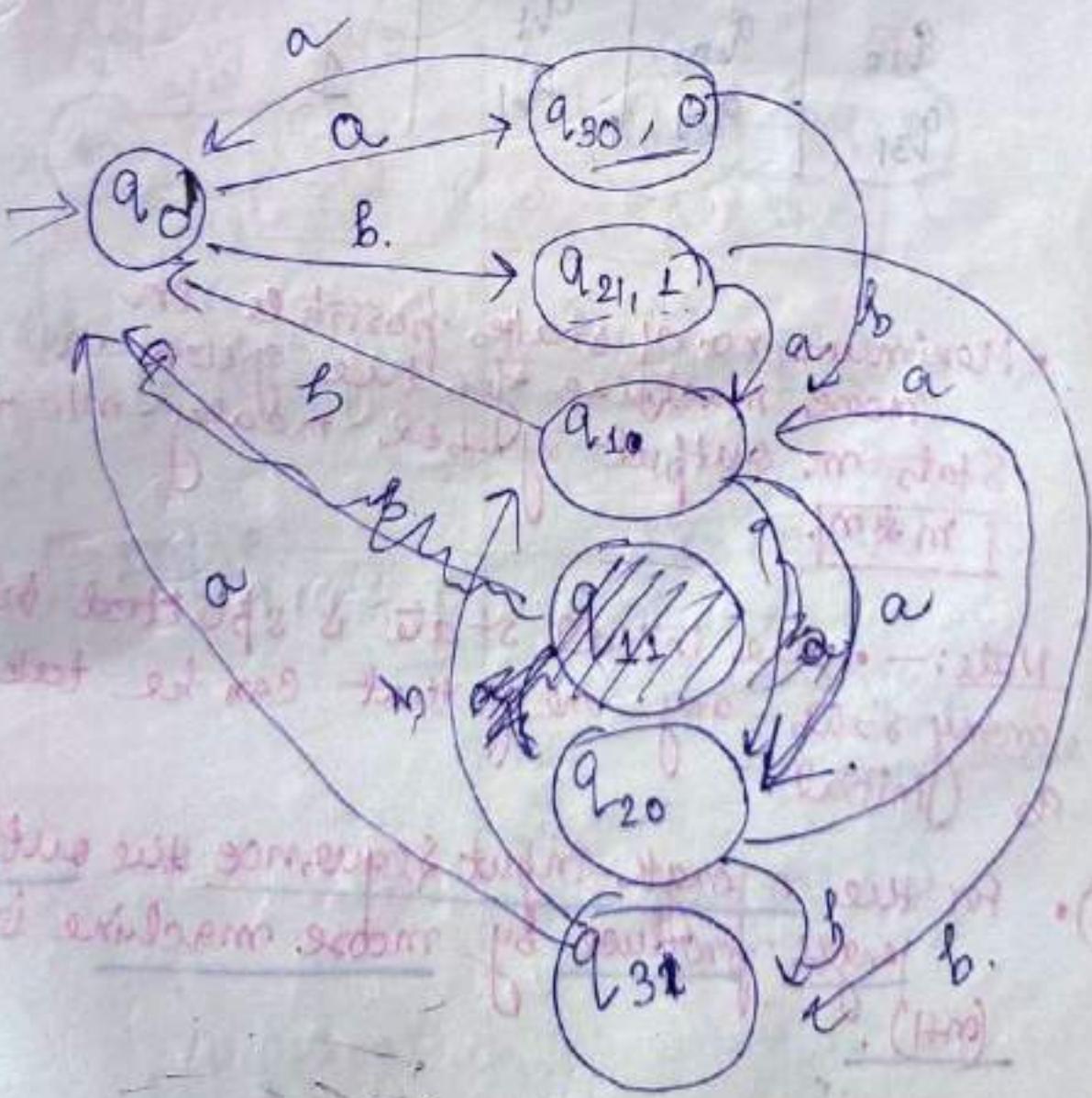
Another tree
moore m/c,
equivalent to
mealy m/c.



Q. Construct the moore m/c
for the following
mealy machine

	$i = a$		$i = b$	
	NS	O/P	NS	O/P
q_0	q_3	0	q_2	1
q_1	q_2	0	q_0	1
q_2	q_1	1	q_3	1
q_3	q_0	1	q_1	1

(109)



	$i=a$ N.S	$i=b$ N.S	O/P
q_0	q_{30}	q_{21}	1
q_1	q_{20}	q_0	1
q_{20}	q_1	q_{31}	0
q_{21}	q_1	q_{31}	1
q_{30}	q_0	q_1	0
q_{31}	q_0	q_1	1

- Maximum no. of states possible in Moore machine for the given n States m output symbol namely $m \times m$

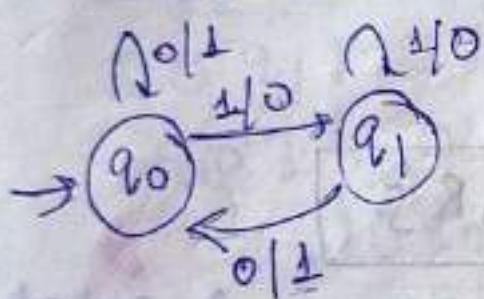
- Note: — • The initial state is splitted into many states any one of that can be taken as Initial.

- ① • For the n length input sequence the output length produced by moore machine is $(n+1)$.

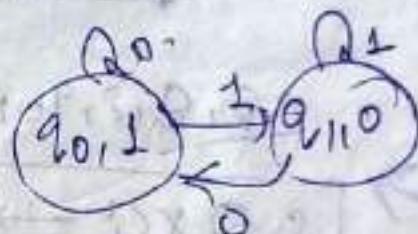
② For the n length input sequence the
output sequence produced by mealy m/c is m only. (110)

③ To construct the equivalent mealy m/c for
 the given moore m/c (or) moore m/c for
 the given mealy m/c, see initial o/p
 produce by moore m/c is neglected.
 Then only equivalence is possible.

Mealy m/c



Moore m/c



① i/p $\boxed{101}(S)$

o/p $\boxed{010}(2)$

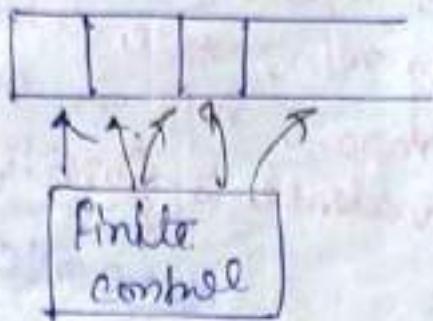
② i/p $\boxed{101}(S)$

o/p $\boxed{1010}(10)$

if we want
 equivalent
 moore m/c
 then

$\boxed{010}(2)$

Q-DFA or Two-way F.A.



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

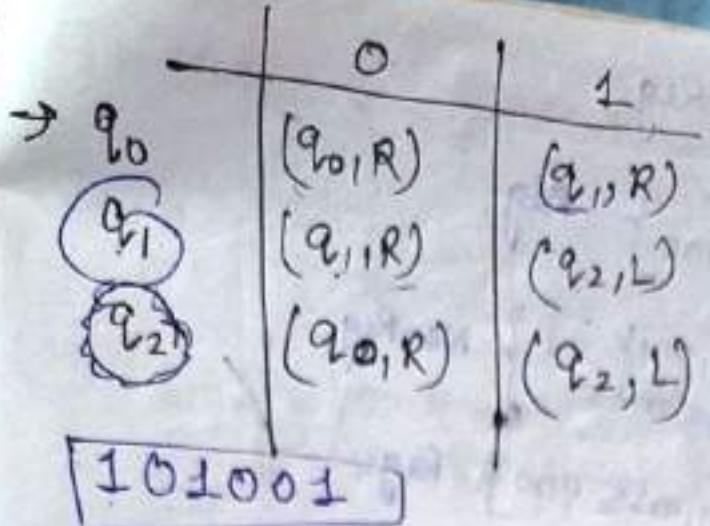
$$\delta: Q \times \Sigma \rightarrow Q \times \{L, R\}$$

Q-DFA is a DFA having turn around capability in the formal definition of Q-DFA

- The Expressive power of Q-DFA is same as DFA.

Acceptance by final state →

- Q. Construct the following Q-DFA,



$Q_0 \xrightarrow{1} 101001$

$1 \xrightarrow{0} Q_1 1001$

~~10~~ $\xrightarrow{0} Q_2 01001$

$10 \xrightarrow{Q_0} 1001$

$101 \xrightarrow{Q_1} 001$

$1010 \xrightarrow{Q_1} 01$

$10100 \xrightarrow{Q_1} 1$

$101000 \xrightarrow{Q_2} 01$

$1010000 \xrightarrow{Q_0} 1$

$1010001 \xrightarrow{Q_1} -$

Accepted

Regular Language :-

Q. Which of the following languages are regular
 (or) which are not regular?

- ①. $L = \{a^n b^n | n \leq 10\}$. Reg.
- ②. $L = \{a^n b^n c^m | n > m, n, m \leq 100\}$. Reg.
- ③. $L = \{a^n b^m c^k | \begin{cases} n, m, k \geq 1 \\ n = m + k \end{cases}\}$. Non-Reg.
- ④. $L = \{a^n b^n c^m | n, m \leq 200\}$. Reg.
- ⑤. $L = \{a^n b^{m+n} c^m | m, n \geq 1\}$. Non-Reg.
- ⑥. $L = \{a^n b^{m+n} | m, n \geq 1\}$. Non-Reg.
- ⑦. $L = \{a^{n^2} b^m | m = n^2\}$. Non-Reg.
- ⑧. $L = \{a^{2m} b^{3m} | n, m \geq 1\}$. Non-Reg.
- ⑨. $L = \{a^n b^m | n+m=100\}$. Reg.
- ⑩. $L = \{a^n b^m | n-m=5\}$. Non-Reg.
- ⑪. $L = \{a^n b^m | n/m=4\}$. Non-Reg.
- ⑫. $L = \{a^n b^m | (n+m) \bmod 3 = 0\}$. Reg.
- ⑬. $L = \{a^n b^m | n > M \text{ or } n < m\}$. Non-Reg.
- ⑭. $L = \{a^n b^m | n \geq m \text{ and } n \leq m^2\}$. Non-Reg.

- JAN
- (17). $L = \{a^m b^{n^3} \mid n, m \geq 1\}$. ^{non-} Reg. (1/2)
- (18). $L = \{x \mid x \in \{a, b\}^*$ } ^{Reg.}
 $m_a(x) \bmod 7 = 0\}$
- (19). $L = \{x \mid x \in \{a, b\}^* \}$ Non Reg. ^{of a too}
 $m_a(x) > m_b(x)\}$
- (20). $L = \{x a y \mid x, y \in \{0, 1\}^*\}$. Reg. (String concatenation)
- (21). $L = \{w w^R \mid w \in a^*\}$. Reg.
- (22). $L = \{w a w^R \mid w \in b^*\}$. Reg. NM-Reg. ^{ab ab}
^{babb}
- (23). $L = \{w x w^R \mid w \in \{0, 1\}^*\}$. Reg. Non-Reg.
- (24). $L = \{w x w^R \mid w, x \in \{0, 1\}^*\}$. Reg.
- (25). $L = \{w x w \mid w \in \{a, b\}^*\}$. Non-Reg. ^{aa ab bb}
- (26). $L = \{w x w \mid w \in \{a, b\}^*\}$. Reg. ^{ab bab}
- (27). $L = \{w \mid w \in \{a, b\}^*, m_a(w) \bmod 3 = 0 \text{ and } m_b(w) \bmod 5 = 0\}$. Reg.
- (28). $L = \{a^{2^n} \mid n \geq 1\}$. Non-Reg.
- (29). $L = \{a^{n!} \mid n \geq 1\}$. Non-Reg.
- (30). $L = \{a^k \mid k \text{ is prime no.}\}$. Non-Reg.
- (31). $L = \{a^n \mid n \text{ is odd number}\}$. Reg.

- (32). $L = \{ 1^{2n+1} \mid n \geq 1 \}$. Reg.
- (33). $L = \{ 1, 2, 4, 8, \dots, 2^n \}$
- all these
no.s in
unary
- Non-Reg.
- (34). Set of all even length palindrome string of English language. Non-Reg.
- (35). Set of all odd length palindrome string of Hindi language. Non-Reg.
- (36). Set of all palindromes of Japanese language. Non-Reg.
- (37). Set of all odd length palindrome string of a language formed over one symbol x. Reg.
- (38). Set of all balanced parenthesis $(())$. Non-Reg.
- (39). Set of all lexical errors produced by compiler. Reg.

(40). Set of all ~~synthetic~~ error produced by compiler. (113)

(41). Set of all Syntax errors produced by compiler. Non-Reg

(42). Total property of Amitabh Bachan family including ~~including~~ a bachelon. Reg.

(43). Total amount of Indian black money in ~~secret~~ Banks. Reg.

(44). C language is ~~Reg~~ language. Non-Reg

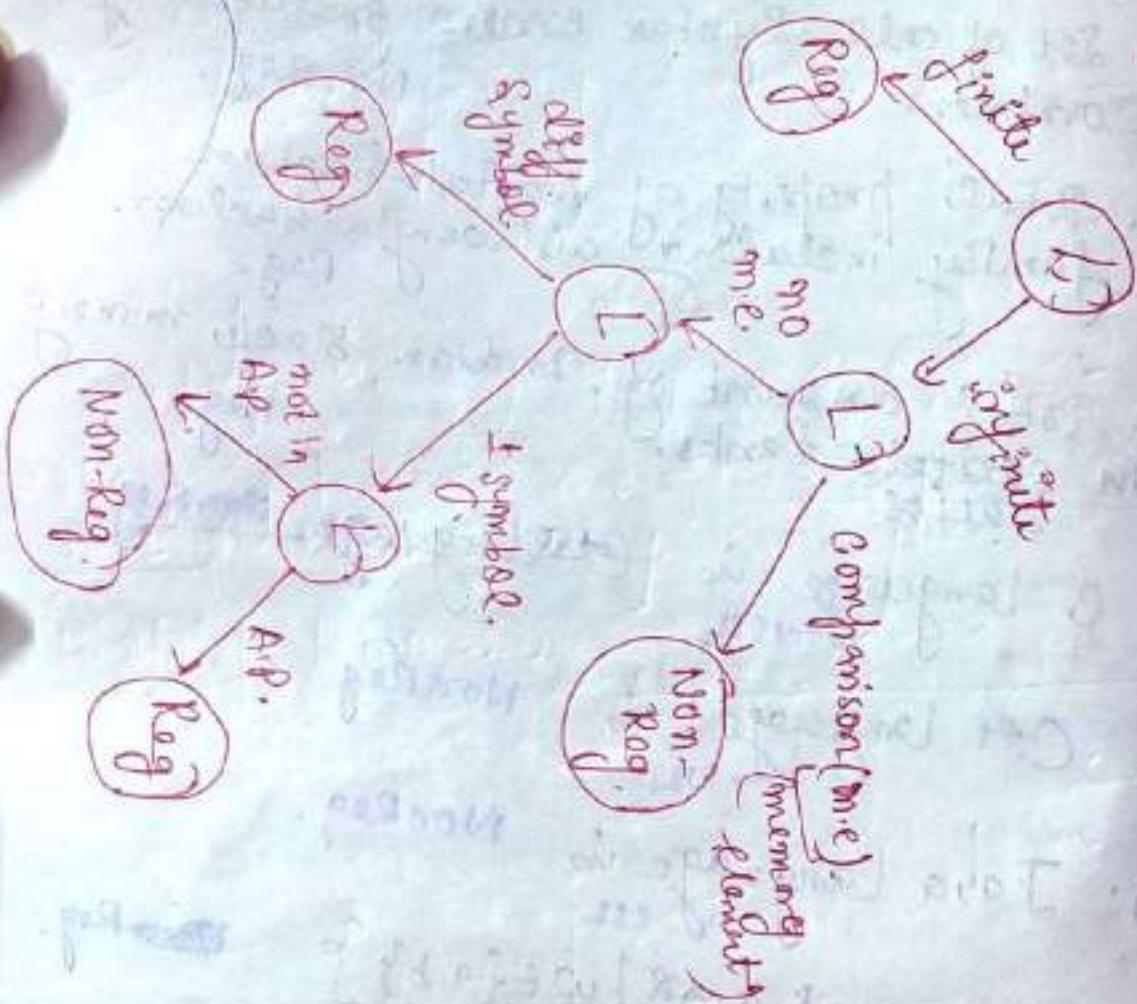
(45). C++ language is ~~Reg~~ Non-Reg

(46). Java language is ~~Reg~~ Non-Reg.

(47). $L = \{w w^R w w^R | w \in \{a, b\}\}$. ~~Reg~~ Non-Reg.

a a a is special string fruit.
for ~~apple~~ apple

300 letters are special symbol of
ternary coding & most used in
religion



- Every finite language is Regular but all Regular Languages not necessarily be finite.
- Palindrome languages over one symbol are regular more than 1 symbol are non-regular.

Q. Which of the following language is regular,

(114)

a) $L = \{waw^R \mid w \in \{a\}^*\}$ {Non Reg}

b) $L = \{0, 101, 11011, 1110111, \dots\}$ {Non Reg}

c) $L = \{anbmck \mid \frac{n+m+k}{n,m,k} \in \mathbb{N}\}$ {Non Reg}

d) None.

$\rightarrow L = \{a, aa, aaaa, \dots\}$

Q. Which of the following language is not regular

a) $L = \{w \mid w \in \{a, b\}^*, n_a(w) \bmod 5 = 0 \text{ and } n_b(w) \bmod 3 = 0\}$

b) $L = \{w \mid w \in \{a, b, c\}^*, n_a(w) \bmod 3 = 0 \text{ and } n_b(w) \bmod 5 = 0 \text{ and } n_c(w) \bmod 7 = 0\}$

c) $L = \{a^k \mid k \text{ is even number}\}$

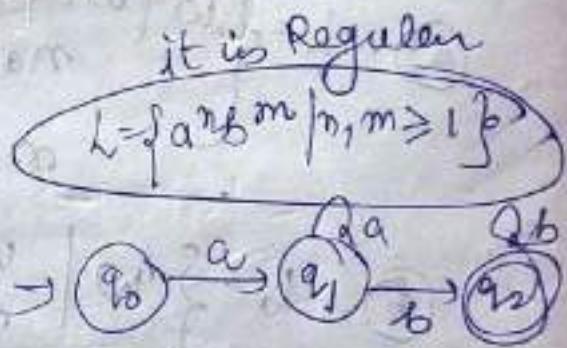
d) None.

~~no question~~ ~~call~~
but in IIT intended
it's question.

~~Plan~~

Pumping lemma :-

- Pumping lemma is a theorem used to prove non-regular language as non-regular.
- Pumping lemma uses proof by contradiction problem solving technique.
- Pumping lemma uses pigeon-hole principle



ab
aab
ahb
aahb

loop string
u | v | w
↓ pump it
u | v | w
 $\frac{i=1}{i=2}$ aab. ✓
aaab. ✓

Step ①: To prove a language L is non-regular assume it as regular.

Assume L is Regular

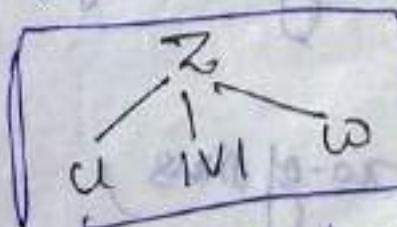
Step ②: If L is regular there exist F.A. for that language assume no. of states will be n . In that language.

FA $\Rightarrow n$

Step ③: Select some string z from L such that its length is greater than equal to no. of strings. According to pigeon-hole principle whenever a string length is greater than equal to no. of states then there exist a loop cycle in automata.

| z | $\geq n$

Step ④: Divide the z string into three parts u, v, w where v is the loop string.



Step ⑤: L is regular for all $i \geq 1$.

uv^iw should belongs to L .
For atleast one i value uv^iw not
belongs to L then L is non-regular.

• \Rightarrow $uv^iw \forall i \geq 1 \in L \Rightarrow$ Regular

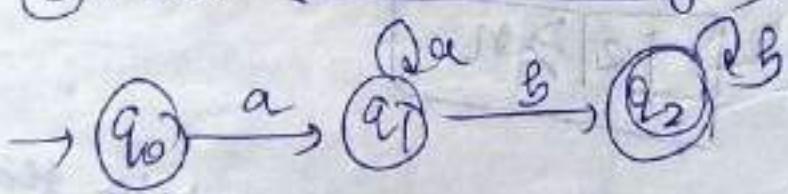
$av^iw \forall i \geq 1 \notin L \Rightarrow$ Non-Regular

• Pumping Lemma is strong for proving non-Reg
as Non-Reg & weak for proving Reg

Q. Prove that the language $L = \{a^n b^n | n \geq 1\}$
is non-regular using pumping lemma.

Sol: (1) L Regular

(2). Let there is a FA of L .



(3). $aabb$

(4). $a|abbb$. \rightarrow any one loop we consider here.
 $u \quad v \quad w$

(5). pump it any no. of times

E-1

aaab.

(116)

i-2 aaab. \notin not belong to
the language

$\notin L$

so, [not Regular]

Note :-

- ① If there exist a finite automata f & there exist a language for that F.A. and all strings length less than the no. of states of the automata then that language is finite language only.
- ② There exist a F.A f & language accept by it is L if there exist atleast one string whose length is greater than or equal to no. of states of automata then that language is infinite language only.

~~V.V.V.V.V~~

Closure properties of Regular language:-

$2+3 = 5$	$4/2 = 2$
$3+4 = 7$	$3/2 = 1.5$

(+) → closed (/) → not closed

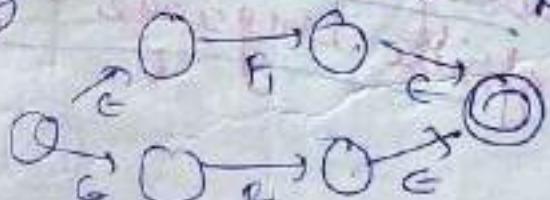
result int result not only int

① Union of :

$$(R_1 L_1) \cup (R_2 L_2) = R_1 L_2$$

↓ ↓ ↓

$R_1 A$ $R_2 A$ If we have RA,
for this so, Reg.



This concept is not used in this field.

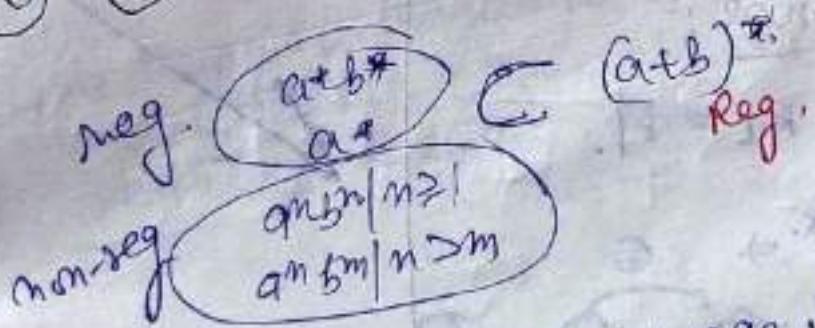
Let L_1 & L_2 are two regular languages,
 R_1 & R_2 are finite automata for L_1 &
 L_2 respectively & then the union of L_1
& L_2 is always regular. because
 $R_1 A$ exist as follows

because

$$\{R \cdot L_1\} \cup \{R \cdot L_2\} = \{R \cdot L_3\}$$

Hence, Regular language closed under union operation.

② Subset operation:

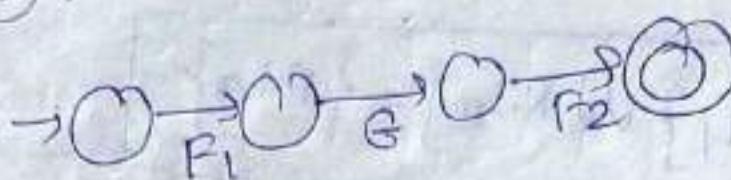


The subset of Reg. language is may or may not regular hence Regular languages not closed under subset operation.

(developed on the basis of J.P. Monk's
 approach through out the session)

③ Concatenation of:

$$\begin{aligned} L_1 &= a^* b^* \\ L_2 &= d^* e^* \end{aligned}$$

 F_1 F_2 $a^n L_1$ $b^n L_2$ $L_1 L_2 \Rightarrow$ 

$L_1 L_2$ is also regular

(not complete).

(Reg.).

(any no of

$a's$ &

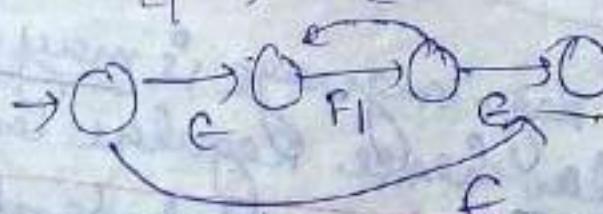
$b's$)

Let L_1 & L_2 are two regular language, F_1 & F_2 are finite automata for L_1 & L_2 are respectively given the concatenation of L_1 & L_2 are always regular because concatenation of R.L exists as follows \rightarrow

Kleene closure:

$$L_1 = R.L = F_1.$$

$$L_1^* \Rightarrow \epsilon$$



Note— $L_1 = a^n$

$L_2 = b^n$

then $L_1 L_2$ is also Regular.
(because the resultant language)

\Leftrightarrow no. of a's followed by no. of b's only).

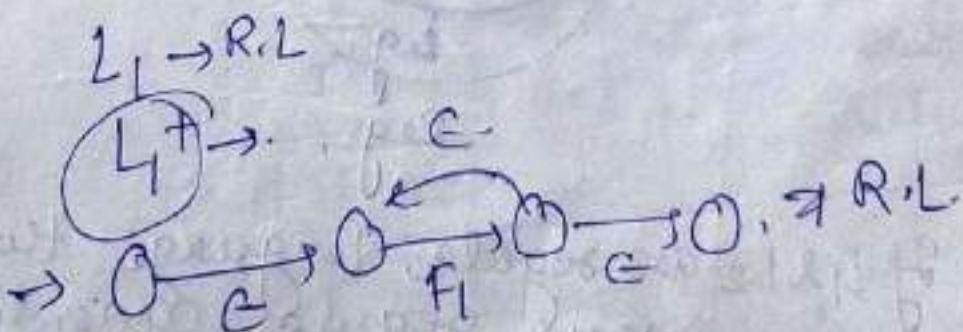
(118)

④ Kleene closure:—

L_1 is regular language F_1 is FA for L_1 in the
Kleene closure of L_1 is always regular
because there exist FA as follows \rightarrow

⑤ Positive closure:—

L_1 is regular language F_1 is FA for L_1 in the
positive closure of L_1 is always
regular because there exist FA as
follows \rightarrow



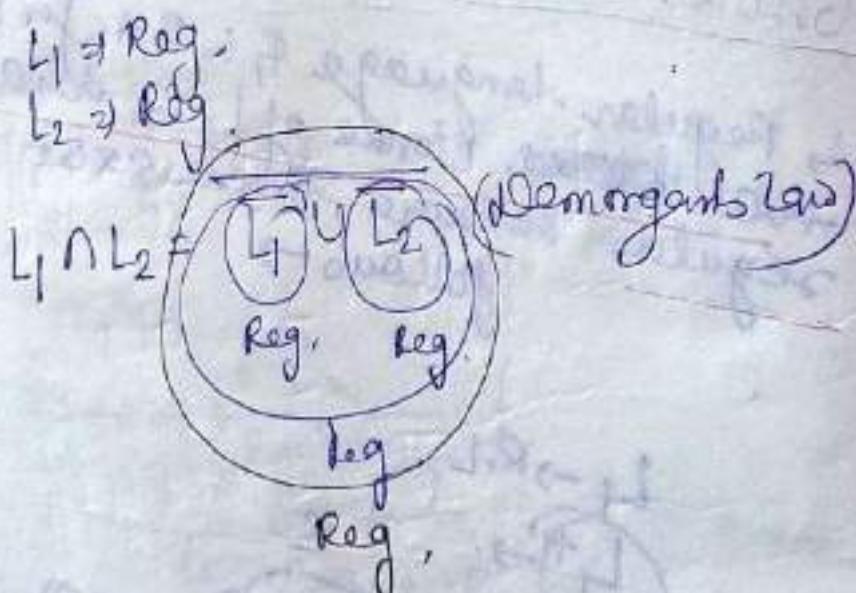
⑥. ~~SM~~ Complement of P.L.—

Complement of a Reg. language is always Reg. because we can construct complement P.A. by interchanging final as non final, non final as final.

$$L_1 \rightarrow R.L \rightarrow F.A. \checkmark \quad \text{P}$$

$$\Sigma^* - L \Rightarrow F.A \vdash R.L$$

⑦. ~~SM~~ Intersection of P.L.—

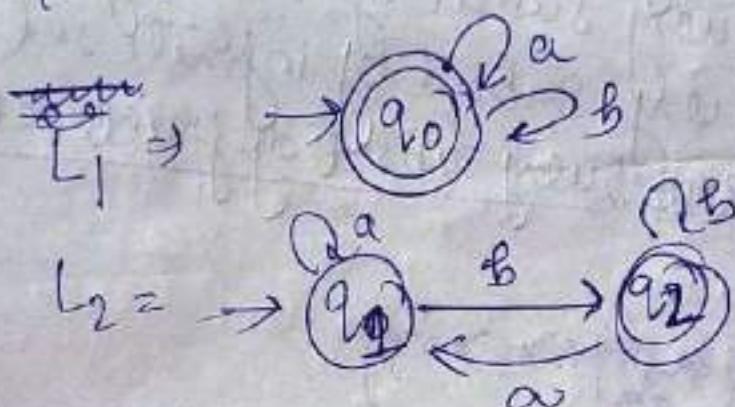


If L_1 & L_2 are regular languages then $L_1 \cap L_2$ is always regular according to De Morgan's law.

(Union & complement are closed under Reg. languages hence intersection is closed.) (119)

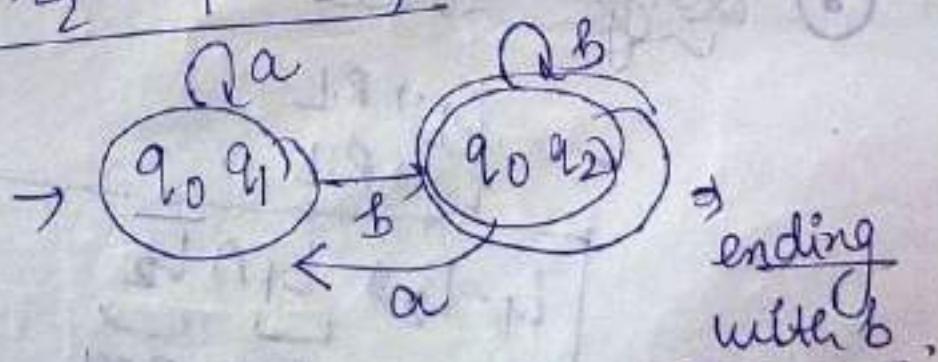
Q. Construct $L_1 \cap L_2$ DFA where $L_1 = (a+b)^*$
 $L_2 = (a+ab)^*b$.

Soln: $L_1 \cap L_2 \Rightarrow (\overline{L_1} \cup \overline{L_2})^*$



$\overline{L_1} \cup \overline{L_2}$

$L_1 \cap L_2 = F_1 \times F_2$



ending with b.

subset \cap subset \Rightarrow subset
 \downarrow \downarrow \downarrow
 $(a+b)^*$ $(a+b)^*b$ $(a+b)^*b$

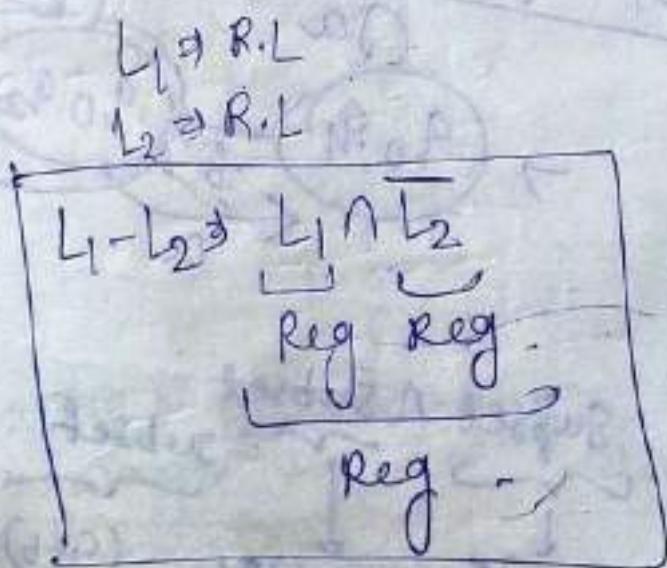
Hence, Regular language is closed under Intersection operation.

Q. Which of the following is true, every subset of Reg.

- ①. Every subset of Reg set is Reg.
- ②. Every subset of non-Reg set is Reg.
- ③. Every subset of infinite set is Reg.
- ④. Every subset of finite set is Reg.

(ans) C $\{L = \{a^m b^n c^m \mid n, m \geq 0\}\}$

⑧. Difference O/P :-



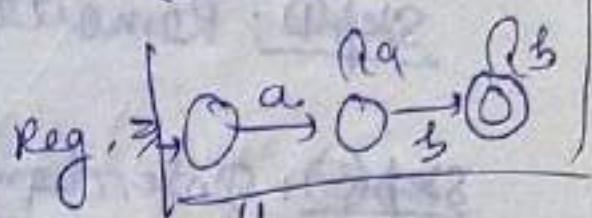
(120)

L_1 & L_2 are regular languages, $L_1 - L_2$ is always
Regular.

⑨. Reversal of :-

Here we reverse the lang

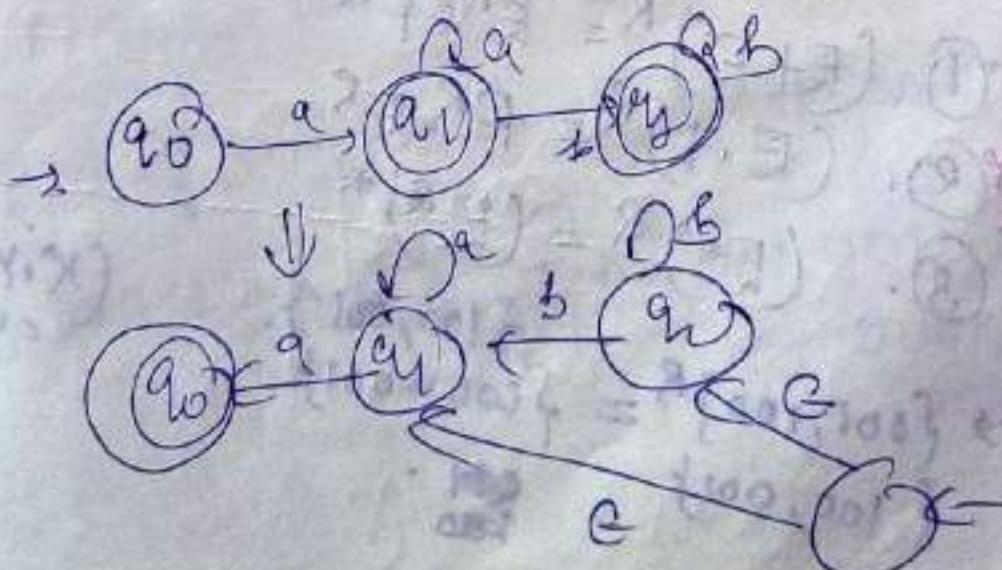
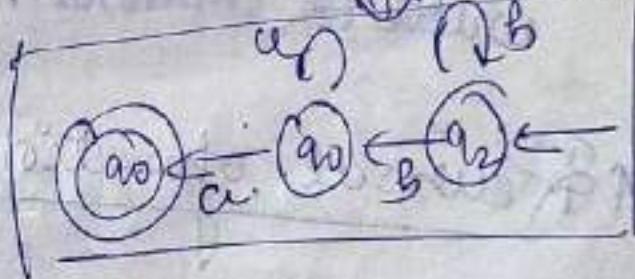
$$L = \{a^n b^m \mid n, m \geq 1\}$$



$$L_2 = \{b^m a^n \mid n, m \geq 1\}$$

Reg. \rightarrow mirror image
method

(1) Initial to final
and final to initial
II. change the transition



PCH

The Reverse of a Reg. language is always
Reg. because Reversal automata
constructed as follow →
(given above)

Step ①: Remove the dead state for given automata.

Step ②: Interchange initial as final & final as initial. & reverse edge direction.

Step ③: If the given automata contains multiple final state then there is a chance of getting multiple initial states in rev. automata & make them into single initial by attaching new initial state with ϵ transition.

⑩ Reversal operation on Regular expressions

$$①. (E+F)^R = E^R + F^R$$

$$②. (E \cdot F)^R = F^R \cdot E^R$$

$$③. (E^*)^R = (E^R)^*$$

$$\rightarrow \{001, 100\}^R = \{100, 001\}$$

$$\{100, 001\}$$

$$\begin{matrix} 001 \\ 100 \end{matrix}$$

$$(X \cdot Y)^R = Y^R \cdot X^R$$

(121)

Hence Reg. language are closed under Reversal operation.

⑩ Init op:—

CLIQUE

Set of all prefixes of all strings of ~~given~~
~~language~~ is known as init of that
language.

ex: $L = \{baa\}$. ex: $L = \{ab, ba\}$

$\text{Init}(L) = \{\epsilon, b, ba\}$ $\text{Init}(L) = \{\epsilon, a, ab, b, ba\}$

$\{b, ba\}$ $\{a, ab, b, ba\}$

$\{\epsilon, b, ba\}$

ex: $L = \{0^n 1^n | n \geq 1\} = \{01, 0011, 000111, \dots\}$

$\text{Init}(L) = \{\epsilon, 0, 01, 00, 001, 0011, 000, 0001, 00011, 000111, \dots\}$

$$= \{ 0^1, 001, 000111, \dots \}$$

ϵ	ϵ	ϵ	
0	00	000	
01	001	0001	
0011	00011	000111	

$$\text{Init}(L) = \{ 0^n 1^m \mid n \geq m \}.$$

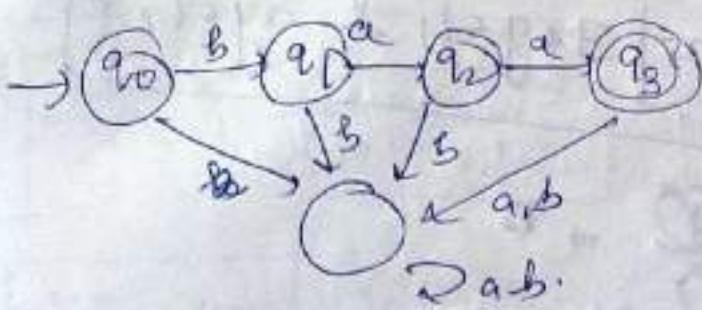
ex: $L = \{ n \in \{0,1\}^* \mid n_0(n) = n_1(n) \}$

$$\text{Init}(L) = \{ \epsilon, 01, 0011, 000111, \dots \}$$

ϵ	ϵ	ϵ	ϵ	ϵ
0	00	000	0001	00011
01	001	0001	00011	000111
0011	00011	000111		

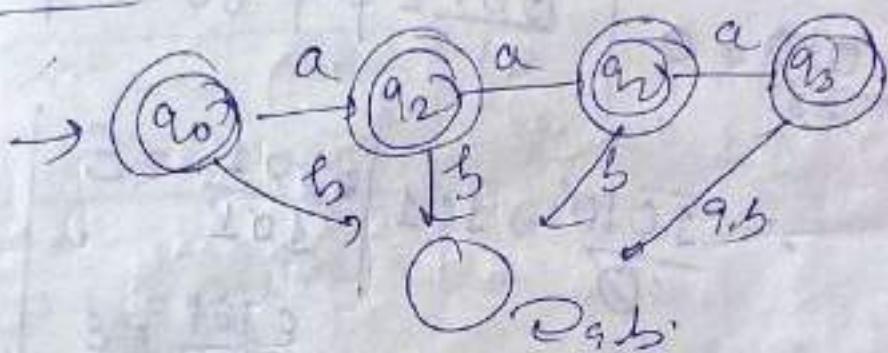
$$= \boxed{(0+1)^*}$$

ex: $L = \{baa\} \rightarrow \text{reg.}$



(122)

$\text{Init}(L) := \{\epsilon, ba, ba, baab\} \rightarrow \text{reg}$



→ make all states as final except dead state.

Init of a regular language is always Regular because there exist P.A. for Init language by making all states as final except dead state hence regular language are closed under Init operation.

Q. Quotient O/P:

$$L_1 / L_2 = \{x \mid \exists x \in L_1 \text{ & } \exists y \in L_2\}$$

$$\frac{L_1}{L_2} = \frac{x \cancel{(y)}}{\cancel{(y)}} \Rightarrow x$$

Right
Cancellation

$$\frac{010}{\cancel{10}} \Rightarrow 0 \quad \left| \frac{001}{11} = \phi \right| \quad \left| \frac{10}{00} = \phi \right|$$

$$\frac{1010}{\cancel{1010}} \Rightarrow 101 \quad \left| \frac{\cancel{1010}}{\cancel{1010}} = \epsilon \right|$$

$$\frac{\epsilon}{\cancel{1010}} = \epsilon$$

$$\frac{101}{\epsilon} = 0 \quad \left\{ \frac{101\epsilon}{\epsilon} = \cancel{101}^0 \right\}$$

$$\frac{\Sigma^*}{\Sigma^{**}} = \frac{\{e, 0, 1, 00, 01, 10, 11, \dots\}}{\{\underline{e}, 0, 1, 00, 01, 10, \dots\}} = \Sigma^*$$

$$\boxed{\frac{\Sigma^*}{\Sigma^{**}} = \Sigma^*} \quad e/e, 0/e, 1/e, \dots$$

$$\boxed{\frac{\Sigma^+}{\Sigma^+} = \Sigma^*} = \frac{\{0, 1, 00, 11, 01, 10, \dots\}}{\{0, 1, 00, 01, 10, 11, \dots\}}$$

$$\left[\frac{\Sigma^+}{\Sigma^-} = \frac{\Sigma^*}{\Sigma^+} = \Sigma^* \right]$$

$$\left[\frac{O^*}{O} = O^* \right] = \frac{\{ \epsilon, 0, 00, \dots \}}{\epsilon 0 0 0} = \{ \epsilon, 0, 00, \dots \}$$

$$\left[\frac{O^* L}{L O} \rightarrow \text{or } \phi \right]$$

Q. Let $L_1 = a^* b a^*$

$$L_2 = b^* a$$

$$L_1 / L_2 = ?$$

- (a) $a^* b a^* + a^*$ ✓
- x (b) c^*
- x (c) $a^* b a^* *$
- x (d) $a^* b a^*$

soln: $\frac{a^* b a^*}{b^* a} \Rightarrow \frac{b, ab, ba, aba, \dots}{a, ba}$ ✓

$$\frac{ba}{ba} = \epsilon$$

$$\frac{aba}{a} \Rightarrow ab$$

(a) b, ab, ba, aba, \dots

(b) $\frac{aba}{a} \Rightarrow ab$ is not there

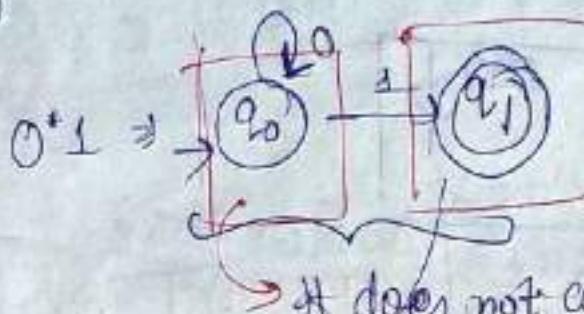
(c) no ϵ

(d) no ϵ

$$\frac{aabab}{abbab} \Rightarrow \phi$$

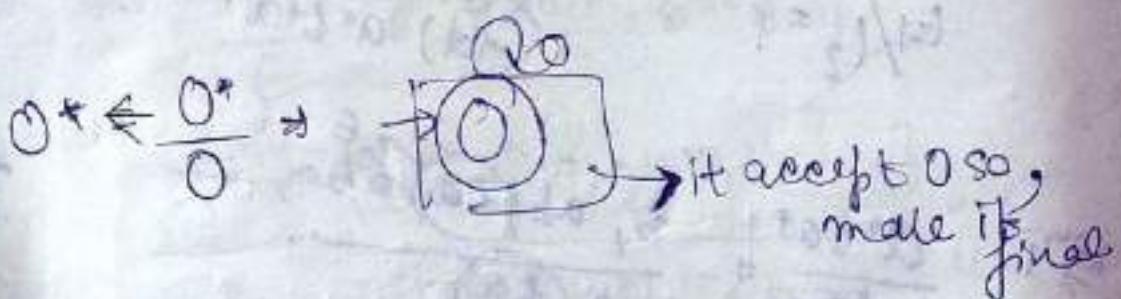
The Quotient of Regular language is always Regular because there exist a finite automata for quotient operation as follows →

$$\frac{L_1}{L_2} \xrightarrow{0^{*1}}$$



→ It does not accept 1^* .

q_1 also not accept any string of 1^* ...



Construct L_1 finite automata, construct the total automata starting from see 1st state if see automata accepts any string of L_2 then make 1st state as final & continue this process for every state. That is the resultant quotient

F.A. hence, reg. lang. are closed under
quotient operation.

(124)

Substitution off :-

$$\Sigma = \{a, b\} \quad L = \{a^n b^m \mid n, m \geq 1\}$$

$$\Delta = \{0, 1\} \quad L' = \{0^n 1^m \mid n, m \geq 1\}$$

Substitution is a mapping from ~~other~~ alphabet Σ to alphabet Δ where each symbol of Σ is replaced by a regular language over the alphabet Δ .

Substitution applied on R.E as follows

$$①. S(\phi) = \phi$$

$$②. S(e) = e$$

$$③. S(a+b) = S(a) + S(b).$$

$$④. S(ab) = S(a) \cdot S(b).$$

$$⑤. S(a^*) = [S(a)]^*$$

Substitution
only
applied
on
symbol not
operator.

$$Q: \quad \Sigma = \{a, b\} \quad D = \{a, b\}^*$$

$$S(0) = (a+b)(a+b)$$

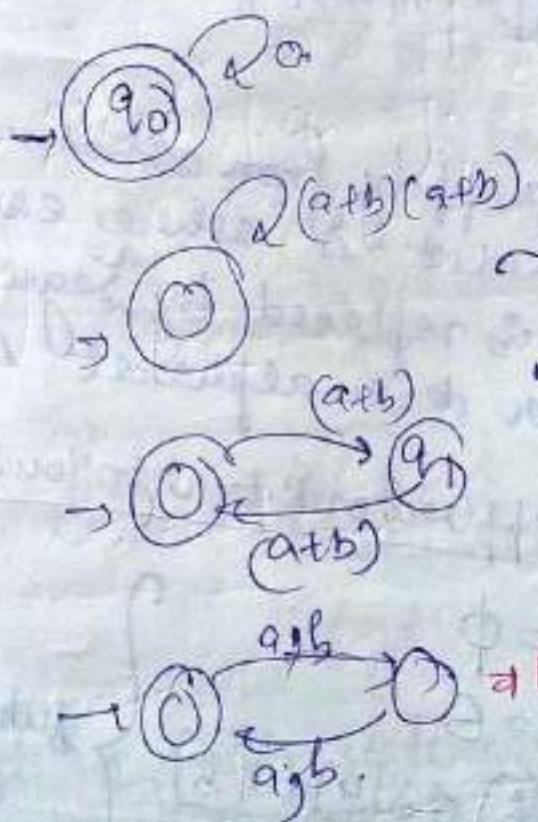
$$L = 0^*$$

$$S(L) = ?$$

$$S(L) = S(0^*)$$

$$= [S(0)]^*$$

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



→ Regular language substitution is always regular because lang. is reg. there exist P.A. for it & substitution are applied on the edges of P.A hence resultant P.A.

is also available, hence tree substituted
Lang. is also reg. (125)

Hence,

Reg. language are closed under
substitution o/p.

(13) Homomorphism o/p:-

- Homomorphism is a special case of substitution in which each symbol of Σ is replaced by single string over the alphabet Δ .

$$h(0) = aa$$

$$h(1) = \epsilon$$

$$L \rightarrow 1^* 0 1^*$$

Hom. applied as follows

$$\textcircled{1} \cdot h(\phi) = \phi$$

$$\textcircled{2} \cdot h(\epsilon) = \epsilon$$

$$\textcircled{3} \cdot h(a+b) = h(a) + h(b)$$

$$\textcircled{4} \cdot h(a \cdot b) = h(a) \cdot h(b)$$

$$\textcircled{5} \cdot h(a^*) = [h(a)]^*$$

$$\Sigma = \{0, 1\} \quad \Delta = \{a, b\}$$

$$Q: \quad \begin{aligned} h(0) &= aa \\ h(1) &= \epsilon \end{aligned} \quad \left. \begin{array}{l} \text{given} \\ \text{ } \end{array} \right\}$$

$$L = \{01^*\}$$

$$\begin{aligned} h(L) &= h(\{01^*\}) \\ &= h(1^*) h(0) h(1^*) \\ &= (\epsilon 1^*) + h(0) (\epsilon 1^*)^* \\ &= \epsilon^* aa \epsilon^* \end{aligned}$$

$$\boxed{h(L) = aa}$$

Homomorphism of a reg. is always Reg.
 Cprove is same as substitution hence,
Reg. Lang. is closed under homomorphism
obj.

④ Inverse H.M :-

Applying Homo. in reverse way is known
 as Inverse Homomorphism. i.e., strings
 are replaced by symbols.

$$E = \{a, b\} \quad D = \{0, 1\}$$

$$h(a) = 0 \quad h(b) = 1$$

$$h(c) = 10$$

$$L = \{1010\}$$

$$h(L^*) = \{ \begin{matrix} abab, & cc, & ba, & cba \\ Baba \end{matrix} \}$$

(126)

$$\begin{aligned} h(h^{-1}(L)) &= \{1010, 1010, 1010, 1010\} \\ &= \{1010\}. \end{aligned}$$

~~Abbabccbab~~

$$Q \quad \Sigma = \{0, 1\} \quad D = \{a, b\}$$

$$h(0) = aa$$

$$h(1) = \underline{c}$$

$$L = 1^* 0 L^*$$

$$h(L) = aa$$

$$h^{-1}(L) = \{ \begin{matrix} \cancel{aaaaaa} \\ 1^* 0 L^* \end{matrix} \}$$

$$\begin{matrix} \cancel{aa} \cancel{c} \\ 1^* 0 \quad 1^* \end{matrix}$$

$$L = h^{-1}(h(L))$$

$$Q = \Sigma = \{0, 1\} \quad D = \{a, b\}$$

$$h(0) = aa$$

$$h(1) = bb$$

$$L_1 = (ab+ba)^*$$

$$\begin{aligned} h^{-1}(L_1) &= (h^{-1}(ab) + h^{-1}(ba))^* \\ &= (h^{-1}(a)h^{-1}(b) + h^{-1}(b)h^{-1}(a))^* \\ &= (\emptyset)^* = \epsilon \end{aligned}$$

$$h(h^{-1}(L_1)) = h(\epsilon) = \epsilon.$$

$$h(h^{-1}(L_1)) \neq L_1.$$

h(h^{-1}(L_1)) need not equivalent to L

The inverse homomorphism of reg. language is always Reg. because in the finite automaton, matching string edges are replaced by corresponding symbols. Hence F.A exist for the inverse homomorphism lang. also.

Hence Reg. lang. closed under Inverse homomorphism o/p,

(18) Infinite union

127

$L_1, L_2, L_3, \dots \rightarrow \{ \text{Reg. Lang.} \}$

$L_1 \cup L_2 \cup L_3 \cup L_4 \cup \dots \rightarrow \{ \begin{array}{l} \text{May or May not} \\ \text{Regular.} \end{array} \}$

$\underbrace{L_1}_{R} \cup \underbrace{L_2}_{R} \cup \underbrace{L_3}_{R} \cup \underbrace{L_4}_{R} \cup \dots = \{ \text{Lang. } m \geq 1 \} \quad (\text{N.R.})$

(By default we always use finite union)

The infinite union of Reg. language is ~~may or may not regular~~ hence Reg. language are not closed under infinite union operation.

(19) Infinite Intersection

$L_1 \cap L_2 \cap L_3 \cap L_4 \cap \dots \cap \{ \begin{array}{l} L_1 \cap L_2 \cap L_3 \cap \dots \cap \\ \text{Reg. Reg. Reg.} \end{array} \} = R \cap R \cap R \cap \dots$

~~may or may not regular~~

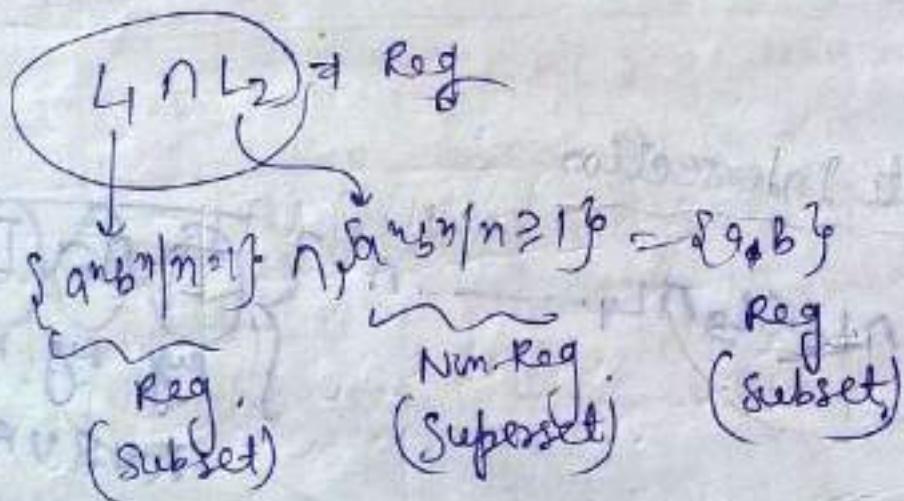
The infinite intersection of regular language may or may not regular. Hence,

Reg. languages not closed under infinite intersection operation.

→ By default union is finite union & intersection is finite intersection.

$$\left\{ \begin{array}{l} L = \{ xw^R \mid w, x \in \{a, b\}^*, y \in \{a, b\}^* \} \text{ Reg.} \\ L = \{ xw^R \mid w \in \{a, b\}^*, y \in \{a, b\}^+ \} \text{ Non-Reg.} \end{array} \right\}$$

$L_1 \rightarrow \text{Reg.}$



- If $L_1 \cap L_2$ is Reg, L_1 is Reg, L_2 need not be regular.

$L_1 \rightarrow \text{Reg.}$

$L_1 \cup L_2 \rightarrow \text{Reg}$

$$(a+b)^* \cup \underbrace{\{a^nb^n | n \geq 1\}}_{\text{reg left non-reg}} = \underbrace{(a+b)^*}_{\text{reg}}$$

- If $L_1 \cup L_2$ is Reg, L_1 is Reg, L_2 need not be Regular.

$L_1 \rightarrow \text{Reg}$

$L_1 \cdot L_2 \rightarrow \text{Reg}$

$$\underbrace{\emptyset}_{\text{reg}} \cdot \underbrace{\{a^nb^n | n \geq 1\}}_{\text{non-reg}} \rightarrow \emptyset \quad \downarrow \text{reg}$$

- If $L_1 \cdot L_2$ is Reg, L_1 is Reg, L_2 need not be Regular.

Decision properties of Regular Languages :-

The following problems are decidable under Reg. Languages:-

- ①. Emptiness problem
- ②. Finiteness problem
- ③. Membership problem
- ④. Equivalence problem.

$F.A = R.L = R.E = R.G$. \rightarrow all have same decision property.

Grammars.

Set of rules used to define strings of a language known as grammar.

Language formulating as follows →

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

$V = \text{set of variables}$ Non-terminals
(capital)

$T = \text{set of terminals}$

$P = \text{no. of productions.}$

S: starting symbol.

(129)

- For every language grammar exist and every grammar generate one language.
- All grammars is of the form,

$$\boxed{\alpha \rightarrow \beta}$$

, where β is replacement of α . and β should contain atleast one non-terminal.

Derivation:-

- ① The process of deriving strings from the given grammar by replacing left hand side part of the grammar by its corresponding right hand side part.
- ② The derivation may be left most derivation or right most derivation.

left most derivation → it is a process of derivation in which left most non-terminal is replaced by its corresponding right hand side parts at every step.

Right most derivation → it is a process of derivation in which right most non-terminal is replaced by its right hand side part at every step.

Parse tree @ Derivation tree :

It is a tree representation of derivation process. Known as parse tree.

Yield of Parse tree (P.T.) :-

All leaf nodes of the parse tree known as yield of that parse tree.

By reading the yield from left to right sentence of the grammar can be generated.
Hence, [yield length is equal to string length generated by that parse tree.]

Sentential form:

- Each step in the derivation is known as sentential form.
- If the derivation is left most derivation then the sentential form is left sentential form.
- If the derivation is right most derivation, the sentential form is right sentential form.
- The sentential form may be combination

of terminals and non-terminals.

- for Reg-lang. there exist a grammar known as Reg grammar. For CFL, there exist a grammar known as CF grammar. For CFL, there exist a grammar known as CS grammar.
- for R.E.L, there exist a grammar known as Unrestricted grammar.

(130)

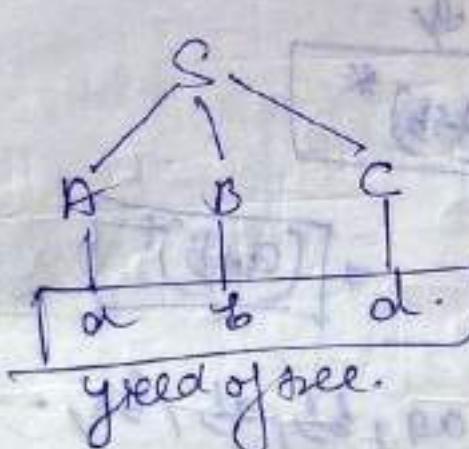
$$\begin{array}{l} \text{*} \\ S \rightarrow ABC \\ A \rightarrow a \\ B \rightarrow b \\ C \rightarrow d \end{array}$$

Left most

A	B	C
↓	a	b
	b	
	ab	c
	ab	
	abd	

Right most

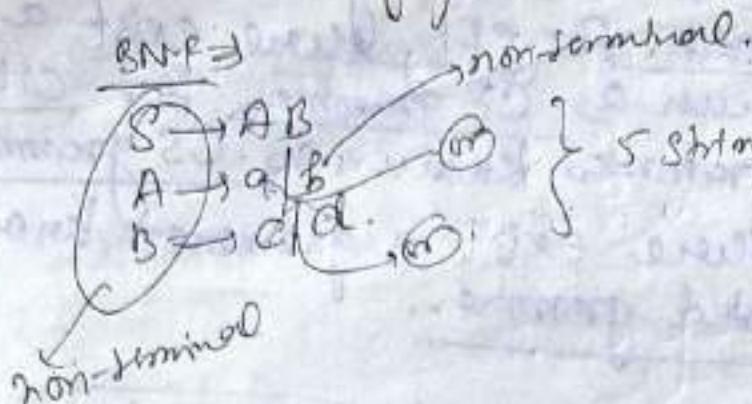
A	B	C
		↓
A	B	d
		↓
A	B	d
		↓
		abd



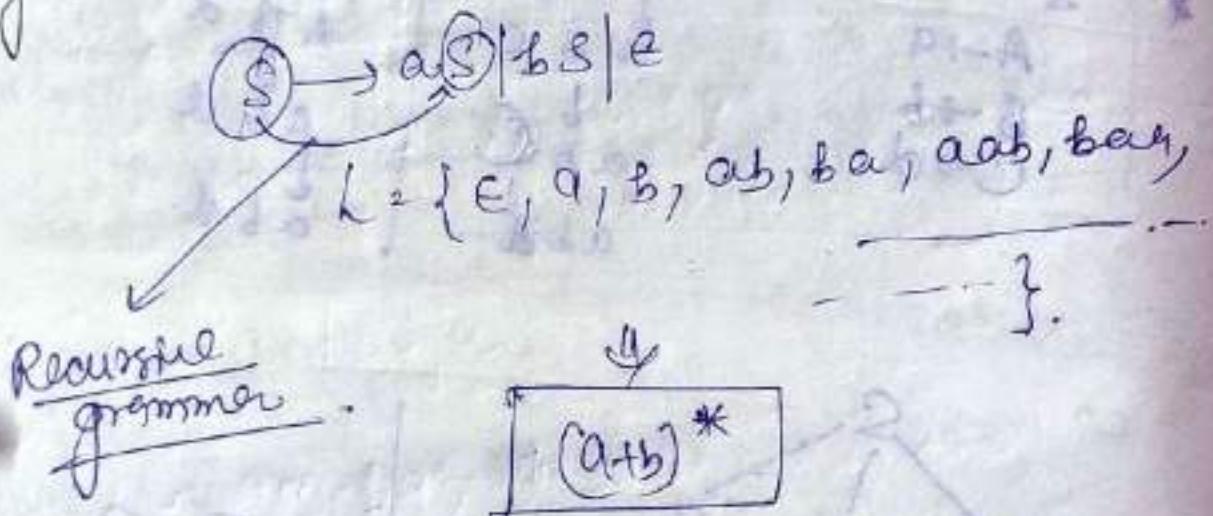
S
 { A B C } all are senten-
 { a B C } ce-Hal form.
 { ab C }
 { ab d }

{ L → Cr } possibly
 { Cr → L } of question, do, etc.

Q. Total no. of non-terminal present on right hand side of grammar must be.



Q. Identify language generated by following grammars \rightarrow



Q. $S \rightarrow aS \mid bS \mid a \mid b \rightarrow \boxed{(a+b)^+}$

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

Q. $S \rightarrow aSb \mid \epsilon$ (Tab S (Non-terminal) kehte hain aata hi means 'as' comparison jaise hote hain)

$L = \{\epsilon, ab, aabb, \dots\} \rightarrow \{a^n b^n \mid n \geq 0\}$

DATE
16/2/09

$S \rightarrow aSa | bSb | \epsilon$

(13)

$L = \{ aae, a\cancel{a}, bb, abba, baab, \dots \}$

even palindrome.

$abaax$

so, it does not
contain starting & ending
with same
like abaa.

Q. $S \rightarrow aSa | bSb | a | b | \dots$

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

odd length
palindrome

Q. $S \rightarrow aSa | bSb | a | b | \epsilon$

All
palindrome

Q. $S \rightarrow aSbS | bSaS | \epsilon$

$$S \rightarrow aSbS \mid bSaS \mid \epsilon$$
$$L = \{ \epsilon, ab, ba, aabbab, ababba, \\ bababa, bbaabe, \\ ababab, \dots \}$$

Equal no. of a s &
Equal no. of b s

are equal

$$L = \{ x \mid n_a(x) = n_b(x) \}$$

Q8. $S \rightarrow aSbS \mid a$

$$L = \{ a, aa, ba, \cancel{bab}, baa, bba, \dots \}$$

Ending with a

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

Q9.

$$S \rightarrow (\underline{\underline{S}}) \mid \epsilon$$

Balanced parentheses

not equal no. of parentheses

(16) $S \rightarrow AB \xrightarrow{a^+ b^+}$

$$\begin{array}{l} A \rightarrow aA | a \xrightarrow{a^+} \\ B \rightarrow BB | B \xrightarrow{b^+} \end{array}$$

(16.23)

$$L = \{ ab, aab, abb, aabb, \dots \}$$

Starting & ending with diff. symbol.

(17) $S \rightarrow aSbb | abb$

$$L = \{ aabb, aaabb, a\cancel{abb}bb, \dots \}$$

$$\boxed{\{ a^n b^n \mid n \geq 1 \}}$$

If there is middle relation means here some confusion going to happen.

(18) $S \rightarrow aSb | aAb$

$$A \rightarrow aA | a \xrightarrow{a^+}$$

$$L = \{ aaAbb, aaabb, aaaaabb, \dots \}$$

$$\boxed{\begin{array}{c} aabb \\ aaabb \end{array}}$$

$\{a_n\}_{n \geq m}$

$$Q. S \rightarrow aAb/aSb/abB.$$

$$\begin{array}{c} A \rightarrow aA/a \longrightarrow a^+ \\ B \rightarrow bB/b \longrightarrow b^+ \end{array}$$

$$\begin{array}{ll} a \sqsubseteq b & a \sqsubseteq b \\ \sqcup \\ aAb & aBb \\ \downarrow & \downarrow \\ a^+ & B^+ \end{array}$$

aab, aabb, aaaaabb,

$a \leq b$
 $a \neq b$

a.B.b.

↓
g+

at

at

$$L = \{q^{n+m} \mid n \neq m\}.$$

$$Q. \quad S \rightarrow A.B \mid aA.bB \cdot (aa)^* (bb)^* \mid a(aa)^* b(bb)^*$$

$$A \rightarrow aa.A \mid e \rightarrow (aa)^*$$

$$B \rightarrow bb.B \mid e \rightarrow (bb)^*$$

$$L = \{ \epsilon, aabb, \underline{aaabb}, ab, \underline{\underline{aab}} \}$$

anonym

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

Grammar Construction

Q1 Design tree grammar,

$$L = \{ a^n b^m \mid n, m \geq 1 \}$$

$$\boxed{\begin{array}{l} S \rightarrow A \# B \\ A \rightarrow aA \mid a \\ B \rightarrow bB \mid b \end{array}}$$

2. $L = \{ a^n b^m \mid n, m \geq 0 \}$

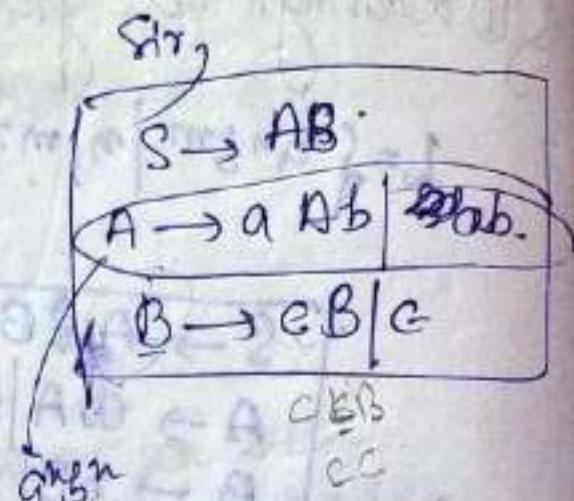
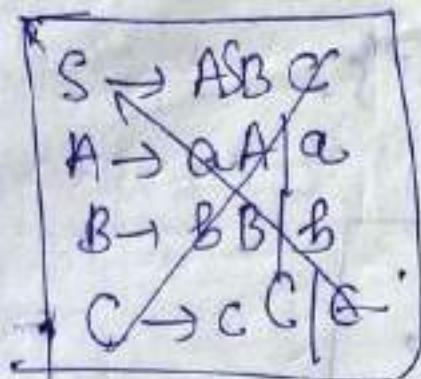
$$\boxed{\begin{array}{l} S \rightarrow AB \\ A \rightarrow aA \mid \epsilon \\ B \rightarrow bB \mid \epsilon \end{array}}$$

3. $L = \{ a^n b^m c^k \mid \begin{array}{l} n \geq 1 \\ m \geq 2 \\ k \geq 0 \end{array} \}$

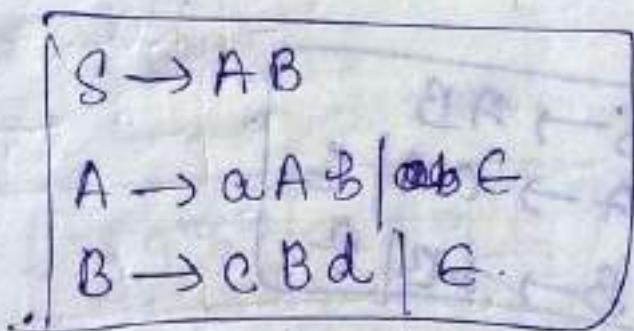
$$\boxed{\begin{array}{l} S \rightarrow ABC \\ A \rightarrow aA \mid a \\ B \rightarrow bB \mid b \\ C \rightarrow CC \mid \epsilon \end{array}}$$

$$\boxed{\begin{array}{l} S \rightarrow A b B . C \\ A \rightarrow aA \mid a \\ B \rightarrow bB \mid b \\ C \rightarrow cC \mid \epsilon \end{array}}$$

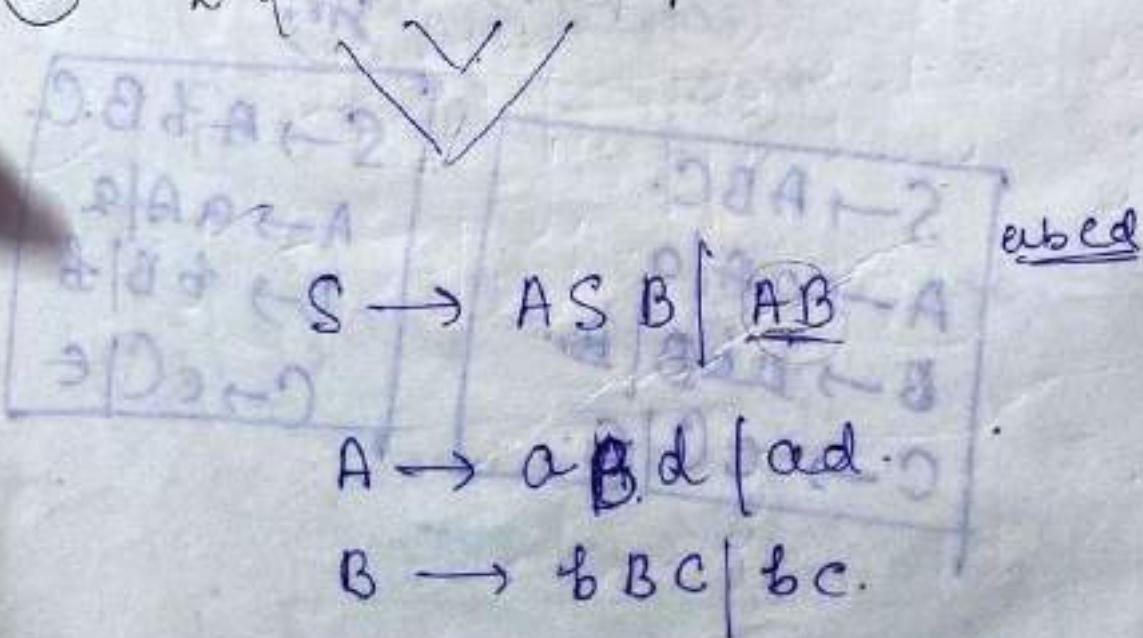
④. $L = \{a^n b^m c^m d^n \mid n, m \geq 1\}$



⑤. $L = \{a^n b^n c^m d^m \mid m \geq 0\}$



⑥. $L = \{a^n b^m c^m d^n \mid n, m \geq 1\}$.



~~S_{1,2}~~ ~~aAd~~

$$\left[\begin{array}{l} S \rightarrow aAd \mid aSd \\ A \rightarrow bAc \mid bc \end{array} \right]$$

(PA)

⑦. $L = \{ a^{n+m} b^m c^n \mid n, m \geq 1 \}$

~~detektiv~~ $S \rightarrow AS \mid AB$

$A \rightarrow aA$

$B \rightarrow bBC \mid bc$

$$\left[\begin{array}{l} S \rightarrow aAC \mid aSc \\ A \rightarrow aAb \mid ab \end{array} \right]$$

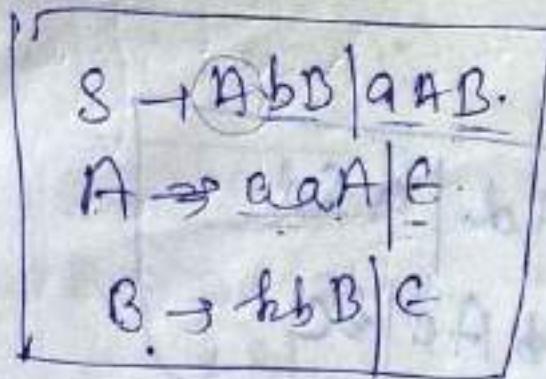
⑧. $L = \{ a^n b^m \mid (n+m) \text{ is odd} \}$

↓ L

odd even
even odd
odd even

a ab
aaaaabb

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



aab, aas,

aaa
aaab

Q. L = {anbm | n < m}

$$S \rightarrow \alpha \underline{ASB} \\ S \rightarrow A \underline{SB}$$

ath
aahs.
a*b*

$$A \rightarrow a \underline{A} | \epsilon \\ B \rightarrow b \underline{B} | b$$

aabb

Ex:

$$\boxed{S \rightarrow a \underline{SB} | a \underline{AB} } \\ A \rightarrow b \underline{B} | b$$

Q. - Find the grammar that generates all odd length palindromic strings of English language.

Soln: $(ab)^*(aa) + abab$

odd length

aaa aaba

25

$$S \rightarrow aSa \Big| bSb \Big| - \dots zSz \Big| \dots$$

136

52 | production

$$\begin{array}{r} 70 \\ \times 20 \\ \hline 1400 \end{array}$$

11. Even length palindrome string of Hindi language.

→ अंतर्मा | अंतर्मा | → अंतर्मा | अंतर्मा

Istehen, → अअ | अअ | आआ|...
— — | गलग

$$\begin{array}{r} 49 \\ \times 3 \\ \hline 147 \end{array}$$

$$\begin{array}{r} 49 \\ \times 49 \\ \hline 441 \end{array}$$

ଗୋଟିଏ କାହାର ପାଇଁ କାହାର
ଜାମାର କାହାର ପାଇଁ କାହାର
କାହାର କାହାର କାହାର

(Q1)

Q. Find the grammar that generates all strings of a s & b s where each string starting & ending with some symbol.

Sol:

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

$$\boxed{S \rightarrow aAa \mid bAb \mid a \mid b} \\ A \rightarrow aA \mid bA \mid \epsilon}$$

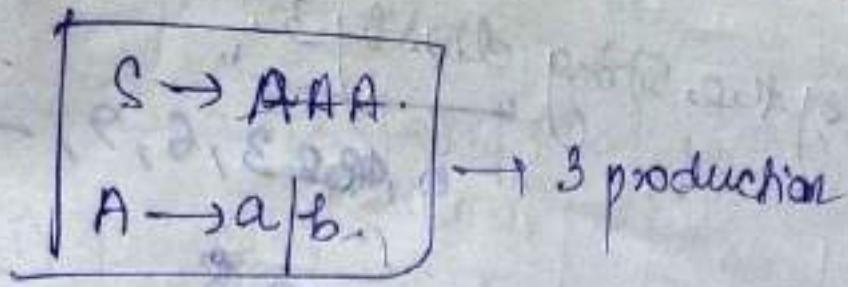
Q. Starting & ending with diff symbol,

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

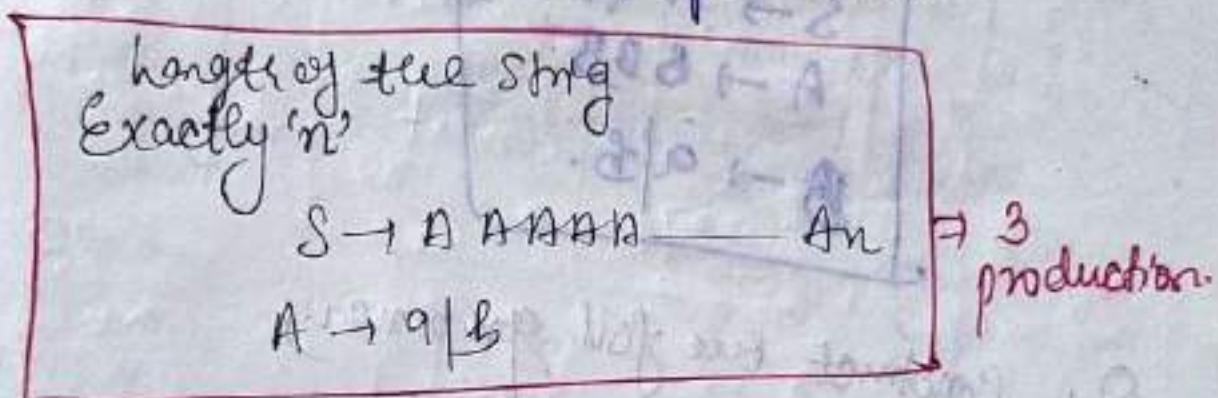
$$\boxed{S \rightarrow aAb \mid bAc} \\ A \rightarrow aA \mid bA \mid \epsilon}$$

Q. Length of the string exactly 3.

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

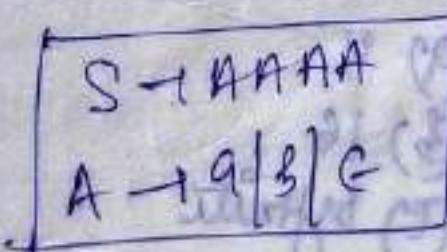


length of the string exactly 9
 $\rightarrow 3$ production.

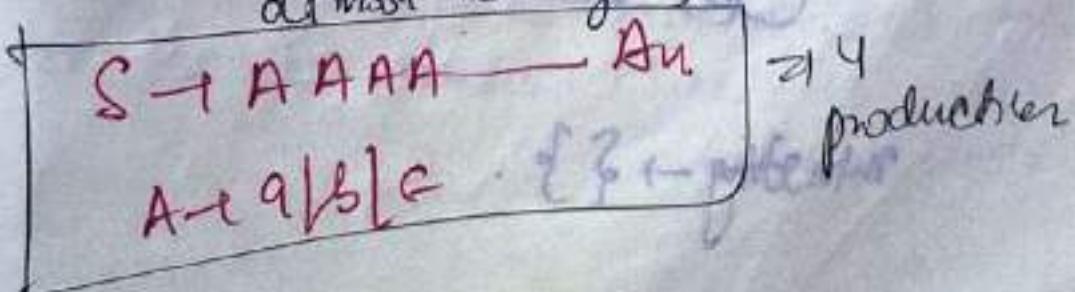


Q. length of the string atmost 4,

$$(a+b+\epsilon)(a+b+\epsilon)(a+b+\epsilon)(a+b+\epsilon)$$



atmost n length,



Q. length of the string div. by 3:

0, 3, 6, 9, ...

② $\left[(a+b)(a+b)(a+b) \right]^*$

$S \rightarrow A S | \epsilon.$

$A \rightarrow B B B$

$B \rightarrow a | b.$

Q. Construct free full grammar,

$S \rightarrow aS | bS | ab | ba.$

How many 4 length strings are generated
by this grammar,

Say:

(A) 4.

(B) 16

(C) Infinite

(D) None.

N.B. — AAAA \rightarrow 2

answering $\rightarrow \{ \}$ $\rightarrow [] / [] + A$

Q The grammar is (137)

$S \rightarrow aS \mid bS \mid ab \mid bSa \mid aNb$

①. Regular grammar

②. CF grammar

③. CS G

④. Unrestricted Grammar.

CF Gn.

$q \rightarrow$

+ (T+N) β α

+ (T+N) β β

$q \rightarrow$

$|q| \geq 1$

+ (T+N) β γ

$\beta \rightarrow$

(T+N) β γ

$\boxed{m \beta x \leftarrow A}$

$\boxed{x \beta \leftarrow A}$

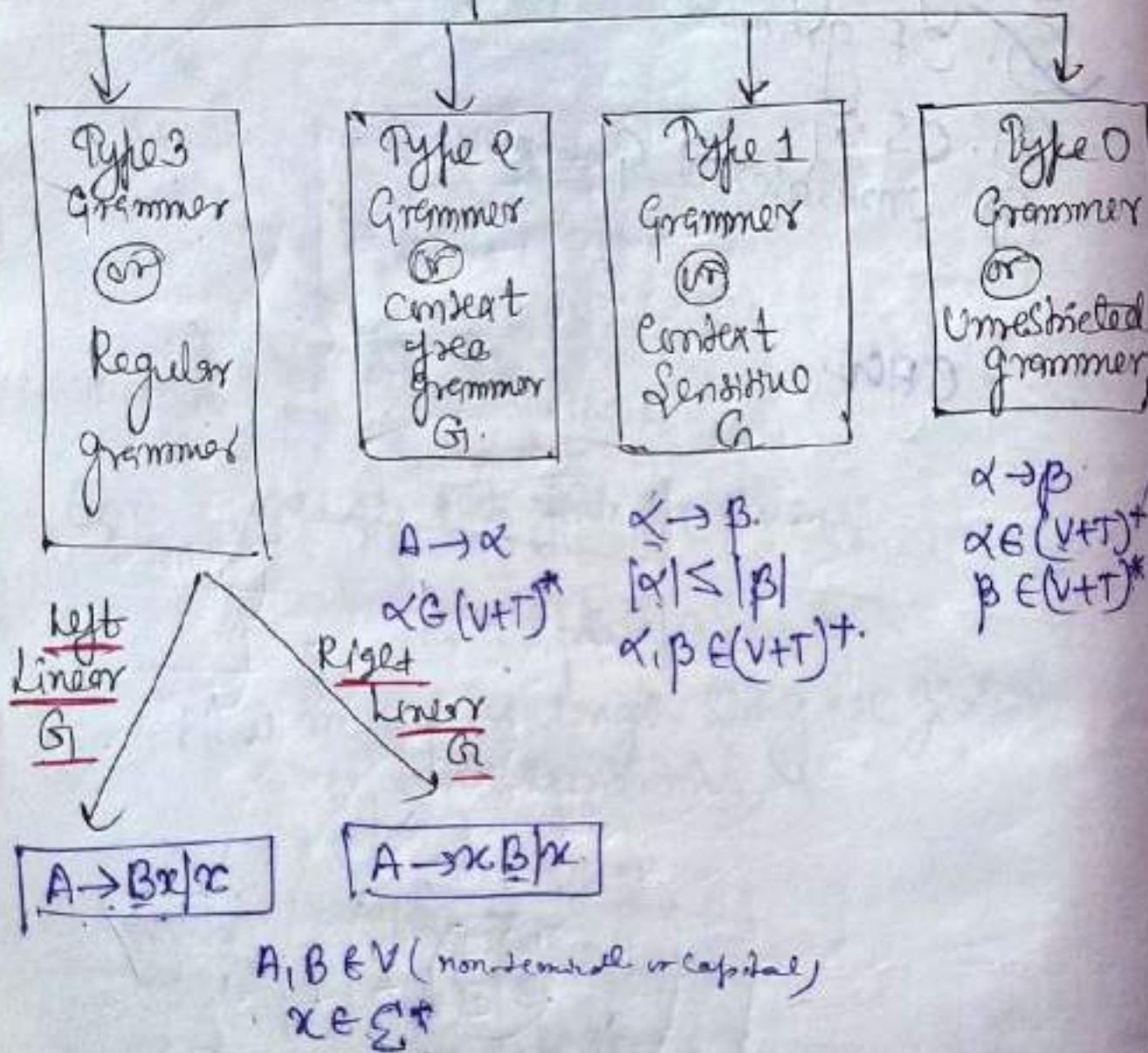
$\beta \in \Sigma$

$\beta \in \Sigma$

$\boxed{-q \leftarrow x}$ only one is running with $m \beta$ #
one position minus blue so endow
lexicon - non

Chomsky invented tree grammar.

Types of grammar



All these grammar is of the form $\alpha \rightarrow \beta$, where α should contain atleast one non-terminal.

Linear Grammer :-

(139)

In any grammer left hand side exactly one non-terminal & Right hand side at most 1 non-terminal exist that is called as Linear Grammer.

The Linear grammer may be left linear (or) right linear (or) middle linear.

If the grammer is either left linear (or) right linear that is Regular ^(but not vice versa) Hence, all regular grammars are linear grammars but all linear grammar need not be regular.

Ex: $S \rightarrow aSb | aB$

linear but not Regular

Ex: $S \rightarrow aS | Sb | a$

Reg. means completely Right L & Left L
other regular

Ex: $S \rightarrow aS | bS | a$

Regular ✓
Right linear ✓

Un

(S)



Context free G,

$$A \rightarrow \alpha B \quad | \quad \alpha \rightarrow \beta C \delta E$$

Type 0

$$\alpha A \beta \rightarrow$$

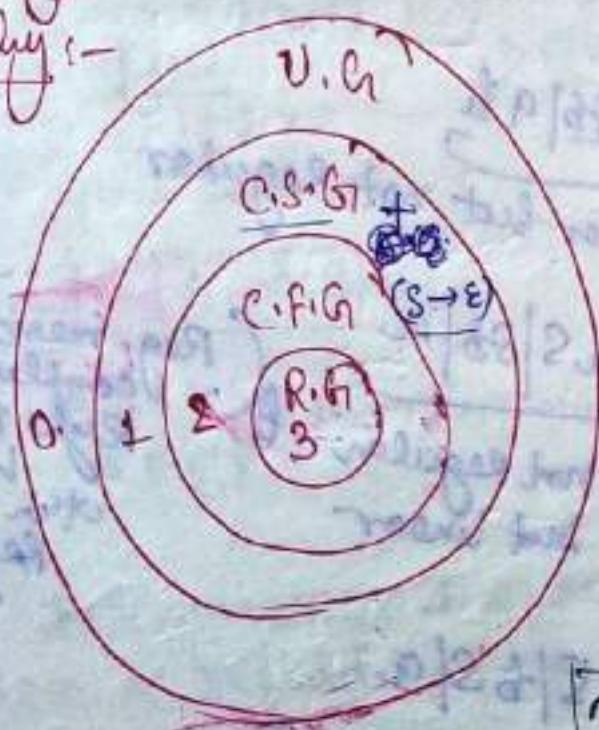
$A \rightarrow B C D \dots$ In Type 2
possible.

Type 0 or

$| \alpha | \geq | \beta |$ possible here
but not in Type 2

Chomsky

hierarchy:-

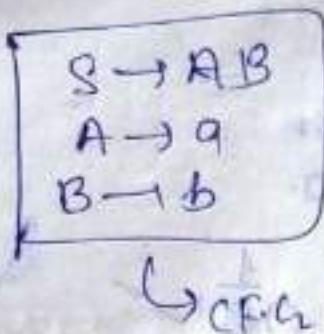


$i=0,1,2$

Type $i+1 \subset$ Type i

We always start from type 3 to questions.

Ex:



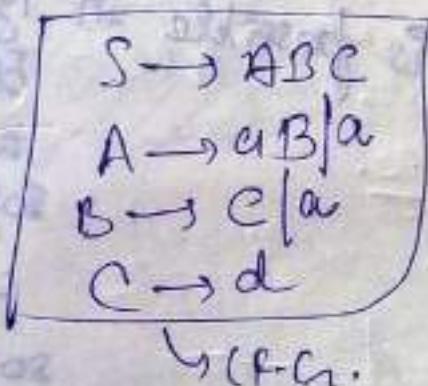
Type 3 X

Type 2 ✓
1 ✓
0 ✓

if 2 satisfy 1, so
has to be
satisfy.

Ex:

Type of following grammar?
So possible types no. = ?

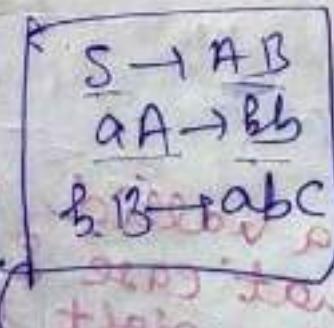


Type 3 X

Type 2 ✓
1 ✓
0 ✓

Ex:

Type of grammar = ? & possible types no. = ?



3 X
2 X.
1 ✓

404

not possible
for 2
 $S \rightarrow AD$

C.S.G. out no tree of
removing out / p

$$\left. \begin{array}{l} S \rightarrow AB' \\ aA \rightarrow bb \\ bB \rightarrow abc \end{array} \right\} \text{or } (V3).$$

Ex: Identify type of grammar

$$\boxed{\begin{array}{l} S \rightarrow ABC | G \\ aA \rightarrow bb \\ bB \rightarrow ba \\ C \rightarrow d \end{array}}$$

1 ✓
 (CNFL)
 CSCh.

$A \rightarrow \epsilon \Rightarrow$ is possible in Type 2 ✓
 Type 3 ✓

Type 1 X
 so, we add
 $S \rightarrow \epsilon$ in
 Type 1
 so, true

Chomsky Hierarchy is
 maintained
 by adding $S \rightarrow \epsilon$ in
 type 1.

Note: $S \rightarrow \epsilon$ is a valid CSCh production
 in that case S should not
 present on the right hand side part
 of the grammar.

ex:

$$S \rightarrow AB | C$$

$$aA \rightarrow aa$$

$$bB \rightarrow bb$$

$$S \rightarrow AB | C$$

$$aA \rightarrow aa$$

$$bB \rightarrow bb$$

(140)

type'0'

→ Apply this condition only for type 0 grammar.

Q. Identify the type of grammar,

$$\begin{array}{l} S \rightarrow AB \\ aA \rightarrow a \\ aB \rightarrow bb \end{array}$$

$$\begin{array}{l} S \rightarrow AB \\ aA \rightarrow bb \\ bB \rightarrow bb \end{array}$$

$$\begin{array}{l} S \rightarrow AB | C \\ aA \rightarrow aa \\ bB \rightarrow bb \end{array}$$

$$\begin{array}{l} S \rightarrow AB | C \\ aA \rightarrow aa \\ B \rightarrow bb \end{array}$$

Context free grammar

$$A \rightarrow x$$

$$aAB \rightarrow axb$$

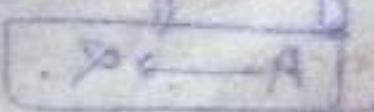
Context Sensitive grammar

$$aAb \rightarrow a\alpha$$

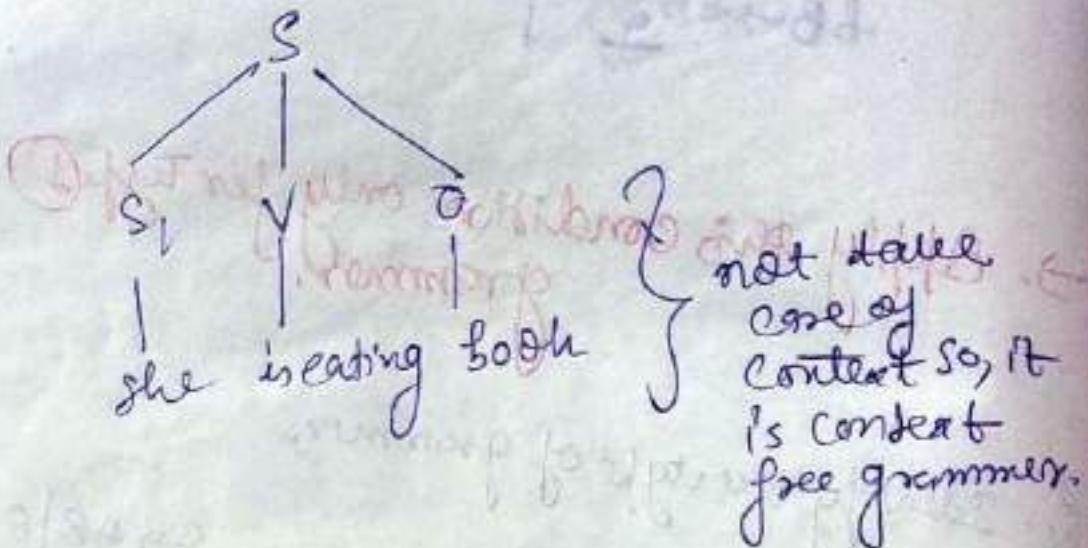
no replace

he is playing cricket,

S V O



type 2

$$\left\{ \begin{array}{l} S \rightarrow S_1 NO \\ S_1 \rightarrow He | she | They. \\ V \rightarrow playing | eating | reading | sleeping. \\ O \rightarrow cricket | book | food. \end{array} \right.$$


- Regular grammars are suitable only to represent regular languages rules.
- Regular grammars are not suitable to represent rules of languages which are having comparison their symbols.
- Context-free grammars are used for comparison languages but they are context-free hence, context sensitive behaviours cannot be represented by using these grammars.
because in Context-free grammar A is replaced by α for the production,

$$A \rightarrow \alpha$$

without looking about its context.
Hence, see grammar production,
Context-sensitive

(141)

$$a(A)b \rightarrow a(\alpha)b$$

means, A is replaced by α in the
context of small a & b only.

Hence, 'C' language, C++, java are context
Sensitive languages because semantic
analysis is done in these languages.

main()

C, C++, Java

int α ; b(α);
;
;
;

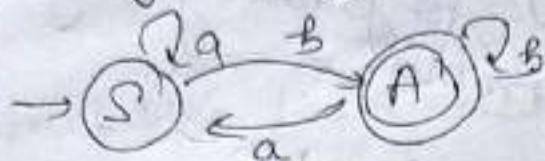
*not allowed to declare
two times a
variable.*

→ Fortran language is context-free language
because no semantic analysis.

✓ By default regular grammar is right
linear grammar.

Regular grammar \iff Finite automata:- ①

Q. Construct the Regular grammar equivalent to the following FA,

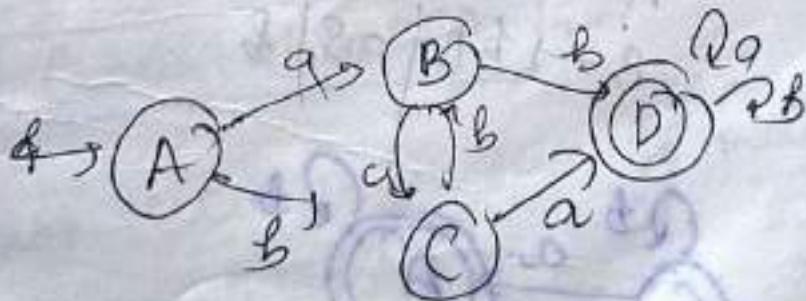


Soln.

$$\begin{aligned} S &\rightarrow aS \mid bA \mid b \\ A &\rightarrow bA \mid aS \mid b \end{aligned} \quad \left. \begin{array}{l} \text{Right linear} \\ \text{grammar.} \end{array} \right\}$$

$$\begin{aligned} V &\rightarrow \{ \text{States} \} \\ T &\rightarrow \Sigma \\ P &\rightarrow S^* \\ S &\rightarrow \{ S \} \end{aligned}$$

Q. Construct the Reg. grammar equivalent to automata,

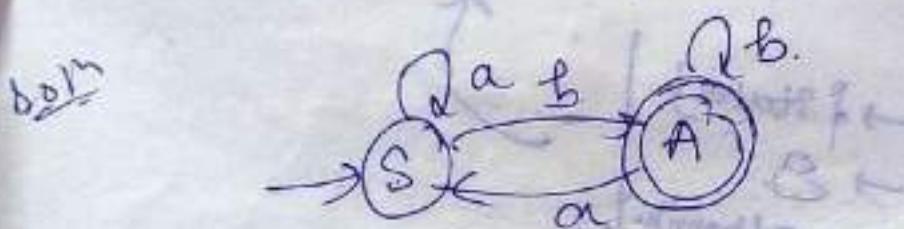


$$\begin{aligned} A &\rightarrow aB \mid bC \\ B &\rightarrow bD \mid b \mid ac \\ C &\rightarrow aD \mid a \mid bB \\ D &\rightarrow aD \mid bD \mid a \mid b \end{aligned} \quad \left. \begin{array}{l} 12 \text{ productions.} \end{array} \right\}$$

→ To construct Regular grammar from the
automata. Note [given automata should
be a DFA].

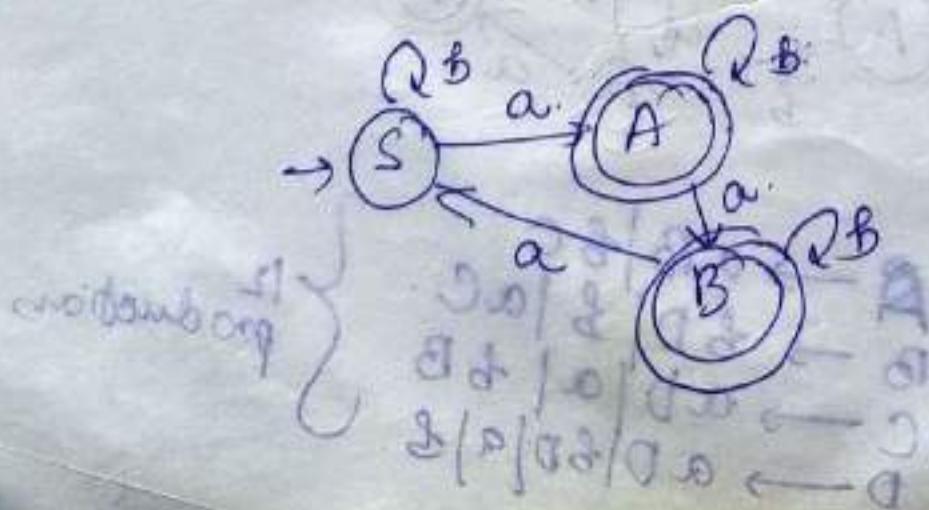
Q. construct the finite automata Equivalent
to following Reg.
grammar,

$$\begin{aligned} S &\rightarrow aS \mid bA \mid B \\ A &\rightarrow aS \mid bA \mid B \end{aligned}$$



Q. $\begin{aligned} S &\rightarrow bS \mid aA \mid a \\ A &\rightarrow bA \mid aB \mid b \mid a \\ B &\rightarrow bB \mid aS \mid b \end{aligned}$

Identify
Lang of given
Reg. grammar.

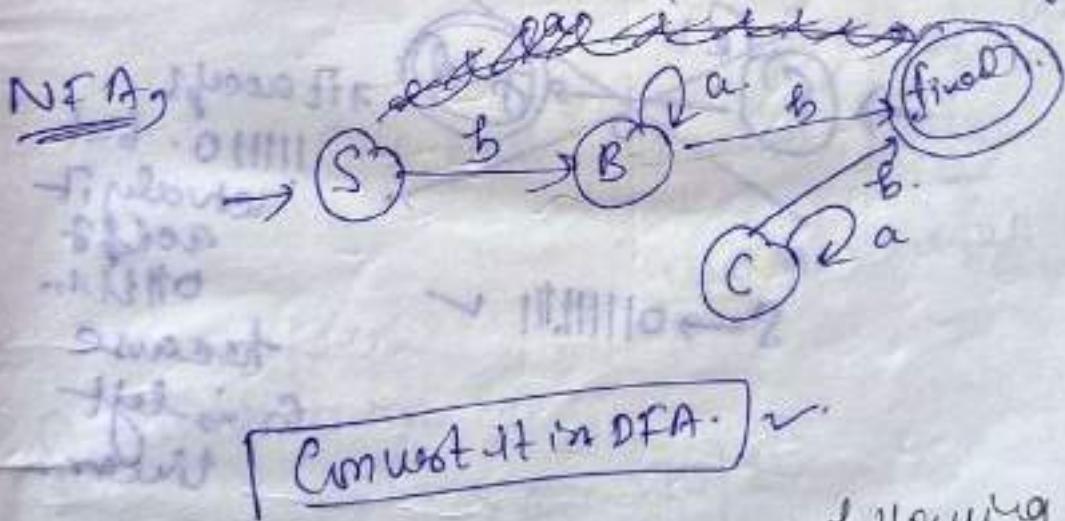


$L = \{a, 3a, ab, aba, bab, baba\}$ 2

No. of
a's, Not divisible by 2

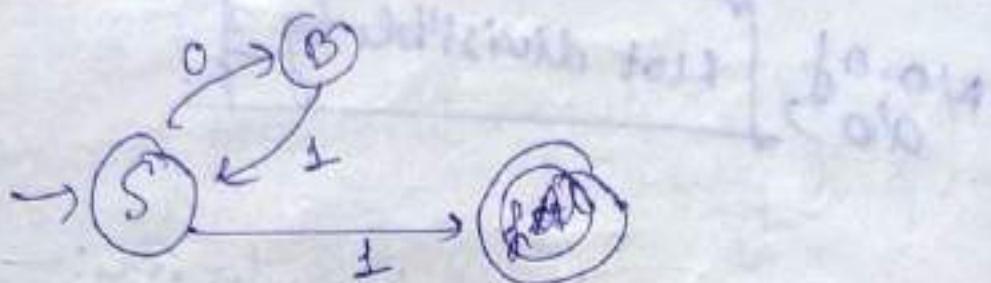
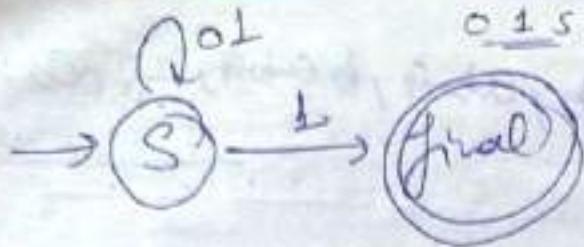
Q. Construct a finite automata for Reg:-

$$\begin{aligned} S &\rightarrow bB \\ B &\rightarrow aB \mid b \\ C &\rightarrow aC \mid b \end{aligned} \quad \left. \begin{array}{l} \text{right linear} \\ \text{gramme so} \\ \text{Regular} \\ \text{grammar.} \end{array} \right.$$



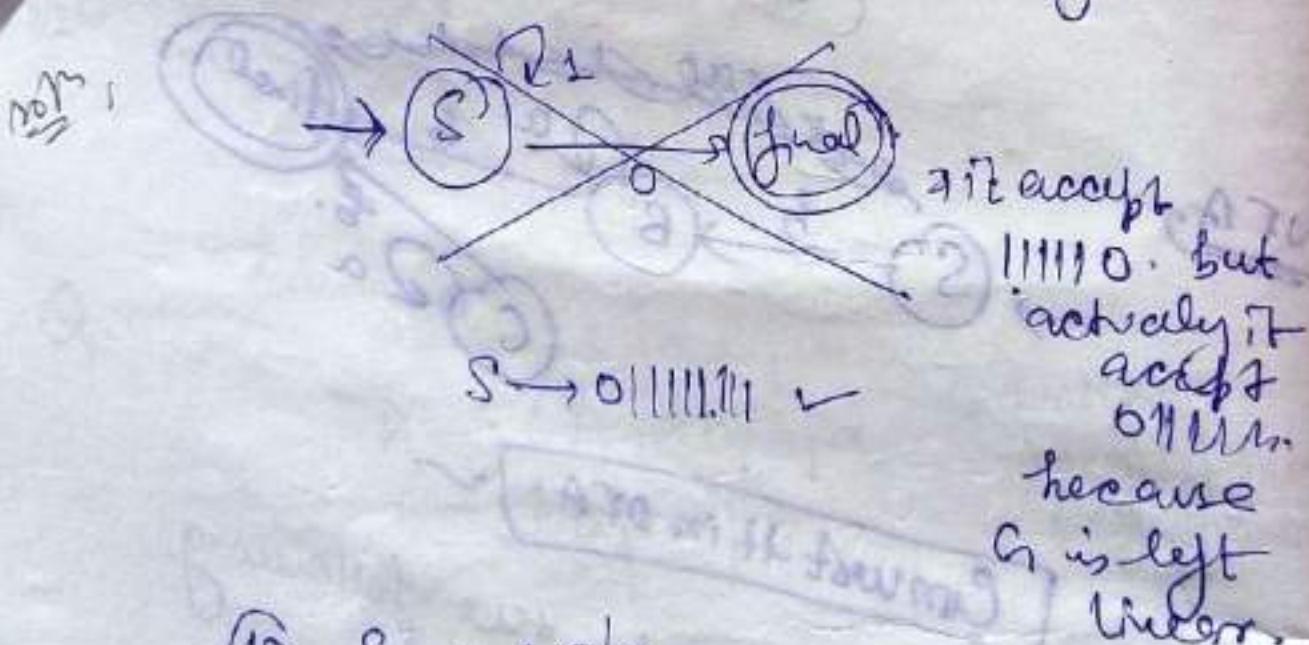
Q. Construct the F.A for the following grammar, 2 2

$$S \rightarrow 01S \mid 1$$

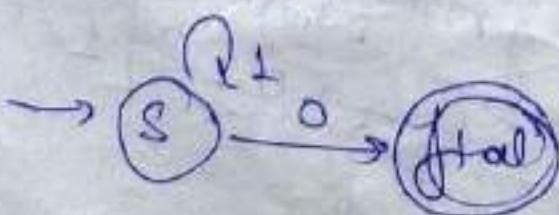


Q. Convert the Regular grammar to P-A equivalent
to R.G,

$S \rightarrow S10$ } left linear.
grammar.

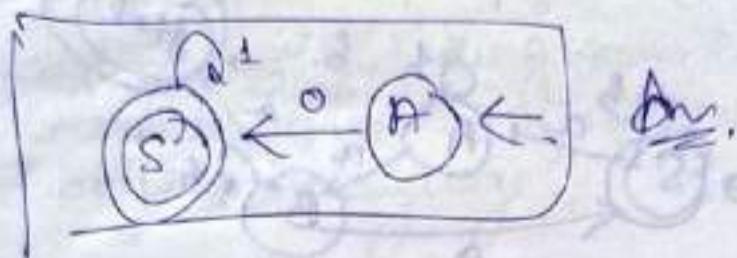


④ $S \rightarrow 1S0$



Now apply Reversal automata (3)

Initial \leftrightarrow final & reverse
final \leftrightarrow initial



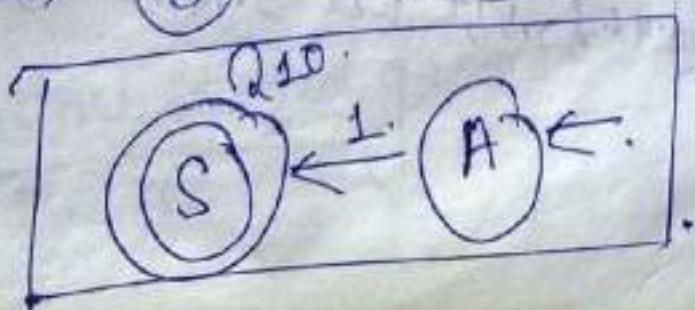
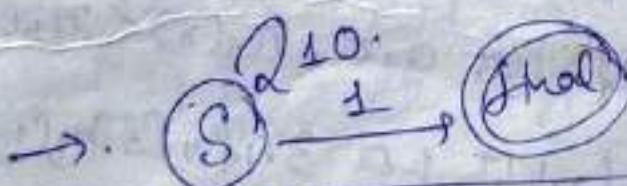
\rightarrow Automat for grammar always Right-linear
we also convert it into Left-linear.

(Automata) \rightarrow Reverse

Now we have Left-linear

Q. construct the P.A.

$$S \rightarrow \text{Q}^10 | 1$$



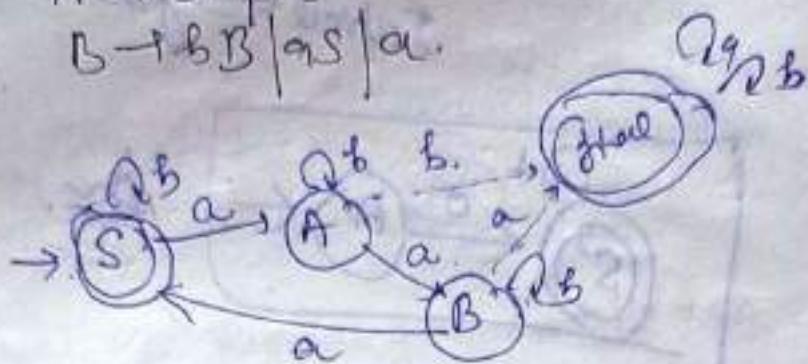
~~Q. Ques~~

$$S \rightarrow bS|aA|b$$

$$A \rightarrow bA|aB$$

$$B \rightarrow bB|aS|a.$$

NFA



	a	b
S	A	S
A	B	(A)J.
B	(S)J	B.
(AJ)	(AJ)	(SJ).
(SJ)	(AJ)	(SJ).

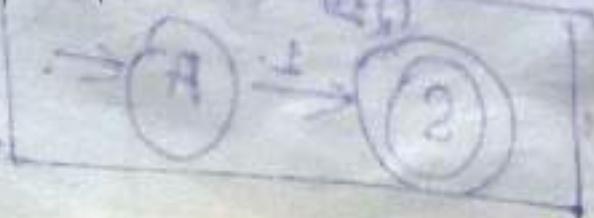
which of the following is lang. of given grammar,

No 1 Reg. $L = \{w | w \in \{a, b\}^* \text{ and } n_a(w) > n_b(w)\}$

No 2 Reg. $L = \{w | w \in \{a, b\}^* \text{ and } n_a(w) < n_b(w)\}$

No 3 Reg. $L = \{w | w \in \{a, b\}^* \text{ and } 3n_a(w) = n_b(w)\}$

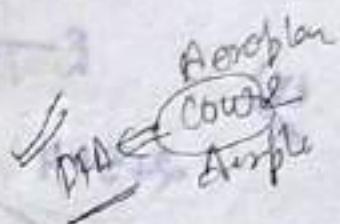
Q. Nme



(4)

(a) $\rightarrow \underline{babba} =$ (b) $\rightarrow aaba =$ (c) \rightarrow

(a), (b), (c) are not reg. but the grammar is right linear reg. grammar so, it generate always reg language.



Ambiguity problem

→ A grammar is said to be ambiguous for atleast one string x there exist more than one left most derivation or more than one right most derivation or more than one parse tree. (or) for one string one left most derivation one right most derivation exist both will produce different parse trees.

Unambiguous grammar →

A grammar is said to be unambiguous if all strings of given grammar there exist exactly one left most derivation (or) exactly one right most derivation (or) only one parse tree (or) one left most derivation, one right most derivation exist for the same string but both will produce same parse tree.

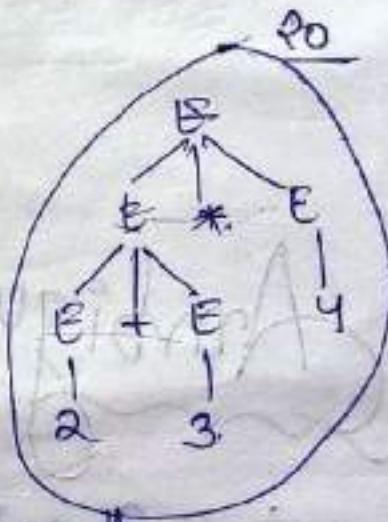
$$E \rightarrow E+E \mid E * E \mid id.$$

ex:

$2+3*4$



14.



PO

here + here
higher precedence

here + here
higher precedence

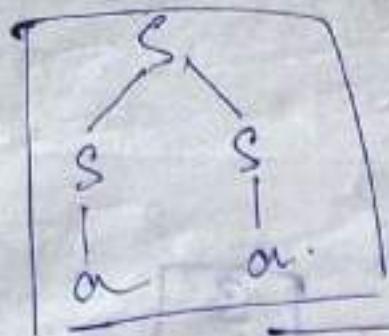
Q. Verify the following grammar is Ambiguous & unambiguous

$$S \rightarrow SS \mid a$$

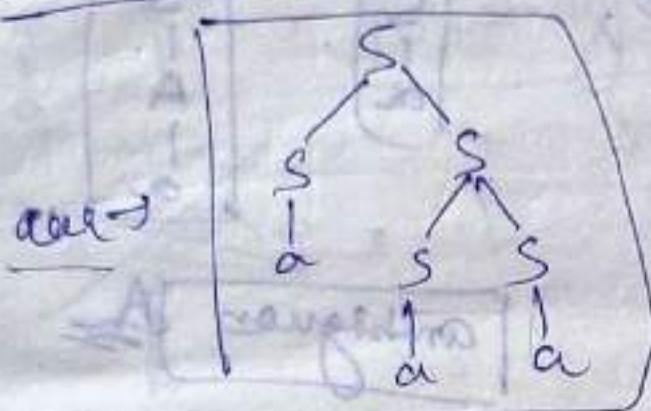
(5)

Q15.

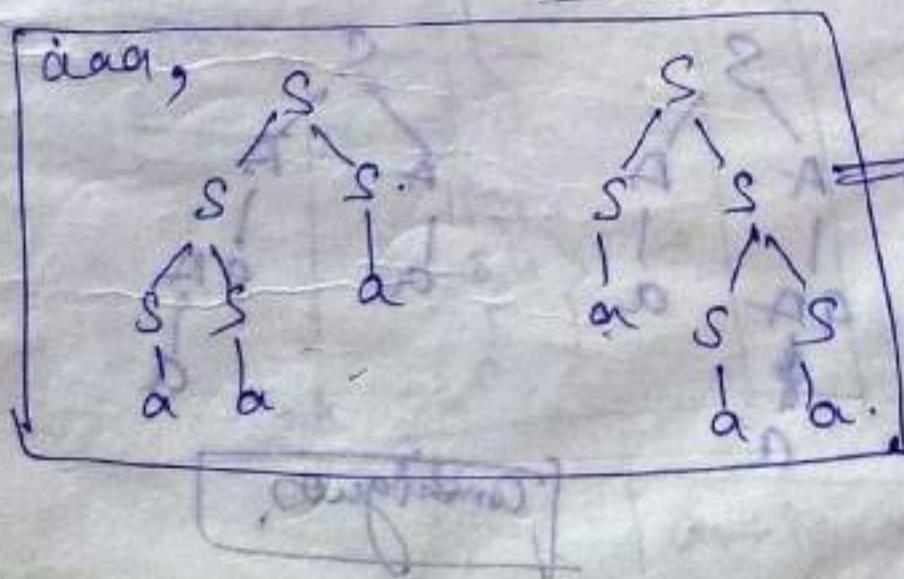
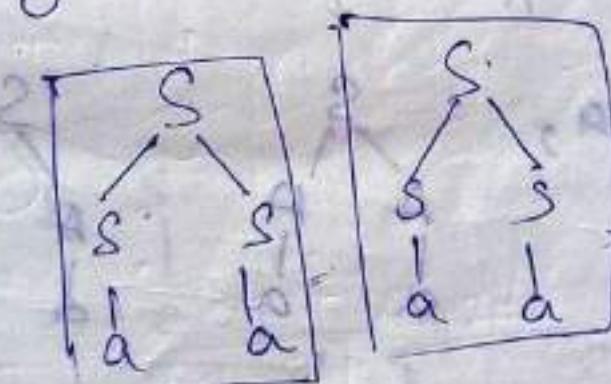
Q9-7



Recursion
infinte string is
generated.



net a. string is aa,



different
string
free
grammar
vs
ambiguous.

B. Ambigües & ambigüez?

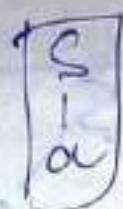
$$S \rightarrow A | a$$

$$A \rightarrow a$$

Relación

defin.

$a,$

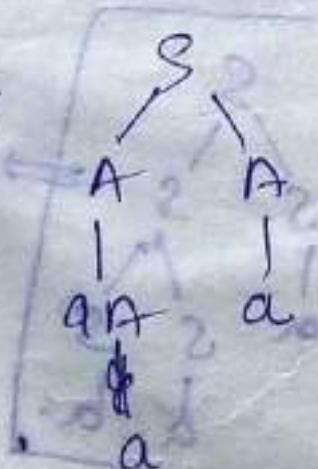
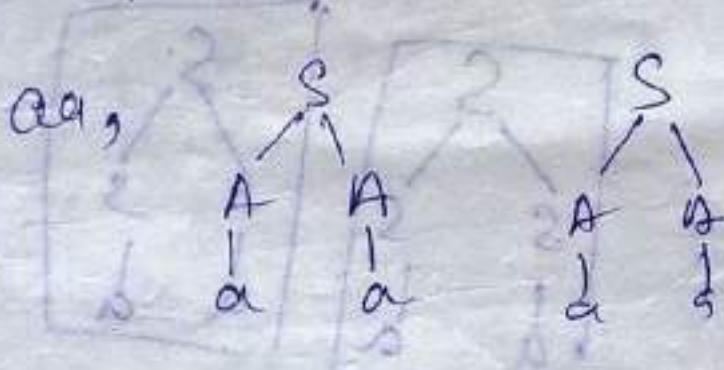


Ambigües \Rightarrow

C.

$$S \rightarrow AA$$

$$A \rightarrow aA | a$$

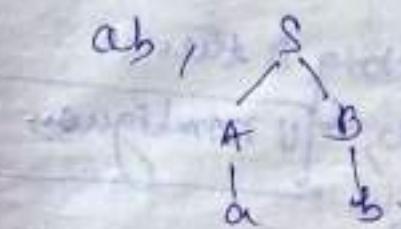


Ambigües.

Q. $S \rightarrow AB$

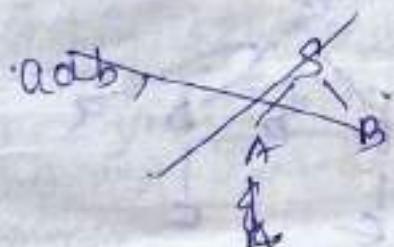
$A \rightarrow a$

$B \rightarrow b$

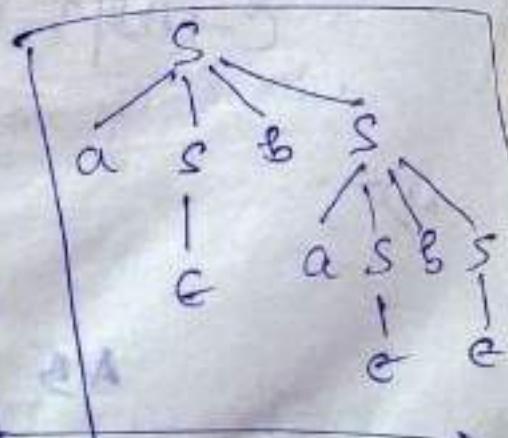
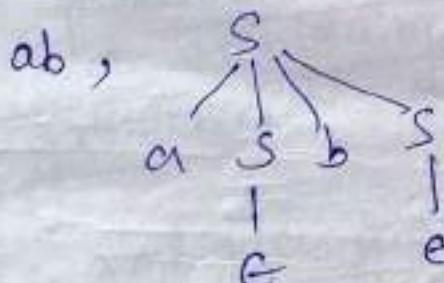


unambiguous

only one string
possible ab.



Q. $S \rightarrow aSbS \mid bSaS \mid \epsilon$



ambiguous

Q. $S \rightarrow AaAb | BbBa$

$A \rightarrow G$

$B \rightarrow C$

Ans:

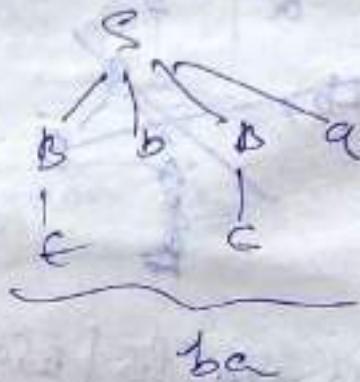
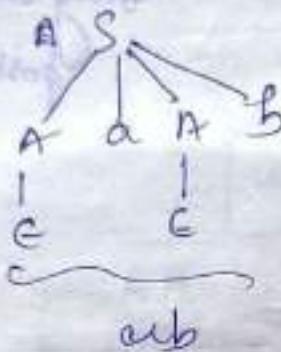
\boxed{ab}, \boxed{ba}

$ab ab$

only two string is there

so,

unambiguous



Q. $S \rightarrow AB | BC$

$A \rightarrow BA | a$

$B \rightarrow CC | b$

$C \rightarrow AB | c$

Ans:

AB

AB
 BA
 AB

ab ab

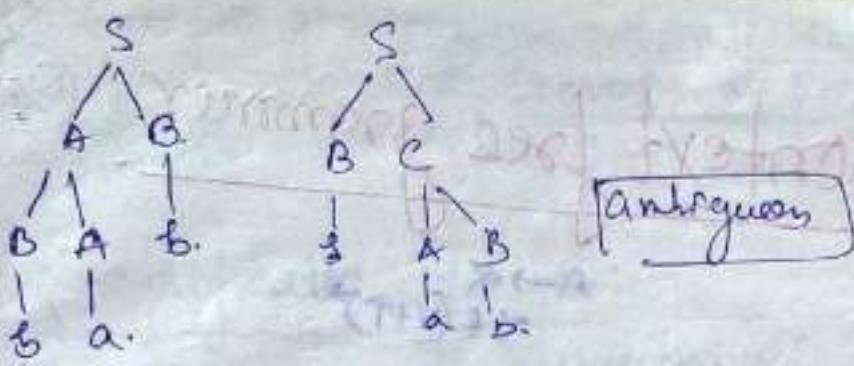
BC

AB
 CC

AB

aabs
aaab

not ambiguous



→ Unambiguity problem of a grammar is undecidable problem because there is no algorithm to decide ambiguity of a grammar.

→ Removal of ambiguity is Undecidable problem hence, it is impossible to remove ambiguity from every ambiguous grammar.

→ The ambiguous grammar from which removal of ambiguity not possible known as inherently Ambiguous grammars.

transfer of basic 2nd year concepts and facts
conceptual understanding for writing

Context free grammar



$$A \rightarrow \alpha \\ \alpha \in (V \cup T)^*$$

$$S \rightarrow aSb | S | e$$

Regular

$$S \rightarrow aSb | ab$$

↓
CFG

$$S \rightarrow AB \\ A \rightarrow a \\ B \rightarrow b$$

↓
CFG

$$S \rightarrow AB | BC \\ A \rightarrow BA | a \\ B \rightarrow CC | b \\ C \rightarrow AB | a$$

↓
CFG

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

CFL but not regular

$$L = \{ ab \} \\ (\text{reg. lang.})$$

Note:- It is undecidable to tell whether a context free grammar generates regular languages or not.

Syntax → structure
Semantics → meaning

→ Context free grammars are used to represent Syntax of programming languages.

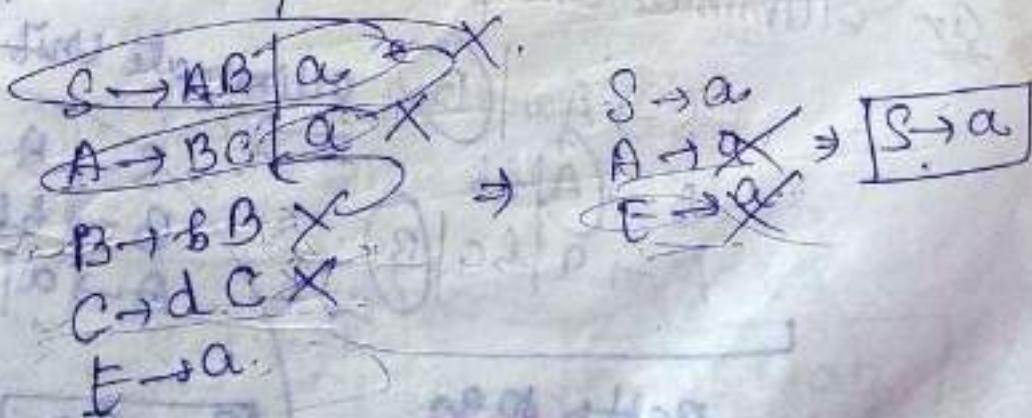
→ Content free grammars are not suitable to represent semantics of programming language.

① Simplification of CFG:-

- Eliminate useless variable \Rightarrow N.T.
- Unit production.
- Null production

To simplify content free grammar eliminate useless variable \Leftrightarrow Non-terminal, unit production \Leftrightarrow Null production.

Elimination of useless variable means, eliminate free variable which are not deriving a terminal string and eliminate the variable which are not required for derivation.



Ex: Eliminate free useless variable →

$$\begin{array}{l}
 S \rightarrow AB \mid \cancel{A} \\
 A \rightarrow aAb \mid bAa \mid a \\
 B \rightarrow bbA \mid aaB \mid AB \\
 C \rightarrow abCa \mid aDBX \\
 D \rightarrow bD \mid aC \cancel{X} \\
 E \rightarrow a \cancel{X}
 \end{array}$$

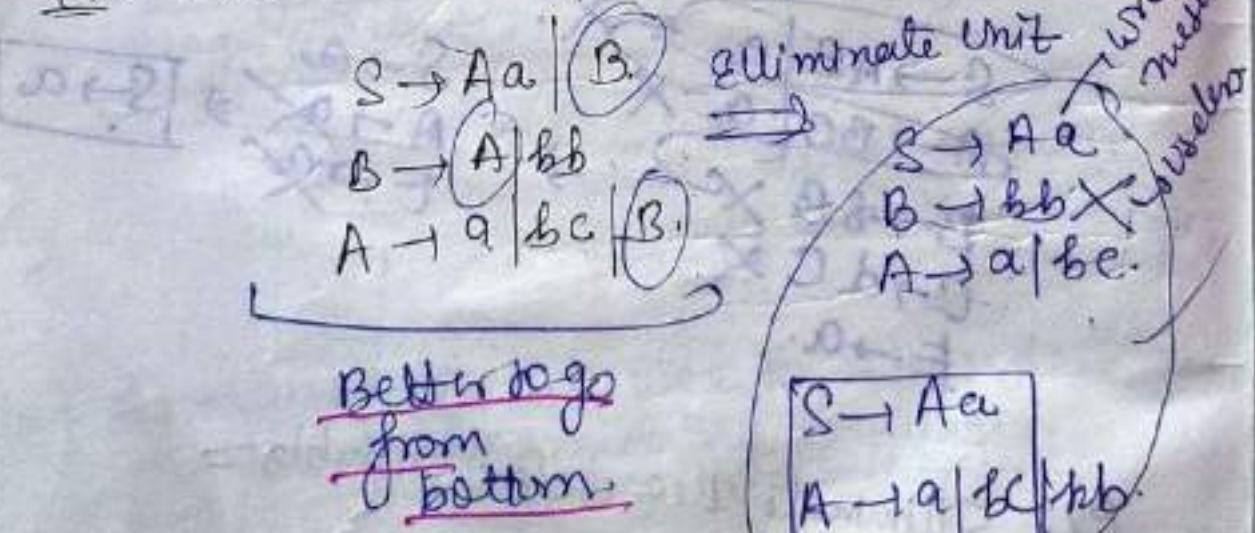
$$\begin{array}{l}
 S \rightarrow AB \\
 A \rightarrow aAb \mid bAa \mid a \\
 B \rightarrow bbA \mid aaB \mid AB
 \end{array}$$

J productions.

Q. Unit production \rightarrow Any production is of the form variable gives a variable.
 $(A \rightarrow B)$

known as unit production.

Ex: Eliminate unit production,



(9)

$$\begin{array}{l}
 A \rightarrow B \\
 B \rightarrow C \\
 C \rightarrow D \\
 D \rightarrow E \\
 E \rightarrow a
 \end{array}
 \quad
 \begin{array}{l}
 A \rightarrow a \\
 B \rightarrow aX \\
 C \rightarrow aX \\
 b \rightarrow aX \\
 E \rightarrow aX
 \end{array}
 \quad
 \Rightarrow [A \rightarrow a]$$

useless

- Replacing tree variable by its right hand side part is done.
- If tree unit variables are productions are eliminated then it is easy to detect useless variable.

S → Aa | a | bc | bb. remove

S → Aa a bc bb.
B → a be bb ×
A → a be bb

useless S → Aa | a | bc | bb
A → a | be | bb

Null production (or) E production \rightarrow
 $(A \rightarrow \epsilon)$.

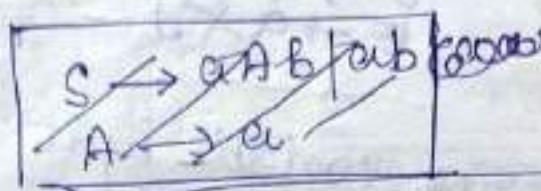
Ex: Eliminate null production,

$$\begin{array}{l}
 S \rightarrow aS, b. \rightarrow S \rightarrow ab \\
 S_1 \rightarrow aS_1 | b | \epsilon. \quad S_1 \rightarrow aS_1 | b
 \end{array}$$

$$S \rightarrow aS, b | ab$$

$$S_1 \rightarrow aS_1 | ab.$$

Q. $S \rightarrow aABbB$ Eliminate null production
 $A \rightarrow a | e$
 $B \rightarrow e | b$



$S \rightarrow aAbB | aAb | ab | abB$
 $A \rightarrow a$
 $B \rightarrow b$

Null
Unit
Trieles

Q. $S \rightarrow AB$
 $A \rightarrow aAA | B$
 $B \rightarrow bBB | e$

$S \rightarrow e | aAB | B | A | A^+BB$

aAAAB

$A \rightarrow aAA$
 $B \rightarrow bBB$

empty

Σ

$S \rightarrow AB | B | A$
 $A \rightarrow aAA | aA | a$
 $B \rightarrow bBB | bB | b$

it accept
 $\{L - f\}$

$S \rightarrow EX$

Note :

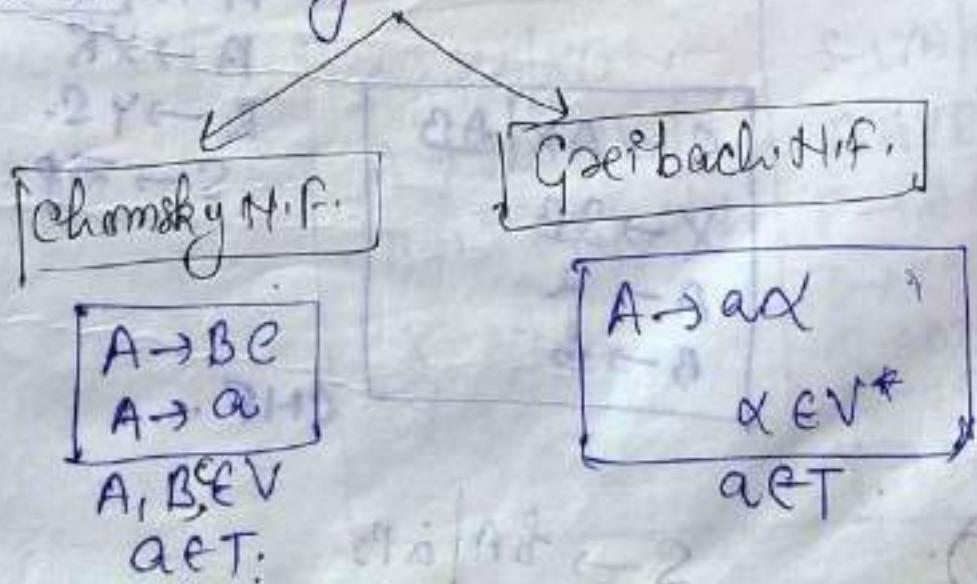
- (1) To simplify the grammar Eliminate
- null production followed by
 - unit production followed by useless variables.
- (2) while removing ϵ production ϵ string may also removed from the language.

③ Normal forms of CFG \rightarrow

as condⁿ applied on anything.

→ Applying condition ① restriction on right hand side part of the grammar known as normal form of the grammar. ($A \rightarrow \underset{1}{\alpha} \underset{2}{\beta}$)
condⁿ

These are mainly two normal forms in CFG



- Before normalizing the CFG should be simplified.
- If the grammar is in CNF, then it is suitable for decision properties.
- If the grammar is in GNF then it is suitable for PDA construction.

Ex: Convert the following grammar into CNF forms

$$S \rightarrow aSb \mid ab$$

$$A \rightarrow BC$$

$$A \rightarrow a$$

Assume terminal
as non-terminal.

$$A \rightarrow a$$

$$B \rightarrow b$$

$$S \rightarrow ASB \mid AB.$$

$$A \rightarrow a$$

$$B \rightarrow b.$$

$$A \rightarrow X \boxed{Y Z B} R$$

$$A \rightarrow X R$$

$$R \rightarrow Y S.$$

$$S \rightarrow Z P.$$

$$S \rightarrow AX \mid AB$$

$$X \rightarrow SB.$$

$$A \rightarrow a$$

$$B \rightarrow b$$

$$\boxed{S \rightarrow A}$$

$$CMR. \rightarrow A$$

(2)

$$S \rightarrow bA \mid aB$$

$$\boxed{S \rightarrow A}$$

$$T \rightarrow A$$

(11)

$$A \rightarrow bAA | aS| a$$

$$B \rightarrow aBB | bS| b$$

$$S \rightarrow BA | AB$$

$$A \rightarrow BAA | AS | A$$

$$B \rightarrow TABB | BS | B$$

$$A \rightarrow BAA$$

X.

$$B \rightarrow ABB$$

X → AA.

$$Y \rightarrow BB$$

not used because

A on right
had said
not only
a many
production.

$$S \rightarrow BA | AB$$

$$A \rightarrow BX | AS | a$$

$$B \rightarrow BY | BS | b$$

$$X \rightarrow AA$$

$$Y \rightarrow BB$$

$$\Rightarrow S \rightarrow YA | XB$$

$$A \rightarrow YAA | XS | a$$

$$B \rightarrow XBB | YS | b$$

$$X \rightarrow a$$

$$Y \rightarrow b$$

$$S \rightarrow YAKN$$

$$A \rightarrow YP | XQ | a$$

$$P \rightarrow AA$$

$$B \rightarrow XQ | YS | b$$

$$Q \rightarrow BB$$

$$X \rightarrow a$$

$$Y \rightarrow b$$

$$Q. \quad S \rightarrow aAB \mid BB$$

$$A \rightarrow a$$

$$B \rightarrow b$$

$$\text{Ans. } S \rightarrow A \underbrace{A B}_{X} \mid BD$$

$$S \rightarrow AX \mid BB$$

$$X \rightarrow AB$$

$$A \rightarrow a$$

$$B \rightarrow b.$$

CNF

$$Q. \quad S \rightarrow aSa \mid bSb \mid c.$$

$$\text{Sol} \rightarrow aac$$

$$S \rightarrow ASA \underbrace{\mid BSB}_{X \quad Y} \mid c.$$

$$S \rightarrow AX \mid BY \mid c$$

$$X \rightarrow SA$$

$$Y \rightarrow SB.$$

$$A \rightarrow a$$

$$B \rightarrow b.$$

$$A \rightarrow a\alpha$$

$$\alpha \in V^*$$

(12)

Q. Convert the following into form of GNF,
 $S \rightarrow aSB|ab$.

Soln:

$$A \rightarrow aV^*$$

1st step :— Connect all the right hand side in tree form of variables.

$$A \rightarrow a$$

$$B \rightarrow b$$

$$S \rightarrow \boxed{ASB|AB}$$

2nd step

$$\begin{array}{ccc} \text{left} & & \text{right} \\ \text{most} & \downarrow & \text{in} \\ \text{is in} & & \text{is in} \\ \hline \text{GNF} & & \text{GNF} \\ \hline \end{array}$$

place it

$$\boxed{\begin{array}{c} S \rightarrow aSB|ab \\ A \rightarrow a \\ B \rightarrow b \end{array}}$$

Q. $S \rightarrow AB$

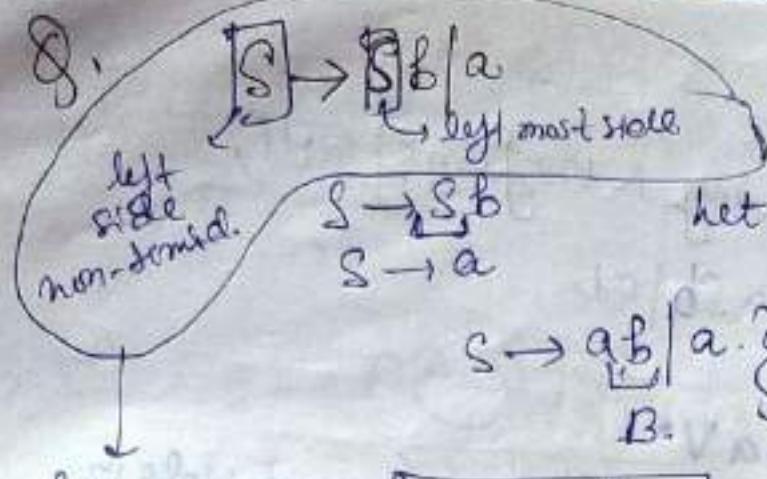
$$A \rightarrow aA|bB|b$$

$$B \rightarrow b$$

see AB.

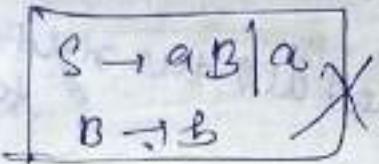
$$\begin{array}{c} \text{not} \\ S \rightarrow AB \end{array}$$

$$\begin{array}{c} S \rightarrow aAB|bBD|bB \\ A \rightarrow aA|bB|b \\ B \rightarrow b \end{array}$$



$S \rightarrow ab|a$ } \times wrong way.

B.



→ Grammer contains left recursion then
it's difficult to convert it into GNF form
Hence left recursion should be
eliminated from the grammer.

Elimination of left recursion: —

$$A \rightarrow A\alpha | \beta .$$

denoting variable present in left
most side of R.H.S part

$$S \rightarrow Sb|a \quad [ab^*] \checkmark$$

$$S \rightarrow Sb|a \quad [ab^*] \checkmark$$

$\xrightarrow{S \rightarrow aS'}$

$\xrightarrow{S' \rightarrow bS'} t$

$$A \rightarrow A\alpha | B.$$

(B)

$$\boxed{A \rightarrow \beta A'}$$

$$A' \rightarrow \alpha A' | \epsilon$$

Any grammar is of the form $\boxed{A \rightarrow A\alpha | B}$ then
the left recursion from that grammar is
removed by rewriting the grammar in the
form of right recursion as follows \rightarrow

In general,

$$A \rightarrow A\alpha_1 | A\alpha_2 | \dots | A\alpha_n | B_1 | B_2 | \dots$$

$$\boxed{A \rightarrow \beta_1 A' | \beta_2 A' | \dots}$$

$$\boxed{A' \rightarrow \alpha_1 A' | \alpha_2 A' | \dots | \epsilon}$$

without
left recursion.

Q. Convert the following grammar into CNF

$$S \rightarrow Sb | a.$$

$$\boxed{S \rightarrow aS'}$$

$$\boxed{S' \rightarrow bS' | \epsilon.}$$

Elliminate null
production

$$\boxed{S \rightarrow aS' | a.}$$

CNF:

Q. Elliminate left recursion & convert into
CNF;

$$S \rightarrow Sa | Sb | e | d$$
$$S \rightarrow es' | ds'$$
$$S' \rightarrow as' | bs' | e$$
$$S \rightarrow e | d | es' | ds'$$
$$S' \rightarrow a | b | as' | bs'$$

Q. Eliminate left recursive from the following grammar
& convert in GNF

$$A \rightarrow Ba | b$$
$$B \rightarrow Ac | d$$

$\overbrace{A}^{\text{Do L.R.}} \rightarrow \underbrace{Ba'}_A | \underbrace{b}_B$. Indirect left recursive.

$$A' \rightarrow a A' | e$$
$$A \rightarrow \underbrace{Ba}_A | b$$
$$\overbrace{Ac}^{\text{Do L.R.}} | \overbrace{d}^{\text{Do L.R.}}$$
$$B \rightarrow \underbrace{Ac}_A | d$$
$$\overbrace{Ba}^{\text{Do L.R.}} | \overbrace{d}^{\text{Do L.R.}}$$
$$A \rightarrow Ba | b$$
$$B \rightarrow \underbrace{Ac}_A | d$$
$$B \rightarrow BaC | bC | d$$
$$\boxed{\begin{array}{l} A \rightarrow Ba | b \\ B \rightarrow dB | bCB \\ B' \rightarrow a(B') | e \end{array}}$$

(14)

Left recursion

Indirect L.R

$$A \rightarrow B a | b$$

$$B \rightarrow A c | d$$

Direct L.R

$$A \rightarrow A \alpha | \beta$$

$$\textcircled{1} \quad A \rightarrow B a | b$$

neglect this

$$\textcircled{2} \quad B \rightarrow A \gamma | d$$

surface form with 2 derives.

$$B \rightarrow B a c | b c | d \Rightarrow$$

$$A \rightarrow B a | b$$

$$B \rightarrow d B' | b e B$$

$$B' \rightarrow a c B' | C$$

$$A \rightarrow B a | b$$

$$B \rightarrow d a c | d a c B' | b c a c B' | b e$$

$$B' \rightarrow a c B'$$

$$[A b c b b | A a b a | b a b | b b a] \xleftarrow{2} S | P A \leftarrow 2$$

$$[A b a | A a | b a | b] \xleftarrow{1} S | P A \leftarrow 1$$

$$[A b a | A a | b a | b] \xleftarrow{1} S | P A \leftarrow 1$$

Q. Eliminate Indirect left recursion,

$$S \rightarrow AaB \\ A \rightarrow Ae | Sd | e.$$

Soln:

$$S \rightarrow AaB \\ A \rightarrow Ac | Sd | e. \\ A \rightarrow AC | Acd | bd | e$$

remove variable

$$S \rightarrow AaB \\ A \rightarrow AC | Acd | bd | e. \\ A \rightarrow bdA' | eA' \\ A' \rightarrow CA' | adA' | e$$

direct left recursion

$$S \rightarrow AaB \\ A \rightarrow bdA' | eA' \\ A' \rightarrow CA' | adA' | e$$

Now remove ϵ ,

$$S \rightarrow AaB \\ A \rightarrow bdC | bdad | bdCA' | bdadA' | e \\ A' \rightarrow C | ad | CA' | adA'$$

Q. Convert the foll. in CNF,

(15)

$$S \rightarrow AA|0$$

$$A \rightarrow SS|1.$$

Soln:

to eliminate 1st left recursion:-

$$S \rightarrow AA|0$$

$$A \rightarrow \boxed{SS}|1$$

$$A \rightarrow AAS|OS|1$$

$$\overbrace{A \rightarrow OSAT|1A'}$$

$$A' \rightarrow ASA'|e$$

eliminate

$$S \rightarrow AA|0$$

$$A \rightarrow OSAT|OS|1A'|1$$

$$A' \rightarrow ASA'|AS$$

$$S \rightarrow \boxed{AA}|0$$

already in CNF replace it

$$S \rightarrow OSAT|OSA|1A'A|1A'|0$$

$$A \rightarrow OSAT|OS|1A|1$$

$$A' \rightarrow \boxed{ASA'|AS}$$

already in CNF replace it

$$A' \rightarrow OSAT|OSA|1A'SA|1SA|1$$

$$A' \rightarrow OSAT|OS|1A'S|1S.$$

$$S \rightarrow OSB'A | OSB | 1AxA | 1A | O$$

$$A \rightarrow OSA' | OS | 1A | 1.$$

$$A' \rightarrow OSA'SA | OSSA' | 1A5A | 1SA |$$

$$\Phi \quad \quad \quad OSALS | OSS | 1A1S | 1S.$$

A

* ** ~~for~~

Note @: To generate m length string required no. of production from CNIF grammar is

$$(2n+1)$$

Note @: To generate m length string tree required no. of production from CNIF grammar is m only.

CNF

$$S \rightarrow AX | AB$$

$$X \rightarrow SB.$$

$$A \rightarrow a$$

$$B \rightarrow b$$

$$[ab] \quad [aab]$$

1234567.

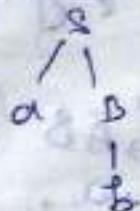
→ $a21 | 1A21 | 1A220 | 1121A20 | 1A21 | 21 | 21Ab | 220 | 21A20$

GNF

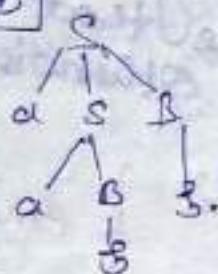
$$S \rightarrow aSB \mid aB$$

$$B \rightarrow B.$$

$aabb$



$aabb$



(16)

Decision properties of CFG @ CFL \Rightarrow

$CFL \Rightarrow CFL \Rightarrow PDA$

The following problems
are decidable for CFG:-

① Emptiness problem

② Finiteness problem

③ Membership problem

④ Equivalence problem is undecidable for
CFL test lib

is impossible to verify given two contexts
for grammars are generating same
language. @ not in all cases.

Emptiness problem \Rightarrow Grammer generates

empty lang. @ not.

Algorithm

→ Eliminate useless variables from the grammar
 If the starting symbol also includes a useless variable then grammar generates empty language. Otherwise non-empty.

Ex: $\begin{array}{l} S \rightarrow AB \\ A \rightarrow a \\ B \rightarrow aB \end{array}$

$\left. \begin{array}{l} S \rightarrow AB \\ A \rightarrow a \\ B \rightarrow aB \end{array} \right\}$ empty.

$\begin{array}{l} S \rightarrow AB \\ A \rightarrow a \\ B \rightarrow aB | C \\ C \rightarrow a^0 \end{array}$

$\left. \begin{array}{l} S \rightarrow AB \\ A \rightarrow a \\ B \rightarrow aB | C \\ C \rightarrow a^0 \end{array} \right\}$ non empty

Finiteness problem \Rightarrow

Verifying the grammar generates finite language
 or infinite language.

Algorithm:

Step 1 \Rightarrow Convert the grammar into CNF form.

Step 2 \Rightarrow Construct the CNF graph.

Step 3 \Rightarrow If the CNF graph contains any cycle then grammar generates infinite lang. otherwise finite lang.

generate infinite lang.

~~Q.~~ $\begin{array}{l} S \rightarrow AB \\ A \rightarrow B \text{ } C | a \\ B \rightarrow CC | b \\ C \rightarrow a \end{array}$

Verify the following grammar generates finite lang or infinite lang.

and also find rank of each variable.

(17)

S^m: $S \rightarrow AB$
 $A \rightarrow BC|a$
 $B \rightarrow CC|b$
 $C \rightarrow a$

} Simplified
already
2 already in
CNF.



Rank possible only for finite.

Rank \rightarrow if the language generated by grammar is finite then we can define rank of each variable.

\rightarrow Rank of a variable v is length of the longest path starting from that variable in the CNF graph.

\rightarrow Rank of a variable v is 'n' given that variable can generate max. of 2^n length string.

Q

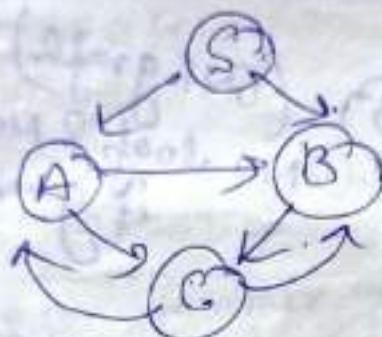
epsilon $\leftarrow \Sigma$

epsilon $\leftarrow Y$

Q. Verify tree foll. grammar generates finite or infinite lang.

$$\begin{aligned}S &\rightarrow AB \\A &\rightarrow BC/a \\B &\rightarrow CC/b \\C &\rightarrow a/AB.\end{aligned}$$

Soln:



2) Infinite lang.
rank not defined.

Membership problem:

- means checking whether string x is generated from the grammar or not.
- its concept is also known as parsing.

Algorithm: — CYK algorithm exist

for membership problem,

$C \rightarrow \text{Cookie}$

$Y \rightarrow \text{Younger}$

$k \rightarrow k$ asami.

(18)

Time complexity of this
 algo is $O(n^3)$

where n is
length of the
input.

Note:

To apply CYK algo, given grammar should be
in Chomsky Normal Form.

Q. check whether the string baaba is member
of the following grammar (not) S

①

$$S \rightarrow AB \mid BC$$

$$A \rightarrow BA \mid a$$

$$B \rightarrow CC \mid b$$

$$C \rightarrow AB \mid a$$

Soln:

the last block contain starting symbols
given string is member of the given
grammar.

Q. which of the following string is member
of the given grammar

1 ✓ (1) ✓ (2) ✓ (3) ✓ (4) ✓ (5) ✓ (6) ✓ (7) ✓ (8)

b ✓
ba X
baa (g) aba
baab
aa
ab

Be
CC
baab.
AB
BA B.

Q. How many substrings of the string
babab are members of
the given grammar.

5.

Q. How many substrings of the given string
babab are not member of the
given grammar.

11 $\Rightarrow (10 + \epsilon)$

Q. How many different substrings of the
given string babab members of
the given grammar.

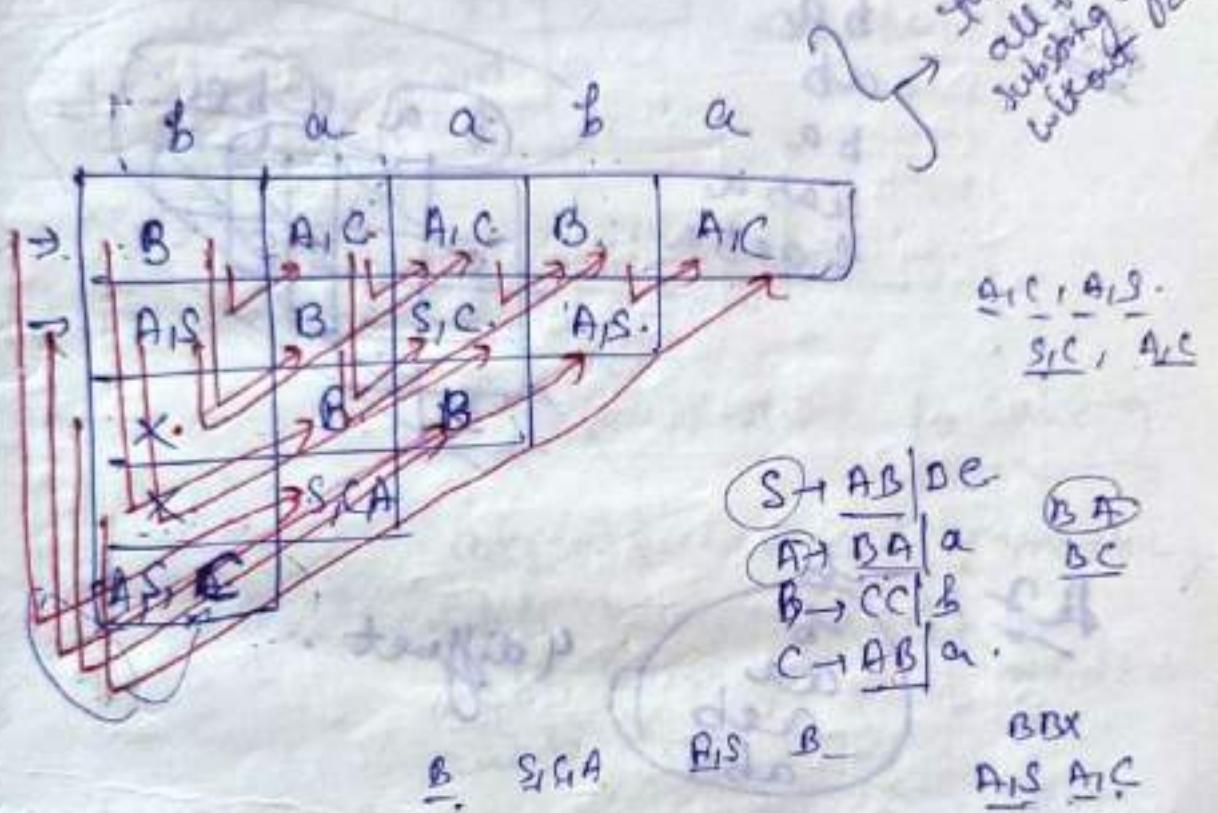
4

Q. How many substrings of the given string
babab are generated from
variable B only.

5 (having B ~~as~~ in box)

Q. How many diff. substrings of the given string $baba$ generated from variable B only? (19)

4 Ans



If we see in the diagonal, if we check for abc
then it is member of power.

$$\begin{array}{ll}
 S \xrightarrow{} AB \mid DE & \\
 A \xrightarrow{} BA \mid a & \text{B } \underline{A} \\
 B \xrightarrow{} CC \mid b & \text{C } \underline{C} \\
 C \xrightarrow{} AB \mid a & \\
 \hline
 \end{array}$$

$\underline{B}, \underline{S}, \underline{A}$

$$\begin{array}{ll}
 \underline{A}, \underline{S} & \underline{B} \\
 \hline
 \end{array}$$

$\underline{B}, \underline{D}, \underline{X}$

$$\begin{array}{ll}
 \underline{A}, \underline{S} & \underline{A}, \underline{C} \\
 \hline
 \end{array}$$

$\underline{B}, \underline{C}$

$$\begin{array}{ll}
 \underline{B}, \underline{C} & \underline{B} \\
 \hline
 \end{array}$$

$\underline{A}, \underline{C}, \underline{B}$

$$\begin{array}{ll}
 \underline{B} & \underline{S}, \underline{C}, \underline{A} \\
 \hline
 \end{array}$$

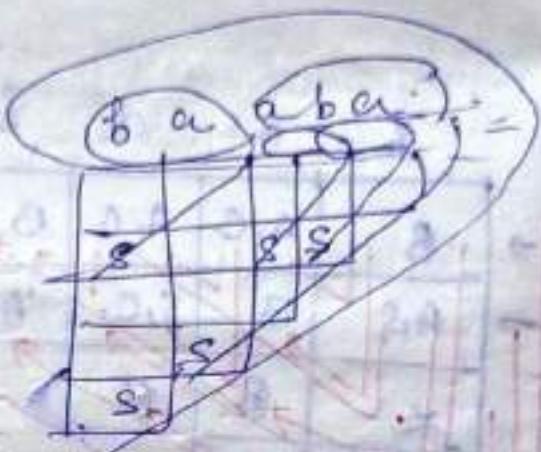
$\underline{A}, \underline{S}, \underline{B}$

(ba)

* each box represent a substring (without S).
 and those boxes which contain
 (S) are the member of grammar.

A 15:

b a
 ab
 ba
 aab a
 b aab a



A 2:

b
 b
 aa
 aeb
 abg

4 offset.

Q. check whether the string aabb
 is a valid member of GLL grammar

or not

$$S \rightarrow AB$$

$$A \rightarrow BB | a$$

$$B \rightarrow AD | B$$

(20)

a	a	b	b	b
A	A	B	B	B
X	S, B	A	A	
B, S	A	S, B		
A	S, B			
S, B				

→

aabb member

aa X

a X.

16 substring

ab ✓

5 member

bb ✓

11 not member

akbb ✓

aab X

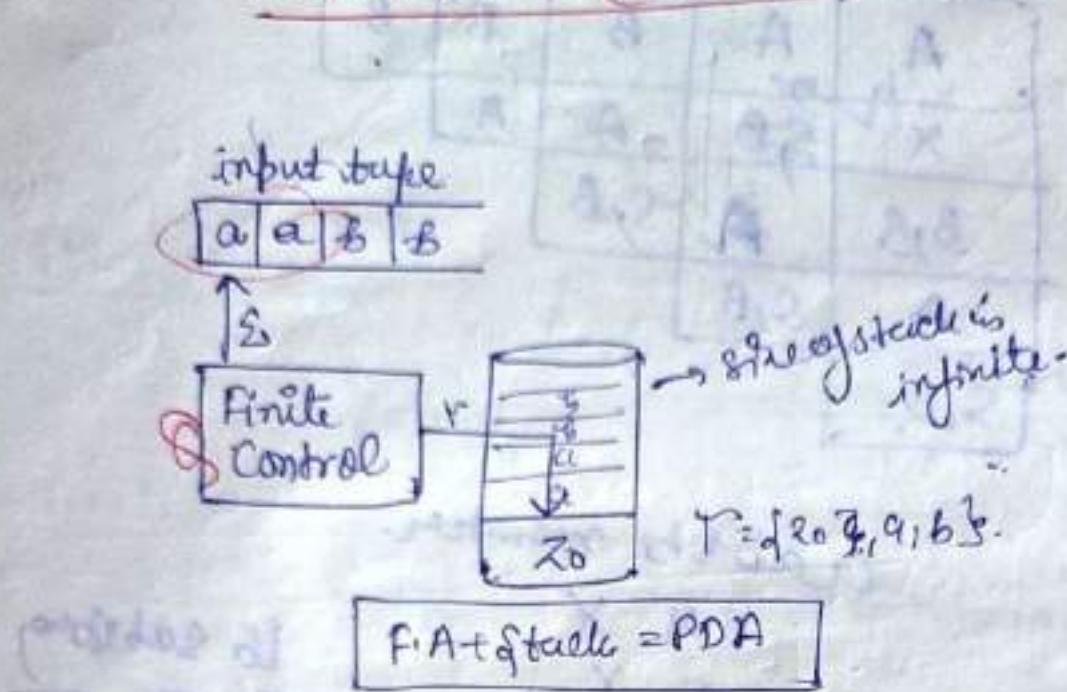
ab
bb
akbb
aabbb
aab

all 5 are
different
members

6 generated by variable A only,

a, bb, bbb, akbb, aabb } 4 different

Pushdown Automata



$$PDA = (Q, \Sigma, \delta, q_0, F, z_0, \Gamma)$$

Q : finite no. of states

Σ : input alphabet

q_0 : initial state

F : set of final states

z_0 = initial stack element

Γ = stack alphabet

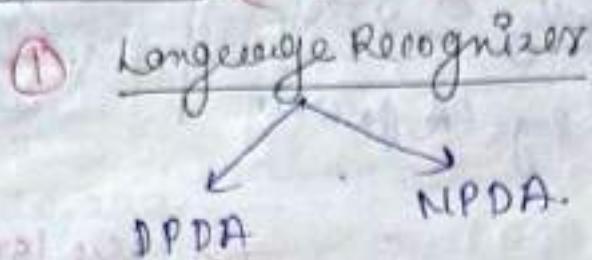
δ = transition function.

Operations → are push
sum set of symbols → are pop
full on stack → pop
two set of symbols → stack
or set of symbols → stack

$$\delta : Q \times \Sigma \cup \{e\} \times \Gamma \rightarrow Q \times \Gamma^*$$

notations

- (1)  transition diagram
- (2)  transitions

Types of PDA (only one type)Acceptance method :

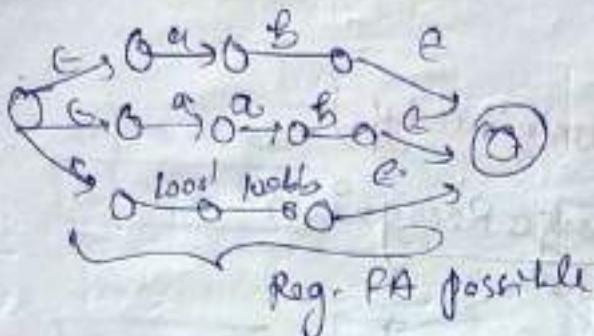
(1) Empty stack \Rightarrow while reading the complete string from left to right end of the string stack of the PDA is empty, then given string is accepted, irrelevant about no. of final states.

(2) Final state \Rightarrow while reading the complete string from left to right, end of the string PDA enters into final state the given string is accepted and irrelevant about stack symbols.

\rightarrow PDA is used to design parser of Compiler.

$$L = \{a^n b^n \mid n \leq 100\}.$$

$\{ab, aabb, aaabb, \dots\}$



$$L = \{a^n b^n \mid n \geq 1\}$$

$\{ab, aabb, \dots\}$



Non-Reg

~~DFA~~

we are going
for PDA.

- Finite automata can recognize the languages which require finite memory but finite automata fails to recognize the languages which require infinite memory.
- The finite memory is possible by storing the symbols on finite states.
- To accept the language which requires infinite memory finite automata with additional storage i.e. pushdown automata is used.
- The size of the stack in PDA is infinite.
- To accept the language $a^n b^n \mid n \geq 1$ by finite automata, total no. of a's should be stored on the states but here no. of ab are infinite hence finite automat not possible with infinite states. Hence for the language $\{a^n b^n \mid n \geq 1\}$ finite automata not possible. Hence, Non-regular.

~~Q.~~ Construct a finite automata for the language that accepts all strings of ab & bbs where each string contains equal occurrences of ab and ba string. (22)

bm: ab and ba

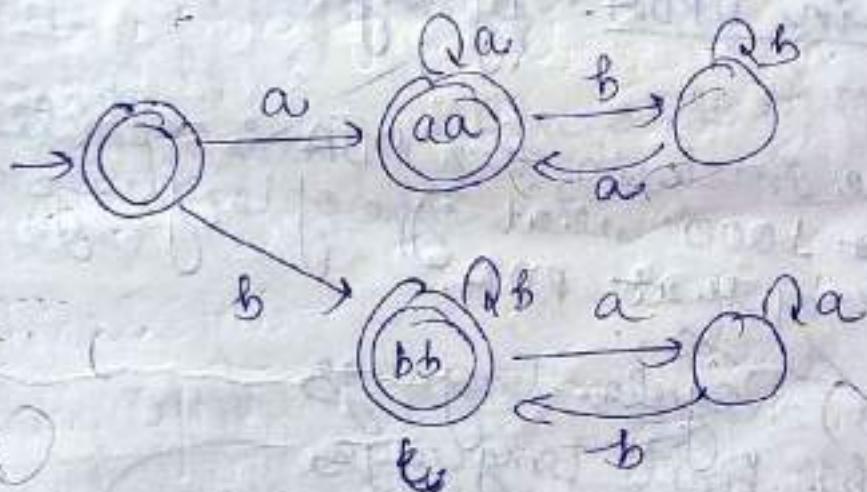
"FA possible here?"

$$L = \{ \text{ } \}$$

a, aa, aaa, \dots
b, bb, bbb, \dots
abba

accepting

baab



~~GATE~~ Q. n_1 no. of languages are accepted by empty stack mechanism of PDA and n_2 no. of languages are accepted by final state mechanism of PDA then which of the following relation is true,

- ~~(A) $n_1 = n_2$~~
- (B) $n_1 < n_2$
- (C) $n_1 > n_2$
- (D) we can't say.

Note: No. of language accepted by using empty stack mechanism and no. of language accepted by using final state mechanism are same.
(in PDA).

Q. If No. of language accepted by final state mechanism is more compare to empty stack mechanism in DPDA. By default, PDA means NPDA.

Q. If size of stack in the PDA is restricted to 1000 what type of languages are accepted by that PDA.

- (A) Regular language
- (B) Finite language
- (C) CFL but not Regular
- (D) Regular but not CFL.

localite
PDS
unit
stack
infinite
lang.
level.



Here it is not CFL
so, we can't use
the stack.
so it
accept Reg
Lang. not
full lang. only.

- # In any PDA, the stack is restricted \oplus not used to language accepted by that PDA is Regular language only.

Drawback of PDA

- PDA fails to recognize the languages which require more than one memory element.

$$L = \{a^n b^n \mid n \geq 1\}$$

aabbcc
↑↑↑

- The languages accepted by PDA are known as CFLs.
 → The language for which PDA not possible are known as non-CFLs.

Construction :-

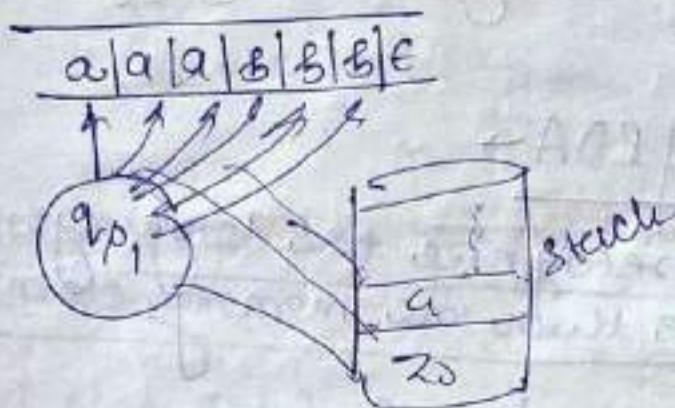
- Q. Construct the push down automat for the lang.

$$L = \{a^n b^n \mid n \geq 1\}$$

BFS:

$$L = \{a^n b^n \mid n \geq 1\}$$

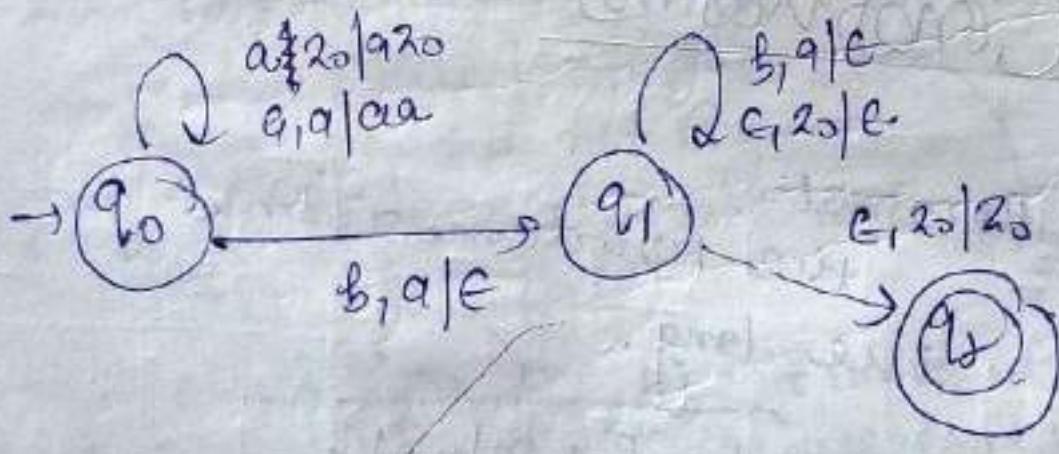
state change
does not depend
on operation
depends on
input
symbol.



$$\begin{aligned} \delta(q_0, a, z_0) &= \delta(q_0, a, z_0) \\ \delta(q_0, a, a) &= \delta(q_0, a, a) \end{aligned} \quad \left. \begin{array}{l} \text{Push all } a \text{ onto stack} \\ \text{do onto stack} \end{array} \right\}$$

$$\begin{aligned} \delta(q_0, b, a) &= \delta(q_1, \epsilon) \\ \delta(q_1, b, a) &= \delta(q_1, \epsilon) \end{aligned} \quad \left. \begin{array}{l} \text{Pop all } b \text{ off stack} \\ \text{ab of stack} \end{array} \right\}$$

$$\delta(a_1, \epsilon, z_0) = (q_1, z_0) \text{ or } (q_1, \epsilon)$$



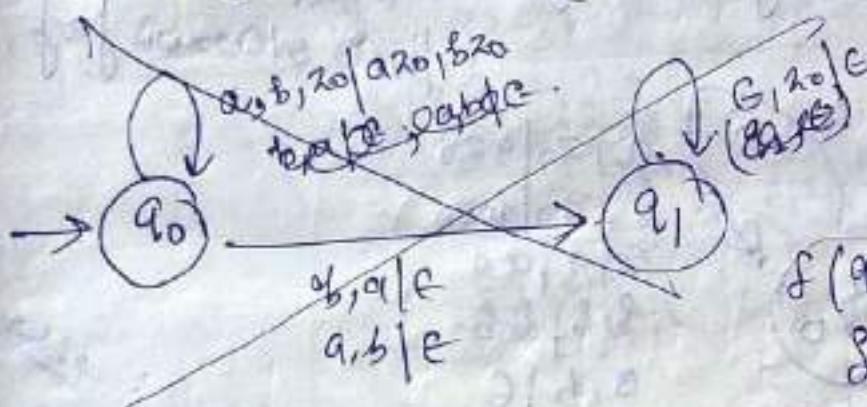
(If we don't change the state in b it accept abab type string which is wrong.)

Q. Construct PDA that accepts all strings of a's and b's where no. of a's & no. of b's are equal in every string. (24)

Soln:

$$\delta(q_0, a, z_0) = (q_0, az_0)$$

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



abab

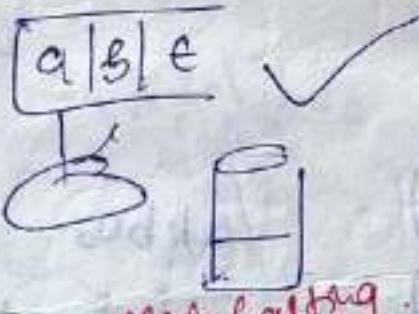


aabb

aa

$$\delta(q_0, b, z_0) = (q_0, bz_0)$$

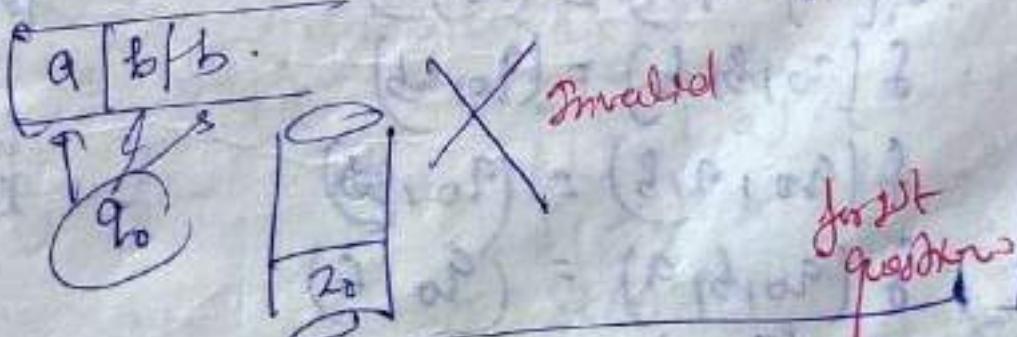
$$\delta(q_0, a, b) = (q_1, \epsilon).$$



valid halting



Invalid



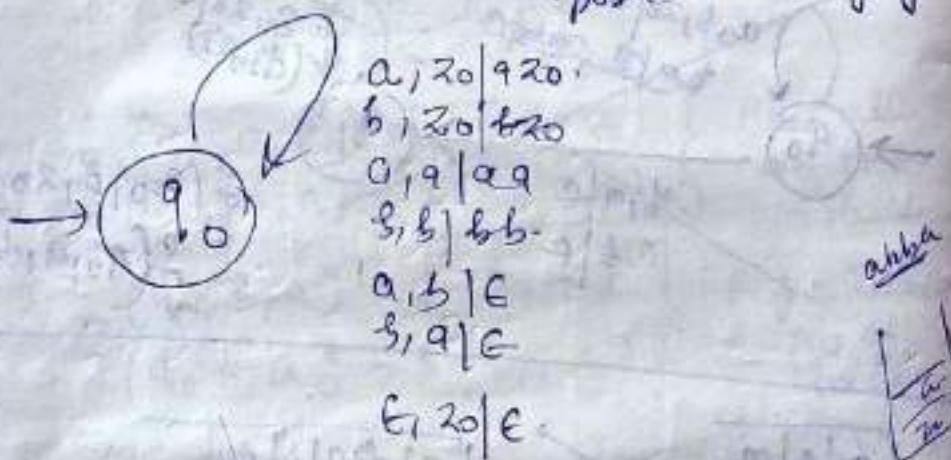
Invalid

for 2nd question

$$L = \left\{ x \mid n \in \{a, b\}^* \right. \\ \left. m_a(n) = m_b(n) \right\}$$

$L = \{ab, ba, abba, abab, babab, \underline{\underline{ahba}}, \dots\}$

there is some symbol on stack &
if it is
push otherwise pop.



$$\delta(q_0, a, z_0) = (q_0, az_0)$$

ahba

$$\delta(q_0, b, z_0) = (q_0, bz_0)$$

$$\delta(z_0, a, a) = (q_0, aa)$$

$$\delta(z_0, b, b) = (q_0, bb)$$

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

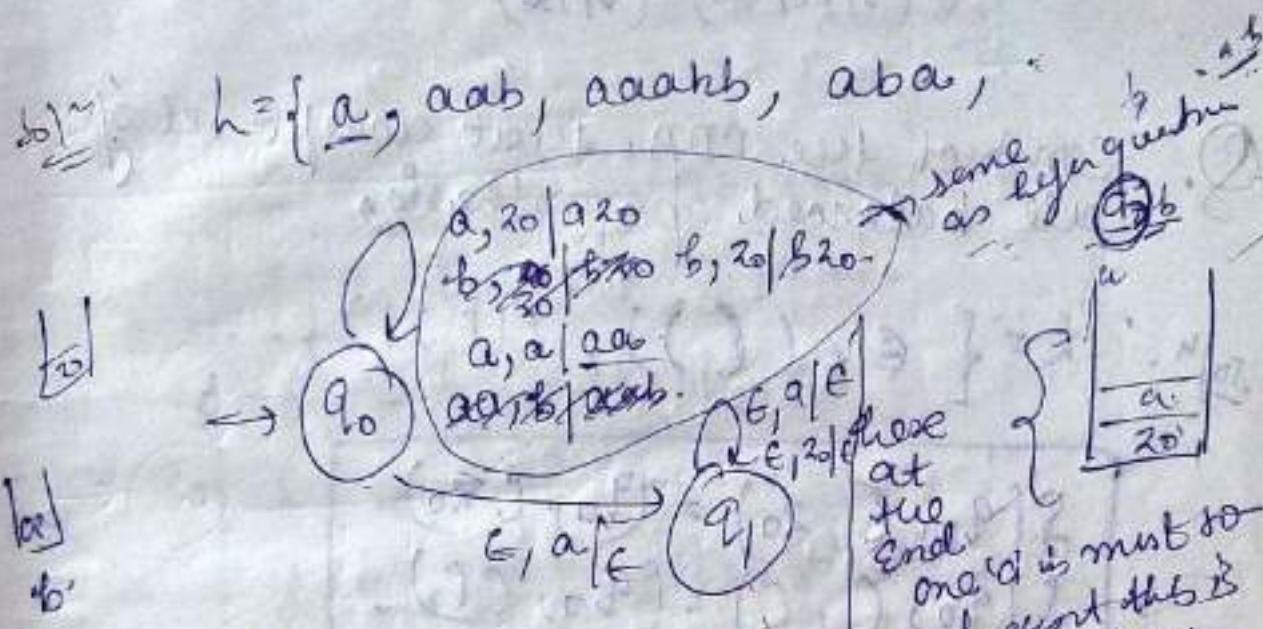
$$\delta(q_0, b, a) = (q_0, ba)$$

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

Q. Construct the PDA,

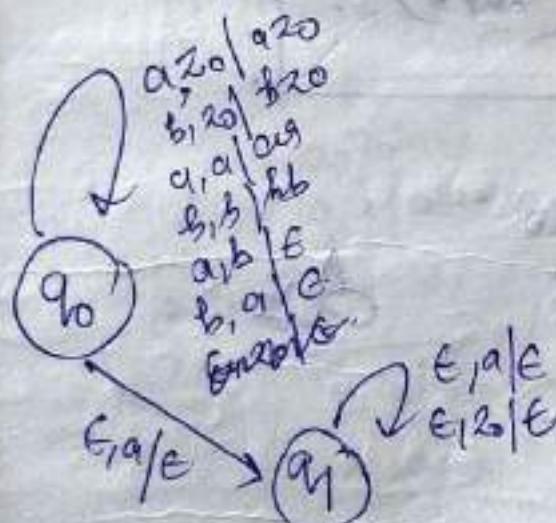
$$L = \{ u \mid n_a(u) \geq n_b(u) \}$$

Ans by 202



babaaabaa, ε.

here at the end one 'a' is must so present this is the logic.



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

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

$$\delta(q_0, a, a) = (q_0, a/a)$$

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

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

$$\delta(q_0, b, ε) = (q_0, b/ε)$$

$$\delta(q_0, a, ε) = (q_0, a/ε)$$

$$\delta(q_0, ε, a) = (q_0, ε/a)$$

$$\delta(q_0, ε, b) = (q_0, ε/b)$$

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

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

$$\delta(q_1, q_1, a) = (q_1, a/a)$$

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

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

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

Q. Construct the PDA. that accept set of all balanced parenthesis,

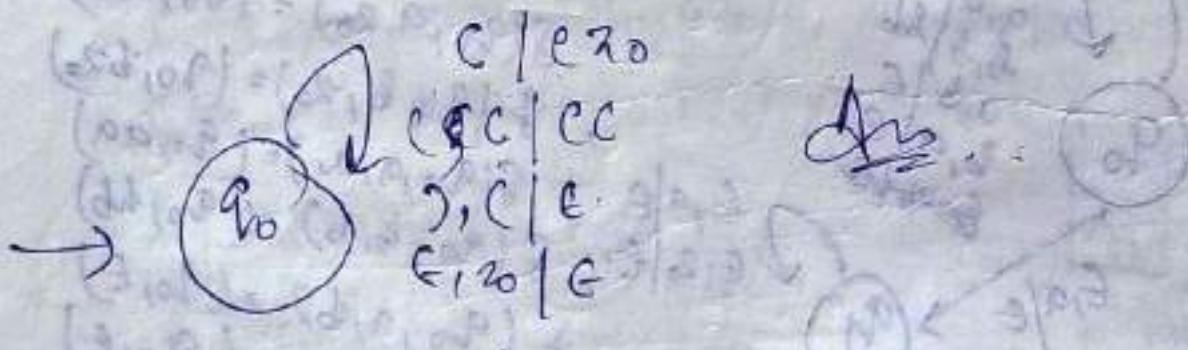
Soln. $L = \{ \epsilon, ((())), ((())) \}$

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

$$\delta(q_0, \epsilon, C) = (q_0, CC)$$

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

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



Q. construct the PDA, for the language

$$L = \{ a^n b^n c^m \mid n, m \geq 1 \}$$

(26)

$h = \{ abC, aabbC, aabbcc, \dots \}$

No. of ab equal to no. of bs after start
any no. of cs.

$$\delta(q_0, a, z_0) = \delta(q_0, q_{20})$$

$$\delta(q_0, a, a) = \delta(q_0, q_a).$$

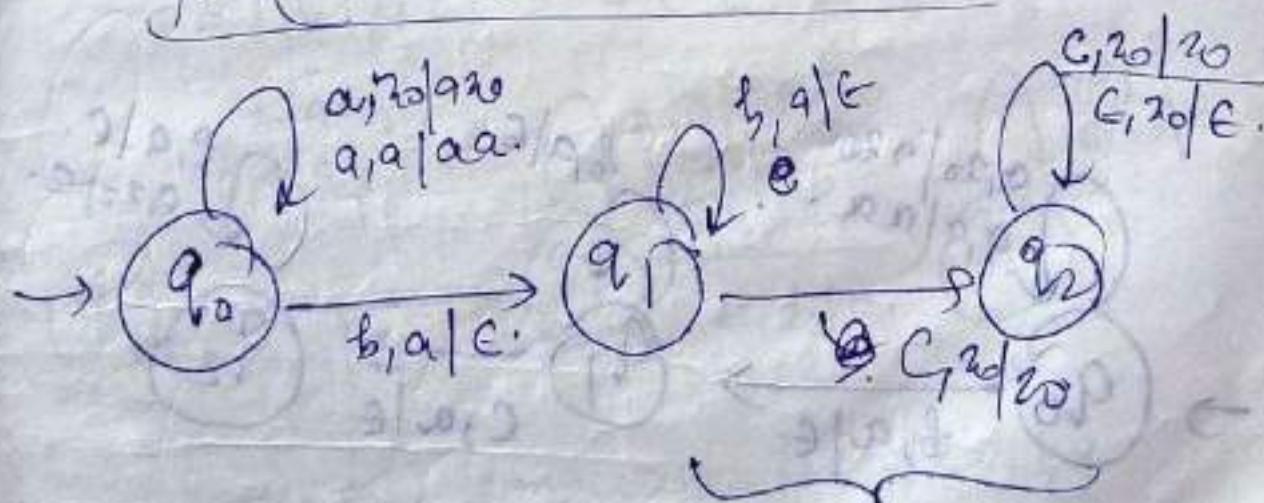
$$\delta(q_0, b, q) = \delta(q_1, \epsilon)$$

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

$$\delta(q_1, c, z_0) = (q_2, z_0)$$

$$\delta(q_2, c, z_0) = (q_2, z_0)$$

$$\delta(q_2, \epsilon, z_0) = (q_2, \epsilon)$$



change the

~~transitions because we
need at least one
'c' necessarily.~~

Q. Construct the PDA,

$$L = \{a^m b^m c^n \mid m, n \geq 1\}$$

Sol:

$$L = \{aabbc, aaabbc, \dots\}$$

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

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

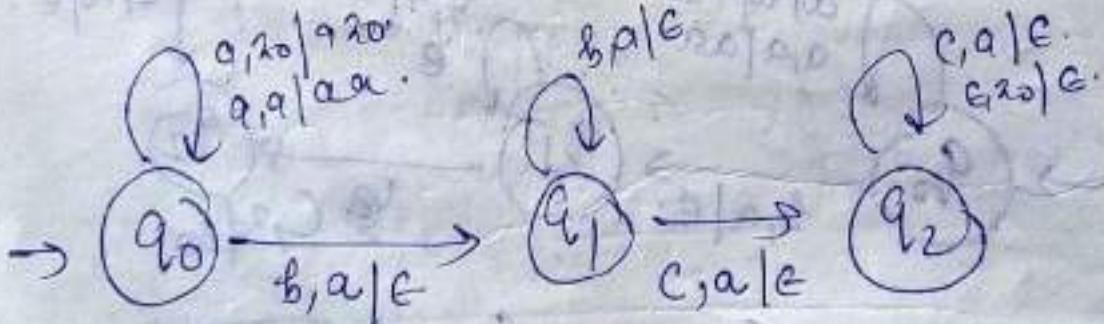
$$\delta(q_0, b, a) = \delta(a_1, \epsilon).$$

$$\delta(q_1, b, a) = \delta(a_1, \epsilon)$$

$$\delta(q_1, c, a) = \delta(q_2, \epsilon)$$

$$\delta(q_2, c, a) = \delta(q_2, \epsilon).$$

$$\delta(q_2, \epsilon, z_0) = \delta(q_2, \epsilon).$$



Q. Construct the PDA,

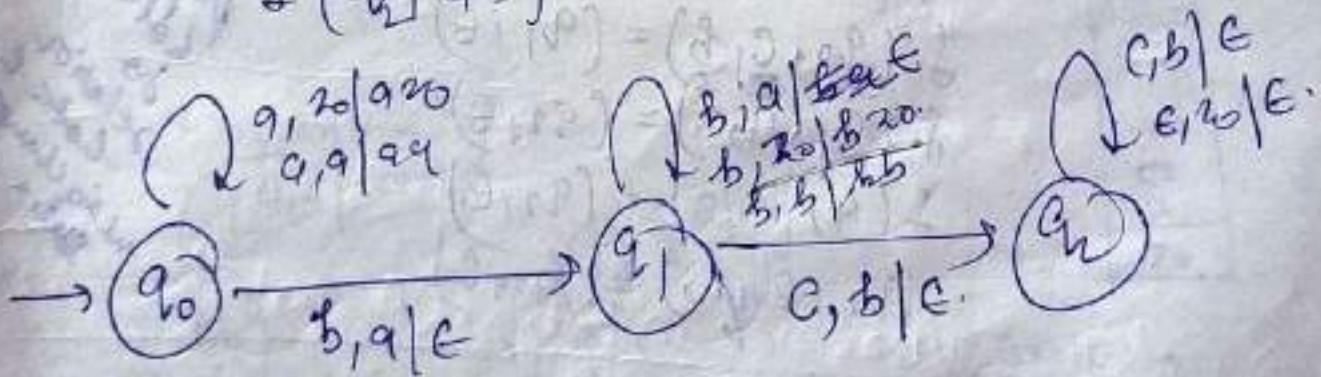
$$L = \{a^m b^{m+n} c^n \mid m, n \geq 1\}$$

• follow me

$L = \{ \text{any no. of } a's \text{ followed by } b's \text{ and } c's \text{ such that no of } a's \text{ & } b's \text{ are equal and } b's \text{ & } c's \text{ are equal} \}$

(27)
amgmb
the poly poly

$$\begin{aligned}
 f(a_0, a, z_0) &= f(a_0, a_0). \\
 f(a_0, a, a) &= d(a_0, aa). \\
 f(a_0, b, a) &= f(a_1, \cancel{aa} \cancel{bb}) \epsilon \\
 f(a_0, b, a) &= f(a_1, \cancel{aa} \cancel{bb}) \epsilon \\
 f(a_1, b, a) &= f(a_1, \cancel{bb}) \epsilon \\
 \cancel{f(a_1, \epsilon, z_0)} &= \cancel{f(a_1, \epsilon z_0)} \times \\
 f(a_1, b, z_0) &= f(a_1, b z_0) \\
 f(a_1, b, \cancel{z_0}) &= f(a_1, bb) \\
 f(a_1, c, b) &= f(a_2, \cancel{bb}) \epsilon \\
 f(a_1, c, b) &= f(a_2, \epsilon) \\
 \end{aligned}$$



Q. Identify language accepted by
this PDA,

$$\delta(q_0, q, z_0) = (q_0, qz_0)$$

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

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

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

$$\delta(q_1, \underline{\epsilon}, q) = (q_1, \epsilon)$$

$$\delta(q_1, \underline{z_0}) = (q_1, \epsilon)$$

$$\begin{bmatrix} a \\ a \\ b \\ a \end{bmatrix}$$

$$h = \{a^n b^m c^m d^n \mid n \geq m\}$$

An

Q. Identify $\delta(q_0, q, z_0) = \delta(q_0, qz_0)$

$$\delta(q_0, q, a) = (q_0, qa)$$

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

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

$$\delta(q_0, c, \underline{b}) = (q_1, \epsilon)$$

$$\delta(q_1, \underline{c}, a) = (q_2, \epsilon)$$

$$\delta(q_2, d, a) = (q_2, \epsilon)$$

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

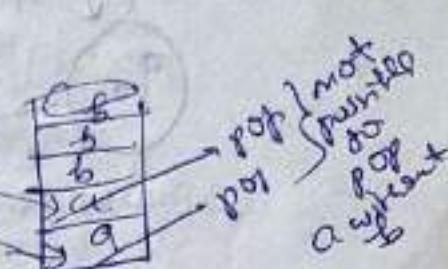
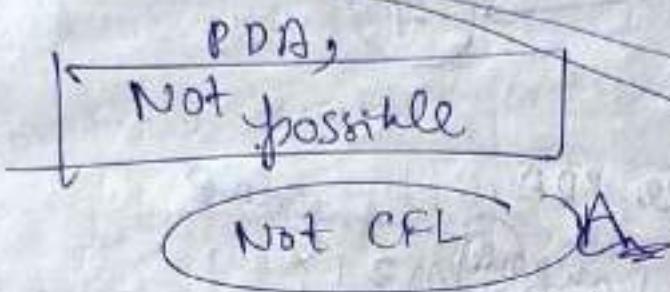
poly
ketan
do
reduksi
ketan
meng
pas

$$h = \{a^m b^m c^m d^n \mid m, n \geq 1\}$$

Q. construct the PDA for the lang
 $L = \{a^n b^m c^n d^m \mid m, n \geq 1\}$

Soln:

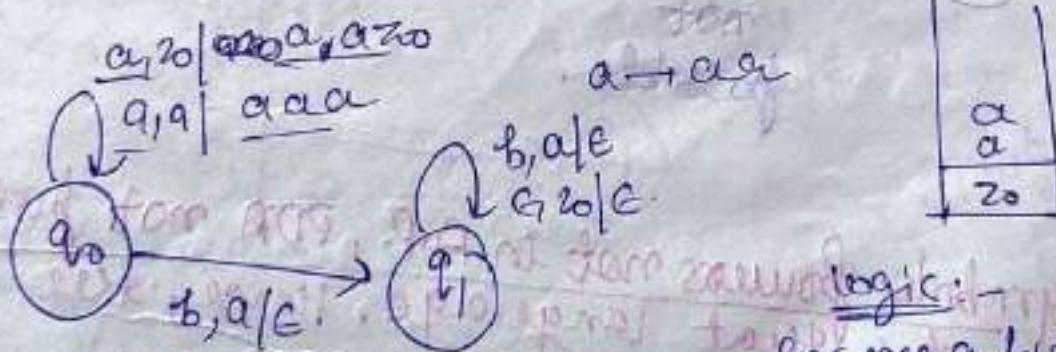
~~over aabbccdd.~~



Q. construct the PDA for the lang,

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

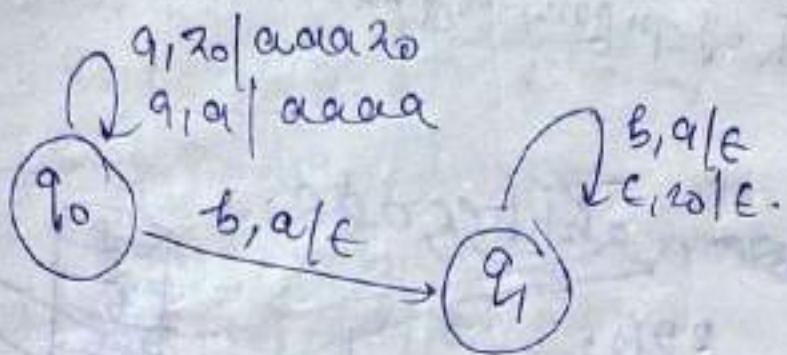
Soln: $L = \{abb, aabb, \dots\}$



for more push too 'a' here

Q. construct the PDA for the lang,

$$L = \{a^n b^{3n} \mid n \geq 1\}$$



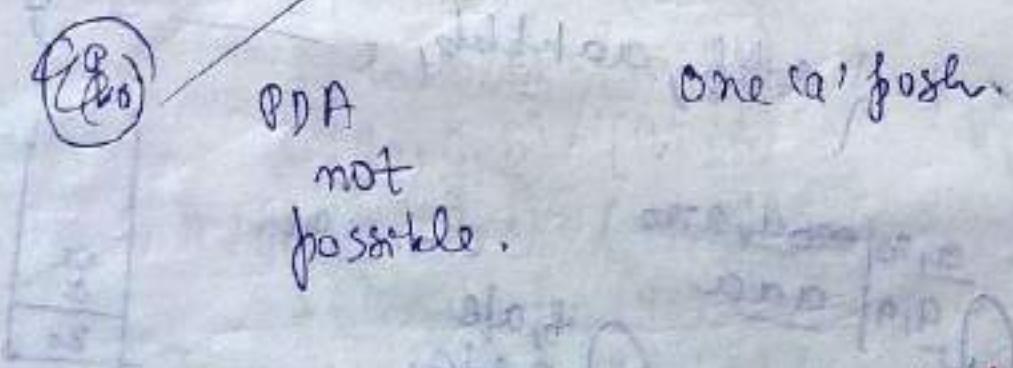
Q. Construct state PDA,

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

$n=1$

$L = \{ab, \underline{aabbb}, \underline{aaa}, \underline{\underline{bbb}}\}$

3



- Symbols powers not in A.P., PDA not possible for that language. Hence, the following languages are non-CFL's.

Q. $L = \{a^n b^{2n} \mid n \geq 1\}$

29

Not possible

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

Not possible.

~~Q.~~ construct the PDA for the language,

~~for~~ $L = \{w c w^R \mid w \in \{a, b\}^*\}$

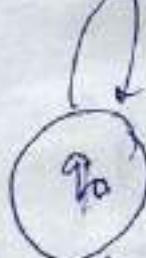
SOM: tab ab @ baba

b	2
a	
b	
a	
w	

$\delta(q_0, a, a) = q_0$ (bypass)

we can't
push &
call pop.

a, b	ab
b, a	ba
a, a	aa
b, b	bb



b, b	ε
a, a	ε
c, c	ε

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

$$\begin{aligned}\delta(q_0, a, a) &= (q_0, a, a) \\ \delta(q_0, b, b) &= (q_0, b, b) \\ \delta(q_0, a, ab) &= (q_0, ab) \\ \delta(q_0, b, ba) &= (q_0, ba) \\ \delta(q_0, a, a) &= (q_0, a, a) \\ \delta(q_0, b, b) &= (q_0, b, b)\end{aligned}$$

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

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

S.F.M.

Q. Construct the PDA for $L = \{wwR \mid w \in \{a, b\}^*\}$

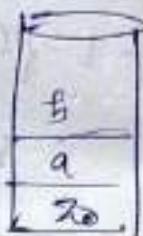
$$L = \{wwR \mid w \in \{a, b\}^*\}$$

ab ba

aa aa

bb bb

$$w = ab \\ aa$$



abab :



Now we know
that here this is
the separator
between w & wR .

PDA possible

✓ not DPDA possible but NPDA possible



$w \quad wR$

$$(q_0, a_1) = (q_0, a_2) / (q_0, \epsilon)$$

$$(q_1, b_1 b) = (q_1, b_2) / (q_1 \epsilon)$$

There are too possibilities we
can't determine that it is
a 3-alphabet of $w @ wR$
if of w we apply 1st operation
and if of wR we

go for 2nd operation

but we can't do this because

to how delhi salti h k kis ka effect
in turing machine can't see this

$$(q_1, \epsilon) =$$

$$(q_1, \epsilon) =$$

$$(q_1, \epsilon) =$$

to, how delhi salti h k kis ka effect

in turing machine can't see this

NPDAI -

For the following languages DPDA not possible but NPDA are possible.

$$\textcircled{1} \quad L = \{ a^i b^j c^k \mid i \in \mathbb{N}, j \in \mathbb{N}, k \in \mathbb{N} \}$$

$$\textcircled{2} \quad L = \{ a^i b^j c^k \mid i = j \text{ or } j = k \}$$

DPDA

NPDA

$aabbccc$
 \uparrow
 $aabbcccc$
 \uparrow

$$\left\{ a^n b^n c^m \right\} \cup \left\{ a^n b^m c^n \right\}$$

$\uparrow \text{push}$
 $(q_0, q_1, z) = (q_0, q_2, \text{by pass}) \text{ or } (q_0, z)$

$$\textcircled{3} \quad L = \left\{ a^n b^n \right\} \cup \left\{ a^n b^{2n} \right\} \cup \left\{ a^n b^{3n} \right\}$$

$aabb$ $aabbhb$ NPDA ✓
 \uparrow \uparrow $aabb$
 $atbzb$ \uparrow

⇒ In this lang. DPDA are possible ↗

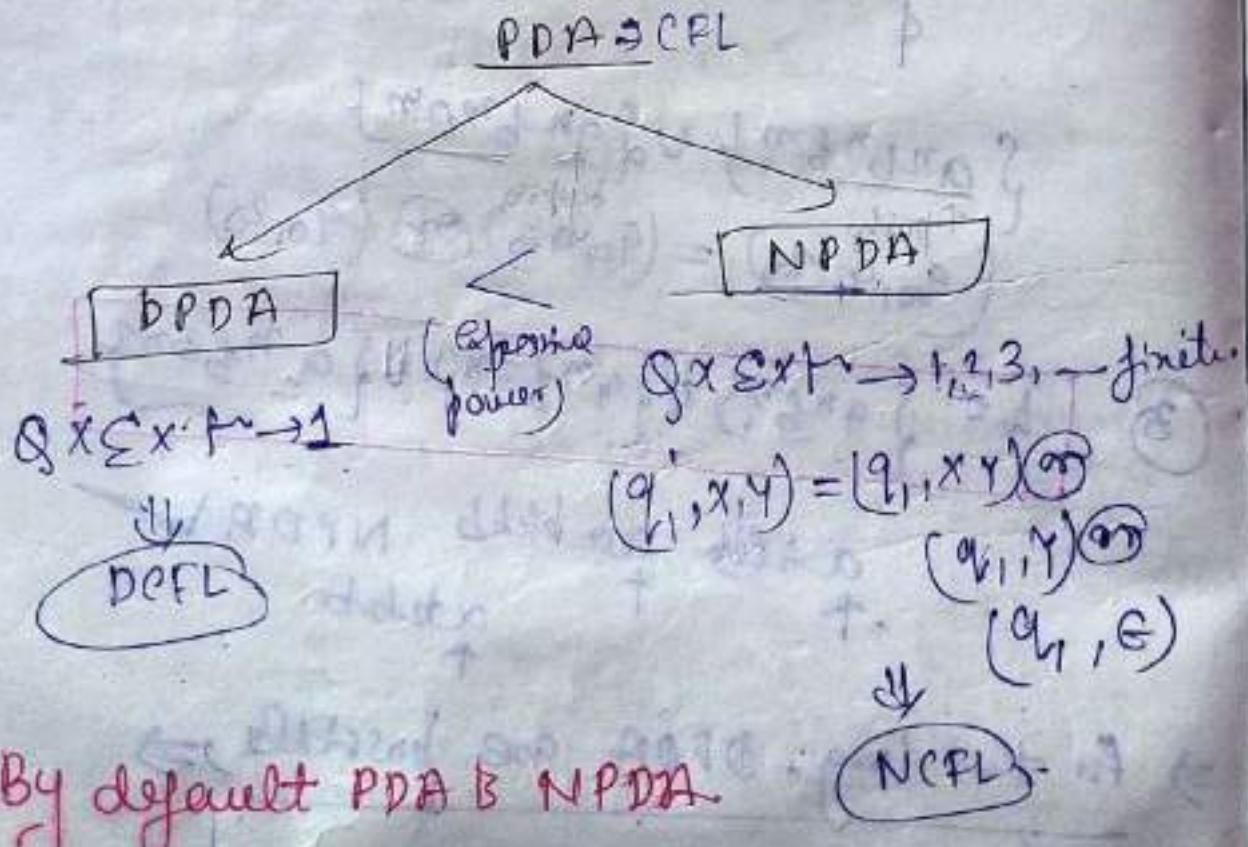
$L = \{a^m b^n c^m\} \oplus \{c^n b^m a^m\}$

bypas 'pop'

DPDA possible

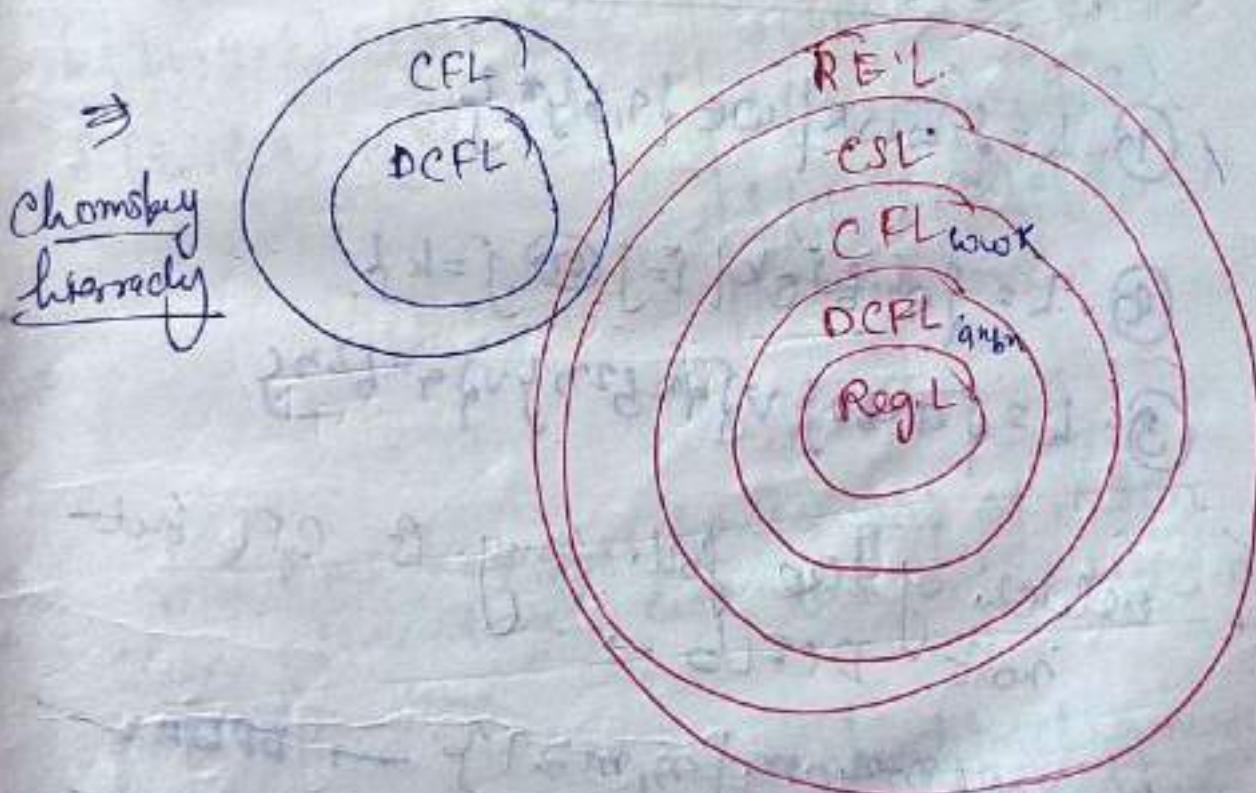
In DPDA given input state, given input symbol & given top of the stack symbols one transition exist.

In NPDPA given state, given input symbol & given top of the stack finite no. of transitions may be possible.



- Every DPDA is NPDA but all NPDA need not be DPDA.
- we cannot convert every NPDAs into DPDA (no algorithm exist). Hence, the expressive power of NPDA is more compare to DPDA.
- for the following languages NPDA only possible but not DPDA.

Ex: $L = \{a^n b^n\} \cup \{a^n b^m\} \cup \{a^m b^n\}$.



→ language accepted by DPDA are known as DCPLs.

→ lang. accepted by NPDAs are known as CFLs or NCFLs.

→ Every DCFL is CFL but every CFL need not be DCFL.

→ The language $a^m b^n$ is DCFL and CFL.

→ The language waw^R is CFL but not DCFL.

The following languages are also
CFL but not DCFLs →

$$①. L = \{ waw^R \mid w \in \{a, b\}^*\}$$

$$②. L = \{ a^i b^j c^k \mid i=j \text{ or } j=k \}$$

$$③. L = \{ a^n b^n \} \cup \{ a^n b^{2n} \} \cup \{ a^n b^{3n} \}$$

Q. Which of the foll. lang. is CFL but
not DCFLs →

(a) $L = \{ a^n b^{n+m} c^m \mid n, m \geq 1 \} \rightarrow \text{DPDA} \checkmark$

(b) $L = \{ w \# w^R \mid w \in \{0, 1\}^* \} \rightarrow \text{DPDA} \checkmark$

(c) $L = \{ a^n b^m c^n d^m \mid n, m \geq 1 \} \rightarrow \text{CFL not possible}$

Ans.

In NPDA how we will construct 32

$\omega\omega^R$,

ab^R

$$L = \{ \omega\omega^R \mid \omega \in \{a, b\}^* \}$$

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

$$\delta(q_0, a, z_0) = (q_0, az_0)$$

$$\delta(q_0, b, z_0) = (q_0, bz_0)$$

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

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

$$\begin{array}{l} (\text{---}, a, a) = (q_0, aa) \xrightarrow{\text{---}} \omega^R \\ \quad \quad \quad (q_0, \epsilon) \end{array}$$

$$(q_0, b, b) = (q_0, bb) \xrightarrow{\text{---}} (q_0, \epsilon)$$

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

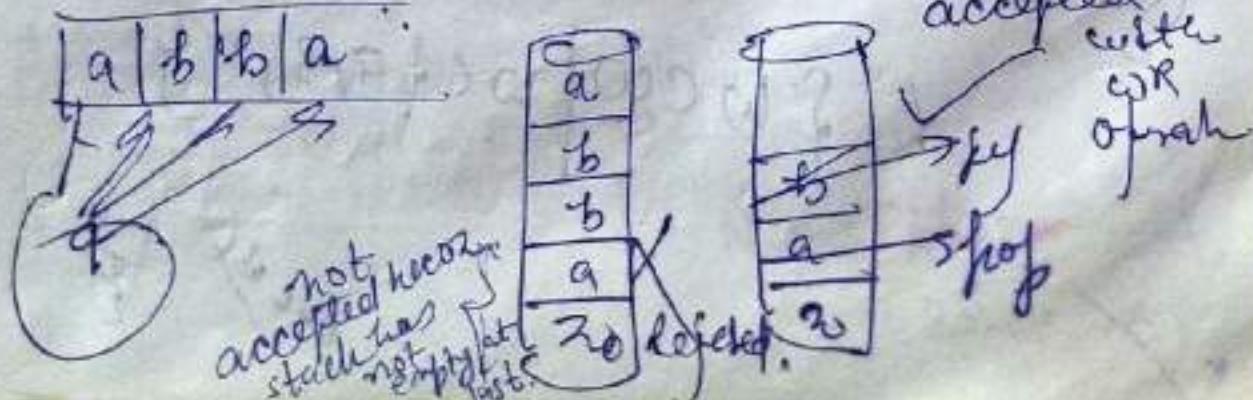
$$\delta(q_1, b, b) = (q_1, \epsilon)$$

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

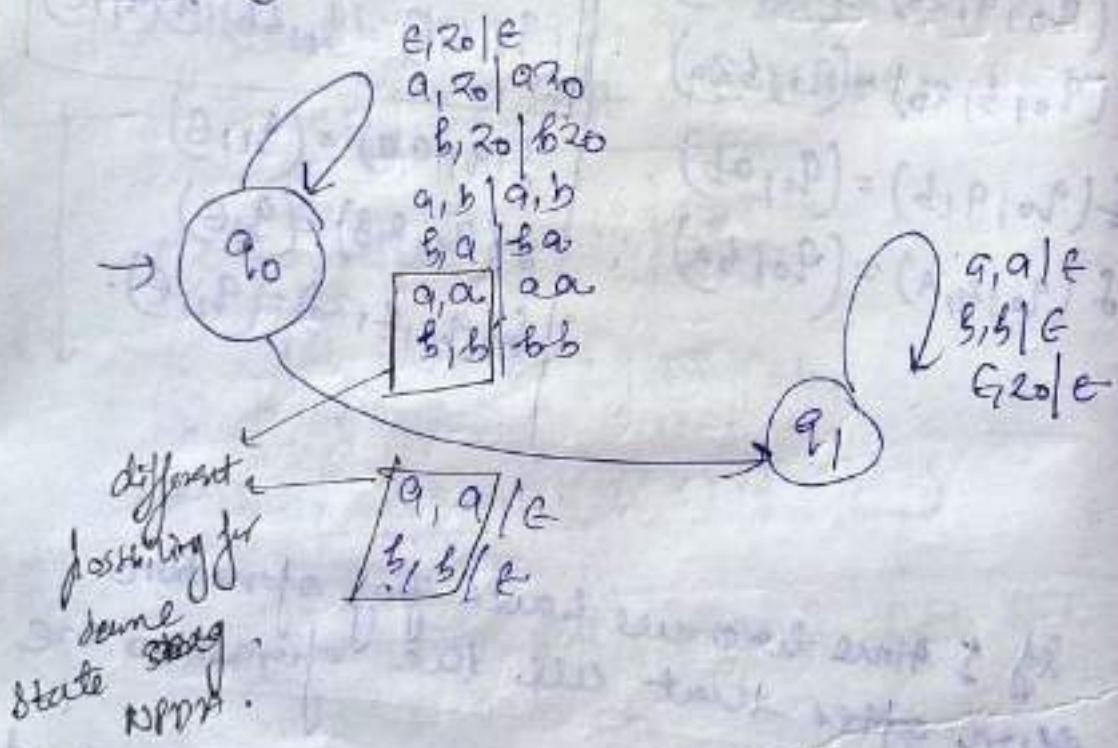
If I time here we have pop operation
done after that all the operations are
pop operation.

1st

backtracking method :-



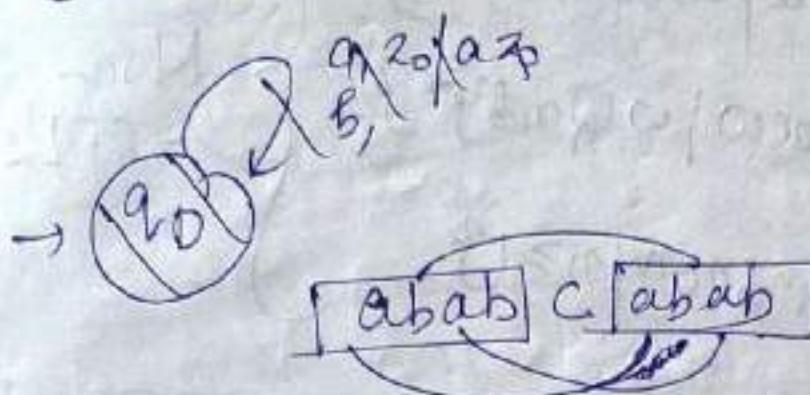
- In @N PDA having multiple possibility then priority will be given for 1st possibility and then 2nd followed by 3rd and so on. If these possibility string is accepted then it is valid string.
- By taking all possibilities also string is rejected then it is invalid string.



$$L = \{ w c w \mid w \in \{a, b\}^*\}$$

33

~~abcaab abcab~~



~~abcab
abcaab~~

b	ab
a	ab
b	ab
a	ab
z	ab

PDA

Not possible

Violating stack priority.

Q. Construct PDA for the language,
 $L = \{ww^R \mid w \in \{a, b\}^*\}$.

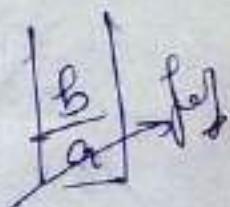
SOL:



ab ab

Not possible

not CFLs.



Note *** The following languages are Non-CFLs →

- ①. $L_1 = \{ w\omega | w \in \{a,b\}^*, \omega \in \{a,b\}^* \}$
 ②. $L_2 = \{ w\omega | w \in \{a,b\}^*, \omega \in \{a,b\}^* \}$. } Non-
 -CFL's.
 ③. $L_3 = \{ a^n b^n c^n | n \geq 1 \}$

→ But their complement is always CFL's

→ i.e., $\Sigma^* - L_1$ } CFL
 Q. Identify lang. accepted by foll PDA, $\Sigma^* - L_2$ } $\Sigma^* - L_3$

$$\delta(q_0, q_1, z_0) = (q_1, \underline{z_0})$$

$$\delta(q_0, \underline{q_1}, z_0) = (q_0, \underline{z_0})$$

$$\delta(q_1, q_1, z_0) = (q_1, \underline{z_0})$$

$$\delta(q_1, b, z_0) = (q_1, \underline{z_0}) \rightarrow \text{by passing}$$

$$\delta(q_2, q_1, z_0) = (q_0, \underline{z_0})$$

$$\delta(q_2, b, z_0) = (q_2, \underline{z_0})$$

$$\delta(q_0, \epsilon, z_0) = (q_0, \underline{z_0})$$



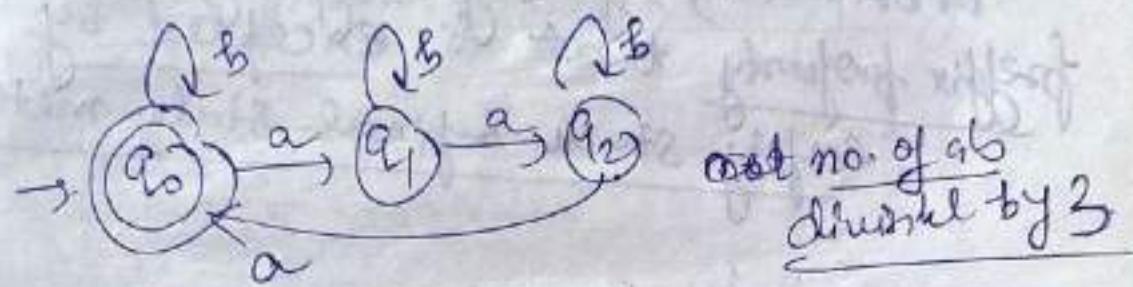
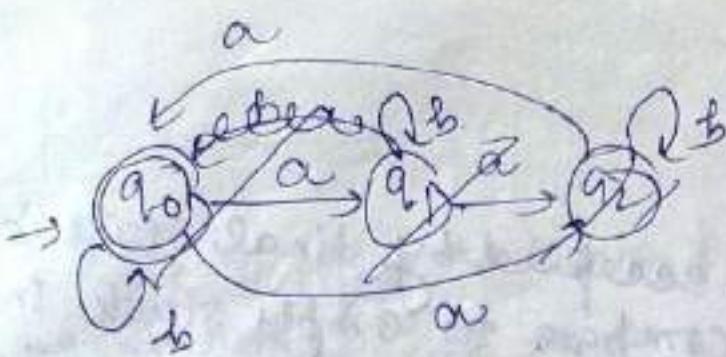
we are not pushing & not popping

so, stack is not used

so, lang. accepted

by it is Reg. lang.

34



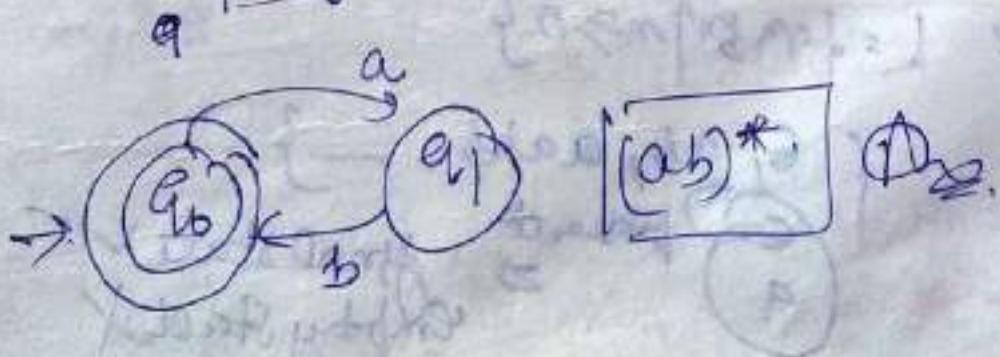
Q. Identify lang. accepted by following PDA,

$$\delta(q_0, a, z_0) = (q_1, \boxed{z_0}) \quad \text{by pushing}$$

$$\delta(q_1, b, z_0) = (q_0, \boxed{z_0})$$

$$\delta(q_0, \epsilon, z_0) = (q_1, \boxed{\epsilon, z_0}) \quad \text{not use empty stack method}$$

Reg. Lang accepting here.



$(ab)^*$

Dfa.

#

L = {languages}

No. of language accepted by final state is more compare to empty stack in DPDA. i.e., if any language having prefix property that is accepted by empty stack & final state method.

Ex: $L = \{a^n b^n \mid n \geq 1\}$

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

ϵ	ϵ	ϵ	}
a	aa	aaa	

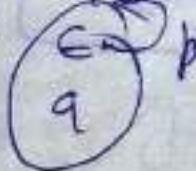
all prefix

ab
aab
aabb

Any lang. don't have prefix property
e.g., accepted by final state method only
but not by empty stack.

Ex: $L = \{a^n b^n \mid n \geq 0\}$

$\{ \epsilon, ab, aabb, \dots \}$



 \Rightarrow final state ✓
 empty stack ✗

Prefix property \rightarrow

a lang. is said to have prefix property if no proper prefix of any string of the language present in the language.

$$L = \{a^n b^n \mid n \geq 1\}.$$

ab	aabb	aaa
e	e	
a	a	

having
prefix property

$$L = \{a^n b^n \mid n \geq 0\} \rightarrow \text{not having prefix property}$$

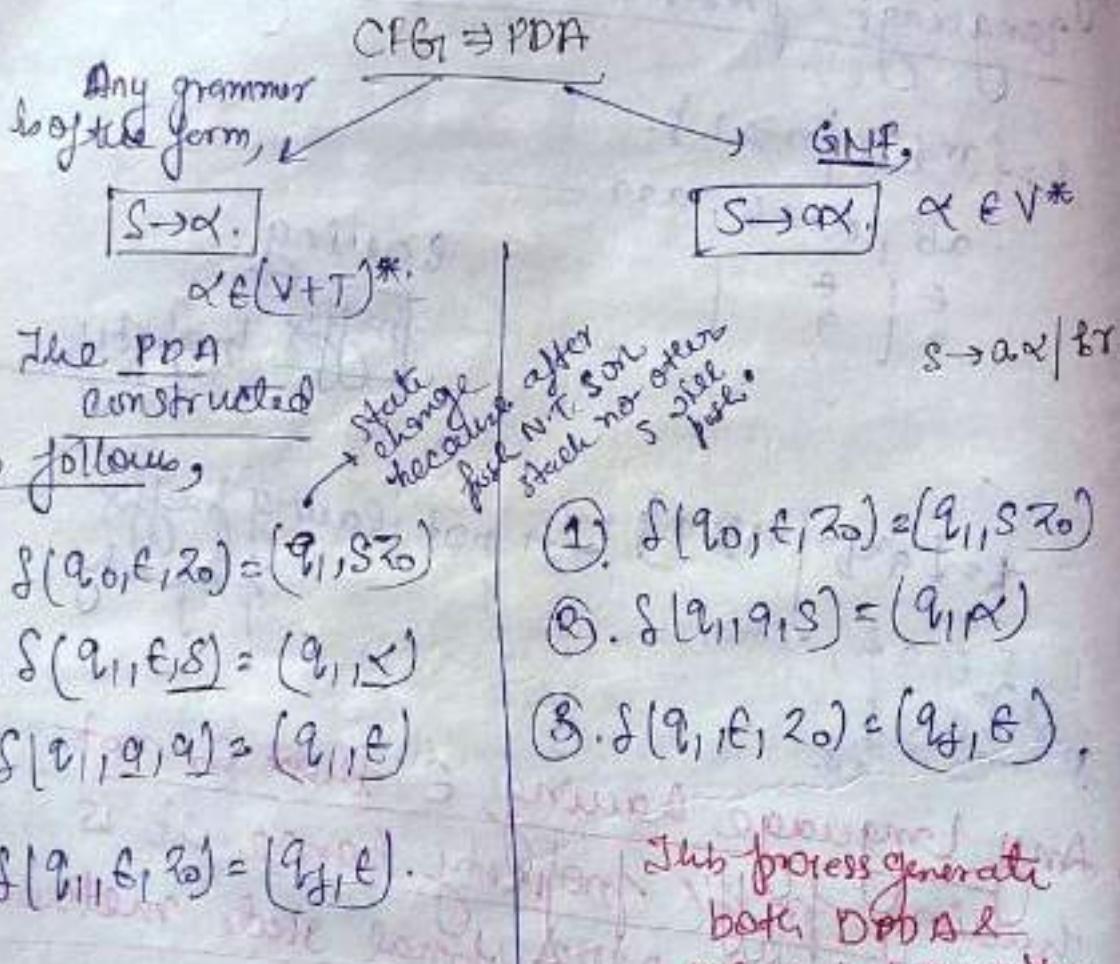
L = {	e	a b	aabb	... }
prefix	e	a b		

- Any language having e exist is not having prefix property hence it is accepted by DPD A final state method only but not by empty stack.

^{proper} Prefix being known set me (language) do not prefix property & only final state use.

CFG to PDA

Construction :-



In this process generates both DPDA & NPDA but more time DPDA.

Q. Construct state PDA for the foll. grammar

$S \rightarrow aSB | ab$

(grammar is ambn).

36

1st step:-

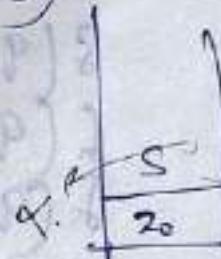
$$① \delta(q_0, \epsilon, z_0) = (q_1, \epsilon, S_0)$$

$$② \delta(q_1, \epsilon, S) = (q_1, aSb) \text{ or } (q_1, ab)$$

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

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

$$③ \delta(q_1, f, z_0) = (q_f, \epsilon)$$



Grammar to PDA always NPDA, DPDA
very less time come.

→ In Grammar to PDA construction the
resultant PDA is NPDA only.

Q.

$$S \rightarrow aA \\ A \rightarrow a\underline{ABD} \mid \underline{B}B \mid a$$

$$B \rightarrow b$$

$$D \rightarrow d$$

$$\textcircled{1}. \quad \delta(q_0, \epsilon, z_0) = (q_1, Sz_0)$$

$$\textcircled{2}. \quad \delta(q_1, \epsilon, S) = (q_1, aA)$$

$$\delta(q_1, \epsilon, A) = (q_1, aABD) \oplus (q_1, bB) \oplus$$

$$\delta(q_1, \epsilon, B) = (q_1, b) \oplus (q_1, a)$$

$$\delta(q_1, a, a) = (q_1, \epsilon), \quad \delta(q_1, \epsilon, D) = (q_1, d)$$

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

$$\delta(q_1, d, d) = (q_1, \epsilon).$$

$$\textcircled{3}. \quad \delta(q_1, \epsilon, z_0) = (q_1, \epsilon)$$

Q. Construct the PDA for the foll GRM grammar,

$$S \rightarrow a(SB) | SB \\ B \rightarrow B | \epsilon$$

\Rightarrow : $\textcircled{1} \quad \delta(q_0, \epsilon, z_0) = (q_1, Sz_0)$

terminal backli stuck me
push n i tota

terminal-terminal pop
operator not here.

$S \rightarrow a\alpha / b\gamma$

$$\textcircled{1} \quad \delta(q_0, \epsilon, z_0) = (q_1, Sz_0)$$

$$\textcircled{2} \quad \delta(q_1, a, S) = (q_1, \alpha)$$

$$\delta(q_1, b, S) = (q_1, \gamma)$$

$$\textcircled{3} \quad \delta(q_1, \epsilon, z_0) = (q_1, \epsilon)$$

$$\textcircled{1} \quad S(z_0, \epsilon, z_0) = (q_1, Sz_0)$$

$$\textcircled{2} \quad \delta(q_1, a, S) = (q_1, \alpha), \{ \delta(q_1, b, S) = (q_1, \beta)$$

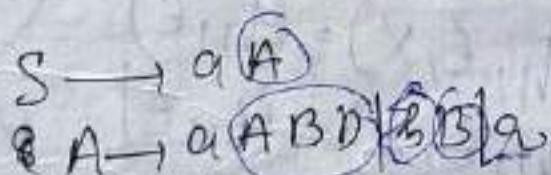
$$\delta(q_1, b, S) = (q_1, \beta)$$

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

$$\textcircled{3} \quad \delta(q_1, \epsilon, z_0) = (q_1, \epsilon)$$

OPDA.

Q. Construct the PDA,



$$\textcircled{1} \quad \delta(q_0, \epsilon, z_0) = (q_1, Sz_0)$$

$$\textcircled{2} \quad \delta(q_1, a, S) = (q_1, A)$$

$$\delta(q_1, a, A) = (q_1, ABD)$$

$$\delta(q_1, b, A) = (q_1, B)$$

NPDA

$\text{BFS}(q_1, \epsilon) = (q_1, \epsilon)$

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

$$\delta(q_1, d, D) = (q_1, \epsilon).$$

③ $\delta(q_1, \epsilon, z_0) = (q_1, \epsilon).$

PDA to CFG constructions:-

Q Construct tree CFG_T equivalent to the following PDA,

$$\delta(q_0, \underline{0}, z_0) = (q_0, \underline{x} z_0) \quad \left. \begin{array}{l} \\ \end{array} \right\} \text{push operation}$$

$$\delta(q_0, \underline{0}, x) = (q_0, \underline{xx})$$

$$\delta(q_0, 1, x) = (q_1, \epsilon)$$

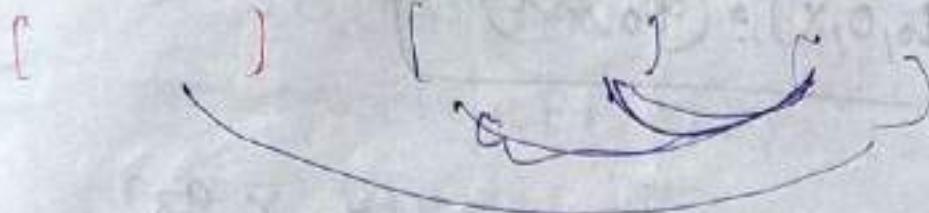
$$\delta(q_1, 1, x) = (q_1, \epsilon)$$

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

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

pop operation

$a^n b^m \mid n \geq m$



$$S \rightarrow [q_0, z_0, q_0]$$

$$S \rightarrow [q_0, z_0, q_1]$$

$$f(q_0, 1, x) = (q_1, \epsilon)$$

$$[q_0, x, q_1] \rightarrow 1 \rightarrow \textcircled{1}$$

$$\underline{f(q_1, 1, x) = (q_1, \epsilon)}$$

$$[q_1, x, q_1] \rightarrow 1 \rightarrow \textcircled{2}$$

$$\underline{f(q_1, \epsilon, x) = (q_1, \epsilon)}$$

$$[q_1, x, q_1] \rightarrow \epsilon \rightarrow \textcircled{3}$$

$$[q_1, z_0, q_1] \rightarrow \epsilon \rightarrow \textcircled{4}$$

$$f(q_0, 0, z_0) = (q_0, x, z_0)$$

$$[q_0, z_0, q_0] \rightarrow 0 [q_0, x, q_0] [q_0, z_0, q_0]$$

$$[q_0, z_0, q_0] \rightarrow 0 [q_0, x, q_1] [q_1, z_0, q_0]$$

$$[q_0, z_0, q_1] \rightarrow 0 [q_0, x, q_0] [q_0, z_0, q_1]$$

$$[q_0, z_0, q_1] \rightarrow 0 [q_0, x, q_1] [q_1, z_0, q_1]$$

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

$$[q_0 \times \underline{q_0}] \rightarrow o [q_0 \times q_0] [q_0 \times q_0]$$

$$[q_0 \times \underline{q_0}] \rightarrow o [q_0 \times q_1] [q_1 \times \underline{q_0}]$$

$$[q_0 \times \underline{q_1}] \rightarrow o [q_0 \times q_0] [q_0 \times q_1]$$

$$[q_0 \times \underline{q_1}] \rightarrow o [q_0 \times q_1] [q_1 \times q_1]$$

$$[q_0, z_0, q_0] = A$$

$$[q_0, z_0, q_1] = B$$

$$[q_1, z_0, q_0] = C$$

$$[q_1, z_0, q_1] = D$$

$$[q_0, x_1, q_0] = E$$

$$[q_0 \times q_1] = F$$

$$[q_1 \times q_0] = G$$

$$[q_1 \times q_1] = H.$$

$$S \rightarrow A/B$$

$$F \rightarrow I$$

$$H \rightarrow I/E$$

$$D \rightarrow B$$

39

~~A → OEA | OFC~~
~~B → OEB | OFD~~
~~C → OEC | OFG*~~
~~D → OEF | OFH~~

Remove useless variable,

~~S → A | B~~
~~F → 1~~
~~H → 1 | E~~
~~D → E~~
~~B → OFD~~
~~F → OFH.~~

CFLs and DCFLs :-

Detection →

Q. which of the following lang. are ~~DCFLs~~

- (1) CFL and regular
- (2) CFL but not regular
- (3) Non-CFL

- ①. $L = \{a^n b^n c^n \mid n \leq 100\}$ ② ① • ^{CFL}
 2 Reg.
- ②. $L = \{a^n b^n c^m \mid n > m\}$ ③ 2 components ③ ^{not} CFL
- ③. $L = \{a^n b^n c^m \mid n = m\}$ ④ ①
- ④. $L = \{a^n b^n c^m \mid n, m \geq 1\}$ ⑤ ② ^{CFL & non-}
 Reg. (one component)
- ⑤. $L = \{a^n b^n c^{n+m} \mid n, m \geq 1\}$ ⑥ ③ ^{2 components}
 $c > b, a$
- ⑥. $L = \{a^n b^{m+n} \mid n, m \geq 1\}$ ②
- ⑦. $L = \{a^i b^j c^k \mid j = i+k\}$ ② ^{one component}
- ⑧. $L = \{a^i b^j \mid i=2j\}$ ②
- ⑨. $L = \{a^i b^j \mid i=j^2\}$ ③ ③
- ⑩. $L = \{a^i b^j \mid i=j+5\}$ ② one component
- ⑪. $L = \{a^n b^m \mid n \geq m \text{ or } n < m\}$ ②
- ⑫. $L = \{a^n b^m c^k \mid n > m > k\}$ ③ ③ ^{nm-CFL}
^{2 components}
- ⑬. ~~$L = \{a^n b^m c^k \mid n > m \text{ or } m < k\}$~~ ②
- ⑭. $L = \{a^n b^m \mid (n+m) \bmod 5 = 0\}$ ①
- ⑮. $L = \{a^i b^j \mid i \neq j\}$ ②
- ⑯. $L = \{a^n b^m \mid n = 2m+1\}$ ②
- ⑰. $L = \{a^i b^j \mid i \neq 2j+1\}$ ②
- ⑱. $L = \{a^n b^m \mid n - m = 5\}$ ②

19. $L = \{a^n b^m \mid n/m = 4\}$ ④ $m=4n$ ② ⑨ ⑩ ⑪ ⑫ ⑬ ⑭

20. $L = \{a^n b^m \mid n * m = 100\}$ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩ ⑪ ⑫ ⑬ ⑭

21. $L = \{a^{i^2} \mid i \geq 1\}$. ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩ ⑪ ⑫ ⑬ ⑭

22. $L = \{a^p \mid p \text{ is prime no.}\}$ ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩ ⑪ ⑫ ⑬ ⑭

23. $L = \{a^k \mid k \text{ is odd number}\}$ ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩ ⑪ ⑫ ⑬ ⑭

24. $L = \{i^{2n+1} \mid n \geq 1\}$. ① ② ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩ ⑪ ⑫ ⑬ ⑭

25. $L = \{a^i b^j c^k d^l \mid i=k \text{ and } j=l\}$. ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩ ⑪ ⑫ ⑬ ⑭

26. $L = \{a^i b^j c^k d^l \mid i=l \text{ and } j=k\}$ ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩ ⑪ ⑫ ⑬ ⑭

~~(C but not DCFL)~~ 27. $L = \{a^i b^j c^k d^l \mid i=k \text{ or } j=l\}$ ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩ ⑪ ⑫ ⑬ ⑭

28. $L = \{a^i b^j c^k d^l \mid i=k=l\}$ ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩ ⑪ ⑫ ⑬ ⑭

~~(C but not DCFL)~~ 29. $L = \{a^i b^j c^k d^l \mid i>k \text{ or } j<l\}$ ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩ ⑪ ⑫ ⑬ ⑭

30. $L = \{waw^R \mid w \in \{a\}^*\}$ ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩ ⑪ ⑫ ⑬ ⑭

$L = \{waw^R \mid w \in \{b\}^*\}$
non-Reg-
CFL

31. $L = \{wxw^R \mid w \in \{a, b\}^*\}$ ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩ ⑪ ⑫ ⑬ ⑭

32. $L = \{wxw^R \mid w, x \in \{a, b\}^*\}$ ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩ ⑪ ⑫ ⑬ ⑭

33. $L = \{ww \mid w \in \{a, b\}^*\}$ ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩ ⑪ ⑫ ⑬ ⑭

34. $L = \{wxw \mid w, x \in \{a, b\}^*\}$ ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩ ⑪ ⑫ ⑬ ⑭

35. $L = \{wxw \mid w \in \{a, b\}^*\}$ ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩ ⑪ ⑫ ⑬ ⑭

36. $L = \{w^* - \{ww \mid w \in \{a, b\}^*\}\}$. ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩ ⑪ ⑫ ⑬ ⑭

(37). $L = \{a^n b^n c^n \mid n \geq 1\}$. (2)

(38) $L = \{w w^R \mid w \in \{a, b\}^*\}$ (3)
not recognize weird word w not recognize weird $w w^R$ } 2 comparison.

(39). $L = \{x \mid x \in (a+b)^* \text{ and } m_a(x) > m_b(x)\}$ (2)

(40). $L = \{x \mid x \in (a, b, c)^* \text{ and } m_a(x) < m_b(x) < m_c(x)\}$

(41) ~~$L = \{w^2 \mid w \in \{a, b, c\}^*\}$~~ (3) ^{not accept in NFA}

(41). $L = \{w^2 \mid w \in \{a, b, c\}^*\text{ and } m_a(w^2) + m_b(w^2) = m_c(w^2)\}$ (3)

(42). $L = \{a^n b^m \mid n, m \text{ are prime nos.}\}$ (3)

(43). $L = \{a^n b^m \mid n \text{ is prime or } m \text{ is prime}\}$ (5)

(44). $L = \{a^n b^n c^m d^m \mid n, m \geq 1\}$ (2)

(45). Set of all even length palindrome string of Hindi language (2)

(46). Set of all odd length palindrome string of Chinese language. (2)

$\{w \mid w \text{ is palindrome over } \{a, b, c\}\} \rightarrow$ (5)

- (47) - Total palindrome strings of Japanese language. (2)
- (48) Set of all lexical error detected by Compiler. (1)
- (49) Set of all syntax error detected by compiler. (2)
- (50) Set of all grammatical errors detected by compiler. (3)
- (51) Total property of Sachin tendulkar. (1) (1) (3)
- (52) $L = \{a^n b^j c^k \mid k = j * n, n, j, k \geq 1\}$. (3)
- (53) $L = \{\omega \mid \text{let}(a, b)^* \text{ } n_a(\omega) = n_b(\omega)\}$ (3)
- (54) $L = \{a^n b^j c^k \mid n < j < k\}$ (3)
- (55) $L = \{a^n b^n c^m \mid n \leq m^2\}$ (3)
- (56) $L = \{a^n b^n \mid n \geq 0, n \neq 13\}$ (12) CFL-Reg. $a^{13} b^{13}$
CFN
- (57) $L = \{xay \mid (x, y) \in \{0, 1\}^*\}$ (1) (no common or Reg. symmetries)
- (58) $L = \{x \underline{w} w^R \mid (\omega, x) \in \{a, b\}^+\}$ (2)
Left-invar. ~~abstainable~~

(59) $L = \{ \overbrace{w w^R}^{1 \text{ component}} x \mid w, x \in \{a, b\}^+ \} \quad (2)$

(60) $L = \{ \underbrace{a^n b^n}_{\text{equal}} c^m \mid n, m \geq 1 \} \quad (3)$
 2 components

Rules to detect CFLs

Any finite language is Regular and CFL.

(1). Any infinite language more than one memory element more than one comparison. i.e., not-CFL.

(2). Any infinite lang. symbols power not in A.P., i.e., non-CFL.

(3). Any infinite lang. symbols power not in A.P., i.e., non-CFL.

(4). All palindrome languages are CFLs.

(5). There is no difference b/w CFL & Regular over one symbol languages. Hence, over one symbol if a language is over it is Regular.

and
if it is non-Regular then it is non-CFL.

CATC 2014

Consider the following 3-languages

~~addition~~

$$L_1 = \{0^n 1^n \mid n \geq 0\}$$

$$L_2 = \{w c w^R \mid w \in \{0,1\}^*\}$$

$$L_3 = \{w w^R \mid w \in \{0,1\}^*\}$$

which of the foll. is DCFLs,

- (1) All (2) only L_1
- (3) $L_1 \& L_2$ (4) None.

2010
Q.

$$L_1 = \{0^i 1^j \mid i \neq j\}$$

$$L_2 = \{0^i 1^j \mid i = j\}$$

$$L_3 = \{0^i 1^j \mid i = 2j+1\}$$

$$L_4 = \{0^i 1^j \mid i \neq 2j\}$$

only L_2 is context free

(1) only L_2 and L_3 are context free

(2) only L_1 and L_3 are context free

(3) All are CFLs

2006
Q.

$$L_1 = \{0^{\frac{n+m}{2}} 1^n 0^m \mid n, m \geq 1\}$$

$$L_2 = \{0^{\frac{n+m}{2}} 1^{\frac{n+m}{2}} 0^m \mid n, m \geq 1\}$$

$$L_3 = \{0^{\frac{n+m}{2}} 1^{\frac{n+m}{2}} 0^{\frac{n+m}{2}} \mid n, m \geq 1\}$$

Context free Scattered

which of the following are non-CPL's

- ① L_1 only
- ② L_2 only
- ③ L_3 only
- ④ $L_2 \& L_3$ only.

Q. Which of the following is DCPL,

- ① $L = \{ww^R \mid w \in \{a,b\}^*\}$
- ② $L = \{a^n b^n c^n \mid n \geq 1\}$
- ③ $L = \{ww \mid w \in \{0,1\}^*\}$
- ④ None.

V.VVV 3m
2015
me and hi

Closure properties of DCPLs

NCPLs :-

{Tabeki closure property facts related to closure
theory like join, meet etc}

① Union operation :- CPLI :-

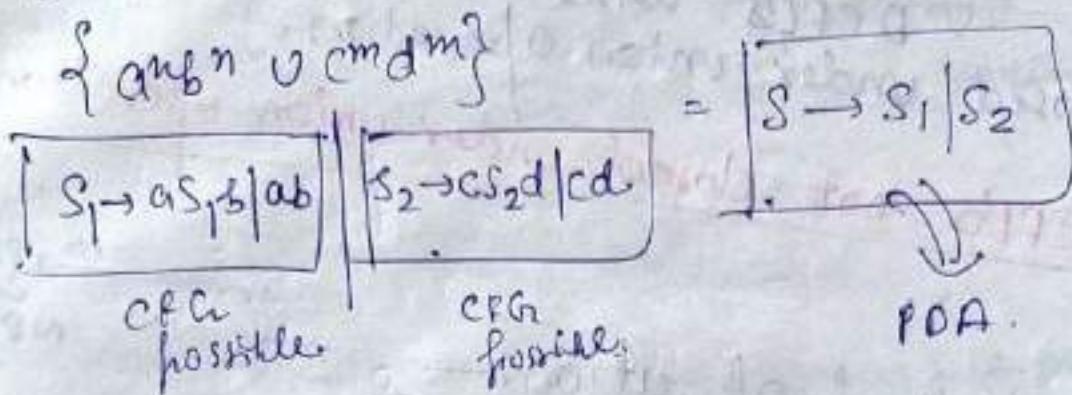
Let L_1 & L_2 are my two CPL, S_1 & S_2 are
CPAs for L_1 & L_2 respectively then the
union of $L_1 \cup L_2$ is always CPL because we
can construct union CFG.

can

construct

CFL \rightarrow

$$\{CFL_1 \cup CFL_2\} = CFL ?$$



DCFL \rightarrow

$$DCFL_1 \cup DCFL_2$$

$$\{a^n b^n c^m\} \cup \{a^n b^m c^m\} = \{a^i b^j c^k \mid i=j \text{ or } j=k\}$$

$\underbrace{\{a^n b^n c^m\}}_{DCFL} \cup \underbrace{\{a^n b^m c^m\}}_{DCFL} = \underbrace{\{a^i b^j c^k \mid i=j \text{ or } j=k\}}_{\text{NOT DCFL}}$

$$\{a^n b^n \mid n \geq 1\} \cup \{a^n b^n c^m \mid n, m \geq 0\} = \{a^n b^n c^m\}$$

$\underbrace{\{a^n b^n \mid n \geq 1\}}_{DCFL} \cup \underbrace{\{a^n b^n c^m \mid n, m \geq 0\}}_{DCFL} = \underbrace{\{a^n b^n c^m\}}_{DCFL}$

Hence CFL are closed under
union operation.

DEFLs:-

Union of two DEFLs may or may not be DEFLs. hence DCFLs are not closed under union operation.

DEFLs not closed under union operation.

(gramm \rightarrow PDA)
NPDA

② Subset operation:-

CPL \Rightarrow

CFL $\{a^n b^n | n \geq 1\}$

The subset of CFLs is may or may not be CFLs hence CFLs are not closed under subset.

$\{a^n b^n | n \geq 0\} \subseteq \{a^n b^n c^m | n, m \geq 0\}$

$\{a^n b^n c^m | n \geq 1\}$

\times DCFL.
 \times CPL.

The subset of DCFLs is may or may not be DCFLs hence DCFLs are not closed under subset.

③ Concatenation operation:-

CFL $\rightarrow L_1 \rightarrow \text{CFL} \rightarrow a^n b^m \quad S_1 \rightarrow aS_1 b / ab$
 $L_2 \rightarrow \text{CFL} \rightarrow a^n b^m \quad S_2 \rightarrow aS_2 b / ab$

$$L_1 \cdot L_2 \Rightarrow \boxed{S \rightarrow S_1 S_2}$$

$\Downarrow \text{PDA}$, because we can construct the grammar directly.

$\rightarrow L_1$ is CFL & L_2 is CFL then concatenation of $L_1 \& L_2$ is also CFL because we can construct the concatenation CFG for this.

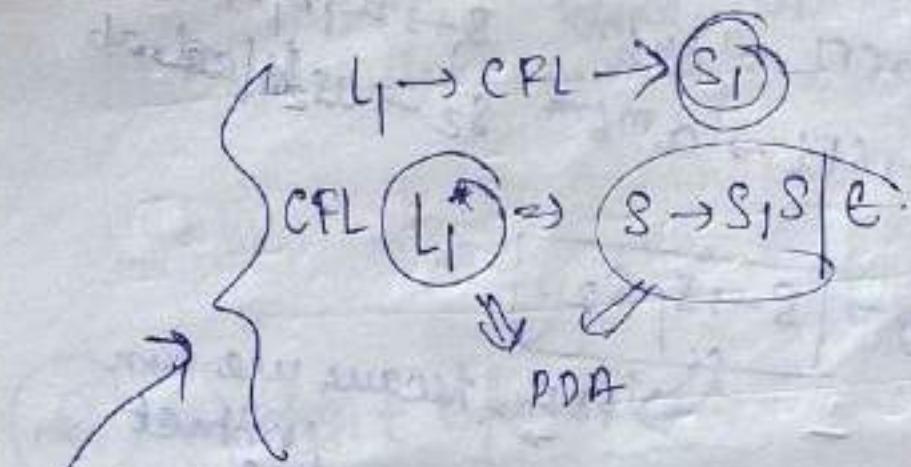
Closed under CFL

DCFL $\rightarrow L_1$ is DCFL & L_2 is DCFL then may concatenation of $L_1 \& L_2$ is closed under DCFL or not may DCFL.

not closed under DCFL

because always grammar to PDA conversion gives NPDAs. some the DPDAs.

Kleene closure:



The Kleene closure of CFL is always CFL because we can construct Kleene closure CFG as follows →

~~closed under~~
CFL is closed

The Kleene closure of DCFL is may
 (i) may not DCFL than DCFL
 not closed under kleene operation

DCFL is not closed

Positive closure:

See the closure of CFL is always CFL because we can construct the closure of CFL as follows—

$$L_1 \rightarrow S_1 \xrightarrow{\text{CFL}} \text{CFL}$$

$$L_1^+ \xrightarrow[\text{CFL}]{} S \xrightarrow{\text{CFL}} S_1 S_1^+ \xrightarrow{\text{CFL}}$$

CFL is closed

See the closure of DCFL is may or may not DCFL then DCFL not closed.

DCFL not closed

Intersection operation—

$$\underbrace{\{a^n b^n | n \geq 1\}}_{\text{CFL}} \cap \underbrace{\{a^n b^m c^m | n, m \geq 0\}}_{\text{not CFL}} = \underbrace{\{a^n b^n | n \geq 1\}}_{\text{CFL}}$$

$$\{a^n b^n c^m | n, m \geq 1\} \cap \{a^n b^m c^m | n, m \geq 1\}$$

$$\begin{array}{l} \text{CFL} = \dots \\ \text{DCFL} \end{array} \Rightarrow \begin{array}{l} \text{CFL / DCFL} \\ \{a^n b^n c^n | n \geq 1\} \\ \text{Not CFL / Not DCFL} \end{array}$$

Intersection is not closed for CPL.

Intersection is not closed for DCPL.

Ex) Intersection with Regular Languages :-

(Don't go through chomsky hierarchy.)



CPL n Regular

$$a^n b^n \cap (a+b)^* = a^n b^n$$

(CPL) (Reg) CFL

$$a^n b^n \cap a^* b^* = a^n b^n$$

(CPL) (Reg) CFL

$$\{a^n b^n\} \cap (a+b)^* = \{a^n b^n\} \quad |n \neq m|$$

(CPL) Reg. (CPL)

$$\{1 \leq n \leq m \mid a^n b^n\} \cap ab = ab$$

(CPL) (Reg) (Reg)

CFL intersection Regular is always
CPL but may or may not Regular
hence,

CFL are closed under intersection with
Regular Lang. operations.

$$X \cap \text{Regular} = X \text{ only}$$

$$\text{DCFL} \cap \text{Reg} = \text{DCFL}$$

$$\text{CFL} \cap \text{Reg} = \text{CFL}$$

$$\text{REL} \cap \text{Reg} = \text{REL}$$

hence
intersection
with Reg.
Lang. operation
closed under
all formal
Lang.

DCFL intersection with Regular is
always DCFL but may or may
not be regular.

Note:- $\text{CFL} \cap \text{CFL} = \text{CSL}$

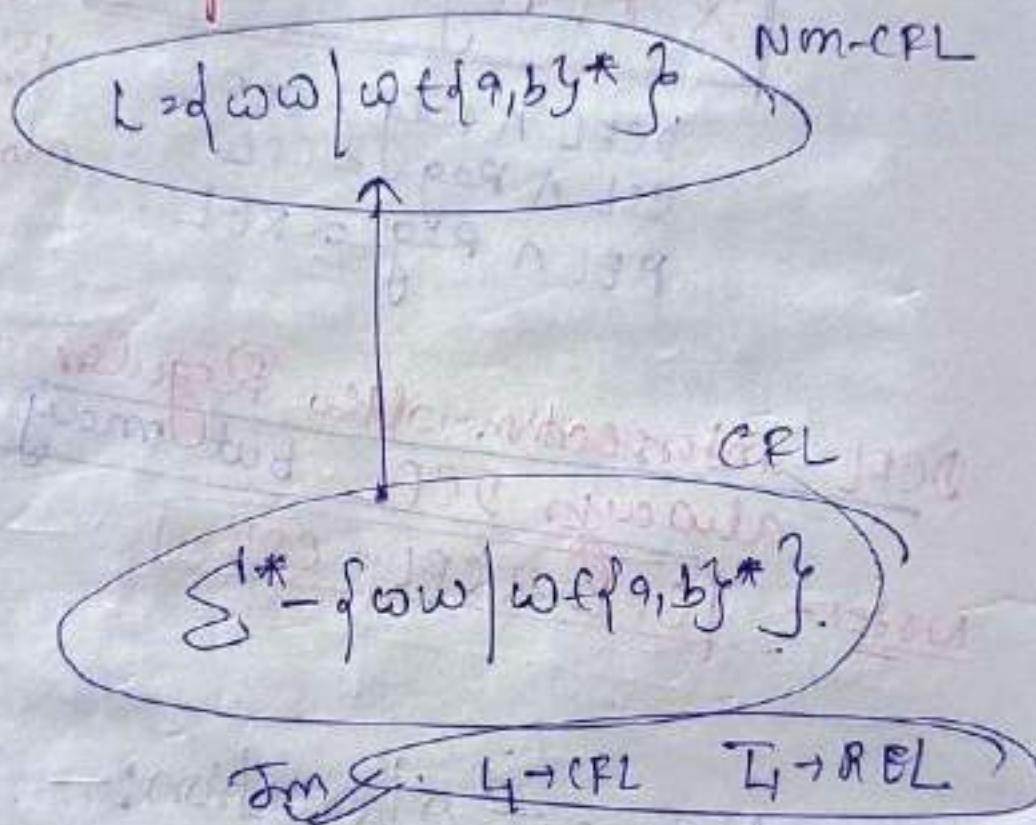
Complement operation:-

Note:-

$$\text{CFL} \cap \text{CFL} = \text{CSL}$$

L
 \uparrow
 T

Complement of a CFL is may or may not be a CFL hence CFL are not closed under complement operation.



~~True: $L_1 \rightarrow \text{CFL} \quad L_2 \rightarrow \text{REBL}$~~

$$L_1 \cap L_2 = \overline{L_1} \cup \overline{L_2}$$

CFL - CFL

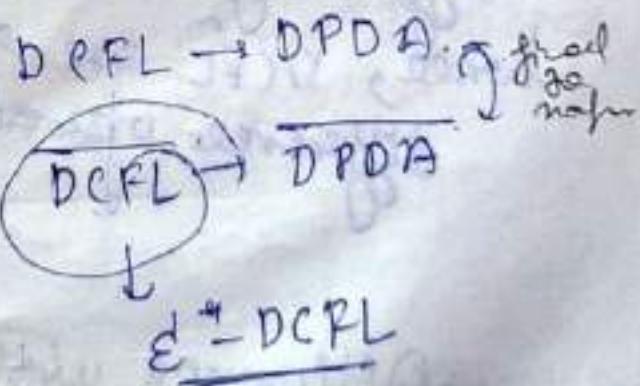
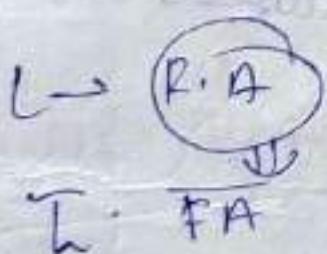
any one of \cup & \cap are not closed

so, $\cup \rightarrow$ closed

so, $\cap \rightarrow$ not closed.

Note:- The complement of DCFL is always DCFL. Hence DCFL are closed under complement operation.

Note:- Complement of a CFL is recursive language.



because we can complement DPDA hence
 $\Sigma^* - DCFL$ is also a DCFL.

Difference operation:-

- $L_1 \text{ & } L_2$ are CFLs then $L_1 - L_2$ may
 (m) may not be CFLs

because

$$L_1 - L_2 = L_1 \cap \overline{L_2}$$

$\overbrace{\quad\quad\quad}^{\text{not closed}}$

$\overbrace{\quad\quad\quad}^{\text{not closed}}$

$\overbrace{\quad\quad\quad}^{\text{not closed}}$

DCFL:-

$$\text{DCFL} - \text{DCFL} = \text{DCFL} \cap \overbrace{\text{DCFL}}^{\text{not closed}}$$

difference of DCFL may or may not be CFLs
hence, DCFL are not closed under
difference operation.

Difference with Regular :-

CFL difference to other Regular is
always CFL

CFL - Reg \Rightarrow CFL \cap Reg

Reg

CFL \cap Reg \Rightarrow CFL

DCFL - Reg \Rightarrow DCFL \cap Reg

Reg

\Rightarrow DCFL \cap Reg

\Rightarrow DCFL.

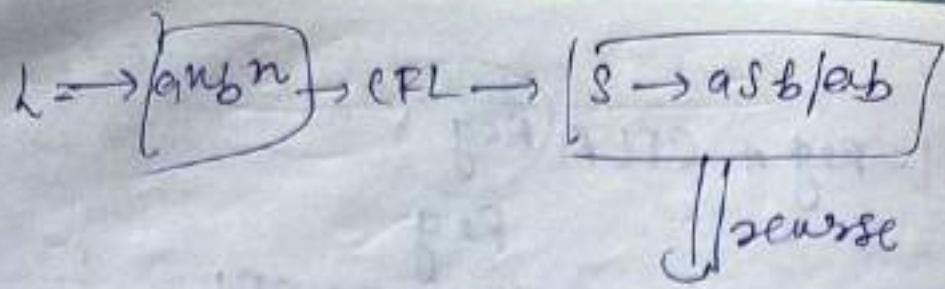
Note:- $X - \text{Reg} = X$ only \rightarrow hence this operation is closed under all formal lang.

DCFL difference with Regular is always DCFL.

Hence, CFL & DCFL are closed under difference with Regular Lang. operation.

Reversal operation:-

\rightarrow Reversal of a CFL is always CFL because we can construct reversal CFG by reversing Right hand side part of the CRN.



$$LR = \boxed{b^n a^n} \leftrightarrow \boxed{S \rightarrow b S a / b a}$$

Hence

CFLs are closed under reversal operation

DCRLs:-

The reversal of a DCRL is may or may not DCRL hence DCRLs are not closed under reversal operation.

DCRLs are not closed under reversal operation.

	<u>CFL</u>	<u>DefL</u>
①. U	✓	✗
②. •	✓	✗
③. *	✓	✗
④. +	✓	✗
⑤. ∩	✗	✗
⑥. ∼	✗	✗
⑦. MultiReg.	✓	✓
⑧. n[complement]	✗	✓
⑨. L ₁ -L ₂	✗	✗
⑩. L ₁ -R	✓	✓
⑪. LR.	✓	✗
⑫. Substitution	✓	✗
⑬. Plurimorphism	✓	✗
⑭. Inverse homomorphism	✓	✗
⑮. Quotient (L ₁ /L ₂)	✗	✗

- ⑯ Quotient (L/R) ✓ ✓
 with Reg

 ⑰. Σ^* ✓ ✓

 ⑱. V with Reg ✓ ✓

Q. Let $P \rightarrow \text{CFL}$, $Q \rightarrow \text{Regular}$ then which of the following is always regular.

- (a) $P \cap Q$ (CFL) ✓ $X \cap \text{Reg} = X$
- (b) $P - Q$. (CFL) ✓ $\text{CFL} - \text{Reg} = \text{Reg}$
- (c) $\Sigma^* - P$ (not CFL) ✓
- (d) $\Sigma^* - Q$...

2009. Let $L = L_1 \cap L_2$ where

$$L_1 = \left\{ a^n b^m c^n c^m \mid n, m \geq 0 \right\} \rightarrow \text{CFL}$$

$$L_2 = \left\{ a^i b^j c^k \mid i, j, k \geq 1 \right\} \rightarrow \text{Reg}$$

- $L \in$
- ①. CFL and Reg.
 - ②. CFL but not Reg.
 - ③. not CFL
 - ④. Reg but not CFL
- $\text{CFL} \cap \text{Reg} \neq \text{CFL}$
 bid may
 be Reg
 @ may
 not be Reg.
 FL

$a^n b^n c$ → result of intersection of
It is $L_1 \cap L_2$
not reg.

Q. Let $L_1 = \{0^p 1^q 0^r \mid p, q, r \geq 0\} \rightarrow \text{Reg}$

$L_2 = \{0^p 1^q 0^r \mid p \neq q \neq r\} \rightarrow \text{CPL}$

which of the foll. is false?

- (a) L_2 is CFL
- (b) $L_1 \cap L_2$ is CFL
- (c) Complement of L_2 is recursive
- (d) complement of L_1 is CPL but not Reg.

Q. Let L_1 & L_2 are DFLs L_3 is CPL

to, L_4 is CPL & L_5 is
Regular then which of the foll.
is false?

- (a) $(L_3 \cup L_4)^c$ is Recursive $(CPL)^c \rightarrow \text{Recursive}$
- (b) $L_4 \cap L_5$ is DFL
- (c) $(L_3^R \cup L_4) \cap L_5 \rightarrow \text{CPL}$
- (d) $L_3 \cap L_4 \rightarrow \text{CPL}$
recurs. recursive not evn

Q. Let $L = \{a^n b^n \mid n \geq 0, c^n \neq 3\}$

CFL

- (A) DCFL
- (B) CFL but not DCFL
- (C) Regular
- (D) non-CFL

$$\{a^n b^n \mid n \geq 0\} - \{a^n b^n \mid n=13\}$$

DCFL

Reg

\Rightarrow DCFL

Ans

Q. Consider the following five languages,

$$L_1 = \{a^p b^q c^r \mid p=3(q+r)\} \quad \text{DCFL} \quad a^{\frac{3q+3r}{2}} b^q c^r$$

$$L_2 = \{a^p b^q c^r \mid p < q \text{ or } q < r\} \quad \text{NCFPL} \quad a^{\frac{p+q+r}{2}}$$

$$L_3 = \{w w^R \mid w \in \{a, b\}^*\} \quad \text{NCFPL}$$

$$L_4 = \{w w^R \mid w \in \{a, b\}^*\} \quad \text{DCFL}$$

$$L_5 = \{a^n b^{2n} \mid n \in \{a^n b^{3n}\} \cup \{a^n b^{4n}\}\} \quad \text{NCFPL}$$

which of the following is true.

- (A) all are CFL & DCFL
- (B) L_1, L_3, L_5 are DCFL L_2, L_4 are CFL
but not DCFL
- (C) L_1, L_4 are DCFL and L_2, L_3, L_5
are CFL but not DCFL

(d) None of these

~~SM~~

Q. Let L_1 & L_2 are CFLs

L_3 & L_4 are regulars
then which of the foll. is always CFL.

$$\text{CFL} \cdot \text{Reg} = \text{CPL}$$

- ①. $(L_1 \cup L_2) \cap (L_3 \cap L_4)$ → CFL
- ②. $(L_1^R \cap L_3^*) \cup (L_2^* \cap L_4^*)$ → CFL
- ③. $[L_1^* \cap (\Sigma^* - L_3)] \cap [L_2^R \cap (\Sigma^* - L_4)]^*$ → CFL or not CFL
- ④. $[L_1^R \cdot L_2^*] \cap [L_4^* \cdot L_3^R]$ → CFL
- ⑤. $(L_1^R \cdot L_3^*) \cap (L_2 \cdot L_4)$ → CFL or not CFL
- ⑥. $(L_1 \cdot L_3) \cdot (L_2 \cdot L_4)$ → CFL
- ⑦. $(L_1 \cdot L_3) - (L_4 \cdot L_3)$ → CFL
- ⑧. $(L_1^R \cdot L_4^*) - (L_2^R \cdot L_3^*)$ → $\text{CFL} - \text{CFL}$ → CFL or not CFL
- ⑨. $(L_1 / L_3) \cap (L_2 / L_4)$ → $\text{CFL} \cap \text{CFL}$ → CFL or not CFL
- ⑩. $(L_1 / L_3) \cap (L_2 / L_3)$ → $\text{CFL} \cap \text{CFL}$ → CFL or not CFL

Ans: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Decision properties of CFL & DCFL

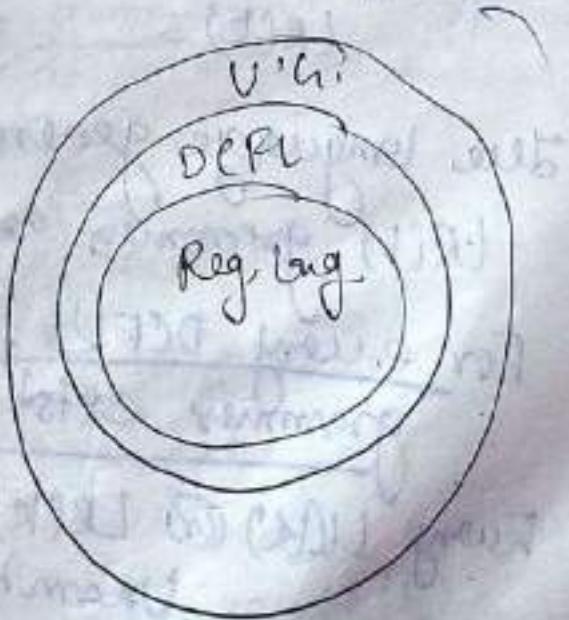
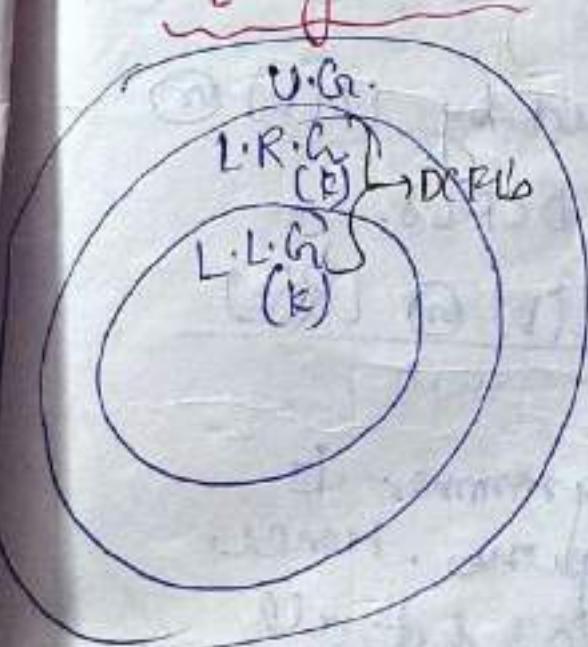
The following problems are decidable under CFL & DCFL →

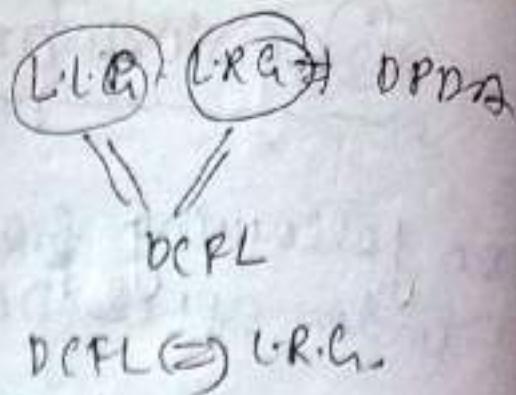
- ①. Emptiness problem
- ②. Finiteness problem
- ③. Membership problem

* Equivalence problem is undecidable for CFL & DCFL.

DCFLs and LR grammars and

LL grammars → (we understand it more
completely).





- Every LL(k) grammar is LR grammar but every LR grammar is need not be LL grammar.
- Every LL(k) is LR(k) grammar but every LR(k) is need not be LL(k).
- Every LL(k) grammar & LR(k) grammar is unambiguous grammar.
But all V.G need not be LL(k) \cap LR(k).
- The language generated by LL(k) \cap LR(k) grammar are DCFL's.
- For every DCFL's, LL(k) \cap LR(k) grammar exist.
- Every LL(k) \cap LR(k) grammar is unambiguous. Hence every DCFL is also V.G, but all unambiguous lang. need not be DCFL's.

→ Every seg. lang. DCFL. Hence, every regular lang. is also unambiguous.

→ Reg. lang. is unambiguous but Regular grammar may be ambiguous.

One lang. unambiguous bolde er to iska
habut gramma to salatka
lakin isme se ek gramma to
unambiguous (at least) to gaya to
R. lang is unambiguous.

Q. consider the following ^{two} statements S_1 & S_2 ,

$S_1 \rightarrow$ for every seg. lang there exist
LR(k) grammar.

$S_2 \rightarrow$ Every Reg. gramm is unambiguous.

$S_2 \rightarrow$ both S_1 & S_2

① Both S_1 & S_2 true

② Both S_1 & S_2 false

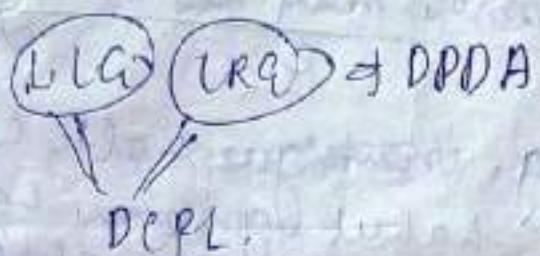
③ S_1 is true S_2 is false

④ S_1 is false S_2 is true.

for every DCFL we have LR(k)
and Reg is closed and DCFL
is finite regular language LR(k).

Note:-

The PDA which is used for parsing algorithm
is DPDA hence parsing grammar generates
DCPLs.

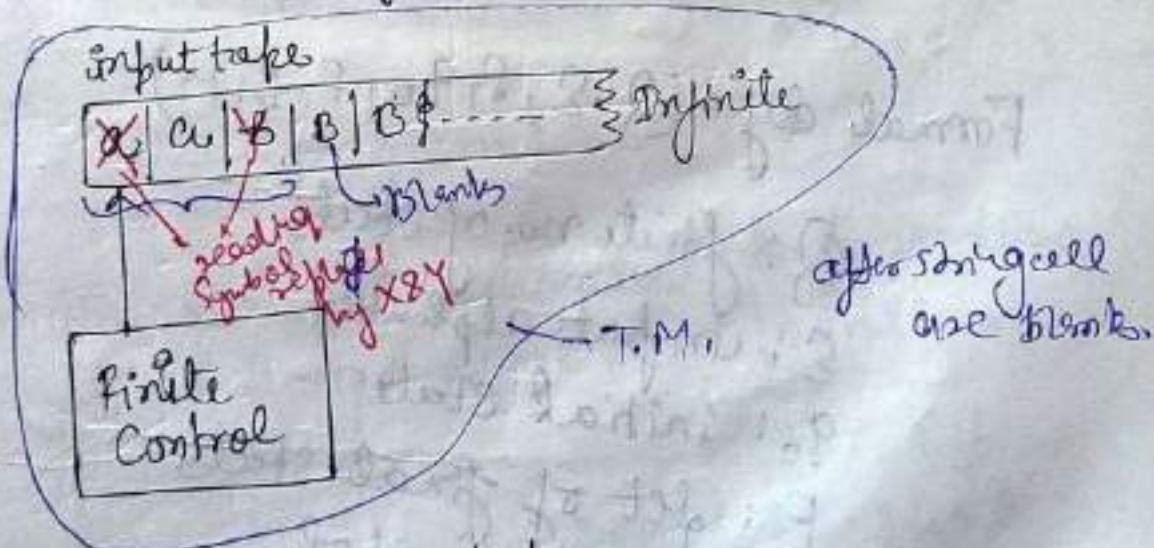


Turing m/c & Recursive

Enumerable language:-

Turing m/c:- equal no procedure type
not as algorithm but goes

- Turing m/c is a mathematical model used to represent the behaviour of general purpose computer. i.e., whatever is possible with turing machine is also possible with computer. whatever is not possible with T.M. that is not possible with computer.



- ①. Infinite length tape
- ②. Read write Capability
- ③. Turnaround Capability.

When compare to finite automata,
T.M having following properties →

- ① Infinite length tape → the input tape of the T.M is one side ended & one side infinite
- ② Read-Write Capability → T.M capable to write, T.M capable to read with the reading symbol by some symbol or other symbol.
- ③ Turnaround Capability → T.M capable to turn left as well as right side direction.

Formal def = $(Q, \Sigma, \delta, q_0, F, B, \mu)$

Q : finite no. of states

Σ : input alphabet

q_0 : initial state

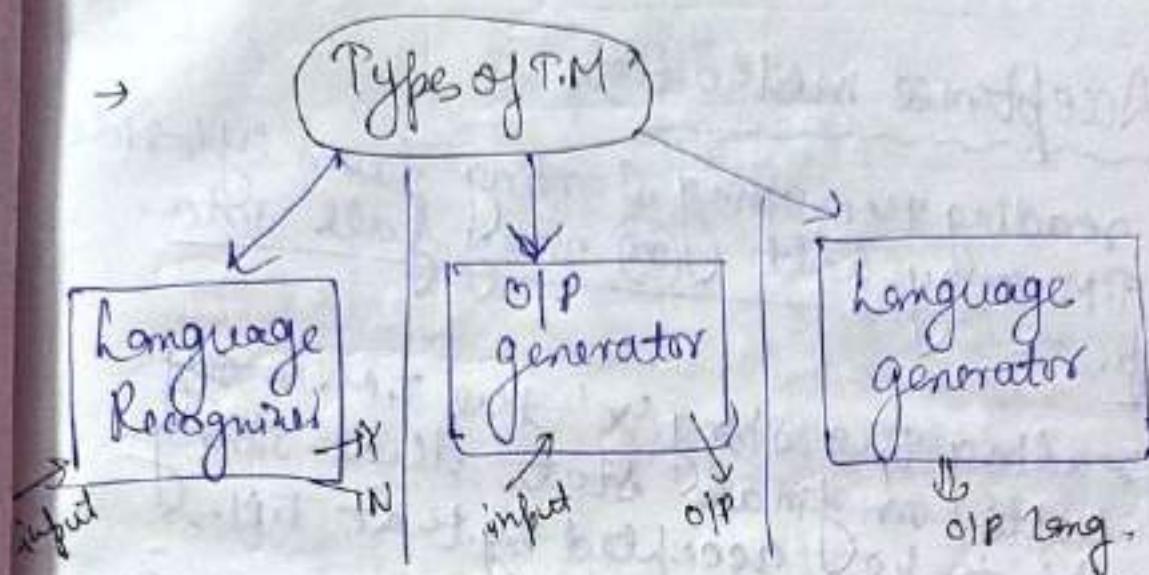
F : set of final states

B : Blank symbol

μ : Tape alphabet

δ : Transition function

$$\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$$



→ Notations

- ①. Transition diagram
- ②. Transition table ✓ ***

(T.M is like a program)

- Turing machine is equivalent to procedure & almost equivalent to algorithm.
- algorithm is also a procedure that halts after finite no. of steps. but procedure no guarantee to halt (may or may not halt).

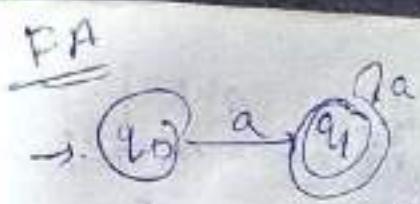
T.M. as language Recognizer →

Acceptance method ⇒

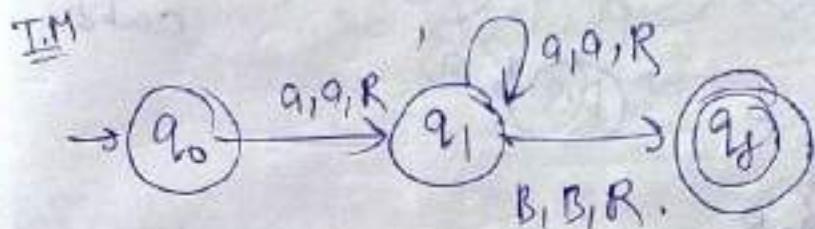
- By reading the string starting from left side T.M. may halt or may fall into loop.
- By reading the string 'X' if the T.M. ~~may~~ halts on final state that string is said to be accepted by that T.M..
- By reading the string 'X' if the T.M. halts on non-final state, that string is said to be rejected by that T.M..
- By reading the string 'X' if the T.M. enters into infinite loop even it is difficult to say string is accepted or rejected.

Q. Construct the T.M. for the lang -

$$L = \{a^n b^n \mid n \geq 1\}$$



$S: Q \times \Sigma \rightarrow Q \times \Sigma \times \{L, R\}$



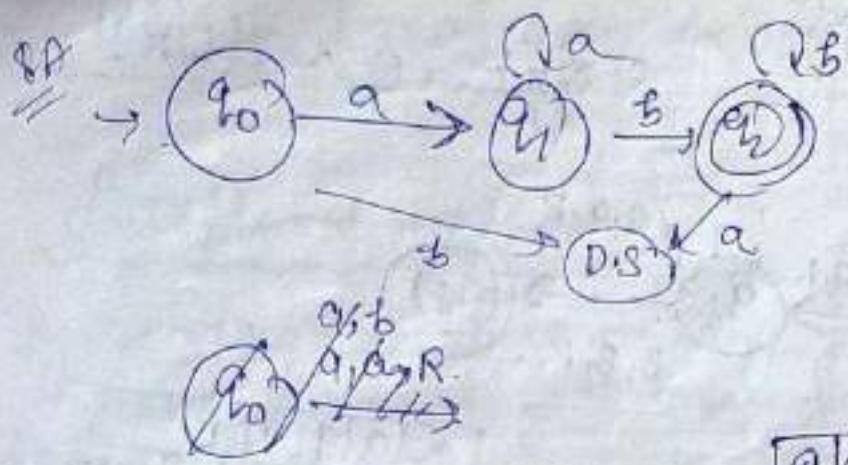
String accepted

a a a a a a B
 ↑ ↓ ↑ ↓ ↑ ↓ accepted
 a₀ a₁ q₁ q_f

	a	1	B
q ₀	(q ₁ , a, R)		HALT
q ₁	(q ₁ , q ₁ , R)	(q _f , B, R)	
q _f		HALT	

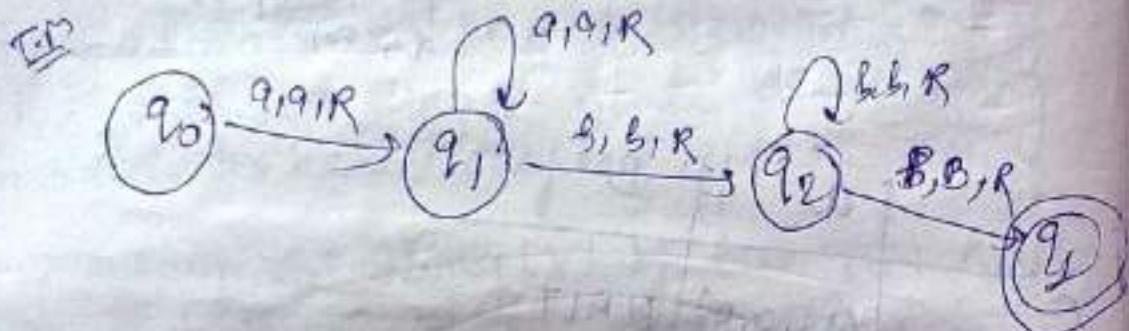
→ Blank mila gaya final state f.e. mean halt.

Q. Construct the TM for the lang
 $L = \{a^n b^m \mid n, m \geq 1\}$.



a^2b^2n
 $n, m_2, 1$
 $n -$
 ab
 $aabb$

$|a|a|b|b|b$



	a	b	B
q_0	q_1, a, R	HALT	HALT
q_1	q_1, a, R	q_2, b, R	HALT
q_2	HALT	q_2, b, R	(q_3, B, F)
q_3	HALT	L	T

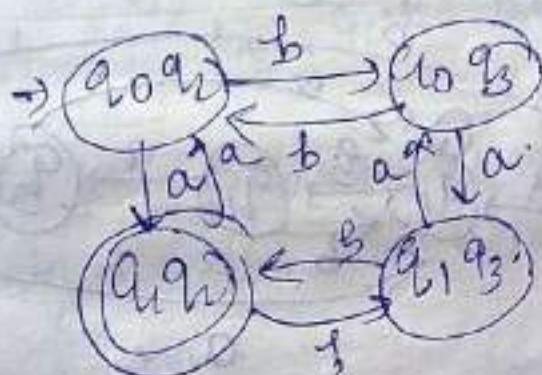
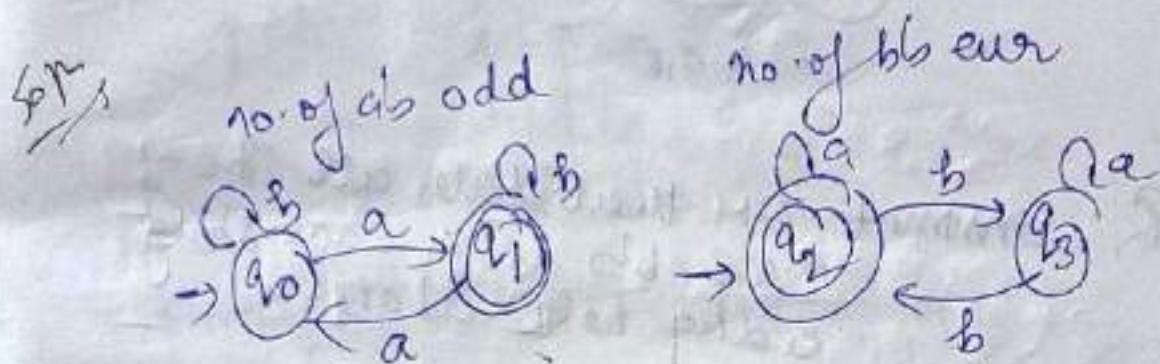
$|a|b|a|b$

$q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2$
 halt rejected

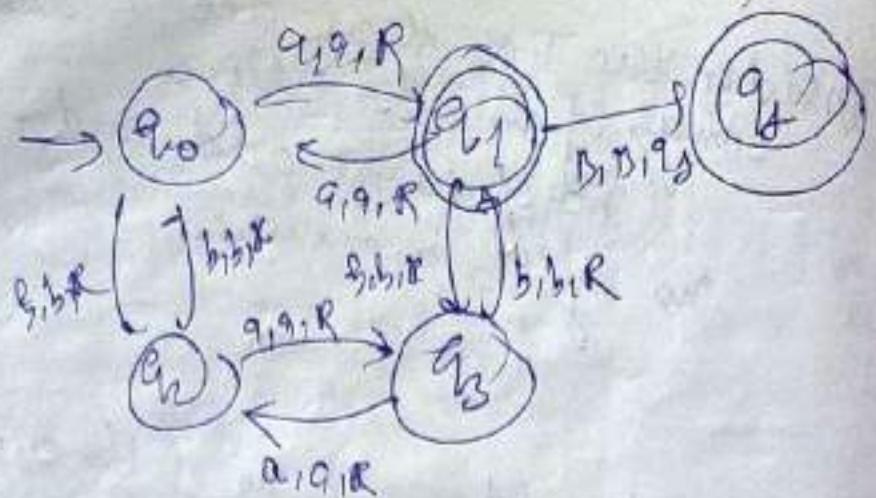
$|a|b|B|B|B$

$q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2$ accept B

Q. Consider the T.M that accept all strings of a s & b s where no. of a s are odd and no. of b s are even.

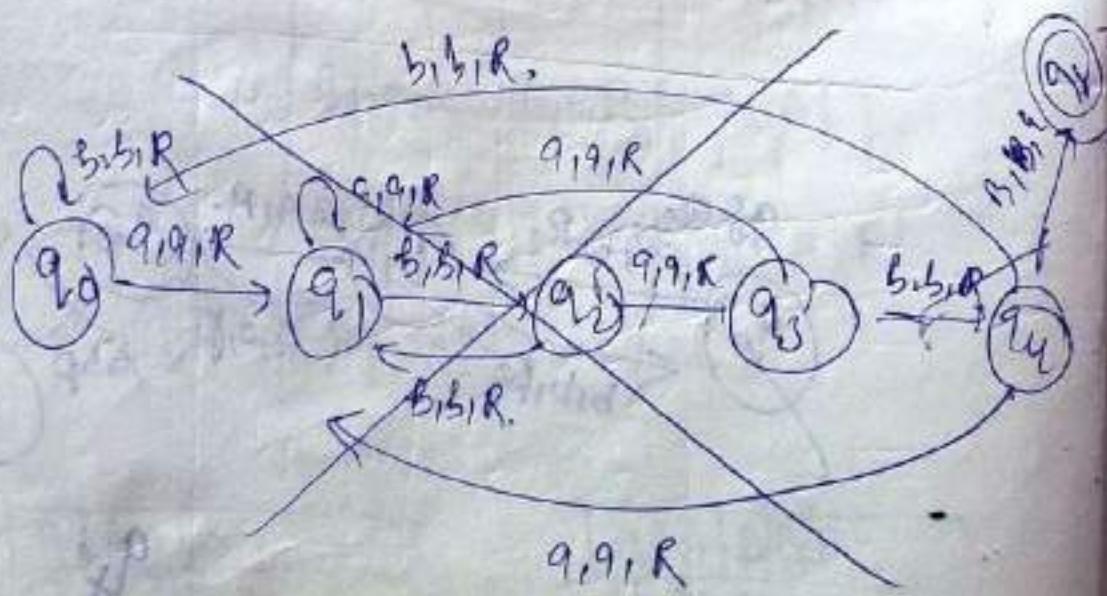
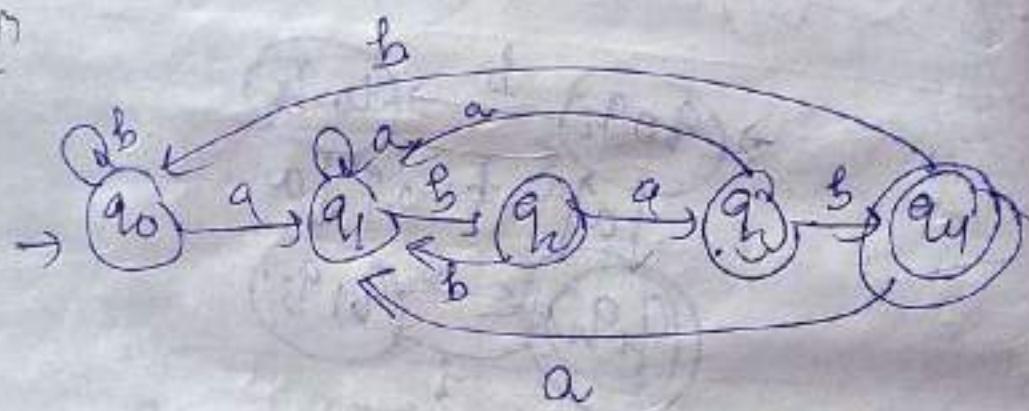


q_f'



Q. Construct TM that takes all strings of $a^k b^k$ where each string ending with $abab$.

Sol:



$(a+b)^*abab$

$a|a|b|b|ab|ab|B$

↑ ↑ ↑ ↑ ↑ ↑
↓ ↓ ↓ ↓ ↘
→ agent cur



Regular lang. replace by some symbol
only because here we have no
comparision.

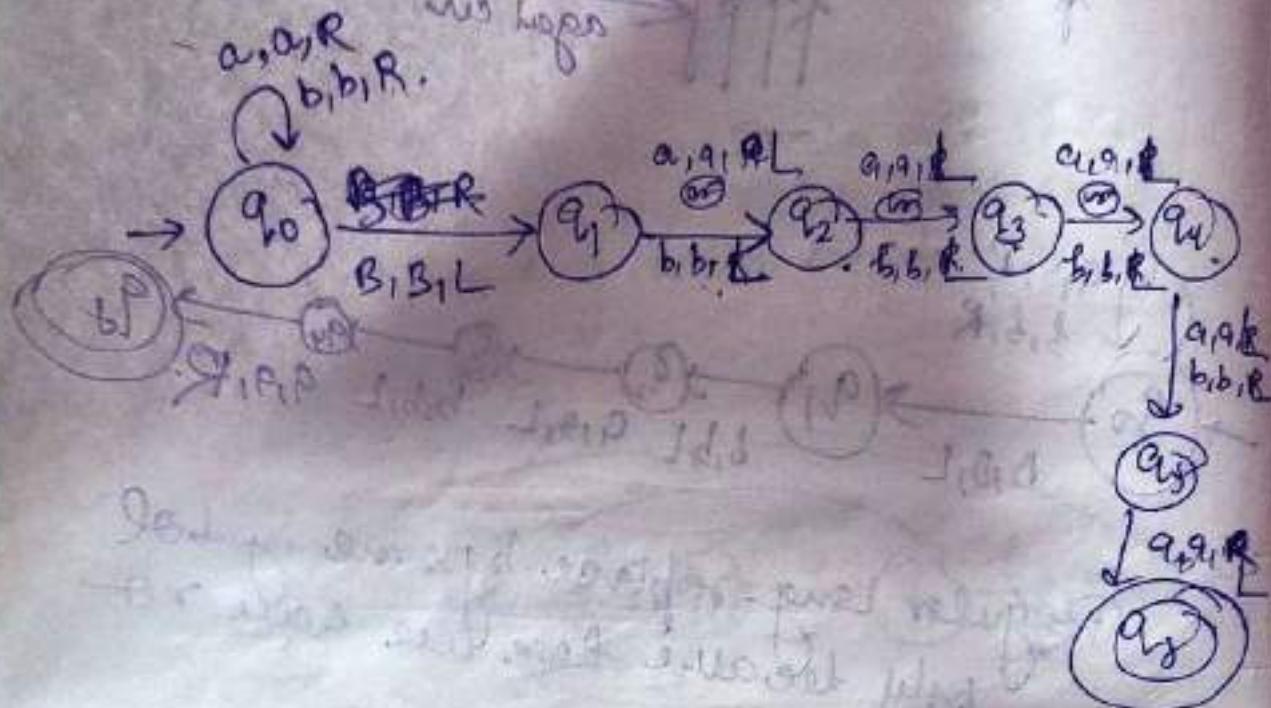
Q. Construct the T.M. that accepts all strings
of $a^m b^n$ where the input symbol is
while reading the string from
R.M.S.

Soln:

$$(a+b)^* a (a_{fb}) (a_{fb}) (a_{fb}) (a_{fb}) B$$

stop

was hopen



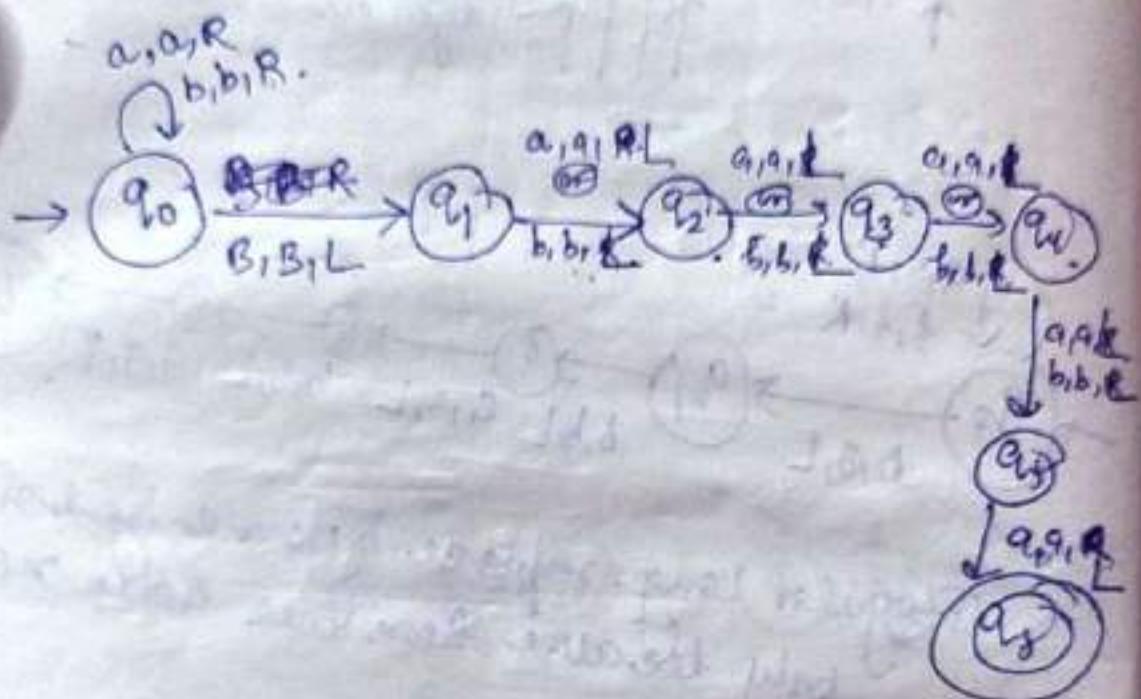
	a	b	B
$\rightarrow q_0$	(q_0, q_1, R)	(q_0, b, R)	(q_1, B, L)
q_1	(q_2, q_1, L)	(q_2, b, L)	Halt
q_2	(q_3, q_1, L)	(q_3, b, L)	Halt
q_3	(q_4, q_1, L)	(q_4, b, L)	Halt
q_4	(q_5, q_1, L)	(q_5, b, L)	Halt
q_5	(q_6, q_1, L)	Halt	Halt
q_6		Halt	Halt

$$2^n \rightarrow 2^5 \rightarrow 32$$

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

$\xrightarrow{\text{B}}$

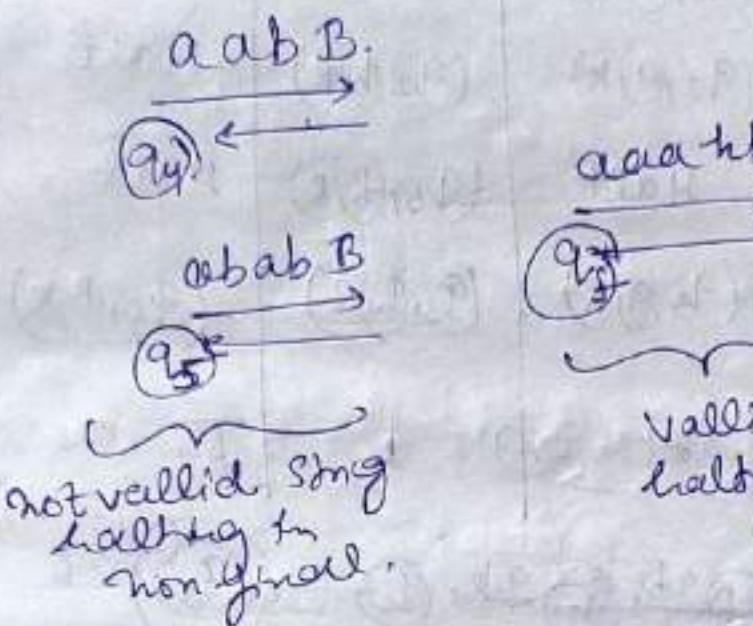
↓
skip
↓



	a	b	B
$\rightarrow q_0$	(q_0, q_1, R)	(q_0, B, R)	(q_1, B_1, L)
q_1	(q_2, a, L)	(q_2, b, L)	Halt
q_2	(q_3, a, L)	(q_3, b, L)	Halt
q_3	(q_4, a, L)	(q_4, B, L)	Halt
q_4	(q_5, a, L)	(q_5, b, L)	Halt
q_5	(q_5, a, L)	Halt	Halt
q_f		Halt	Halt

Q. Find the halting state for input aab.
in above problem

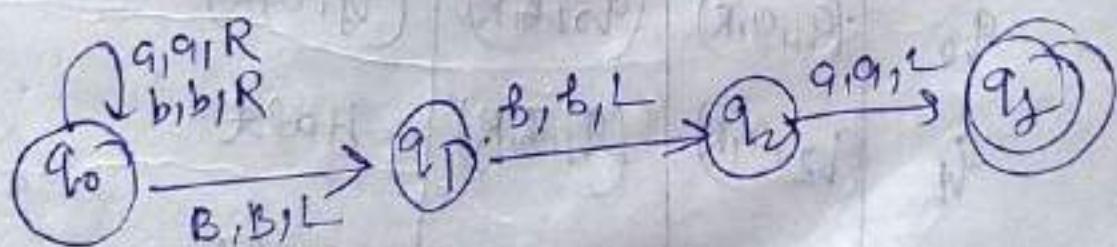
S.P.:



Q. Construct the T.M. that accept all strings of $a^k b^l$ where each string is ending with ~~ab~~ as substring.

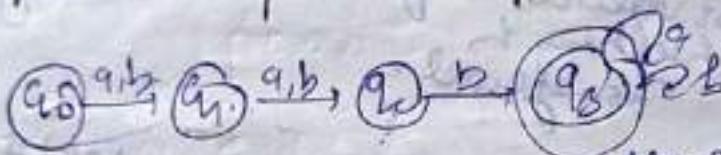
S.P.:

$$(aab)^* abB$$



Q. Identify language accepted by following T.M,

	a	b	B	
$\rightarrow q_0$	(q_1, a, R)	(q_1, b, R)	Halt	always accept Reg. lang. because the input ab one not replace
q_1	(q_2, a, R)	(q_2, b, R)	Halt	
q_2	Halt	(q_3, b, R)	Halt	
q_3	(q_3, a, R)	(q_3, b, R)	(q_4, B, R)	
q_4	H	A	L	T



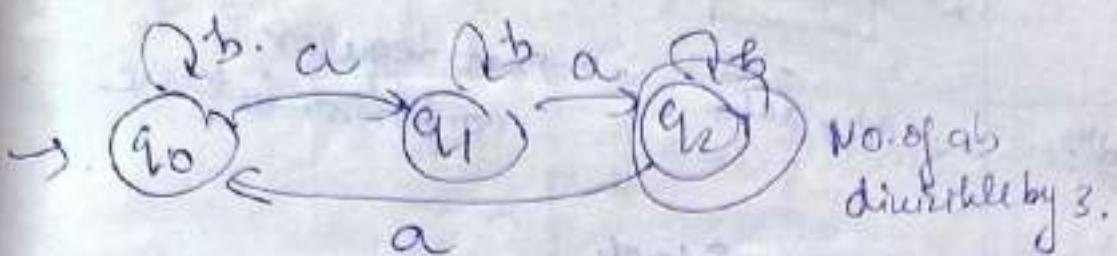
$$\bullet \frac{(a+b)(a+b) \ b \ (a+b)^*}{\longrightarrow}$$

3rd symbol from LHS is b .
i.e. of

	a	b	B	
q_0	(q_1, a, R)	(q_0, b, R)	(q_4, B, R)	do not accept
q_1	(q_2, a, R)	(q_1, b, R)	Halt	
q_2	(q_3, a, R)	(q_2, b, R)	Halt	
q_4	H	A	L	T

$a^* b a^* b a^*$

$b^* ab^* a$



Q.

	a	b	B
$\rightarrow q_0$	(q_1, a, R)	(q_0, b, R)	halt
q_1	(q_1, a, R)	(q_2, b, R)	(q_2, B, R)
q_2	u	A	L



\rightarrow If the symbol is not replaced so, it generate always regular lang. or automata.



Q) Construct the T.M. for the language

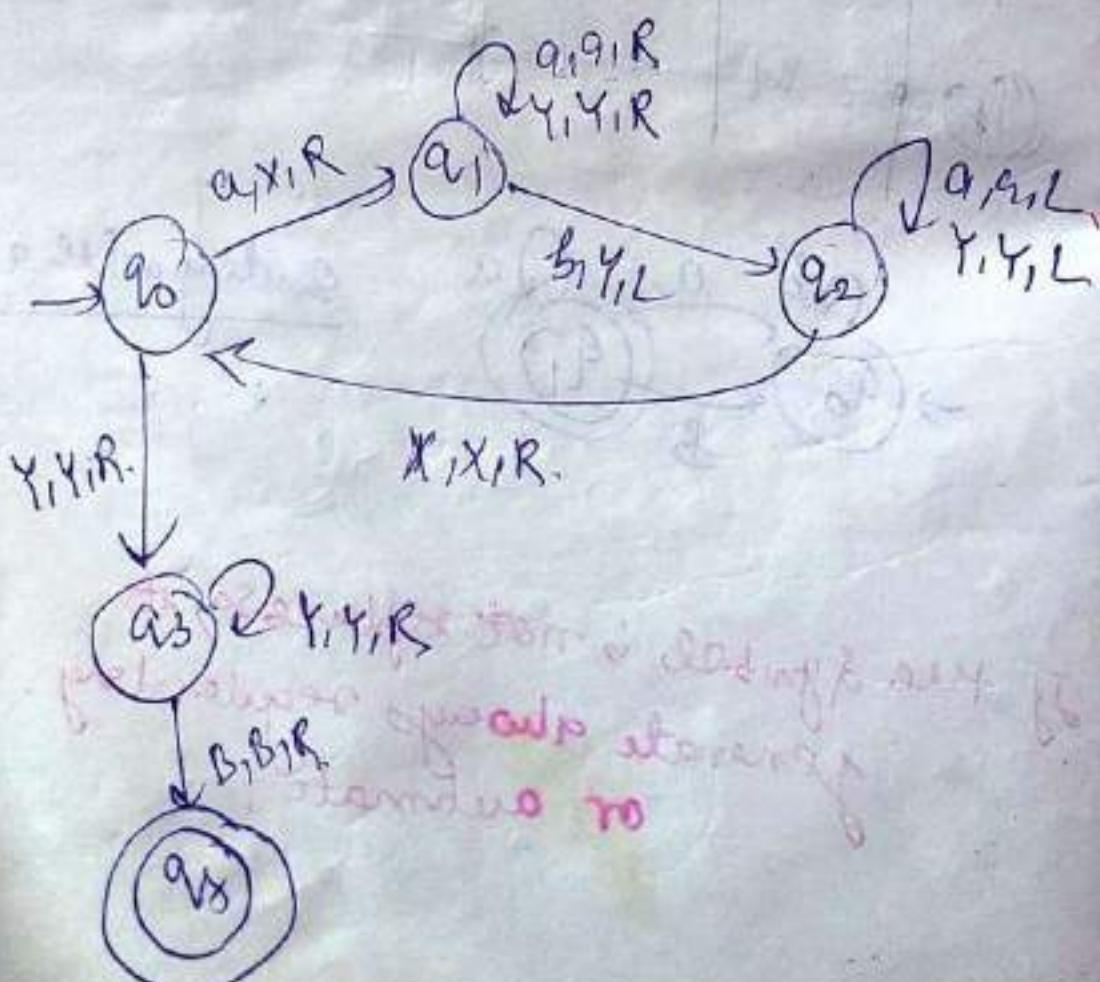
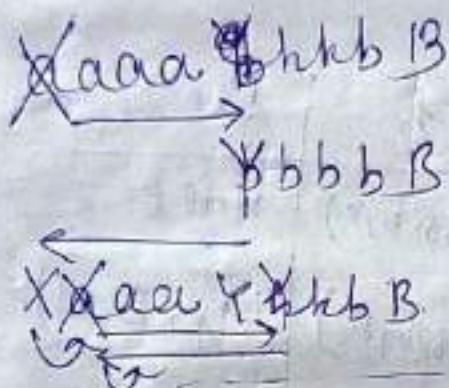
$$L = \{a^n b^n \mid n \geq 1\}$$

Soln.

ab, aabb

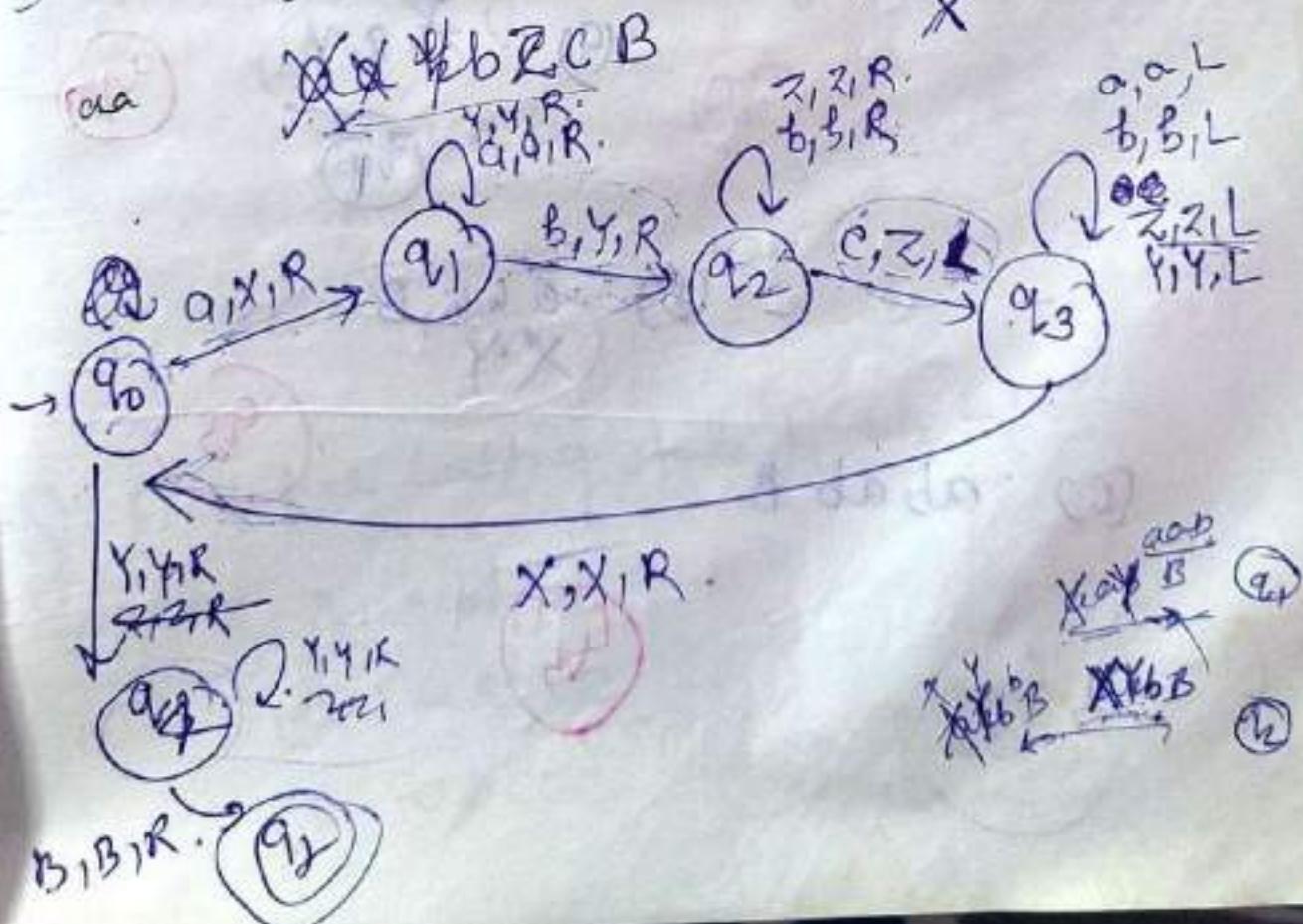
→ not regular
comparison is there

There is no stack.



	a	b	x	y	B
$\rightarrow q_0$	q_1, X, R	Halt	Halt	q_3, Y, R	Halt
q_1	q_2, a, R	q_2, Y, L	Halt	q_1, Y, R	Halt
q_2	q_2, a, L	Halt	q_0, X, R	q_2, Y, L	Halt
q_3	Halt	Halt	Halt	q_3, Y, R	q_4, B, R
q_4	← Halt				

Q2. $L = \{a^n b^n c^n \mid n \geq 1\}$ ~~XaXbZcB.~~



	a	b	x	y	z	
q ₀	q ₁ , XIR.	Halt	Halt	Halt (q ₃ , YR)		
q ₁	q ₁ , q ₁ R.	q ₂ , YIR	Halt	q ₁ , YIR Halt	Halt	
q ₂	Halt	q ₃ , b ₁ R.	Halt	Halt	q ₂ , ZIR	Halt
q ₃	q ₃					

Q. what are the halting states for the given input in the above TM of

q₁ (q₁, q₃)

- (A) ab
- (B) ab
- (C) abab

ab B
X Y
q₁

(B) ab B
X Y
q₃

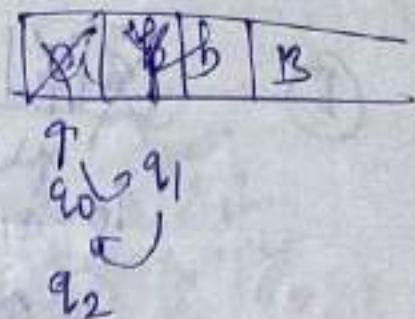
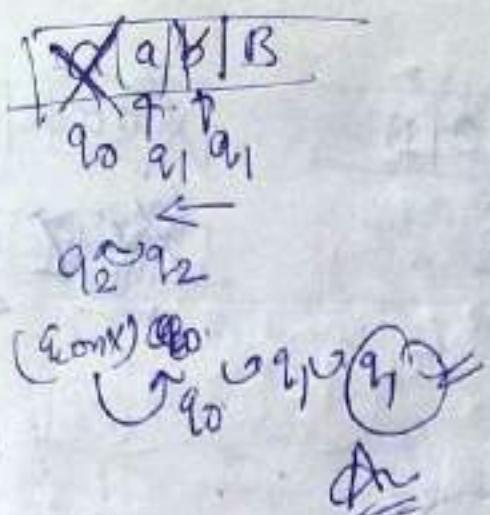
(C) abab B
q₃

q₃

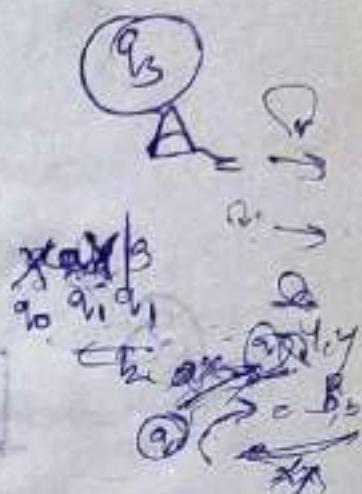
B
halt

Xhalt

halt



[a b a b] B.



Q. Halting state for baba. for q₁.

baba B

q₀

\xrightarrow{a}
no transition
so, q₀.

Q. Find the halting states for this input

①. aabbcc

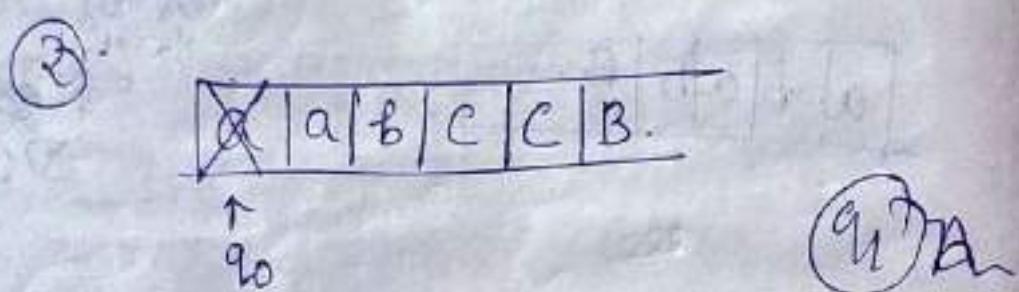
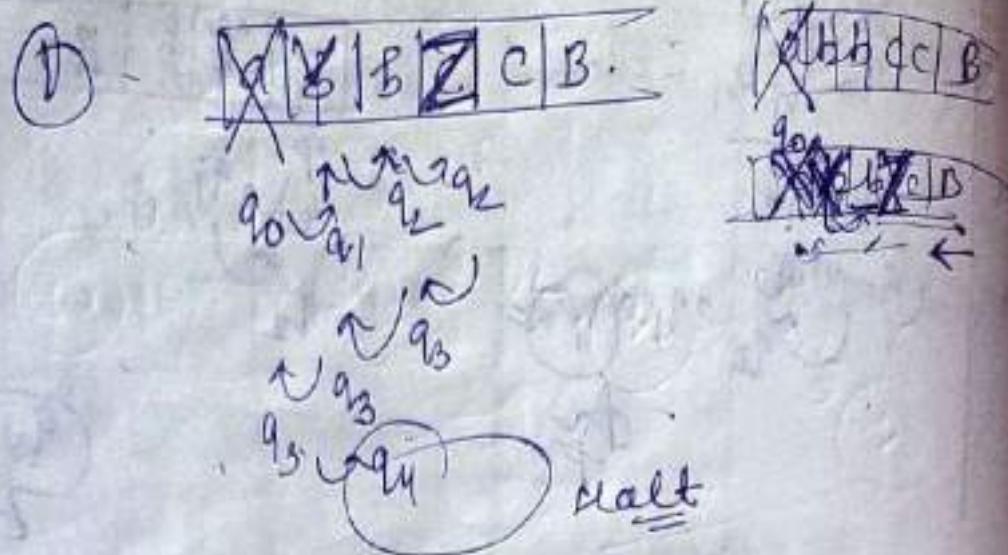
q₄

②. aabccc

q₅

③. abccee

q₆

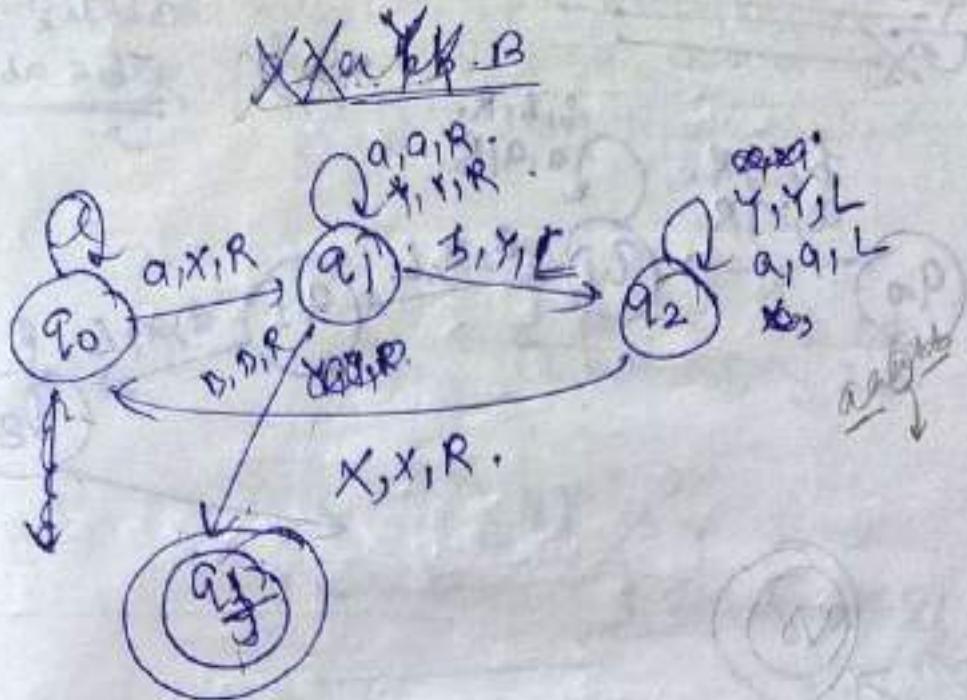


③

(a) n

Q. Is the language
Note. The language fragment is CSL

Q. construct the T.M. for the language.
 $L = \{a^n b^m \mid n \geq m\}$.

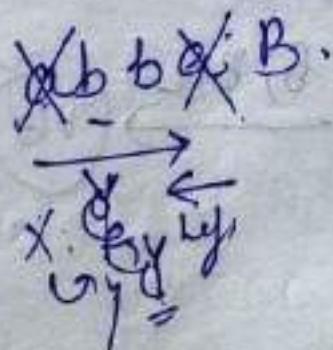


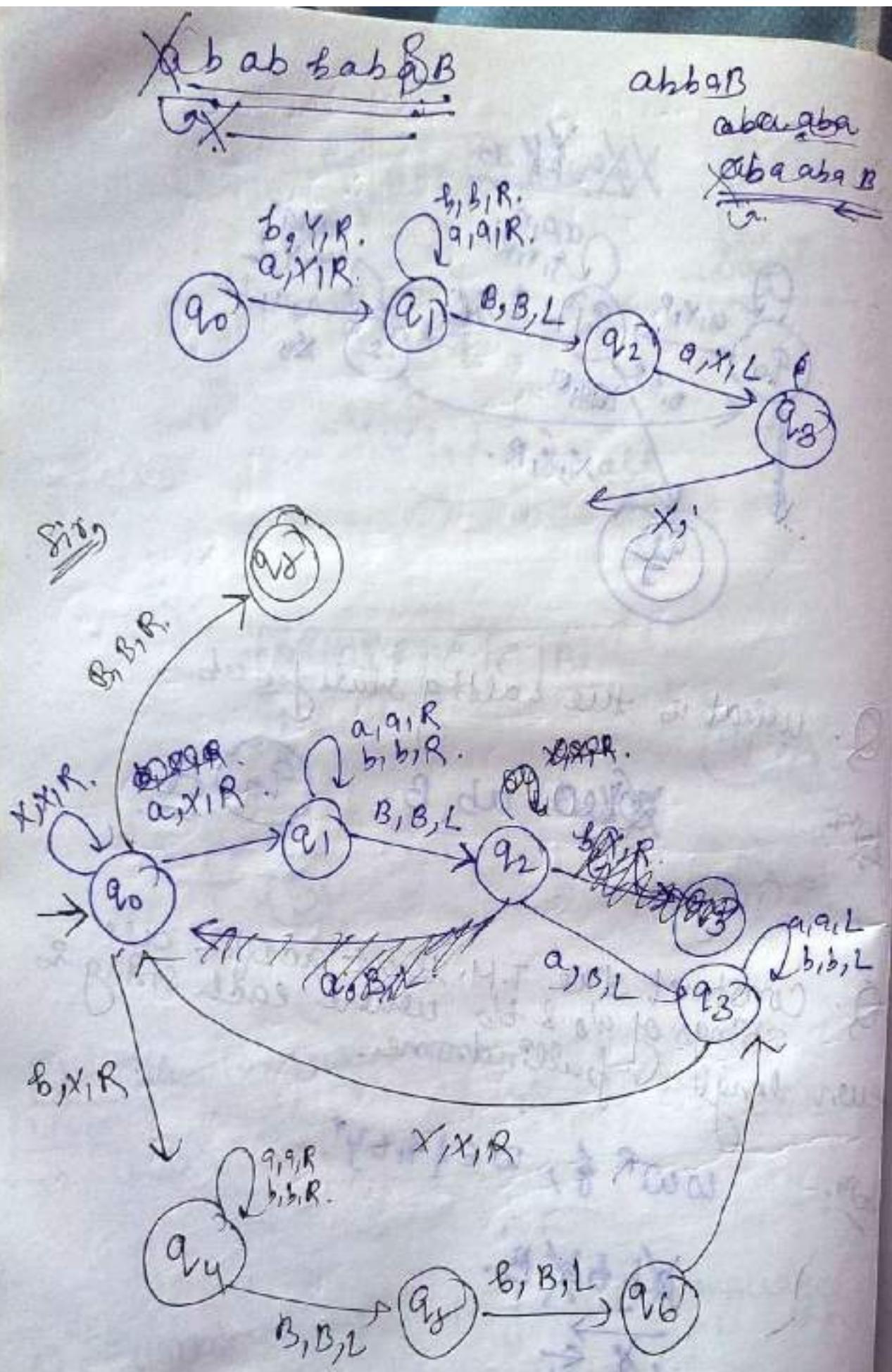
Q. What is the halting state for ab.

~~Q. 3.~~ ~~ab~~ ~~B~~, ~~ab~~ ~~B~~. ~~q0~~ ~~B~~.

Q. Construct the T.M. that accepts all strings of $a_i b_j$ where each string is even length of palindrome.

~~Q. 3.~~ $w w^R f$, $w \in \{a, b\}^*$.





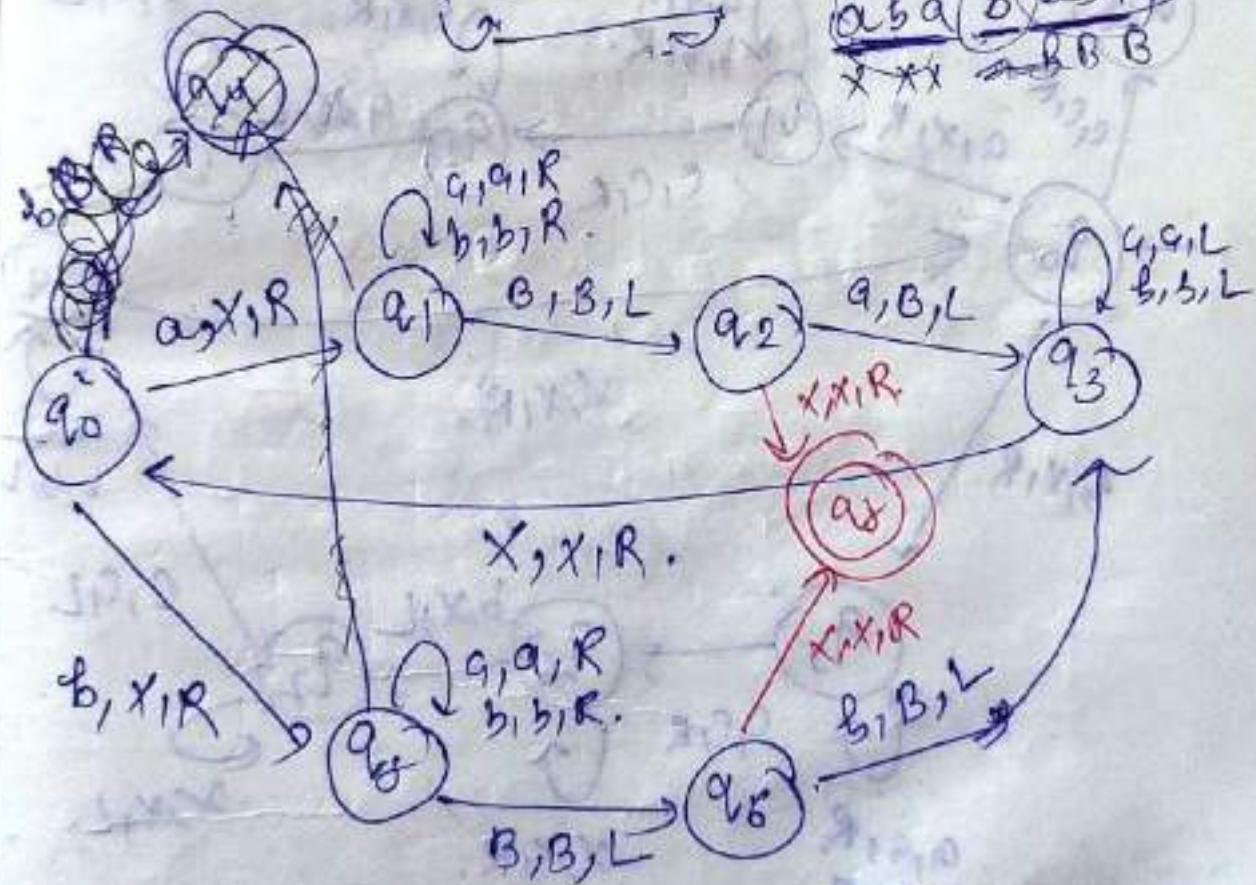
Q. construct state T.M. that accepts all strings of 9 or 8 bits where each string is odd length pattern.

Ans:

ab aba baba

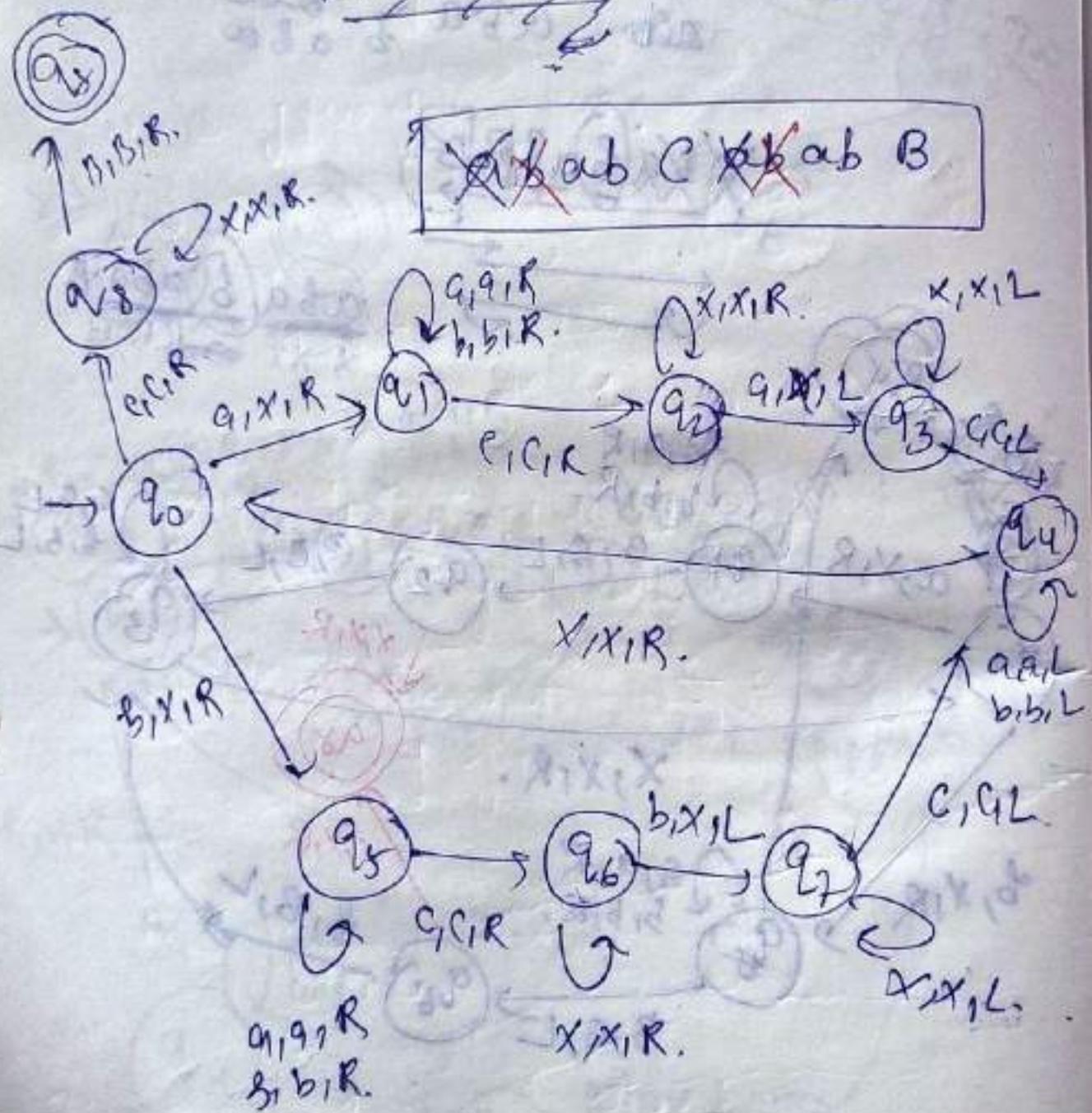
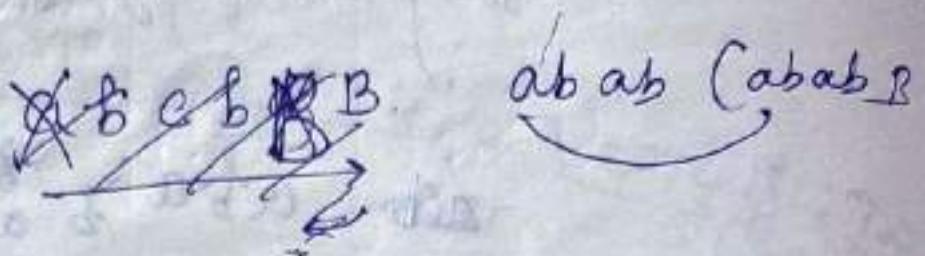
~~babababB~~

~~babababB~~



ab ab ab B

Q. Construct the TM for the language
 $L = \{www | w \in \{a,b\}^*\}$



~~X/X/R & b, b/R~~ $\Rightarrow q_4$

Q. Find the halting states for the foll types -

- ① abca
- ②. aabcbb
- ③. abcabab

~~X~~ ~~a₁ G a₂~~
a₀ a₁ a₂ a₃
a₀ → a₁ ← a₂
a₃ a₄ a₅ a₆.

~~X~~ ab c bb.

~~X~~ ab c bb ab
a₃.

(a₈)

ab [e] ab [a] b

GATE

Q Consider the following T.M. &

	0	1	B
q_0	$(q_1, 1, R)$	$(q_1, 1, R)$	Halt
q_1	$(q_1, 1, R)$	$(q_0, 1, L)$	(q_0, B, L)

110
110

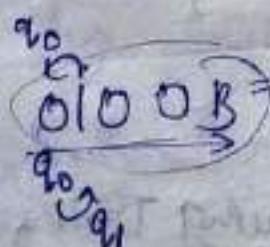
which of the foll is true?

(a) T.M. halts on all inputs ending with 0. $\{(01)^* 0\}$

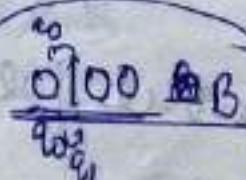
(b) T.M. halts on all inputs ending with 1. $\{(01)^* 1\}$.

(c) T.M. does not halt any input string of $(01)^*$.

(d) T.M. does not halt on any input string of $(001)^*$.



0100 B.



0100

00

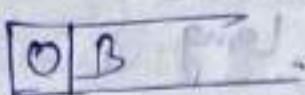
$(01)^*$

Q104B

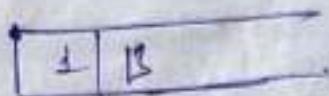
(a)

(b)

(a)



(b)



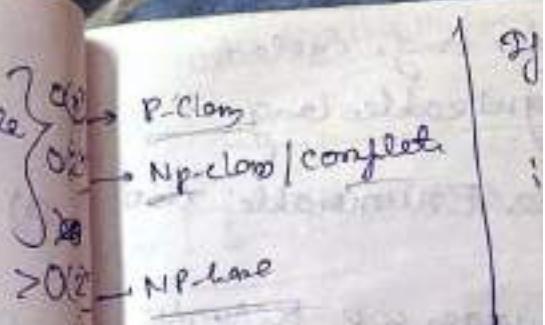
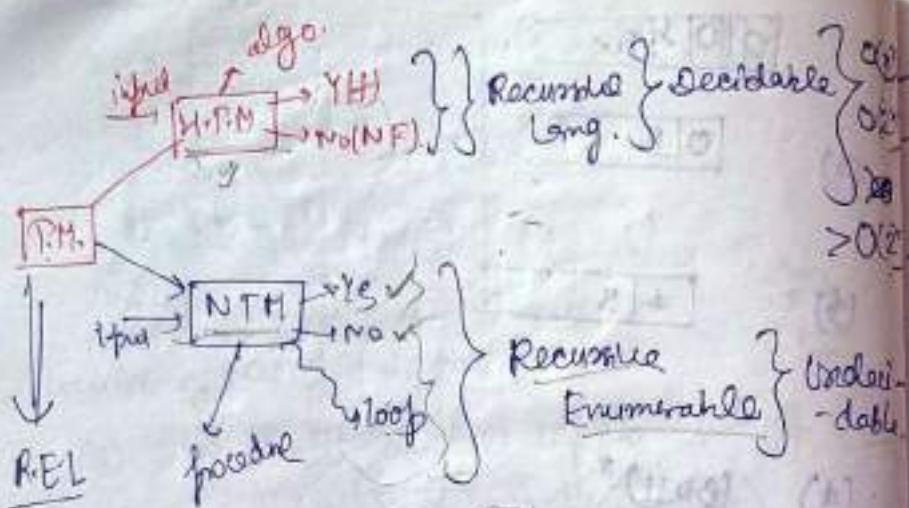
(d)

(0 or 1)

it halts on ϵ .

but the given automata
is not halt on ϵ .

	0	1	B
q_0	$(q_1, 1, R)$	$(q_1, 1, R)$	(q_1, B, R)
q_1	$(q_1, 1, R)$	$(q_0, 1, L)$	(q_0, B, L)
(a)	H	A	T.



on see
invalid string,
non-final state
→ The halting T.M. is
hence, recursive.

→ Recursive Long. &
decidable (L)

Recursive

Recursive language:-

Definition (① Detection)

A lang 'L' is said to be recursive

→ A lang. 'L' is recursive if for that lang. 'L' for that lang. inputs and (one) inputs. that is, T.M. halt on fin.

→ String is invalid fall into

- $O(n)$ → P-Clm
- $O(2^n)$ → NP-Clm / complete
- $O(2^n)$ → NP-hard

If there exist some T.M.

from that lang.

that halts on all

inputs. i.e., that is

the string is valid

string T.M. halts

final state, string is

the T.M. halts on

on the
invalid string,
non-final state.

- The halting T.M is equivalent to algorithm.
Hence, recursive languages are decidable.
- Recursive lang. also called as Turing
decidable lang.

Recursive Enumerable lang:-

- A lang. L is said to be recursive enumerable if there exist a T.M. for that lang. that halts on some inputs and may not halt on some inputs. that is, string is valid string. T.M. halts on final state.
- String is invalid T.M. may halt or may fall into infinite loop.

→ Recursive Enumerable lang. called as Turing Recognizable lang.

→ The lang. is recursive. Enumerable that is undecidable.

→ All recursive languages are Recursive Enumerable but all Recursive Enumerable need not to be recursive.



→ By default P.M. means may or may not halting \rightarrow turing machine

$$S \rightarrow aSb \mid bSa \mid a \mid b.$$

$$S \rightarrow aSb \mid bSa \mid a \mid b.$$

a
b
S



variable Uniqueness Testing :-

$$①. S \rightarrow ss/a \rightarrow A.b(aaa) X_1$$

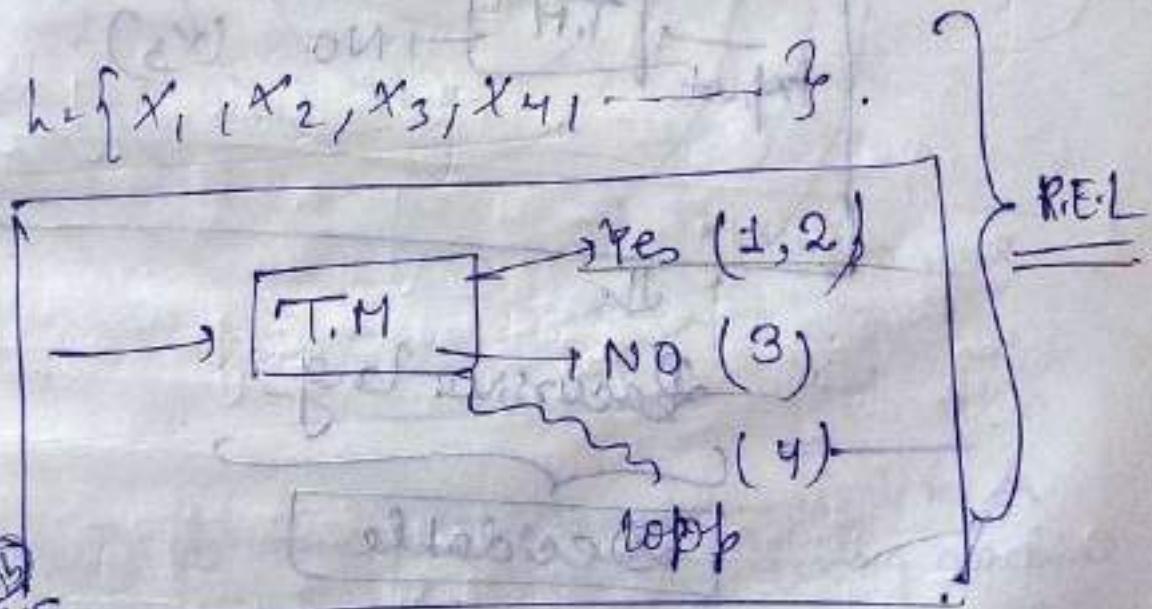
$$②. S \rightarrow abab/ssas/e \rightarrow abab X_2$$

$$③. S \rightarrow AB \\ A \rightarrow a \\ B \rightarrow b \rightarrow u.v X_3$$

$$④. S \rightarrow abb/bsa/a/b \rightarrow u.v.w X_4$$

R
RCA

$$h = \{X_1, X_2, X_3, X_4, \dots\}$$



↓
Undeletable

$$S \rightarrow ss/a \quad S \rightarrow abab/ssas/e$$

$$S \rightarrow abb/bsa/a/b$$

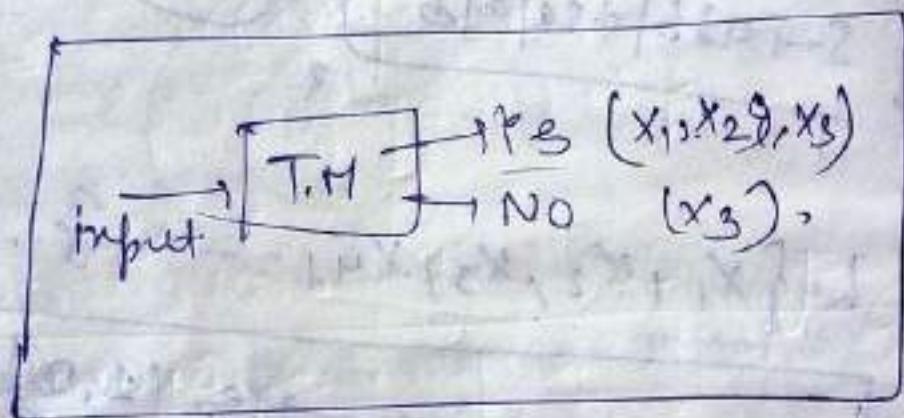
Principle
String :-

$S \rightarrow SS|S$ x_1'

$S \rightarrow aS|bS$ x_2

$S \rightarrow aS|bSa|bS|a$ x_3

$S \rightarrow aS|bSa|bSSa|bS.$ $x_4.$



Recursive lang

Decidable

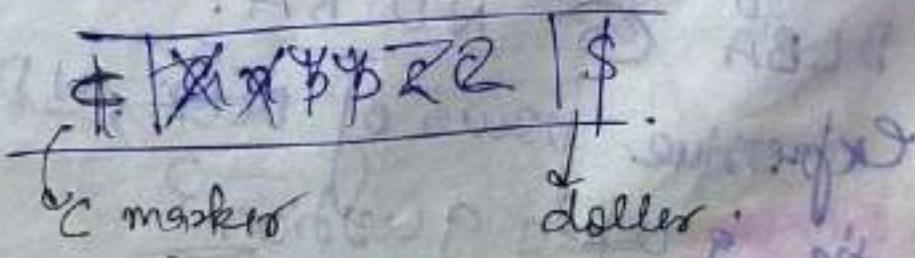
Undecidable

Chomsky hierarchy :-



LBA (Linear bounded automata) → is like a turing machine.

But in this the tape is bounded
So, we called linear bounded automata.



LBA is nothing but a T.M. having boundary on left as well as right side. direction that is T.M. cannot move before the symbol \$ and after the symbol \$, and LBA having turnaround capability and read-write capability:

The lang. accepted by LBA are known as CSL.



ex: $\{a^n b^n \mid n \geq 1\}$

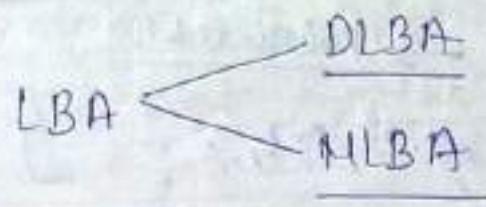
$\{www \mid w \in \{a,b\}^*\}$

E is not allowed in the CSL languages.

Even though E may be allowed in the context sensitive grammars in order to maintain Chomsky hierarchy.

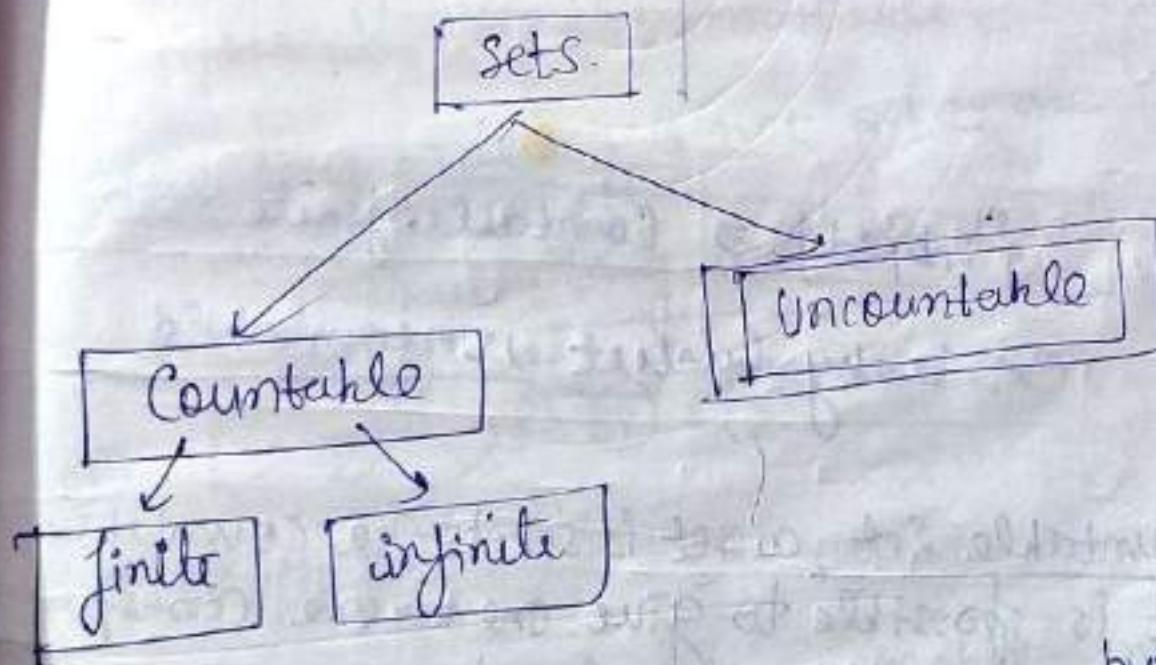
Types of LBA :- only one type \rightarrow Lang. recognizer

for every CSL we can construct DLBA \otimes NDLBA. But the expressive power of DLBA, NDLBA is a open question?



→ There are uncountable no. of lang. for which T.M.
not possible.

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



River (no. of liters of water) \rightarrow countably infinite
 No. of hair on head? Countable ~~finite~~ but very huge.

countable
finite

discrete
continuous

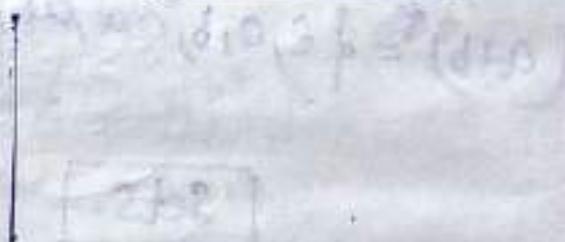
~~Ex:~~ Set of real no.'s \rightarrow Uncountable

$$= \{0.1, 0.2, 0.3, \dots\}$$

infinites
no. of sets

0.1 0.2
square
can't all be glued. \rightarrow natural no's

No. of points in a line \rightarrow Uncountable



$\{a, b, c, d, \dots\} \rightarrow$ Countable finite

Ex: leaky bucket \rightarrow Uncountable

✓ Countable set, a set is said to be countable if it is possible to give one-to-one correspondence with natural no's.

Ex: Complete lang.

Ex: Set of natural no's

Ex: Total population of the world after 100 years.

• Integers

• Rational no's

Uncountable set, a set is said to be uncountable if it is not possible to give one to one correspondence b/w natural no.s.

Ex: set of real no.s.

Ex: set no. of point in a line

Ex: power set of complete lang

leach bucket

Text → Irrational no.s.

• Set of lang accepted by T.M. is countable, see set of uncountable lang. are uncountable. i.e., the no. of lang. for which T.M. is not possible is called Uncountable.

• The lang. which is not R.E.L one
indecidable:-
see following lang. are not R.E.L:-

①. $L = \{ \text{01}^n 2^n \mid n \in \mathbb{N}\} \cap P(\Sigma^*)$.

②. Set of real no.s.

Q. Let Σ be non empty alphabet and $2\Sigma^*$ is power set of Σ^* then which of the following one true:-
give following one true:-

- ①. Both $2\Sigma^*$ and Σ^* are countable.
②. $2\Sigma^*$ is countable & Σ^* is uncountable.

- ~~③. Both Σ^* and $\mathbb{Q}^{\mathbb{N}}$ are uncountable~~
- ~~④. $\mathbb{Q}^{\mathbb{N}}$ is uncountable & Σ^* is countable.~~

Note:-

~~Id, diagonalization lang. is not REL
Hence there is no T.M. exist for
that lang.~~

~~→ RPL & not REL both are indeedable
lang.~~

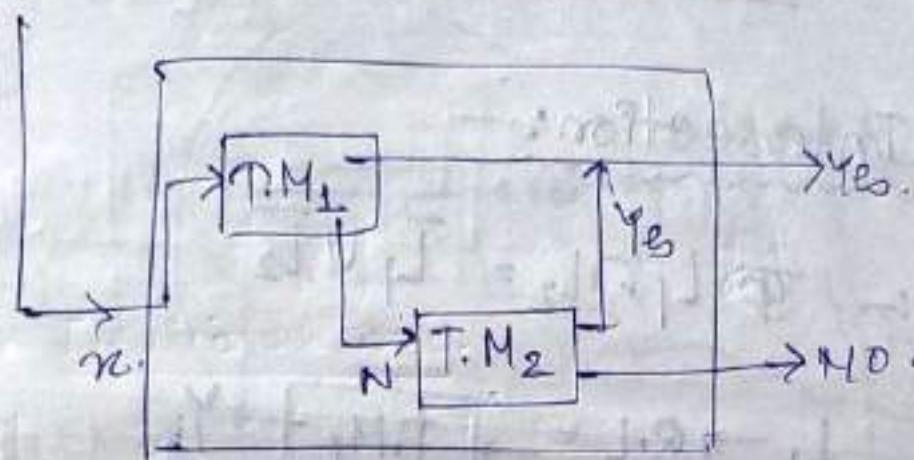
Close properties of Recursively Lang and Recursively Enumerable Language:-

① Union operation:-

For Recursive Lang $L_1 \rightarrow$ Recursive Lang \leftrightarrow $\boxed{\text{H.T.M.}} \xrightarrow{\text{Supt}} \text{NO}$

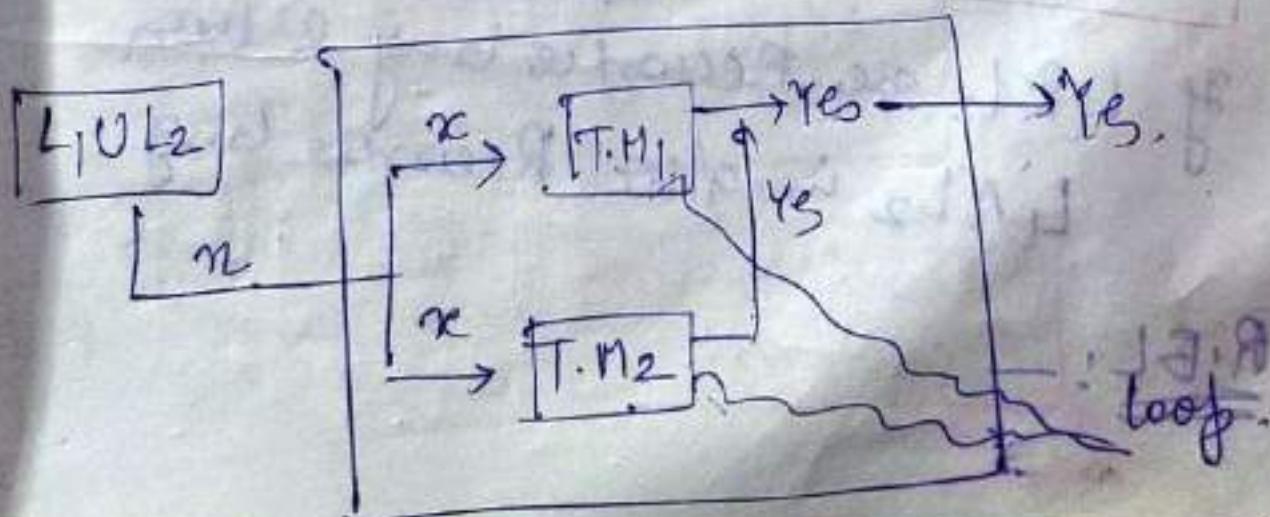
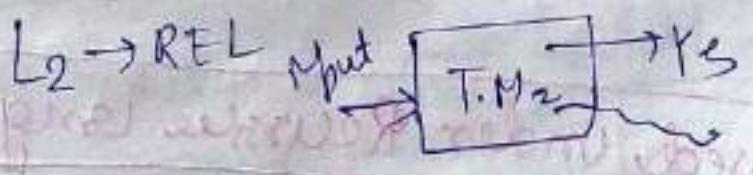
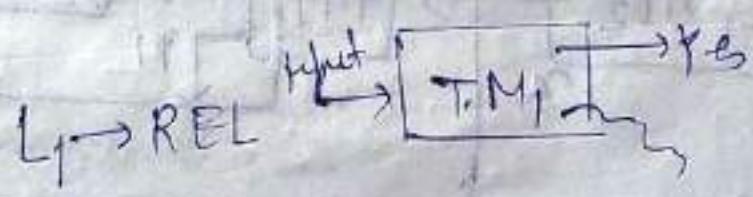
$L_2 \rightarrow$ Recursive Lang \leftrightarrow $\boxed{\text{H.T.M.}} \xrightarrow{\text{NO}} \text{NO}$

$L_1 \cup L_2$



L_1 & L_2 are Recursive languages. Then $L_1 \cup L_2$ always Recursive. Hence, Recursive Lang. are closed under Union operation.

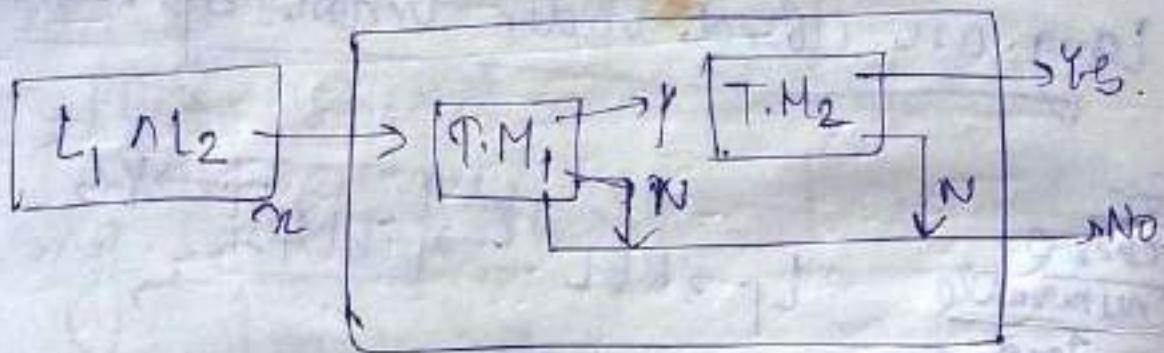
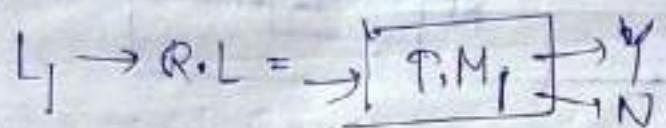
For
Recursive
Enumerable
Lang. \rightarrow



→ Both Recursive and REL lang. are
Closed under union operation.

② Intersection:—

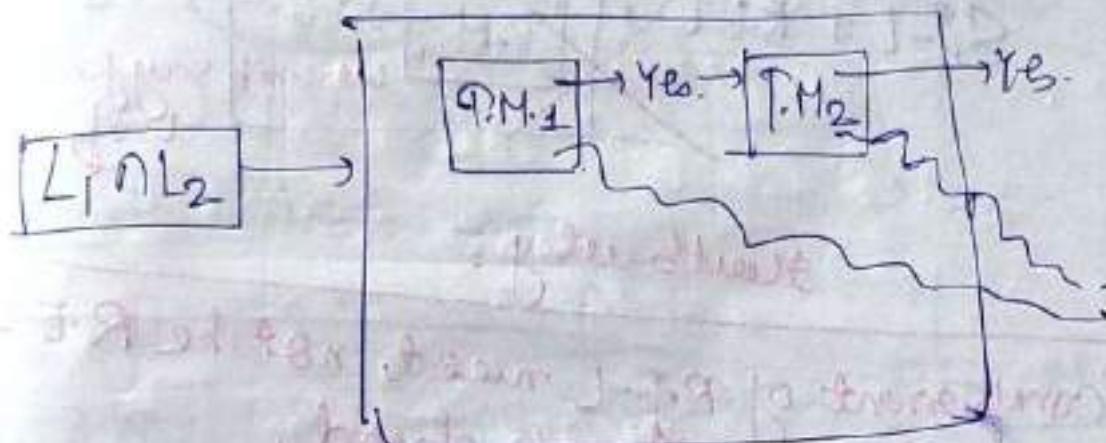
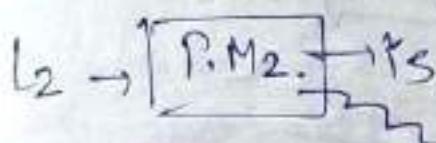
Recursive:— $L_1 \cap L_2 = \overline{L_1} \cup \overline{L_2}$



Closed under Recursive lang

If L_1 & L_2 are Recursive lang & then
 $L_1 \cap L_2$ is also Recursive lang.

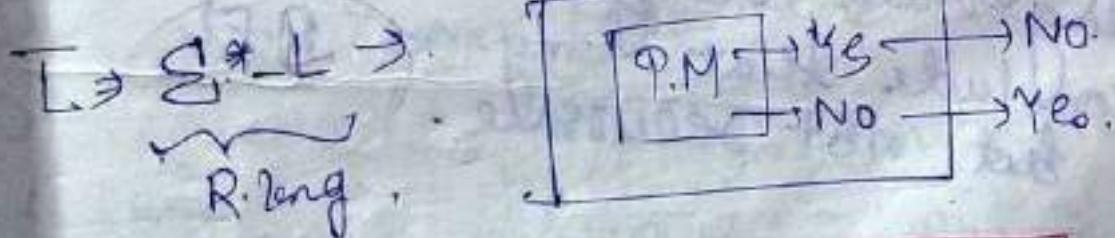
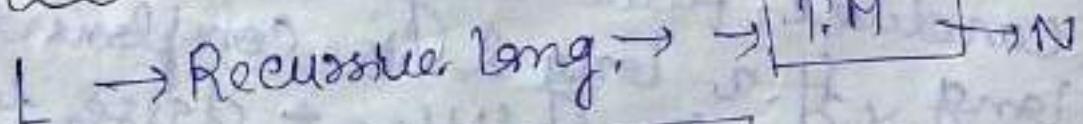
R.E.L:—



L_1 & L_2 are R.E.L then $L_1 \cap L_2$ are also
R.E.L

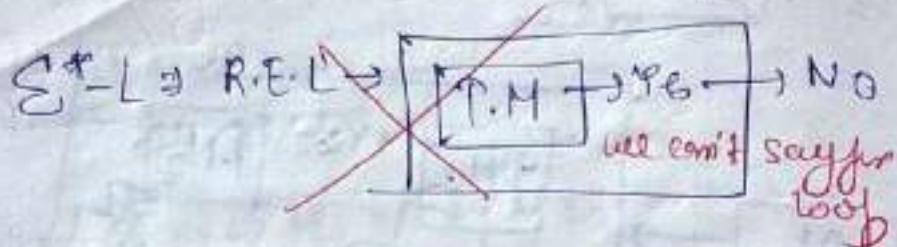
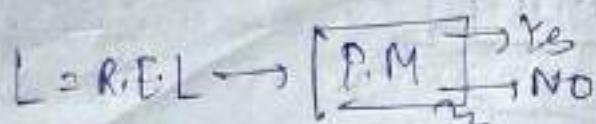
Closed under R.E.L.

Complement: —



Recursive lang. closed under complement.

R.E.L \Rightarrow

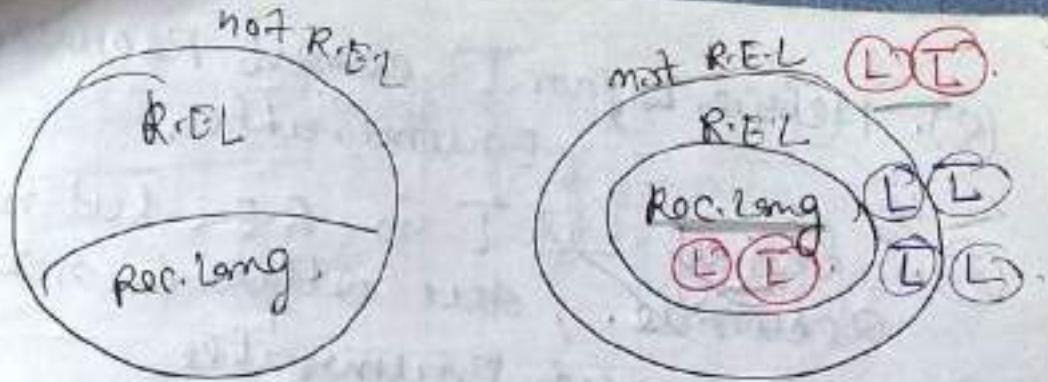


therefore

complement of R.E.L need not be R.E.L
hence, not closed.

Note:- Complement operation is used as a tool to differentiate Recursive languages and Recursive Enumerable lang. If it is a lang. & its complement falling under the same class we can conclude that lang as Recursive. If a lang & its complement falling under different classes we can conclude that lang. as R.E.L but not recursive.

→ For the pair of complementary langs L & \bar{L} the foll are true



Possibilities:-

- ①. Both L & T are not R.E.L.
- ②. Both L & T are Recursive. Then L should be Recursive.
- ③. One of L & T is R.E.L. but not Recursive. Then the other is not R.E.L.

because even recursive R.E.L.

Q. Which of the following is true:-

A long L & T is its complement, which of the following is not possible,

- ①. Both L & T are Recursive lang.

- close
- (Q). Neither L nor T is Recursive Enumerable.
 - (3). One of L & T is R.E.L but not Recursive, the other is not.
 - (2). Recursive Enumerable
 - ~~(1).~~ Both L & T Recursive B.L but not Recursive.

2013

- (Q). Which of the foll. Stmt is false \rightarrow
 - (1). For every NDTM, there exist an equivalent DTM.
 - (2). Turing recognizable lang. are closed under Union & Intersection.
 - (3). Turing decidable lang. are closed under Intersection & complementation.
 - ~~(4).~~ Turing Recognizable lang. are closed under Union & Complementation.
- close

Operation	Reg. Lang.	DEFL	CFL	CSL	Rec Lang	R.B.L
① Union operation	✓	✗	✓	✓	✓	✓
② Concatenation operation	✓	✗	✓	✓	✓	✓
③ Intersection operation	✓	✗	✗	✓	✓	✓
④ Complement operation	✓	✗✓	✗	?✓	✓	✗
⑤ Difference operation	✓	✗	✗	✗✓	✓	✗

⑥ Difference with Regular	✓	✓	✓	✓	✓	✓
⑦ Intersection with Reg	✓	✓	✓	✓	✓	✓
⑧ Kleene closure	✓	✗	✓	✓	✗	✓
⑨ plus closure	✓	✗	✓	✓	✗	✓
⑩ Reversal operation	✓	✗	✓	✓	✓	✓
⑪ No		✗				
⑫ empty		✗				

⑪. <u>Substitution</u>	✓	✗	✓	✓	✗	✗
⑫. <u>Homomorphism</u>	✓	✗	✓	✓	✗	✓
⑬. <u>Principle homomor- phism</u>	✓	✓	✓	✓	✓	✓
⑭. <u>Unit</u>	✓	✓	✓	✗	✗	✗
⑮. <u>Cycle</u> <small>(ablette for lone leaf)</small>	✓	✗	✓	✓	✗	✓
Subset	✗	✗	✗			

Q. L_1 is Reg. lang, L_2 is DCFL, L_3 is REL
but not Recursive then which
of the foll is false?

- (1). $L_1 \cap L_2$ is DCFL
- (2). $L_1 \cup L_2$ is CFL*
- (3). $L_1 \cap L_3$ is Recursive lang.
- (4). $L_1 \cap L_2 \cap L_3$ is REL. \Rightarrow REC, {REC}

Q. L_1 is Recursive lang, L_2 is RE-L but not
Recursive then which of
the foll is true.

- (1). $\overline{L_1}$ and $\overline{L_2}$ is Recursive Enumerable.
- (2). $\overline{L_1}$ is Recursive and $\overline{L_2}$ is
recursie enumerable.
- (3). $\overline{L_1}$ is RE-L but not
recursive
& $\overline{L_2}$ is not RE-L.
- (4). None of these.

Decision properties of R.L

R.E.L:-

The following problem are undecidable

for R.L & RBL's:-

①. Emptiness problem

②. Finiteness problem.

③. Equivalence problem

④. Regularity problem.} checking whether

T.M. accepts lang. Reg. lang

or not ?

⑤ Membership problem:- checking

whether given string is accepted by
given T.M. or not.

→ Membership problem is decidable for
Recursive lang. and Undecidable
for Recursively enumerable lang.

T.M. as O/P generator:-

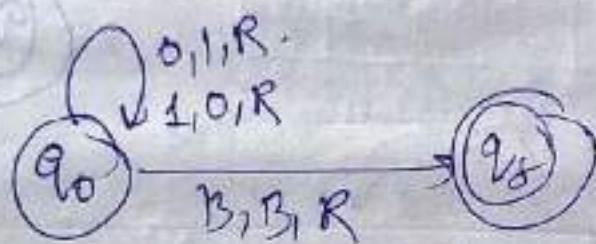
Q. Construct the T.M. that produces 1's complement of given binary no. as output.

input

0	1	0	1	B	B
---	---	---	---	---	---

output

1	0	1	0	B	B-
---	---	---	---	---	----



* Any T.M. in the world require atleast 2 states.

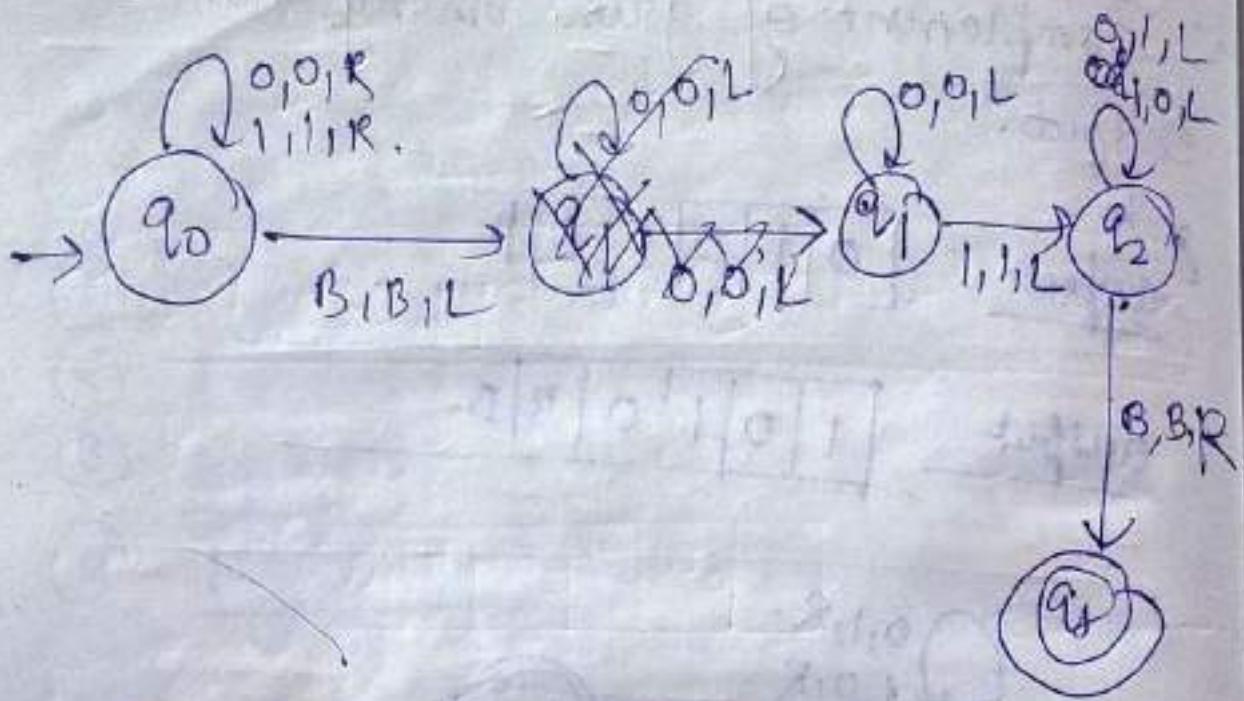
Q. Construct the T.M. that produce 1's complement of the given binary no. as output (Assume that T.M take input from both sides)

input

0	1	0	1	B	B
---	---	---	---	---	---

output

$\rightarrow \underline{S \ 0 \ 1 \ | \ 0 \ 0 \ 1 \ | \ 0 \ 3} B.$



Spis	$x+y$	$n!$
$x+3$		$3!$
$3+y$		$3!$
$x+3$		$(-2)!$
$y+2$		$(-3)!$
\vdots		{ no comb }
17		
halts		it does
on		not halt
all		on all
input		input variables
total score		
partial scores		

fun fn performed by T.M. are classified into three types

- ①. Total Recursive fn
- ②. Partial Recursive fn

- A fn is said to be Total Recursive if there exist a T.M. for that fn that halts on all fn values.
- Total Recursive fn's are similar to Recursive languages
ex: addition fn
Subtraction fn

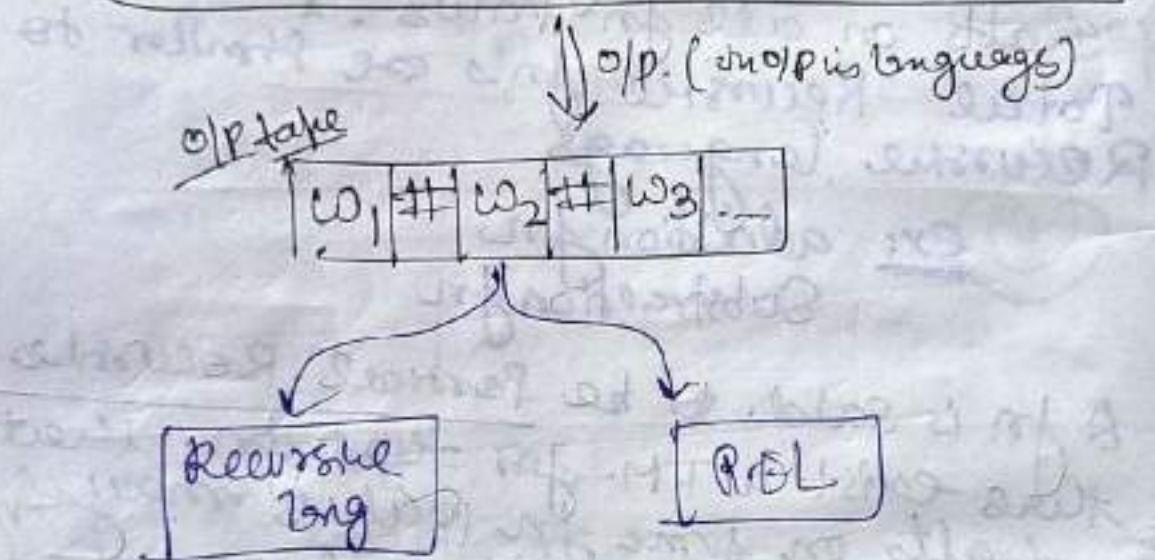
- A fn is said to be Partial Recursive if there exist a T.M. for that fn. that halts on some fn values may not halt on some fn values.
- Partial Recursive fn are similar to R.E.L.
Ex: (m) factorial fn

T.M. as Language Generator
(or) Enumerator

There is no input, only output tape
is there \Leftrightarrow output is available.

Ex: `for [i=1; i<=100; i++] {
 cout << "Boss"; } }` without
any input.

T.M as Lang generator \Leftrightarrow T.M as Enumerator



→ Enumerator is a T.M, having O/P tape
& no input tape whenever \Leftrightarrow
enumerator is on, the T.M keep
on produces string on the O/P tape
where each string is separated by a
symbol, the lang. produced by
Enumerator are exactly equivalent to languages accepted

by T.M hence Enumerator produces
Recursive lang. and R.E.L.

Recursive lang.: — lang. produced by Enumerator set to be Recursive if the strings of the language are in alphabetical order or Lexicographic order or decreasing order or increasing order.

R.E.L.: — a lang. produced by Enumerator said to be recursive enumerable the strings of that lang. are arranged in alphabetical order or Lexicographic order or any order.

→ Every Recursive lang are R.E.L But R.E.L are not recursive here also.

for($i=1$; $i \geq 1$; $i++$)
 print("Boss")

for($i=1$; $i \leq 100$; $i++$)
 print("Boss")

R.E.L and A

Q. If the strings produced by enumerator are effectively enumerated in lexicographic order.

- (1). REL but not R.E
- (2). not REL
- (3). RE & R.E
- (4). None

means
increasing
length
order
 $(\epsilon, 0, 1, 00, 10,$
etc.)
increasing
length

Undecidability

Decidable

A problem is said to be decidable, if there exist halting T.M. for that problem or there exist algorithm to solve the problem.

Undecidable

A problem is said to be undecidable

there is no T.M. for that problem
⑥ no algorithm to solve that problem (Even though T.M. exist it is not halting T.M.).

- The Recursive lang. are decidable.
→ The R.Enumerable \Leftrightarrow not R.E.L
are Undecidable.

(decision property are also the decidability property)

- Travelling salesman problem \rightarrow decidable.
(algo. exist).
→ Sorting \rightarrow decidable.
→ Searching \rightarrow decidable.
→ Cyclic algo

Note:-

The following problems are decidable \Rightarrow

- ①. Travelling Sales person
- ②. Matrix multiplication
- ③. Membership problem (Cyclic algo)
- ④. Sorting problem

⑤. Semiring problem

The following problems are undecidable problems →

①. format's equation

$$x^n + y^n \neq z^n, n \geq 3 \quad (\text{undecidable problem})$$

What is the value of x, y, z satisfy this eqn. not able to find out the soln.

②. Post's Correspondence problem

③. Ambiguity problem

~~Reduction~~

CoC & Ambig

$A \propto B$
(know) (new)

DFA

$S \rightarrow a|bs|a|b$
 $A \rightarrow ab|ba|c$

$B \rightarrow$

Reduction Technique



represent reduction.

→ A problem A is reduced into B means we can conclude the problem B with the help of problem A.

→ If A is known problem and B is a new problem then

A $\not\propto$ B
(known) (new)

①. A is UD, B is also UD. [UD \rightarrow UD]

②. B is D, A is also D. [D \leftarrow D]

③. A is D, we cannot conclude about B. [D \rightarrow X]

④. If B is UD, we cannot conclude about A. [X \leftarrow UD]

→ Aeroplane
janda
ki
likin

→ Aeroplane
ke parts ko janda kia kaise ki
ve parts ni janda kia kaise ki
ve parts ki

→ To perform reduction at least one known problem is required.

→ $A \propto B \propto C \propto D$

If A is UD given B is UD, C is UD and D is UD.

Q Let P_1, P_2, P_3 are 3 problems &
it is known that P_1 is decidable,
 P_2 is UD which of the foll is
true?

① If P_1 reduces to P_3 , P_3 is
decidable.

② P_3 reduces to P_2 , P_3 is
UD.

③ ~~P_2~~ P_2 reduces to P_3 , P_3 is UD.

④ P_3 reduces to $\overline{P_2}$, then P_3
is D.

Post Correspondence

Post Correspondence

problem :-

$$\text{ex. } \textcircled{1} \quad A = \{ 1, \begin{matrix} 1 \\ 2 \end{matrix}, 10111, \begin{matrix} 1 \\ 2 \end{matrix} \} \\ B = \{ 111, \begin{matrix} 1 \\ 2 \end{matrix}, 10, \begin{matrix} 1 \\ 2 \\ 3 \end{matrix}, 0 \}$$

$$\begin{array}{c} 213\alpha \\ 10111110 \\ 101110 \end{array} \left. \begin{array}{c} 312\alpha \\ 10110111 \\ 011110 \end{array} \right\} \begin{array}{c} 132\alpha \\ 10110111 \\ 011110 \end{array} \left. \begin{array}{c} 321\alpha \\ 10110111 \\ 011110 \end{array} \right\} \begin{array}{c} 321\alpha \\ 10110111 \\ 011110 \end{array}$$

$$\begin{array}{c} 2113 \\ 10110000 \\ 1011110 \end{array} \left. \begin{array}{c} (so)^\eta \\ (so)^\eta \end{array} \right\} \checkmark (so)^\eta$$

$$\text{ex. } \textcircled{2} \quad A = \{ bbab, ab, baa, b \} \\ B = \{ a, abb, aa, bbb \}.$$

ex: $A = \{bbab, ab, ba, b\}$

 $B = \{a, abb, aa, bbb\}$

End ke
songar
tas
post founder
poker

4 1

bbbab

bbba

2143

A

(abbabbba)

(ahhb@ahhb@aa)

2143.

ex(3) $A = \{ba, ab, a, baa, b\}$

$B = \{bab, baa, ba, a, aba\}$

vs area

1 5 2 3 4 4 3 5 3 1 5

b abababaab
babababaa
baaa

815

1 5 2 3 4 4 3 4 ✓ A

babababaaaabaa

bab ababaa ba@aa@ba@aa

babab
bababab
bab
bababab
bab
bababab

Ex: ④	$A = \{aaa, bbb, aab\}$	$1 \quad \quad 2 \quad \quad 3$	
	$B = \{abx, bbb, b^y\}$		

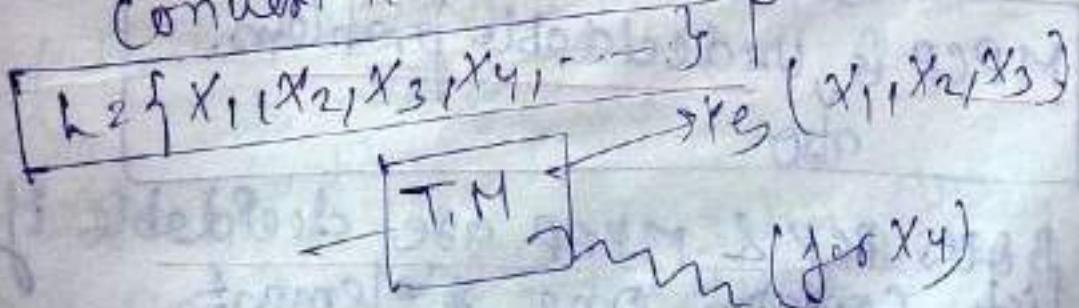
We can't generate 3 consecutive ab's
in B.

b3bb
bbb bbb

No solution possible

- Let ex: ① $\rightarrow x_1$ (halt)
ex: ② $\rightarrow x_2$ (halt)
ex: ③ $\rightarrow x_3$ (halt)
ex: ④ $\rightarrow x_4$. (loop).

Convert it into the language,



Post correspondence problem is undecidable problem because no algorithm exist.

Modified Post Correspondence problem

(modified PCP) :-

→ The difference b/w PCP and MPBP is some is required to start with 1st string of each list in MPBP.

→ MPCP & PCP
Hence, if PCP is decidable, MPCP also decidable.
if MPCP undecidable, PCP also undecidable.

→ MPCP is undecidable problem.
also

→ Both PCP & MPCP are decidable if list contains one element.

$$\left(\begin{array}{l} \text{PCP} \times A, \text{c} \\ \text{U.D.} \\ \text{PCP} \times X \\ \text{U.D.} \end{array} \right)$$

Undecidability of formal language:-

① Intersection Empty problem is $L_1 \cap L_2 = \emptyset$?

→ means

Checking whether intersection of given two languages is empty or non-empty.

→ To see this problem is decidable Intersection operation should be closed and Emptiness algorithm should be available.

→ Emptiness problem is reduced into Intersection empty problem
(that is to solve this problem we are taking the help of Intersection emptiness problem).

→ For Regular languages Intersection empty problem is decidable; for DCFLs Intersection empty problem is undecidable.

$$L_1 \cap L_2 = \emptyset$$

If emptiness is UD then
UD.

Emptiness is D then D.

→ For regular lang decidable, Recursive

regular	→ decidable
CPL	→ UD
DCPL	→ UD
CSL	→ UD UD
RBL	→ UD.

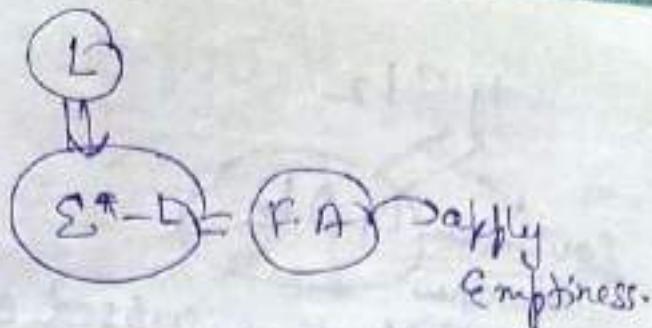
② Completeness problem:-

$$\text{if } L = \Sigma^*, ?$$

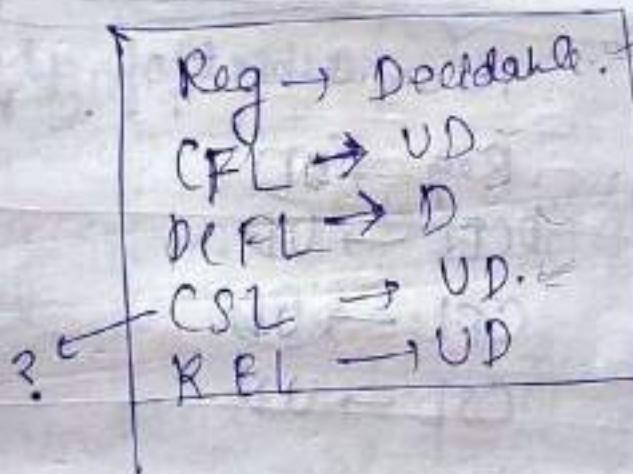
→ Means to checking whether given lang is complete lang or not.

→ This problem is also known as Totality problem.

Reg. L



If $\Sigma^* - D$ is closed, Express decidability
so, it is decidable.



③ Subset problem :- is $L_1 \subseteq L_2$?

Checking whether L_1 is subset of L_2 or not.

Reg \rightarrow	UD
DCFL \rightarrow	UD
CFL \rightarrow	UD
CSL \rightarrow	UD
R.L \rightarrow	UD
R.E.L \rightarrow	UD

$L_1 \subseteq L_2$
 equivalence
 topology
 $L_1 \cap L_2$
 if it's L_1 then subset otherwise
 not.

Intersection closed & equivalence available
 means subset of ω is true.

Reg \rightarrow UD
 DCFL \rightarrow UD
 CFL \rightarrow UD
 CSL \rightarrow UD

④ Intersection finiteness problem

is $L_1 \cap L_2$ finite or infinite?

→ Checking whether intersection of two lang. is finite or infinite.

$\cap \rightarrow$ closed.
 finite \rightarrow decidable

$RL \rightarrow D$
$DFL \rightarrow UD$
$CSL \rightarrow UD$
$R.L \rightarrow UD$
$REL \rightarrow UD.$

(B) Regularity problem:-

means checking whether given lang is regular or not,
 or given automata accepts reg lang, or not.

This problem is D for only for Regular & DCFL.

& undecidable(UD) for all the remaining formal languages.

D D D D D D

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Problem

	<u>Reg-L</u>	<u>DCFL</u>	<u>CPL</u>	<u>CSL</u>	<u>Recursive</u>	<u>REL</u>
①. is $w \in L$	D	D	D	D	D	UD
②. is $L = \phi$?	D	D	D	UD	UD	UD
③. is $L = \text{finite}$ not?	D	D	D	UD	UD	UD
④. is $L_1 = L_2$	D	UD	UD	UD	UD	UD
⑤. is $L_1 \cap L_2 = \phi$?	D	UD	UD	UD	UD	UD
⑥. is $L = \Sigma^*$?	D	D	UD	UD	UD	UD
⑦. is $L_1 \subseteq L_2$?	D	UD	UD	UD	UD	UD
⑧. is $L_1 \cap L_2$ finite or not?	D	UD	UD	UD	UD	UD
⑨. is $L = \text{Reg}$?	D	D	UD	UD	UD	UD
⑩. Is complement of lang is lang of same type or not?	D	D	UD	? (D)	D	UD
⑪. Is intersection of two languages lang of same type or not?	D <i>(because A is closed)</i>	UD	UD	D	D	D

Problems of Turing machine P:-

①. Halting problem of T.M. \Rightarrow

→ Checking whether by reading string x T.M. halts or not.

→ This problem is undecidable for recursive lang. & U.D. for

R.E.L.

②. Semi partially decidable problem \Rightarrow

→ If my problem is decidable under recursive language and U.D. under R.E.L. that is partially decidable.

→ Membership problem also partially decidable problem.

→ In general Partially Decidable problem also consider as Undecidable problems

see following problems about T.M.
is undecidable \rightarrow

• T.M. prints a specific letter

① whether the lang. accepted by
T.M. is context-free or not

② whether the lang. accepted by
T.M. is Reg. or not.

③ whether the T.M. accepts
at least 2 strings or not.

\rightarrow halting problem
empty problem, etc.
 \rightarrow nonempty problem

These foll. problems corresponds to T.M.
are decidable \rightarrow

① whether the T.M. having 5 states
or not.

② whether T.M. halts within 10 moves
on every input or not.

③ whether the T.M. never reaches any
state other than initial state or

not.
(State entry problems of T.M.'s are
decidable).

④ T.M. computes the product of two output.

Accept these 3 all are undecidable)

The following problems are undecidable regarding programming languages:

- (1) whether size a given program ever produces an output.
- (2) whether the given two programs produce the same output or not.
- (3) whether a given program loops forever on some input or not.

gate
2014

M is a T.M. that accepts string of length 2014 what long is it not? That long is.

- (1) Decidable & RBL.
- (2) UD & RBL
- (3) UD & not RBL T.M. is turing, why not RBL.
- (4) Decidable but not RBL.

- 2014
- (1) which of the foll problem is UD?
 - (2) checking whether string is member of g the given CFG or not.

Q. Checking whether lang. generated by given CFG is empty or not.

Q. Checking whether CFG is ambiguous or nonambiguous

Q. Checking whether CFG generates finite lang. or infinite lang.

Q. Which of the foll are undecidable?

① G is a CFG, is $L(G) = \emptyset$ or not?

② M is a T.M / is $L(M) = \text{Reg.}$? UD in the case of T.M.

③ Is $L(\text{DFA}) = L(\text{NFA})$ or not?

④ Are the CFGs given

G_1 & G_2

intersection not closed under CFL.

G_1 & G_2 are CFGs is $L(G_1) \cap L(G_2)$ is finite or infinite?

Q. Which of the foll are decidable?

① Does a given program ever produce an output?

② If L is a CFL then \overline{L} is also CFL?

- Q. Checking whether lang. generated by given CFG is empty or not.
- Q. Checking whether CFG is ambiguous
- Q. Checking whether CFG generates finite lang. or infinite lang.

- ²⁰¹³
Q. Which of the foll are undecidable?
- (1) G_1 is a CFG, is $L(G_1) = \emptyset$ or not?
 - (2) M is a T.M, is $L(M) = \text{Reg.}$? UD in the case of T.M.
 - (3) Is $L(\text{DFA}) = L(\text{NFA})$ or not?
 - (4) Are the CFGs G_1 & G_2 such that $L(G_1) \cap L(G_2)$ is finite or infinite? intersection not closed under CFLs.

- Q. Which of the foll are decidable?
- (1) Does a given program ever produce an output?
 - (2) If L is a CFL then \overline{L} is also CFL?

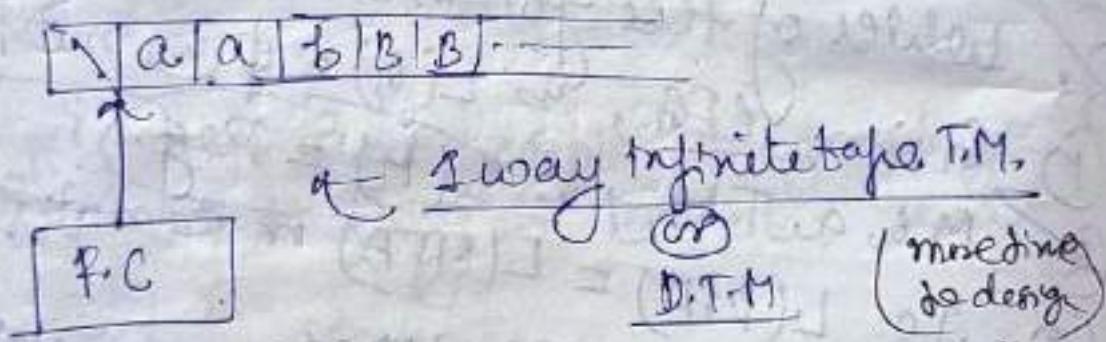
Q. If L is Recursive lang., T is also recursive lang.?

Q. If each G is a CFG is

$$L(G) = \Sigma^* ?$$

complement closed & emptiness decidable.

Modifications of Turing machine?



The following are modified version of T.M. :-

- (1). Two-way infinite tape T.M. } only modified to infinite & bidirectional
- (2). Two-tape P.M. } Expressive power is same.
- (3). Multitape T.M. } Expressive power is same.
- (4). Non-deterministic T.M. }
- (5). Universal T.M. }

Note:- If you modify tape T.M., the expressive power of T.M. remains same (neither increases nor decreases).

2-way infinite tape \rightarrow Tape is infinite in both dir's.

Mult.

2-tape T.M. \rightarrow Contains 2-tapes which is ∞ in both dirn.

multi-tape T.M. \rightarrow Contains multiple tapes, each tape is infinite in both dirn.

NDT.M \rightarrow from the given state on the given tape symbol has having finite no. of choices for the next move.

Ex: $(q, X) = (q_1^X, R)$

(q_1, Y, L)

(q_0, Z, R)

Universal T.M. Every other T.M. can be simulated by

Using this T.M.

→ The expressive power of NTM & DTM are same hence every NTM is converted into DTM.

polynomial → small time

exponent → huge time

For

→ Every non-DTM of polynomial time complexity we can construct an equivalent DTM of exponential time complexity.

Complexity Theory

(most) logarithmic poly. exponential

$$O(1) \leq O(\log n) \leq O(n^k) \leq O(2^n)$$

big time

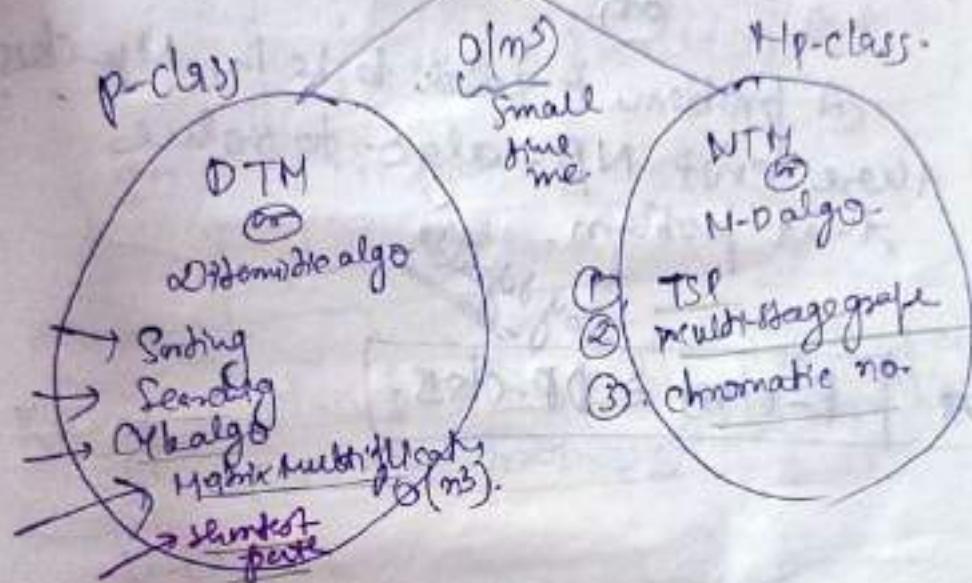
Travelling Salesman } = $O(2^n)$

very large time. $O(n^k)$

TM are
united

Small time me DTM banana difficult late h
to NTM banana.

Complexity theory



P-class languages

A lang. is said to be p-class if that lang. have exist deterministic T.M for that lang in polynomial time $\Theta(n^3)$ a problem is said to be p-class problem if there exist deterministic algo to solve that problem.

$O(n^3)$ - a language
from xyz
having xyz is workable.

NP-class :-

→ A problem is said to be NP-class if there exist NDTM for that long in polynomial time
or

A problem is said to be in NP-class, if there exist NP-algo to solve that problem.

* $P\text{-class} \subseteq NP\text{-class}$ *by default*



Is $P = NP$? → open question

For all the NP-class problem algo from P-class we convert
bar de matlab.

NTM to DTM
me baur baare do

$P = NP$ but atleast one NTM belongs
from DTM not bona fide
takhe aage ham sakte hain.

$P \neq NP \Rightarrow$ atleast
 $P = NP \Rightarrow$ possible & future.

so, we consider.

$P \neq NP$

Every P-class problem is NP-class but all
NP-class problem need not be
P-class.

$P = NP$ is a open question?

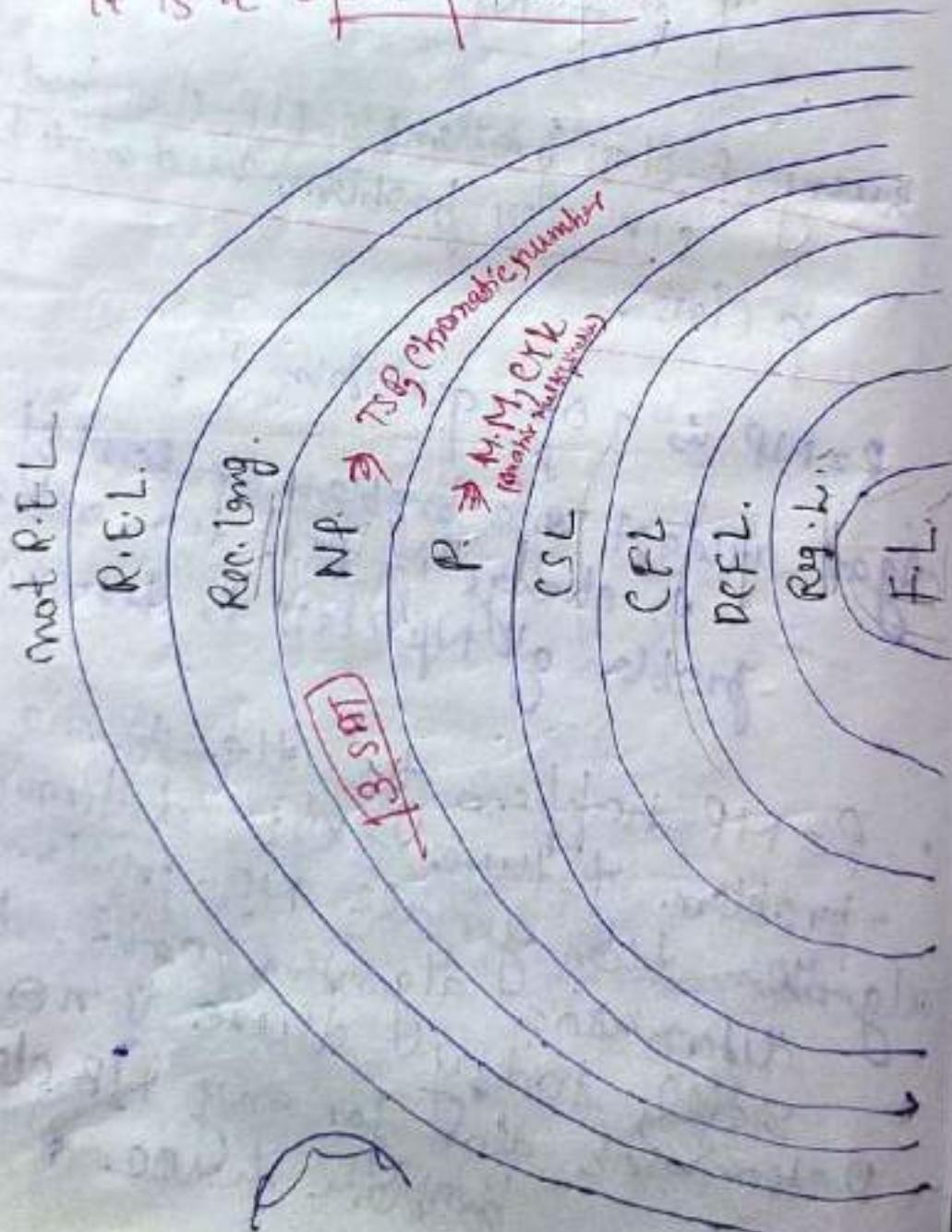
Agar atleast one problem for ~~exists~~ of NTM
is change in DTM then all such
problem of NP-class is deterministic.

- $P = NP$ happens for atleast one NP -
problem if there exist deterministic
algorithm then for all NP - problem also -
deterministic algorithm exist. but
still today there is no
deterministic algo for any NP-class
problem. hence

$P = NP$ may happens
 $P \neq NP$ may happens

Hence $P = NP ?$ is a open question.
 but a student prove that,

$P \neq NP$
 but on this there is a
 claim on it so, here it again
 it is a open question.



Closure properties of P and NP:-

Both P-class & NP-class languages are closed under following →

- ①. Union operation
- ②. Concatenation operation
- ③. Kleene closure "
- ④. Intersection "
- ⑤. tve closure "

P-class languages are closed under Complement
but the complement of NP-class is unknown?

Hence the problem,

$$NP = \text{Co-NP. ?}$$

complement

is a open question

ATG
2003

Q. Let $L = \begin{cases} (a+b)^* & \text{if } P = NP \\ \emptyset & \text{if } P \neq NP \end{cases}$ then L is

~~(a)~~ Regular

~~(b)~~ RCL but not Recursive

~~(c)~~ RCL but not CL

~~(d)~~ we can't decide whether
 $P = NP$ is

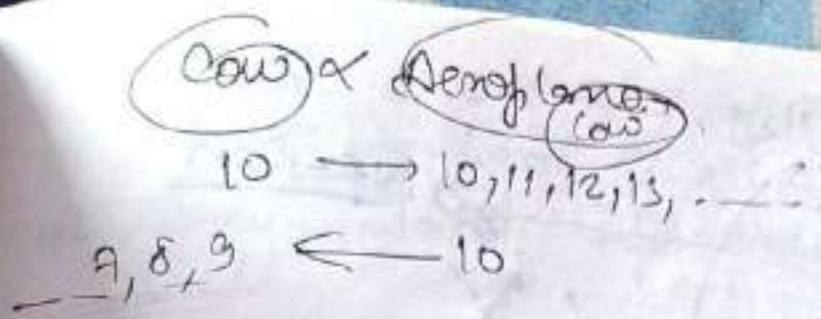
~~(e)~~ known for problem.

Here $(a+b)^*$ and \emptyset also is Regular
then there is no need
of P & NP problem.

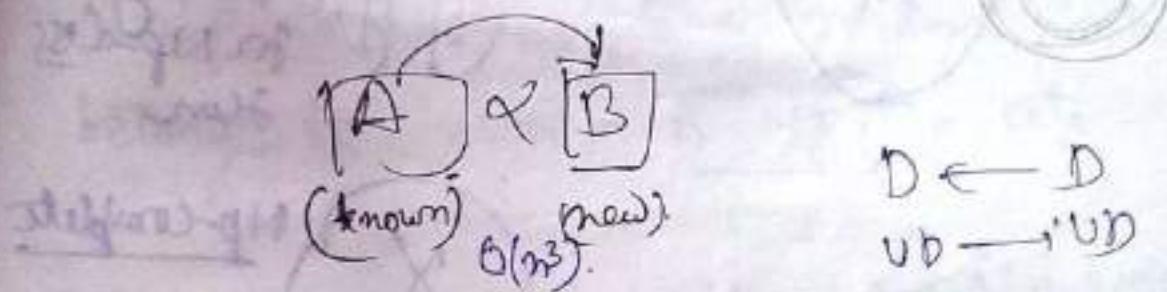
~~Cow & Aeroplane.~~

Polynomial reduction:

A problem A is polynomially reduced
to B (problem) means we can
conclude the problem B with
the help of problem A this
redn takes poly. time hence
it is known as poly. reduction
also.



$\frac{\text{If } A \text{ reduces } B \text{ the fall are}}{\text{the possibility} \rightarrow}$



$$\begin{array}{l} D \leftarrow D \\ UD \rightarrow UD \end{array}$$

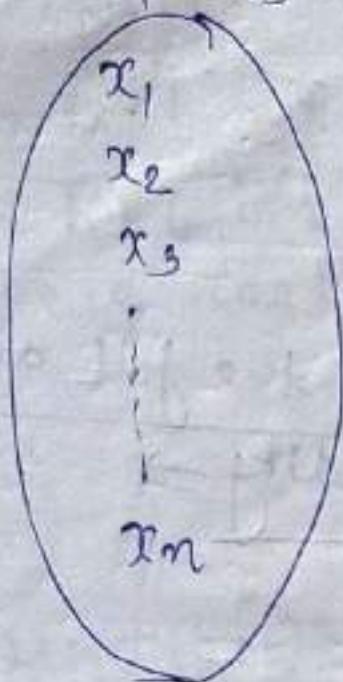
①. $P \leftarrow P$ } Decidable problem
 ②. $P, NP \leftarrow NP$

③. $SP \rightarrow P, NP, \text{separ}, RPL, \dots$
 ④. $NP \rightarrow MP, \text{separ}, REL, \dots$
 Undecidable

NP-hard and NP-Complete \Rightarrow

$$P \subset NP \subset NP^{NP}$$

NP-class



NP-hard

X X.

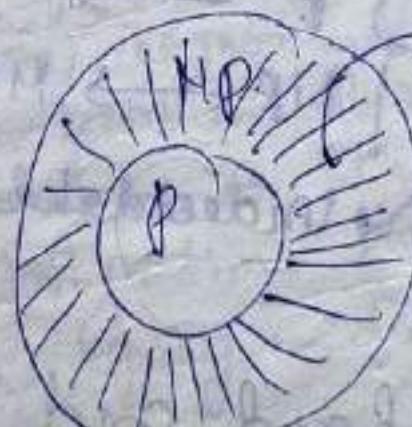
NP, Recursive, RBL,
not RBL -

↓ If it belongs
to NP-class
then

NP-complete



CFL but
not regular



$$P \cap \text{NP-Complete} = \emptyset$$

NP-
complete
but not
P is NP
but not P.

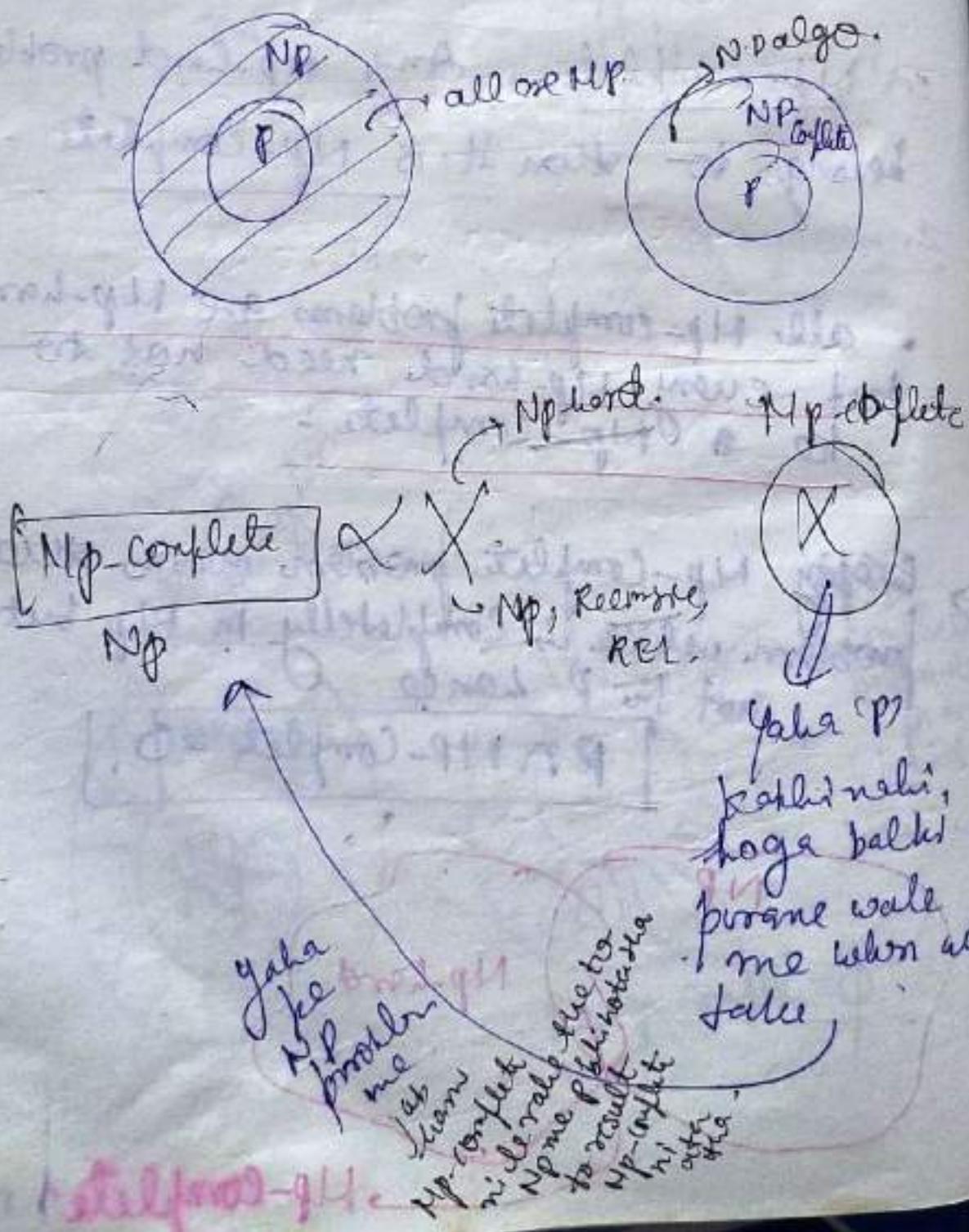
- A problem is $\otimes X$ is said to be empty-hard if all NP-problems are polynomially reduced to X .
- NP-hard problems may belongs to NP-class \cap may not belongs to NP-class.
- NP-complete \rightarrow Any NP-hard problem belongs to then it is NP-complete.
 all NP-complete problems are NP-hard
 but every NP-hard need not to be a NP-complete.

\rightarrow Empty NP-Complete problem means the problem which is completely in NP but not in P hence

$$P \cap \text{NP-Complete} = \emptyset,$$


Note:—

For at least one NP-complete problem, deterministic algorithm exist then $P = NP$ happen but still today it is unknown.



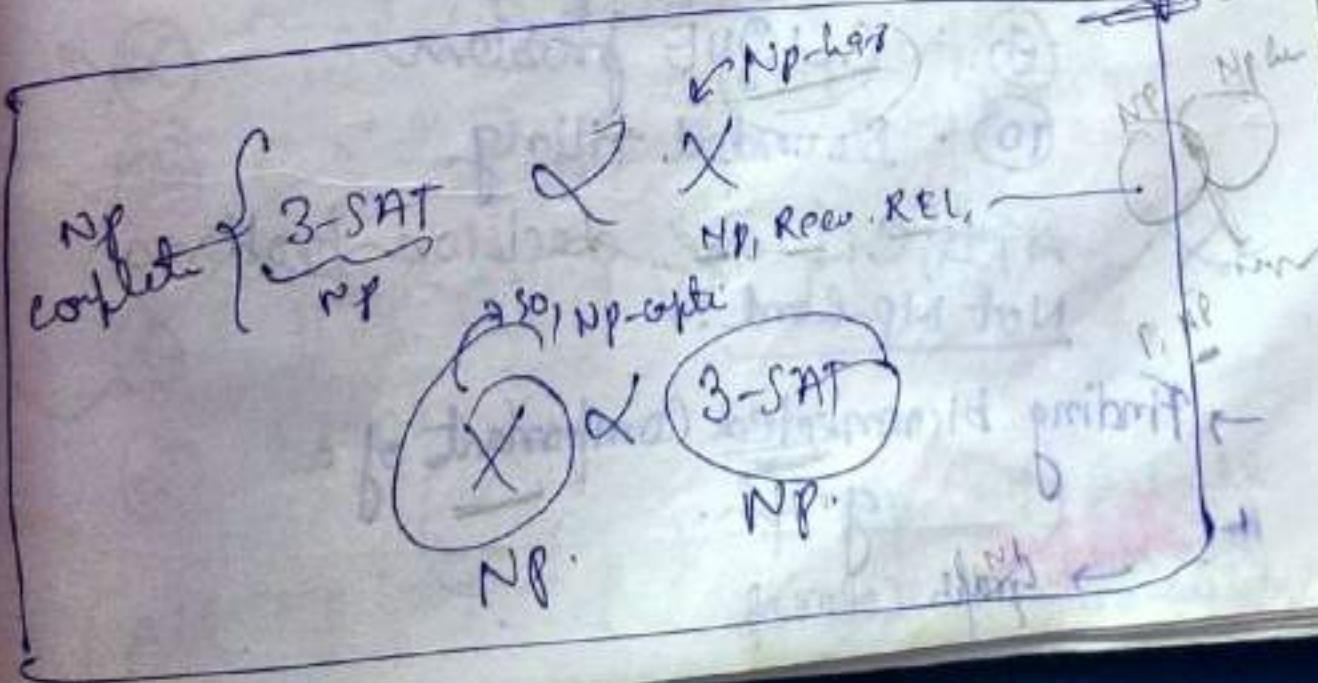
A problem is not solvable in poly. no.
mean we prove that
problem is NP-complete.

Cocke's Cooke's method →

Note:-

- To prove a problem X is NP-complete take some known NP-complete problem and polynomially reduce it to X . Then X is NP-hard. Again if X is reduced to known NP-Complete problems.
- 3-SAT is the 1st problem proved as NP-complete problem.

In Concept



Q. The foll. are some of the known NP-complete problems:

- ①. TS problem
- ②. (3-SAT,
4-SAT, 5-SAT, ..., n-SAT)

→ 2-SAT is p-class problem.

- ③. Chromatic no. problem
- ④. Vertex Cover problem
- ⑤. Independent Set problem
- ⑥. Hamiltonian Circuit problem
- ⑦. 0/1 knapsack problem
- ⑧. Subset Sum problem
- ⑨. CLIQUE problem
- ⑩. Bounded tiling.

Q. A, B, C, D are decision problems
Not NP-hard:-

→ Finding biconnected Component of a graph.

→ Graph coloring

A & B.

- | | |
|----|------------------------------------|
| ①. | UD \rightarrow UD |
| ②. | D \leftarrow D |
| ③. | P \leftarrow P |
| ④. | P, NP \leftarrow NP |
| ⑤. | Rec. Long. \leftarrow Rec. Long. |
| ⑥. | REL \leftarrow REL |
| ⑦. | Np-hard \rightarrow Np-hard |

Q. A, B, C, D are decision problems

$$\underline{A \& B \rightarrow \text{NP}}$$

$$\underline{B \& C}$$

$$\underline{C \& D \rightarrow \text{NP-hard}}$$

~~From Q. A is NP-hard then which of the following is true.~~

- (a) ~~TSP $\&$ D \Rightarrow D is NP-complete~~
- (b) ~~C $\&$ 3-SAT \Rightarrow C is NP-complete but not NP-hard~~
- (c) ~~D $\&$ TSP \Rightarrow D is NP-complete - hard~~
- (d) ~~None,~~

(All NP-complete problem is NP-hard) but all NP-hard need not be NP problems

Q. $S = \text{NP-complete}$

Q & R two other problem
not known to be an NP then

$\begin{array}{c} P \neq NP \\ | \\ Q \& S \text{ NP-complete} \\ | \\ NP \neq \{S\} \& R \text{ hard} \end{array}$

then either of the foll is true

- (1) - R is NP-complete
- (2) - Q is NP-complete
- (3) - Q is NP-hard non belong to P.
- (4) - R is MP-hard.

Theorem: a language is recursive if and only if it can be effectively enumerated in lexicographic order.

Given a T.M. 'M' over the input alphabet Σ , any state q of M and a word Σ^* , does the computation of M on w visit the state q ?

→ is undecidable but partially decidable.