04.01.2023
Tuesday

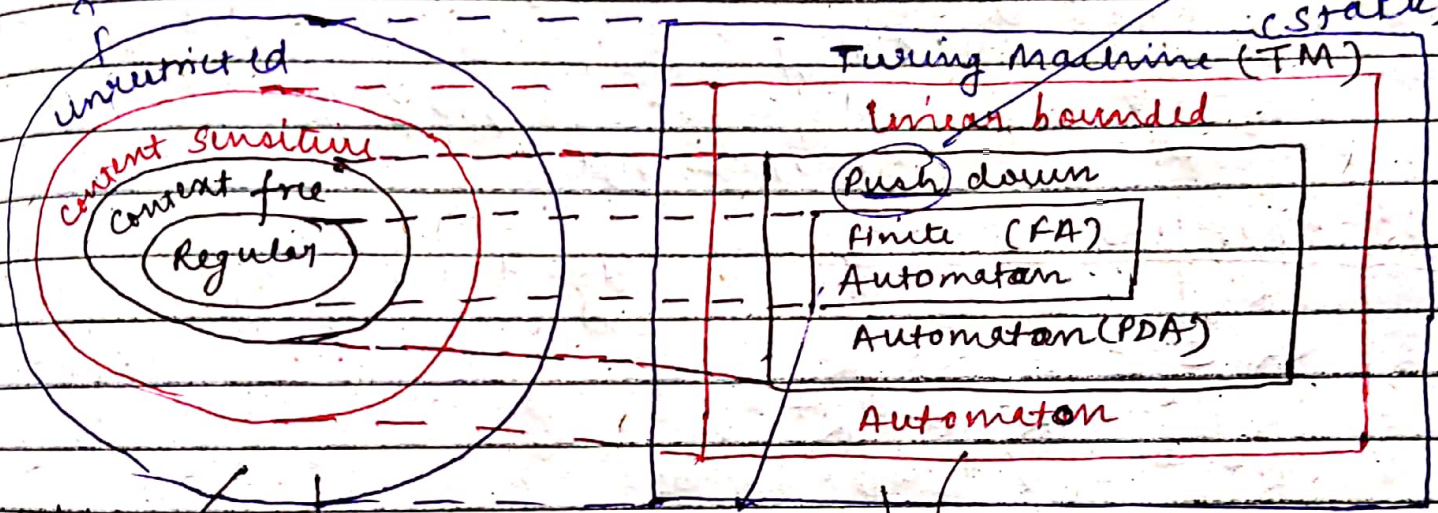
Regular Type-3

Linked Topic of Next Sem :- Compiler Design

THEORY OF COMPUTATION

↓ paperwork of computer

↓ related to computer → perform



Δ Language | Δ Machines

Recursive language | Recursive Enumerable language

No external memory

Exactly it is paper model of computer

Clearly,

Regular ⊂ Context free ⊂ Context sensitive ⊂ Recursive ⊂ Recursive Enumerable

proper subset

Power of Machine = P(Machine)

$P(FA) < P(PDA) < P(LBA) < P(TM)$

exactly maps the computer (can do any language)

(*) In computer, we don't get Nobel prize instead there is Turing prize.

06.01.2023 ~~Thursday~~ Friday

$\epsilon \cdot \epsilon = \epsilon$

OS : It is a manager which does the (internal) work on behalf of the general user

ϕ = Empty set
ϵ = Empty string
$S = \{\epsilon\}$ $|S| = 1$ $|\phi| = 0$
cardinality

Properties of RE

$R - a \cdot \{\} = \phi$
R

① $R \cdot \phi = \phi \cdot R = \phi$
② $R \cdot \epsilon = \epsilon \cdot R = R$
③ $R \cdot R^* = R^* \cdot R = R^+$
④ $(R^*)^* = R^*$
⑤ $\epsilon + RR^* = R^*R + \epsilon$ $\phi = R^*$
⑥ $(P+Q)^* = (P^* + Q^*)^* = (P^* + Q)^* = (P + Q^*)^*$
$= (P^* \cdot Q^*)^*$
$[\neq (P^* Q)^* \neq (P \cdot Q^*)^* \neq P^* Q^*]$
All Q followed by P
⑦ $(P+Q) \cdot R = P \cdot R + Q \cdot R$
⑧ $P \cdot (Q+R) = P \cdot Q + P \cdot R$
⑨ $P^*(Q \cdot P^*)^* = (P^* \cdot Q)^*$ $P^* = (P+Q)^*$
⑩ $\epsilon^* = \phi^*$
$\epsilon^* = \{\epsilon, \epsilon, \epsilon\epsilon, \epsilon\epsilon\epsilon, \ldots\}$
$\phi^* = \{\epsilon, \phi, \phi\phi, \ldots\}$

ϵ ~~P~~ Q PQ
ϵ P ~~Q~~ ~~QP~~

P^* (ϵ P, Q, PQ, PP, QP, QQ, PQ

Finite Automata (FA)

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

Q: set of finite ~~set~~ states

q_0: An initial state $q_0 \in Q$ → belongs

Σ: set of finite input symbols

F: set of finite final states $F \subseteq Q$

δ: Transistion function

δ

- DFA (Deterministic Finite Automata)
- NFA (Non Deterministic FA)

To empty set or anything

powerset

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

Any state Q by having i/p we reach Q

or

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

where $q_i, q_j \in Q$

$a \in \Sigma$

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

Any state Q by having i/p we reach empty set (Non deterministic) or multiple set

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

$q_i \in Q$

$A \subseteq Q$

$a \in \{\Sigma \cup \epsilon\}$

Transistion function of DFA is always total function

↓

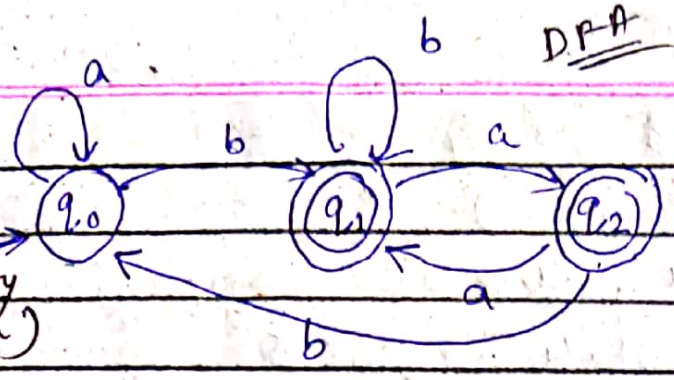
Total function: from every state from each i/p symbol there must be exactly one transistion

Deterministic nature

Initial state = Always single but there are multiple final state

DFA

Example



State with ~~final state~~ → double circle

state ~~edge~~ having edge from (empty edge) namely the state to particular state Then that state

initial state

q_0

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

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

$F = \{q_1, q_2\}$

$\delta_1(q_0, a) \rightarrow q_0$

$\delta_2(q_0, b) \rightarrow q_1$

Accepted any string this machine

a a b b a

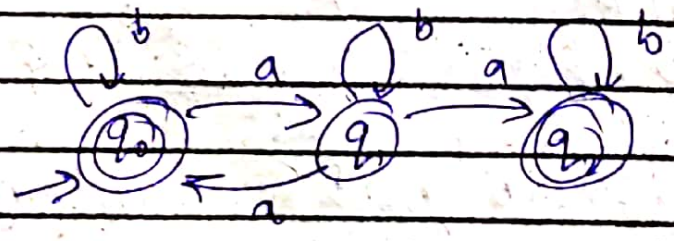
q_0 q_0 q_0 q_1 q_1 q_2

which is final state (double circle)

NOTE (1) Every DFA is also an NFA

(2) Every NFA is not a DFA but we can convert it to equivalent DFA

Any string 'S' is accepted by FA when we reach to the final state



~~Check for all~~ DFA States

(1) ~~All~~ Total f^n

(2) Lead to only 1 state by 1 symbol i/p

NFA

(i) No ~~state~~ transition for $\delta(q_2, a)$

(ii) Multiple transitions for $\delta(q_1, a)$

Non deterministic

a a b b a

q_0 q_1 q_2 q_2 q_2 → No transition

$\delta(q_1, a)$ q_0 q_0 q_0 q_1 ≠ final state

multiple path → check both

→ Not accepted string by machine

Theory of Automata
Peter Linz
~~Theory of~~
John C Martin
Lemon
KLP Mishra

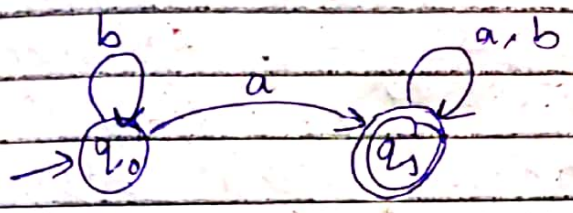
easy to construct NFA than DFA

Q Find the DFA for the following regular languages
① all the strings of 'ab' have atleast 1 'a'

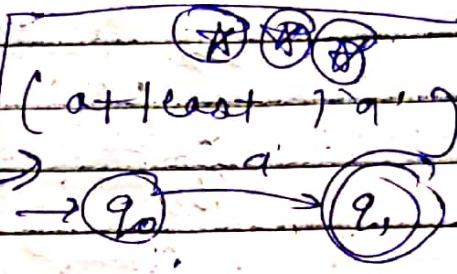
when there is optional then use self loop
(any no. of b's before a)
[The part which can be skipped]

Compulsary Symbol
⇓
must be made a state & must be a state that needs to be visited

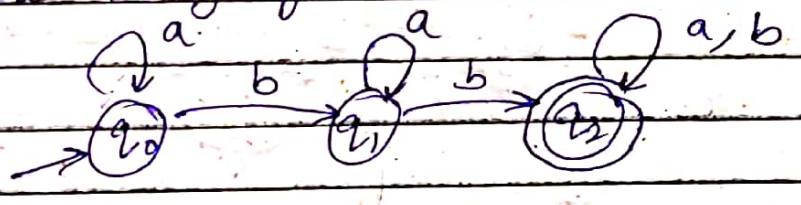
DFA → all the i/p symbols must ~~be~~ have transistion state



(at least 1 'a')

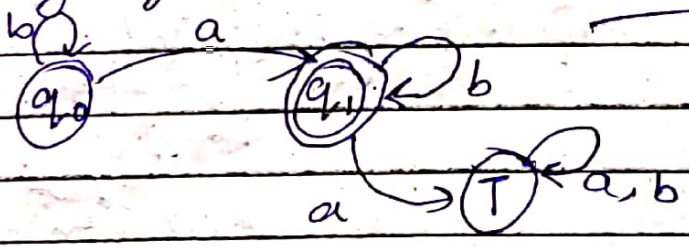


② All the strings of 'ab' have atleast 2 b's



③ All the strings of 'ab' have exactly 1 'a'

Sol:



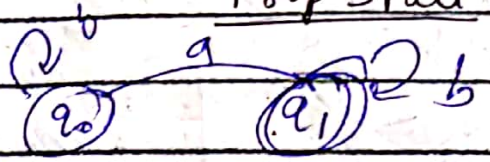
priority to draw this

Trap State : Always have self loop of i/p symbols
Dummy State (No chance of again going back)

Final state पर End है

(4) **NOTE:** DFA can have ~~atleast~~ at most one Trap State.

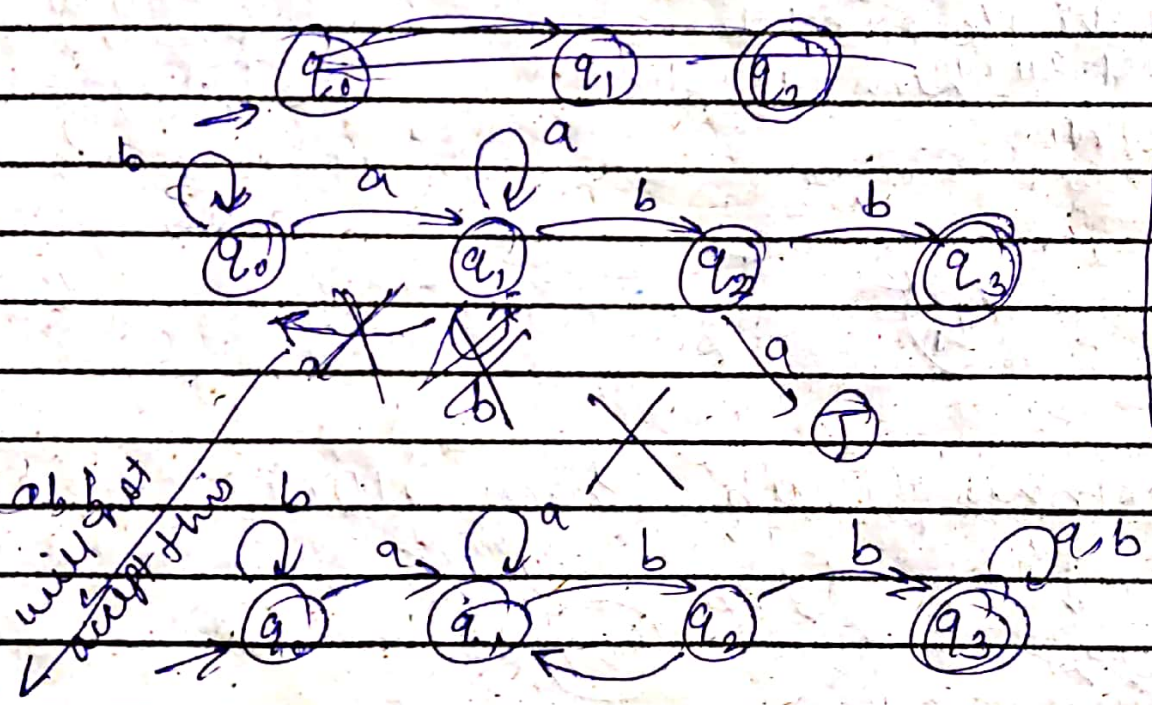
NFA



(4) All the strings 'ab' have substring 'abb'

Sol:

priority



① First preference to self loop

② Then to any other state if

aabbb (Final पर नहीं पहुंचते)

One requirement is fulfilled

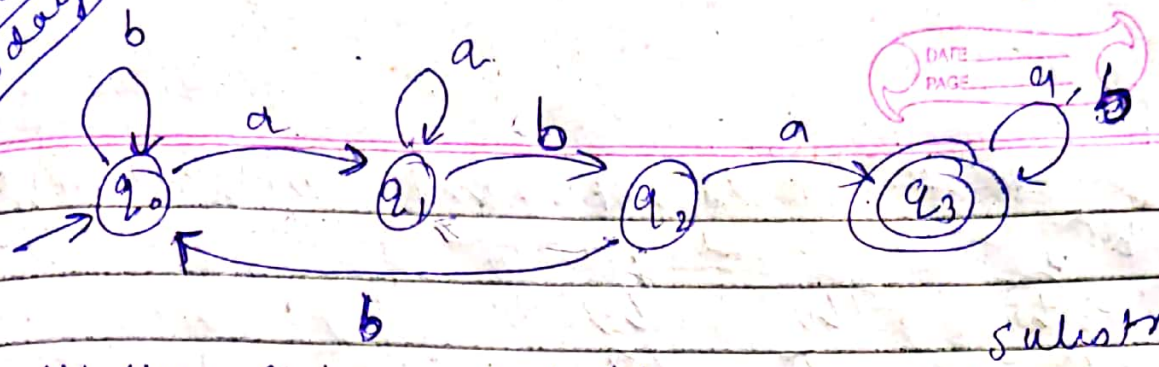
No tension of string 'ab'

Moving from the just nearer state from q2

(5) All the strings of ab have sub string 'aba'

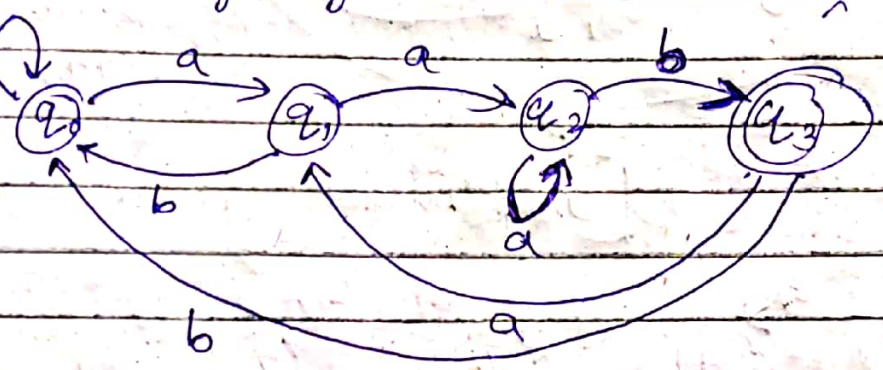
17.01.2023
Tuesday

loop may be skipped



(6) All the strings of 'ab' end with substring 'aab'

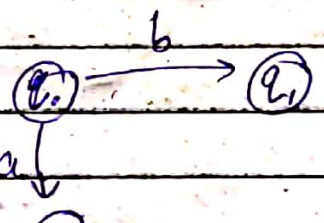
(4)



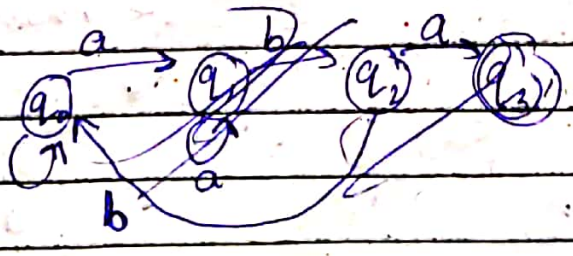
NOTE: Traps should be used carefully.
(only when no other state is possible)

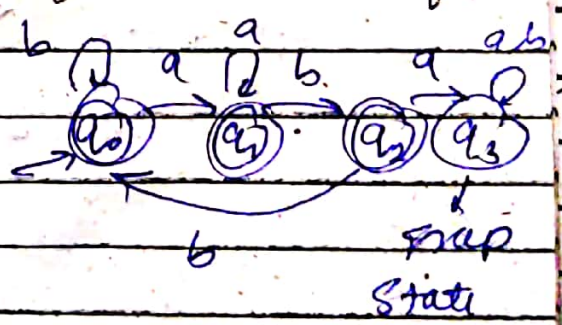
COMPLEMENT OF DFA

(5) All the strings of 'ab' do not have substring 'aba'



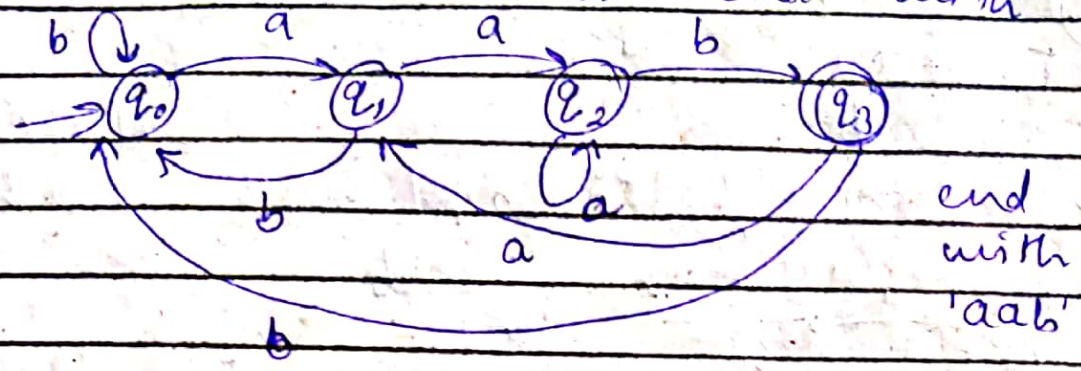
Let $M(Q, q_0, \Sigma, \delta, F)$ is a DFA
then its complement
$\bar{M} = M_1$ is $M_1(Q, q_0, \Sigma, \delta, Q-F)$
In other words change final to Non final and Non final to final.





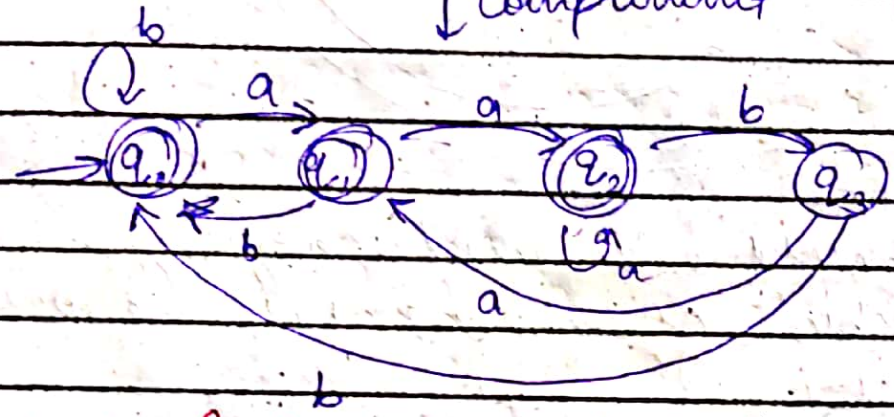
⑦ All ——"—— do not end with 'aab'

Sol:



end with 'aab'

↓ complement



do not end with 'aab'

Language Pattern

$L = \{ \text{def}^n \mid \text{cond}^n \}$

⑧ (i) $L = \{ a\,w\,a \mid w \in \{a,b\}^* \}$

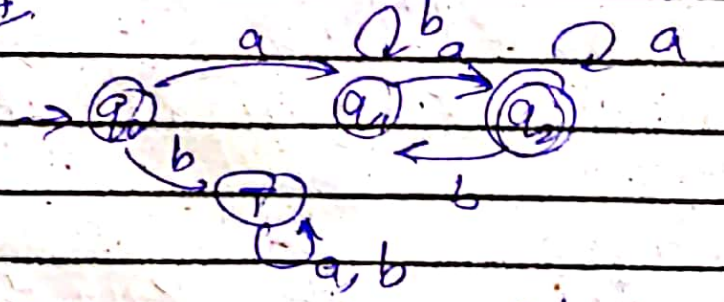
↳ Replace by any string of 'a,b'

→ Smallest string possible aa

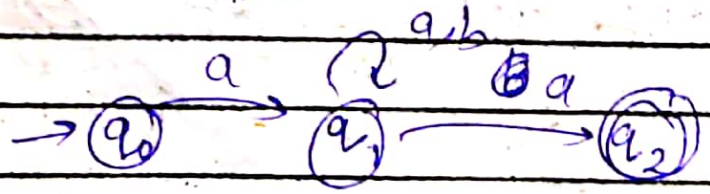
← min^n first & then generalise

Sol:

DFA



NFA



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

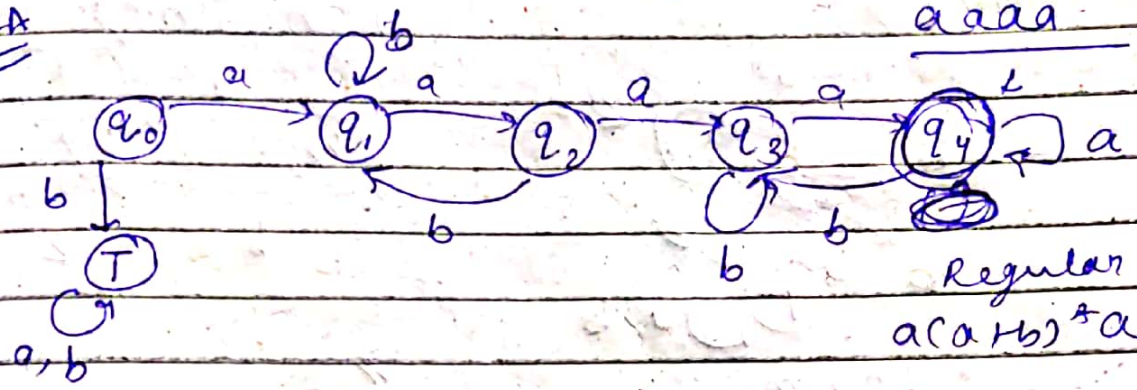
$w \quad w \longrightarrow$ Then same string at both but here no restriction

(ii) $L = \{a w_1 aa w_2 a \mid w_1, w_2 \in \{a,b\}^*\}$

construct DFA.

draw the first smallest string → aaaa

Sol: DFA

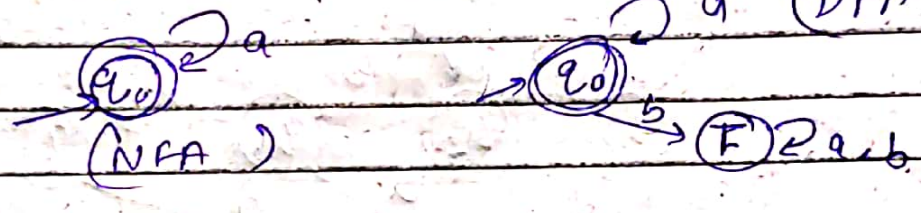


Regular expression

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

(iii) $L = \{a^n \mid n \geq 0\}$

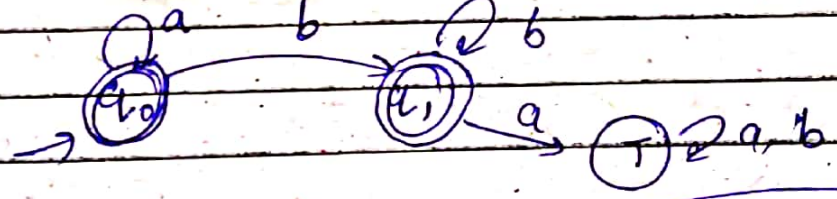
Sol: a^* (RE)



(NFA) (DFA)

(iv) $L = \{a^n b^m \mid n, m \geq 0\}$

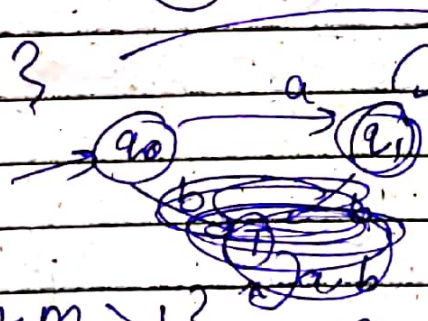
Sol: DFA



$a^* b^*$ (RE)

(v) $L = \{a^n \mid n \geq 1\}$

Sol: a^+ (RE)

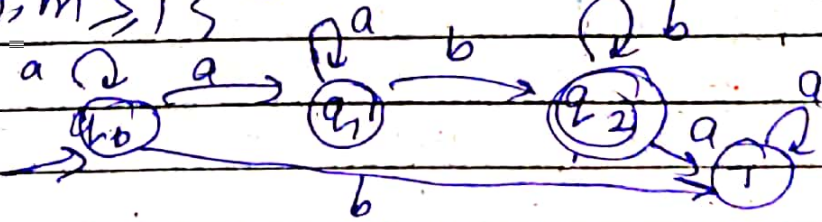


only 1 variable in language

(vi) $L = \{a^n b^m \mid n, m \geq 1\}$

Sol: $a^+ b^+$ (RE)

First string "ab"



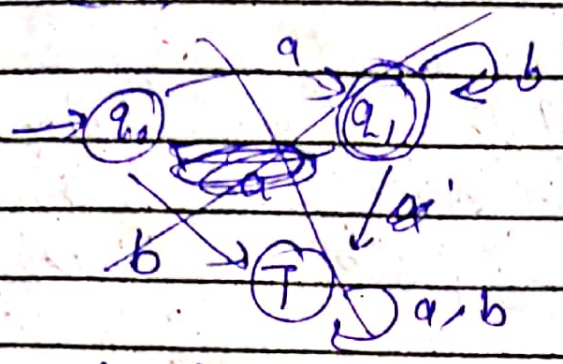
Scanned with CamScanner

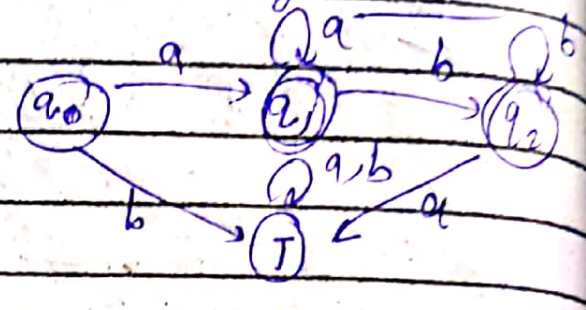
20.01.2023
Theory Friday

(vii) $L = \{a^n b^m \mid n \geq 1, m \geq 0\}$

Sol: $a^+ b^*$ $= a a^* b^*$
(RE)

Min. string 'a'

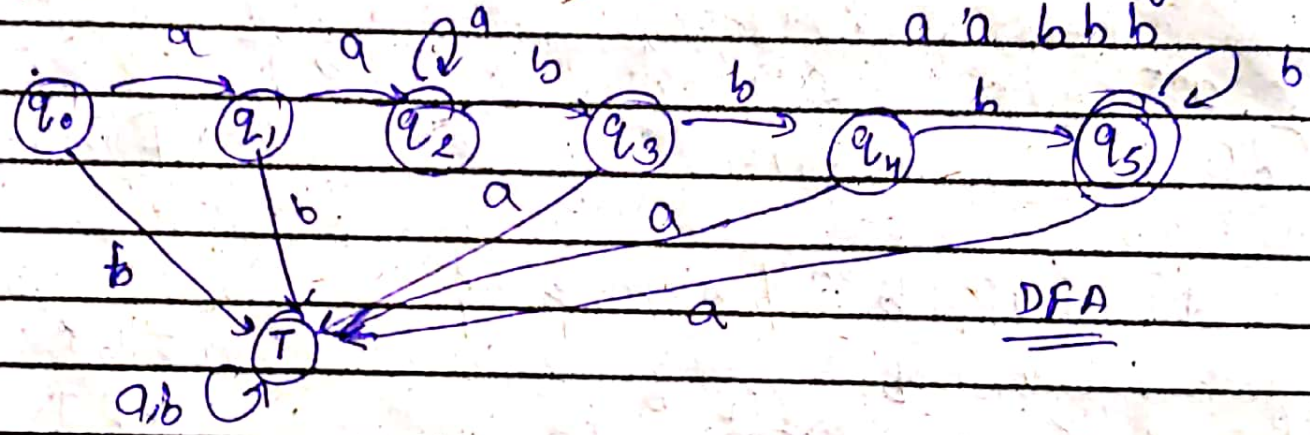




(viii) $L = \{a^n b^m \mid n \geq 2, m \geq 3\}$

Sol: RE $= aaa^* \, bbb \, b^*$

Min. string
aabbb

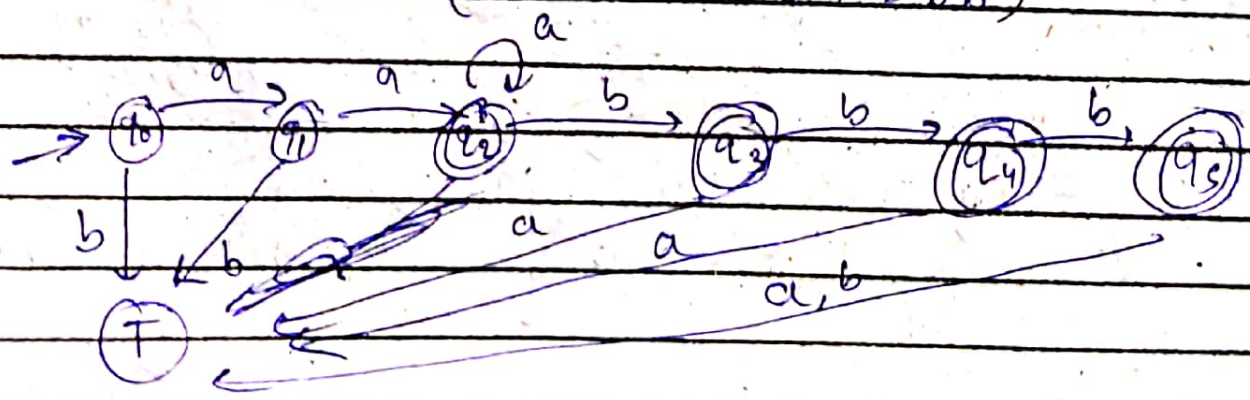


DFA

(IX) $L = \{a^n b^m \mid n \geq 2, m \leq 3\}$

$aaa^* + aaa^* b + aaa^* bb + aaa^* bbb$

RE $= aaa^* . (\epsilon + b + bb + bbb)$



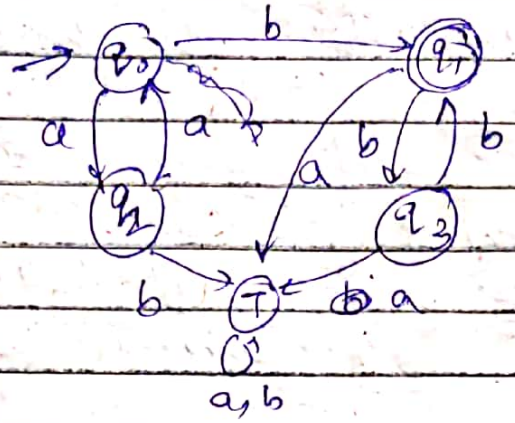
Scanned with CamScanner

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

↳ even no. of a & odd no. of b

$RE = (aa)^* \, b \, (bb)^*$

Min. string = b



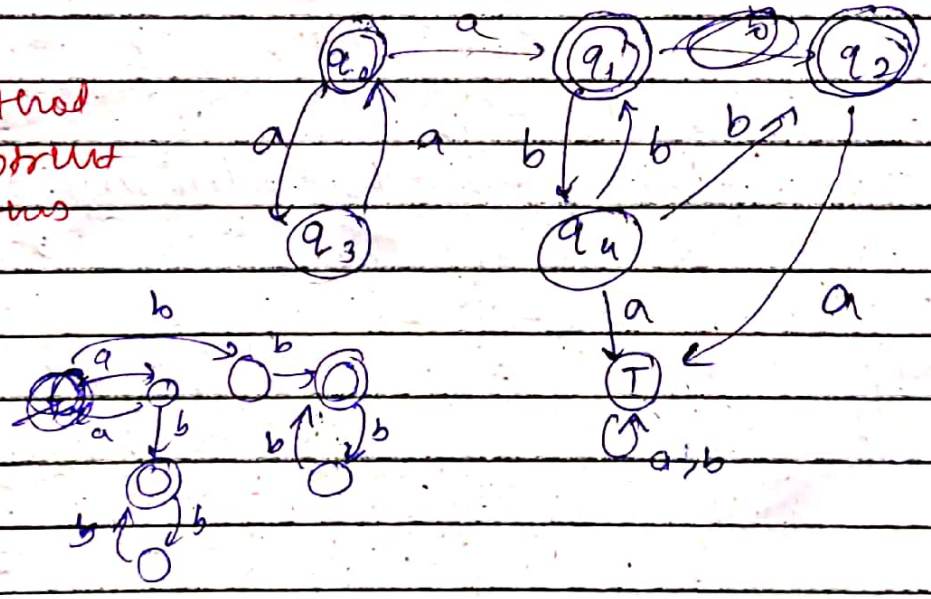
(xi) $L = \{a^n b^m \mid (n+m) \text{ is even}\}$

Sol: n odd m odd — n even m even

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

~~ab + 1~~

later we learn diff. method to construct this



Divisible & Modulo Problems (let by n)

(1). No. of states are fixed i.e., 'n'

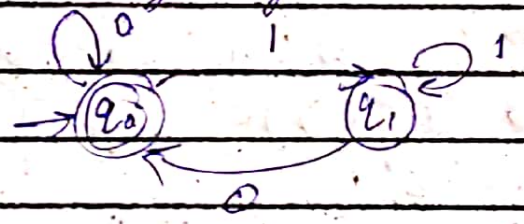
(2) Name of the states are also fixed which is fixed on the modulo result i.e, $q_0, q_1, q_2, \ldots, q_{(n-1)}$ (remainder)

(3) Initial State : q_0 and final states depends on required modulo result.

For Divisible problem final state will be q_0
remainder = 0

Q) All the strings of '01' are divisible 2

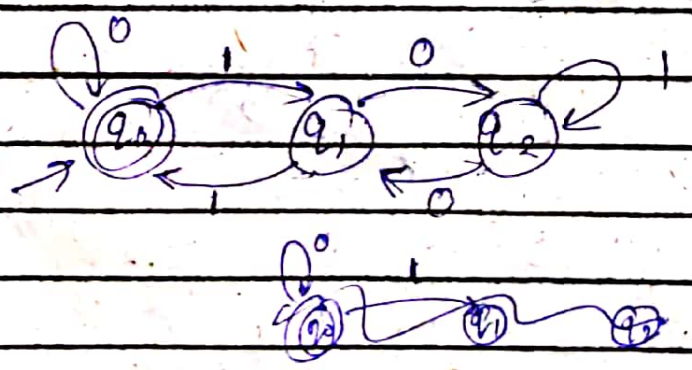
Sol:



$0 \% 2 = 0$
$1 \% 2 = 1$

This is 0001 q_1 STS
means $(1\ 0) \% 2 = 0$
$(1\ 1) \% 2 = 1$

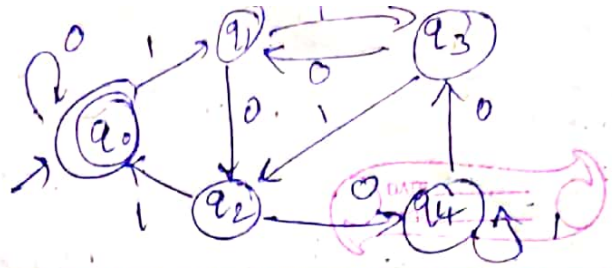
Q) All → '01' are divisible by (3) → No. of states



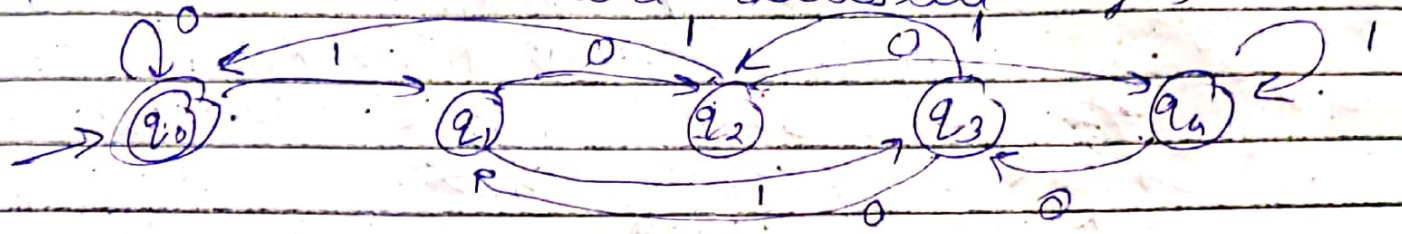
$0 \% 3 = 0$
$1 \% 3 = 1$

q_1 { 010 % 3 = 2
011 % 3 = 0

q_2 { 100 % 3 = 1
101 % 3 = 2



Q) All ——n —— are divisible by 5

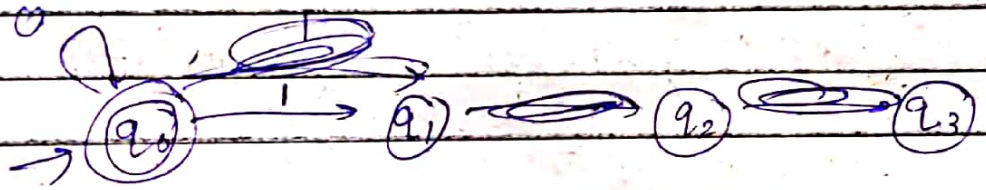


$q_1 \begin{cases} 0\overset{2}{10} \% 5 = 2 \\ 011 \% 5 = 3 \end{cases}$ $q_0 \begin{cases} 0 \% 5 = 0 \\ 1 \% 5 = 1 \end{cases}$

$q_3 = \begin{cases} \overbrace{0110}^{6} \% 5 = 1 \\ \underbrace{0111}_{7} \% 5 = 2 \end{cases}$ $q_2 \begin{cases} 0\overset{4}{100} \% 5 = 4 \\ 0\underset{5}{101} \% 5 = 0 \end{cases}$

$q_4 = \begin{cases} \overbrace{1000}^{8} \% 5 = 3 \\ \underbrace{1001}_{9} \% 5 = 4 \end{cases}$ $\overbrace{1000}^{8} \% 5$ $\overbrace{1001}^{9} \% 5$

Q) ——n —— 4

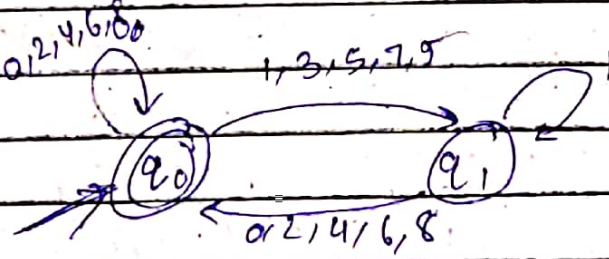


Q) All the strings of decimal alphabets are divisible by 2
↳ 0 to 9

Sol:

$\Sigma = [0-9]$

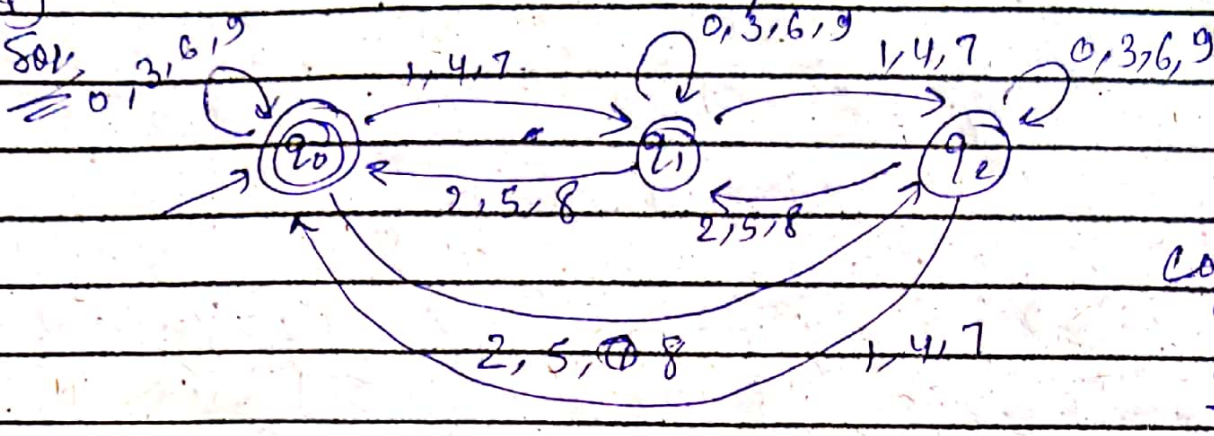
$0 \% 2 = 0$



Select any one number of set A proved All will also reach there

Q) ——→ Decimal alphabets are ——→ by 3

Sol:



$10 \% 3 = 1$
$400 \% 3 = 1$

Consider only 0, 1, 2 of each group

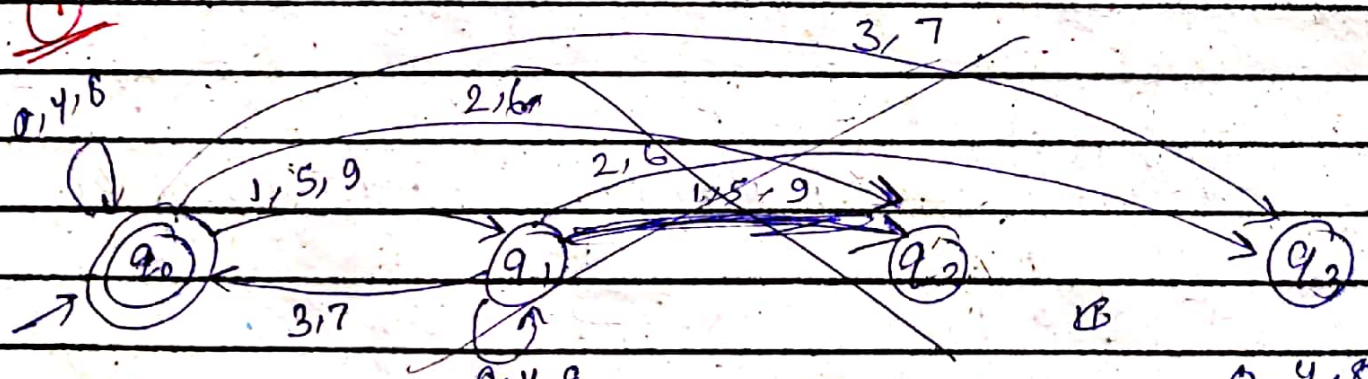
q_1 { $10 \% 3 = 1$, $11 \% 3 = 2$, 1, 4, 7, $12 \% 3 = 0$ }

q_3 = { $80 \% 3 = 2$, $81 \% 3 = 0$, $82 \% 3 = 1$ } Three were 2 paths Take any path you will reach at right place

$470 \% 3 = 2$
$471 \% 3 = 0$
$472 \% 3 = 1$

Q) ——→ Decimal Alphabets are ——→ by 4

(1)



0, 4, 8
1, 5, 9
2, 6
3, 7

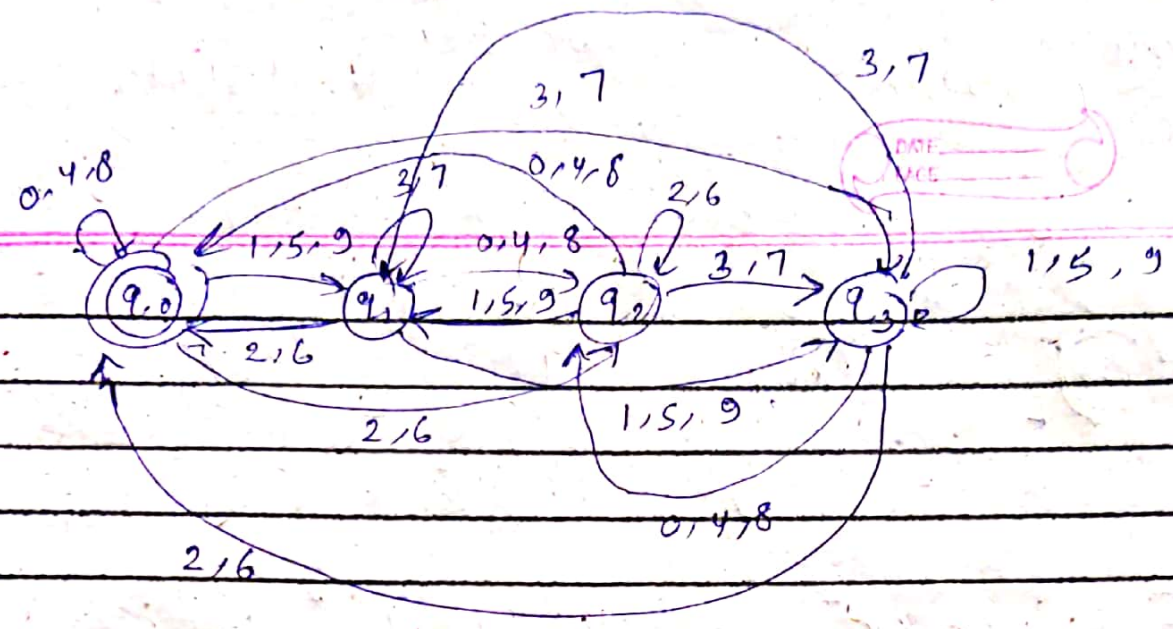
10 11 12 13

14 34

11 31

100
101
102
103

12 32
13 17

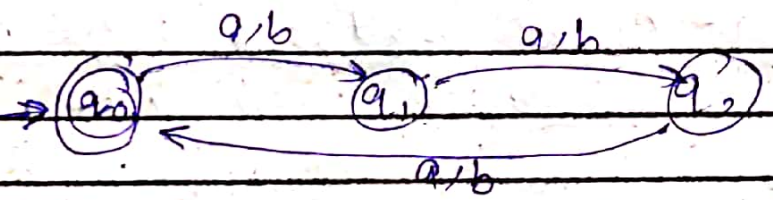


Q) $L = \{ |w| \bmod 3 = 0 \mid w \in \{a,b\}^* \}$

mod $\leftarrow |w|$ = length of 3

Sol: States are fixed = 3 = n

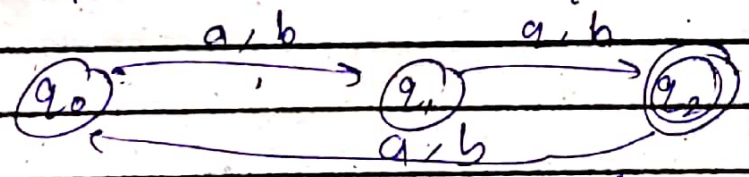
Final state = required modulo = q_0 (= 0.)



$[(a+b)\cdot(a+b)\cdot(a+b)]^*$

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

Sol:



RE $\cancel{a/b}$ $\cancel{(a+b)(a+b)}$ $[(a+b)\cdot(a+b)(a+b)]^*$

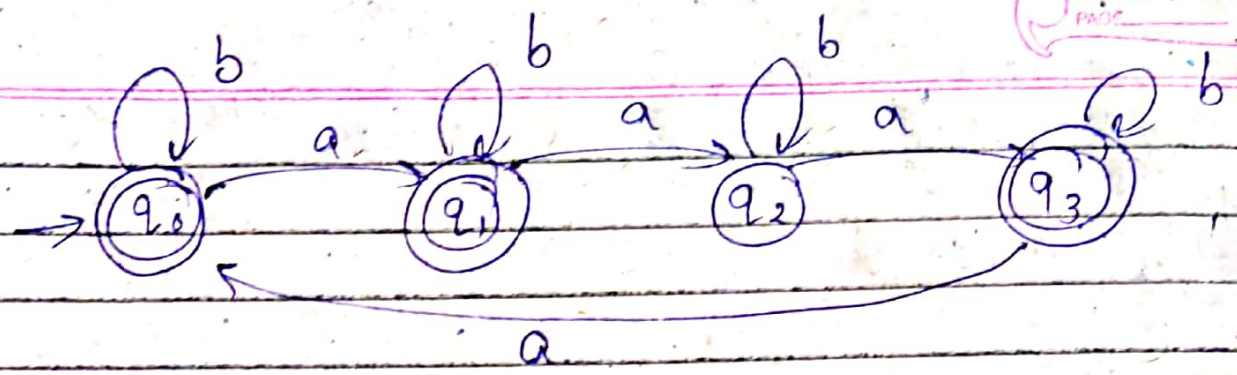
Q) $L = \{ n_a(w) \bmod 4 \neq 2 \mid w \in \{a,b\}^* \}$

↳ no. of a's in w

Sol:

Complement problem

Others than 2 all final state



$b^* + b^*ab^* + b^*ab^*ab^*ab^*$ ~~compl~~

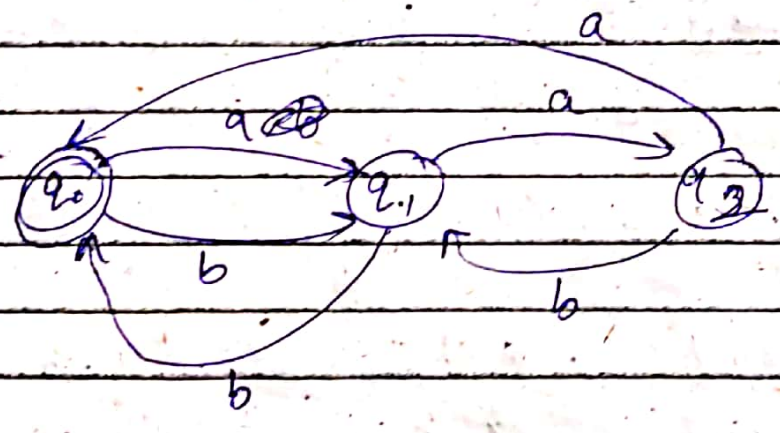
$+ b^*ab^*ab^*ab^*ab^*$

UNION AND INTERSECTION OF TWO DFAs

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

(done 2 pages ahead)

Sol:



23.01.2023
Monday

Q) All the strings of 'ab' have at least 1 'a' or 2 'b's

Sol: "or" "and"

Union & Intersection of 2 DFAs

Let M_1 & M_2 are 2 DFAs where

$M_1(Q_1, q_x, \Sigma, \delta_1, F_1)$

and $M_2(Q_2, q_y, \Sigma, \delta_2, F_2)$

Now we define one more DFA $M_3(Q_3, q_z, \Sigma, \delta_3, F_3)$ by cross product of M_1 & M_2 where $Q_3 = Q_1 \times Q_2$

↳ (Cartesian product)

$q_z = q_{xy}$

$\delta_3 = \delta_1 \times \delta_2$

{Cross product of transition only when they have same I/P symbol}:

$\forall\ \delta_1(q_A, a) \rightarrow q_C \in \delta_1$

$\forall\ \delta_2(q_B, a) \rightarrow q_D \in \delta_2$

State pair q_{AB}

$\forall\ \delta_3(q_{AB}, a) \rightarrow q_{CD} \in \delta_3$

{Based on final state → Union or Intersection}

F_3 = set of final states of M_3, $F_3 \in Q_3$

For union of M_1 & M_2 → either final state from M_1 or M_2

$F_3 = \{ q_{AB} \mid q_A \in F_1 \text{ or } q_B \in F_2 \}$

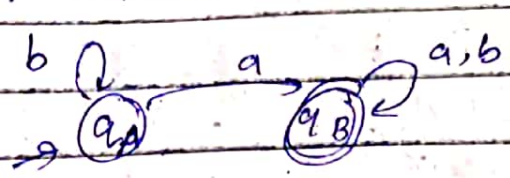
For intersection of M_1 & M_2

$F_3 = \{ q_{AB} \mid q_A \in F_1 \ \& \ q_B \in F_2 \}$

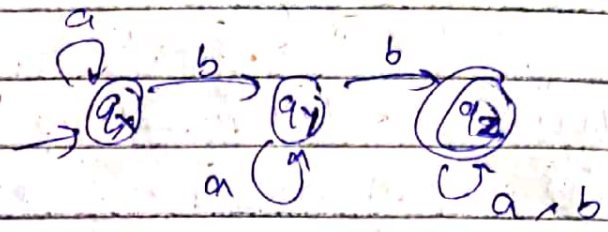
Scanned with CamScanner

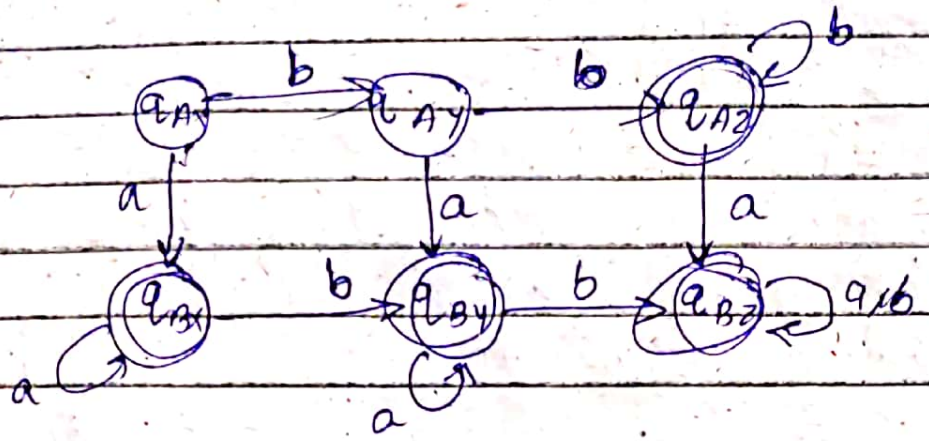
1

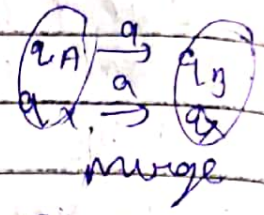
M_1 : atleast 1 'a'

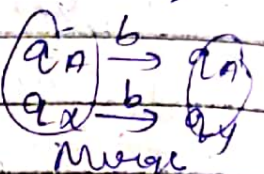


M_2 : atleast 2 'b's









$q_{Ax} \xrightarrow{b} q_{Ay}$

Final state (In union) either B or Z

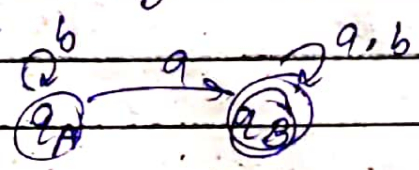
q_{Bx}, q_{By}, q_{Bz}, q_{Az}

(in intersection) B and Z q_{Bz}

Q) All the strings of 'ab' have atleast 1 'a' and exactly 1 'b'

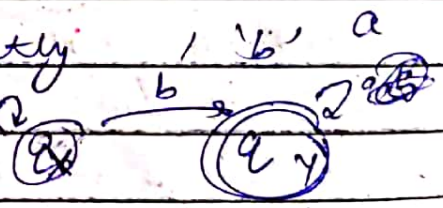
Sol: Exactly 1 'a'

M_1

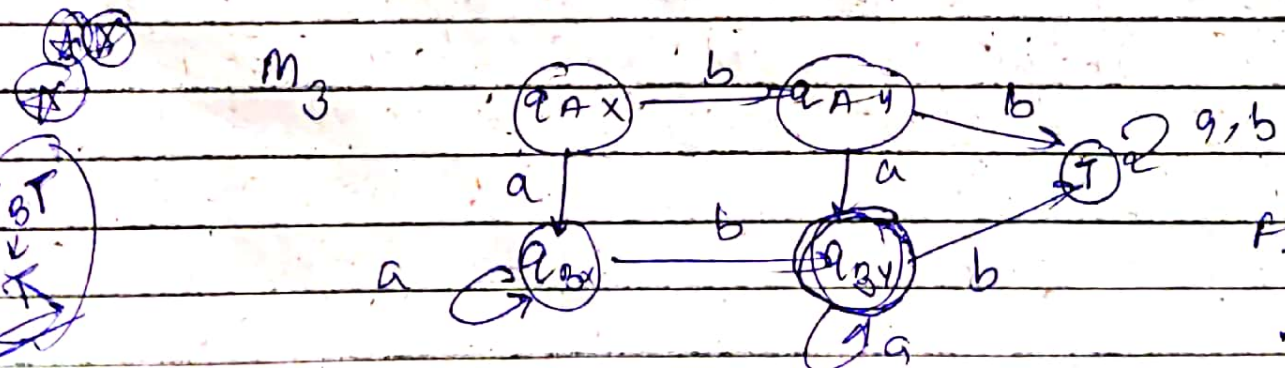


Exactly 1 'b'

M_2



M_3



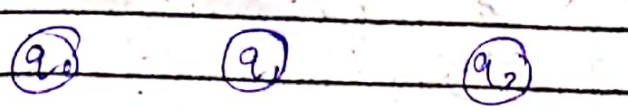
FS. q_{BY}

(Intersection)

NOTE: In case of Trap state only consider it once as DFA can have atmost 1 Trap state.

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

Sol: M_1: $n_a(w) \bmod 3$ $\qquad$ M_2: $n_b(w) \bmod 3$

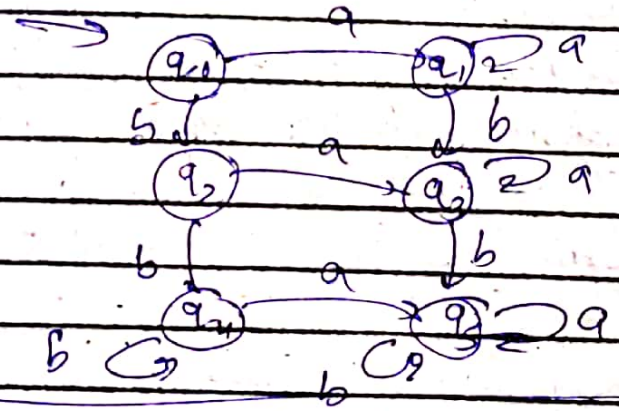


Observation in (1):

Vertical Axis $\quad M_1 \quad$ Problem 1

Horizontal Axis $\quad M_2 \quad$ Problem 2

or vice versa axis



~~Vertical~~

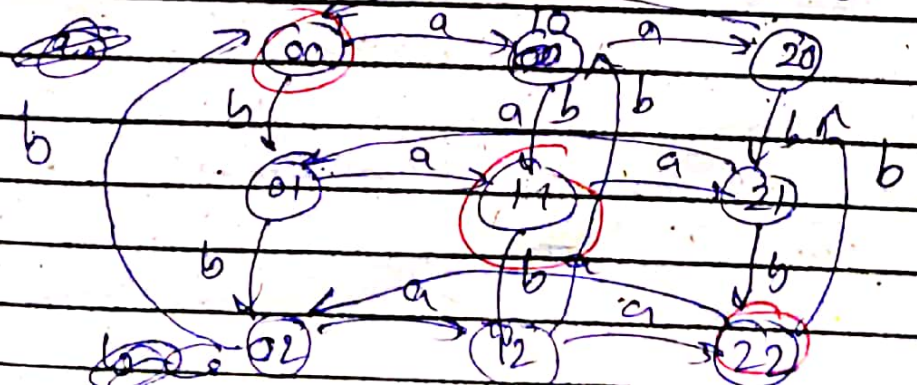
~~Horizontal~~ $\rightarrow M_1$

~~Vertical~~ $\rightarrow M_2$

M_1 DFA replicated M_2's state time & same for M_1

(1) No. of States = No. of states (M_1) × No. of states (M_2)

$= 3 \times 3 = 9$

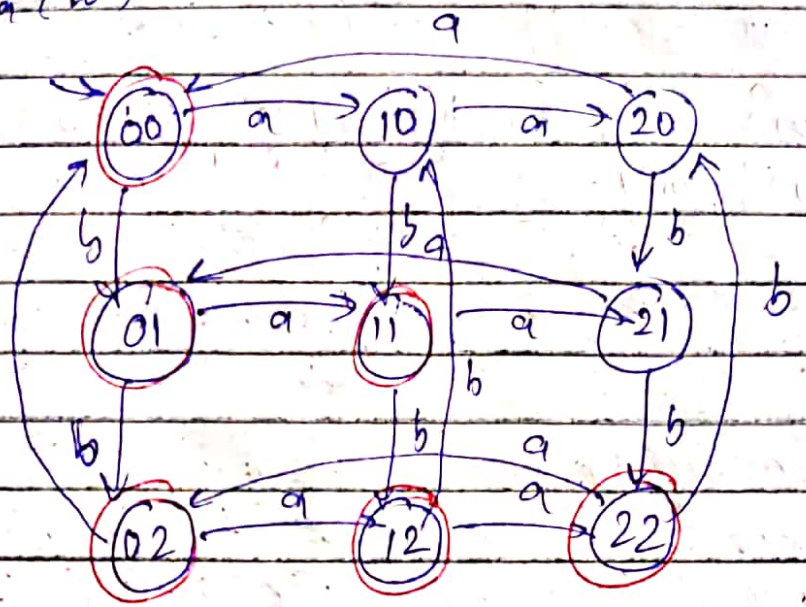


a b → No. of a, No. of b

06.02.2023
Monday.

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

Sol: ~~$n_a(w)$~~



a	b
a	b

Regular Expression to Finite Automata

(i) Follow POS Method — We get DFA from RE directly.

Follow POS ($\{x\}$) = set of all possible positions that can follow position pos 'x'.

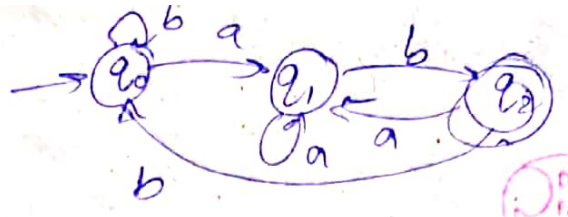
Step 1 Add unique end marker '#' at the end of R.
R #

Step 2. Assign a position ~~to~~ number to every symbol present in R updated regular expression by:
Step 1. (if any symbol present multiple times, we assign it a new position number).

ex. $R = (a+b)^* ab$
[All strings of ab that ends with ab

Step 1 $R\# = (a+b)^* a.b \#$

Step 2. position 1 2 3 4 5 — new no. to new position.



Step 3 Now find follow pos of each position

Prepare table

$(a+b)^*$

= ab or aa

	Positions	Follow POS
a	{1}	~~{1}, {2}, {3}~~ {1,2,3}
b	{2}	{1, 2, 3}
a	{3}	{4}
b	{4}	{5}
#	{5}	—

Step 4 Identify initial state. ~~St~~ : set of all positions from where regular expression can begin will represent initial state.

$(a+b)^*$ a or b (1) (2) or ε (3) ~~a (next state)~~ → initial state

→ (1,2,3)

Step 5 To identify transitions of each state

First find positions that represents input symbol.

look for pos of a which is transition

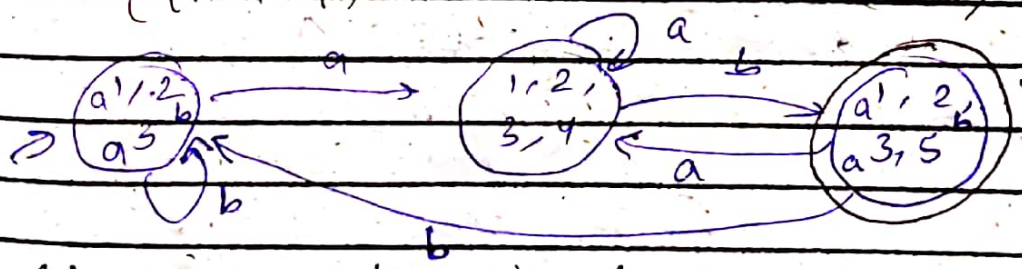
$\delta(\{1,2,3\}, a)$ = Follow Pos ({1}) ∪ Follow pos({3})

= {1,2,3} ∪ {4} = {1, 2, 3, 4}

$\delta(\{1,2,3\}, b)$ = Follow POS ({2}) = {1, 2, 3}

self loop

new state



For transition

Step 6: To identify final state(s):

We have to check the position of unique end marker in states. All the states in which position of # is present will become the final states

unique end marker

Q) $((a+b)^* a b^* + a^* b (ab)^*)$ → put brackets & then put #

Sol: RE = $(\underset{1}{(}\underset{2}{a}\underset{3}{+}\underset{4}{b})^* \underset{5}{a}\underset{6}{b}^* + \underset{7}{a}^* \underset{8}{b} (\underset{9}{ab})^* \underset{9}{)}$ #

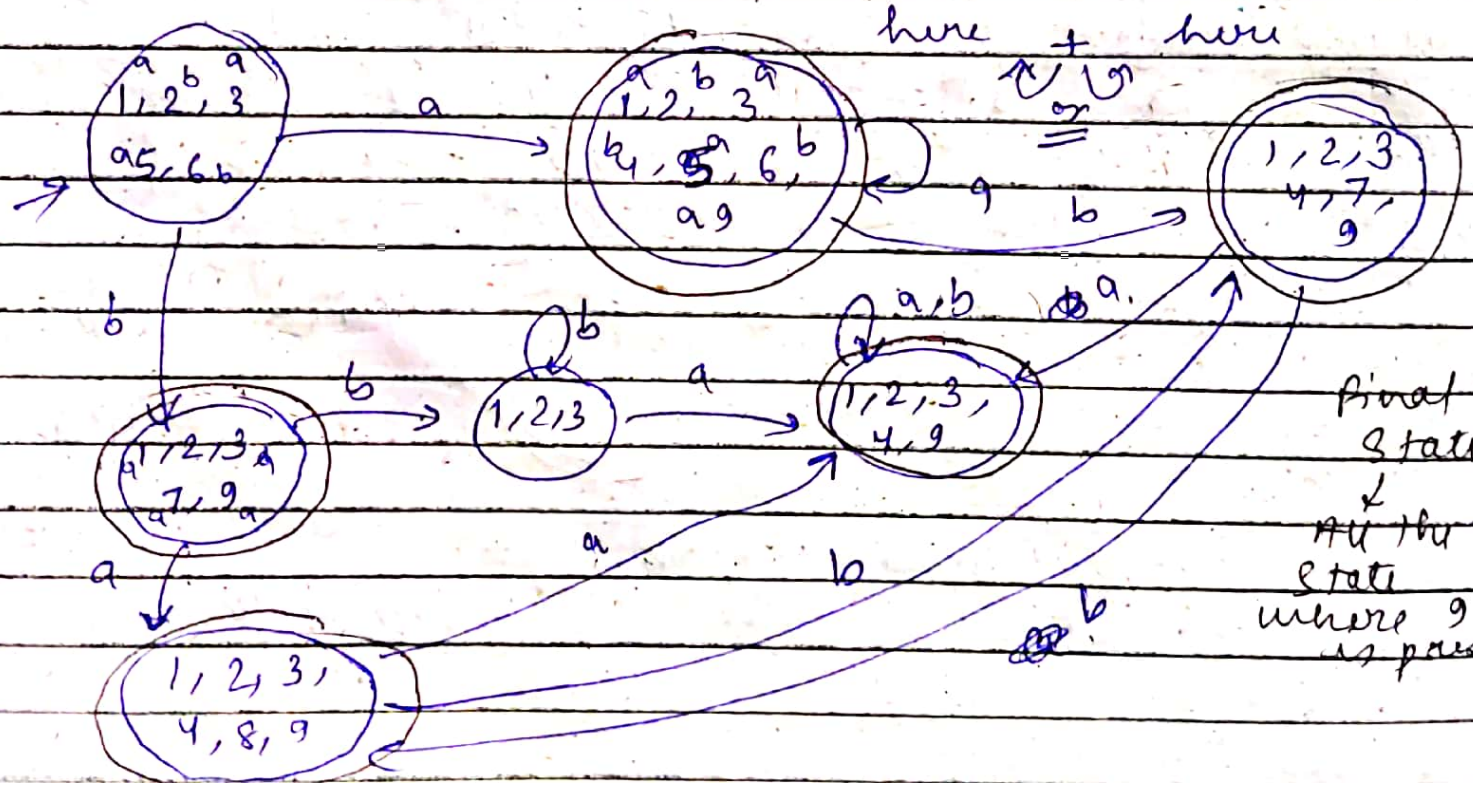
	Position	Follow POS
a	{1}	{1,2,3}
b	{2}	{1,2,3}
a	{3}	{4,9}
b	{4}	{4,9}
a	{5}	{5,6}
b	{6}	{7,9}
a	{7}	{8}
b	{8}	{7,9}
	{9}	—

don't take these in follow pos

$(ab)^*$ = ab abab

1,2,3,4,7,9

Initial State = 1,2,3,5,6 (Observation) here + here



Final State ↓ All the state where 9 is present

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

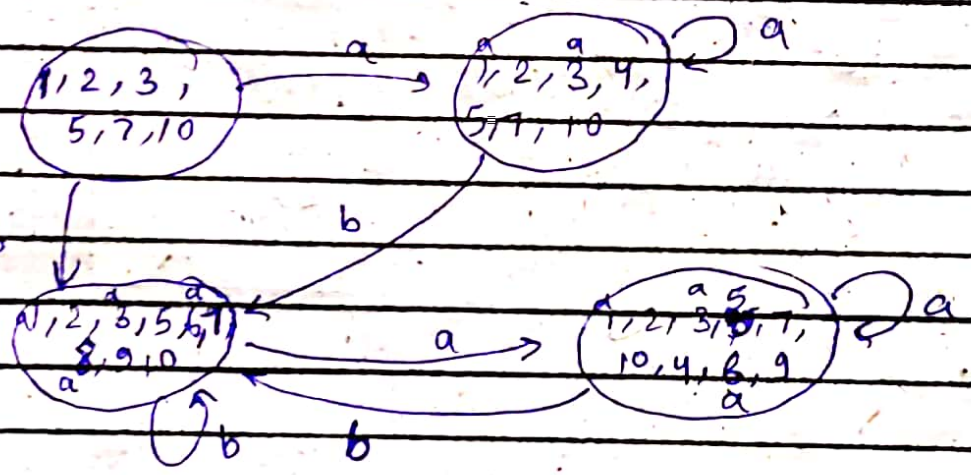
Sol: $R\# = ((a+b)^*(a.b)^* + (ba)^* b(a+b)^*)^* \#$

1 2 3 4 5 6 7 8 9 10

	Pos	Follow Pos
a	{1}	{1, 2, 3, ~~10~~ 5, 7, 10}
b	{2}	{1, 2, 3, ~~10~~ 5, 7, 10}
a	{3}	{4} ~~10~~
b	{4}	~~{3, 10}~~ {1, 2, 3, 5, 7, 10} {2}
b	{5}	{6}
a	{6}	{5, 7} ~~1, 2, 3, 10~~
b	{7}	{8, 9, 1, 2, 3, 5, 7, 10}
a	{8}	{8, 9, 10, 1, 2, 3, 5, 7}
b	{9}	{8, 9, 10, 1, 2, 3, 5, 7}
#	{10}	—

Repeat itself which

Initial state 1, 2, 3, 5, 7, 10



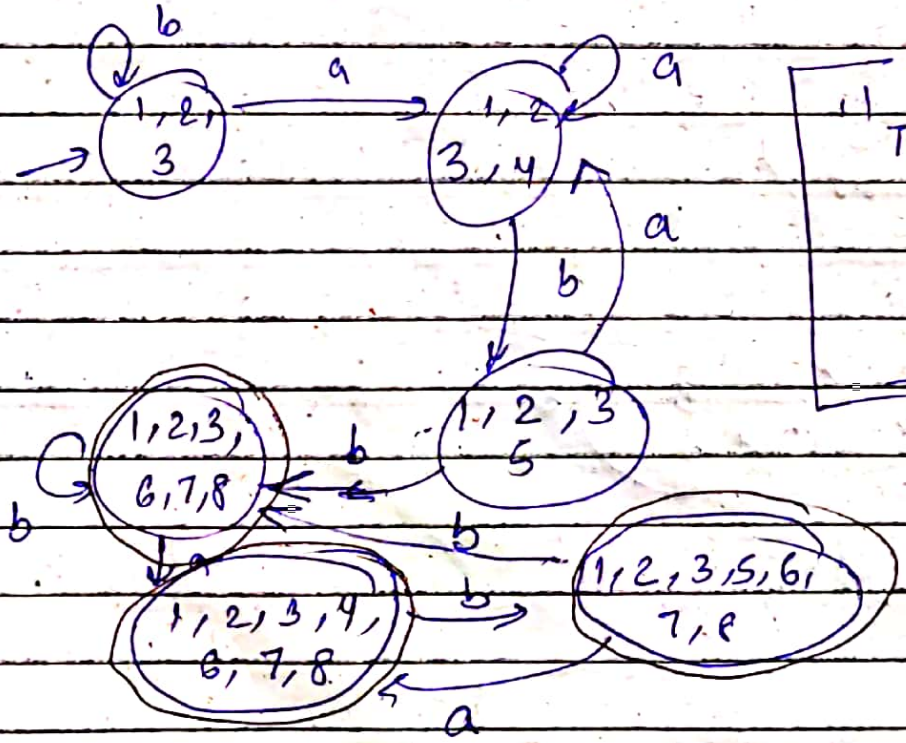
03.02.2023
Thursday

Q) Find DFA using FollowPos method.

$R = (a+b)^* \; a \; bb \; (a+b)^*$

Sol: $R\# = \underset{1}{(a} \underset{2}{+b)^*} \; \underset{3}{a} \; \underset{4}{b} \; \underset{5}{b} \; \underset{6}{(a} \underset{7}{+b)^*} \; \underset{8}{\#}$

	Pos	FollowPos
a	{1}	{1,2,3}
b	{2}	{1,2,3}
a	{3}	{4}
b	{4}	{5}
b	{5}	{6,7,8}
a	{6}	{6,7,8}
b	{7}	{6,7,8}
#	{8}	–

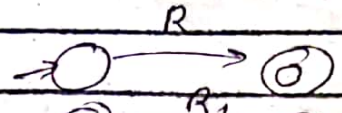


"This is not a minimum DFA"

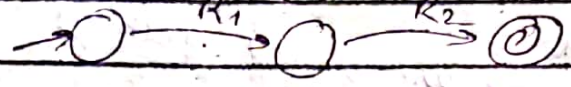
Myhill Nerode Theorem & we will study it later

Regular Expression to Finite Automata

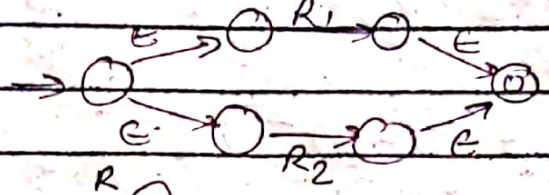
① R



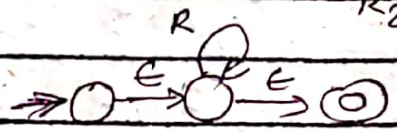
② $R_1 . R_2$



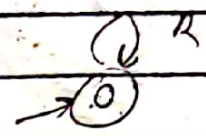
③ $R_1 + R_2$



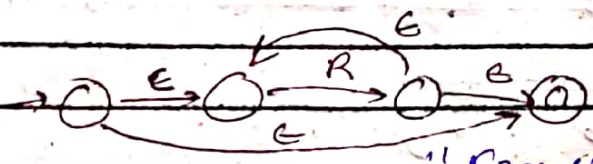
④ R^*



Simplified version

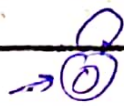


"For smaller expression"

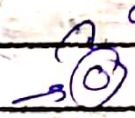


"For complex structure"

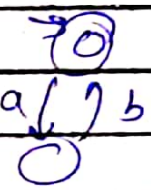
a^*



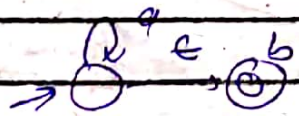
$(a+b)^*$



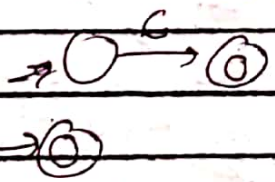
$(a+b)^*$



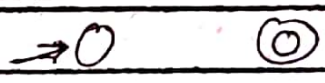
a^*b^*

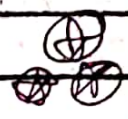


⑤ ϵ



⑥ ϕ





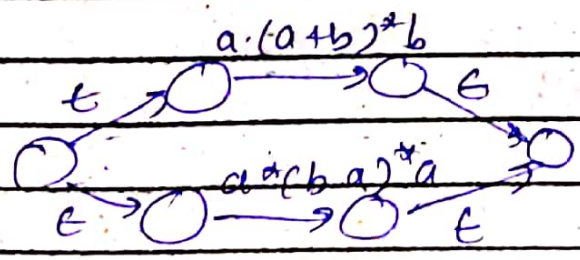
NOTE: Use minimum no. of ϵ

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

① First check whole

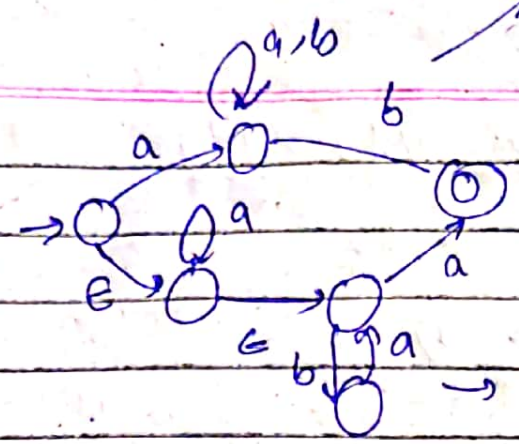
② check go for '+'

Sol:



Scanned with CamScanner

This is FA & not DFA



Draw this

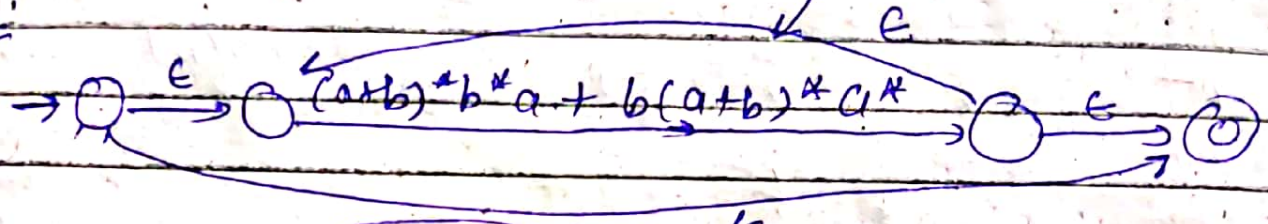
→ we were getting $(a \cdot b)^*$

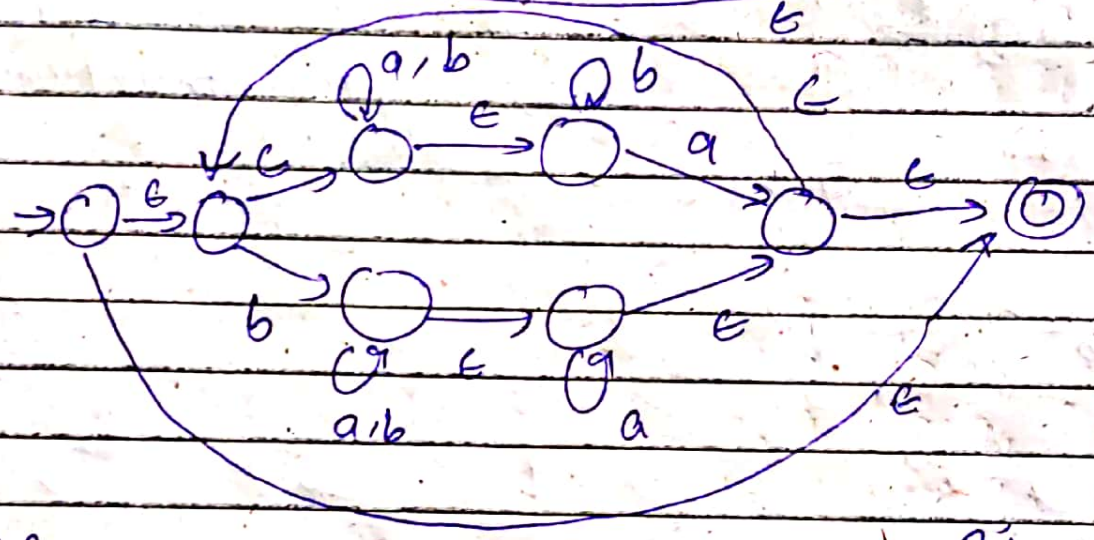
↳ 2nd way to represent

Q) $((a+b)^* \cdot b^* \cdot a + b(a+b)^* a^*)^*$

M.1:

Sol:

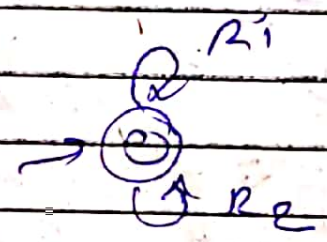


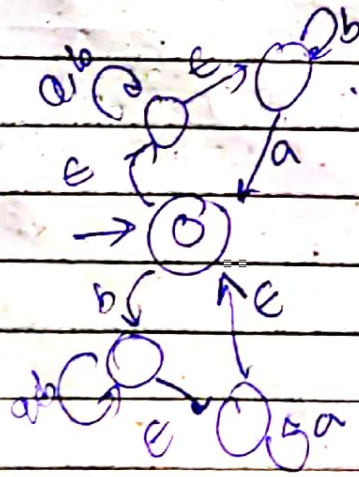


Preferred one

M.2.

$R_1 + R_2$





Scanned with CamScanner

Subpart → Dedicated Symbol → Subpart

No ϵ

upper part

ϵ is used only when there is 2 subpart

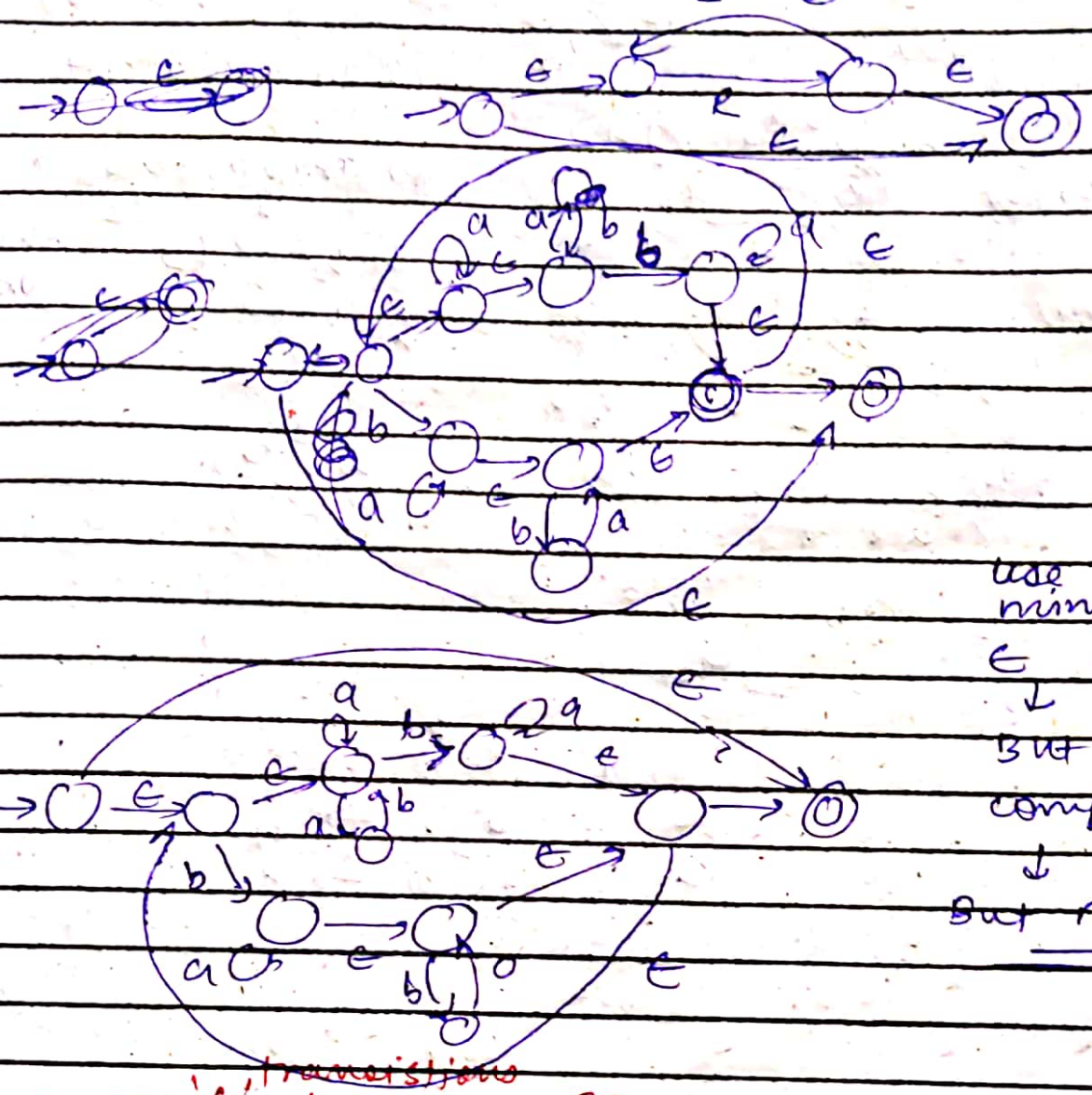
$(R_1)^* \cdot (R_2)^*$

comm part

+ is also subpart

Q) $\underbrace{((a+ba)^* b \cdot a^*}_{R_1} + \underbrace{b \cdot a^* (ba)^*}_{R_2})^*$

Sol.



use minimum ϵ ↓ but no compulsory ↓ but minimum

Remove 'ε' transitions from FA

$x \xrightarrow{\epsilon} y$

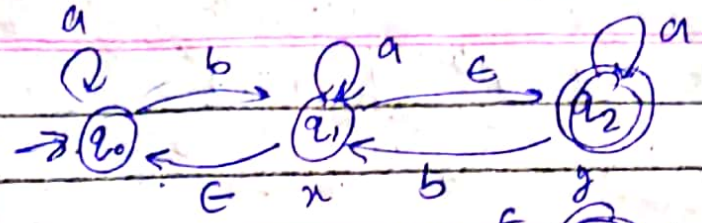
(i) If 'x' is an initial state then make 'y' also initial

(ii) If y is a final state then make 'x' also final

(iii) Replicate all the transitions of 'y' from 'x'

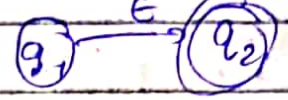
(Check in order

Ex

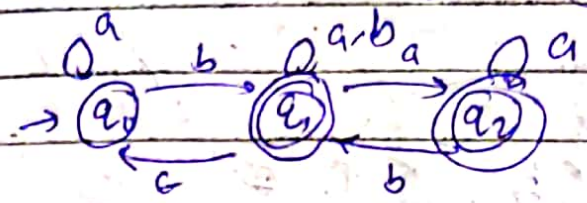


Choose any 'ϵ' & operate

It's remove



$\therefore q_1$ = final state



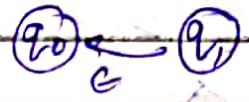
All Transitions from q_2

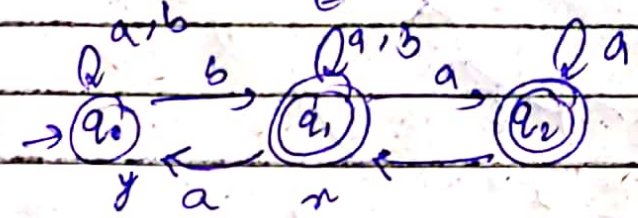
$q_1 \cancel{q_2} \xrightarrow{a} q_2$

$q_1 \cancel{q_2} \xrightarrow{b} q_1$

Change src

Remove

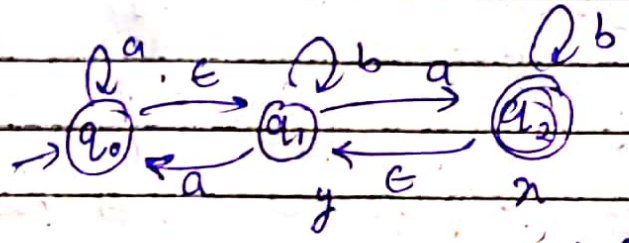




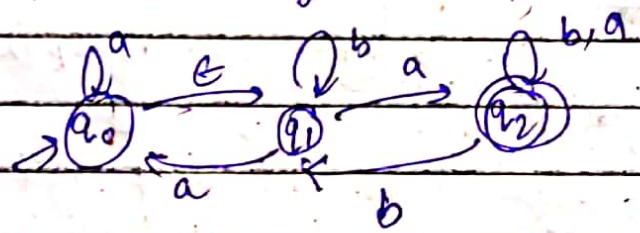
$q_1 \cancel{q_0} \xrightarrow{a} q_0$

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

Q)



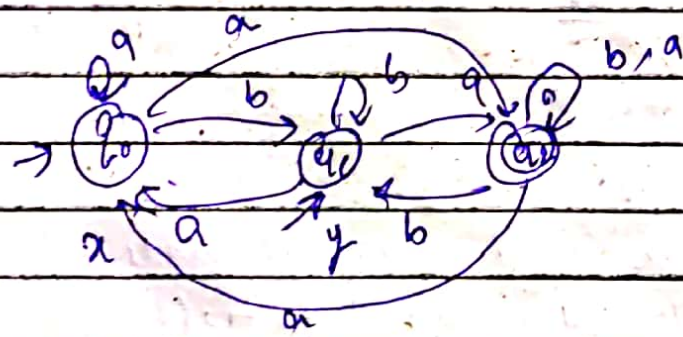
Sol:



$q_2 \cancel{q_1} \xrightarrow{a} q_2$

$q_2 \cancel{q_1} \xrightarrow{b} q_1$

$q_2 \cancel{q_1} \xrightarrow{a} q_0$



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

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

$q_0 \cancel{q_1} \xrightarrow{a} q_2$

It's not a correct FA.

Why?

Reason: Two input states not a FA

14.02.2023
Tuesday

NFA to DFA CONVERSIONS

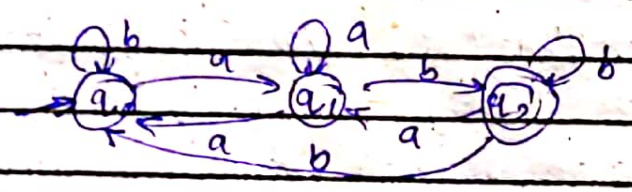
NFA without 'ε' transitions

Set Subset Method to convert NFA without 'ε' to DFA

1) Prepare NFA transition table.
2) With the help of NFA transition table we prepare DFA transition table.
   (a) First prepare initial state which is same as initial state of NFA
   (b) For any state of DFA prepare its transition

$$\delta(q_i, a) = NFA(\delta(q_i, a))$$

When NFA $(\delta(q_i, a))$ has more than states then merge them to form a single state.



NFA transition table

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

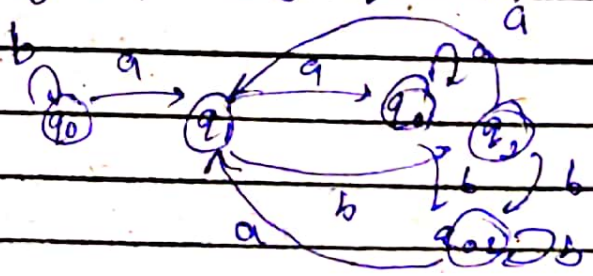
जो नया ε आये उसका Transition बना ओ।

DFA Transition table

States	Inputs a	b
→ q_0	q_1	q_0
q_1	q_{01}	q_2
q_{01}	q_{01}	q_{02}
q_2	q_1	q_{02}
q_{02}	q_1	q_{02}

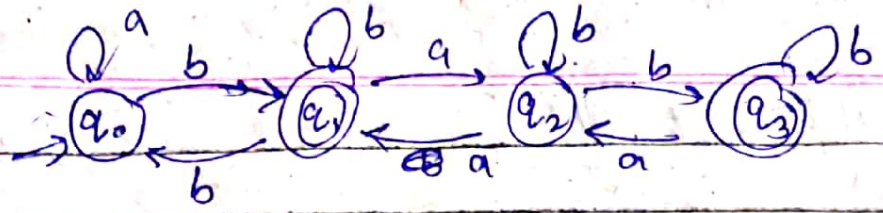
single तो सही वरना ~~एक~~ New State

//Merge ~~up~~ S of q_0 & q_1



Final states of DFA: All the states in which any of the final states of DFA is present

any state where 2 is present

Q) 

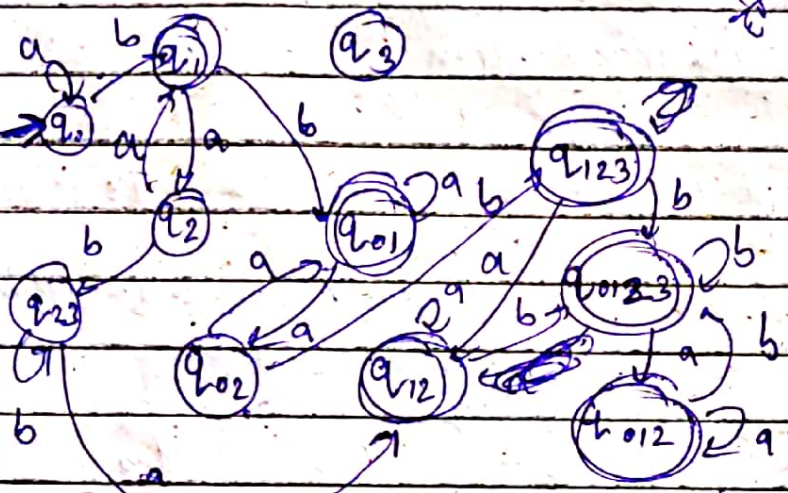
Sol. NFA transition

State	a	b
→ q_0	q_0	q_1
q_1	q_2	$\{q_0, q_1\}$
q_2	q_1	$\{q_2, q_3\}$
q_3	q_2	q_3

DFA

States	a	b
→ q_0	q_0	q_1
q_1	q_2	q_{01}
q_2	q_1	q_{23}
q_{01}	q_{02}	q_{01}
q_{23}	q_{12}	q_{23}
q_{02}	q_{01}	q_{123}
q_{12}	q_{12}	q_{0123}
q_{123}	q_{12}	q_{0123}
q_{0123}	q_{012}	q_{0123}
q_{012}	q_{012}	q_{0123}

सभी states में 3 तरह के होते हैं



→ Diagram not necessary.

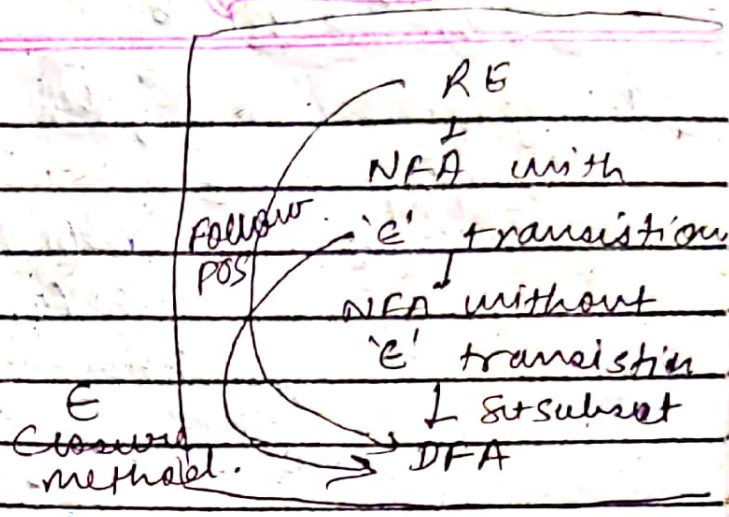
If any NFA has M states then its equivalent DFA can have maximum 2^M states

Trap state + other states

1 2 3
12 13 1 2
23 1 2
123 3×2 = 6

State → ε Closure
Transition → कुछ नहीं !

NFA to DFA
with 'ε' transitions

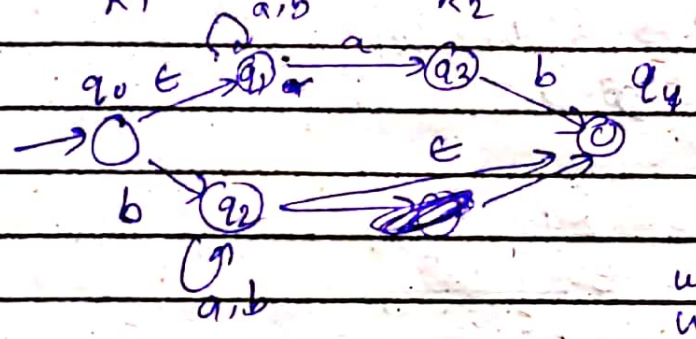


ε Closure Method :-

ε-closure({x}) = set of all states where we can reach via 'ε' transistions

Q) $((a+b)^* ab + b(a+b)^*)$
R_1 R_2

Sol:



Initial state of DFA
= ε-closure (Initial State of NFA)
= ε-Closure ({q0})
A = {q0, q1}
without ip we can reach here

A = ε Closure ({q0}) = {q0, q1}

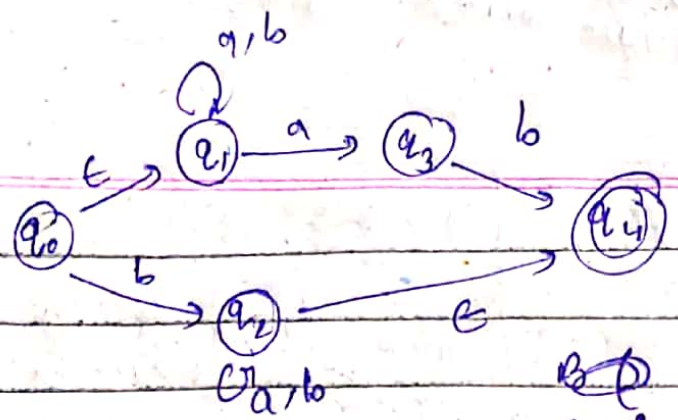
DFA transistion table

States	a	b
→ A	B	C
B	B	D
C	E	C
D	B	F
E	E	C
F	B	F

δ(A, a) = δ({q0, q1}, a)
= NFA(δ(q0, a)) ∪ NFA(δ(q1, a))
NO {q1, q3}
= {q1, q3}
ε-Closure (δ(A, a)) = ε Closure ({q1, q3})
= {q1, q3} = B
= ε-Closure ({q1, q3})
δ(A, b) = δ({q0, q1}, b) = {q1, q2}
ε Closure (δ(A, b)) = ε Closure ({q1, q2})
= {q1, q2, q4} = C

Scanned with CamScanner

First find transition
Then find ∈-closure to get next state



$\delta(B,b) = \delta(\{q_1, q_3\}, b) = \{q_1, q_4\}$

$\in(q_1, q_4) = \boxed{\{q_1, q_4\} = D}$

transition ← $\delta(B,a) = \delta\{q_1, q_3\}, a\} = \{q_1, q_3\}$

∈ closure ← $\in\{q_1, q_4\} = \{q_1, q_4\} = D$

$\delta(C,a) = \delta(\{q_1, q_2, q_4\}, a) = \{q_1, q_2, q_3\}$

$\in(\{q_1, q_2, q_3\}) = \boxed{\{q_1, q_2, q_3, q_4\} = E}$

$\delta(C,b) = \delta(\{q_1, q_2, q_4\}, b) = \{q_1, q_2\}$

$\in\{q_1, q_4\} = \{q_1, q_2\} = C$

$\delta(D,a) = \delta(\{q_1, q_4\}, a) = \{q_1, q_3\} = B$

$\in(q_1, q_3) = \{q_1, q_3\} = B$

$\delta(D,b) = \delta\{q_1, q_4\}, b\} = \boxed{\{q_1\} = F}$

$\in\{q_1\}$

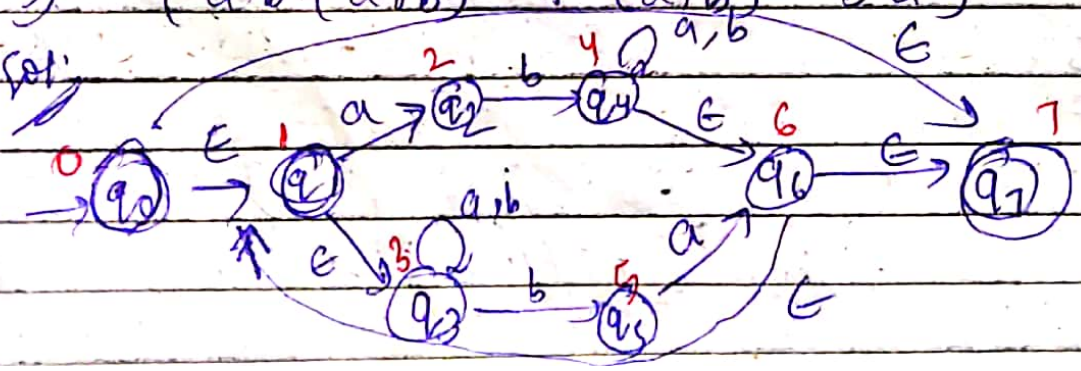
$\delta(E,a) = \delta(\{q_1, q_2, q_3, q_4\}, a\} = (q_1, q_2, q_3) = E$

$\delta(E,b) = \delta(\{q_1, q_2, q_3, q_4\}, b) = \{q_1, q_2, q_4\} = C$

$\in\{q_1, q_2, q_4\} = \{q_1, q_2, q_4\}$

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

Sol:



Scanned with CamScanner

← Closure of state आती है)

○ Final States
States containing

(A) = {0, 1, 3, 7}

DFA

States	a	b
→ (A)	B	C
B	D	E
C	F	C
D	D	C
(E)	G	E
(F)	B	C
(G)	G	E

$A = \epsilon\ closure(\{q_0\}) = \{q_0, q_1, q_3, q_7\}$

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

$\epsilon\{q_2, q_3\} = \boxed{\{q_2, q_3\} = B}$ $B = \{2, 3\}$

$(A, b) = (\{q_0, q_1, q_3, q_7\}, b) = \{q_3, q_5\}$

$\epsilon\{q_3, q_5\} = \{q_3, q_5\} = C$ $C = \{3, 5\}$

(B, a)

$((q_2 q_3), a) = \{q_3\}$ direct transition

$\epsilon\{q_3\} = \{q_3\} = D$ $D = \{3\}$

$\{(q_2, q_3), b\} = (q_3, q_4, q_5)$

$\epsilon\{q_3, q_4, q_5\} = \{q_3, q_5, q_4, q_7, q_1\} = E$ $E = \{1, 3, 5, 6, 7\}$

C $\{3, 5\}, a = \{q_3, q_6\}$ → all the states using "ε" trans

$\epsilon\{q_3, q_6\} = \{q_3, q_6, q_7, q_1, q_3\} = F$ $F = \{1, 3, 6, 7\}$

$\{3, 5\}, b = \{3, 5\}$

$\epsilon\{3, 5\} = C$

D $\{q_3, a\} = \{q_3\}$ $\epsilon\{q_3\} = D$

$\{q_3, b\} = \{q_3, q_5\}$ $\epsilon\{q_3, q_5\} = C$

E $(q_1, q_2, q_3, q_4, q_5, q_6, q_7), a) = \{q_2, q_3, q_4, q_6\}$

$\epsilon\{q_2, q_3, q_4, q_6\} = \{q_2, q_3, q_4, q_6, q_7, q_1\} = 1, 2, 3, 6, 7$

$G = \{1, 2, 3, 4, 6, 7\}$

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

$\epsilon\{q_3, q_4, q_5\} = E$

F $\{q_1, q_3, q_6, q_7\}, a\} = \{q_2, q_3\} = B$

$\{$ — $b\} = \{3, 5, \} = C$

Necessary condⁿ
↓ True
Suff condⁿ

~~Also~~ Sometimes check final set to see repetitive set

$\underline{q}$ $(1,2,3,4,6,7), a$ $\{q_2\ q_3\ q_4\}$ ✓

$\in q_2\ q_3\ q_4 = \{q_2\ q_3\ q_4\ q_6\ q_1\}$

$1,2,3,4,6,7 = G$

$(1,2,3,4,6,7), b = \{q_1, 3, 5, q_7\} = E$

NOTE: ↑ will not give.

All the methods discussed above ~~are not~~ minimum State DFA.

MINIMUM STATE DFA :- (Myhill Nerode Algo)

Every Regular language has a unique minimum state DFA.

The main objective is to minimize states of DFA. When in DFA, 2 or more states have same behaviour then we can merge them to a single state.

Let 2 states q_i & q_j are equivalent iff (if and only if)

(a) both q_i & q_j are either final states or non final states. (Necessary condⁿ)

(b) Now we have to check their transitions equivalency (sufficient condⁿ)

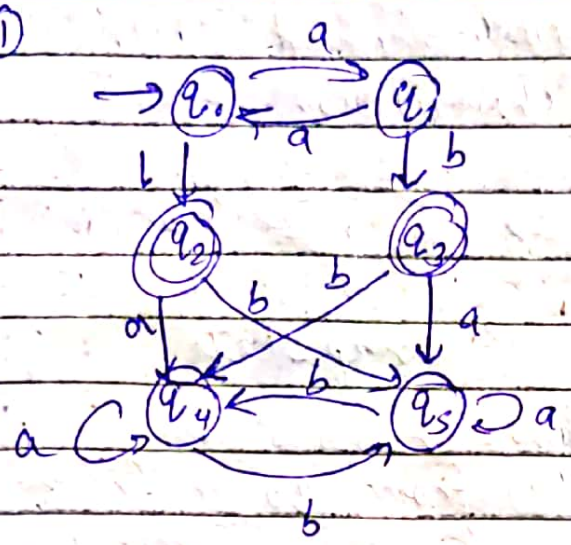
$\delta(q_i, a) \rightarrow$	F	NF ✓	F ✗	NF ✗	F: Final
$\delta(q_j, a) \rightarrow$	F	NF	NF	F	NF: Non Final

where $\forall\ a \in \Sigma^*$ all
↳ i/p's streams

Scanned with CamScanner

$^6C_2 = \frac{6 \times 5}{2} = 15$

ex ①



(i) Prepare a table we enter only asymmetric pairs

(ii) (a) ✓ cross F NF

(b)

→ last to 2nd

	q_5	q_4	q_3	q_2	q_1
q_0					$\{q_0 q_1\}$ ✓
q_1					
q_2			$\{q_0 q_1\}$		
q_3					
q_4	$\{q_2 q_3\}$ $\{q_4, q_5\}$ ✓				

First to 2nd last

Σ^* is verified automatically

$q_2 \xrightarrow{b} q_5$ NF $q_2 \xrightarrow{a} q_4$ NF

$q_3 \xrightarrow{b} q_4$ NF $q_3 \xrightarrow{a} q_5$ NF

No need to place entry again & again, chain directly in diagram

$q_4 \xrightarrow{a} q_4$ NF $q_4 \xrightarrow{b} q_4$

$q_5 \xrightarrow{a} q_5$ NF $q_5 \xrightarrow{b} q_5$

(No need to write (same)) if will come all src state

src dest

$q_0 \xrightarrow{a} q_1$ NF

$q_1 \xrightarrow{a} q_0$ NF

$q_0 \xrightarrow{b} q_2$ F

$q_1 \xrightarrow{b} q_3$ F

Σ^* not satisfied partially equivalent

put src in dest^n

$q_0 \xrightarrow{a} q_1$ NF ✓

$q_1 \xrightarrow{a} q_4$ NF

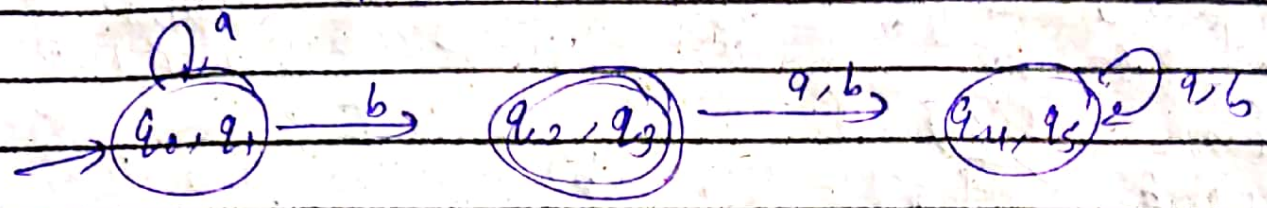
$q_0 \xrightarrow{b} q_2$ F ✗

$q_1 \xrightarrow{b} q_5$ NF

even fails on one :- failed

18.02.2023
Friday

where there is something write merged state as row & column characters.

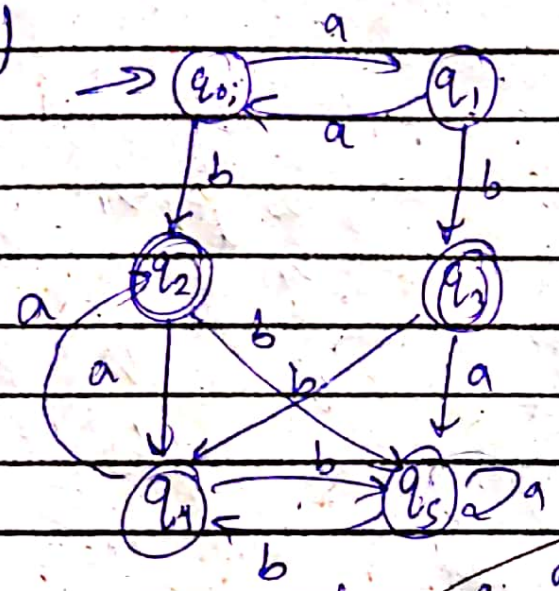


Trap state

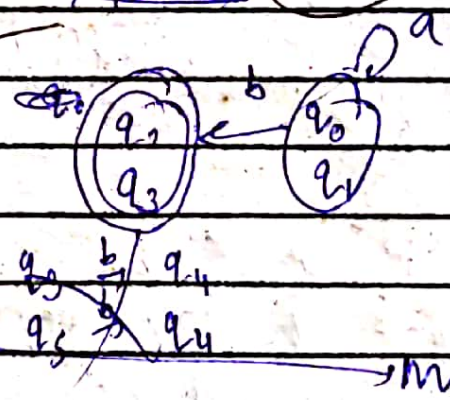
a^*b

See any transition either from q_0 or from q_1 for a & b

Q)



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



$q_0 \xrightarrow{a} q_1$
$q_1 \xrightarrow{a} q_0$
q_{01} q_{01}
$q_0 \xrightarrow{b} q_2$
$q_1 \xrightarrow{b} q_3$

$q_3 \xrightarrow{a} q_5$
$q_5 \xrightarrow{a} q_5$

$q_3 \xrightarrow{b} q_4$
$q_5 \xrightarrow{b} q_4$

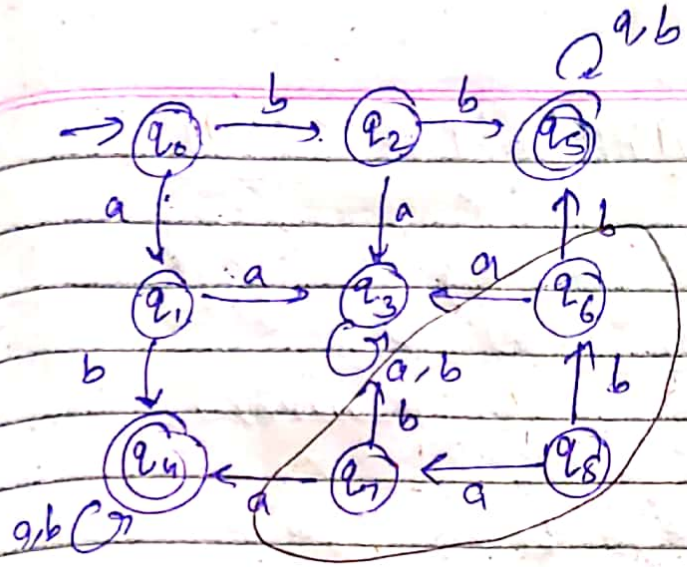
If there is entry & then we do cross & then also cross that entry where its ~~STE is~~ location is present & repeat

$q_0 \xrightarrow{b \text{ or } a} F$ q_2

$q_1 \xrightarrow{b \text{ or } a} NF$ q_5

Σ^* cond^n check automatically

Min. DFA only (only entry cross)



Dead State : A state which is not connected to initial state directly or indirectly is called dead state. Before minimization, identify all dead states & remove them

Ex: q_7, q_8, q_6 not equivalent

	q_5	q_4	q_3	q_2	q_1
q_0					
q_1			{0,3}		
q_2			{0,3}		
q_3					
q_4	{1,2}				
q_5	{4,5}				

4, 5 → final, put ✗ in others

$q_0 \xrightarrow{a} q_1$ NF ✓
$q_3 \xrightarrow{a} q_3$ NF
$q_0 \xrightarrow{b} q_2$ NF
$q_3 \xrightarrow{b} q_3$ NF

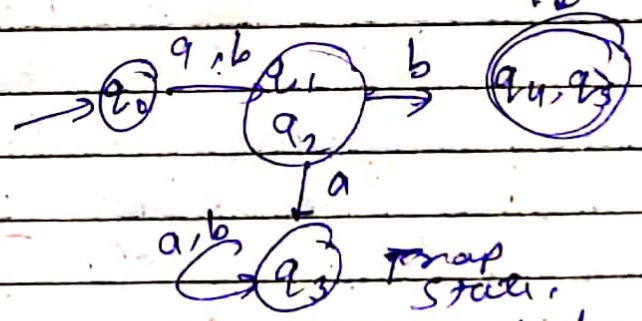
entry this location

$q_1 \xrightarrow{a} q_3$ NF ✓
$q_2 \xrightarrow{a} q_3$ NF ✓
$q_1 \xrightarrow{b} q_4$ F ✓
$q_2 \xrightarrow{b} q_5$ F ✓

There is no location Don't worry

$q_4 \xrightarrow{a,b} q_4$
$q_5 \xrightarrow{a,b} q_5$

Check all the unmarked state



Trap state.

Take states which were not merged.

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

↳ ~~with~~ All strings of a, b with 2nd symbol b.

FINITE AUTOMATA TO REGULAR EXPRESSION

Arden's Theorem :-

$$\boxed{R = P + R \cdot Q = PQ^*}$$

Proof

Step 1. $R = P + R \cdot Q$ → Recursive

Step 2. $R = P + (P + R \cdot Q)\, Q$

$\Rightarrow R = P + PQ + RQ^2$

Step 3. $\Rightarrow R = P + PQ\,(P + RQ)\, Q^2$

$\Rightarrow R = P + PQ^0 + PQ^2 + R \cdot Q^3$

$\vdots$

Step n $\Rightarrow R = P + PQ + PQ^2 + \cdots + PQ^{n-1} + RQ^n$

Step ∞ $\Rightarrow R = P + PQ + PQ^2 + \cdots + PQ^n + \cdots + PQ^{\infty-1} + RQ^{\infty}$

$= P(\epsilon + Q + Q^2 + \cdots)$

$\Rightarrow R = PQ^*$

RQ^{∞} ↓ discard this as going to infinite

Finite Automata to RE

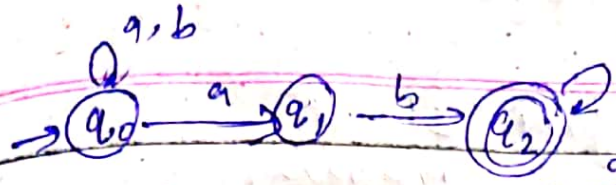
Step 1. Find the equation of each state in terms of all incoming transitions to it.

Step 2. Apply Arden's theorem & find the ~~eqn~~ regular expression of each state.

Step 3. $RE = \sum q_i$, $\forall\, q_i \in F$

Scanned with CamScanner

Don forget 'ε' transition initial state



It's NFA

$\therefore q_0 \xrightarrow{a} q_0$ all the strings of a, b have substring "ab".

Step 1 $q_0 = \epsilon + q_0 a + q_0 b$

$\Rightarrow q_0 = \epsilon + q_0(a+b)$ (distributive law) → (I)

$q_1 = q_0 a$ $q_1 \xrightarrow{b} q_2$, $q_2 \xrightarrow{a} q_2$ → (II)

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

$\Rightarrow q_2 = q_1 b + q_2(a+b)$ → (III)

Step 2 (II) → can't apply Arden's theorem as form not matches.

(I) & (III):

Let do on (III):

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

$R = P + R.Q$

Arden's Theorem: $R = PQ^*$

$\therefore q_2 = q_1 b (a+b)^*$ → (IV)

Arden's Theorem (I):

$q_0 = \epsilon + q_0(a+b)$

$\Rightarrow q_0 = \epsilon (a+b)^*$

$q_0 = (a+b)^*$

Substitute in (V)

Use eqn (II) in (IV)

$q_2 = q_0 a b (a+b)^*$ → (V)

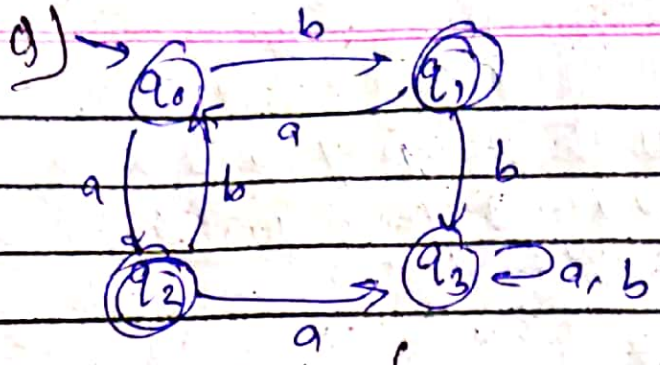
$\Rightarrow q_2 = (a+b)^* ab (a+b)^*$

NOTE

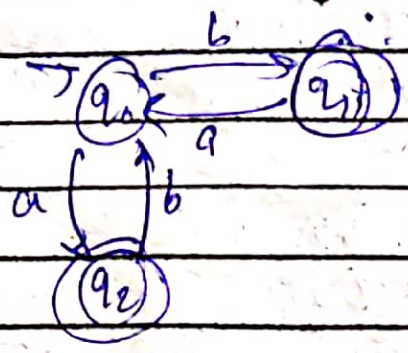
By you applying Arden's theorem, Remove all dead & dummy (Trap) states from DFA.

We can't come back, No need to use equation

9)



Before start
renaming
q_3 as trap
state
(dummy state)



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

$q_1 = q_0 b$

$q_2 = q_0 a$

$\Rightarrow \cancel{q_1 = (\epsilon + q_1 a + q_2 b)}$

Can't apply Arden's theorem as any

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

$R = P + RQ$ $\quad q_0 = \epsilon + q_0 (ba + ab)$

$\Rightarrow R = PQ^*$ Arden's theorem $\quad q_0 = \epsilon (ba + ab)^*$

$\Rightarrow q_0 = (ba + ab)^*$

Final Regular expression = Σ final states

$= q_1 + q_2$

$= q_0 b + q_0 a$

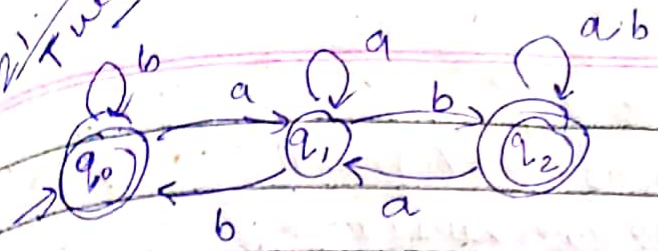
$= (ba + ab)^* b + (ba + ab)^* a$

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

loop of either ab or ba

end that with either a or b

21.02.2023
Tues day



$q_0 = \epsilon + q_0 b + q_1 b \rightarrow$ (I)

$q_1 = q_1 a + q_0 a + q_2 a \rightarrow$ (II)

$q_2 = q_1 b + q_2 (a+b) \rightarrow$ (III)

AT on (I)

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

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

$\Rightarrow q_0 = b^* + q_1 b b^* \rightarrow$ (IV)

AT on (III)

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

$\Rightarrow q_2 = \cancel{ab q_1 + b^2 q_1}$

$q_1 b (a+b)^* \rightarrow$ (V)

[AT $R = P + RQ$
$\Rightarrow R = PQ$]

put in (II):

$\cancel{q_1 = q_1 a + b + q_1 b^2 + ab q_1 + b^2 q_1}$

$\cancel{\Rightarrow q_1 = b + (a + 2b^2 + ab) q_1}$

$\cancel{\Rightarrow q_1 = b (a + ab + 2b^2)}$

[AT]

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

$= q_1 a + ab^* + abb^* q_1 + ab q_1 (a+b)^*$

$\Rightarrow q_1 = ab^* + (a + abb^* + ab(a+b)^*) q_1$

$\Rightarrow q_1 = ab^* (a + abb^* + ab(a+b)^*)^*$ [AT]

put in I

$q_0 = b^* + ab^* ((a + abb^* + ab(a+b)^*)^*$

RE $= q_0 + q_2$ [PS]

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

$= b^* + q_1 (bb^* + b(a+b)^*)$

$= b^* + [b^* + ab^* ((a + abb^* + ab(a+b)^*)^*) (bb^* + b(a+b)^*)]$

capital letters

REGULAR GRAMMAR

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

where V = set of finite variables.

T = set of finite terminals → small letters

S = starting variable. $S \in V$

P = set of finite productions

There are 2 types of Regular Grammar

P

Left linear Regular	Right linear Regular
derivation will grow in left dir^n	derivation will grow in right direction
$A \to Ba$	$A \to aB$
or $A \to a$	or $A \to a$

Derivation :- set of steps to generate a string from grammar is called derivation

NOTE: where $A, B \in V$ always capital as Variables (growing)

& $a \in T^*$ & small as terminal (end)

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

Ans: $S \to aS / bS / \epsilon$ → Termination and^n

Easy to design RLR than LLR

$S \to aS$ or $S \to bS$

$\to a(aS)$ $\to bbS$

$\to aa(aS)$

$\to aaaS$

→ RLR

Minimum string possible

There are multiple ways to [illegible] grammar

$S \to Sa / Sb / \epsilon$ (left linear)

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

Right linear $\longleftrightarrow$ left linear

$S \to aS / A$ (number of a's) $\quad S \to aS / bS$

non-terminal $\quad$ why A? $\to$ ~~ab~~

$A \to bS / \epsilon$

↳ min. Termination pt.

$(S \to A \to \epsilon)$

left linear $\quad \times \left[\begin{array}{l} S \to Sa / A \\ A \to Ab / \epsilon \end{array} \right] \times$

↳ left dir^n growing $\quad S \to Sb / a$

$A \to Aa / \epsilon$

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

RLR $\quad S \to aS / aaA \quad \quad S \to aaS$

↳ min a' cond^n $\quad$ even no. of a's

$A \to bS / b$

↳ min cond^n for b

Non-y b!

LLR $\quad S \to Sb / aAb$

$A \to Aa / aa$

Another way

$S \to aS / aabA$

$A \to aA / \epsilon$

(4) $L = \{a^n b^m \mid n \geq 1, m \leq 3\}$

RLR $\quad S \to aS / aA$

$A \to bbb / bb / b / \epsilon$

LLR $\quad S \to Abbb / Abb / Ab / aA$

$A \to Aa / a$

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

RLR $S \to aaS \mid A$ even no. of a's

$A \to bbA \mid b$ odd no. of b's

$\to$ even no. of b + 1

LLR $S \to Sbb \mid Ab$

$A \to Aaa \mid \epsilon$

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

RLR $S \to aaS \mid aaX \mid aaY \mid Z$

$X \to Z \to bbZ \mid \epsilon$

$Y \to bbY \mid b$

LLR $S \to \cancel{Saa \mid Ya \mid Z}$ $Sbb \mid Yb \mid Z$

$Y \to \cancel{Ybb} \mid \cancel{b}\, aa$

$Z \to Z\cancel{bb} \mid \epsilon$

$S \to S_1 \mid S_2 S_1$

$S_1 \to aa\cancel{X} \mid \cancel{X}Y$ $S_2 \to aa\cancel{X} \mid ab\cancel{X}Y$

$Y \to bbY \mid \epsilon$

even even odd odd

$S \to S_1 \mid S_2$ $S_1 \to S_1bb \mid A$ $S_2 \to S_2bb \mid Aab$

$A \to Aaa \mid \epsilon$

Scanned with CamScanner

3.02.2023

If not mention then take binary

DFA & NFA differentiate on the basis of Transition fn

FA (Finite Automata)

without o/p → DFA (5 tuples) → $Q \times \Sigma \to Q$

M/c → Machine

with o/p → NFA → $Q \times \Sigma \to 2^Q$

with o/p → Moore M/c → $Q \to \Delta$; Mealy M/c → $Q \times \Sigma \to \Delta$

differentiate on the basis of o/p transition fn

o/p transition fn

6 tuple

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

Q → set of states

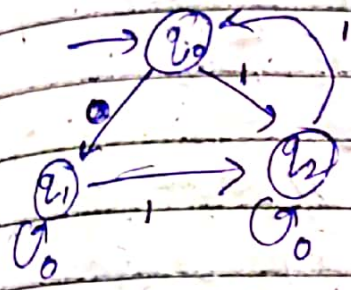
Σ → finite set of I/p

q_0 → initial state

δ → $Q \times \Sigma \to Q$ same as DFA

Δ → set of o/p symbol

Δ → o/p transition fn

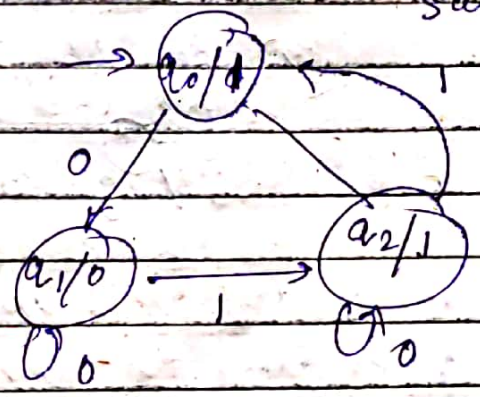


Not any machine

No final state

Not M & M M/c

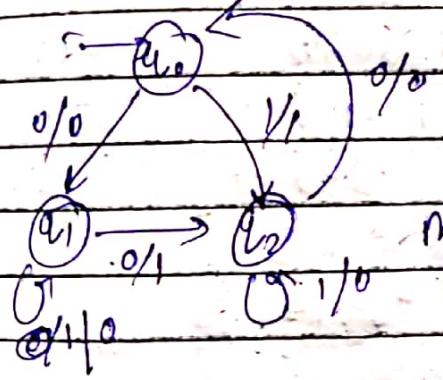
No o/p symbol



$Q \to \Delta$

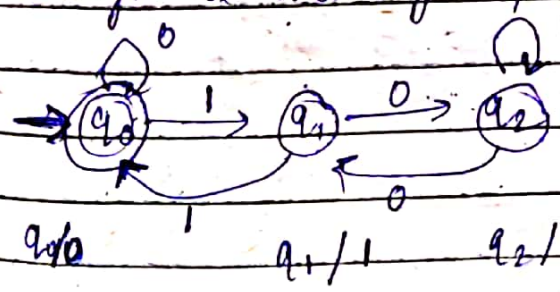
Moore M/c

State q2 o/p



Mealy M/c

Q) Design a Moore m/c which is divisible by 3



DFA

10 = 2

11 = 0

100 = 1

101

To convert into Moore m/c remove "finite state" & put remainder in state

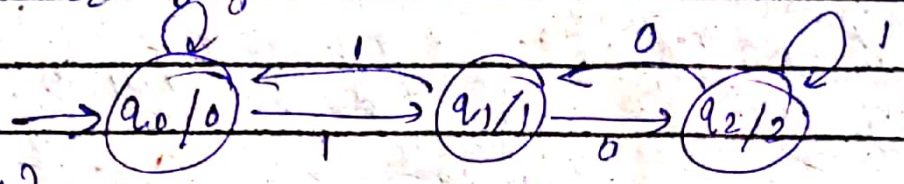
Scanned with CamScanner

Trick cyclic 012 for 4 0123

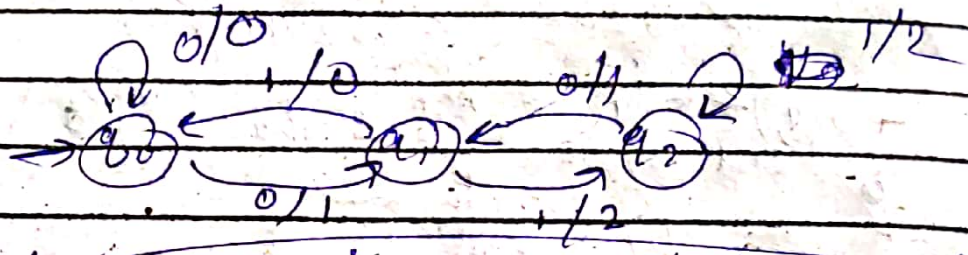
	0	1
q_0	q_0	q_1
q_1	q_2	q_0
q_2	q_1	q_2

} construct graph & get DFA

Moore M/c (inside state)



Mealy M/c (on transition)

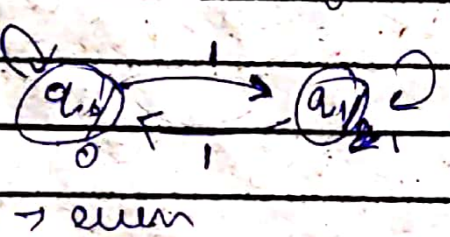


Method: put Moore's state o/p symbol on incoming o/p symbol in transition

Trick

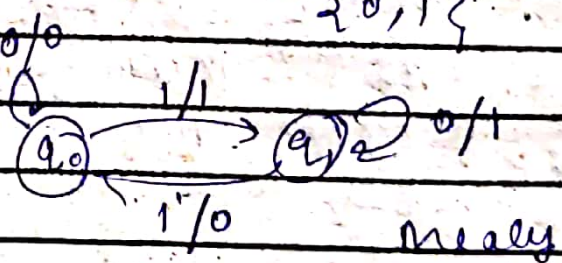
check 110 = 6 6 / 3 = 20

Q) Design a Mealy m/c which produce o/p even & odd according to number of 1 i/p symbol over {0,1}.



0 → even

1 → odd



Mealy m/c

m state n o/p Moore → m state Mealy

m state n o/p Mealy → m×n state Moore

27.02.2023
Monday

Why Right linear is easy? Right in dir^n of thinking

Q) All the strings of 'ab' have substring aab

Sol. $X = aX / bX / aabY$
$Y = aY / bY / \epsilon$

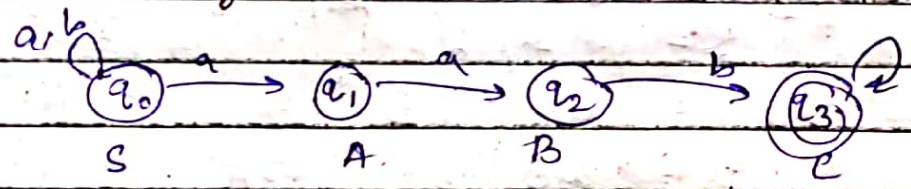
for such questions find DFA / NFA and then convert it into Regular Grammar

FA to Regular Grammar

(a) Right linear :-

① Assign a variable name to each state of FA (DFA or NFA any) where starting state by starting variable S.

② For each transition $\delta(a_i, a) \rightarrow a_j$, write a production $A \rightarrow aB$ where A & B are variables corresponding to states, a_i & a_j respectively.



Just look ~~any~~ outgoing transitions

$S \rightarrow aS / bS / aA$
$A \rightarrow aB$
$B \rightarrow bC$ $\quad C \rightarrow aC / bC / \epsilon$

③ For each final state variable, add an extra production 'ϵ'

(b) Left linear :- (just reverse)

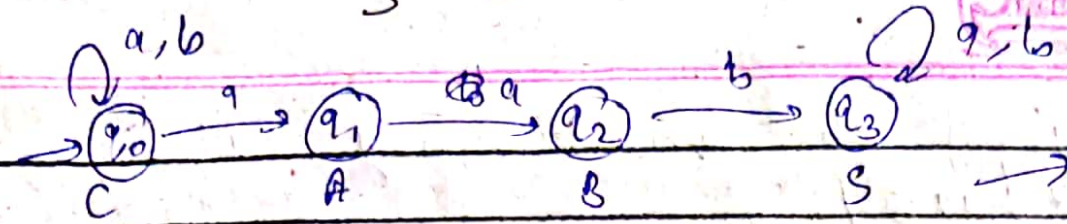
① Assign a variable name to each state of FA where final state by starting variable S.

② For each transition $\delta(a_i, a) \rightarrow a_j$
$B \rightarrow Aa$

② For initial state variable

That's a nomenclature that we give starting variable S ~~but~~

NOTE: don't forget to write ending p...



→ we can change names

Transition look at incoming ~~trac~~ & write "Production" or transition

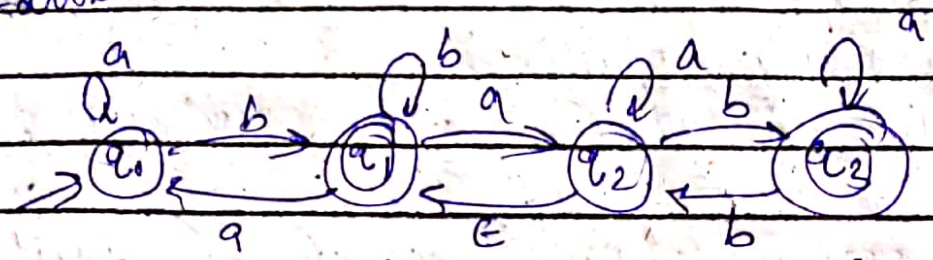
$S = Sa / Sb / Bb$

$A = Ca$

$B = Aa$

$C = Ca / Cb / \epsilon$ → Initial

Q)



Sol

	S	A	B	C	Right
	A	S_2	B	S_1	left

Right linear

$S = aS / bA$

$A = bA / aS / aB$ → $/\epsilon$ → A

$B = aB / bC / \epsilon A$

$C = aC / bB / \epsilon$

$\boxed{\epsilon A = A}$ Note

when q_2 is final state

Left linear

Starting variable always unique

use concept $S = S_1 / S_2 / \ldots$

~~$S = Sa / Aa / \epsilon$~~ Start from final

$S = S_1 / S_2$

$S_1 = S_1 a / Bb$

$S_2 = S_2 b / Ab / B\epsilon$

$A = Aa / S_2 a / \epsilon$

$B = S_1 b / S_2 a / Ba$

Assign ϵ to starting state initial

(Final state [illegible] [illegible])

Regular Grammar to FA :-

(a) Right Linear :-

① The variables of grammar represents states of FA where starting variable ~~by~~ represents initial state

② All the productions must be in the form $A \to aB$ or $A \to \epsilon$
where $A \to aB$ or $A \to \epsilon$

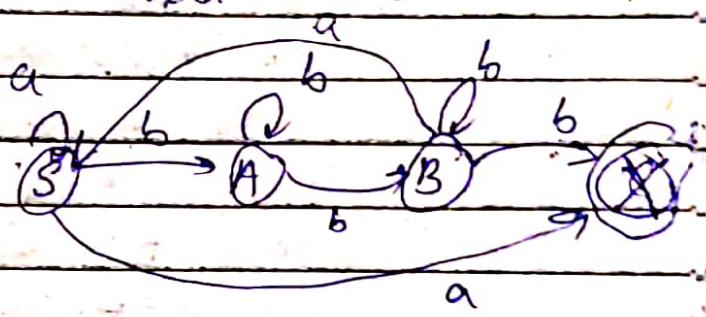
③ For each production $A \to aB$ add a transition $\delta(A, a) \to B$ in FA

④ Final states all the variables which have 'ε' production

Q) $S \to aS / bA / a$
$A \to bA / bB$
$B \to aS / bB / b$

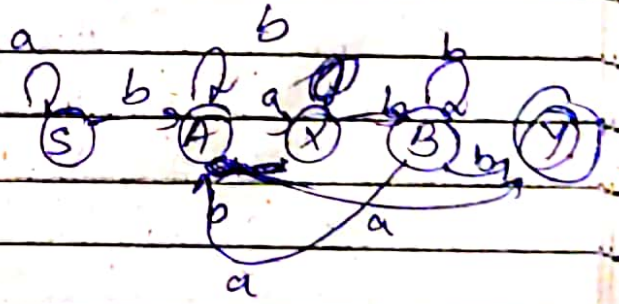
Sol: All transition not ~~have~~ are in desired form
$A \to aB$
or $A \to \epsilon$
Add extra variable

Sol: $S \to aS / bA / aX$
$A \to bA / bB$
$B \to aS / bB / bX$
$X \to \epsilon$
↳ Null a/(n) final



Q) $S \to aS / bA$
$A \to bA / abB / a$
$B \to bB / aA / b$
Correction required

Sol: $S \to aS / bA$
$A \to bA / aX / aY$
$X \to \cancel{bA} / bB$
$B \to bB / aA / bY$
$Y \to \epsilon \longrightarrow$ Final State



Scanned with CamScanner

RE → FA
FA → RE
RG → FA
FA → RG

b) Left linear :-

① The states of FA are represented by variables where starting variable represents final state

② All the productions must be in the form
$A \to Ba$ where $A, B \in V$ &
or $A \to \epsilon$ $a \in T$

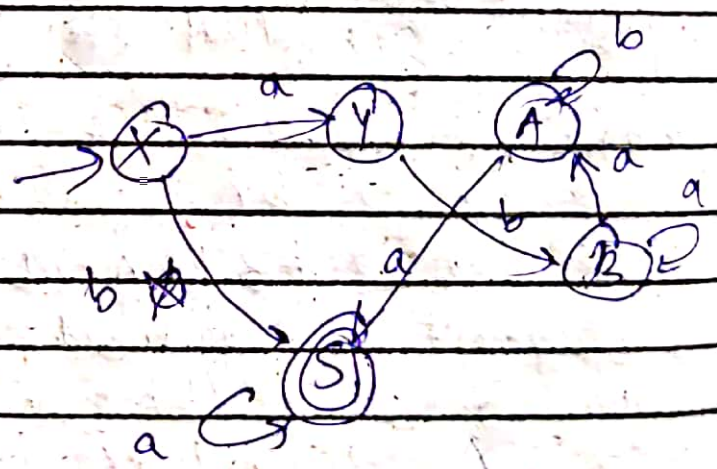
③ For each production $A \to Ba$ add a transition
$\delta(B, a) \to A$

④ Initial State : Variable with 'ε' production.
Thus can be many ε but change them to one variable as there is only 1 initial state possible :-

Eg. $S \to Sa / Aa / b$
$A \to Ab / Ba$
$B \to Ba / ab$

Sol. $S \to Sa / Aa / b$
$A \to Ab / Ba$
$B \to Ba / \underbrace{Xa}_{Y} b$

$S \to Sa / Aa / Xb$
$A \to Ab / Ba$
$B \to Ba / Yb$
$Y \to Xa$
$X \to \epsilon$ start with ε production



01.03.2023
Wednesday

(any one) NFA

What is Regular language?

// DFA

* A language for which there exists an FA that accepts all strings of it called Regular language.
* A language for which there exists a regular expression (RE) that generates all the strings of it called Regular language.
* A language for which there exists a regular grammar that generates all the strings of it called Regular language.

Pumping lemma : It is used to prove whether a language is Regular or not.
(We will not use now)

$L = \{a^n b^n \mid n \geq 1\}$ Not a Regular language

Non Regular languages :-

* FA is a memory less machine (no external memory (has internal memory) in it) so we can not compare two or more variable with it.
* Every finite language is Regular.
(But that does not means every infinite language is not regular)

Check whether they are R / NR → either draw FA / write RE / write RG

① $L = \{a^n b^n \mid n \geq 1\}$ NR equal a & b but can't store. (memoryless)

② $L = \{a^n b^m \mid n \neq m\}$ NR can't compare

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

compare a & b with c but memoryless

(4) $L = \{a^n b^m \mid n = m^2\}$ NR

comparision means NR

(5) $L = \{a^n b^m \mid (n+m) \geq 2^{1000000}\}$ R

compare with constant

$L = \{a^n b^m \mid (n+m) \geq 5\}$

$a^* bbbbbb^* + aa^* bbbbb^* + \dots + aaaaa a^* b^*$

all combinations

(6) $L = \{a^n b^n c^n \mid n \geq 0\}$ NR compare

(7) $L = \{w w^r \mid w \in \{a,b\}^*\}$ where w^r is reverse of w

Even length palindrome

NR → we need to store w to get reverse w

(8) $L = \{w c w^r \mid w \in \{a,b\}^*$ & c is a terminal$\}$

Odd length palindrome

NR

(9) $L = \{w_1 c w_2 \mid w_1, w_2 \in \{a,b\}^*, c$ is a terminal$\}$ R

(10) $L = \{w c w \mid w \in \{a,b\}^*$ & c is a terminal$\}$

(11) $L = \{ww \mid w \in \{a,b\}^*\}$ NR

Ensuring same copy need to store it

(12) $L = \{a^n b^n c^n \mid n \leq 2^{1000000}\}$

$\epsilon + abc + aabbcc + \dots + a^{2^{1000000}} b^{2^{1000000}} c^{2^{1000000}}$ R

↳ finite

(13) $L = \{a^p \mid p$ is prime$\}$ NR

algo to know prime & store

(14) $L = \{w x w^r \mid w, x \in \{a,b\}^+\}$ R

(15) $L = \{u w w^r v \mid u, v, w \in \{a,b\}^+\}$ R

Scanned with CamScanner

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

ab ba

Scanned with CamScanner

Scanned with CamScanner

04.08.2022
Tuesday

Central Tendency

Mean
Median
Mode

Median

(I) Raw form

(1) Increasing / Decreasing order

(2) No. of element.

(3) Even → Middle two elements
Odd → Middle element

(II)

x	f	cf
1	5	5
2	7	12
3	9	21
5	11	32
7	7	39

$\frac{39}{2} =$

$N/2$

(III)

x	f	cf
1-10	5	5
10-20	6	11
20-30	10	21
30-40	12	33
40-50	9	42

cf

f

$$m = l + \left(\frac{\frac{N}{2} - cf}{f}\right) \times h$$

Class size

$N/2 = 21$

21.2 just greater

CONTEXT FREE LANGUAGE (CFL)

Content Free Grammar

CFG (V, T, S, P)

V: set of finite variables or non terminals

T: set of terminals.

S: Starting variable $S \in V$

P: set of finite productions

$\forall P: A \rightarrow \alpha$

where $A \in V$ & $\alpha \in (V \cup T)^*$

① $L = \{a^n b^n \mid n \geq 0\}$

Sol: $S \rightarrow aSb \mid \epsilon$

same no of a & same no of b

$a^n b^n$

S

aSb

$aasbb$

$aaaSbbb$

aSb

Left most a Right most b

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

Sol.

$S \rightarrow aSc \mid aAc$ → min. cond^n

$A \rightarrow bA \mid b$

or $S \rightarrow bc \mid M$

③ $L = \{a^n b^n c^m \mid n, m \geq 0\}$

Sol. $S \rightarrow aSb \mid A$

$A \rightarrow cA \mid \epsilon$

or

$S \rightarrow A/B$

$A \rightarrow aAb/\epsilon$

$B \rightarrow cB/\epsilon$

Scanned with CamScanner

21.03.2023 Tuesday (8)

2 = = 4 = = 6
3 = = 5 = = 7

(4) $L = \{a^n b^m c^m d^n \mid m \geq 1 ; n \geq 0\}$

$S \to aSd \mid A$

$A \to bAc \mid bc$

3142

bAc

ϵ

(5) $L = \{a^n b^n c^m d^m \mid m \geq 0, n \geq 1\}$

$S = A \mid B$

$A = aAb \mid ab$

$B = cBd \mid \epsilon$

(6) $L = \{a^n b^m c^{m+n} \mid m, n > 0\}$

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

$S = A \mid B$

$A = bAc \mid$

$S = bSc \mid B$

$B = aBc \mid \epsilon$

$S \to aSc \mid B$

$B \to bBc \mid \epsilon$

(7) $L = \{a^n b^{m+n} c^m \mid m, n \geq 1\}$

$S =$

$S \to$ division do this

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

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

$a^{n+m} b^{n+m}$

can't go ahead

$S \to S_1 \mid S_2 \mid ab$

$S_1 \to aS_1b \mid \epsilon$

$S_2 \to cS_2d \mid \epsilon$

$S = aA \mid B$

$A \to aAb \mid \epsilon$

$B \to bBc \mid \epsilon$

(8) $L = \{a^n b^m \mid n, m > 0\}$

$n \neq m$

$S = S_1 \mid S_2$

$S_1 = aS_1b \mid aA$

$A = aA \mid \epsilon$

$S_2 = aS_2b \mid bB$

$B = bB \mid \epsilon$

$S \to aSb \mid A \mid B$

$A \to aA \mid a \to$ (I)

$B \to bB \mid b \to$ (II)

$n \neq m$

$n > m$ (I) $\quad n < m$ (II)

Scanned with CamScanner

even length pallindrome

(9) $L = \{ w \cdot w^r \mid w \in \{a,b\}^* \}$ w_r is the reverse of w

$S \to aSb$, $aSb \to aaSbb$, $aaSbb \to aaaSbbb$

$S = aSa \mid bSb \mid \epsilon$

(10) $L = \{ wcw^r \mid w \in \{a,b\}^* \}$ c is a terminal

$S = aSa \mid bSb \mid \cancel{a \mid b}\ c$

odd length pallindrome

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

$S = aS_1 \mid bS_2$

$S \to aSb \mid bSa \mid \epsilon$ × check abba

$S_1 = bS$ ✓ $S \to aSbS \mid bSaS \mid \epsilon$

Derivations

To generate the string from grammar set of steps to generate a string from grammar is called Derivation

There is ~~only~~ atmost single variable in Regular grammar but not in CFG.

Types of derivations

LMD
(Left Most Derivations)
In each step of derivation left most variable is replaced by its production

RMD
(Right Most Derivation)
In each step of derivation right most variable is replaced by its ~~derivation~~ production

Scanned with CamScanner

If $S \rightarrow aSbS / bSaS / \epsilon$ is a CFG. Generate string aabbba or abba, can be generated

Step to generate LMD is called left sentential

LMD

Steps	left sentential form (always start with S)	Productions (see arrow of string)
1.	$\underline{S}$	$S \rightarrow aSbS$
2.	$a\underline{S}bS$	$S \rightarrow aSbS$
3.	$aa\underline{S}bSbS$	$S \rightarrow \epsilon$
4.	$aab\underline{S}bS$	$S \rightarrow \epsilon$
5.	$aabb\underline{S}$	$S \rightarrow bSaS$
6.	$aabbb\underline{S}aS$	$S \rightarrow \epsilon$
7.	$aabbba\underline{S}$	$S \rightarrow \epsilon$
8.	$aabbba$	–

RMD

Steps	~~left~~ right sentential form	Productions (see at each step which production to choose greedily)
1.	$\underline{S}$	$S \rightarrow aSbS$
2.	$aSb\underline{S}$	$S \rightarrow bSaS$
3.	$aSbb\underline{S}a\underline{S}$	$S \rightarrow \epsilon$
4.	$aSbb\underline{S}a$	$S \rightarrow \epsilon$
5.	$a\underline{S}bba$	$S \rightarrow aSbS$
6.	$aaSb\underline{S}bba$	$S \rightarrow \epsilon$
7.	$aa\underline{S}bbba$	$S \rightarrow \epsilon$
8.	$aabbba$	–

Scanned with CamScanner

Derivation tree or Parse tree :-

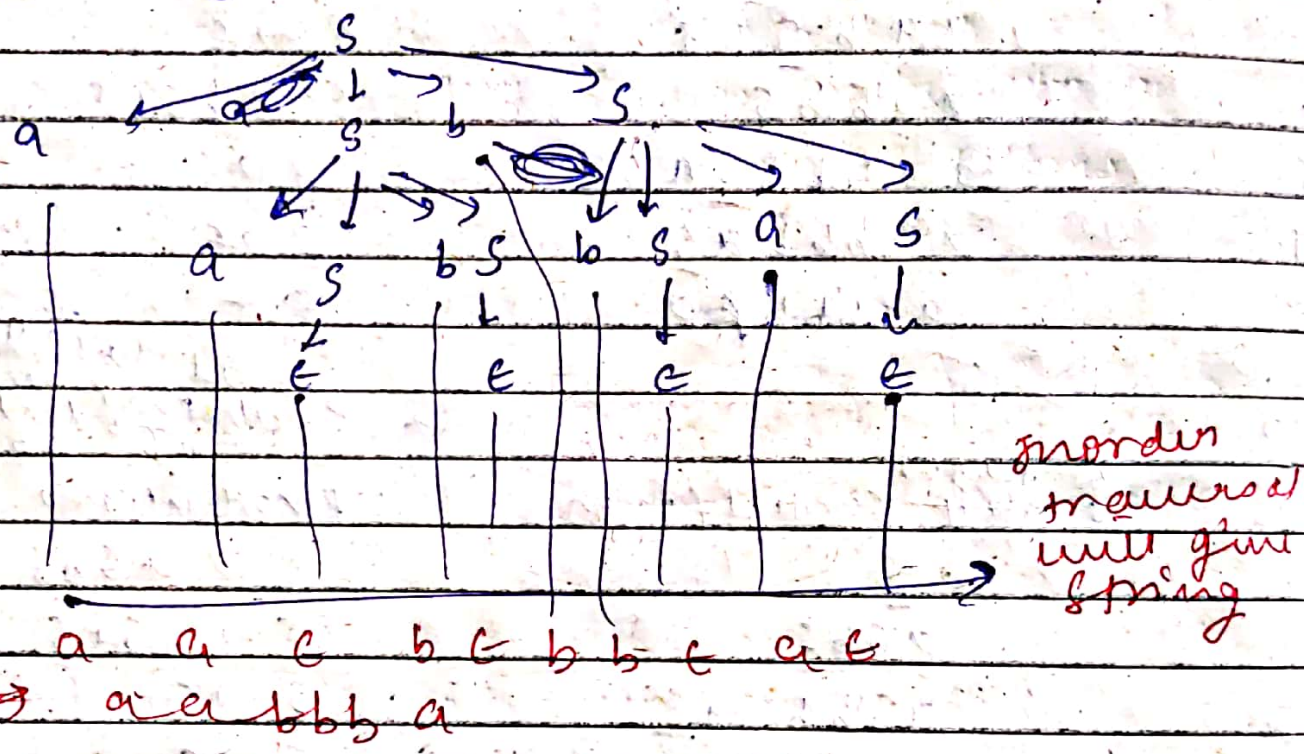
It is a graphical way in the form of tree to represent derivation of string from the grammar.

① The root of the ~~string~~ tree is starting variable

② All the variables represent non leaves node while all terminals represent leaves.

Q) $S \to aSbS \mid bSaS \mid \epsilon$ Represent aabbba using parse tree

Sol Start with S



Ambiguous Grammar :

A grammar is called ambiguous when there exists more than one LMDs, RMDs or Parse trees for any string of the grammar

Best way to find Parse Tree. → Graphical method

Q) $S \to aSbS \mid bSaS \mid \epsilon$

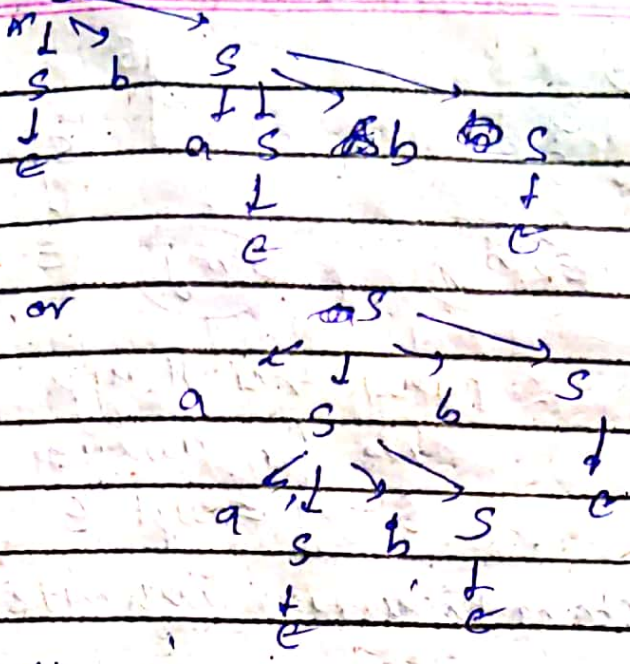
abab

↓

smallest string of above grammar

or

bab a.



parse 2 trees

∴ ~~Above~~ grammar is ambiguous.

Inherently Ambiguous Grammar:

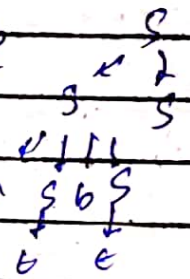
A grammar is called ambiguous when there exists more than one LMDs, RMDs or Parse trees for every string of the grammar.

$S \to aSbS \mid bSaS \mid \epsilon$ is

Not inherently ambiguous as ϵ has only one Parse tree & also example on previous page.

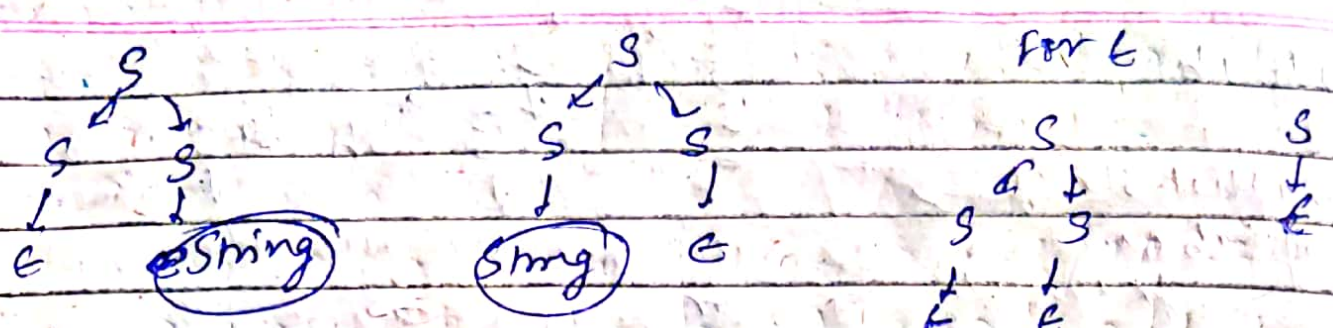
* Grammar $S \to SS \mid aSbS \mid bSaS \mid \epsilon$ is inherently ambiguous → language is S and but wrong to represent grammar is different

ab



even for ab, ba, & ϵ there are multiple parse trees.

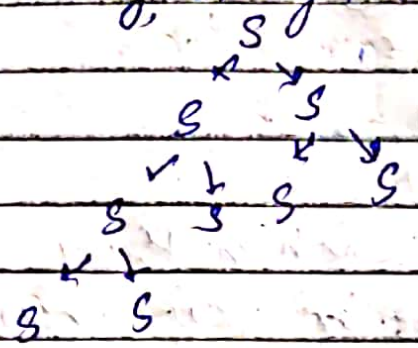
23.03.2023
Thursday.



2 parse tree of type this for every string except ε

2 for ε

* for any string there are infinite no. of parse trees



any S can make string & make ε others

Ambiguous

Inherently Ambiguous

Simplication of CFG

(i) Remove 'ε' productions
(ii) Remove unit productions
(iii) Remove useless production

(i) **Remove 'ε' productions:-**

1. Identify variables which are producing 'ε'
2. Write CFG without ε production
3. Substitute variable 'ε' one at a time & generate

new production

(i) $S \to aSb / \epsilon$ $\Rightarrow$ $S \to aSb / ab$

(ii) $S \to aSAb / Ab$
$A \to aAb / \epsilon$
$B \to bB / \epsilon$

$\to$ $S \to aSAb / AbSb / Ab / b$
$A \to aAb / ab$
$B \to bB / b$

(iii) $S \to aACb / bBCa$
$A \to aA / \epsilon$
$B \to bB / \epsilon$
$C \to AB / d$

$\to$ $S \to aACb / bBCa / aCb / bCa$
$A \to aA / a$
$B \to bB / b$
$C \to AB / d / B / A / \epsilon$

Now C produces ϵ ↓ put $A = \epsilon$ in eqn & make production. Similarly for B

$S \to aACb / bBCa / aCb / bCa / aAb / bBa / ab / ba$
$A \to aA / a$
$B \to bB / b$
$C \to AB / d / B / A$

(II) Removal of unit productions ~~of the form $A \to B$ where $A, B \in V$~~ $\boxed{\text{Unit Production } A \to B}$

(i) Identify all the unit production of the form $A \to B$ where $A, B \in V$

(ii) For each unit production of the form $A \to B$ do substitute all the productions of the variable B in variable A

Productions of type
$A \to A$
can be removed
neglected
Meaningless production → unit production → when variable defines variable not small

(i) $S \to aSb / A$
$A \to aA / b$] → $S \to aSb / aA / b$
$A \to aA / b$

(ii) $S \to aSb / A$
$A \to aA / B / b$
$B \to bB / d$] → $S \to aSb / aA / bB / d / b$
$A \to aA / bB / d / b$
$B \to bB / d$

or one at a time

Replacing A
$S \to aSb / aA / B / b$
$A \to aA / B / b$
$B \to bB / d$ → $S \to aSb / aA / bB / d / b$
$A \to aA / bB / d / b$
Replacing B → $B \to bB / d$

(iii) In some then can be loop.
$S \to aSb / A$
$A \to aA / B / b$
$B \to bB / A / d$ $\xrightarrow{A}$ $S \to aSb / aA / B / b$
$A \to aA / bB / A / d / b$
$B \to bB / aA / bB / B / b$
↳ can neglect on same production

$S \to aSb / aA / bB / b / d$
$A \to aA / bB / d / b$
$B \to bB / aA / b / d$
$A \to A$, $B \to B$ } neglected

∴ We can remove variable as productions are same.
but careful
change any production of A & B.

Check in any order that steps

(iii) Removal of useless productions :-

(a) All the variables must be ~~terminal~~, attached with starting variable directly or indirectly.

(b) All the variables must be terminated directly / indirectly.

(i) $S \to aA \mid b$
$A \to bB$
$B \to Bd \mid d$

Variable A is directly attached to S while variable B is indirectly via A.
Variable S & B are directly terminating but A is termination via B.
∴ All variables are attached directly / indirectly to S
A, B, S → are terminating directly / indirectly
But C is non Terminating (Self loop) $C \to \alpha C \gamma$ type

(ii) $S \to aSAb \mid Bd$
$A \to bCd \mid b$
$B \to bB \mid a$
$C \to aCd \mid bBCa$

Remove all production involving C

$S \to aSAb \mid Bd$
$A \to b$
$B \to bB \mid a$

Final grammar

(iii) $S \to aASd \mid bD \mid a$
$A \to aAb \mid bB$
$B \to bBa \mid aD$
$C \to aAb \mid bB \mid C$
$D \to bB \mid aA \mid Sd$

$C \to A \; B \; C \; D$
∴ C is not directly / indirectly connected to S
Remove C

	S	A	B	C	D
Terminates	✓	✓	✓	✗	✓

Final grammar without C

$S \to A \to D \to B$

All connected to S

(iv) $S \to aAd \mid bBa$

$A \to aA \mid bD \mid d$

$B \to bD \mid aBA$

$D \to aB \mid bDS$

*	S	A	B	D
	Terminate ✓	✓	$B \to B$	$D \to D$

For terminating

Order of Simplication

(i) Remove Null

(ii) Remove Unit Production

(iii) Remove Useless

} He of first look 2nd letter

$n < S$: n (2) S (3)

Normal forms of CFG :-

B your converting CFG to any of the normal form first simplify it.

There are 2 types of normal form of CFG :-

(i) Chomsky Normal Form (CNF)

(ii) Griebach Normal Form (GNF)

(i) CNF

$\forall P$: (Productions)

$A \to BC$ (2 variables)

or $A \to a$ (1 terminal)

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

eg $S \to AS \mid b$

$A \to SA \mid a$

Scanned with CamScanner

First simplify
(i) Remove Null

eg $S \to aSb / \epsilon$

$S \to aSb / ab$

If production is in CNF then leave it as it is

do process

आगे और Termination
- Two ~~step~~
$a \to A$
$b \to B$ (new variable)

$S \to AS B / AB$
$A \to a$
$B \to b$

$SB = X$

$S \to AX / AB$
$A \to A$
$B \to b$
$X \to SB$

$AS = X$

$S \to XB / AB$
$A \to a$
$B \to b$
$X \to AS$

Both are correct

eg $S \to aAb / bS$
$A \to bA / \epsilon$
$B \to aS / A$
$C \to aA / bB / d$

Remove 'ε' →

$S \to aAb / bS$
$A \to bA / b$
$B \to aS / A$
$C \to aA / bB / d$

Remove unit product

$B \to A$

CNF
$A \to a$
$b \to A$

S	A	B	C
✓	✗	✗	✗

$S \to aAb / bS$
$A \to bA / b$
$B \to aS / bA / b$
$C \to aA / bB / d$

any one to

$S \to aAb / bS$
$A \to bA / X$
$X \to b$

X Y (fresh new variable)

$S \to XAY / YS$
$A \to YA / X$
$X \to a$
$Y \to b$

$XA = Z$

$AY = Z$

$S \to ZY / YS$
$A \to YA / X$
$X \to a$
$Y \to b$

$\to XA$

or

$S \to XZ / YS$
$A \to YA / X$
$X \to a$
$Y \to b$
$Z \to AY$

Scanned with CamScanner

~~CFG~~ GNF

$\forall P: \quad A \to a\alpha \quad \to V^* = \{\epsilon, \alpha_1, \alpha_1\alpha_2, \alpha_1\alpha_2\alpha_3$

where $\alpha \in V^*$

$a \in T$

ex $S \to aAb / \epsilon$ ⟶ Remove 'ϵ'

$\downarrow$ $S \to aAb / ab$

$\downarrow$ $S \to aAB / aB$ $\quad B \to bB^0$

$B \to b$ ✓ ✓ $\quad \downarrow \epsilon$

Q) $S \to AB / aSb$

$A \to Bd / b$

$B \to bB / \epsilon$ ⟶ 'ϵ' Remove

Sol: Simplify CFG

$S \to AB / aS_1 / A$ Remove 'ϵ' $\quad S \to AB / aSb / B / A$

$A \to Bd / d / \epsilon / B$ ⟶ of A $\quad A \to Bd / d / B$

$B \to bB / b$ $\quad B \to bB / b\phi$

$\downarrow$ Remove 'ϵ' of S

substitute A to GNF

$S \to AB / aSb / bB / b / Bd / d / ab$ Remove unit $\quad S \to AB / aSb / B / A / ab$

$A \to Bd / d / bB / b$ ⟵ production $\quad A \to Bd / d / B$

$B \to bB / b$ $\quad B \to bB / b$

No useless production

Identify already in GNF

$S \to BdB / dB / bBB / bB / aSb / bB / b / Bd / d / ab$

$A \to bBd / bd / dbB / b$

$B \to bB / b$

Why GNF & CNF is required?

put : $B \to bB / d$ where GNF not there

$S \to bBdB / bdB / dB / bBB / bB / aSb / bBd / bd / d /$
$bB / b / ab$
$A \to bBd / bd / d / bB / b$
$B \to bB / d$

$\downarrow$ put $b = X$, $c = Y$

$S \to bBYB / bYB / dB / bBB / aSX / bBY / bY / d / bB / b / aX$
$A \to bBY / bY / d / bB / b$
$B \to bB / d$
$X \to b$ $\quad$ $Y \to d$

① $S \to Sa / b$ Direct left Recursion

Sol: $S \to SX / Y$
$X \to a$
$Y \to b$
] ✗ Can't do

$S \to S \to S \to a$
$S \to a$

② $S \to AS / b$
$A \to SA / a$
] Indirect left Recursion

S

We need to first remove left recursion & then do the GNF.

[Part of Compiler Design (next sem)]

Advantages of
CNF :- (i) Every derivation of a string of n letters has exactly 2n-1 steps. Thus one can determine if a string is present in the language by exhaustive search of all derivation (ii) Parser can use Binary Trees

GNF: Used to prove equivalence of CFGs & PDAs
(Left most Derivations) is Guaranteed to Accept PDAs

Every derivation of a string 's' contains |s| applications

(external memory in terms of stack)

→ stack

PDA (Push Down Automata)

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

Q : set of finite state

q_0 : an initial state $q_0 \in Q$

Σ : set of finite I/P symbols

T : set of finite stack symbols

δ : Transition function

F : set of finite final states.

$$\delta : Q \times (\Sigma \cup \epsilon) \times T \longrightarrow Q \times T^*$$

from any state Q — we can take transition or NULL — taking Top T — going to Q (any other state) — either a T or any T

Replacement

$$\delta(q_i, a, X) \longrightarrow (q_j, Y)$$

q_i & $q_j \in Q$

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

$X \in T$

$Y \in T^*$

Z_0

Z_0 = a special stack symbol that indicates stack is empty

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

aaaa bbbb

→

A
A
A
A
Z_0

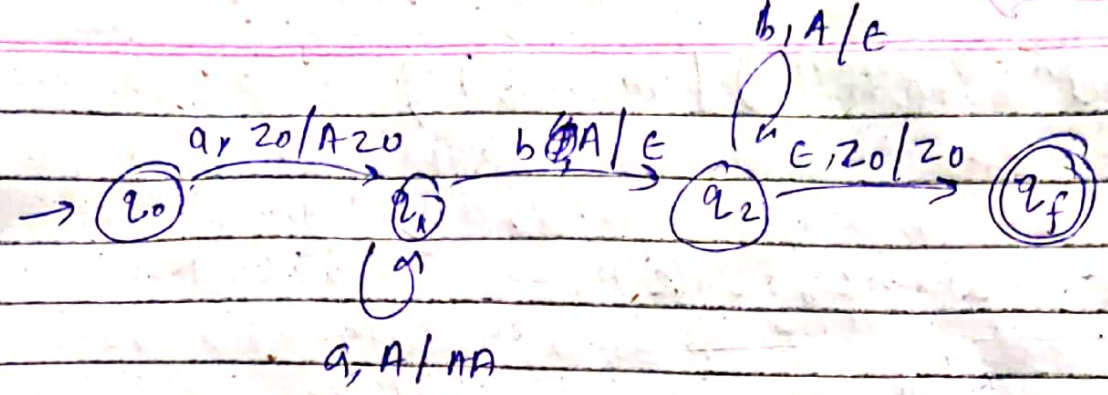
a आये तो push करो

b को लगाओ

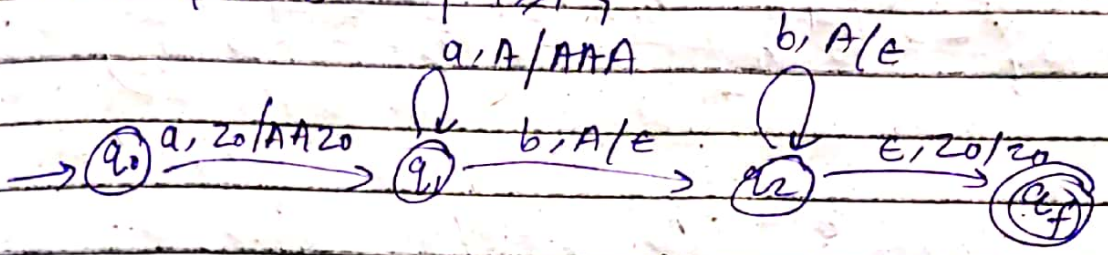
जब तक Z_0 नहीं आता।

There is no need for min. State PDA

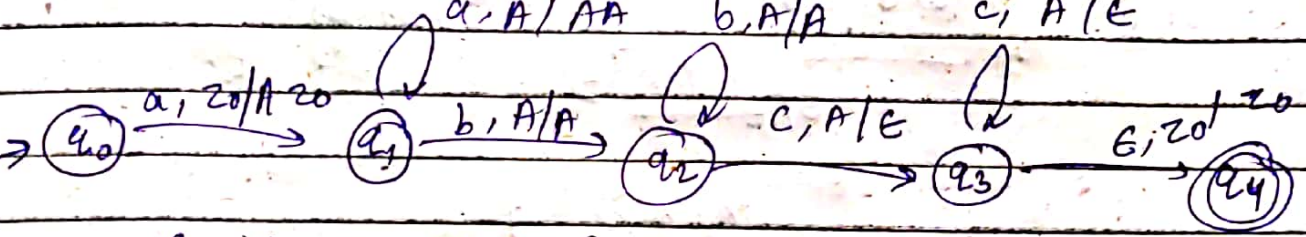
Tip most symbol हमेशा रखो जो होगा next को साथ लगा दो



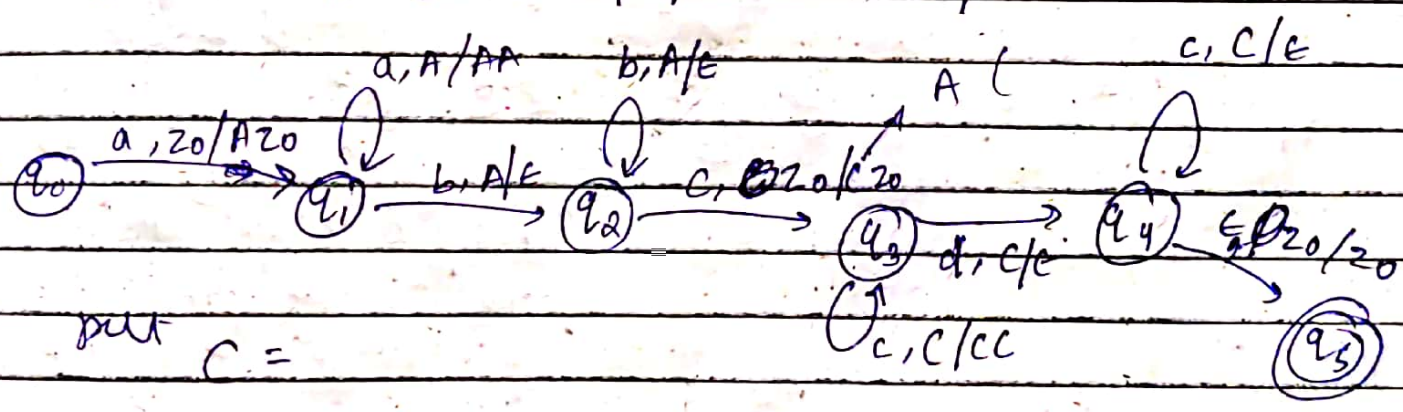
(ii) $L = \{a^n b^{2n} \mid n \geq 1\}$ push AA



(iii) $L = \{a^n b^m c^n \mid m, n \geq 1\}$



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



put C =

28.03.2023
Tuesday.

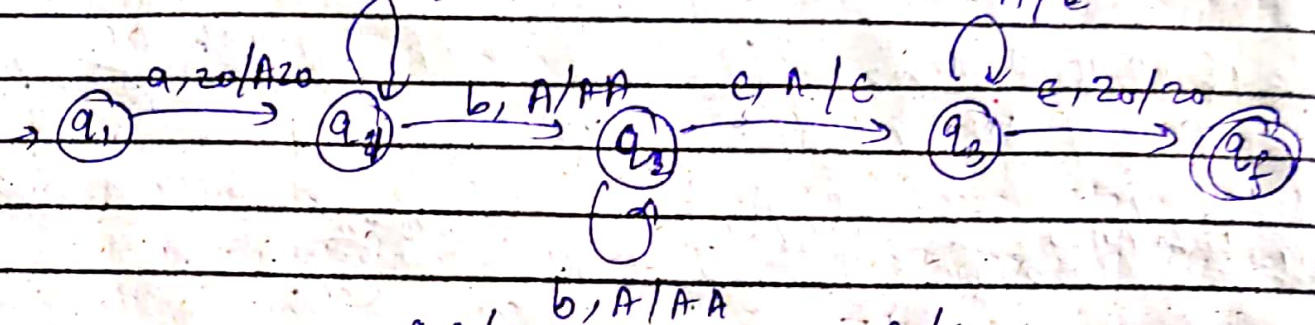
(v) $L = \{ a^n b^m c^{m+n} \mid m, n \geq 1 \}$ → break approach

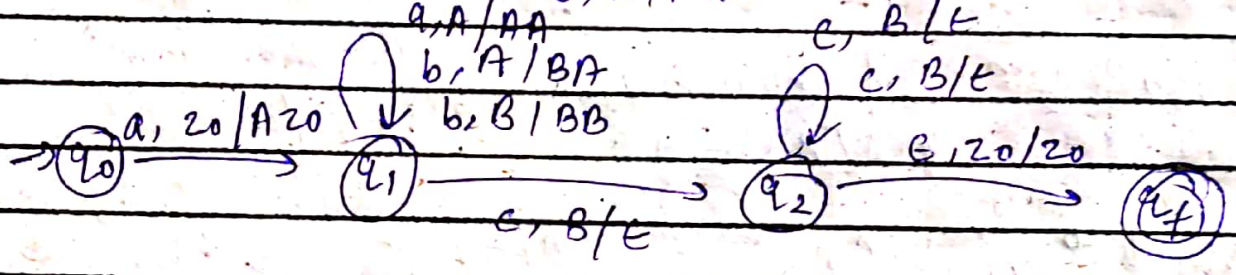
or

$n(a) + n(b) = n(c)$

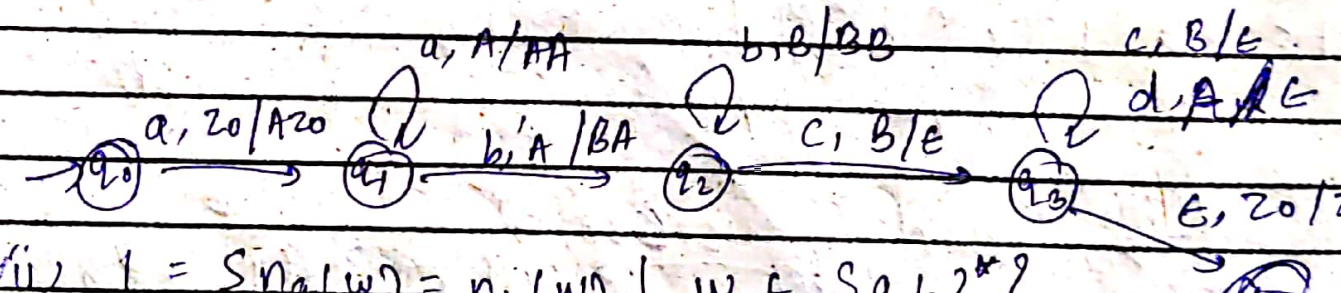
$a^n \underbrace{b^m c^m}_{\text{equal}} c^n$

equal





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



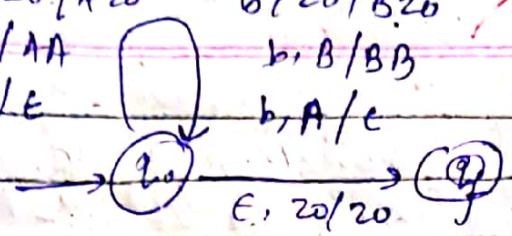
(vii) $L = \{ n_a(w) = n_b(w) \mid w \in \{a, b\}^* \}$

How to select 1st transition

4 case

last	current	transition
a	a	a, A/AA
a	b	b, A/ε
b	a	a, B/ε
b	b	b, B/BB

28.03.2023
Tues day.



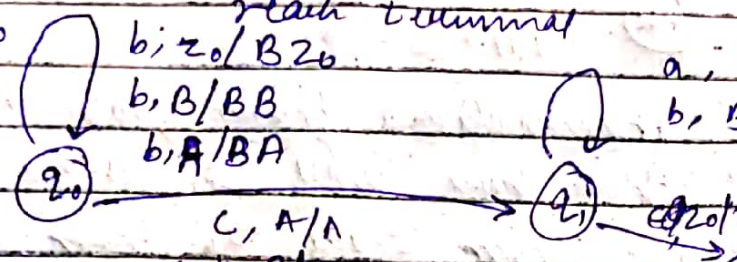
→ NPDA (look on cwbr-next → next → page to see DPDA)

Why each transition on one self loop?

Because they can come in any order.

(8) $L = \{ wcw^r \mid w \in \{a,b\}^* \text{ \& } c \text{ is a terminal} \}$

push complete w & pop it completely when we reach terminal



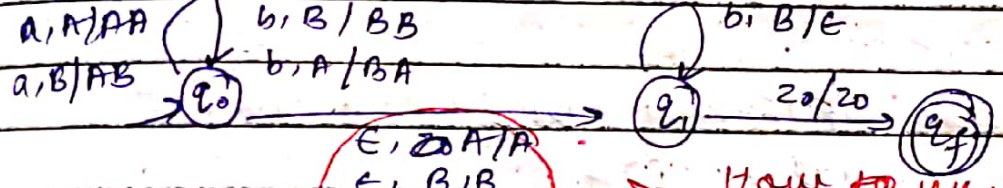
Why not ε because we want to store it

Null w as $\{a,b\}^*$

only two cases as palindrome if last was A then 'a' should be there & same for 'b'

{--- a c ---}
{--- b c ---}

(9) $L = \{ ww^r \mid w \in \{a,b\}^* \}$



→ How to we will know when to transit is not clear

abbbba → we can push whole string on stack

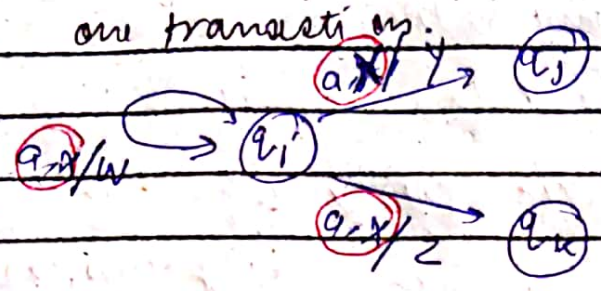
εabbbba
aεbbbba
abεbbba

length of w is 'n' then (n+1) attempts are required to check string is accepted by machine or not

Note

Deterministic PDA

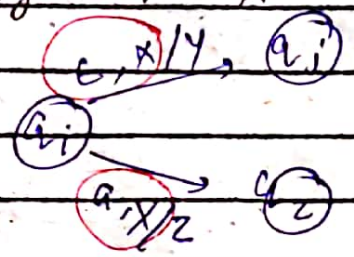
① From any state q_i for any combination of i/p symbol 'a' and stack symbol 'X', there must be atmost one transition.



Same input & stack transⁿ but diff. o/p transⁿ

∴ NPDA

② If there is a 'ε' transitions from state q_i on stack symbol 'X' then there will ~~do~~ not be any other i/p transition from state q_i ~~from~~ on stack symbol 'X'



NPDA

$$\boxed{\begin{array}{l} ww^r = \text{DPDA} \\ ww^r = \text{NPDA} \end{array}}$$

NOTE :-

Every DPDA is also NPDA but every NPDA can not be converted to equivalent DPDA

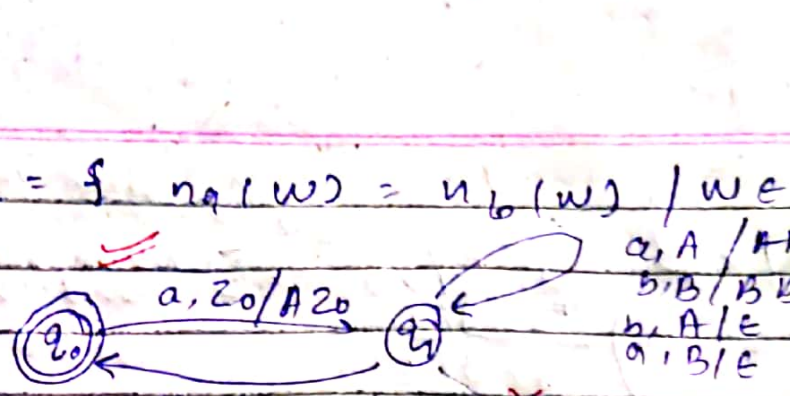
Power (NPDA) > Power (DPDA)

DPDA ⊂ NPDA

DPDA is (Proper Subset NPDA)

The language accepted by DPDA is called DCFL (Deterministic CFL). A language accepted by NPDA is CFL.

(vii) $L = \{ n_a(w) = n_b(w) \mid w \in \{a,b\}^+ \}$



CFG to PDA

Method 1. Using GNF

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

① Convert CFG to GNF

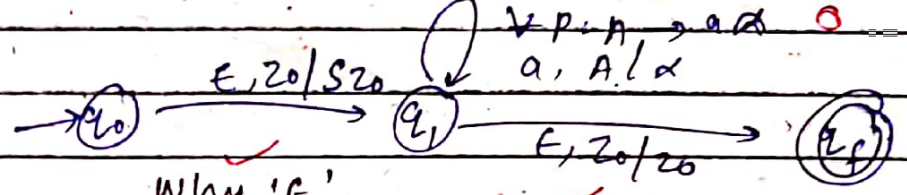
② Define transⁿ of PDA

$PDA(\{q_0, q_1, q_f\}, q_0, T, V, \delta, \{q_f\})$

$\forall \delta$: 1) $\delta(q_0, \epsilon, Z_0) \rightarrow (q_1, SZ_0)$

2) $\forall P: A \rightarrow a\alpha$ where $A \in V, \alpha \in V^*, a \in T$
define transⁿ:
$\delta(q_1, a, A) \rightarrow (q_1, \alpha)$

3) $\delta(q_1, \epsilon, Z_0) \rightarrow (q_f, Z_0)$



Why 'ε' to push Z0 onto stack

✓ → fixed

○ → ~~Trans~~ Production in CNF ?

(1) $S \rightarrow aSb \mid ab$

Sol: $S \rightarrow aSB \mid aB$

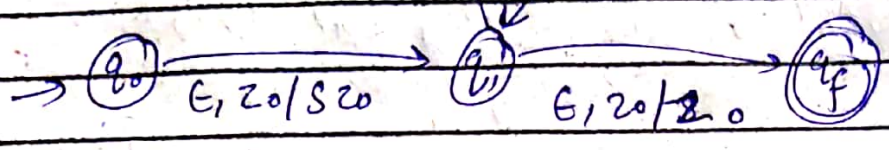
$B \rightarrow b$ 'ε'

$\boxed{a, S}/SB$ → (1) When we want 'a' part.

$\boxed{a, S}/B$ → (2) When we want to end 'a' part

$b, B/\epsilon$

NPDA



29.03.2023
Wednesday

(1) $S \to aSb \mid ab$

Sol: $S \to aSB \mid aB$

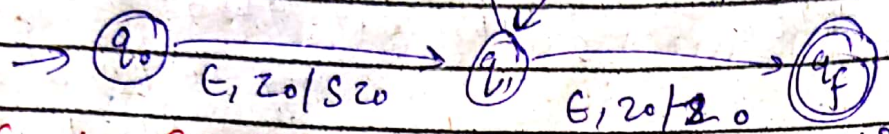
$B \to b$ 'ε'

$\boxed{a, S}/SB$ → (1) When we want 'a' part

$\boxed{a, S}/B$ → (2) When we want to end 'a' part

$b, B/\epsilon$

NPDA



CFG to PDA:

(NO GNF required)

Method-2 Let CFG (V, T, S, P) then its equivalent ~~PDA $(\{q_0, q_1, q_f\}$~~ PDA $(\{q_0, q_1, q_f\}, q_0, T, T \cup V, \delta, \{q_f\})$

$\forall$ δ defined from productions

(i) Initial : $\delta(q_0, \epsilon, Z_0) = (q_1, SZ_0)$

(ii) (a) For Production $\forall P$: $A \to \alpha$ where $A \in V$ & $\alpha \in (V \cup T)^*$

define transistions as

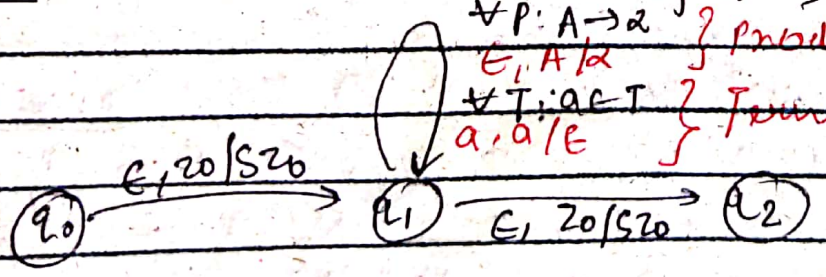
$\delta(q_1, \epsilon, A) = (q_1, \alpha)$

(b) For Terminal $\forall T$: $a \in T$

define transistions as

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

(iii) final: $\delta(q_1, \epsilon, Z_0) = (q_f, Z_0)$



Scanned with CamScanner

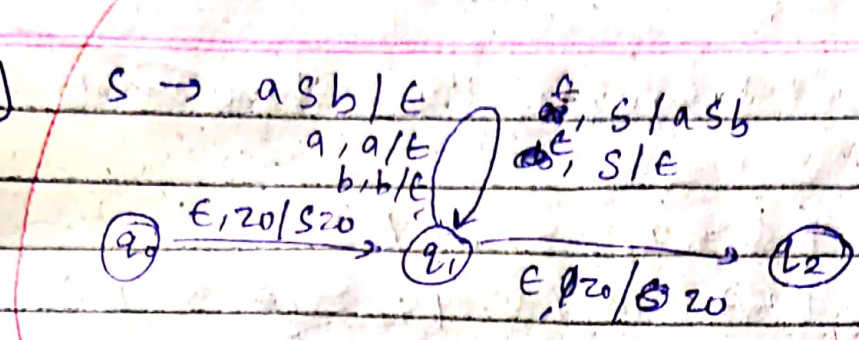
$A \in \{x_1, x_2, \ldots x_n\}$
$x_1 * x_2 = x_k$
$x_1, x_2, x_k \in A$] Closure property

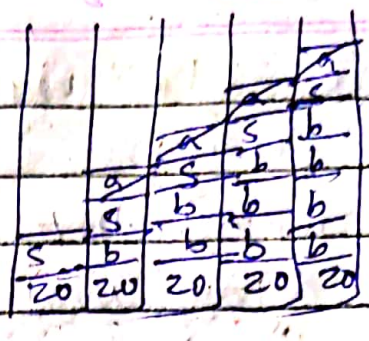
aaaabbbb

Q) $S \to aSb \mid \epsilon$

$a, a/\epsilon$
$b, b/\epsilon$

$\epsilon, S/aSb$
$\epsilon, S/\epsilon$





* Closure Properties of Regular Language

① **Union:** Let R_1 & R_2 are two Regular language then $R_1 \cup R_2 = R_3$ also a Regular language.

Proof: If R_1 & R_2 are regular languages then there exists DFAs for them to accept all the string of them.

$R_1 \Rightarrow M_1(Q_1, q_x, \Sigma, \delta_1, F_1)$

& $R_2 \Rightarrow M_2(Q_2, q_y, \Sigma, \delta_2, F_2)$

Then define one more DFA M_3 by $M_1 \cup M_2$ as

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

where $Q_3 = Q_1 \times Q_2$ (Done earlier)

$q_{xy} \in Q_3$

$\delta_3 = \delta_1 \times \delta_2$

$F_3 = F_1 \cup F_2$

∴ language accepted by M_3 is R_3 which is $R_1 \cup R_2$

② **Intersection:** Same as union

Proof. Just replace '$\cup$' → '$\cap$'

(Done earlier) union Intersection

③ **Complement:** Let R_1 is a regular language then $R_1 = R_2$ is also Regular

Proof:

Scanned with CamScanner

If R_1 is a regular language then there exists DFA for it.

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

Now define another DFA $M_2(Q, q_0, \Sigma, \delta, Q-F)$

Then the language accepted by M_2 is R_2 which is $\overline{R_1}$.

④ **Set Difference**: Let R_1 & R_2 are two Regular language then $R_1 - R_2 = R_3$ also Regular language.

Proof: $R_1 - R_2 = R_1 \cap \overline{R_2}$

$\overline{R_2} \rightarrow$ Regular language

$\cap \rightarrow$ Regular

$\therefore$ Regular language.

⑤ **Kleen's Closure**: (*) Let R_1 is a regular language then $R_1^* = R_2$ is also Regular language.

Proof: If R_1 is a regular language then there exists FA for it.

$R_1 \Rightarrow$ FA $M_1(Q, q_0, \Sigma, \delta, F)$

Now define another DFA $M_2(Q, q_0, \Sigma, \delta_2, F)$

$\delta_2 = \delta_1 \cup \{\delta(q_0, \epsilon) \rightarrow q_f\} \cup$

$\{\delta(q_f, \epsilon) \rightarrow q_0\}$, where $q_f \in F$

⑥ **Reversal**: Let R_1 is a Regular language then $R_1^R = R_2$ is also Regular language.

Proof: If R_1 is regular then there exists FA for R_1 = FA $M_1(Q, q_0, \Sigma, \delta, F)$

where $\delta_2 = \delta^R$

Scanned with CamScanner

change :

VTSP → Cram-man

define another FA $M_2(Q, \Sigma, \delta_2, \{q_0\})$

$\forall \cdot \delta(q_i, a) \rightarrow q_j \in \delta$

then

$\forall \delta(q_j, a) \rightarrow q_i \in \delta_2$

The language accepted by M_2 is R_2 which is R_1^R

* Closure Properties of CFL

① Union let L_1 & L_2 are two CFLs then $L_1 \cup L_2 = L_3$ is also CFL

Proof: If L_1 & L_2 are 2 CFLs then there exists CFGs for them

$L_1 = G(L_1)\ (V_1, T_1, S_1, P_1)$

$L_2 = G(L_2)\ (V_2, T_2, S_2, P_2)$

Now define $G_3\ (V_3, T_3, S_3, P_3)$

where $V_3 = V_1 \cup V_2 \cup \{S_3\}$

$T_3 = T_1 \cup T_2$

$S_3 \in V_3$

$P_3 = P_1 \cup P_2 \cup \{S_3 \rightarrow S_1 / S_2\}$

∴ The language ~~accepted~~ generated by G_3 is Regular language

② Kleen's Closure : (*) let L_1 is CFL then $L_1^* = L_2$ is also CFL

Proof: ~~Lets~~ If L_1 is CFL then there exists CFG for it.

Scanned with CamScanner

$L_1 = G(L_1)(V, T, S, P)$

Then define $G_2(V_1, T, S_2, P_2)$

where $V_1 = V \cup \{S_2\}$

$P_2 = P \cup \{S_2 \to SS_2 / \epsilon\}$

further generate होता रहे ∀ 2π NUM

∴ The language generated by G_2 is ~~RG~~ CFG

③ Reversal : If L_1 is CFL then $L_1^R = L_2$ is also CFL.

Proof : If L_1 is CFL then there exists CFG for it.

$L_1 = G(L_1)(V, T, S, P)$

then define $G_2(V, T, S, P_2)$

where $\forall P_2 : A \to \beta$ where $\beta = \alpha^R$

$\forall P : A \to \alpha$

∴ The language generated by G_2 is CFG

NOTE: Intersection, Complement and Set Difference are not closure ~~under~~ for CFL

31.03.2023
Friday

TURING MACHINE (TM)

Can do any computation

Infinite length Tape

| # | # | string | | # | # |

↳ Blank symbol

Read or Write Head
(Pts to first character of string)

Every machine ~~other~~ ~~or~~ have Head R/W but speciality is that head R/W can moves left or right or can remain at that position.

TM (Q, q_0, Σ, T, δ, F)

Q : set of finite states

q_0 : An initial state $q_0 \in Q$

Σ : set of finite I/p symbols

(Tape) T : set of finite tape symbols

F : set of finite final state

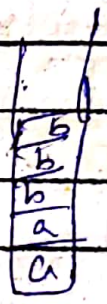
δ : $Q \times (\Sigma \cup T) \rightarrow Q \times (\Sigma \cup T) \times \{L, R, N\}$ replaced by

[from any state Q either we choose i/p symbol or tape symbol we reach to state Q by replacing it with either i/p symbol or tape symbol & remember the direction of Head.]

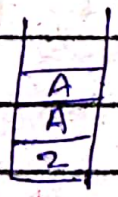
NCFL (Non Content Free Language)

* CFL (Content Free Language)
  (i) A language is CFL when there exists a PDA for it that accepts all strings of the language
  (ii) A language is CFL when there exists a CFG for it that generates all strings of it.

* Pumping lemma :- We can prove language is not CFL by it.
  (But we will do without it)



Q1. ① $L = \{a^n b^n c^n \mid n \geq 1\}$ NCFL
② $L = \{a^n b^m c^n d^m \mid m, n \geq 1\}$ NCFL
③ $L = \{a^n b^m c^m d^n \mid m, n \geq 1\}$ ~~NCFL~~ CFL
④ $L = \{a^n b^m \mid n = m^2\}$ NCFL



Not linear

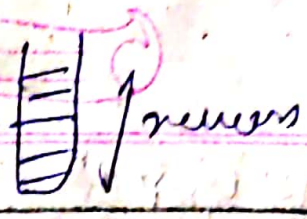
NOTE: In CFL we can compare two variables linearly at a time only $y = mx + c$, can other two $y \propto x$

⑤ $L = \{a^n b^m c^k \mid n = m \times k\}$ NCFL
Pt. 3 or 2 k

⑥ $L = \{a^n b^m c^k \mid m = n + k\}$, CFL
for linear equation

⑦ $L = \{a^p \mid p \text{ is prime}\}$ NCFL
⑧ $L = \{wcw \mid w \in \{a, b\}^*\}$ NCFL → we will have w times ∴ NCFL

Scanned with CamScanner



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

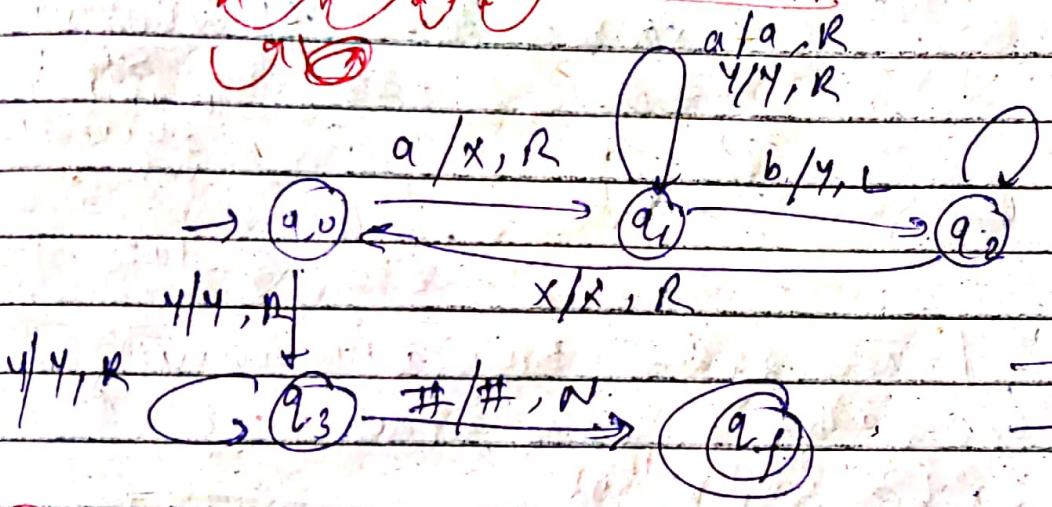
TURING MACHINE

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

#	a	a	a	a	b	b	b	b	#

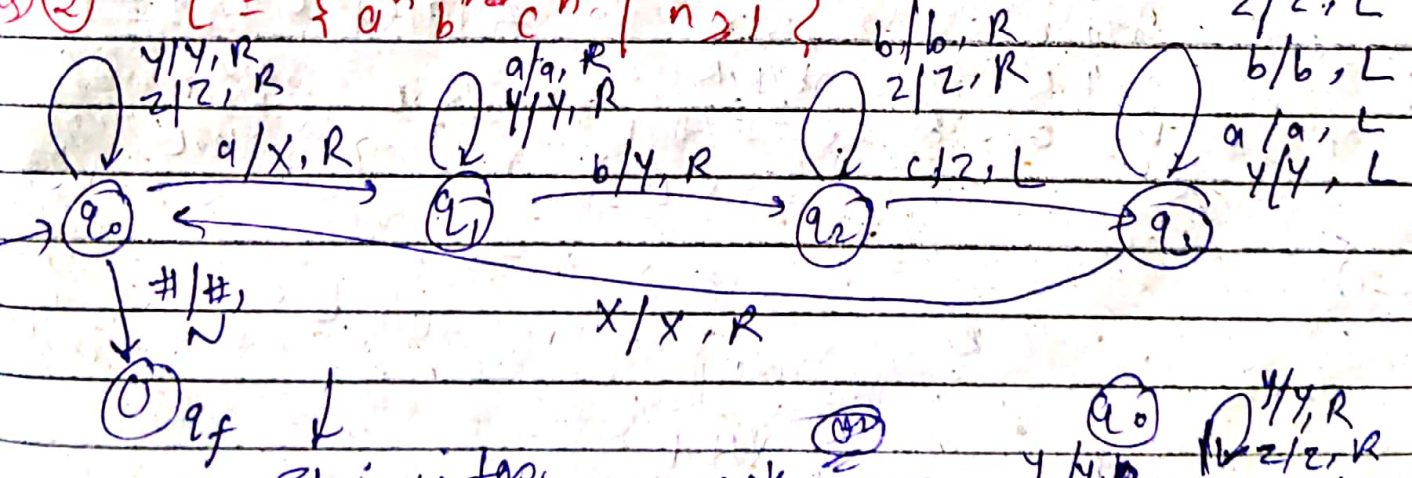
#	X	a	a	a	b	b	b	b	#

#	X	X	a	a	Y	Y	b	b	#

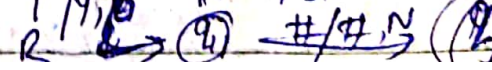


X	Y
a a	b b c c

Q(2) $L = \{a^n b^n c^n \mid n \geq 1\}$

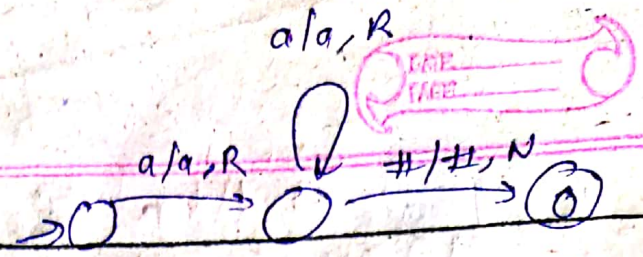


This is for $(a^n b^n c^n)^k$ $k \geq 0$



Replace A by B ~~with~~ A/B, dir^n move in dir^n

Q① $L = \{a^n \mid n \geq 1\}$

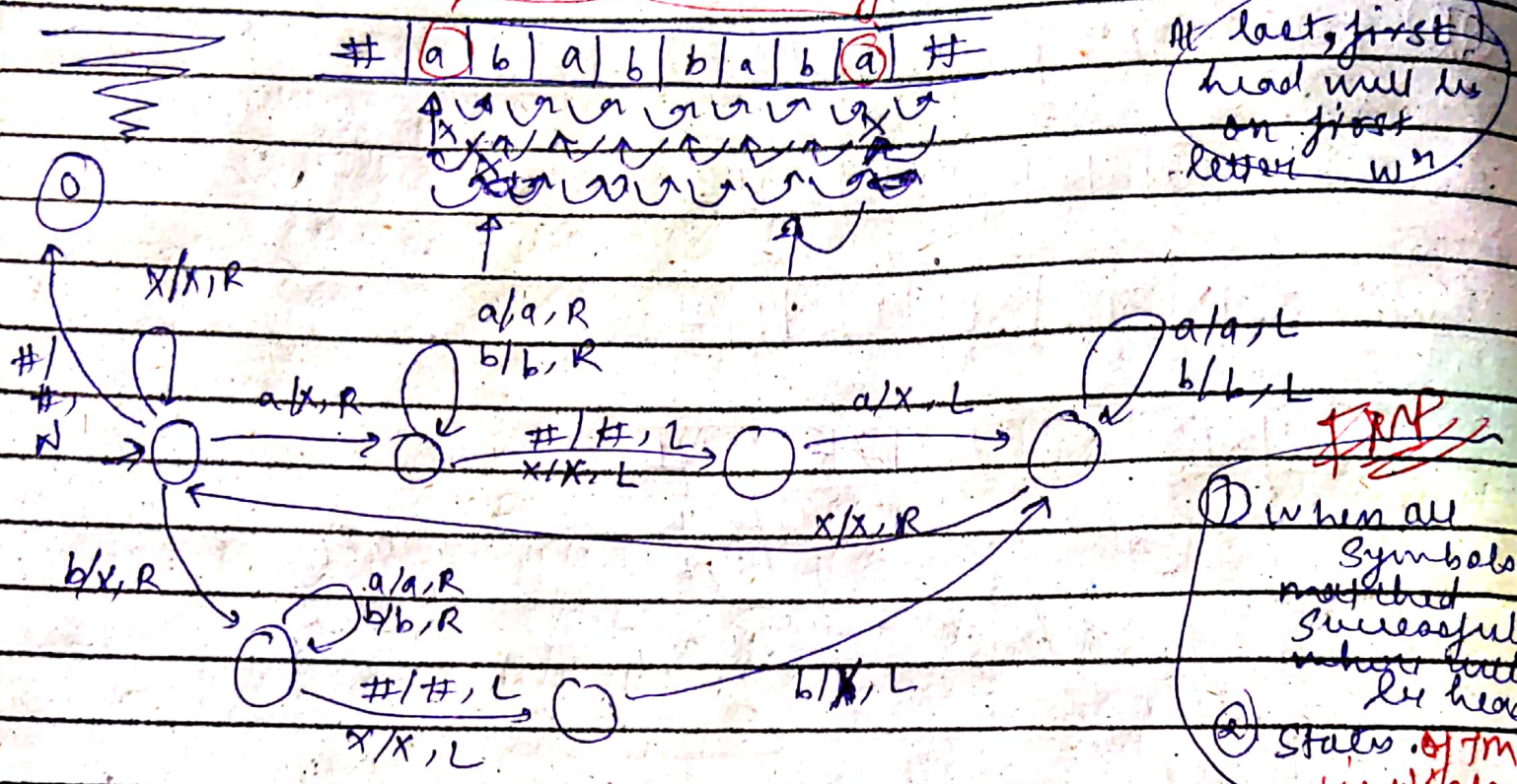


Q② $L = \{ww^r \mid w \in \{a,b\}^*\}$ abccba

Same symbol

#	a	b	a	b	b	a	b	a	#

last, first head will be on first letter w^r



① when all symbols matched successfully when reach to head

② States of TM (initial to which)

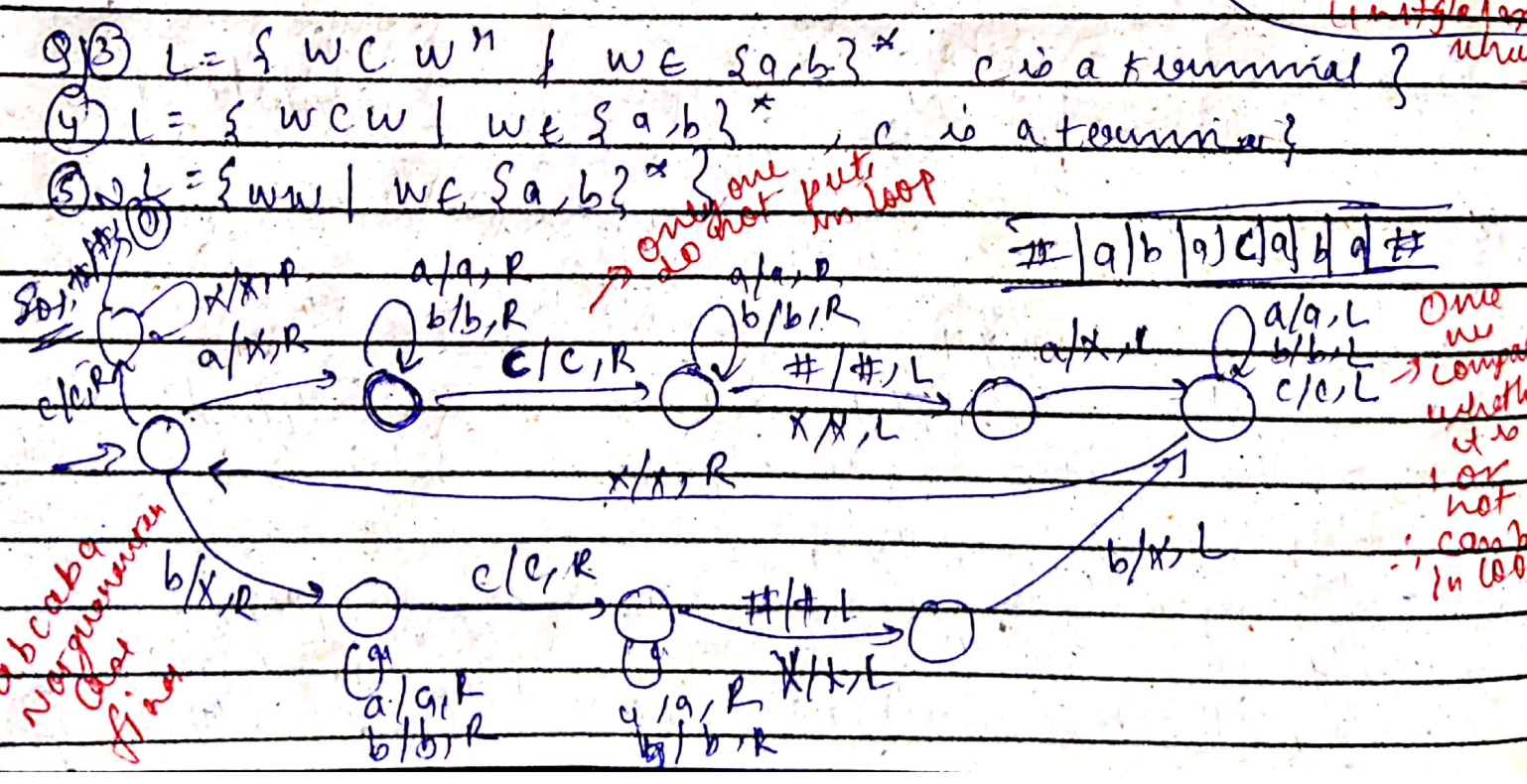
Q③ $L = \{wcw^r \mid w \in \{a,b\}^*$ c is a terminal$\}$

④ $L = \{wcw \mid w \in \{a,b\}^*$, c is a terminal$\}$

⑤ $L = \{ww \mid w \in \{a,b\}^*\}$

only one do not put in loop

#	a	b	a	c	a	b	a	#

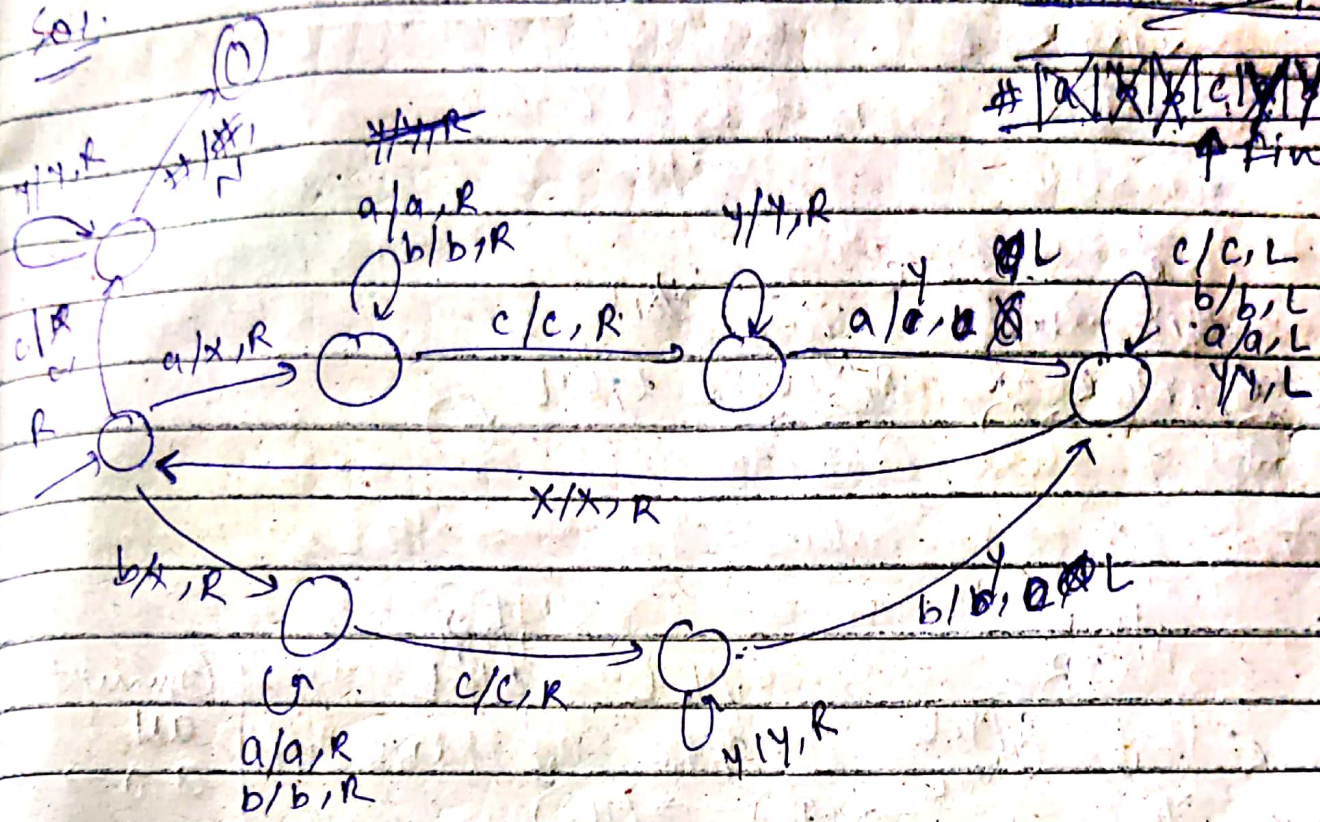


One we compare whether it is or not can be in loop

abcaba Nondeterministic find

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

Sol:



| a | b | b | a | b | b |

| a | x | x | c | y | y | y |

↑ final head pos.

Types of TM

TM

NTM (Non Deterministic Turing Machine)

DTM (Deterministic Turing Machine)

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

$2^{Q \times (\Sigma \cup T) \times \{L,R,N\}}$

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

At most one transition

Analogy NFA DFA

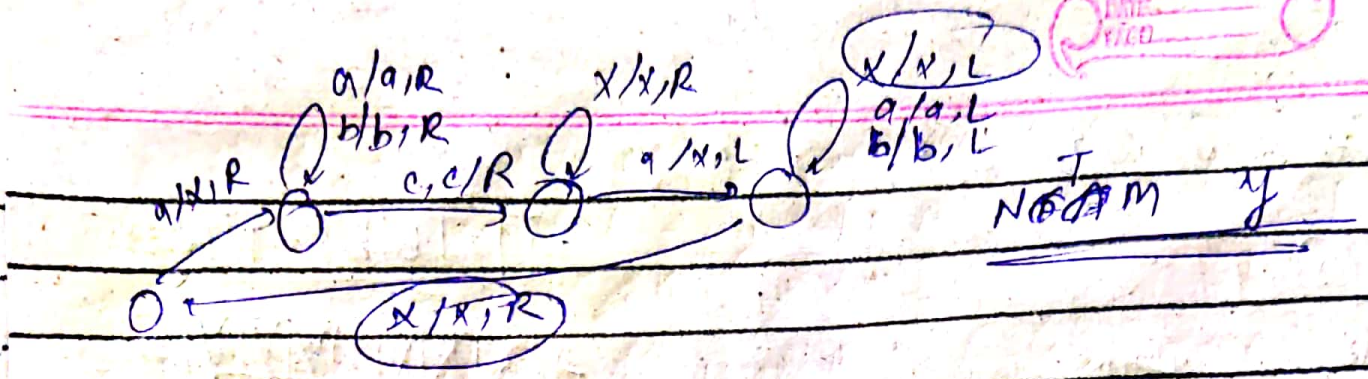
(diff. → DFA total function but no ε-cycle in DTM)

NOTE:

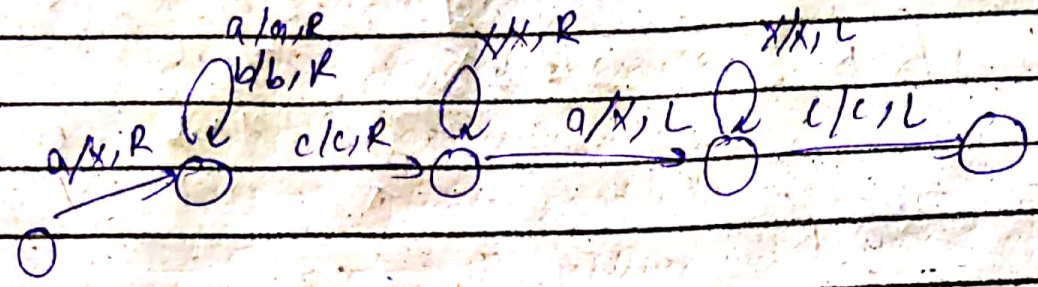
Every NTM can be converted to equivalent DTM. Every DTM is an NTM.

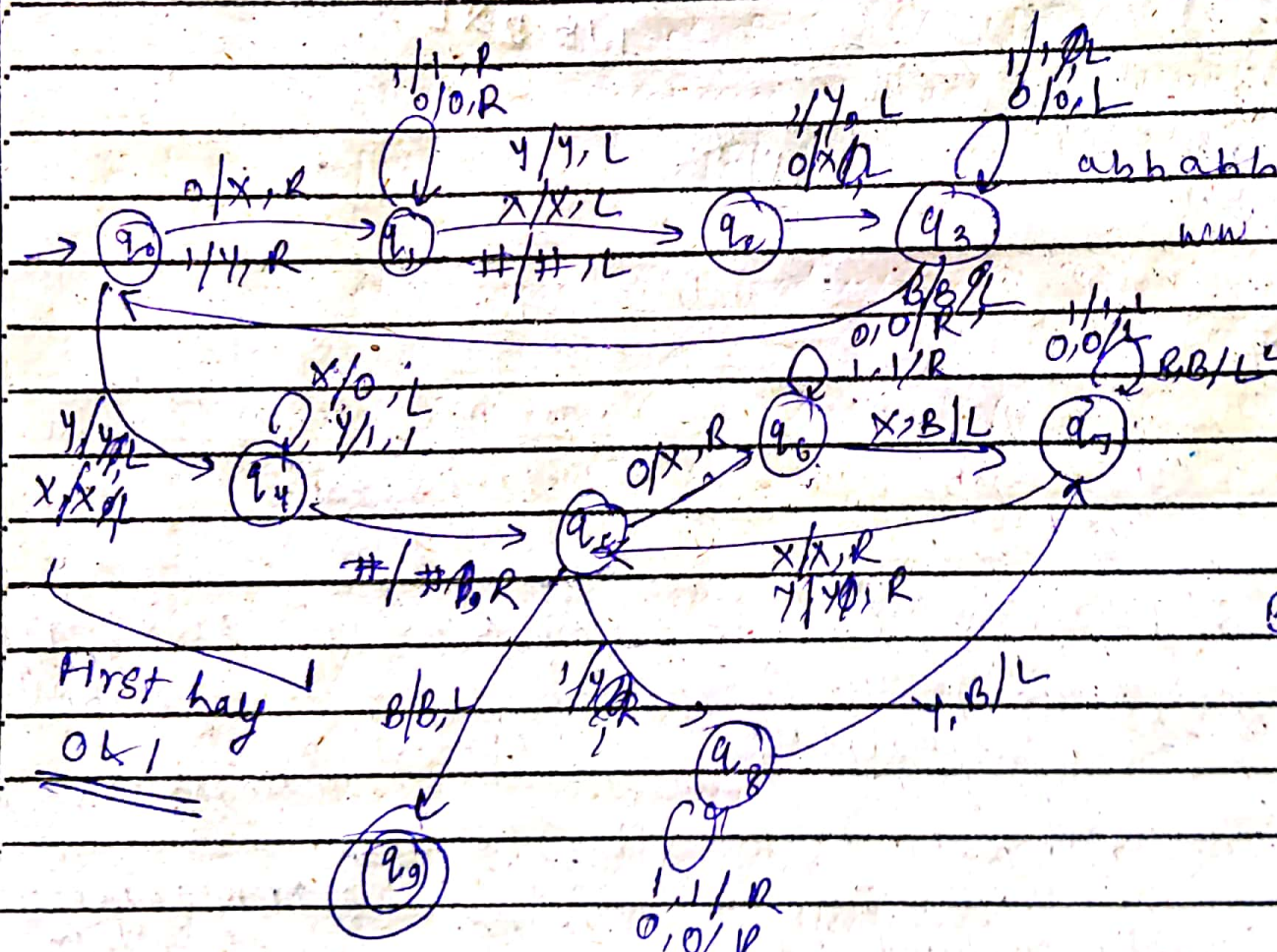
Power (NTM) = Power (DTM)

No need to
Same State in
any other



NFA M





First half
ok!

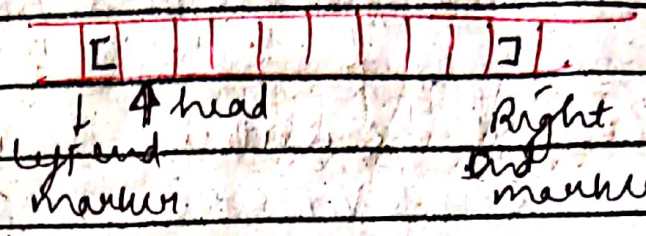
B → xAy
there
are
visited

X Y X X a Y X

Linear Bounded Automata (LBA)

→ Bounded

TM: infinite length



LBA is a non deterministic TM where the tape of TM is bounded by the special tape symbol '[' left end marker & ']' right end marker and no transistion can move to left of left end marker & right of right end marker.

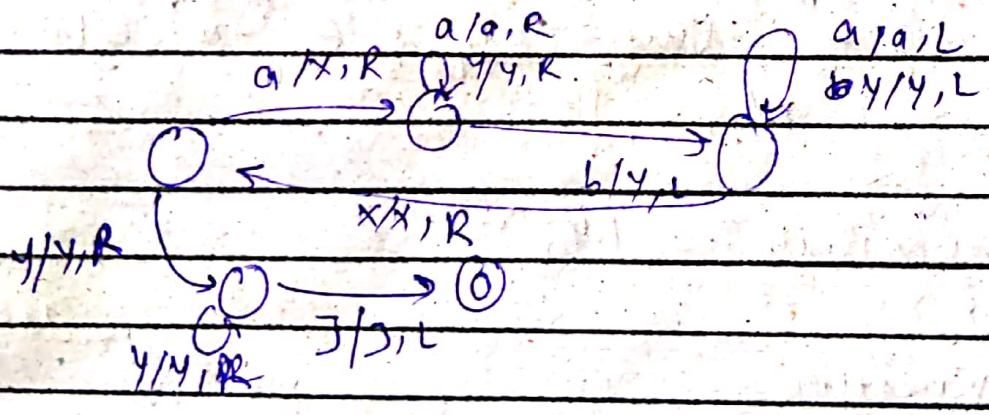
$$\boxed{\begin{aligned} \delta(q_i, '[') &\rightarrow (q_i, '[', R) \\ \delta(q_i, ']') &\rightarrow (q_j, ']', L) \end{aligned}}$$

→ draw TM

→ find transistion

Some authors start from '[' can keep 'N' and some from first string symbol

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



11.04.2023
Tuesday

Context Sensitive Language (CSL)

The language accepted by LBA is called CSL

ex $L = \{a^n b^n c^n \mid n \geq 0\}$ CSL (not CFL)

Context Sensitive Grammar (CSG)

CSG (V, T, S, P)

V = set of finite variables

T = set of finite terminals

S = Starting variables

P = set of finite products

$\forall P: \alpha \to \beta$ where $|\alpha| \leq |\beta|$

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

also α contains at least 1 variable

$\beta \neq \epsilon$ $|\alpha| \geq 1$ $\therefore |\beta| \neq 0$ as $|\beta| \geq |\alpha|$

NOTE

① CSG can not have 'ϵ' productions.

② Language generated by CSG, $L(CSG)$ is ϵ-free CSL

↳ CSL can contain ϵ but CSG is ϵ free CSL

$$\boxed{CSL = L(CSG) \cup \{\epsilon\}}$$

Unrestricted Grammar (UG)

UG (V, T, S, P)

V: set of variables

T: set of Terminals

S: Starting Variables

P: set of finite productions

No restriction on this but LHS should have one variable.

$\forall P: \alpha \to \beta$ where $\alpha \in (V \cup T)^+$ and contains at least one terminal $\beta \in (V \cup T)^*$

all CSG productions will be the subpart of UG.

→ can't be ε in CFG

Design CSG for the following

① $L = \{a^n b^n c^n \mid n \geq 0\}$ ∴ $n \geq 1$

$S \to aSbc$ → aaabcbcbcbc ✗

Intuition: aaabcbcbcbc

→ aaaaBcBcBcBc

→ aaaaBBcBcBcc

aaaaBBBcBccc

aaaaBBBBcccc

→ aaaabBBBcccc

→ aaaabbBBcccc

→ aaaabbbBcccc

→ aaaabbbbcccc

$S \to aSBc \mid aBc$

$cB \to Bc$

$aB \to ab$

$bB \to bb$

or

$S \to aSBc \mid abc$

$cB \to Bc$

$bB \to bb$

problem

$B \to b$

Carries

Not CFG

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

Design CSG.

Sol:

→ Unrestricted Grammar

$S \to aSBc \mid \epsilon$

$cB \to Bc$

$aB \to ab$

$bB \to bb$.

or

$S \to aSBC \mid \epsilon$

$CB \to BC$

$aB \to ab$.

$bB \to bb$

$bC \to bc$

$cC \to cc$

or

$S \to aSbC \mid \epsilon$ ✗

$Cb \to bC$

$bC \to bc$ ✗

aaaabbbbcc

aaabcbcbc

$Cb \to bC$

~~aaabcbcb~~

aaabbcbcc

aaabbbbccc

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

Sol:

$S \to S_1 S_2$

$S_1 \to aS_1Bc \mid aBc$

$S_2 \to bS_2d \mid Bd$ (swapping)

$cB \to Bc$

$aB \to ab$

$bB \to bb$

$bC \to bc$

$cC \to cc$

[linear]

aaacccbbdd

$\to$ aaacccBBdd

$cB \to Bc$

$\to$ aaa cccBcBdd

aaa cBcBcdd

aaa BcBccdd

aaa BBcccdd

$\to$ aaabBcccdd

$\to$ aaabbcccdd

$\to$ aaabbccCdd

$\to$ aaabbccccdd

or

$S \to X/Y$

$X \to aXc \mid ac$

$Y \to BYd \mid Bd$

$aB \to ab$

$bB \to bb$

Swapping

aaacccbbdd

aaa cCbCbdd

aaa CbC bCdd

aaa bCbCCdd

aaa bbcccdd

or

$S \to X/Y$

$X \to aXc \mid ac$

$Y \to bYd \mid bd$

$cb \to bc$

$cd \to cd$

$cc \to cc$

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

$S \to aSBc \mid abc \mid \epsilon$

$cB \to Bc$

$bB \to bb$

13.04.2023
Thursday

Language generated by UG

UG generates 2 languages - Recursive & Recursive Enumerable.

L(UG) - Recursive
- Recursive Enumerable

Recursive lang:- A language is recursive when there exists the TM(L) that satisfies

(a) $\forall$ w $\in$ L, TM(L) halts on h_a (halt accept)

TM(L) - h_a halt accept
- h_r halt reject

We can convert any TM to two halt states TM

(b) $\forall$ w $\notin$ L, TM(L) halts on h_r halt reject.

TM(L) ~~w/ Rec~~ L Recursive
- $\forall$ w $\in$ L → h_a Yes
- $\forall$ w $\notin$ L → h_r No

Recursive Enumerable language:- A language L is RE when there exists TM(L) that satisfies either

(a) $\forall$ w $\in$ L, TM(L) halts on ~~the~~ h_a ~~halt state~~ accept

(b) $\forall$ w $\notin$ L, ~~TM~~ undecidable (may loop forever)

or

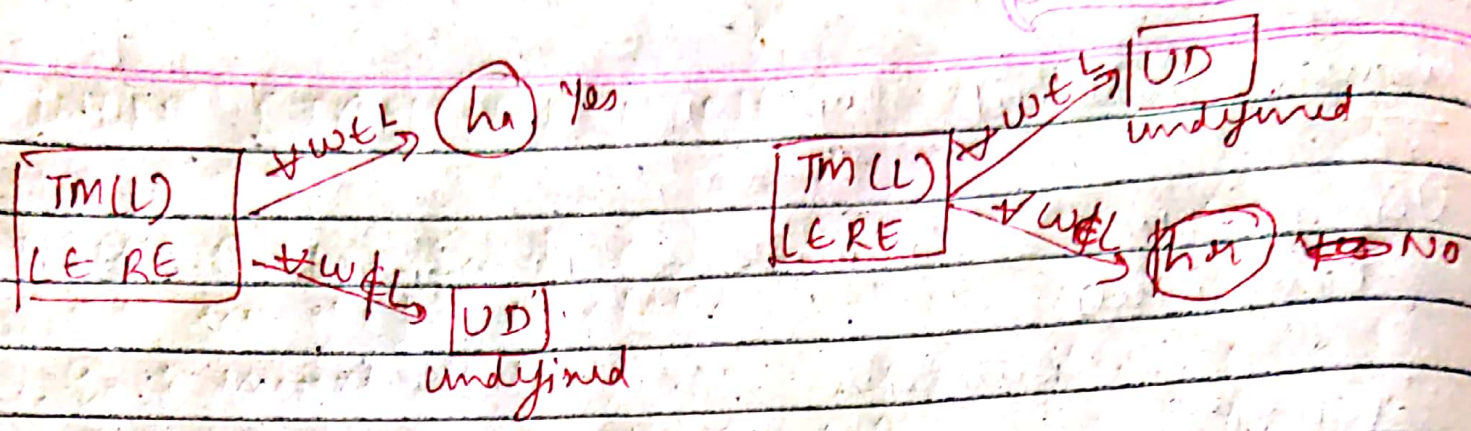
(a) $\forall$ w $\in$ L, undecidable (may loop forever)

(b) $\forall$ w $\notin$ L, TM(L) halts on h_r halt reject

Scanned with CamScanner

Recur.: If we know YES (h_a) then we know (h_r) No

Recur Enu: Don't know about any one of them.

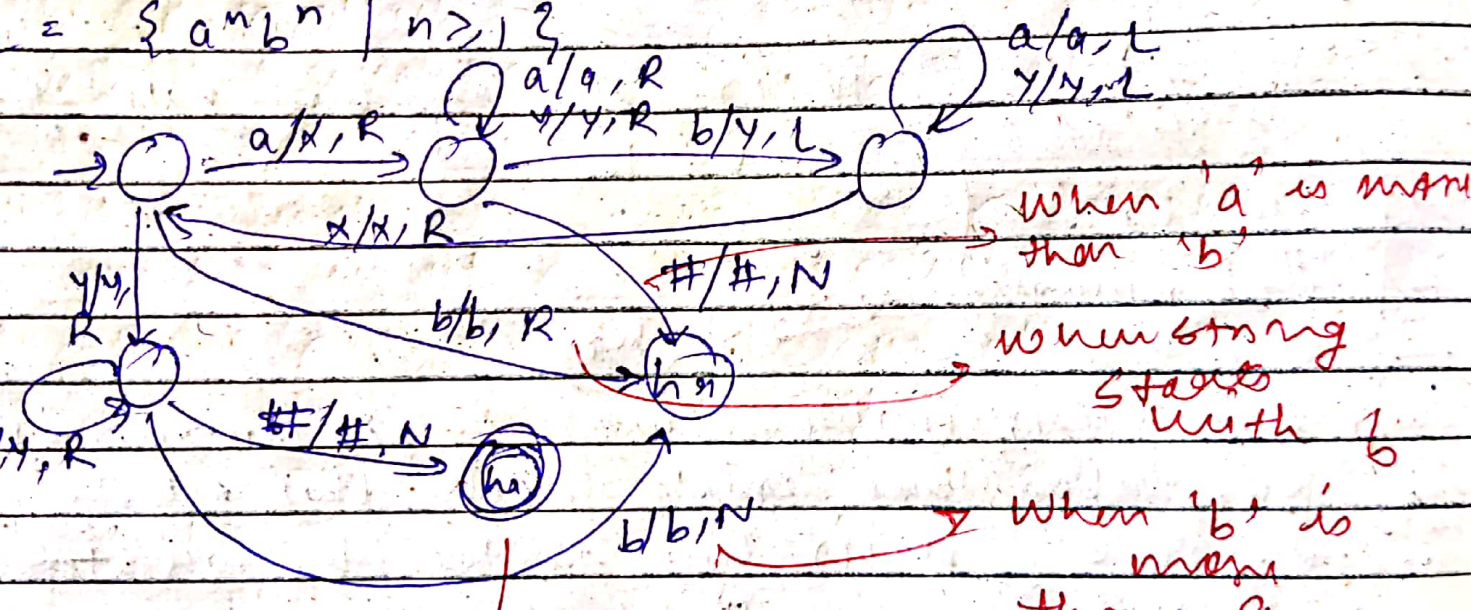


NOTE:

① Recursive language is closed under complementation
Replace h_a by h_r & h_r by h_a. (we know both)
↳ totally decidable while

② Recursive Enumerable lang. is not closed under complementation
↳ partially undecidable

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



Final state = h_a

When h_r & Guard

Closure Properties.

	Regular	Deterministic Context Free	Context Free (CFL)	Context Sensitive	Recursive	Recursively Enumerable
1) Union	✓	✗ (CFL)	✓	✓	✓	✓
2) Intersection	✓	✗ (CSL)	✗ (CSL)	✓	✓	✓
3) Complementation	✓	✓	✗ (CSL)	✓	✓	✗ (Undecidable)
4) Set difference	✓	✗ (CSL)	✗ (CSL)	✓	✓	(Undecidable) ✗ $L_1 - L_2 = L_1 \cap \overline{L_2}$
5) Kleen's Closure	✓	✗ (CFL)	✓	✓	✓	✓
6) Concat	✓	✗ (CFL)	✓	✓	✓	✓
7) Reversal	✓	✗ (CFL)	✓	✓	✓	✓

NOTE: If something is ✗ then look in higher language if its not answer

Union, Complementation

DCFL ∩ DCFL = CSL

$L_1 = \{a^n b^m c^n \mid n, m \geq 0\}$ → DCFL

∩

$L_2 = \{a^n b^n c^m \mid n, m \geq 0\}$ → DCFL

$L_3 = \{a^n b^n c^n \mid n \geq 0\}$ → CSL

$\{\epsilon, ac, b, abc, \ldots\}$

$\{\epsilon, ab, c, abc, \ldots\}$

complement.

NP-Hard, Completness Turing

FINITE STATE MACHINE

FSM $(Q, q_0, \Sigma, T, \delta)$

Q : set of finite states

q_0 : initial state

Σ : set of finite I/p symbols

T : set of finite o/p symbols

δ : Transition function

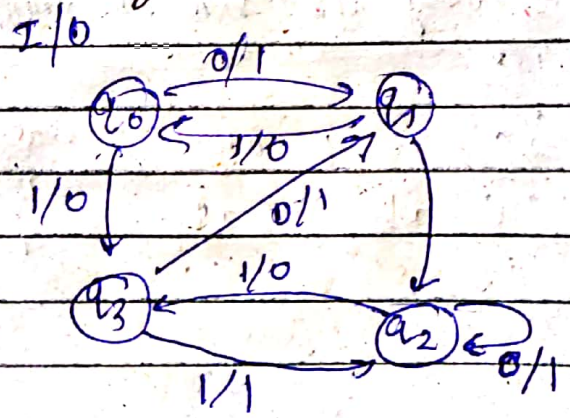
$$\delta : Q \times \Sigma \rightarrow Q \times T$$

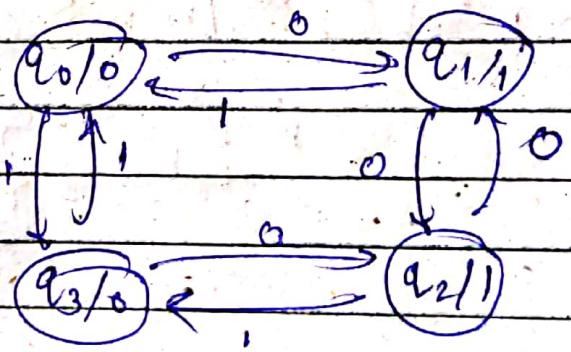
There are 2 types of FSM :-

FSM

Mealy m/c	Moore m/c
1) The o/p is attached with the transitions	1) The o/p is attached with the state
2) When 'n' length i/p is supplied we get 'n' length o/p	2) When 'n' length i/p is supplied we get '(n+1)' length o/p.

Moore → 0 शुरुआत में आया





(Black वाले को [illegible]
Blue वाला mechanical [illegible].)

I/P 01011101

O/P 10100110

I/P 10101100

O/P $q_0 q_3 q_2 q_3 q_2 q_3 q_0 q_1 q_0$

O/P 00101001

[illegible] O/P

Moore M/C to Mealy M/C

1. Prepare transition table of moore M/C
2. According to that prepare transition table of mealy M/C

States	i/p 0	i/p 1	o/p
q_0	q_1	q_3	0
q_1	q_2	q_0	1
q_2	q_1	q_3	1
q_3	q_2	q_0	0

→

States	(i/p)/(o/p) 0	1
q_0	q_1 1	q_3 0
q_1	q_2 1	q_0 0
q_2	q_1 1	q_3 0
q_3	q_2 1	q_0 0

output o/p of these symbols.

Mealy ~~to~~ M/C to Moore M/C

1. Prepare transition table of mealy M/C
2. According to that prepare transition table of Moore M/C.

Scanned with CamScanner

Mealy M/c

States	I/P		O/P	
→ q_0	q_1	0	q_2	0
q_1	q_2	1	q_3	0
q_2	q_0	0	q_1	0
q_3	q_1	1	q_2	0

Moore M/c

States	I/P 0	1	O/P
q_0	q_{10}	q_{20}	0
q_{10}	q_{21}	q_3	0
q_{11}	q_{21}	q_3	1
q_{20}	q_0	q_{10}	0
q_{21}	q_0	q_{10}	1
q_3	q_{11}	q_{20}	0

See the occurrence of q_0 only

q_1 two q_{10} q_{11}

Transition को देखकर

(1)

$q_0 \xrightarrow{0} q_{10}$