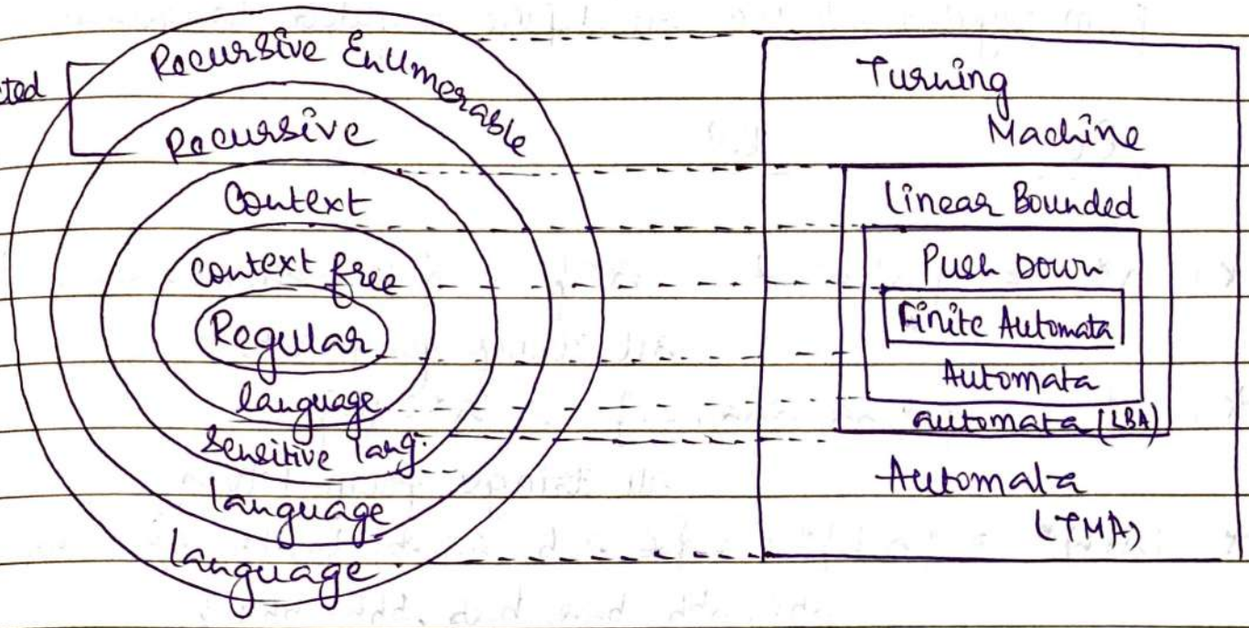
THEORY OF COMPUTATION

paper model of computer (computation)



PDA - External memory in the form of stack to store data.

LBA more powerful than PDA.

Diagonalization lang. Category - language above recursive enumerable language.

TM - exactly same working as computer.

Sum up

Regular ⊂ Context free ⊂ Context Sensitive ⊂ Recursive lang. ⊂ Recursive Enumerable

Power (FA) < Power (PDA) < Power (LBA) < Power (TM)

• $\{+, \cdot, *\}$ → concatenate, → Kleen's closure
either or
• zero length string $\lambda = \epsilon$ (Null String)

REGULAR LANGUAGE

REGULAR EXPRESSION - R.E generates regular set and from regular set we can define regular language.

R.E	R.S

* $a^* = \{a^0, a^1, a^2 \ldots a^\infty\} = \{\epsilon, a, aa, aaa, aaaa \ldots\}$
all strings from 0 to ∞.
* $a^+ = \{a, aa, aaa \ldots\} = a^* - \{\epsilon\}$
all strings from 1 to ∞
* $(a+b)^* = \{a, b\}^* = \{\epsilon, a, b, aa, ab, ba, bb, aaa, aab, aba, abb, baa, bab, bbb, bba\}$
* $(a.b)^* = \{\epsilon, ab, abab, ababab \ldots\}$

$$(a.b)^* \subset (a+b)^*$$

Eg- Construct R.E for the following regular languages:

(i) All the strings of ab have atleast one a

(I.) $(a+b)^* . a$ — all strings end with a
$a.(a+b)^*$ — complete set of a and b; all strings end with b or a

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

(III.) $(a+b)^* a (a+b)^*$ General Solution.

(ii) All the strings of a, b have exactly one a
$b^* a b^*$

All the strings of a, b have exactly two b's

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

All the strings of a, b have atleast one pair of consecutive two b's

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

All the strings of a, b do not have consecutive two b's.

~~$a^* b a^* b a^*$~~ ~~$a^* (a.b)^* . a^* (a.b)^* . a^*$~~

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

All the strings of a, b have substring aba.

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

All the strings of a, b end with bb.

$(a+b)^* bb$

All the strings of a, b do not end with bb.

~~$(a+b)^* . (b+\epsilon)$~~

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

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

All the strings of a, b begin and end with same symbol.

$a.(a+b)^* a + b.(a+b)^* b + \not{\phi} + a + b$

All the strings of a, b do not have substrings aba.

Any language can be defined as

$$L = \{ \text{Def}^n \mid \text{Cond}^n \}$$

where

* $L = \{a^n \mid n \geq 0\}$
  $= \{a^0, a, aa, aaa \ldots\} = a^*$

* $L = \{a^n b^m \mid n \geq 0, m \geq 1\}$
  $= \{b + abb \ldots\}$
  $= a^* b^* b$

* $L = \{a^n b^m \mid n \geq 2, m \leq 3\}$
  $= \{aabbb, aaabb, aaaab, aaaaa(\epsilon)\}$
  $= aa.a^*(bbb + bb + b + \epsilon)$

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

* $L = \{a^n b^m \mid (n+m) \text{ is even}\}$
  $= (aa)^*.(bb)^* + a^*.b^* \; a(aa)^* \; b(bb)^*$

* $L = \{a^n b^m \mid n \geq 3, m \leq 2\}$
  find RE for $\bar{L}$.

$$\begin{bmatrix} n < 3 & m > 2 \\ n \leq 2 & m \geq 3 \end{bmatrix}$$

$aabbb$, a

$(aa + a + \epsilon)\, bbb.b^*$

$L_1 = \bar{L} = \{a^n b^m \mid n < 3 \text{ OR } m > 2\}$

Empty Set - No element
Empty String - length 0.

	C_1	C_2	C_1 or C_2
$a^*bbbb^* + (\epsilon+a+aa)b^*$	0	0	0
$+(\epsilon+a+aa).bbbb^*$	0	1	1
	1	0	1
	1	1	1

For $a^n b^m$ ⟶ $\boxed{a^*b^*}$ Universal Set.

$a^*b^* - L$ (Not possible) ⊖ not an operator.

Properties of Regular operator/Expression

- $R.\emptyset = \emptyset.R = \emptyset$ $\quad \emptyset$ = Empty Set, $|\emptyset| = 0$.
- $R.\epsilon = \epsilon.R = R$
- $\emptyset^* = \epsilon^* = \{\epsilon\} = \epsilon$

  $\emptyset^* = \{\epsilon, \emptyset, \emptyset.\emptyset, \emptyset.\emptyset.\emptyset, \emptyset.\emptyset.\emptyset.\emptyset \ldots\}$

  $= \{\epsilon, \emptyset, \emptyset, \emptyset \ldots\}$

  $= \{\epsilon, \emptyset\} = \{\epsilon\}$

  ↑ due to *

  $\epsilon^* = \{\epsilon, \epsilon, \epsilon\epsilon, \epsilon\epsilon\epsilon, \epsilon\epsilon\epsilon\epsilon \ldots\}$

  $= \{\epsilon\}$
- $R + \emptyset = \emptyset + R = R$
- $R + \epsilon = \epsilon + R$ $\quad (\neq R \quad a^+ + \epsilon = a^* \ (\neq a^+))$
- $(R^*)^* = R^*$

  $(R^*)^* = \{\epsilon, R^*, R^*.R^*, R^*R^*R^* \ldots\}$

  $= (\{\epsilon, R, R.R, R.RR \ldots\})^*$

  $(= (\epsilon + R + RR + RRR + \ldots)^*$

  ↖ either or.
- $R.R^* = R^*.R = R^+$
- $\epsilon + RR^* = \epsilon + R^*R = R^*$

- $(P+Q)^* = (P^*+Q^*)^* \neq (P^*+Q)^* = (P+Q^*)^* = (P^*.Q^*)^*$

$$\begin{cases} (P^*Q)^* \\ (P.Q^*)^* \\ (P.Q)^* \end{cases} \subset (P+Q)^*$$

- $P.(Q+R) = P.Q + P.R$
- $P.R + Q.R = (P+Q).R$
- $P^*(QP^*)^* = (P^*.Q)^*.P^*$

$$\begin{cases} P^*(QP^*)^* = \{\epsilon, P, Q, PP, QQ, PQ, QP \ldots\} \\ (P^*Q)^*.P^* = \{\epsilon, P, Q, PP, QQ \ldots\} \end{cases}$$

Check whether the regular sets of these regular expressions are same or not.

$$P^*(QP^*)^* = \{\epsilon, P, Q, PP, PQ, QP, QQ \ldots\}$$
$$= (P+Q)^*$$
$$(P^*Q)^*.P^* = \{\epsilon, P, Q, PP, PQ, QP, QQ \ldots\}$$
$$= (P+Q)^*$$

Zero, single, double length strings present.
So, all strings are possible.

Finite Automata (FA)

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

Q: Set of finite states

q_0: an initial state $q_0 \in Q$

Σ: Set of finite input symbols.

F: Set of final finite state $F \subseteq Q$

δ: Transition function

Deterministic FA (DFA)
(atleast one transition req.)

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

Or

$\delta(q_i, a) \to q_j$

where $q_i, q_j \in Q$

$a \in \Sigma$

Total function

From every state for each input symbol there must be exactly one transition.

Non-deterministic FA (NFA)

$\delta : Q \times (\Sigma \cup \epsilon) \to 2^Q$

(Power set of states)

Or

$\delta(q_i, a) \to A$

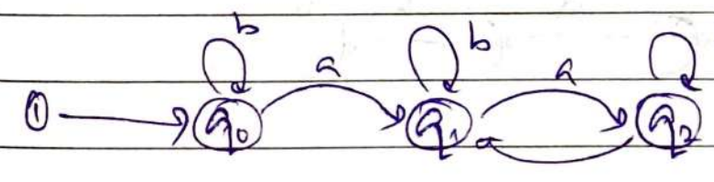
where $q_i \in Q$

$A \subseteq Q$

$a \in (\Sigma \cup \epsilon)$

NOTE:

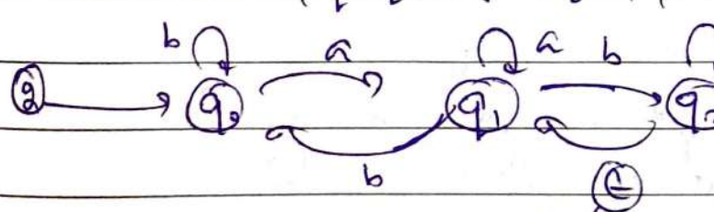
* Every DFA is also an NFA.
* Every NFA is not a DFA but we can convert every NFA to equivalent DFA. Power (DFA) = Power (NFA)



$\delta(q_0, a) \to q_1$

$\delta(q_1, b) \to q_2$.

DFA ($\{q_0, q_1, q_2\}$, q_0, $\{a, b\}$, δ, $\{q_2\}$)



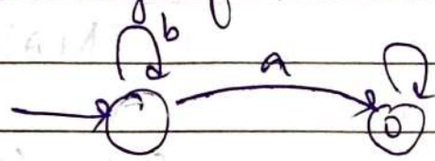
NFA ($\{q_0, q_1, q_2\}$, q_0, $\{a, b\}$, δ, $\{q_1, q_2\}$)

$\delta(q_1, a) \rightarrow q_1$

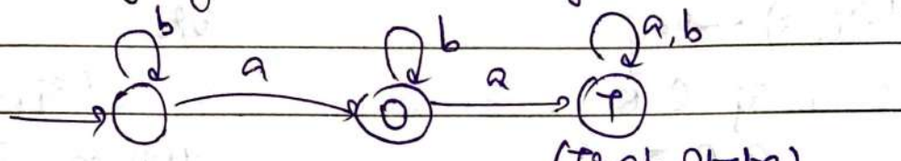
$\delta(q_1, b) \rightarrow \{q_0, q_2\}$

Q- And DFA for the following regular languages.

(i) All the strings of a, b have atleast one a.



(ii) All the strings of a, b have exactly one a.



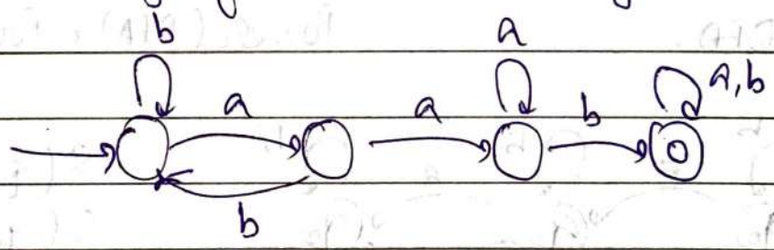
(Trap state)
(dummy)

Can be a part of DFA but not NFA.

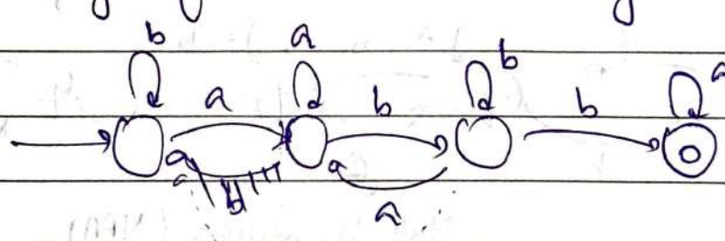
Every DFA can have atmost one trap state.

Situation in which we want to include a and b but it violates the condition. (to fulfill the req. of DFA)

(iii) All the strings of a, b have substring aab.

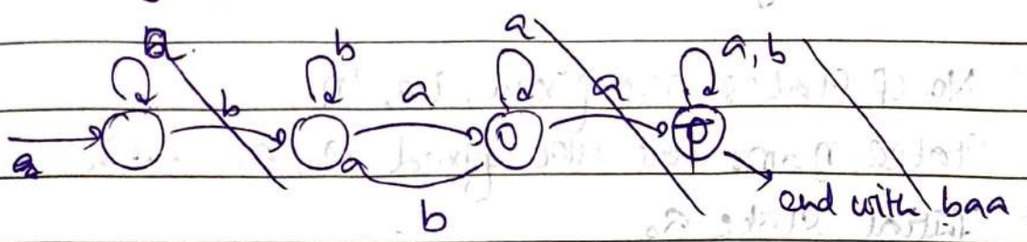


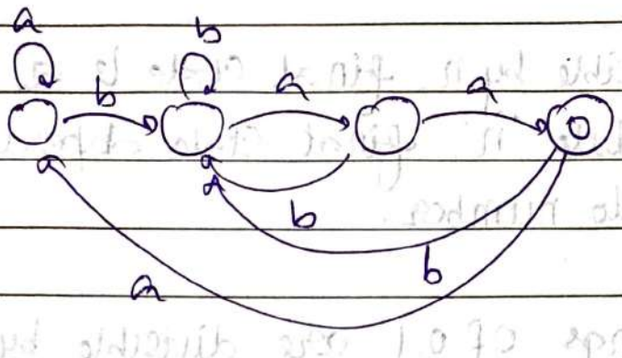
(iv) All the strings of a, b have substring abb.



If initial, abb not present.

(v) All the strings of a,b end with baa.



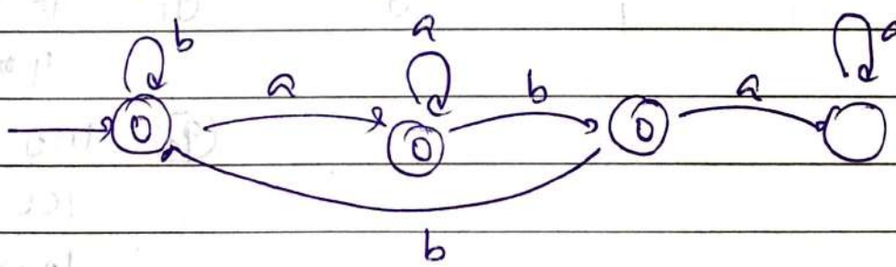


(vi) All the strings of a,b do not have substrings aba

Compliment of DFA

Let a DFA(Q, q_0, Σ, S, F) with 'n' states then it's compliment DFA (Q, q_0, Σ, S, Q-F) is also with n states.

Change final to non-final
Non-final to final.

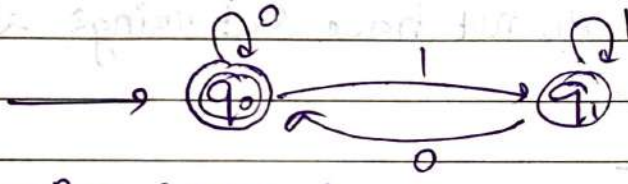


All the strings of a,b do not have substring aba.

Divisible by n or modulo-n problem

(i) No. of States are fixed, i.e, 'n'
(ii) States name are also fixed $q_0, q_1, \ldots q_n$
(iii) Initial State q_0
(iv) final state
  (i) For divisible by n, final state is q_0
  (ii) For modulo 'n', final state depends on desired modulo number.

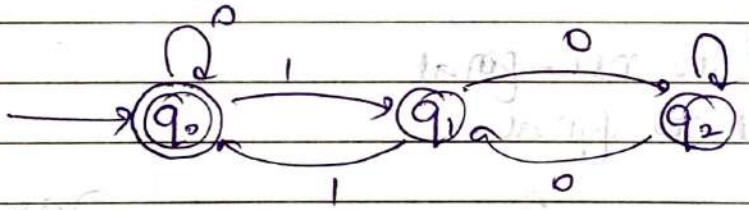
* All the strings of 0,1 are divisible by 2.



q_0 Both initial & final state.

0%2 = 0
1%2 = 1
10%2 = 0
11%2 = 1

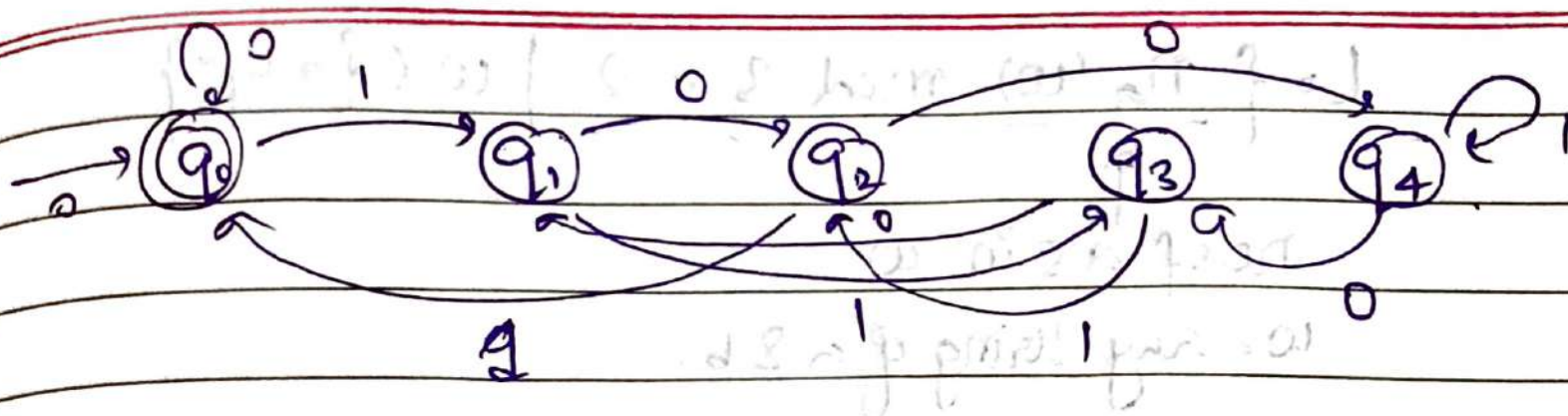
* All the strings of 0,1 are divisible by 3.

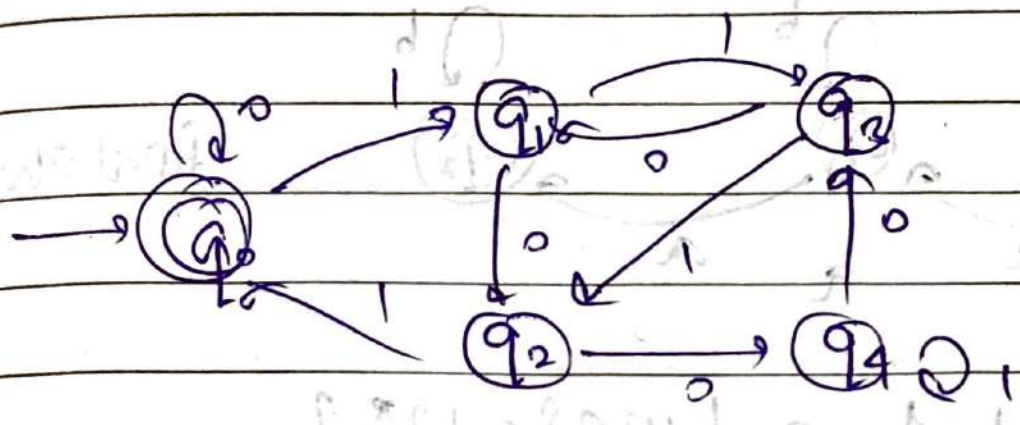


0%3 = 0
1%3 = 1
10%3 = 2
11%3 = 0
q_1 ...1110
100%3 = 1
101%3 = 2
q_2

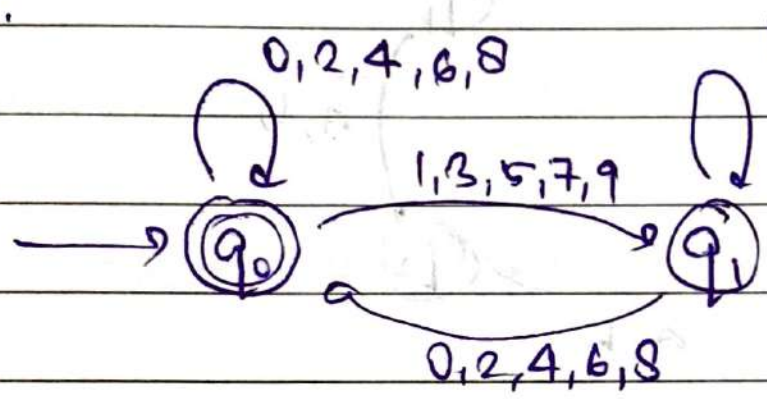
* All the strings of 0,1 are divisible by 5.

0%5 = 0
1%5 = 1
10%5 = 2
11%5 = 3
100%5 = 4
101%5 = 0
110%5 = 1
111%5 = 2
1000%5 = 3
1001%5 = 4





* All the strings of decimal alphabets $\Sigma[0,9]$ are divisible by 2.

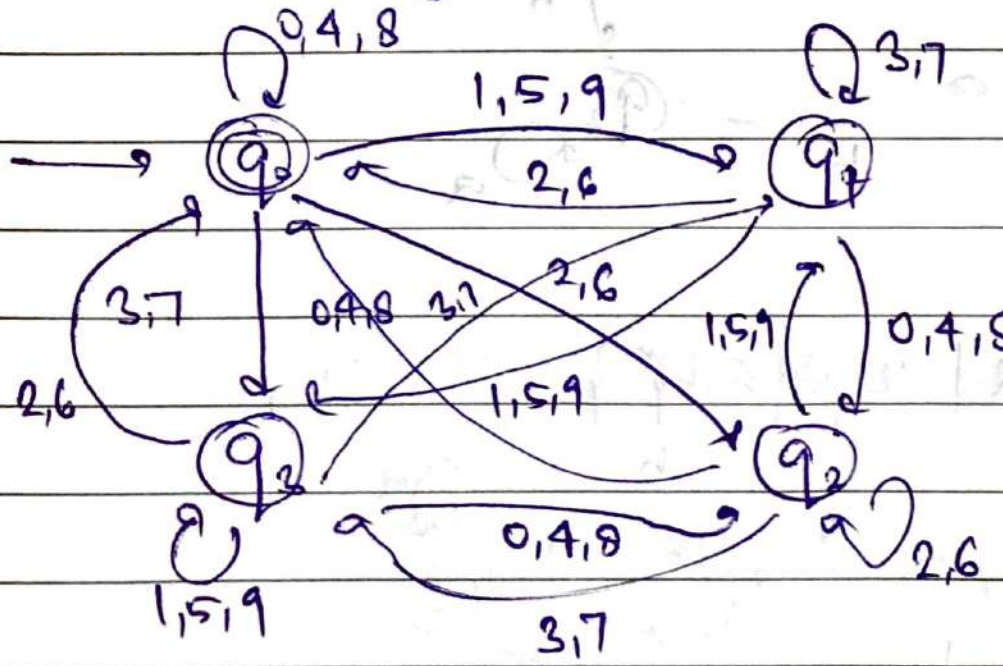


For next state, only transition of one member of a group is shown.

74 % 2 = 0 75 % 2 = 1

* All the strings of decimal alphabets are divisible by 4.

No. of groups = based on Remainder



10 % 4 = 2.
12 % 4 = 0
13 % 4 = 1
11 % 4 = 3

22 % 4 = 2
24 % 4 = 0.
21 % 4 = 1
23 % 4 = 3

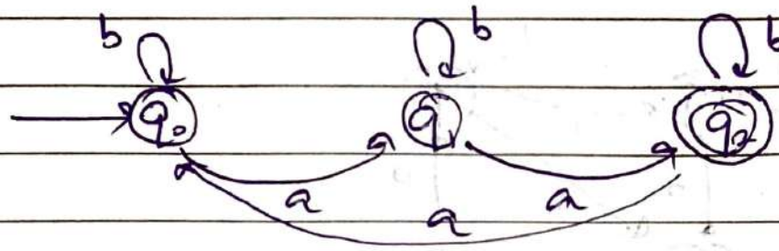
33 % 4 = 1
30 % 4 = 2
31 % 4 = 3

32 % 4 = 0

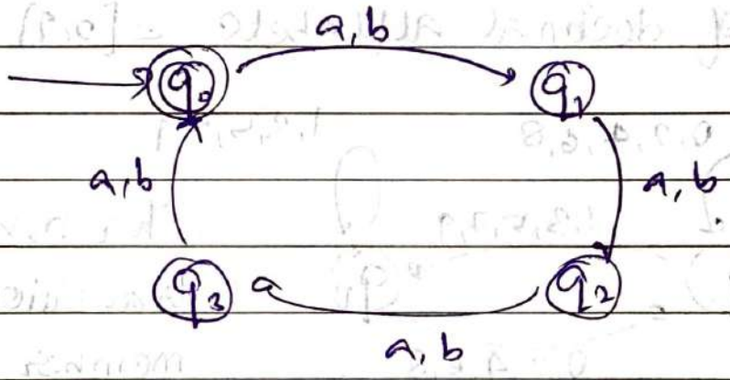
* $L = \{ n_a(w) \bmod 3 = 2 \mid w \in \{a,b\}^* \}$

$n_a(w)$: no. of a's in w

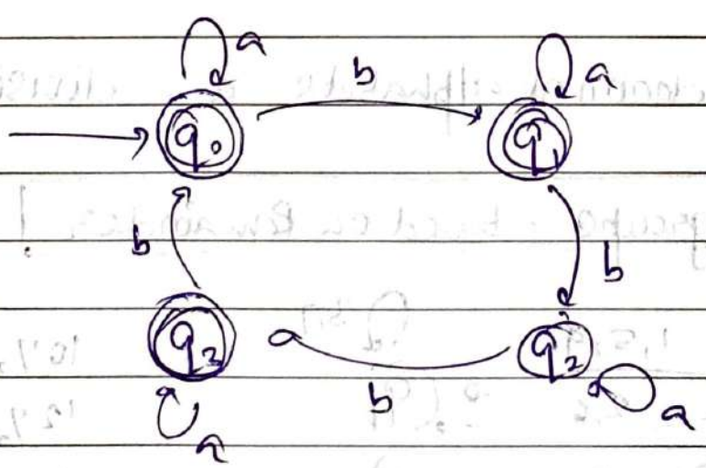
w = any string of a & b.



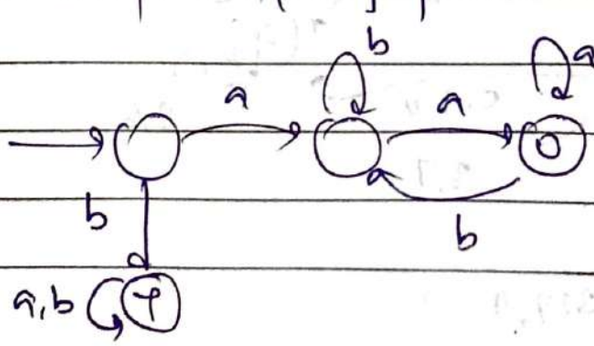
* $L = \{ |w| \bmod 4 = 0 \mid w \in \{a,b\}^* \}$.



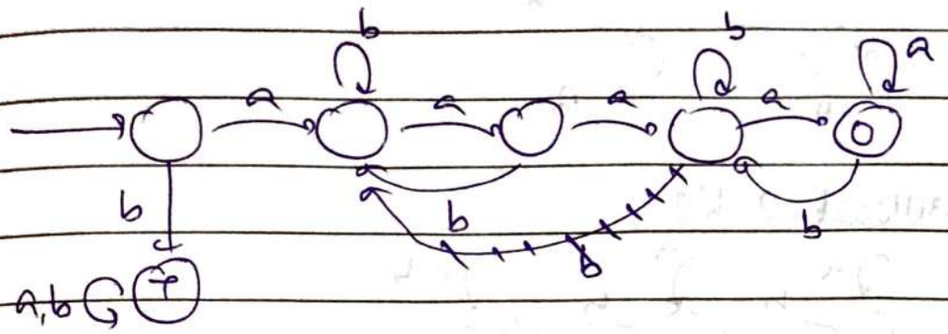
* $L = \{ n_b(w) \bmod 4 \neq 2 \mid w \in \{a,b\}^* \}$.



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



* $L = \{aw_1aaw_2a \mid w_1, w_2 \in \{a,b\}^*\}$.



* All the strings of a, b have atleast one a and atleast 2 b's.

Union and Intersection of two DFA's

Let two DFA's are DFA $M_1(Q_1, q_1, \Sigma, \delta_1, F_1)$ and DFA $M_2(Q_2, q_2, \Sigma, \delta_2, F_2)$

The cross product of M_1 and M_2 is $M_3 = M_1 \times M_2$.

$M_3(Q_3, q_{xy}, \Sigma, \delta_3, F_3)$

where, $Q_3 = Q_1 \times Q_2$

$\delta_3 = \delta_1 \times \delta_2$

Eg $\delta(q_A, a) \longrightarrow q_B$ δ_1 DFA M_1

$\delta(q_C, a) \longrightarrow q_D$ δ_2 DFA M_2

$\delta(q_{AC}, a) \longrightarrow q_{BD}$ δ_3 DFA M_3.

F_3

Case I: Intersection (AND) $F_3 = F_1 \times F_2$.

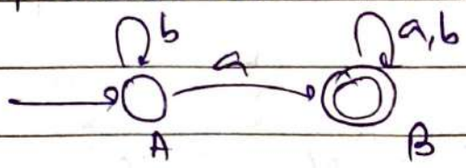
Case II: Union (OR) $\forall q_{AB} \in F_3$.

where $A \in F_1$

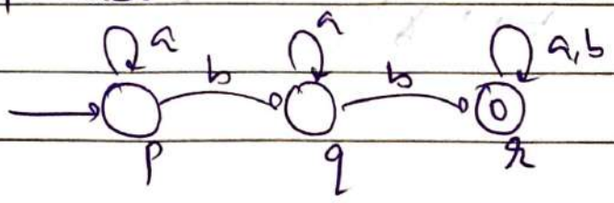
or $B \in F_2$.

In DFA,
Atmost one trapped State.

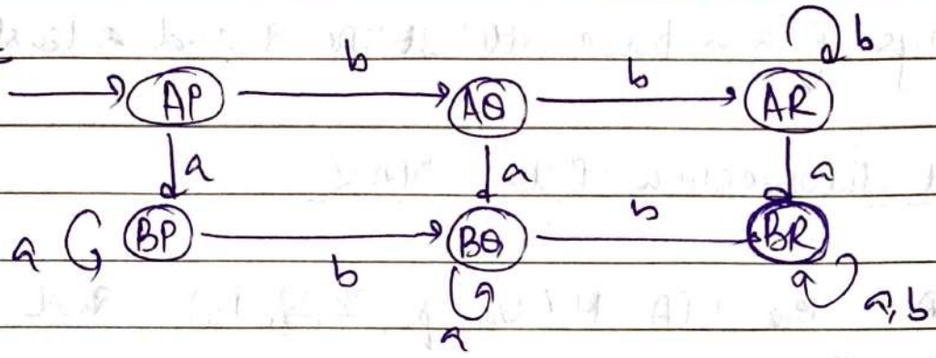
DFA of atleast one a.

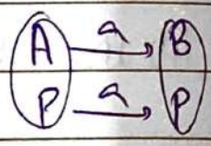


DFA of atleast 2 b's



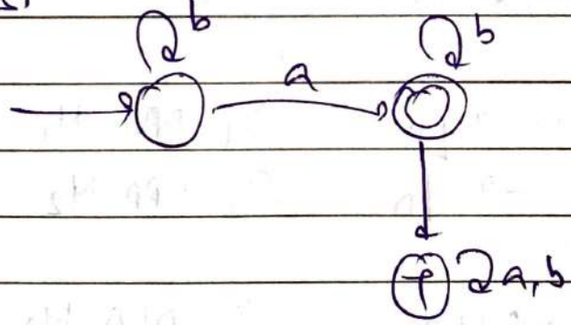
Intersection

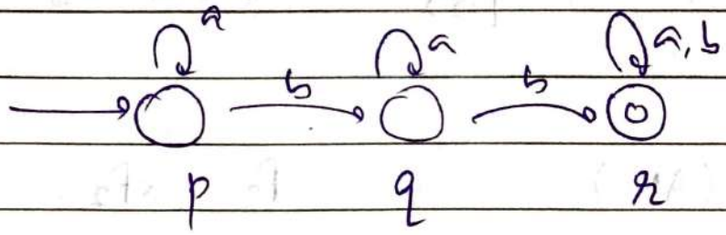


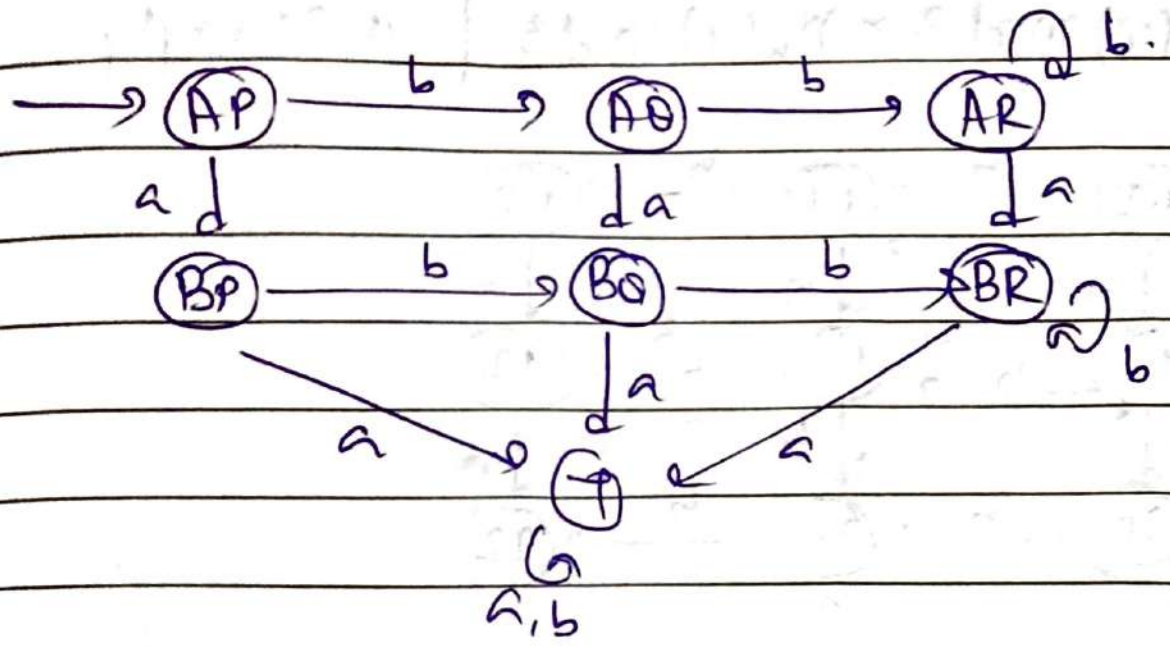


Union- Final States of 1 and 2 both,
AR, BP, BQ and BR.

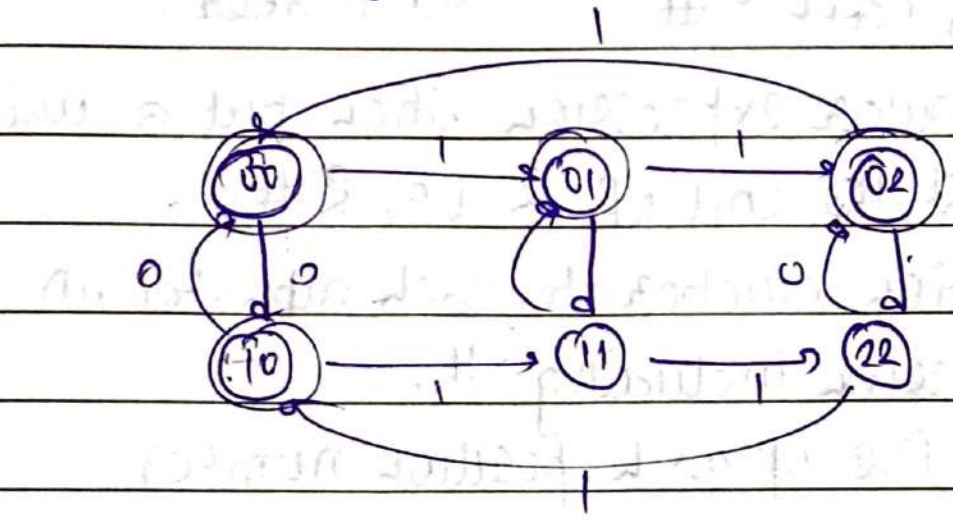
★ All the strings of a,b have exactly one a and atleast 2 b's.



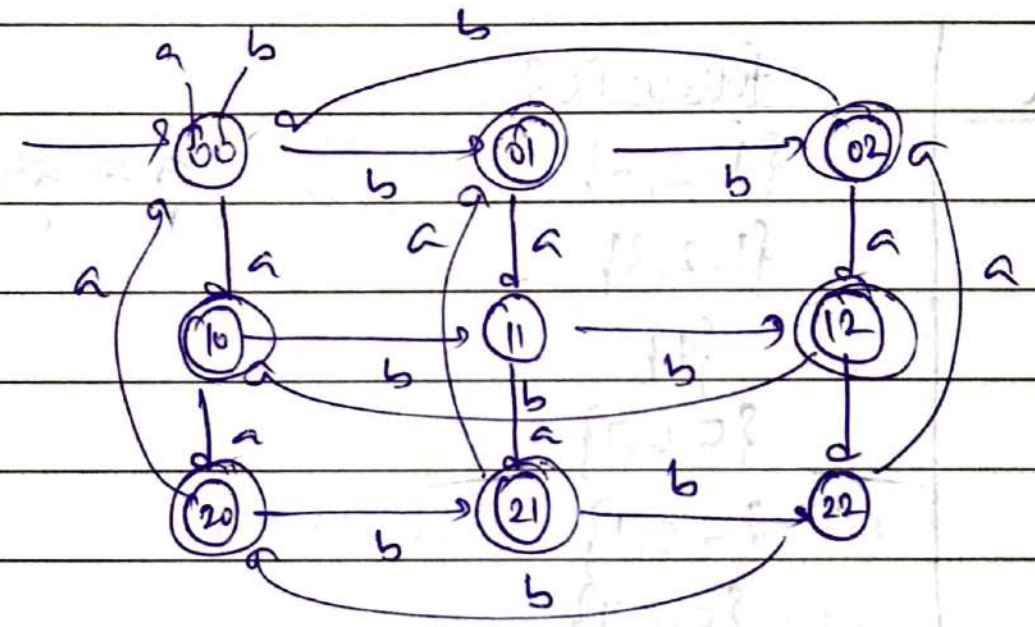




* All the ~~Strings~~ of 0,1 ~~are divisible~~ in which no. of 0's are divisible by 2 or no. of 1's are divisible by 3.



* $L = \{ |n_a(w) - n_b(w)| \bmod 3 \neq 0 \mid w \in \{a,b\}^+ \}$

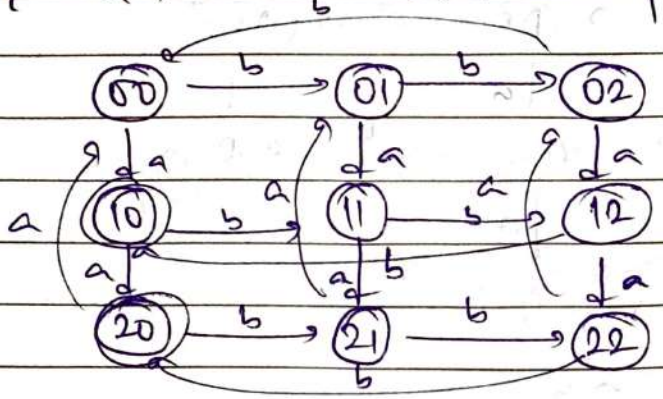


All States other than diagonal will be final State because

$0-0=0$
$1-1=0$
$2-2=0$

) return.

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



Regular Expression to DFA:

$(a+b)^* aa (a+b)^* \#$ (1 2 3 4 5 6 7) ← unique end marker.

(i) Let R is a regular expression then put a unique end marker at the end of R, i.e, R#

(ii) Assign a position number to each alphabet in the regular expression including '#'.

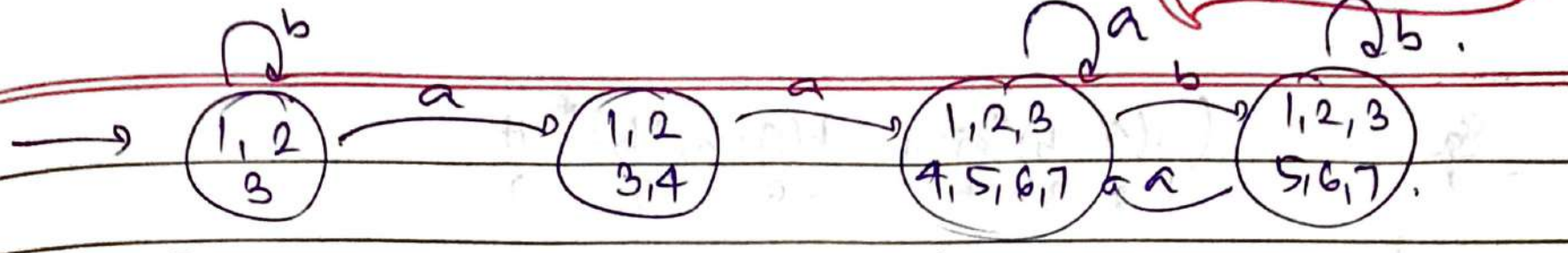
(iii) Find follow Pos of each position number.

Follow Pos ({x}) = Set of all positions numbers which follow position 'x' in the regular expression.

Position	follow Pos
{1}	{1,2,3}
{2}	{1,2,3}
{3}	{4}
{4}	{5,6,7}
{5}	{5,6,7}
{6}	{5,6,7}
{7}	—

a a a b a a
1 1 1 2 1 3

(iv) Initial State of DFA - All the positions from where regular expression can begin.

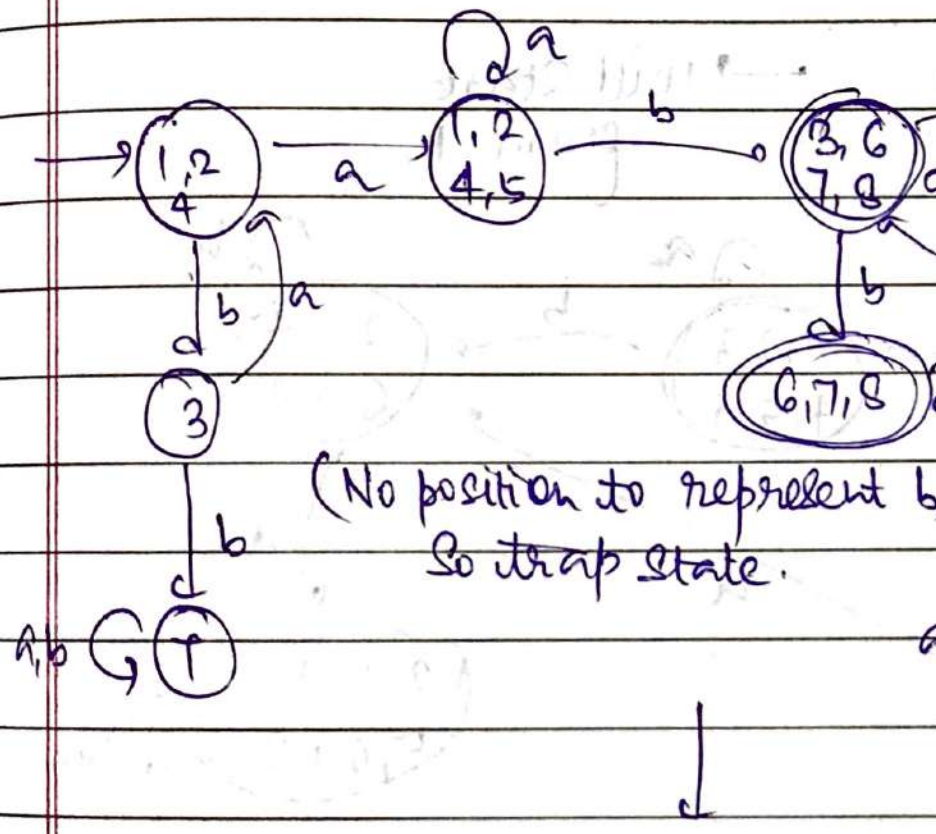


$S(\{1,2,3\}, a) = \text{FollowPos}(\{1\}) \cup \text{FollowPos}(\{3\})$
$= \{1,2,3\} \cup \{4\}$
$= \{1,2,3,4\}.$

(V) All the states which contain position number of Symbol '#' are final states.

Since we can draw DFA in 3 States only - So minimization of DFA is required.

Eg- $(a+ba)^*.a.b\,(a+b)^*\,\#$
$\quad\;\; 1\;2\;3\quad 4\;5\quad 6\quad 7\quad 8$



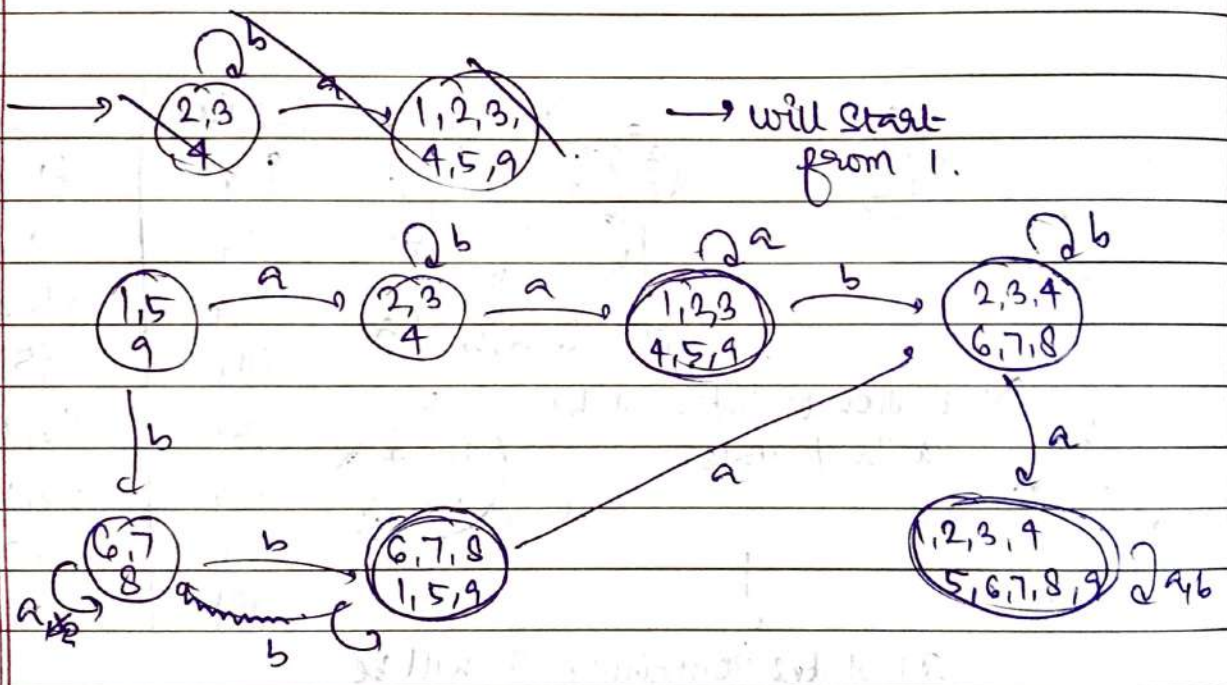
Position	follow POS
{1}	{1,3,4}
{2}	{3}
{3}	{1,2,4}
{4}	{5}
{5}	{6,7,8}
{6}	{6,7,8}
{7}	{6,7,8}
{8}	—

All states containing 8 will be final state.

4 final states.

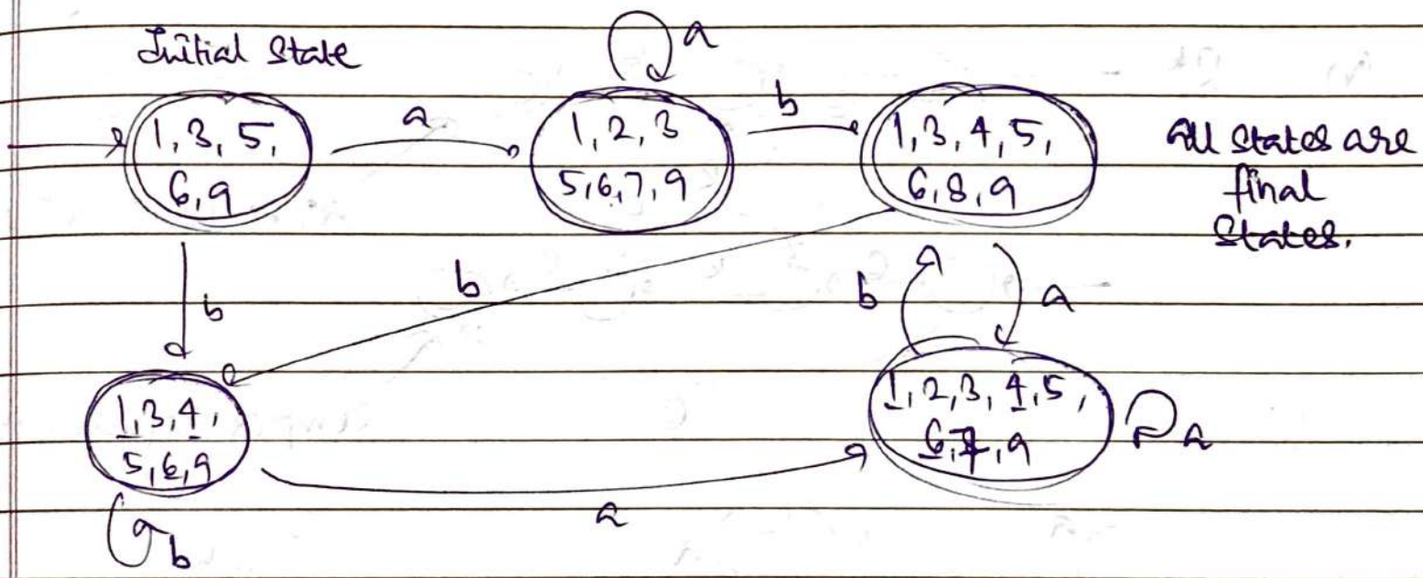
Eg: $(a(a+b)^*a + b(a+b)^*b)^*\#$
$\quad\quad 1\ 2\ 3\ 4 \quad\quad 5\ 6\ 7\ 8 \quad\quad 9$

Position	follow Pos
{1}	{2,3,4}
{2}	{2,3,4}
{3}	{2,3,4}
{4}	{1,5,9}
{5}	{6,7,8}
{6}	{6,7,8}
{7}	{6,7,8}
{8}	{1,5,9}
{9}	—



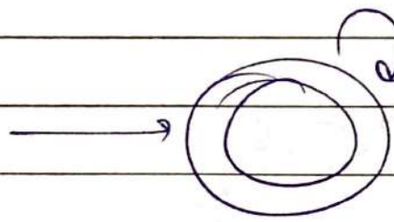
$((a.b)^*b.a^* + b^*a(b.a)^*)^* \#$
$\quad 1\ 2\ 3\ 4 \quad\quad 5\ 6\ 7\ 8 \quad\quad 9$

Position	follow pos
{1}	{2}
{2}	{1,3}
{3}	{3,4,5,6,9}
{4}	{3,1,5,6,4,9}
{5}	{5,6}
{6}	{7,6,5,1,3,9}
{7}	{8}
{8}	{1,3,5,6,7,9}
{9}	–

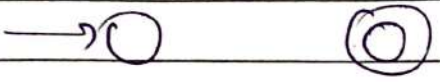


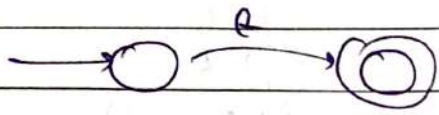
Such case

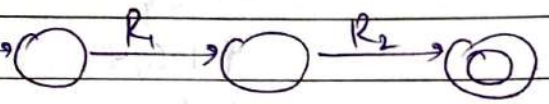
Minimization of DFA

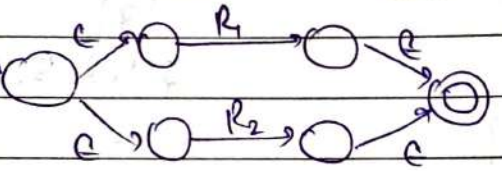


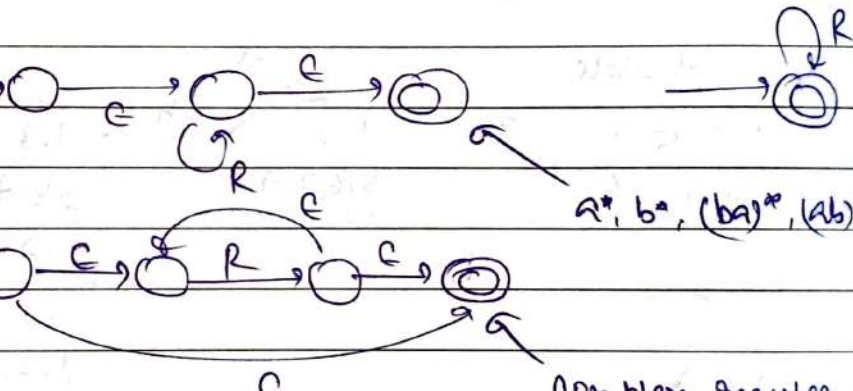
Regular Expression to FA

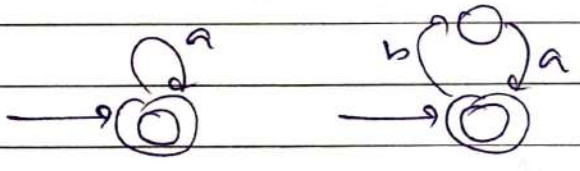
(i) $\emptyset$  (FA without final States)

(ii) ϵ  (when only FA has initial state as a final state then it will accept ϵ string)

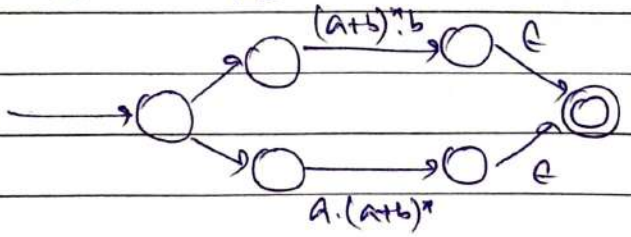
(iii) $R_1 . R_2$ 

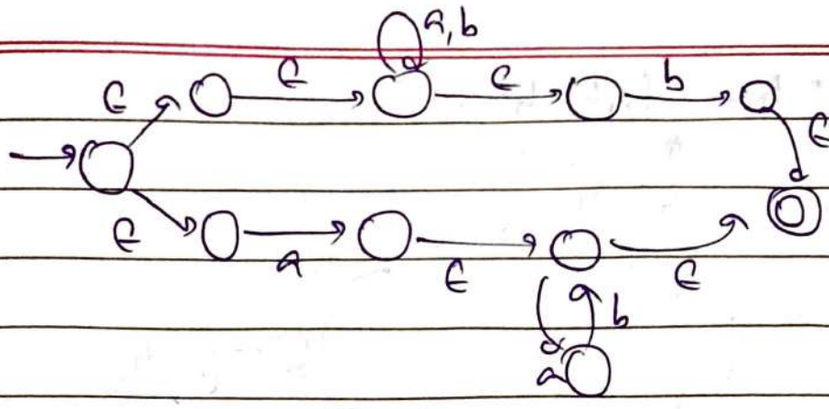
(iv) $R_1 + R_2 =$ 

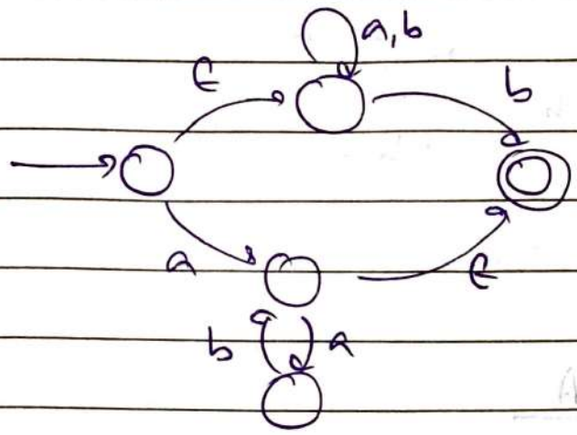
(v) $R^* =$ 



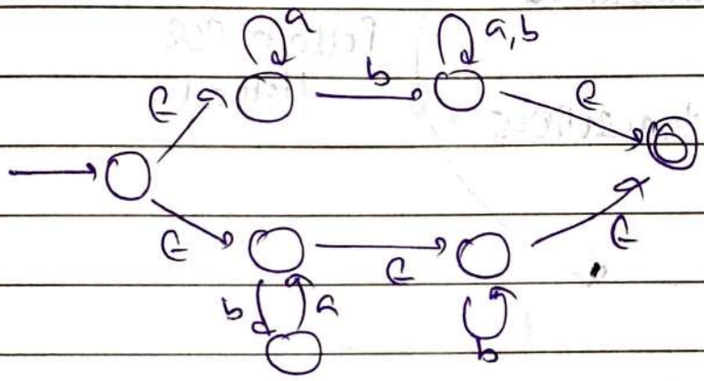
Eg $\underbrace{(a+b)^*b}_{R_1} + \underbrace{a(a+b)^*}_{R_2}$



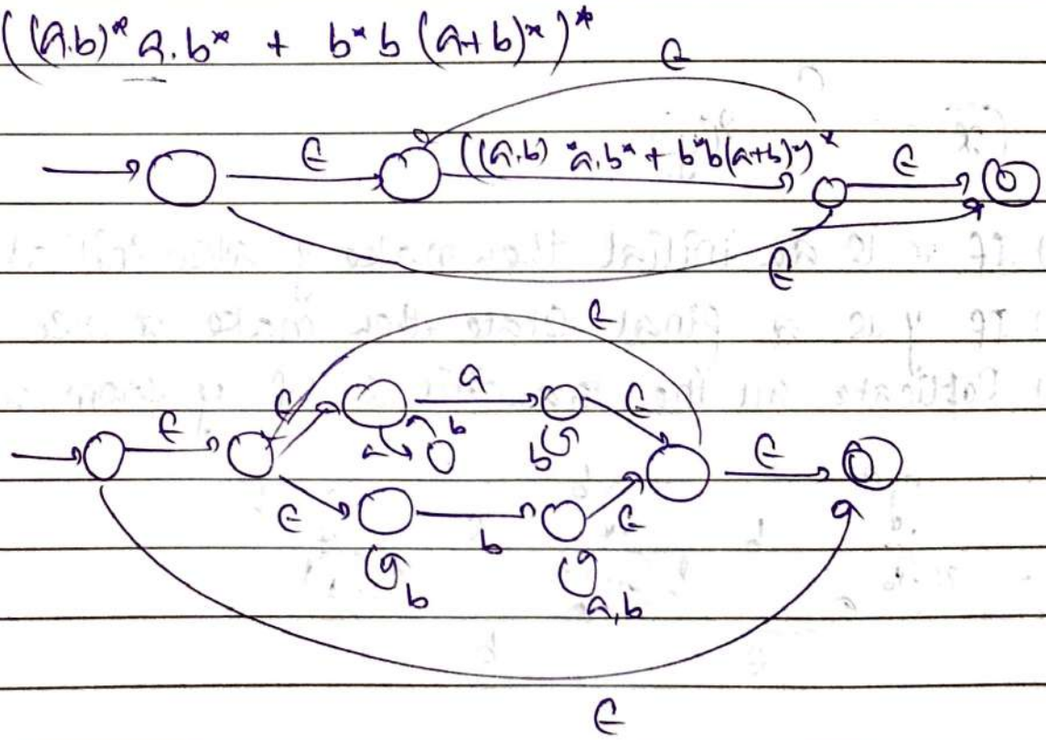




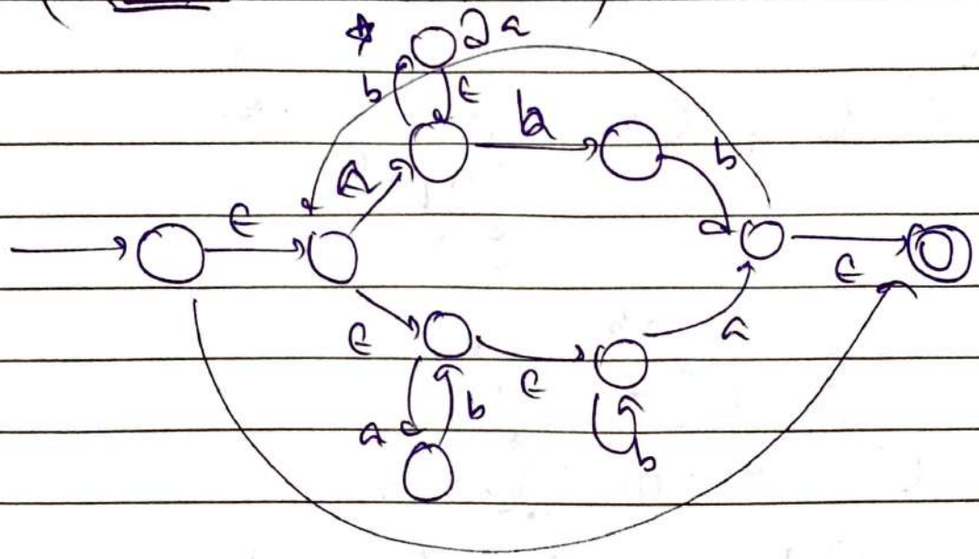
Eg- $a^*b(a+b)^* + (ba)^*b^*$



Eg- $((a.b)^*a.b^* + b^*b(a+b)^*)^*$



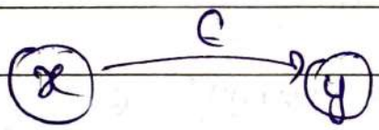
Eg. $(a.(b.a)^* b + (a.b)^* b^* a)^*$



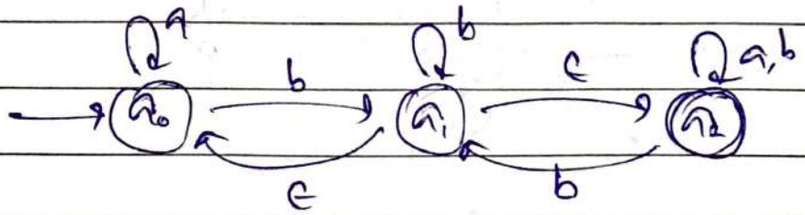
Remove ε- from NFA

R.E
↓
NFA with ε transitions
↓
NFA without ε transitions
↓
DFA
↓
Minimized DFA

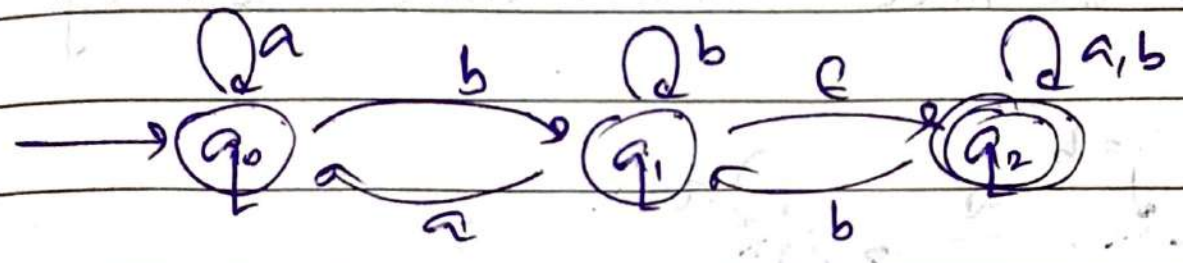
Follow POS Method.



(i) If x is an initial then make y also initial
(ii) If y is a final state then make x also final.
(iii) Replicate all the transitions of y from x.

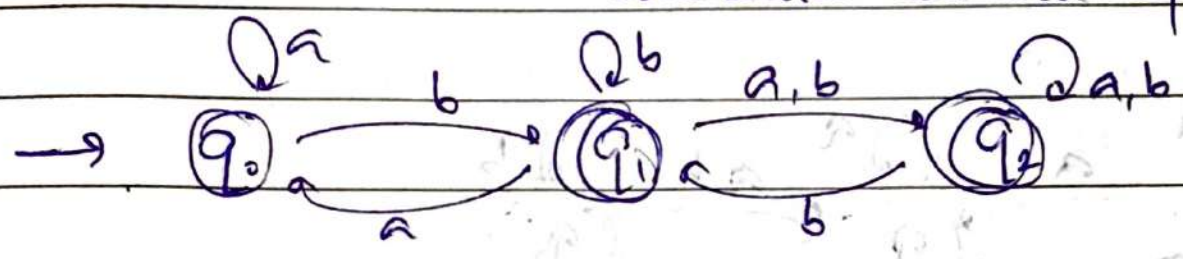


NOTE: Remove only one Null transition at a time.



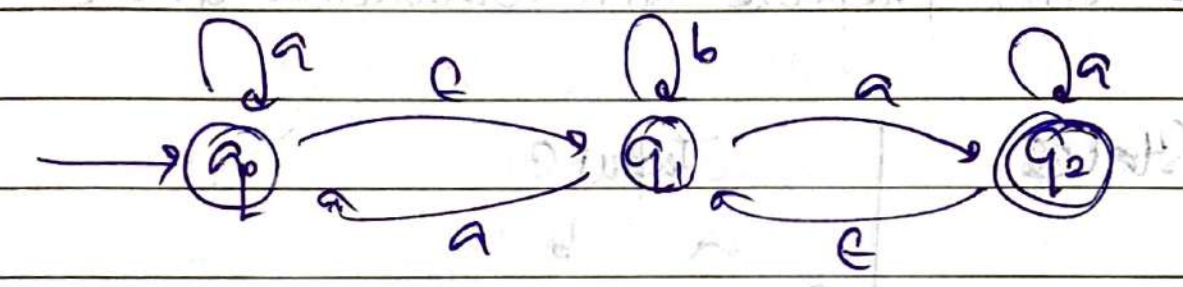
$q_1\ q_0 \xrightarrow{a} q_0$
$q_1\ q_0 \xrightarrow{b} q_1$

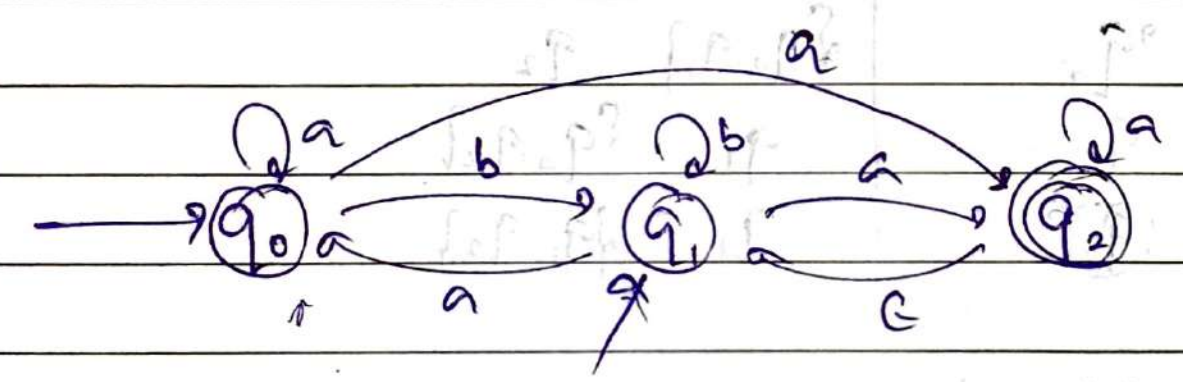
Even if there is a null transition we will replicate that too!



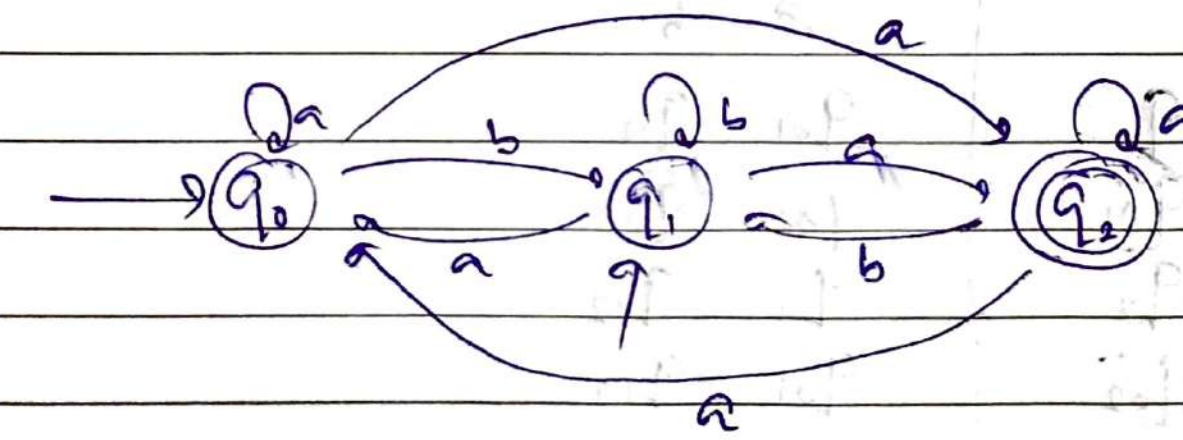
$q_1\ q_2 \xrightarrow{a} q_2$
$q_1\ q_2 \xrightarrow{b} q_1, q_2$

Eg-



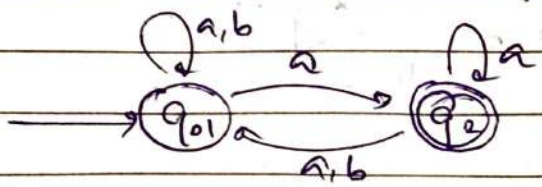


$q_0\ q_1 \xrightarrow{a} q_0, q_2$
$q_0\ q_1 \xrightarrow{b} q_1$

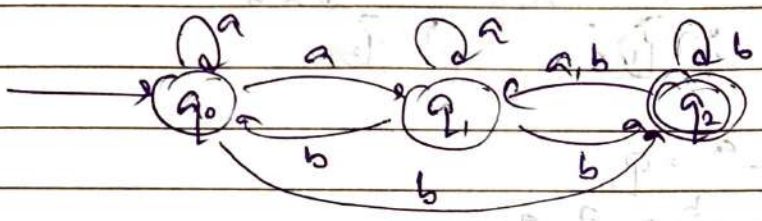


$q_2\ q_1 \xrightarrow{a} q_0, q_2$
$q_2\ q_1 \xrightarrow{b} q_1$

NOTE: Every FA should have exactly one initial state. In this case, we have 2 initial states, so merge them.



NFA (with ε transitions) to DFA
(Set-Subset Method)



(i) From given NFA, prepare NFA transition table.

States	Inputs	
	a	b
→ q_0	$\{q_0, q_1\}$	q_2
q_1	q_1	$\{q_0, q_2\}$
(*q_2)	q_1	$\{q_1, q_2\}$

(ii) DFA transition table.

States	Inputs	
	a	b
→ q_0	q_{01}	q_2
q_{01}	q_{01}	q_{02}
(*q_2)	q_1	q_{12}
(*q_{02})	q_{01}	q_{12}
q_{12}	q_1	q_{02}
(*q_{12})	q_1	q_{012}
(*q_{012})	q_{012}	q_{012}

q_{xyz} in a DFA then $\forall a \in \Sigma$, to prepare transition

$$\delta(q_{xyz}, a) = \delta(q_x, a) \cup \delta(q_y, a) \cup \delta(q_z, a)$$

Eg-

NFA transition table

States	Inputs	
	a	b
$\rightarrow q_0$	q_1	q_2
q_1	$\{q_0, q_2\}$	q_3
q_2	q_3	$\{q_0, q_2\}$
q_3	$\{q_1, q_2\}$	q_0

DFA transition table

States	Inputs	
	a	b
q_0	q_1	q_2
q_1	q_{02}	q_3
q_{02}	q_{13}	q_{02}
q_{13}	q_{012}	q_{03}
q_{03}	q_{12}	q_{02}
q_{12}	q_{023}	q_{023}
q_2	q_3	q_{02}
q_3	q_{12}	q_0
q_{012}	q_{0123}	q_{023}
q_{023}	q_{123}	q_{02}
q_{123}	q_{0123}	q_{023}
q_{0123}	q_{0123}	q_{023}

Let an NFA has 'N' States then it's equivalent DFA can have maximum ___ States

Trap State

NOTE:

Maximum No. of States in equivalent DFA = $2^N - X$.

$= \boxed{2^N}$

Trap State - Null transition.

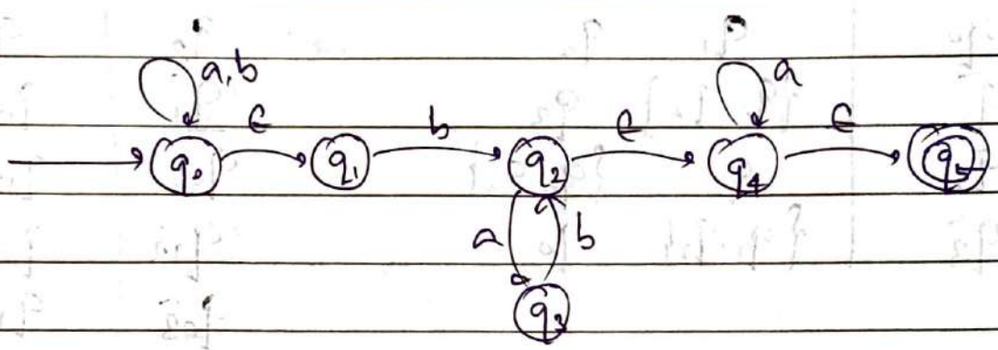
So, total States. $\boxed{2N}$ (max.)

ε - CLOSURE METHOD

Convert NFA (with ε-transition) to DFA

ε-closure ({x}) = Set of all states where we can reach from state x by using ε-transition.

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



DFA initial State = ε-closure ({initial State of NFA})

ε-closure ({q_0}) = A.

ε-closure of any state will include itself too!

A = ε-closure ({q_0}) = {q_0, q_1}

→ $S(A,a) = S(\{q_0,q_1\},a)$
$= \{q_0\}$

ε-closure ($S(A,a)$) = ε-closure ({q_0}) = A.

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

ε-closure ($S(A,b)$) = ε-closure ({q_0,q_2})
$= \{q_0,q_1,q_2,q_4,q_5\}$
$= B$

~~B = ε-closure ({q_0,q_1,q_2,q_4,q_5})~~

$B = \epsilon\text{-closure}(\{q_0, q_2\}) = \{q_0, q_1, q_2, q_4, q_5\}$

$\delta(B, a) = \delta(\{q_0, q_1, q_2, q_4, q_5\}, a)$
$= \{q_0, q_3, q_4\}$
$\epsilon\text{-closure}(\delta(B, a)) = \epsilon\text{-closure}(\{q_0, q_3, q_4\})$
$= \{q_0, q_1, q_3, q_4, q_5\}$
$C = \epsilon\text{-closure}(\{q_0, q_3, q_4\}) = \{q_0, q_1, q_3, q_4, q_5\}$

$\delta(B, b) = \delta(\{q_0, q_1, q_2, q_4, q_5\}, b)$
$= \{q_0, q_2\} = B$.
$\epsilon\text{-closure}(\delta(B, b)) = B$.

$\delta(C, a) = \delta(\{q_0, q_1, q_3, q_4, q_5\}, a) = \{q_0, q_4\}$
$\epsilon\text{-closure}(\delta(C, a)) = \epsilon\text{-closure}(\{q_0, q_4\})$
$= \{q_0, q_1, q_4, q_5\} = D$

$\delta(C, b) = \delta(\{q_0, q_1, q_3, q_4, q_5\}, b) = \{q_0, q_2\}$
$\epsilon\text{-closure}(\delta(C, b)) = \epsilon\text{-closure}(\{q_0, q_2\}) = B$

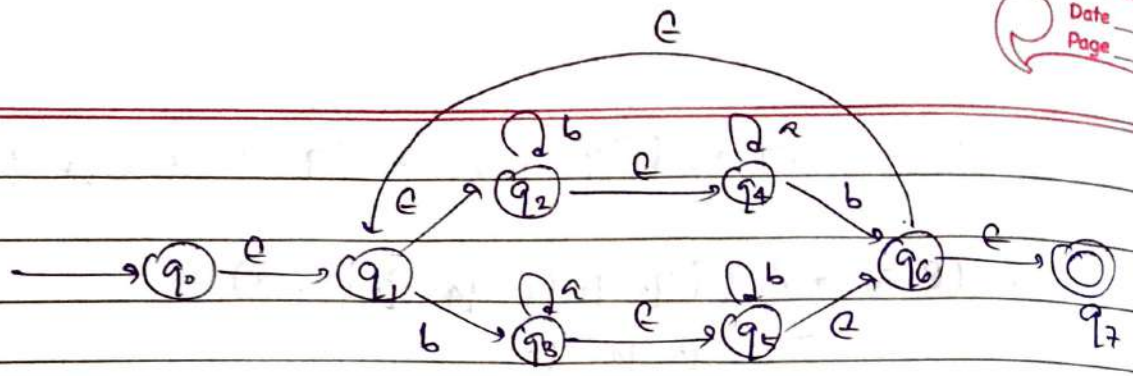
$D = \epsilon\text{-closure}\{q_0, q_4\}^E = \{q_0, q_1, q_4, q_5\}$

$\delta(D, a) = \delta(\{q_0, q_1, q_4, q_5\}, a) = \{q_0, q_4\}$
$\epsilon\text{-closure}(\delta(D, a)) = D$

$\delta(D, b) = \delta(\{q_0, q_1, q_4, q_5\}, b) = \{q_0, q_2\} = B$

States	Input a	Input b
→ A	A	B
(B)	C	B
(C)	D	B
(D)	D	B

Eg-



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

$S(A, a) = S(\{q_0, q_1, q_2, q_4\}, a)$
$= \{q_4\}$
$\epsilon\text{-closure}(S(A, a)) = \epsilon\text{-closure}(\{q_4\}) = \{q_4\} = B.$
$S(A, b) = S(\{q_0, q_1, q_2, q_4\}, b)$
$= \{q_3, q_2, q_6\}$
$\epsilon\text{-closure}(S(A, b)) = \epsilon\text{-closure}(\{q_3, q_2, q_6\})$
$= \{q_3, q_5, q_6, q_7, q_2, q_4, q_1\}$
$= C.$

- $B = \epsilon\text{-closure}(\{q_4\}) = \{q_3, q_2, q_6\}\ \{q_4\}.$
- $C = \epsilon\text{-closure}(\{q_3, q_2, q_6\}) = \{q_1, q_2, q_3, q_4, q_5, q_6, q_7\}$

$S(B, a) = S(\{q_4\}, a) = \{q_4\}.$
$\epsilon\text{-closure}\{q_4\} = q_4 = B.$
$S(B, b) = S(\{q_4\}, b) = \{q_6\}.$
$\epsilon\text{-closure}(\{q_6\}) = \{q_1, q_6, q_7, q_2, q_4\} = D.$

- $D = \epsilon\text{-closure}(\{q_6\})\ q_2, q_4, q_6, q_7.$
$= \{q_1, q_2, q_4, q_6, q_7\}$

$S(C, a) = S(\{q_1, q_2, q_3, q_4, q_5, q_6, q_7\}, a)$
$= \{q_3, q_4\}$
$\epsilon\text{-closure}\{q_3, q_4\} = \{q_3, q_5, q_6, q_1, q_7, q_2, q_4\}$
$= C$

$\delta(C,b) = \delta(\{q_1,q_2,q_3,q_4,q_5,q_6,q_7\}, b)$

$= \{q_3, q_5, q_2, q_6\}$

ϵ-closure $\{q_2,q_3,q_5,q_6\} = \{q_2,q_4,q_6,q_7,q_1,q_3,q_5\}$

$= C$

$\delta(D,a) = \delta(\{q_1,q_2,q_4,q_6,q_7\}, a)$

$= \{q_4\}$

ϵ-closure $\{q_4\} = \{q_4\} = B$.

$\delta(D,b) = \delta(\{q_1,q_2,q_4,q_6,q_7\}, b)$

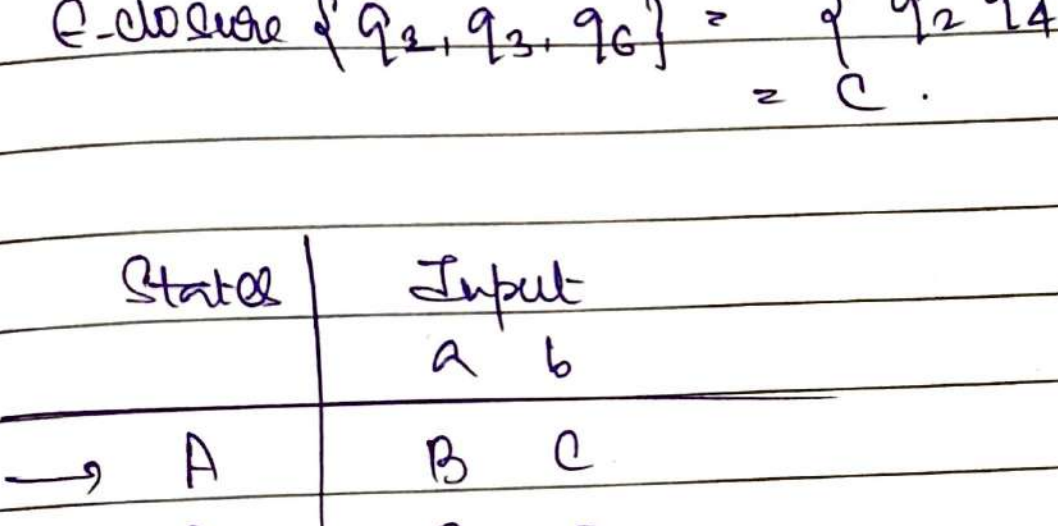
$= \{q_3,q_2,q_6\}$ = E.

ϵ-closure $\{q_2,q_3,q_6\} = \{q_2\,q_4,q_6,q_7,q_3,q_5,q_1\}$

$= C$.

States	Input a	Input b
→ A	B	C
B	B	D
(C)	C	C
(D)	B	C

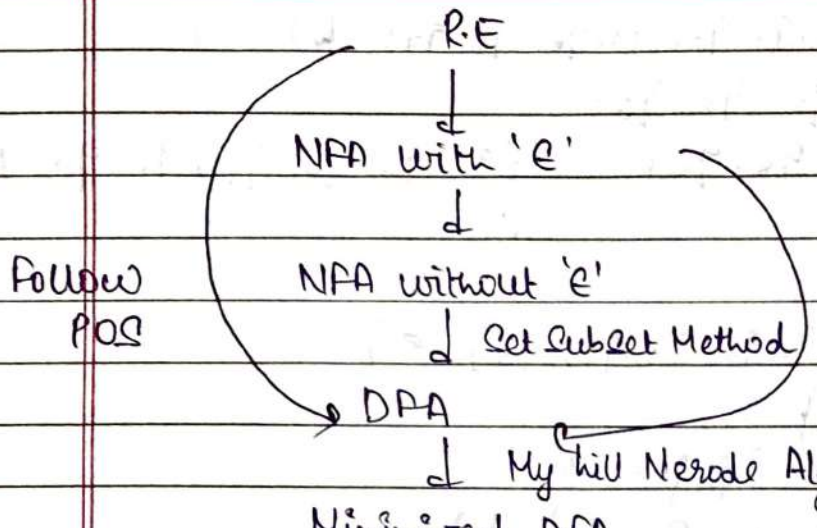
Eg- $((a.b)^*(b.a)^* + a^*b.a^*)$



- A = ϵ-closure $\{q_0\} = \{q_0,q_1,q_3,q_7\}$

$\delta(A,a) = \delta(\{q_0,q_1,q_3,q_7\}, a) = \{q_2\}$

ϵ-closure $\{q_2\} = T$



Every regular language has a unique minimized DFA that accepts all the strings of it.

⇓

Minimization of DFA : Myhill Nerode Algorithm

In minimization of DFA, we find equivalency between states of DFA.

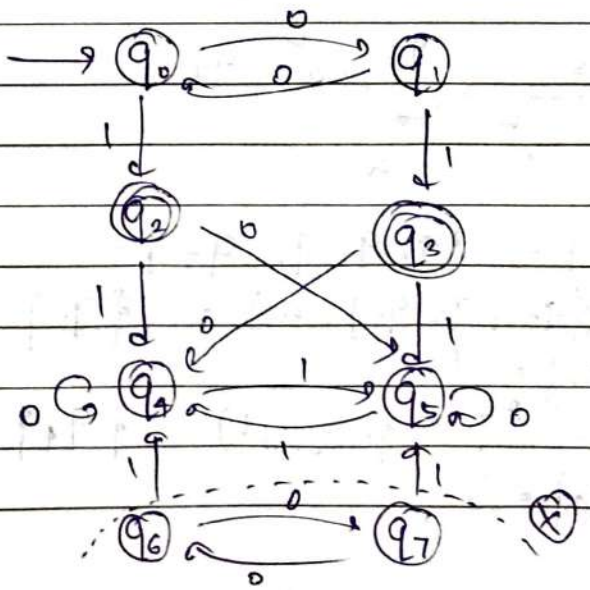
Let states q_i & $q_j \in Q$ can be equivalent.

(i) Both q_i and q_j are either final or non-final states.

(ii) $\forall a \in \Sigma$, $\delta(q_i, a^*)$ = F NA NF F

& $\delta(q_j, a^*)$ = F NA F NF

✓ ✓ × ×



Dead state: State which is not connected to initial state directly or indirectly.

- If any dead state, before minimization of DFA, all of them are removed.

- Both q_6 and q_7 are removed from DFA along with their transitions.

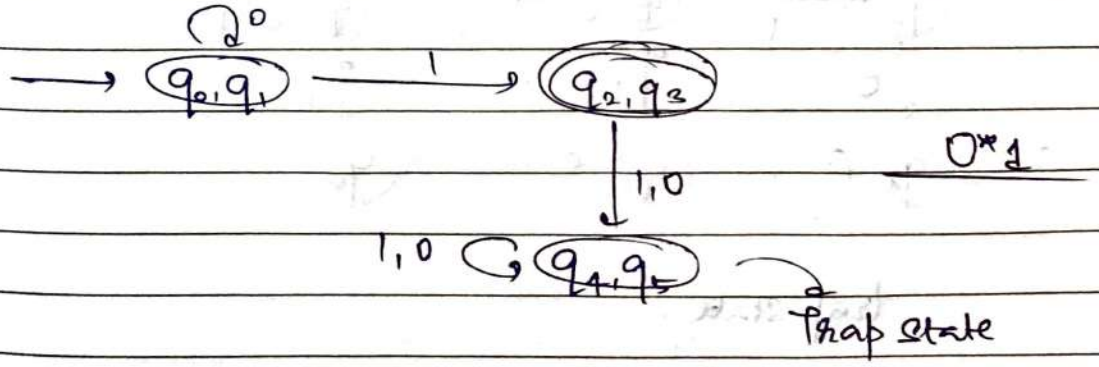
	q_5	q_4	q_3	q_2	q_1
q_0	X	X	X	X	(q_0, q_1)
q_1	X	X	X	X	
q_2	X	X	(q_0, q_1)		
q_3	X	X			
q_4	(q_2, q_3) (q_4, q_5)				

X → F and NF

State-transition equivalency.

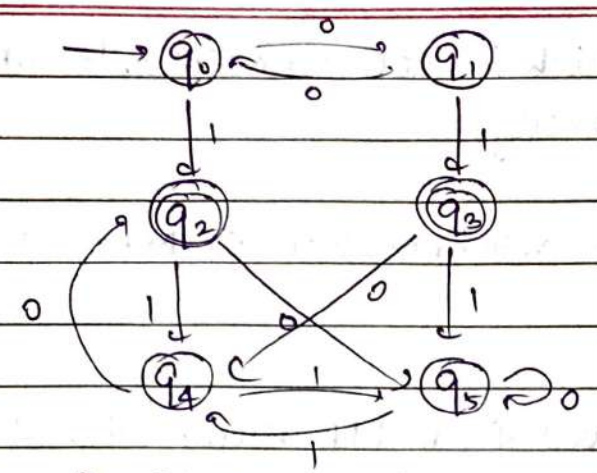
$q_0 \xrightarrow{0}$ (NF) q_1 — At destination location, place the entry of source location.
$q_1 \xrightarrow{0}$ (NF) q_0
$q_0 \xrightarrow{1}$ (F) q_2
$q_1 \xrightarrow{1}$ (F) q_3

$q_2 \xrightarrow{0} q_4$ (NF)
$q_3 \xrightarrow{0} q_5$ (NF)



0^*1

Eg-



	q_5	q_4	q_3	q_2	q_1
q_0	X	X	X	X	~~X~~
q_1	X	X	X	X	
q_2	X	X	~~X~~		
q_3	X	X			
q_4	X				

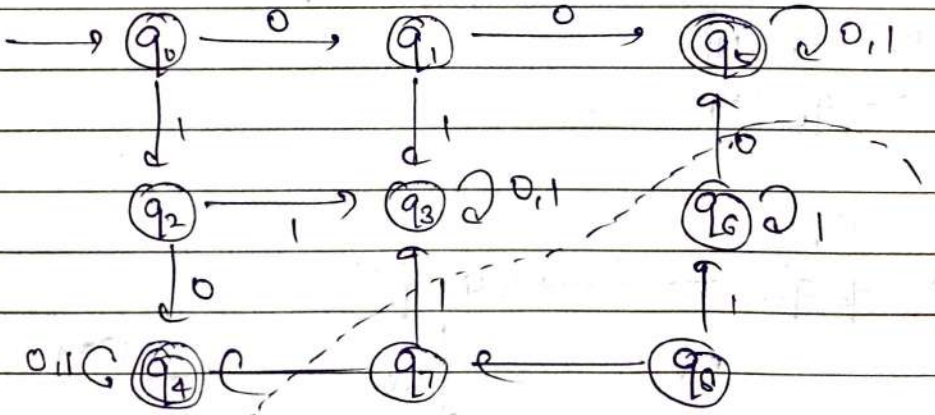
$q_0 \xrightarrow{0} q_1$
$q_1 \xrightarrow{0} q_0$
$q_0 \xrightarrow{1} q_2$
$q_1 \xrightarrow{1} q_3$

$q_4 \xrightarrow{0} q_2$ ✓
$q_5 \xrightarrow{0} q_5$ ✗

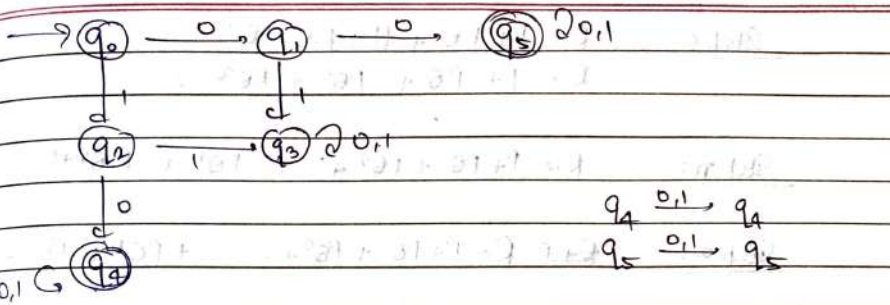
Already minimized.

$q_0 \xrightarrow{110}$ F ✗
$q_1 \xrightarrow{110}$ NF

Test*



trap state.



$q_4 \xrightarrow{0,1} q_4$

$q_5 \xrightarrow{0,1} q_5$

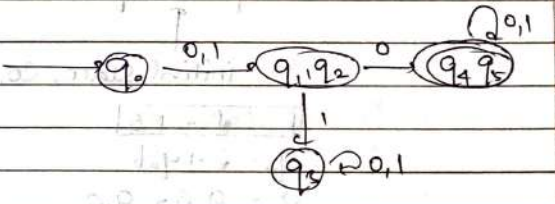
	q_5	q_4	q_3	q_2	q_1
q_0	X	X	X	X	X
q_1	X	X	X	X	
q_2	X	X	X		
q_3	X	X			
q_4	(q_4,q_5)				

$q_0 \xrightarrow{0} q_1$ $\quad q_0 \xrightarrow{1} q_2$

$q_3 \xrightarrow{0} q_2$ $\quad q_3 \xrightarrow{1} q_3$

8 state)

	q_5	q_4	q_3	q_2	q_1
q_0	X	X	X	X	X
q_1	X	X	X	✓	
q_2	X	X	X		
q_3	X	X			
q_4	(q_1,q_2) (q_4,q_5)				



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

FA to Regular Expression

ARDEN'S THEOREM

$$R = P + R.Q = P.Q^*$$

Proof: $R = P + R.Q$

Step 1 $R = P + (P + R.Q).Q$

$R = P + P.Q + R.Q^2$

Step 2 $R = P + P.Q + (P + R.Q).Q^2$

$R = P + P.Q + PQ^2 + RQ^3$

Step n: $R = P + PQ + PQ^2 + \ldots PQ^n + R.Q^{n+1}$

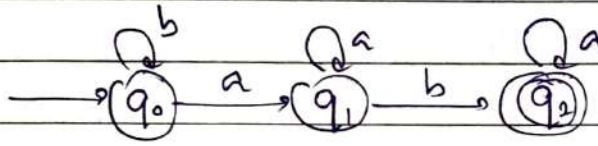
Step ∞: ~~R+P~~ $R = P + PQ + PQ^2 + \ldots + PQ^n + PQ^\infty + RQ^\infty$

$R = P + PQ + PQ^2 + PQ^3 + \ldots$

$= P(\epsilon + Q + Q^2 + Q^3 + \ldots)$

$\boxed{R = P.Q^*}$

Eg-



$q_0 = \epsilon + q_0.b$ equation of each state - all the incoming transitions

↑ initial state, S_0, ϵ.

$\boxed{R = P + R.Q}$

$q_0 = \epsilon + q_0.b$ —(1.)

$q_1 = q_0.a + q_1.a$ —(2.)

$q_2 = q_1.b + q_2.a + q_2.b$ —(3.)

Applying Arden's th. on eq. (3.)

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

$R = P + R.Q$

$q_2 = q_1.b\,(a+b)^*$

Applying Arden's th. on eq. (2)

$q_1 = q_0 a.a^*$

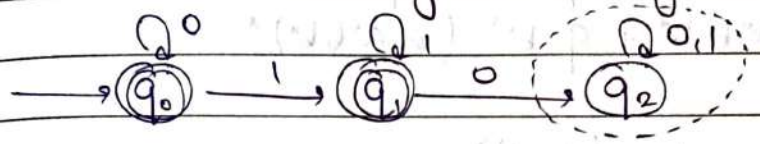
Applying Arden's th. on eq (1.)

$q_0 = b^*$

$q_0 = b^*$ $q_1 = b^*a.a^*$ $q_2 = b^*a.a^*b(a+b)^*$

↓

Strings containing ab as substring.



NOTE: Before applying Arden's theorem remove all trap and dead state.

$\forall q_i \in F$ (many final states)

$= q_0 + q_1 + \ldots\ldots q_n$

(connect the final expression of all the final states with '+' operators)

$q_0 = \epsilon + 0.q_0$

$q_1 = q_0 1 + 1.q_1$

$q_0 = 0^*$

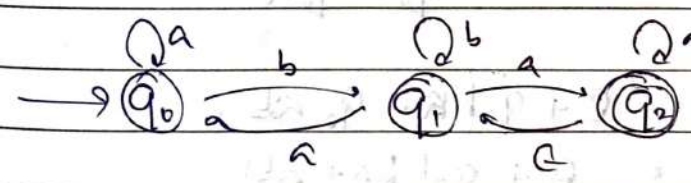
$q_1 = q_0.1.1^*$

$q_1 = 0^*.1.1^*$

$= 0^* + 0^*.1.1^*$

$= 0^*(\epsilon + 1.1^*)$

$= 0^*.1^*$



$q_0 = \epsilon + q_0.a + q_1 a$

$q_1 = q_0.b + q_1.b + q_2.\epsilon$

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

$q_0 = (\epsilon + q_1 a).a^*$

$q_1 = (q_0 b + q_2 \epsilon).b^*$

$q_2 = q_1 a.(a+b)^*$

$q_1 = (q_0 b + q_2 \epsilon) \cdot b^*$

$= ((\epsilon + q_1 a)\, a^* \cdot b + q_2 \epsilon)\, b^*$

$= (\epsilon + q_1 a)\, a^* \cdot b \cdot b^* + q_2 \epsilon b^*$

$= a^* b \cdot b^* + q_2 b^* + q_1 \cdot a \cdot a^* \cdot b \cdot b^*$

$q_1 = (a^* b \cdot b^* + q_2 b^*)\,(a \cdot a^* b b^*)^*$

$q_2 = q_1 a\,(a+b)^*$

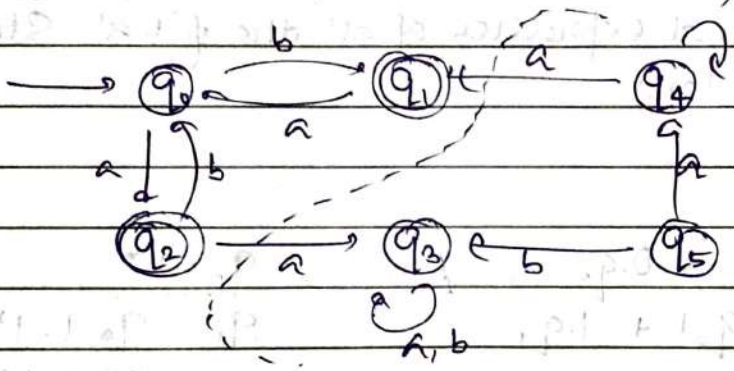
$= (a^* b b^* + q_2 b^*)\, a\,(a+b)^*$

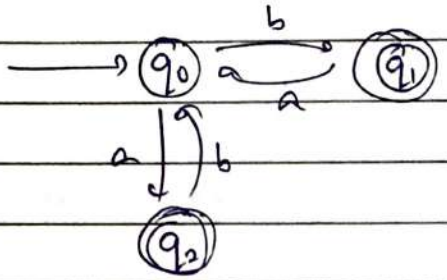
$= a^* b b^* a (a+b)^* + q_2 b^* a (a+b)^*$

$q_2 = a^* b b^* a (a+b)^* \,(b^* a (a+b)^*)^*$

Ans- $q_2 = a^* b (a \cdot a^* b + a(a+b)^* + b)^* a\,(a+b)^*$

Eg-





$q_0 = \epsilon + q_1 a + q_2 b.$

$q_1 = q_0 b$

$q_2 = q_0 a$

$q_0 = \epsilon + q_0 b a + q_0 a b.$

$= \epsilon + q_0 (ba + ab)$

$q_0 = (ba + ab)^*$

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

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

Final solution → $q_1 + q_2$

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

REGULAR GRAMMAR

Any grammar G can be defined as $G(V, T, S, P)$

V: Set of finite variables (Non-terminals)

T: Set of finite terminals

S: starting variable $S \in V$

P: Set of finite production.

Set of productions - Set of rules for the grammar.

Regular Grammar

Left Linear Regular (Derivation grows in left direction)	Right Linear Regular (Derivation grows in right direction)
$\forall P$: $A \to Ba$ or $A \to a$	$\forall P$: $A \to aB$ or $A \to a$
where $A, B \in V$	where $A, B \in V$
$a \in T^*$	$a \in T^*$

Derivation - generate string from grammar.

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

Right Linear $S \to aS \mid bS \mid \epsilon$

left Linear $S \to Sa \mid Sb \mid \epsilon$

Eg- $L = \{ a^n \mid n \geq 1 \}$

RL:

$S \rightarrow aS \mid a$

LL:

$S \rightarrow Sa \mid a$

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

RL:

$S \rightarrow aS \mid A$

$A \rightarrow bA \mid \epsilon$

LL:

$S \rightarrow Sb \mid A$

$A \rightarrow Aa \mid \epsilon$

Eg- $L = \{ a^n b^m \mid n \geq 1, m \leq 2 \}$.

RL:

$S \rightarrow aS \mid aA$

$A \rightarrow b \mid bb \mid \epsilon$

LL:

$S \rightarrow Ab \mid Abb \mid A$

$A \rightarrow Aa \mid a$

OR

$S \rightarrow Aabb \mid Aab \mid Aa$

$A \rightarrow Aa \mid \epsilon$

Eg- $L = \{ a^{2n} \cdot b^{2m+1} \mid n, m \geq 0 \}$

RL:

$S \rightarrow aaS \mid bA$

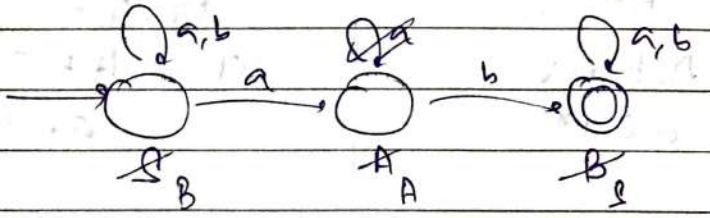
$A \rightarrow bbA \mid \epsilon$

LL:

$S \rightarrow Sbb \mid Ab$

$A \rightarrow Aaa \mid \epsilon$

Eg- All the strings of a, b have substring ab.



Right Linear :

(i.) Define variable name for each state.

(Starting state by starting variable)

(ii) For all transitions of type (q_i, a)

$\forall \delta(q_i, a) \longrightarrow q_j$

(A) (B)

Write a production $q_i \longrightarrow aq_j$

ie, $A \longrightarrow aB$.

(iii) For every finite state variable add an extra production 'ϵ'.

$S \longrightarrow aS \mid bS \mid aA$

$A \longrightarrow bB$

$B \longrightarrow aB \mid bB \mid \epsilon$

(RL)

Left linear:

(i) Define variable name for each state (final state by starting variable).

(Traverse each transition in reverse order).

(ii) For every $\delta(q_i, a) \longrightarrow q_j$

(A) (B)

Production $q_j \longrightarrow q_i a$

$B \longrightarrow Aa$

(iii) Add an extra production 'ϵ' for variable (starting state variable).

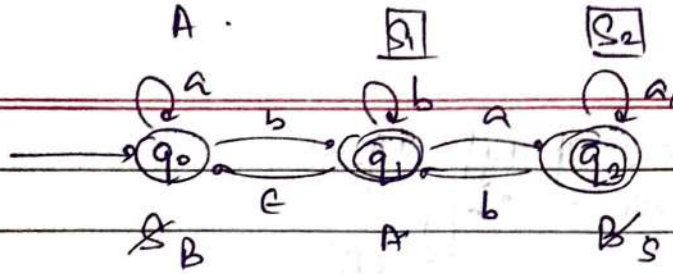
$S \longrightarrow Sa \mid Sb \mid Ab.$

$A \longrightarrow Ba$

$B \longrightarrow Ba \mid bBb \mid \epsilon$

(LL)

Eg-



RL: $S \rightarrow aS \mid bA$

$A \rightarrow bA \mid aB \mid S \mid \epsilon$

$B \rightarrow aB \mid bB \mid bA \mid \epsilon$

LL: $S \rightarrow Sa \mid Sb \mid Aa$

$A \rightarrow Ab \mid bSb \mid Bb$

$B \rightarrow Ba \mid A \mid \epsilon$

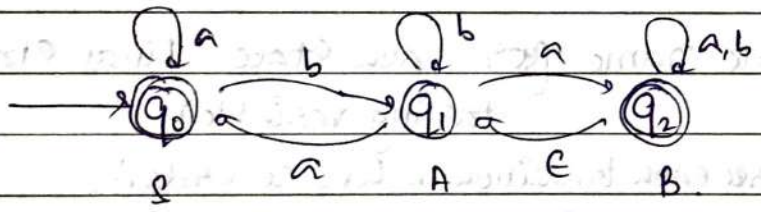
* $S \rightarrow S_1 \mid S_2$

$S_1 \rightarrow S_1b \mid Ab \mid S_2b$

$S_2 \rightarrow S_2a \mid S_2b \mid S_1a$

$A \rightarrow Aa \mid S_1 \mid \epsilon$

Eg-



RL: $S \rightarrow aS \mid bA \mid \epsilon$

$A \rightarrow bA \mid aS \mid aB$

$B \rightarrow bB \mid aB \mid \epsilon A \mid \epsilon$

LL: $S \rightarrow S_1 \mid S_2$

$S_1 \rightarrow S_1a \mid Aa \mid \epsilon$ ← Starting State.

$S_2 \rightarrow S_2a \mid S_2b \mid Aa$

$A \rightarrow Ab \mid S_1b \mid S_2\epsilon$

Regular Grammar to FA

Right Linear to FA

(i) All the production of grammar should be in the form:

$A \longrightarrow aB$ or $A \longrightarrow \in$

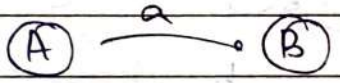
where $A, B \in V$ & $a \in T$

$A \longrightarrow abB$ ⊗ Not applicable → Convert.

(ii) Every variable of grammar represent a state of FA. (starting variable represents an initial state of FA)

(iii) All the variables which have '$\in$' production that represent final state of FA.

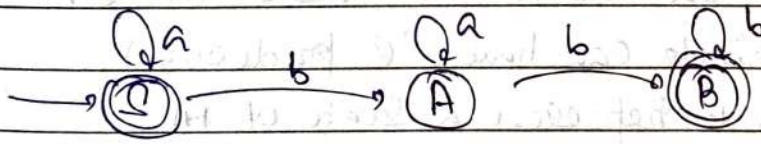
(iv) For every production of grammar $A \longrightarrow aB$, create a transition & state)



eg- $S \longrightarrow aS \mid bA \mid \in$ → final state.

$A \longrightarrow aA \mid bB$

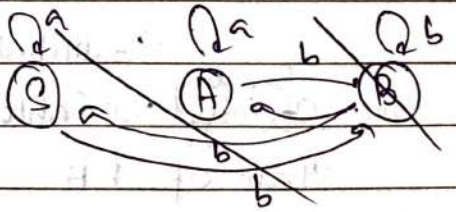
$B \longrightarrow bB \mid \in$



$S \longrightarrow aS \mid bB \mid aX$

$A \longrightarrow aA \mid bB \mid \in$

$B \longrightarrow bB \mid bA \mid aX$

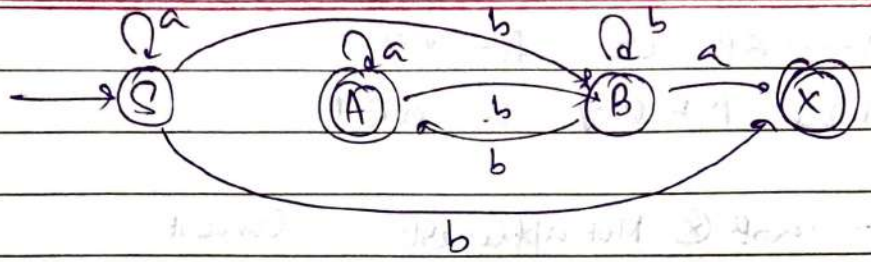


↓

Already in the required form

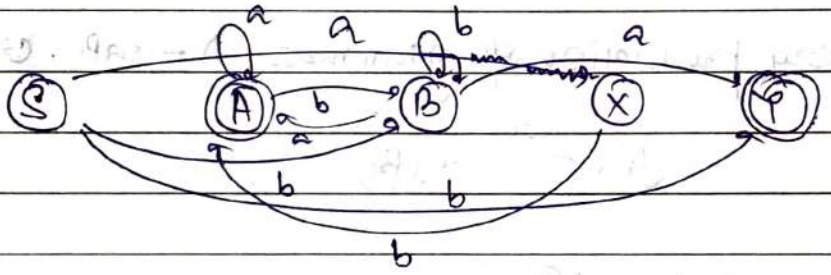
$X \longrightarrow \in$ ↴

Now all the productions are in required form.



Eg. $S \rightarrow aA \mid bB \mid b$
$A \rightarrow aA \mid bB \mid \epsilon$
$B \rightarrow bB \mid aA \mid a$.

$S \rightarrow aX \mid bB \mid bY$
$A \rightarrow aA \mid bB \mid \epsilon$
$B \rightarrow bB \mid aA \mid aY$
$X \rightarrow bA$
$Y \rightarrow \epsilon$



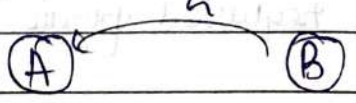
Left Linear to FA

(i) All the productions of grammar should be in the form $A \rightarrow Ba$ or $A \rightarrow \epsilon$ where $A, B \in V$ & $a \in T$ (only one variable can have 'ϵ' production)

(ii) Every variable represent a State of FA.
(variable which has 'ϵ' production represents an initial State of FA)

(iii) ~~Strictly~~ Starting variable of grammar represents final State of F.A

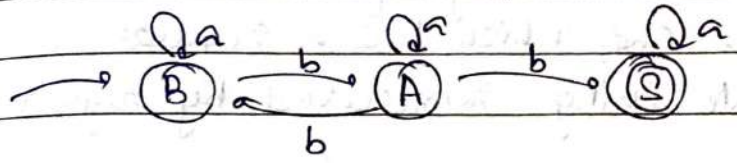
(iv) For every production $A \rightarrow Ba$, create a transition



$S \rightarrow Sa \mid Ab$

$A \rightarrow Aa \mid Bb$

$B \rightarrow Ba \mid Ab \mid \epsilon$



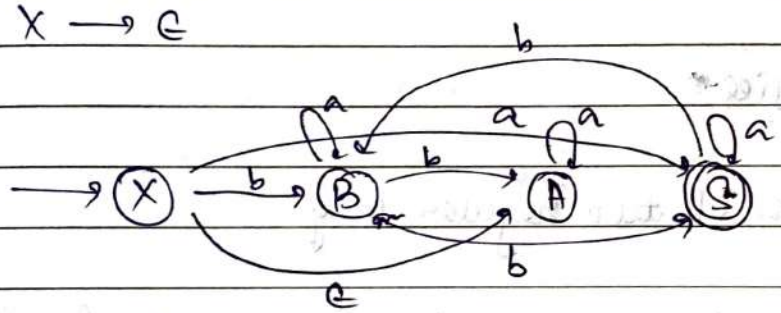
Eg

$S \rightarrow Sa \mid Bb \mid Xa$ → Xa

$A \rightarrow Aa \mid Bb \mid X\epsilon$ → Xε

$B \rightarrow Sb \mid Ba \mid Xb$ → Xb

$X \rightarrow \epsilon$



Eg.

(X)

$S \rightarrow Aaba \mid Sb$

$A \rightarrow Ab \mid Ba \mid b$

$B \rightarrow Bba \mid Aa \mid \epsilon$

(Y)

(Z)

$S \rightarrow Xba \mid Sb$

$A \rightarrow Ab \mid Ba \mid b$

$B \rightarrow Ya \mid Aa \mid \epsilon$

$X \rightarrow Aa$

$Y \rightarrow Bb.$

$S \rightarrow Za \mid Sb$

$A \rightarrow Ab \mid Ba \mid Wb$

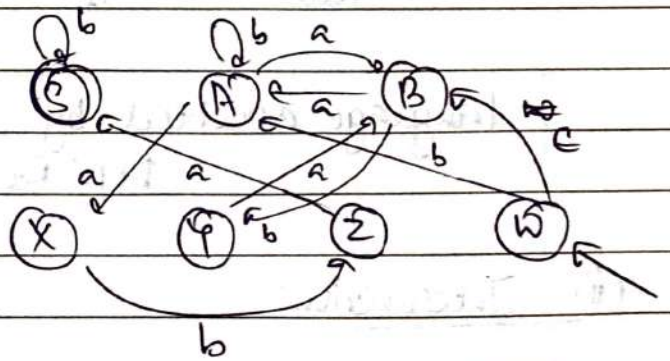
$B \rightarrow Ya \mid Aa \mid W\epsilon$

$X \rightarrow Aa$

$Y \rightarrow Bb$

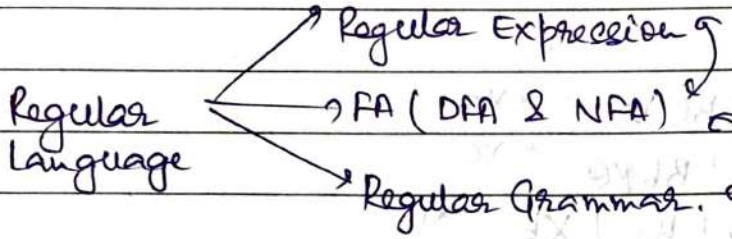
$Z \rightarrow Xb$

$W \rightarrow \epsilon$



REGULAR LANGUAGE

Regular language is a language for which there exists a finite automata and which has a regular grammar which accepts all the strings represented by regular lang.



Closure properties:

(i) **Union:** Union of two regular lang.

Let R_1 and R_2 are two regular languages, then $R_1 \cup R_2$ is also regular.

$$R_1 \cup R_2 = R_3.$$

Proof: Let M_1 is a DFA corresponding to R_1 and M_2 for R_2.

$$M_1 = DFA(R_1)$$
$$M_2 = DFA(R_2)$$

Now, $M_1 \cup M_2 = M_3$ is also a DFA.

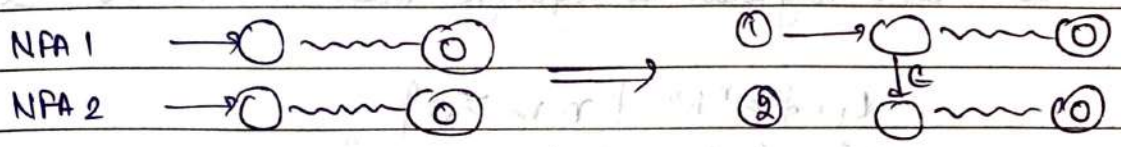
Language accepted by DFA M_3 will be R_3

$$DFA(R_3) = M_3.$$

(ii) **Intersection:**

Let R_1 and R_2 be two regular lang, $R_1 \cap R_2 = R_3$ is also regular.

Proof: (Same)



(iii) Set Difference:

Let R_1 and R_2 be two regular languages. $R_1 - R_2 = R_3$ is also regular.

$$R_3 = R_1 - R_2 = R_1 \cap \overline{R_2}$$

DFA is closed under complementation. — (Change final to NF & NF to final state)

So R_3 is also regular.

(iv) Complement:

Let R_1 is a regular language then $\overline{R_1} = R_2$ is also regular language.

Proof: Let M_1 is a DFA for regular lang. R_1

$$M_1(Q, q_0, \Sigma, \delta, F) = DFA(R_1)$$

Let M_2 is another DFA that is obtained by changing final states to non-final states and non-final states to final states in DFA(M_1)

$$M_2(Q, q_0, \Sigma, \delta, Q - F)$$

Now, regular language accepted by M_2 will be

$$R_2 = \overline{R_1}.$$

(v) Reversal :

Let R_1 is a regular language then $R_1^R = R_2$ is also regular.

$$L_1 = \{ a^n b^m \mid n, m \geq 0 \}$$
$$L_1^R = \{ b^m, a^n \mid n, m \geq 0 \}$$ ← reverse all the strings.

Proof: (Change F to Initial and Initial to F)
And reverse the direction of all the transitions.

Let M_1 is a DFA for R_1
Change final states to an initial state and initial to final and also change the direction of all transition of DFA M_1 we get another (FA) M_2. lang. accepted by M_2 will be $R_2 = R_1^R$.

(vi) Kleen's Closure (*) :

Let R_1 is a regular language then $R_1^* = R_2$ is also regular.

Proof: let M_1 is an FA for regular language R_1
$M_1 = FA(R_1) = (\theta, q_0, \Sigma, S, F)$.

Construct a new FA M_2 as
$M_2 = (\theta, q_0, \Sigma, S_2, F_2)$
when $S_2 (S \cup \{ S(q_i, \epsilon) \longrightarrow q_0$ where $q_i \in f \}$
& $F_2 = F \cup \{q_0\}$

Language accepted by M_2 is $R_2 = R_1^*$

(vii) Concate:

Let R_1 & R_2 are two regular languages. Then $R_1.R_2 = R_3$ is also a regular language.

Proof: let M_1 and M_2 are two FA for R_1 and R_2 respectively

$M_1 = FA(R_1) = (Q_1, q_x, \Sigma_y, \delta_1, f_1)$

and $M_2 = FA(R_2) = (Q_2, q_y, \Sigma_z, \delta_2, f_2)$

Now, construct new FA $M_3(Q_3, q_x, \Sigma_3, \delta_3, f_3)$

Where, $Q_3 = Q_1 \cup Q_2$

$\Sigma_3 = \Sigma_1 \cup \Sigma_2$

$\delta_3 = \delta_1 \cup \delta_2 \cup \{ \delta(q_i, \epsilon) \longrightarrow q_y \text{ where } q_i \in F_1 \}$

$f_3 = f_2$

language accepted by DFA M_1 will be $R_3 = R_1.R_2$.

Non-Regular languages

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

* Finite automata is a memoryless machine which cannot compare two or more variables.
* Every finite language is regular.

2. $L = \{ a^n b^m \mid n \neq m \}$ NR
   (Because we have to compare no. of a's & b's).
3. $L = \{ a^n b^m c^k \mid n + m = k \}$ NR.
4. $L = \{ w.w^R \mid w \in \{a,b\}^* \}$ NR
   (compare w with w^R).
5. $L = \{ wcw^R \mid c \text{ is a terminal & } w \in \{a,b\}^* \}$
   NR

6. $L = \{ w x w^R \mid x, w \in \{a,b\}^+ \}$ R

w — x — w^R

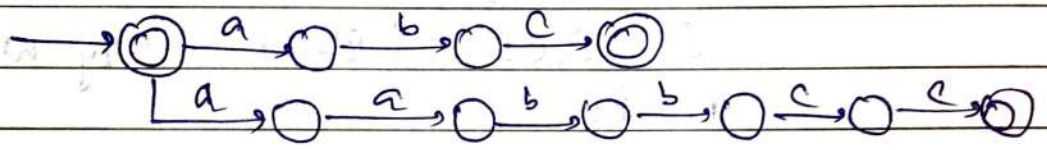
$\underbrace{abb}_{w}\ \underbrace{ab}_{x}\ \underbrace{bba}_{w^R}$ $\quad a(a+b)^+a + b(a+b)^+b$

7. $L = \{ w x w^R \mid x, w \in \{a,b\}^+ \}$ NR

x fixed = 10 ← 10

8. $L = \{ a^n b^n c^n \mid n \le 2^{100} \}$ R

finite (n is finite number)



9. $L = \{ a^n b^m \mid n+m \ge 2^{100} \}$ ~~NR~~

$n+m \ge 5$

no. of combinations are finite.

n	m	
0	5	$a^* bbbbb b^* +$
1	4	$aa^* bbbb b^* +$
2	3	$aaa^* bbb b^* +$
3	2	$aaaa^* bb b^* +$
4	1	$aaaaa^* b b^* +$
5	0	$aaaaaa^* b^*$.

10. $L = \{ w_1 c w_2 \mid w_1, w_2 \in \{a,b\}^* \}$ R

c is a terminal

w_1 and w_2 are independent of each other.

11. $L = \{ wcw \mid w \in \{a,b\}^* \}$ NR

c is a terminal.

→ we have to first store w.

12. $L = \{ ww \mid w \in \{a,b\}^* \}$ N.R

13. $L = \{ a^p \mid p \text{ is prime} \}$ N.R

We need to calculate p, we require memory.

Context free languages

Context free grammar $G(V, T, S, P)$

V: Set of finite variables.

T: Set of finite terminals.

S: Starting variable $S \in V$.

P: Set of finite productions.

$\forall P : A \rightarrow \alpha$

where $A \in V$

$\alpha \in (V \cup T)^*$

For every context free lang., there exists a context free grammar.

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

$S \rightarrow aSb \mid \in$

S

aSb

Left most 'a' ↓ Right most 'b'

$aaSbb$

$aaaSbbb$

⋮

$aAabbb\ldots$ (S→∈)

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

$S \rightarrow aSbb \mid abb$ (n ≥ 1)

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

~~$S \rightarrow aSbSc \mid abc \in$~~

$S \rightarrow aSc/A$ – same no. of a's and c's

$A \rightarrow bA \mid \in$

Eg- $L = \{a^n b^n c^m \mid n, m \geq 1\}$

$S \longrightarrow aSb \mid abA$

$A \longrightarrow cA \mid c$ ⊗

$S \longrightarrow AB$ (divide it into two parts)

$A \longrightarrow aAb \mid ab$

$B \longrightarrow cB \mid c$

Eg- $L = \{a^n b^m c^m d^n \mid m, n \geq 1\}$

~~$S \longrightarrow aSd \mid ad$~~ aAd.

$S \longrightarrow AB$.

$A \longrightarrow aAd \mid ad$

$S \longrightarrow aSd \mid aAd$

$A \longrightarrow bAc \mid bc$

Eg- $L = \{a^n b^n c^m d^m \mid m, n \geq 0\}$.

$S \longrightarrow AB$.

$A \longrightarrow aAb \mid \epsilon$

$B \longrightarrow cBd \mid \epsilon$

Eg- $L = \{a^n b^m c^{m+n} \mid m, n \geq 0\}$

$a^n b^m c^m . c^n$

$S \longrightarrow aSc \mid A$

$A \longrightarrow bAc \mid \epsilon$

Eg- $L = \{a^n b^{m+n} c^m \mid n, m \geq 0\}$

$a^n b^n b^m c^m$

$S \longrightarrow AB$

$A \longrightarrow aAb \mid \epsilon$

$B \longrightarrow bBc \mid \epsilon$

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

$S \rightarrow A/B \mid aSb$

$A \rightarrow aA \mid a$

$B \rightarrow bB \mid b$

Eg. $L = \{n_a(w) = n_b(w) \mid w \in \{a,b\}^*\}$

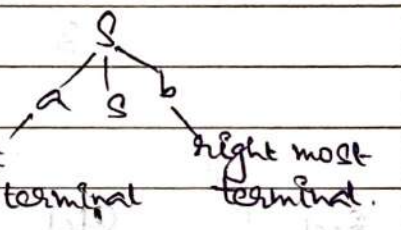
$S \rightarrow aSb \mid bSa$ $\quad S \rightarrow aSbS \mid bSaS \mid \epsilon$

$a^n b^n \cup b^n a^n$.

Eg. $L = \{wcw^R \mid w \in \{a,b\}^* \text{ and } c \text{ is a terminal}\}$.

$S \rightarrow aSa \mid bSb \mid c$

(Odd length palindrome).



Eg. $L = \{ww^R \mid w \in \{a,b\}^*\}$.

$S \rightarrow aSa \mid bSb \mid \epsilon$ (Even length palindrome).

DERIVATION

Two standard derivations are:

- **Left-most Derivation** (In each step of derivation, we replace/use left-most variable).
- **Right-most Derivation** (In each step of derivation, we replace/use rightmost variable).

FG (V, T, S, P)

P{ S → aSbS | bSaS | ∈ }

String "abbaab."

LSF

Each step of left most derivation is called LSF

Step	Left Sentential form	Production
1.	S	S → aSbS
2.	aSbS	S → ∈
3.	abS	S → bSaS
4.	abbSaS	S → ∈
5.	abbaS	S → aSbS
6.	abbaaSbS	S → ∈
7.	abbaabS	S → ∈
8.	abbaab	—

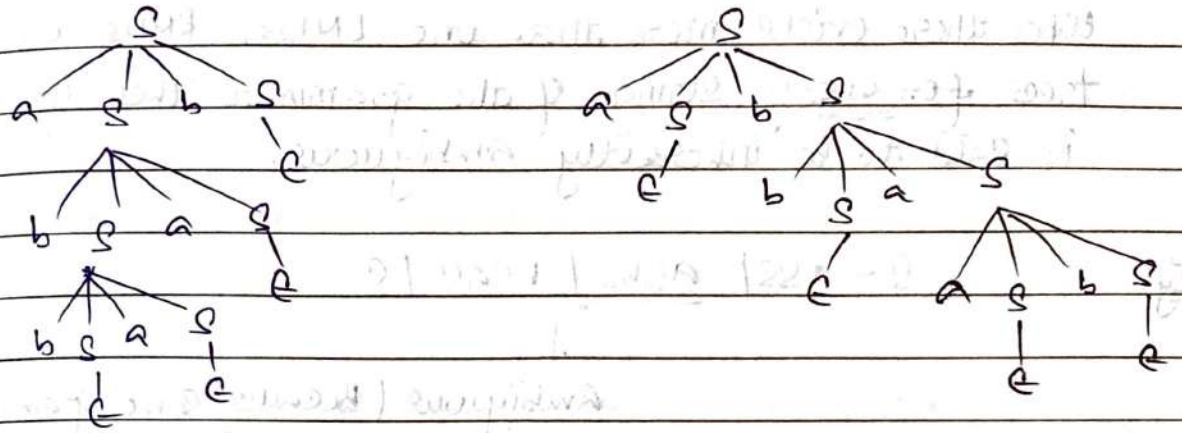
Each step right most derivation is called.

Step	Right Sentential form	Production
1.	S	S → aSbS
2.	aSbS	S → bSaS
3.	aSbbSaS	S → aSbS
4.	aSbbSaaSbS	S → ∈
5.	aSbbSaaSb	S → ∈
6.	aSbbSaab	S → ∈
7.	aSbbaab	S → ∈
8.	abbaab	—

Derivation Tree (parse Tree)

- It is a graphical way to represent derivation in tree form.

- All the variables at non-leaf level and terminals at leaf level.

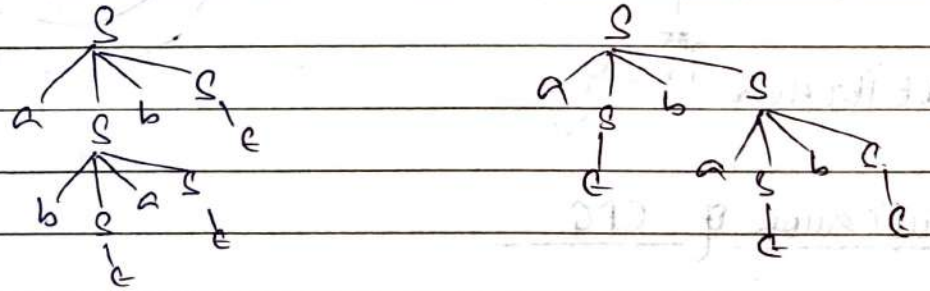


Ambiguous Grammar

When there exists more than one LMDs, RMDs or parse trees for any string of the grammar, then grammar is said to be ambiguous.

(Best way)

Eg. Find the min. length string for which more than one parse tree exists.



abab.

abab.

4-length string

Abab, baba, baab, abba

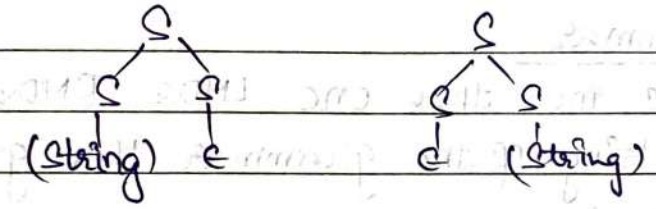
Inherently Ambiguous Grammar

When there exists more than one LMDs, RMDs or parse trees for every string of the grammar then grammar is said to be inherently ambiguous.

Eg- $S \rightarrow SS \mid aSbS \mid bSaS \mid \epsilon$

↓

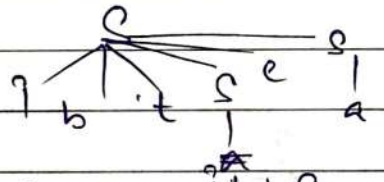
Ambiguous (Because same grammar as previous question)



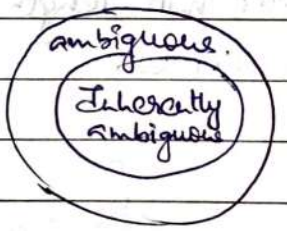
Eg- $S \rightarrow iEtS \mid iEtSeS \mid a$

$E \rightarrow b.$

↳ $S \rightarrow ibtS \mid ibtSeS \mid a$



ibtibtaea



Simplification of CFG

Remove

(i) ε-productions

(ii) Unit "

(iii) Useless "

(i) Remove ε-productions

Identify all variables producing ε.

$A \xrightarrow{*} \epsilon$

(a) Identify all the variables which are producing 'ε'.
(b) Write CFG without 'ε' productions.
(c) Generate new productions by substituting variables 'ε' of step (a) in the set of productions of step (b).

$S \to aSb \mid \epsilon$
$S \to aSb$ (without ε)
$S \to aSb \mid ab$ (substituting S as ε)

Eg. $S \to aAbB \mid b$
$A \to aA \mid \epsilon$
$B \to bB \mid \epsilon$

$S \to aAbB \mid b \mid abB \mid aAb \mid ab$
$A \to aA \mid a$
$B \to bB \mid b$

Substituting one at a time (produce all possible)

Eg. $S \to AaBC \mid aS$
$A \to bB \mid \epsilon$
$B \to aA \mid \epsilon$
$C \to AB \mid d$

$S \to AaBC \mid aS \mid aBC \mid AaC \mid aC$
$A \to bB \mid b$
$B \to aA \mid a$
$C \to AB \mid d \mid B \mid A \mid \epsilon$

C is indirectly producing ε.

$S \to AaBC \mid aS \mid aBC \mid AaC \mid aC \mid AaB \mid aB \mid Aa \mid a$
$A \to bB \mid b$
$B \to aA \mid a$
$C \to AB \mid d \mid B \mid A$

(ii) Remove Unit Productions

$A \longrightarrow B$

(a) Identify all unit productions in CFG.
(b) For each unit production $A \longrightarrow B$ to substitute all the productions of variable 'B' in variable 'A', and remove $A \to B$.

Eg- $S \longrightarrow dSb \mid A$
$A \longrightarrow bAa \mid a$ $(S \to A)$

$S \longrightarrow dSb \mid bAa \mid a$
$A \longrightarrow bAa \mid a$

Eg- $S \longrightarrow aS \mid A$
$A \longrightarrow bA \mid B$
$B \longrightarrow aB \mid b$

$S \longrightarrow aS \mid bA \mid B$
$A \longrightarrow bA \mid B$
$B \longrightarrow aB \mid b.$

$S \longrightarrow aS \mid bA \mid aB \mid b$
$A \longrightarrow bA \mid aB \mid b$
$B \longrightarrow aB \mid b.$

Eg- $S \longrightarrow aSb \mid B$
$A \longrightarrow bSa \mid B \mid b$
$B \longrightarrow aAb \mid A \mid a$

$(S \to B)$
$S \longrightarrow aSb \mid aAb \mid A \mid a \mid$
$A \longrightarrow bSa \mid aAb \mid A \mid a \mid b.$
$B \longrightarrow aAb \mid A \mid a.$

$A \to A$ useless production

$S \longrightarrow aSb \mid aAb \mid a \mid b \mid bSA \mid$
$A \longrightarrow bSa \mid aAb \mid \mid a \mid b$
$B \longrightarrow aAb \mid bSa \mid a \mid b$

productions of A & B are exactly same $\xrightarrow{\text{further simplification}}$ $A = B.$

(iii) Remove useless productions

(a) All the variables must be attached to starting variable directly or indirectly.

$S \rightarrow aAb$ — Variable 'A' is directly attached to 'S' but variable 'B' is indirectly attached (via 'A')
$A \rightarrow aB$
$B \rightarrow a$

(b) All the variables must be terminated directly or indirectly in the same example.

S is terminating via A
A is terminating via B and
B is directly terminating.

→ If any of these two productions fail, then we remove all the productions of that variable.

Eg. $S \rightarrow aS \mid A$
$A \rightarrow bA \mid a$
$\times$ $B \rightarrow bS \mid b$ — B is not attached to the starting variable S, so all the productions of B are removed.

Eg. $S \rightarrow aAS b \mid b$
$\times$ $A \rightarrow aA \mid bBS$) Recursive Dependency
$\times$ $B \rightarrow bBS \mid bA$) (A depends on B and vice-versa)

Remove all productions of A and B

$\boxed{S \rightarrow b}$

Eg- $S \rightarrow aABb \mid b$

$A \rightarrow aAS \mid bBA$ — will not terminate.

$B \rightarrow bBS \mid a$

$C \rightarrow aAb \mid bBa \mid AS \mid bS \mid a \mid C$ — X

No connection (attachment)

$S \rightarrow b$

$B \rightarrow bBS \mid a$ → no attachment with S.

$\therefore S \rightarrow b$

Order for Simplification → (i) Null
(ii) Unit
(iii) Useless.

Eg- $S \rightarrow aAb \mid bS \mid B$

$A \rightarrow aA \mid bBa$

$B \rightarrow bSa \mid \epsilon$

$C \rightarrow bAa \mid \epsilon \mid AS$

Null

$S \rightarrow aAb \mid bS \mid B \mid \epsilon \mid b$

$A \rightarrow aA \mid bBa \mid ba$

$B \rightarrow bSa \mid ba$

$C \rightarrow bAa \mid AS \mid \epsilon$

$S \rightarrow aAb \mid bS \mid B \mid b$

$A \rightarrow aA \mid bBa \mid ba$

$B \rightarrow bSa \mid ba$

$C \rightarrow bAa \mid AS \mid a$

Unit

$S \rightarrow aAb \mid bS \mid b \mid bSa \mid ba$

$A \rightarrow aA \mid bBa \mid ba$

$B \rightarrow bSa \mid ba$

$C \rightarrow bAa \mid aS \mid a$] X

Useless

$S \rightarrow aAb \mid bS \mid bSa \mid ba \mid b$

$A \rightarrow aA \mid bBa \mid ba$

$B \rightarrow bSa \mid ba$.

Normal forms of CFG

(i) Chomskey Normal form (CNF)
(ii) Greibach Normal form (GNF)

First simplify CFG before converting it to any of the normal form.

① Chomskey Normal form (CNF)

VP : $A \to BC$

Or $A \to a$

where $A, B, C \in V$ & $a \in T$.

Eg- $S \to AS \mid a$, $A \to SA \mid b$ $\Rightarrow$ $S \to aSb \mid \epsilon$, $S \to aSb \mid ab$, $S \to ASB \mid AB$, $A \to a$, $B \to b$

$S \to CB \mid AB$, $C \to AS$, $A \to a$, $B \to b$.

Eg $S \to aSb \mid A$, $A \to aS \mid B$, $B \to bS \mid \epsilon$

Null. $S \to aSb \mid A$, $A \to aS \mid B$, $B \to bS \mid \epsilon$

Null $S \to aSb \mid A$, $A \to aS \mid B \mid \epsilon$, $B \to bS$

$S \to aSb \mid A \mid \epsilon$, $A \to aS \mid B$, $B \to bS$.

$S \to aSb \mid ab \mid A$, $A \to aS \mid a \mid B$, $B \to bS \mid b$

(No unit) Useless Unit $S \to aSb \mid ab \mid A$, $A \to aS \mid a \mid bS \mid b$, $B \to bS \mid b$

$S \to aSb \mid ab \mid aS \mid bS \mid a \mid b$, $A \to aS \mid a \mid bS \mid b$, $B \to bS \mid b$

useless.

$S \to aSb \mid ab \mid aS \mid bS \mid a \mid b$

→ already in CNF.

$S \to ASB \mid AS \mid BS \mid a \mid b \mid AB$

$A \to a$

$B \to b$.

CNF → $S \to CB \mid AS \mid BS \mid a \mid b \mid AB$

$C \to AS$

$A \to a$

$B \to b$

Eg- $S \to aSAb \mid bA$

$A \to bA \mid aS \mid B \mid \epsilon$

$B \to bS \mid b$

Null. $S \to aSAb \mid bA \mid aSb \mid b$

$A \to bA \mid aS \mid B \mid b$

$B \to bS \mid b$

Unit $S \to aSAb \mid bA \mid aSb \mid b$

$A \to bA \mid aS \mid b \mid bS$

$B \to bS \mid b$.

Useless $S \to aSAb \mid bA \mid aSb \mid b$

$A \to bA \mid aS \mid b \mid bS$

$S \to aSAb \mid bA \mid aSb \mid b$

$A \to bA \mid aS \mid b \mid bS$

$S \to XSAY \mid YA \mid XSY \mid b$

$A \to YA \mid XS \mid b \mid YS$

$X \to a$

$Y \to b$

S → CAY | YA | CY | b
A → YA | XS | b | YS
X → a
Y → b
C → XS

S → DY | YA | CY | b
A → YA | XS | b | YS
X → a
Y → b
C → XS
D → CA .

② GNF (Greibach Normal Form)

∀P : A → aα
where A ∈ V
a ∈ T
and α ∈ V*

Eg S → aSb | ∈
Simplify context free grammar S → aSb | ab

S → aSB | aB
B → b

Eg S → aSb | bA
A → aA | bSa | B
B → bA | ∈

① S → aSb | bA
A → aA | bSa | B | ∈
B → bA

② S → aSb | bA | b
A → aA | bSa | B | a
B → bA | b

③ S → aSb | bA | b
A → aA | bSA | a | bA | b
B → bA | b

④ S → aSb | bA | b
A → aA | bSA | bA | a | b

⑤ S → aSB | bA | b
A → aA | bSC | bA | a | b
B → b
C → a

Classmate
Date
Page

Eg- $S \to Sa \mid b$

Eg. $S \to aSb \mid Ab$
$A \to BA \mid b \mid \epsilon$
$B \to bS \mid aAb \mid a$

① $S \to aSb \mid Ab \mid b$
$A \to BA \mid b \mid B$
$B \to bS \mid aAb \mid a \mid ab$

② $S \to aSb \mid Ab \mid b$
$A \to BA \mid b \mid bS \mid aAb \mid a \mid ab$
$B \to bS \mid aAb \mid a \mid ab$

$S \to aSb \mid Ab \mid b$
$A \to bSA \mid aAbA \mid aA \mid abA \mid b \mid bS \mid aAb \mid a \mid ab$
$B \to bS \mid aAb \mid a \mid ab.$

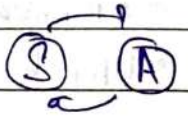
$S \to aSb \mid bSAb \mid aAbAb \mid aAb \mid abAb \mid bb \mid bSb \mid aAbb \mid ab \mid abb \mid b$
$A \to bSA \mid aAbA \mid aA \mid abA \mid b \mid bS \mid aAb \mid a \mid ab$
$B \to bS \mid aAb \mid a \mid ab.$
useless.

Eg. $S \to Sa \mid b$
↳ left recursion.

Left recursion

Direct
$S \to SA \mid b$

Indirect
$S \to AS \mid a$
$A \to SA \mid b.$



To remove indirect, first convert it into direct recursion and remove like direct recursion.

$A \longrightarrow A\alpha \mid \beta$ — $L = \{\beta\alpha^n \mid n \geq 0\}$

$\alpha, \beta \in (V \cup T)^*$

$A \longrightarrow \beta A'$

$A' \longrightarrow \alpha A' \mid \in$ — $L = \{\beta\alpha^n \mid n \geq 0\}$

$A \longrightarrow A\alpha_1 \mid A\alpha_2 \mid \ldots\ldots \mid A\alpha_n \mid \beta_1 \mid \beta_2 \ldots\ldots \mid \beta_m$

$A \longrightarrow \beta_1 A' \mid \beta_2 A' \ldots\ldots \mid \beta_m A'$

$A' \longrightarrow \alpha_1 A' \mid \alpha_2 A' \mid \ldots\ldots \mid \alpha_n A' \mid \in$

Eg. $S \longrightarrow Sa \mid b$

$S \longrightarrow bS'$

$S' \longrightarrow aS' \mid \in$ — remove $\in$

$S \longrightarrow bS' \mid b$

$S' \longrightarrow aS' \mid a$ — Removal of $\in$

Indirect left Recursion

First convert indirect left recursion to direct left recursion.

Eg. $S \longrightarrow As \mid a$

$A \longrightarrow Sa \mid b$ ← production of S in A

$S \longrightarrow As \mid a$

$A \longrightarrow \underbrace{As}_{\alpha} A \mid \underbrace{aA}_{\beta_1} \mid \underbrace{b}_{\beta_2}$ (direct left recursion)

$S \longrightarrow As \mid a$

$A \longrightarrow aAA' \mid bA'$

$A' \longrightarrow SAA' \mid \in$

$\longrightarrow$

$S \longrightarrow As \mid a$

$A \longrightarrow aAA' \mid bA' \mid aA \mid b$

$A' \longrightarrow SAA' \mid SA$

$S \longrightarrow aAA'S \mid bA'S \mid aAS \mid bS \mid a$

$A \longrightarrow aAA' \mid bA' \mid aA \mid b$

$A' \longrightarrow SAA' \mid SA$

(applications of CNF)

$A' \rightarrow aAA'SAA' \mid bA'SAA' \mid aASAA' \mid bSAA' \mid aAA' \mid aAA'SA \mid bA'SA \mid aASA \mid bSA \mid aA.$

GNF is used to convert CFG to PDA

PDA (Push-Down Automata)

Extra memory, unlike FA, to compare variables in external stack memory.

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

Q: Set of finite States

q_0: an initial state $q_0 \in Q$

Σ: Set of finite input Symbols

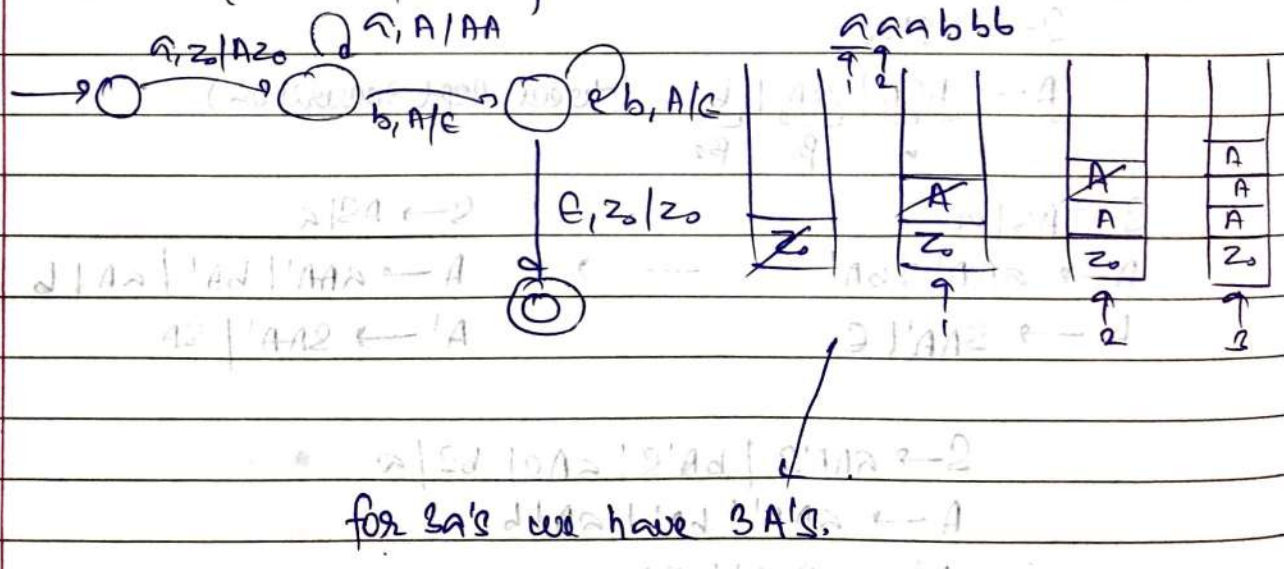
Γ: Set of finite stack Symbol, where z_0 is a special stack symbol which indicates stack is empty.

F: Set of finite final States $F \subseteq Q$

δ: $Q \times (\Sigma \cup \epsilon) \times \Gamma \longrightarrow Q \times \Gamma^*$

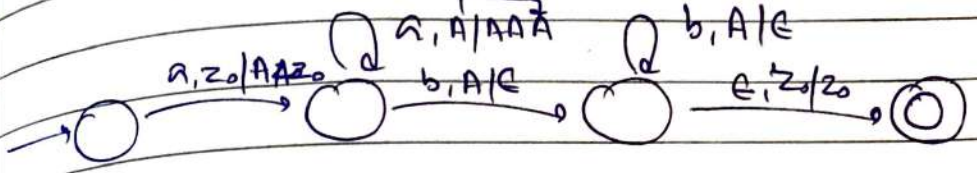
(replaced by)

Eg. $L = \{a^n b^n \mid n \geq 1\}$

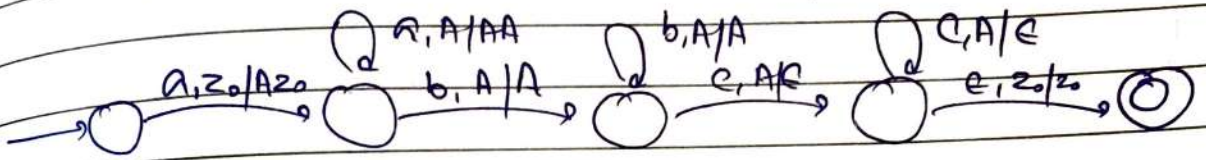


for 3a's we have 3A's.

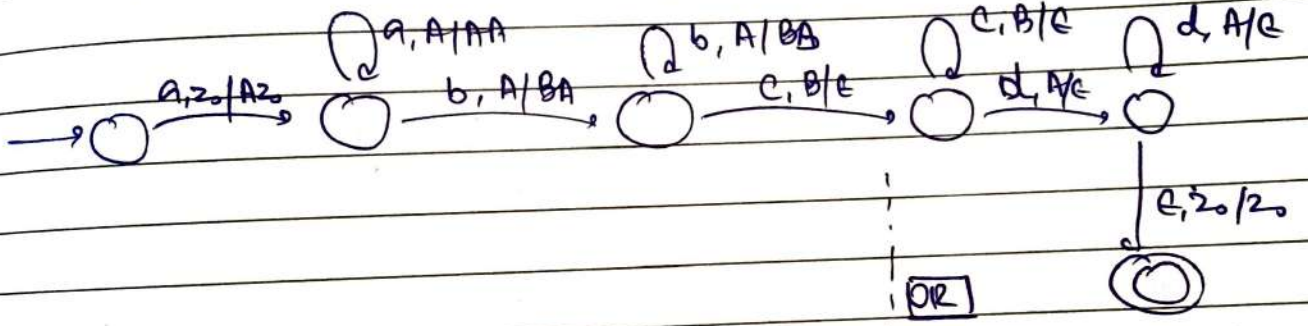
Eg- $L = \{ a^n b^{2n} \mid n \geq 1 \}$ aabbbb.



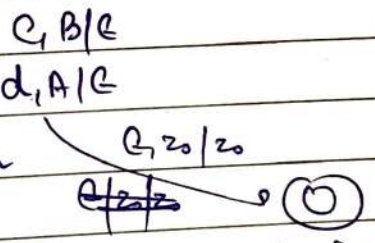
Eg- $L = \{ a^n b^m c^n \mid n, m \geq 1 \}$



Eg- $L = \{ a^n b^m c^m d^n \mid m, n \geq 1 \}$

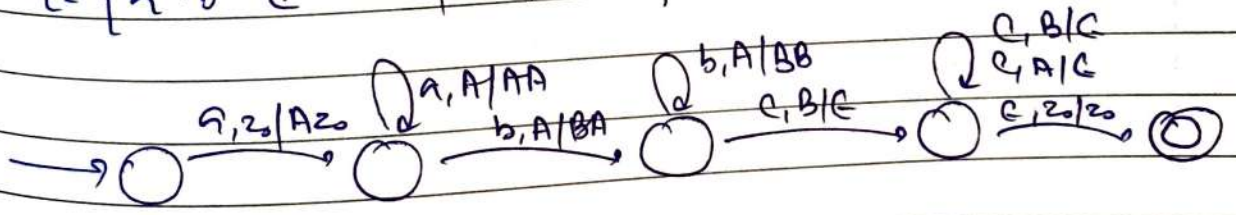


First of all B's will be removed only then A's.

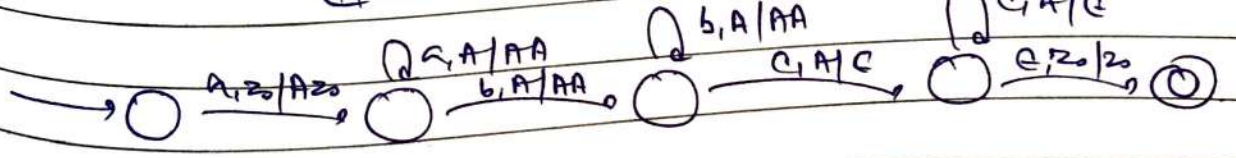


(Order is important, but in this case it is possible).

Eg- $L = \{ a^n b^m c^{n+m} \mid n, m \geq 1 \}$



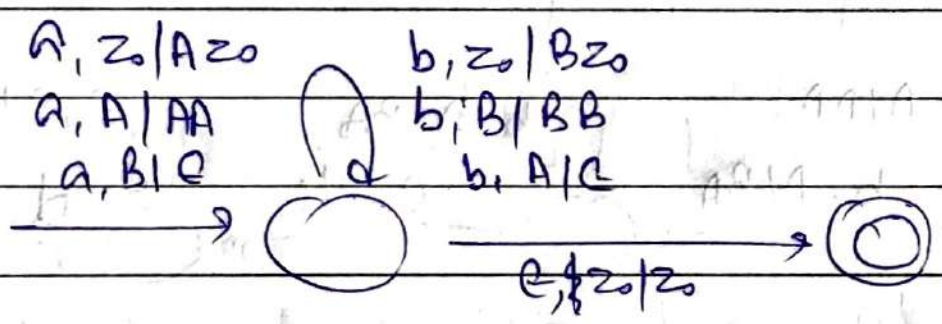
Saves 1 transition.



Eg- $L = \{ n_a(w) = n_b(w) \mid w \in \{a,b\}^* \}$

Top	Input	
~~b~~	~~B~~	$a \to A$
		$b \to B$

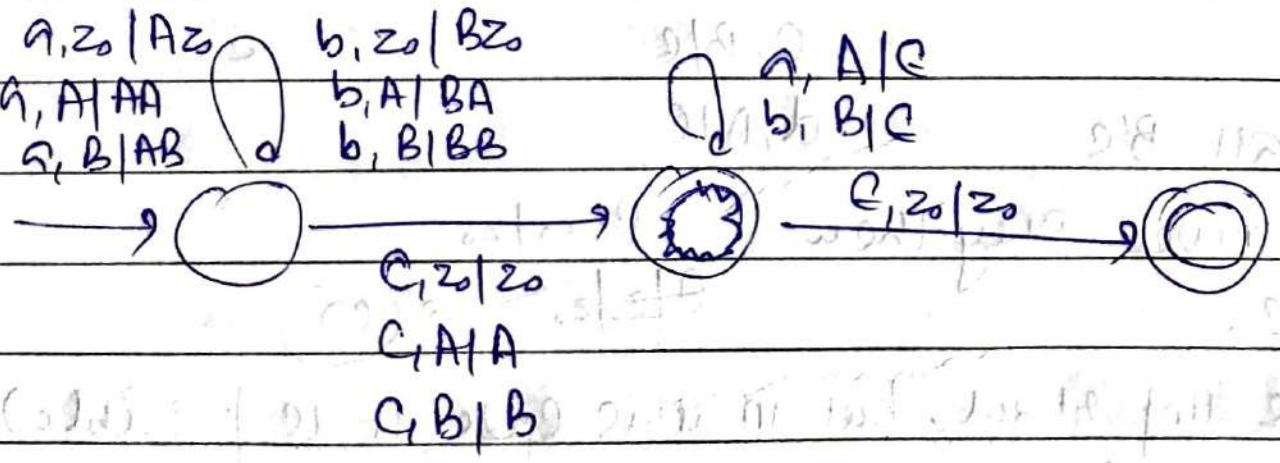
	Top	Input	
Case I:	A	a	AA (push A)
Case II:	A	b	$\in$
Case III:	B	a	$\in$
Case IV:	B	b	BB



Eg- $L = \{ wcw^R \mid w \in \{a,b\}^* \}$

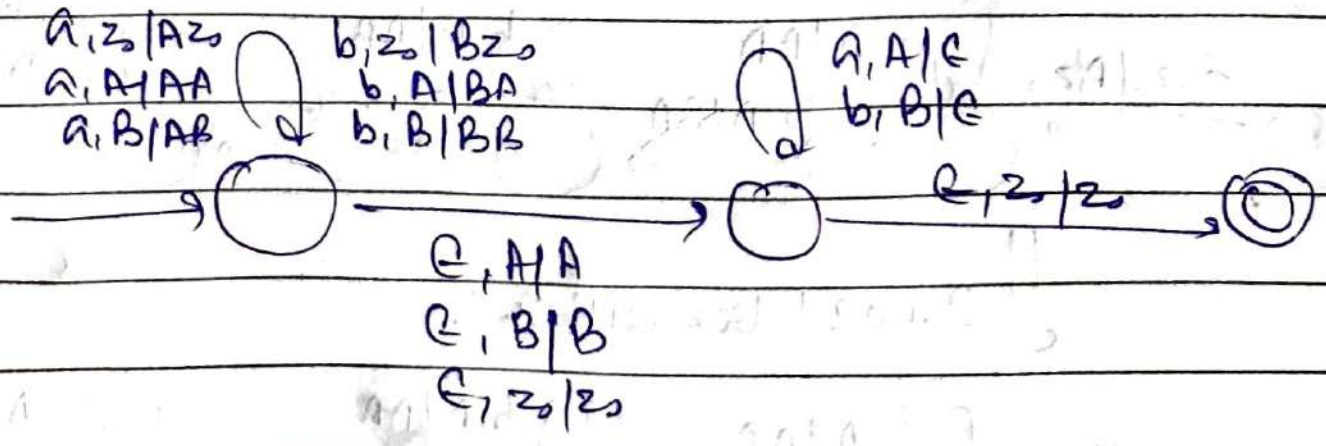
DPDA

C is a terminal.



Eg- $L = \{ ww^R \mid w \in \{a,b\}^* \}$

NPDA



NPDA - no. of attempts are req. to accept a string
DPDA - Definite no. of attempts are there.

3
∧ ∈ bababbbaba
ab ∈ ababbbaba
aba ∈ bbbbaba
abab ∈ bbbaba
ababb ∈ bbaba.

After $\boxed{n/2}$ attempts we can decide whether a string will be accepted by NPDA or not.

PDA

Deterministic PDA
- P(DPDA)

Non-Deterministic PDA.
- P(NPDA)

$\subset$ – Power of NPDA more.

Strings accepted by DPDA will be accepted by NPDA but vice-versa not true. eg- ww^R.

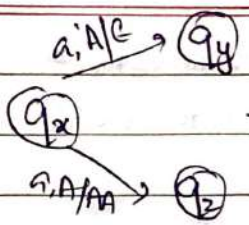
- Language accepted by DPDA is is called DCFL (Deterministic Context free lang)
- Language accepted by NPDA is called CFL (Context-free lang.)

$\boxed{DCFL \subset CFL}$
proper subset.

Deterministic PDA:

Two conditions by which we can decide whether its DPDA or NPDA :-

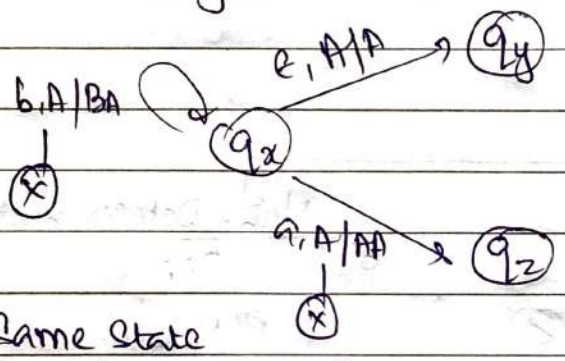
* For every pair of input and stack symbol, there should be at most one transition from every state of PDA.



— Not DPDA.

— NPDA

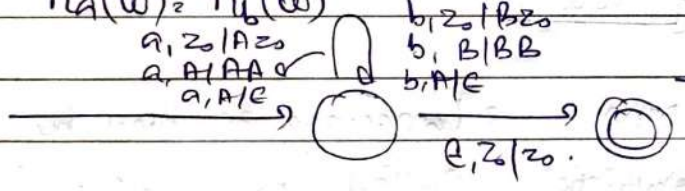
★ Case II: If there exists a null transition from State q_x On Stack Symbol 'A', then from the same State q_x there should not be any other input transition using same Stack Symbol.



— Not possible.

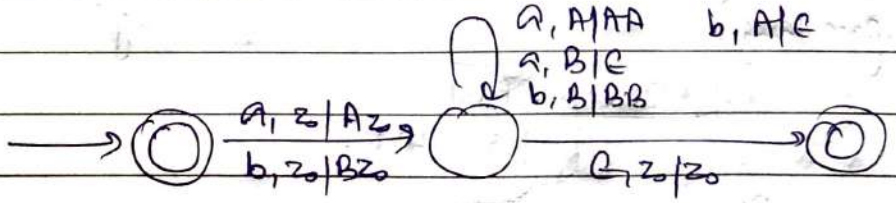
Same State
Symbol, any transition not allowed.

→ $n_a(w) = n_b(w)$



NPDA

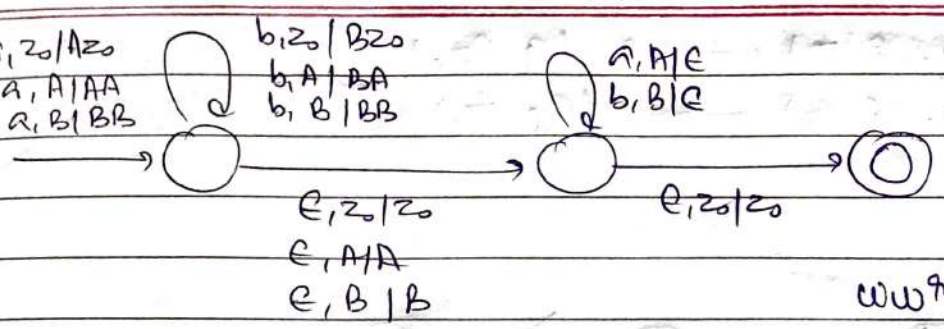
Think!



abababab...

abba...

Not accepted.

Eg- 

NPDA

$WW^R \mid W \in \{a, b\}^+$

NOTE: Every ~~NDF~~ NPDA cannot be converted to equivalent DPDA | ——(above example)

P(NPDA) > P(DPDA)

CFG to PDA

Method I: (All the variables of CFG will act as Stack Symbols)

(i) Convert CFG to GNF form.

(ii) Productions of GNF of CFG

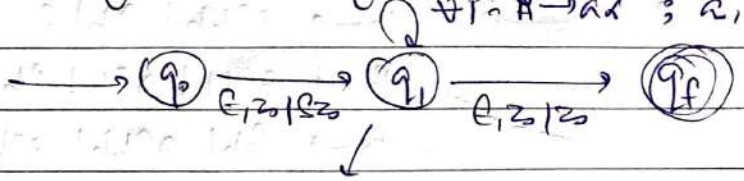
$A \rightarrow a\alpha$ where $A \in V$

$a \in T, \alpha \in V^*$

(iii) We write a transition for every production of GNF

Or $\delta(q_1, a, A) \longrightarrow (q_1, \alpha)$ (replaced)

(iii) Structure of PDA is defined by 3 States.



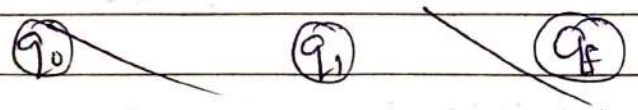
Self-loop (no. of productions in GNF form)

Rest all the transitions are self-looped on q_1.

Initial: $\delta(q_0, \epsilon, z_0) \longrightarrow (q_1, Sz_0)$

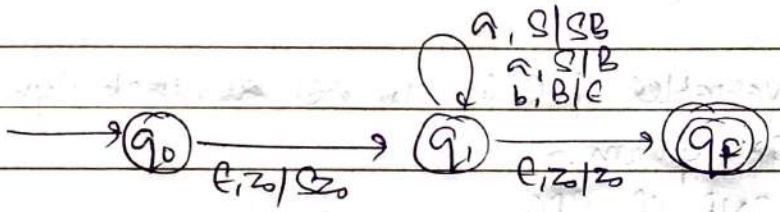
final: $\delta(q_1, \epsilon, z_0) \longrightarrow (q_f, z_0)$

$\Rightarrow$ $\delta(q_0, \epsilon, z_0) \rightarrow$
$\delta(q_1, \epsilon, z_0) \rightarrow$

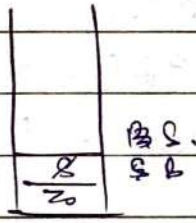


Eg- $S \rightarrow aSb \mid \epsilon$

GNF. $\begin{cases} S \rightarrow aSB \mid aB \\ B \rightarrow b \end{cases}$



aaaabbbb.



Eg- $S \rightarrow aSb \mid A$
$A \rightarrow aS \mid B$
$B \rightarrow Sb \mid \epsilon$

$\Rightarrow$ $S \rightarrow aS$.

$B \rightarrow aSbb \mid Ab \mid \epsilon$
$B \rightarrow aSbb \mid aSb \mid Bb \mid \epsilon$.
$B \rightarrow Bb \mid aSbb \mid aSb \mid \epsilon$

① Null

$S \rightarrow aSb \mid A$
$A \rightarrow aS \mid B \mid \epsilon$
$B \rightarrow Sb$

$S \rightarrow aSb \mid \epsilon \mid A$
$A \rightarrow aS \mid B$
$B \rightarrow Sb$.

$S \rightarrow aSb \mid A \mid ab$
$A \rightarrow aS \mid B \mid a$
$B \rightarrow Sb \mid b$

eg

$S \to aSb \mid A$
$A \to aS \mid B$
$B \to Sb \mid \epsilon$

Check for Recursion →

$S \to aSb \mid A$
$A \to aS \mid B$
$B \to aSbb \mid Ab \mid \epsilon$
$B \to aSbb \mid aSb \mid Bb \mid \epsilon$

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

↓ Remove Null

$S \to aSb \mid A$
$A \to aS \mid B$
$B \to aSbbB' \mid aSbB' \mid B' \mid aSbb \mid aSb \mid \epsilon$
$B' \to bB' \mid b$

$S \to aSb \mid A \mid \epsilon$
$A \to aS \mid B \mid \epsilon$
$B \to aSbbB' \mid aSbB' \mid B' \mid aSbb \mid aSb$
$B' \to bB' \mid b$

$S \to aSb \mid A \mid ab$
$A \to aS \mid B \mid a$
$B \to aSbbB' \mid aSbB' \mid B' \mid aSbb \mid aSb \mid abbB' \mid abB' \mid abb \mid ab$
$B' \to bB' \mid b.$

↓ Unit production.

$S \to aSb \mid A \mid ab$
$A \to aS \mid B \mid a$
$B \to aSbbB' \mid aSbB' \mid B' \mid aSbb \mid aSb \mid abbB' \mid abB' \mid abb \mid ab \mid bB' \mid b.$
$B' \to bB' \mid b.$

↓

$S \to aSb \mid A \mid ab$
$A \to aS \mid a \mid aSbbB' \mid aSbB' \mid aSbb \mid aSb \mid abbB' \mid abB' \mid abb \mid ab \mid bB' \mid b$
$B \to aSbbB' \mid aSbB' \mid aSbb \mid aSb \mid abbB' \mid abB' \mid abb \mid ab \mid bB' \mid b$
$B' \to bB' \mid b.$

$S \rightarrow aSb \mid ab \mid aS \mid a \mid aSbbB' \mid aSbB' \mid aSbb \mid abbB' \mid abB' \mid abb \mid ab \mid bB' \mid b$

Useless. $\times \begin{cases} A \rightarrow aS \ldots - \\ B \rightarrow \\ B' \rightarrow bB' \mid b. \end{cases}$

$S \rightarrow aSb \mid aS \mid a \mid aSbbB' \mid aSbB' \mid aSbb \mid abbB' \mid abB' \mid abb \mid ab \mid bB' \mid b.$

CFG $B' \rightarrow bB' \mid b.$

↓

GNF

$S \rightarrow aSB \mid aS \mid a \mid aSBBB' \mid aSBB' \mid aSBB \mid aBBB' \mid aBB' \mid aBB \mid aB \mid bB' \mid b$

$B' \rightarrow bB' \mid b$

$B \rightarrow b$

19.02.18

CFG to PDA

Method II

(i) The structure of PDA will have 3 states $\{q_0, q_1, q_f\}$.

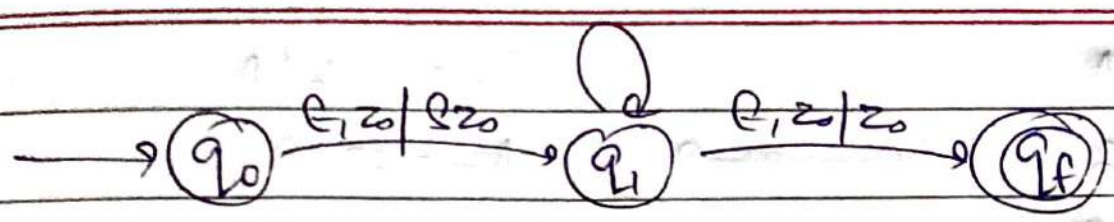
(ii) Initial transition is fixed, i.e,

$$\delta(q_0, \epsilon, Z_0) \longrightarrow (q_1, SZ_0)$$

(iii) Final transition is also fixed.

$$\delta(q_1, \epsilon, Z_0) \longrightarrow (q_f, \cancel{\delta} Z_0)$$

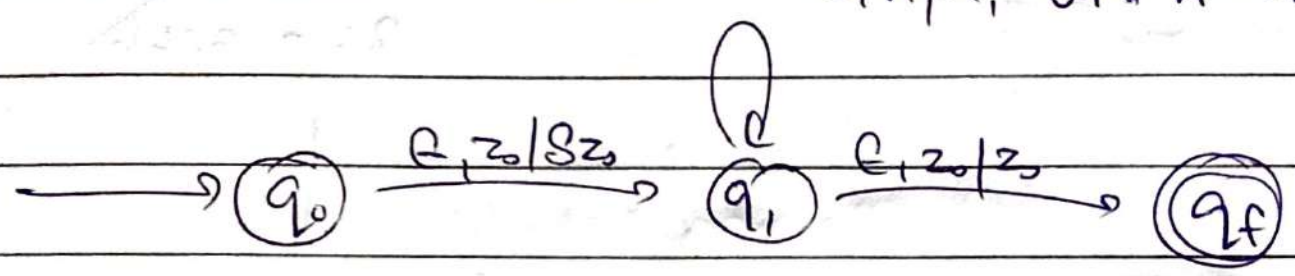
And remaining transitions are self-looped on state q_1.



For every production $P: A \to \alpha$ where $A \in V$ & $\alpha \in (V \cup T)^*$ we write a 'ϵ'-transition as

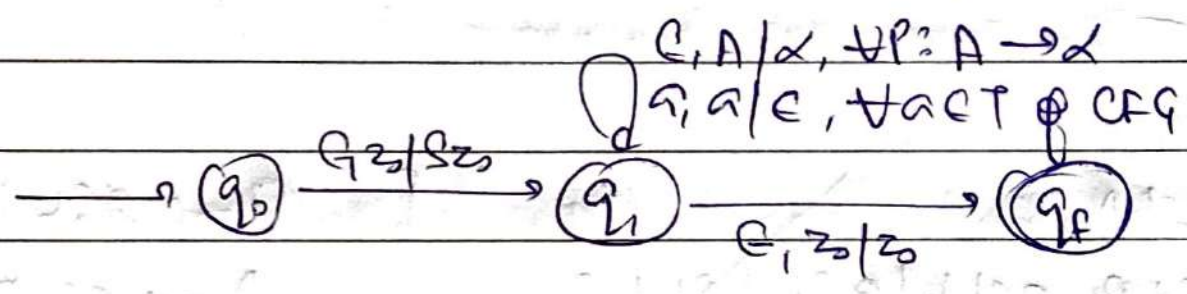
$$\delta(q, \epsilon, A) \longrightarrow (q, \alpha)$$

$\epsilon, A/\alpha$, $\forall P: A \to \alpha$

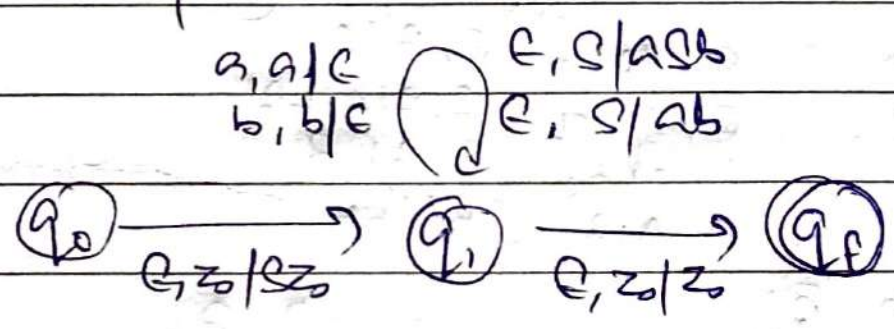


For every terminal $a \in T$ of CFG, we write a transition

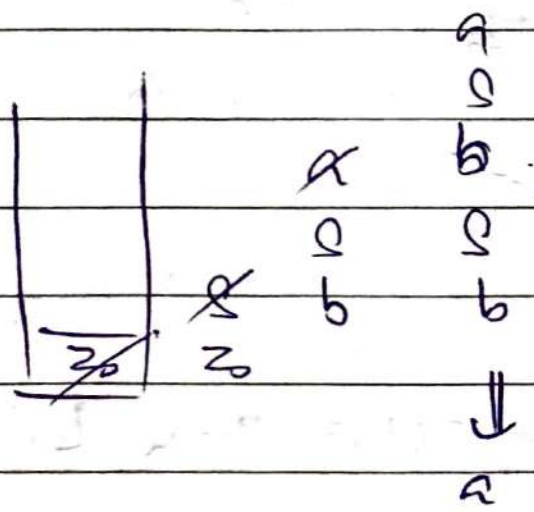
$$\delta(q_1, a, a) \longrightarrow (q_1, \epsilon)$$



$S \to aSb \mid ab$



aaaabbbb.

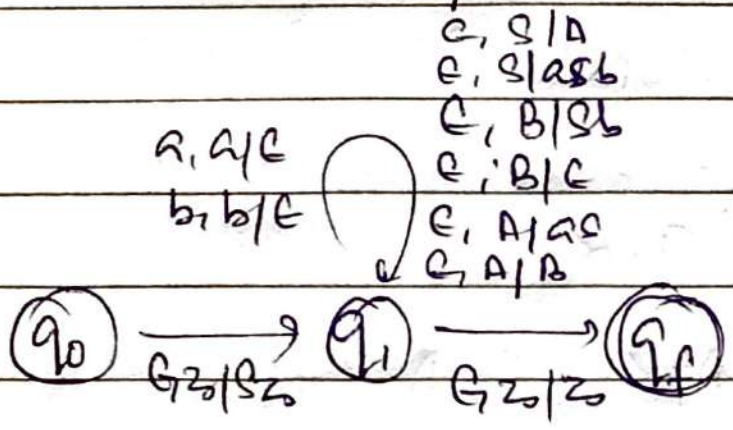


Eg-

$S \to aSb \mid A$

$A \to aS \mid B$

$B \to Sb \mid \epsilon$



$S \to aSb \mid A$

$A \to aS \mid B$

$B \to aSbb \mid Ab \mid \epsilon$

$S \to aSb \mid A$

$A \to aS \mid aSbb \mid Ab \mid$

$B \to aSbb$

$S \to aSb \mid A$

$A \to aS \mid B$.

$B \to aSbb \mid aSb \mid Bb \mid \epsilon$

Remove left Recursion.

$S \to aSb \mid A$

$A \to aS \mid B$.

$B \to aSbb \mid Ab \mid \epsilon$

$S \to aSb \mid A$

$A \to aS \mid B \mid aSbb \mid aSb \mid Bb \mid \epsilon$

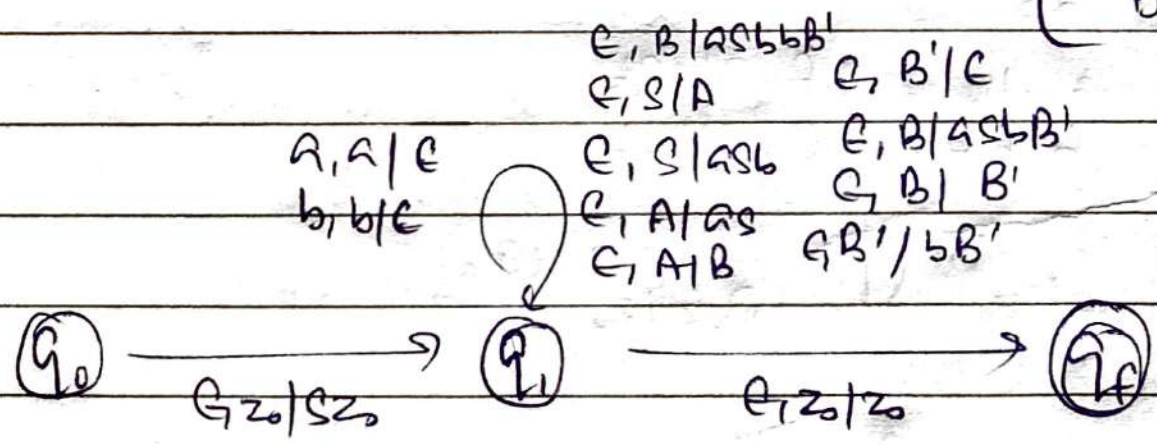
$B \to a$

$S \to aSb \mid A$

$A \to aS \mid B$.

$B \to aSbbB' \mid aSbB' \mid \epsilon$

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



CLOSURE PROPERTIES OF CFL

(i) **Union:** Let L_1 and L_2 are two CFLs then $L_1 \cup L_2 = L_3$ is also CFL

Proof: If L_1 and L_2 are CFL then there exists CFG that generates L_1 and L_2 completely. Let $G_1(V_1, T_1, S_1, P_1)$ is a CFL for L_1 and $G_2(V_2, T_2, S_2, P_2)$ is a CFG for L_2.

A new G_3 is defined by $G_1 \cup G_2$ as $G_3(V_3, T_3, S, P_3)$

where $V_3 = V_1 \cup V_2$

$T_3 = T_1 \cup T_2$

S = Starting variable of G_2

$P_3 = P_1 \cup P_2 \cup \{ S \rightarrow S_1 | S_2 \}$

The lang. generated by G_3 is $L_3 = L_1 \cup L_2$.

(ii) Compliment: CFLs are not closed under compliment.

L_1 = CFL $\quad$ $L_2 = \overline{L_1}$ not CFL

(iii) Intersection: L_1 and L_2 are two CFLs

$L_1 \cap L_2 = L_3$ (X) Not CFL.

CFLs are not closed under intersection.

(iv) Set Difference: If L_1 and L_2 are CFLs then $L_1 - L_2 = L_3$ is not CFL.

(v) Kleen's Closure: If L_1 is a CFL then $L_1^* = L_2$ is also CFL.

(vi) Concates: If L_1 and L_2 are CFLs then $L_1 . L_2 = L_3$ is also CFL.

(vii) Reversal: If L_1 is a CFL then

$L_1^R = L_2$ is also a CFL.

Date
Page

0

* **Kleen's Closure** – $L_1^* = \{\epsilon, L_1, L_1 \cdot L_1, L_1 \cdot L_1 \cdot L_1 \ldots\}$

Proof: If L_1 is a CFL then there exists CFG for L_1

$CFG(L_1) = G_1(V_1, T_1, S_1, P_1)$ ~ $S \to SL_1 | \epsilon$

Now define a new CFG $= G_2(V_2, T_2, S, P_2)$ with the help of G_1

where $V_2 = V_1 \cup \{S\}$

$T_2 = T_1$

S_2 = Starting variable $S \in V_2$

$P_2 = P_1 \cup \{S \to S.S_1 \mid \epsilon\}$

or

$S \to S_1.S$

Grammar G_2 generates language $L_2 = L_1^*$

* **Concate** – let L_1 and L_2 are two CFLs then $L_1 \cdot L_2 = L_3$ is also a CFL.

L_1 and L_2 are CFLs then there exists CFGs for them to generate them completely.

$CFL(L_1) = G_1(V_1, T_1, S_1, P_1)$

$CFL(L_2) = G_2(V_2, T_2, S_2, P_2)$

Now, define a new CFG $G_3(V_3, T_3, S_3, P_3)$ with the help of G_1 and G_2.

where $V_3 = V_1 \cup V_2 \cup \{S\}$

$T_3 = T_1 \cup T_2$

S_3 = Starting variable, $S \in V_3$

$P_3 = P_1 \cup P_2 \cup \{S \to S_1.S_2\}$

G_3 generates $L_3 = L_1 \cdot L_2$

* Reversal: Let L_1 is a CFL then $L_1^R = L_2$ is also CFL.

$CFG(L_1) = G_1(V_1, T_1, S_1, P_1)$

Now define a new CFG = $G_2(V_2, T_2, S_1, P_2)$ with the help of G_1.

Where $V_2 = V_1$

$T_2 = T_1$

S_2 Starting variable $S \in \{V_2\}$

$S \to aSa \mid \epsilon$

$S \to bSa \mid \epsilon$ (reverse)

$P_2 = \{ \forall P_i : A \to \alpha$ of G_1

we write productions of G_2 as $A \to \beta$ where $\boxed{\beta = \alpha^R}$ $\}$

Closure Properties of DCFL:

* Compliment: Let L_1 is DCFL then $\overline{L_1} = L_2$ is also a DCFL. (H.W) पता करना ही है!!

PDA to CFG

PDA $(Q, q_0, \Sigma, \Gamma, \delta, F)$ $Z_0 \in \Gamma$ initial stack Symbol.

CFG (V, T, S, P)

where $V = [q_i \times q_j]$ where $q_i, q_j \in Q$ & $X \in \Gamma$

↙ two states and a stack symbol. (for defining a variable)

Case I: $\delta(q_x, a, A) = (q_y, \epsilon)$ — top symbol in stack re... by ϵ.

$[q_x A q_y] \longrightarrow a \qquad a \in (\Sigma \cup \epsilon)$

Single production corresponding to transition.

Case II: $\delta(q_x, a, A) = (q_y, B_1 B_2 \ldots B_n)$

$[q_x a$ ~~q_i~~$] \longrightarrow a [q_y$ ~~B_1~~ ~~q_{j_1}~~$] [q_{j_1} B_2$ ~~q_{j_2}~~$]$
$[q_j$

$[q_x a \underline{q_i}] \longrightarrow a [q_y B_1 q_{j_1}] [q_{j_1} B_2 q_{j_2}] [q_{j_2} B_2 q_{j_2}] \ldots$
$\ldots [q_{j_{n-1}} B_n \underline{q_{j_n}}]$
q_i

where $q_i, q_{j_1 \text{ to } n-1} \in Q$.

Suppose: $Q = \{q_0, q_1\}$

$\delta(q_0, a, A) \longrightarrow (q_1, AA)$

$[q_0 A q_i] \longrightarrow a [q_1 A q_{j_1}] [q_{j_1} A q_i]$

$q_i = q_0$

$[q_0 A q_0] \longrightarrow a [q_1 A q_0] [q_0 A q_0]$
$a [q_1 A q_1] [q_1 A q_0]$

$[q_0 A q_1] \rightarrow a[q_1 A q_0][q_0 A q_1] / a[q_1 A q_1][q_1 A q_1]$

Case III: Productions of Starting variable.

$S \rightarrow [q_0 z_0 q_i]$ where $q_i \in Q$

$S \rightarrow [q_0 z_0 q_0] / [q_0 z_0 q_1]$

Suppose: $Q = \{q_0, q_1\}$

$\delta(q_0, a, A) \longrightarrow (q_1, ABC)$

$\Rightarrow S \rightarrow [q_0 z_0 q_0] / [q_0 z_0 q_1] / q_0 A q_1$

Eg- PDA to CFG

PDA $(Q = \{q_0, q_1\}, q_0, \Sigma = \{a, b\}, T = \{z, A, B\}, \delta, F = \{q_1\})$

① $\delta(q_0, a, z_0) = (q_0, Az_0)$ $S \rightarrow [q_0 z_0 q_0] / [q_0 z_0 q_1]$

② $\delta(q_0, a, A) = (q_0, AA)$

③ $\delta(q_0, a, B) = (q_0, \epsilon) \Rightarrow [q_0 B q_0] \rightarrow a$

④ $\delta(q_0, b, z_0) = (q_0, Bz_0)$

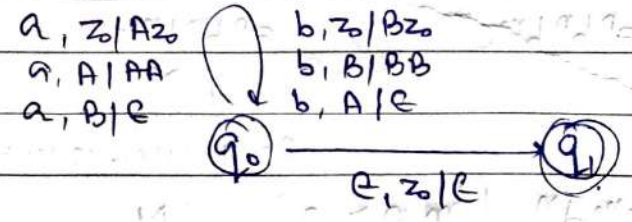
⑤ $\delta(q_0, b, B) = (q_0, BB) \Rightarrow q$

⑥ $\delta(q_0, b, A) = (q_0, \epsilon) \Rightarrow [q_0 A q_0] \rightarrow b$

⑦ $\delta(q_0, \epsilon, z_0) = (q_1, \epsilon) \Rightarrow [q_0 z_0 q_1] \rightarrow \epsilon$

Replacement of State Symbol.

Find CFG equivalent to PDA.



for starting state, $S \longrightarrow [q_0 z_0 q_0] / [q_0 z_0 q_1]$

① $[q_0 z_0 q_i] \longrightarrow a[q_0 A q_{j1}][q_{j2} z_0 q_i]$

$[q_0 z_0 q_0] \longrightarrow a[q_0 A q_0][q_0 z_0 q_0] / a[q_0 A q_1][q_1 z_0 q_0]$

$[q_0 z_0 q_1] \longrightarrow a[q_0 A q_0][q_0 z_0 q_1] / a[q_0 A q_1][q_1 z_0 q_1]$

② $[q_0 A q_i] \longrightarrow a[q_0 A q_{j1}][q_{j1} A q_i]$

④ $[q_0 z_0 q_i] \longrightarrow b[q_0 B q_{j1}][q_{j1} z_0 q_i]$

⑤ $[q_0 B q_i] \longrightarrow b[q_0 B q_{j1}][q_{j1} B q_i]$

Non-Context free Languages

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

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

↓

Context free language

- We can compare two variables at a time.
- We can compare two variables that are linearly dependent. $y = mx + c$.

$L = \{a^n b^m \mid n = 2m\}$ CFL.

$L = \{a^n b^m \mid n = 3m + 100\}$ CFL

$L = \{a^n b^m \mid m = n^2\}$ NCFL

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

$a^n b^n b^m c^m$ $\quad$ $a^n b^m b^n c^m$ ✗

↓

Comparison is not linear type.

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

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

Compare two variables but not at a time (think of stack)

$L = \{ n_a(w) = n_b(w) + n_c(w) \mid w \in \{a,b\}^* \}$ — CFL

(Comparing only two variables)

$L = \{ a^n b^m c^k \mid n = m/k \}$ — NCFL

$L = \{ a^n b^m c^k \mid n = m * k \}$ — NCFL

] no linear dependency.

$L = \{ w a^n b^m w^R \mid n \neq m \ \& \ w \in \{a,b\}^* \}$ — CFL

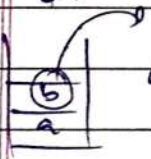
(1st, 2nd — Separate two comparisons.)

$L = \{ ww^R w \mid w \in \{a,b\}^* \}$ — comparison of three variables — NCFL

$L = \{ wcw \mid w \in \{a,b\}^* \}$ → Only reverse comp. possible — NCFL

$L = \{ ww \mid w \in \{a,b\}^* \}$ — NCFL

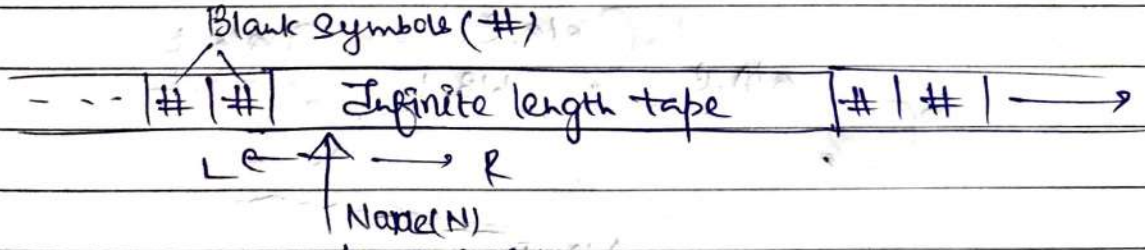
$L = \{ wxw^R \mid w, x \in \{a,b\}^* \}$ — Regular CFL



$L = \{ a^p \mid p \text{ is prime} \}$ — we need to check prime, and we have only stack through which it is not possible to compute p. NCFL

TURING MACHINE

It can do comparison as well as computation.



After every transition, Head of TM can move L, R or not move (N)

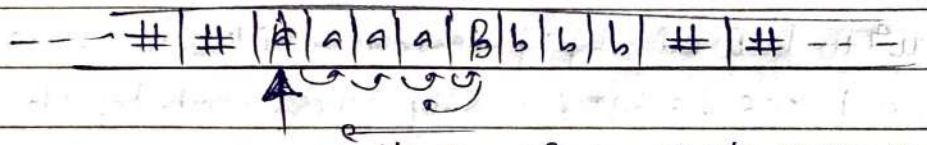
FA / PDA — We can only read from the tape
- Cannot write.
- Can move only forward.

Eg- $L = \{a^n b^n \mid n \geq 1\}$

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

T: State of finite tape symbols.

$\delta: Q \times (\Sigma \cup T) \longrightarrow Q \times (\Sigma \cup T) \times \{L, R, N\}$

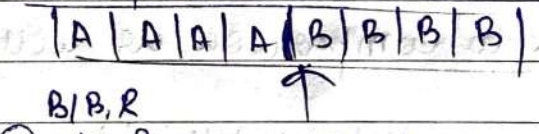


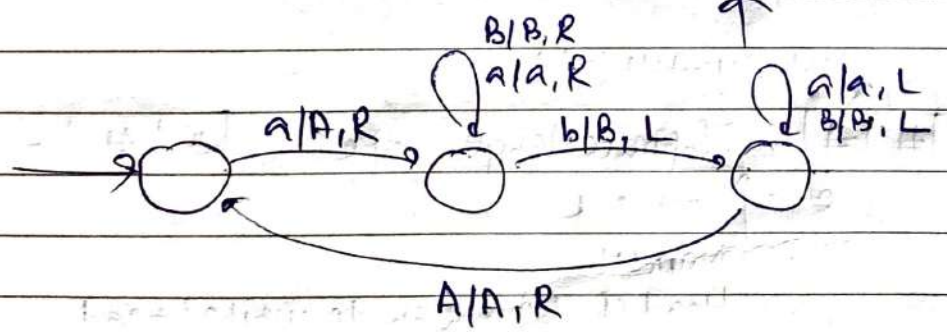
① A a a a B b b b

② A A A a a a B b b b.

after this, move till we get a small b.

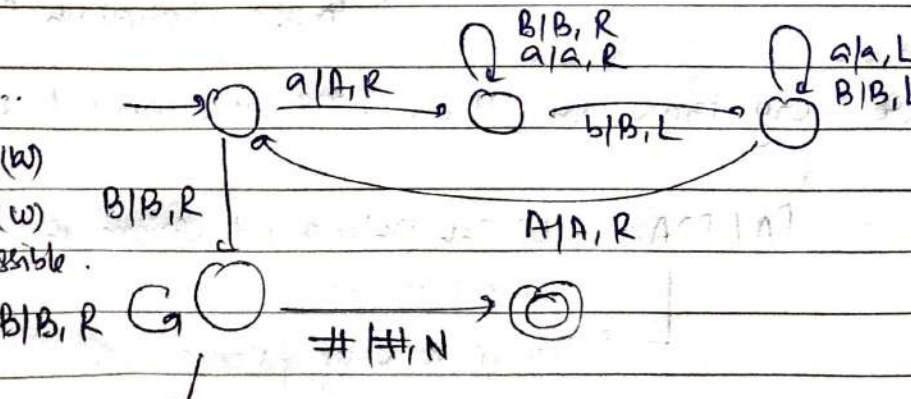
Last position of Head. (First B.)





If aabbbbbb ...

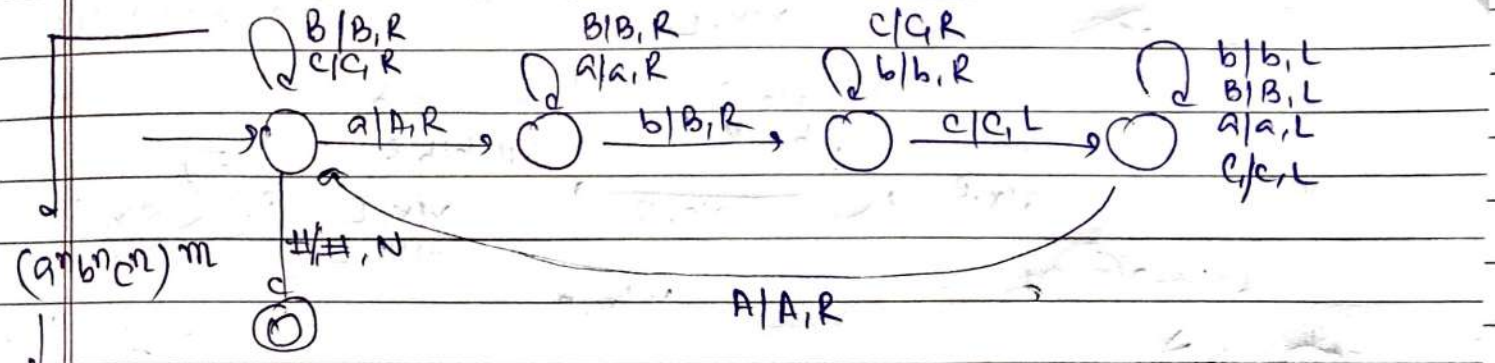
When $n_a(w) = n_b(w)$ Only then possible.



machine will halt here, string not accepted (not final state)

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

- - - # |A|a|a|B|b|b|C|c|c|#| - - -

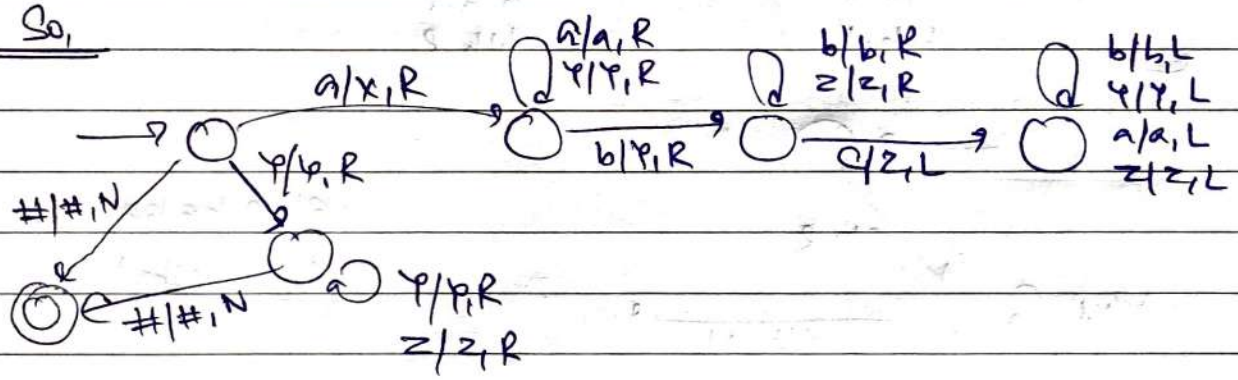


$(a^n b^n c^n)^m$

TM for different lang

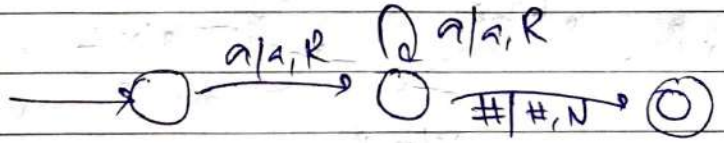
Only Null → #/#,N →

So,

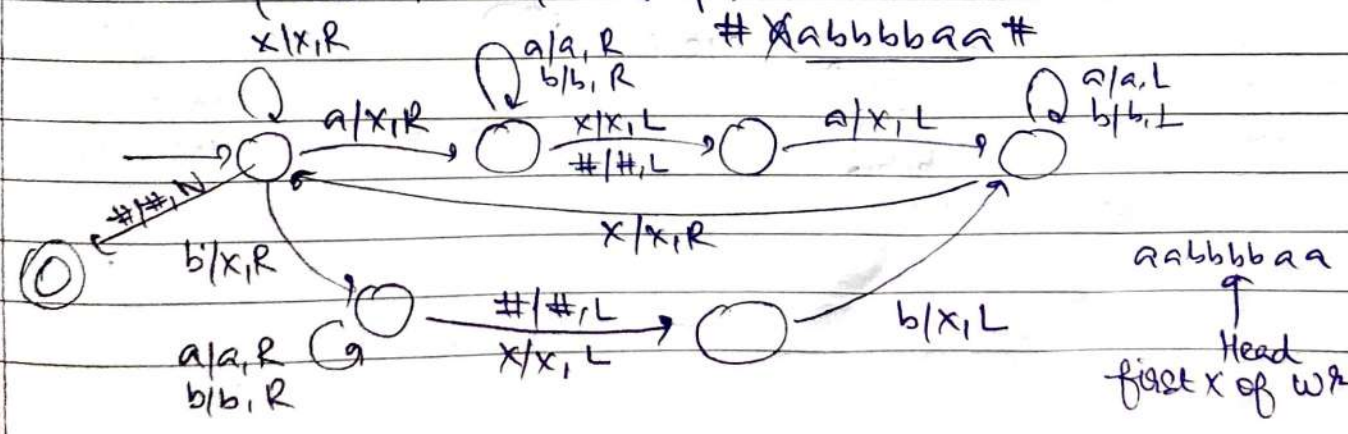


Eg- $L = \{a^n \mid n \geq 1\}$

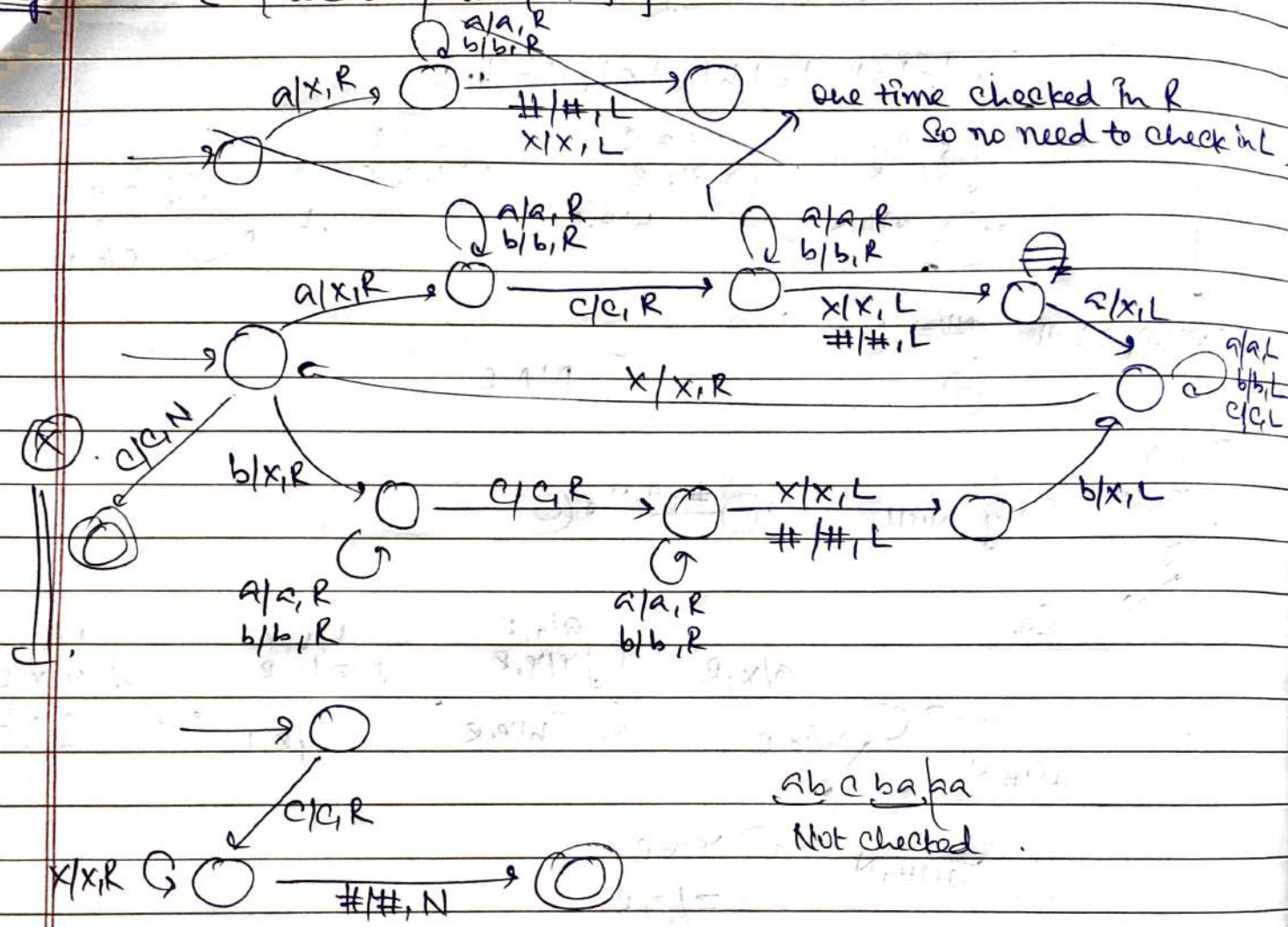
Scanning.



Eg- $L = \{ww^R \mid w \in \{a,b\}^*\}$



Eg- $L = \{wcw^R \mid w \in \{a,b\}^*\}$ c is a terminal.

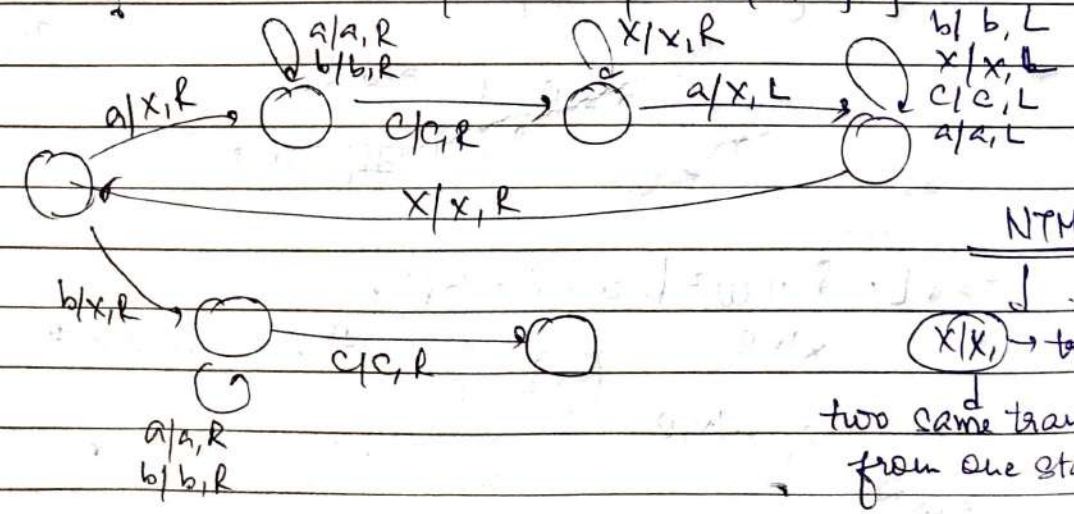


abcbaaa
Not checked.

02.04.18

Eg- wcw = ?

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



NTM

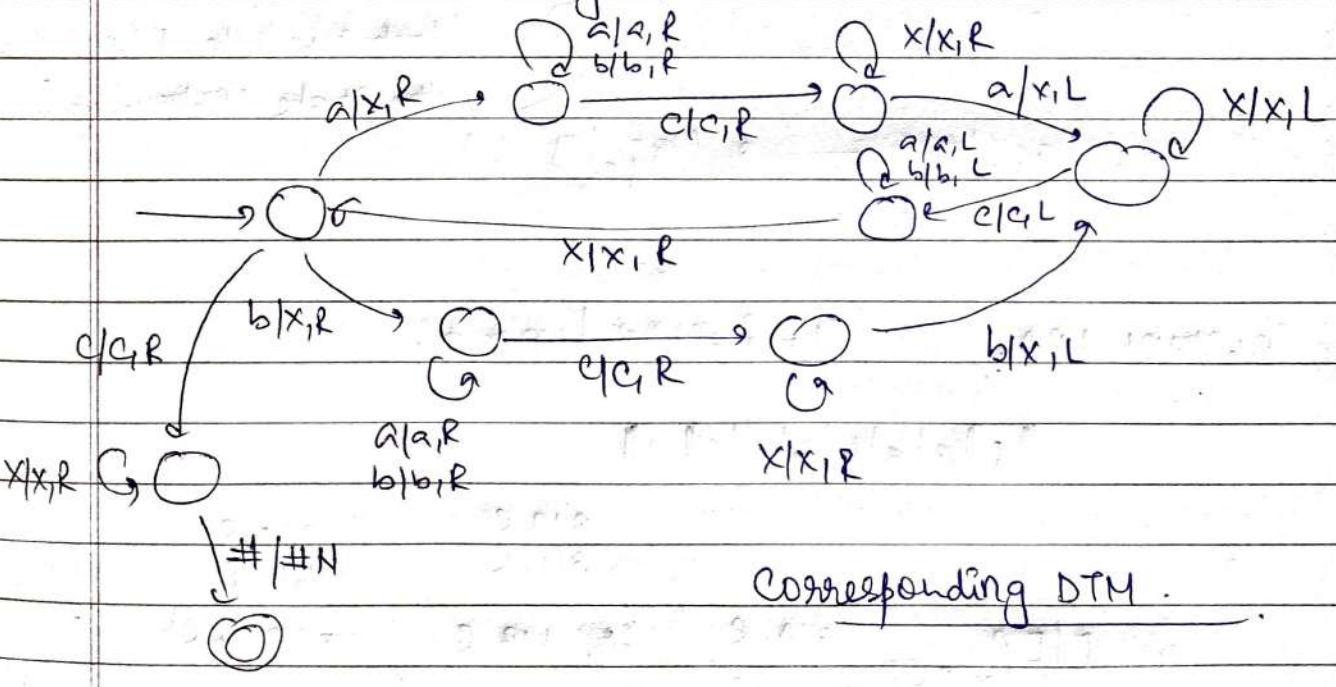
X/X, → transition

two same transition from one state.

TM

Non-deterministic TM	Deterministic TM
$\delta: Q \times (\Sigma \cup T) \rightarrow 2^Q \times (\Sigma \cup T) \times \{L, R, N\}$	Every NTM can be converted to equivalent DTM.
Almost one transition for $\Sigma \cup T$ from every state	Power (NTM) = Power (DTM)

There is no such specific method to convert NTM to DTM, we have to decide using transitions.



Corresponding DTM.

Eg. $L = \{ww \mid w \in \{a,b\}^+\}$

How to find the center??

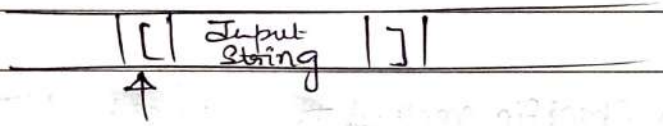
LBA (Linear Bounded Automata)

(Non-deterministic Turing Machine).

length of Tape of TM is finite and bounded by two special tape symbols. First '[' is called left end marker and second ']' is called right end marker.

The input string must be between '[' and ']' on tape of TM.

No transition can reach left of left end marker '[' and right of right end marker ']'.

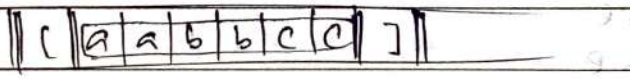


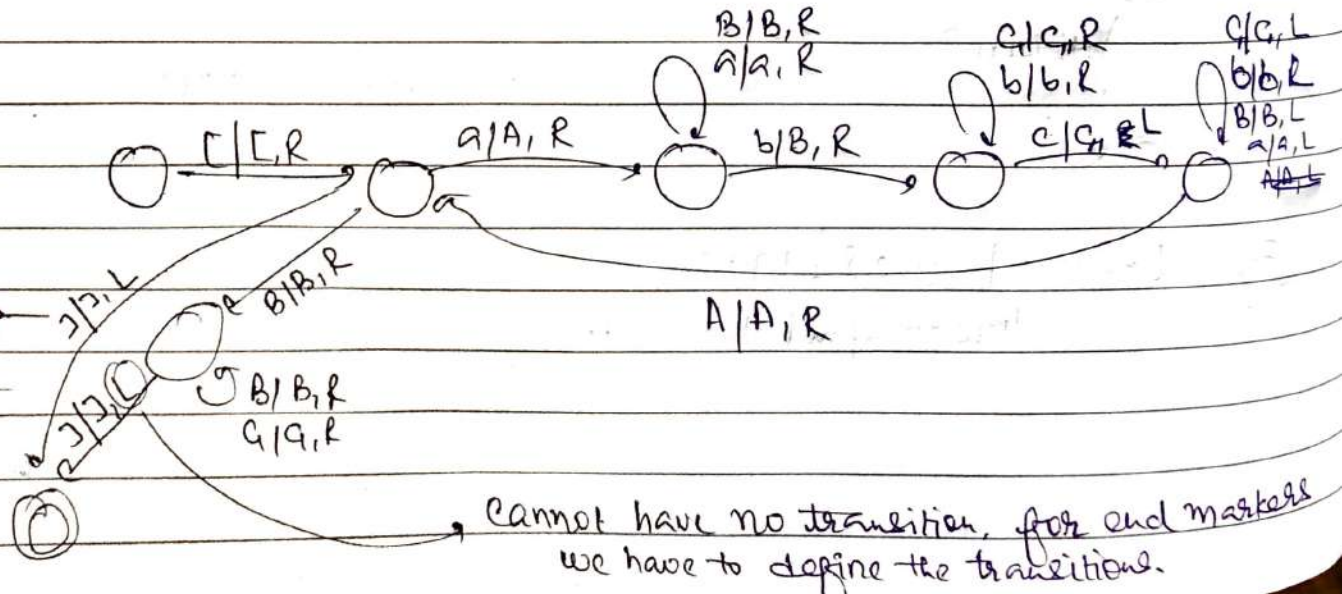
$\delta(q_i, '[') \longrightarrow (q_j, '[', R)$ $q_i \neq q_j$

(Loop cannot be defined on the state containing [)

$\delta(q_i, ']') \longrightarrow (q_j, ']', L)$

Eg- Construct LBA. $L = \{a^n b^n c^n \mid n \geq 0\}$





05.04.18

CONTEXT SENSITIVE LANGUAGES (CSL)

→ ★ Every known defined lang is CSL.

∴ $|\alpha| \leq |\beta|$ cannot write CSG (no null production)

CSG (V, T, S, P)

V: Set of finite variables

T: Set of finite terminals

S: Starting variables

P: Set of finite productions.

$\forall P: \alpha \longrightarrow \beta$

where $|\alpha| \leq |\beta|$ $\quad \alpha, \beta \in (V \cup T)^+$

and α contains atleast one variable.

Imp ★ language generated by CSG is a null free CSL.

Eg- ~~Eg~~ Define CSL for $L = \{a^n b^n c^n \mid n \geq 1\}$

CSG (V, T, S, P)

V: {a, b, c}

T: {c}

S: {a}

P:

CSG Since a, b, c are equal in number, so they should be there in a single production.

$S \longrightarrow aSBC \mid abc$

$CB \longrightarrow BC$

$bB \longrightarrow bb$

$aaaSBcBcBc$

$CB \longrightarrow Bc$

$aaaSBBCBcc$

$aaaSBBBccc$

terminate $aaaabcBBBccc$

$aaaabBBBcccc$

$bB \longrightarrow bb$

$aaaabbbbcccc$

Eg- $L = \{a^{n+m} b^m c^{n+m} \mid m, n \geq 1\}$

$a^n a^m b^m c^m c^n$

$S \rightarrow aSc \mid aXc$

$X \rightarrow aXBc \mid abc$

$cB \rightarrow Bc$

$bB \rightarrow bb$

Eg- ① $L = \{ww \mid w \in \{a,b\}^*\}$

② $L = \{ww \mid w \in \{a,b\}^+\}$

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

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

$S \rightarrow XY \mid abcd$

$X \rightarrow aXBc$

Unrestricted Grammar

$UG(V, T, S, P)$

$\forall P: \alpha \rightarrow \beta$

where $\alpha \in (V \cup T)^+$

and it contains atleast one variable

$\beta \in (V \cup T)^*$

No restriction on the right hand side (independent of each other).

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

$S \rightarrow aSBC \mid \in$

$CB \rightarrow BC$

$bB \rightarrow bb$

$B \rightarrow b$

UG generates two languages —
- Recursive (Turing decidable)
- Recursive Enumerable. (Turing accepter)
  - Only accepts does not decide.

Recursive language: Let L is a recursive language then there exists a TM(L) that behaves as follows

(i) $\forall w \in L$ Then TM(L) halts on state h_a

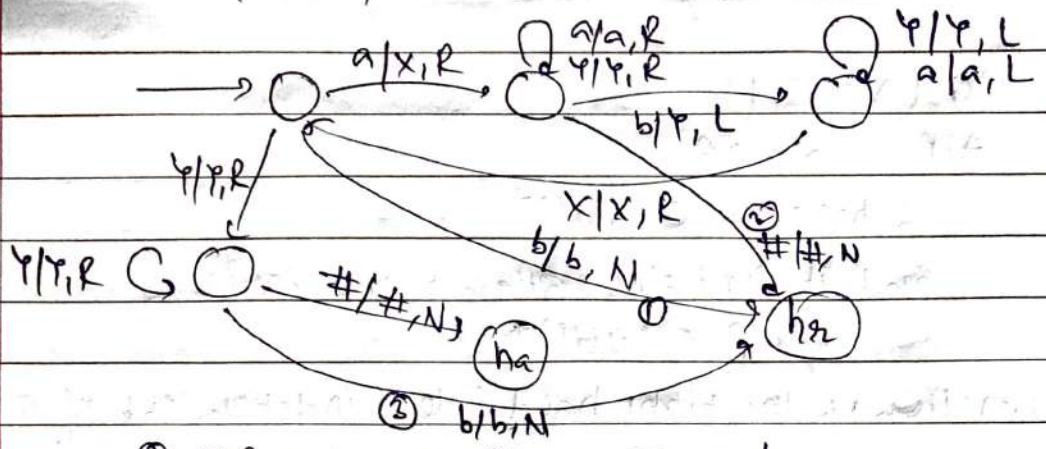
$TM(L) = \{ h_a, h_r \}$

h_a = halt accept

h_r = halt reject.

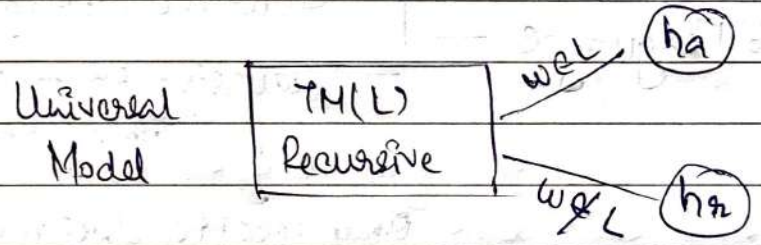
(ii) $\forall w \notin L$ Then TM(L) halts on state h_r

$L = \{a^n b^n \mid n \geq 1\}$



① string cannot start from b.
② more a's than b's.
③ more b's than a's.

Recursive languages are closed under compliment.
(ha, hr - compliments accepted)



06.04.18 **Recursive Enumerable**

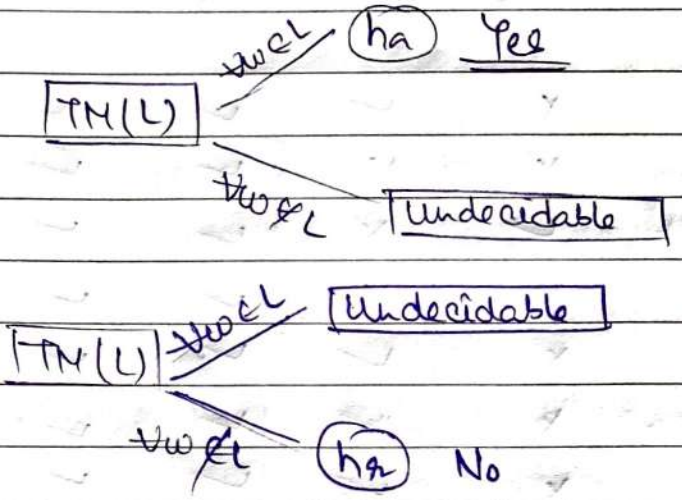
let L is a recursive enumerable lang, then there exists a TM(L) that behaves as other
(i) $\forall w \in L$ then TM(L) halts on ha
(ii) $\forall w \notin L$ then undecidable.

(OR)

Notes → FA and PDA will never loop for every
Why?
Because we move only for the string.
Once string end, no movement, but in LBA, we can move L and R

(b) $\forall w \in L$ then undecidable

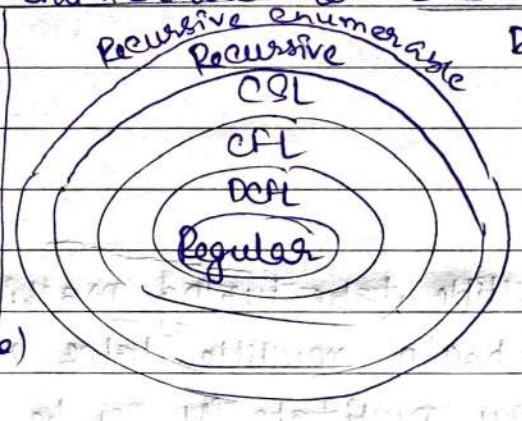
(D) $\forall w \in L$ then TM(L) halts on h_R.



PROPERTIES:

* If L & $\bar{L}$ both are recursive enumerable then L is

Chomsky Hierarchy ⟹



Diagonalization lang. or totally undecidable

→ Recursive enumerable (partially decidable)

→ Recursive (fully decidable) – L and $\bar{L}$ recursive, L recursive.

Regular ⊂ DCFL ⊂ CFL ⊂ CSL ⊂ Recursive ⊂ Recursive Enumerable

Diagonalization languages or totally undecidable } above Recursive enumerable.

Closure Properties of Languages:

	Regular	DCFL	CFL	CSL	Recursive	Recursive Enumerable
Union	✓	X	✓	✓	✓	✓
Interaction	✓	X	X	✓	✓	✓
Compliment	✓	✓	X	✓	✓	X
Set Diff.	✓	X	X	✓	✓	X
Reversal	✓	X	✓	✓	✓	✓
Kleen's Closure	✓	X	✓	✓	✓	✓
Concate	✓	X	✓	✓	✓	✓

Every known lang. is CSL and we have h_a and h_R

→ totally decidable

Multiple tape turing machine:

* It has 'n' multiple tapes each with infinite length.
* Every multitape TM can be converted to equivalent single tape TM

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

↓

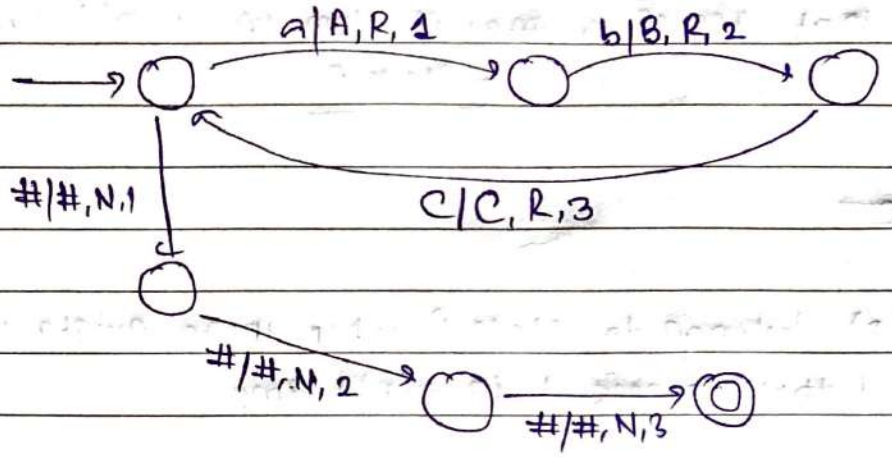
Divide the input and put them each in a tape.

1 |#|a|a|a|a|#|

2 |#|b|b|b|b|#|

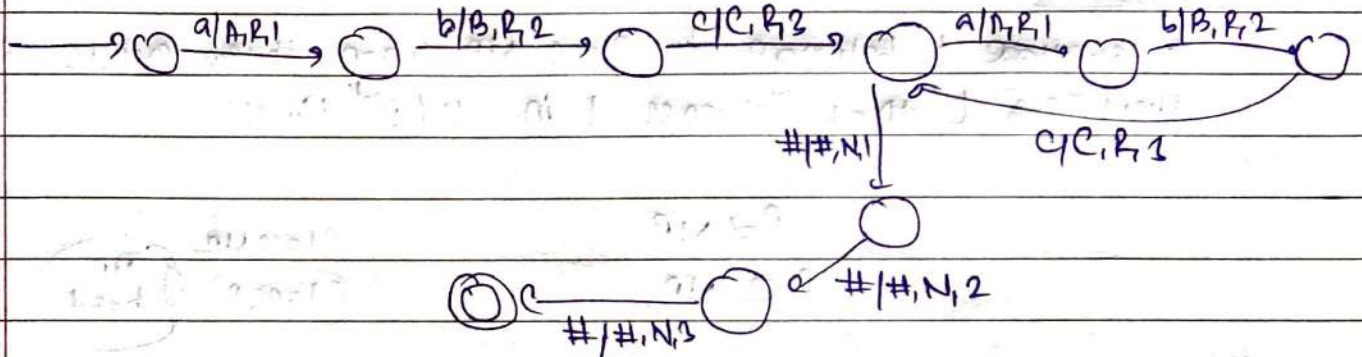
3 |#|c|c|c|c|#|

→ easy as compared to single tape TM.



Every single tape to multi-tape — possible.

$(n \geq 1)$



P & NP Completeness

Class P, Class NP, NP complete and NP hard.

Class P problem: $O(n^k)$ $k > 0$.
(Col. of order)
polynomial ~~type~~ time algo.

NP: Non-deterministic polynomial time algo.
$O(2^{n^k})$ $k > 0$.

→ **DTM** (Class P) $\delta : Q \times \Sigma \cup \Gamma \longrightarrow Q \times \Sigma \cup \Gamma \times \{L, R, N\}$.

NTM: (Class NP) $\delta : Q \times \Sigma \cup \Gamma \longrightarrow 2^Q \times \Sigma \cup \Gamma \times \{L, R, N\}$

M/c
Problem that access a ~~ma~~ problem in $O(n^k)$ time
– Class P

~~Language~~

Class P

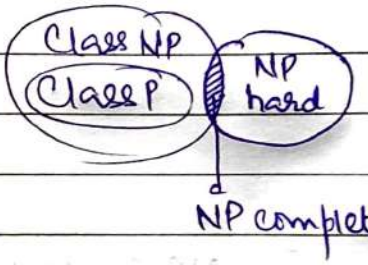
a problem (P) belongs to class P when there exists language L & a DTM for L that process L is $O(n^k)$ time.

Class NP

a language L belongs to class NP when there exists an NTM for L that process L in $O(2^{n^k}$ times)

$P \neq NP$

$P \subset NP$.



To jisme closed nhi hai, vo usme undecidable hogya!