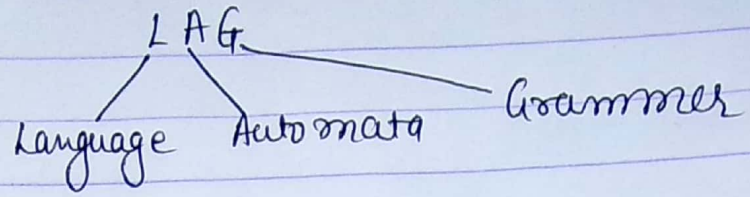
27/01/2024

Automata Theory

TOC – Mathematical Model



Language :-

① Symbol. ⇒ {a, b, c, 0, 1, 2, 3}

To understand a language you have to identify symbols unit.

② Alphabet Σ (a, b) ← finite set of symbols

③ String (words) collection/sequence of alphabets

L = {a, b, aa, ab, ba, bb}.

Length of string = 2
{aa, ab, ba, bb}

length = 3
{aaa, aab, aba, abb, baa, bab, bba, bbb}

④ Language ← collection of strings.

Q ⇒ Σ(a, b)

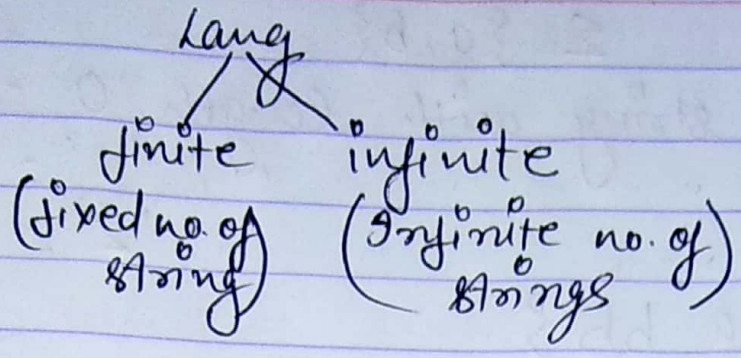
L_1 = strings of length 3 ← finite

L_2 = Strings starts from a and end with b

L_2 = {aa, aaa, aaaa, aba, abba ...}

L_3 = String with length 0

= ϵ



L

L1 = String length = 2

L1 = {aa, ba, ab, bb}

L2 = atleast one a

{a, aa, aaa :---

ab, aab, abba, abbba}

Q ⇒ find if 'abba' is part of L2.

Automata ⇒ model or machine

⇒ whether a string is part of language

FA

PDA

TM

Power of Σ $\Sigma = \{a, b\}$

Σ^0 = set of all string with length '0' = {ϵ} (NULL string)

Σ^1 = " " " " " " '1' = {a, b}

Σ^2 = " " " " " " '2' =

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

Σ^* ⇒ **Kleene closure**

⇒ Set of all strings of all length possible on a & b possible

⇒ $(a+b)^*$

⇒ {a, b, aa, bb, ----} Infinite language

Σ^+ ⇒ positive closure

= set of all strings possible but not Σ^0

$$\Sigma^* = \Sigma^+ + \Sigma^0$$

$$\Sigma^* - \Sigma^0 = \Sigma^+$$

Grammer

Grammer = A grammer 'G' is dfined as Quadruple

$G = \{V, T, P, S\}$

Grammer whether perticular sentence is part of the language

My name is Snigdha.

✓ Are you a student?

✗ a student are you?

$G = \{V, T, P, S\}$

- V — Variable (Capital letter)
- T — Terminal (Small letters)
- P — product^n rule
- S — Start symbol

eg1 $S \rightarrow aSb/\epsilon$

$L = \{\epsilon, asb, aaSbb, aaaSbbb \ldots\}$

$L = \{\epsilon, ab, aabb, aaabbb \ldots\}$

$L = \{a$ followed by equal no. of $b\}$

$= a^n b^n \quad n \geq 0$

= Is 'abab' part of this language L?

eg2

$S \rightarrow SS$

$S \rightarrow asb$

$S \rightarrow bsa$

$S \rightarrow \epsilon$

$L = \{\epsilon, asb, babsab, asbbsa \ldots$

$= \{\epsilon, ab, abab, abba \ldots\}$

$L =$ no. of a equal to no. of b in a string}

Page
Date

Chapter No 1 :- FINITE AUTOMATA.

Q1. What is Automata?

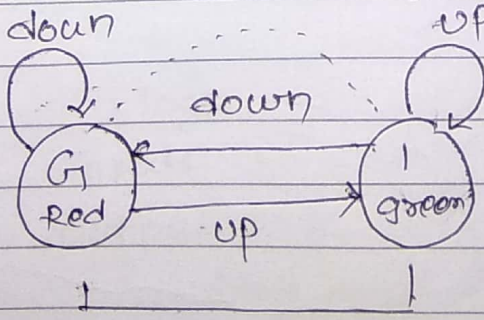
Ans. An automation is an abstract model of a digital computer.

1. An automation has a mechanism to read the input from a input tape.
2. A machine with intermediate stages which on recival of inputs choose a particular path from the initial stage to the final stage to produce a valid output, such a machine having finite number of intermediate stages is called finite automata or finite state machine.

* Basic concepts.

1. Alphabet :- set of input symbols eg Σ $\{0, 1\}$.
2. String :- sequential arrangement of ~~alphabet~~ input symbols.
3. Empty string :- ϵ or null
4. language :- collection of strings.
5. String length :-

* State Transition Digram



finite Automata.

Note :- Use theory in Each sum.

Page
Date

* Finite state Machine. (Theory)

1. FSM consists of finite set of states 'S' which alert on reciving the input set I to produce output set O

2. FSM can be mathamatically represented as follows.

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

where,

Q :- set of states

Σ :- Alphabetic set of input symbols

δ :- Transition function

$\delta - Q \times \Sigma \rightarrow Q$ -used to find next state.

q_0 :- Initial state.

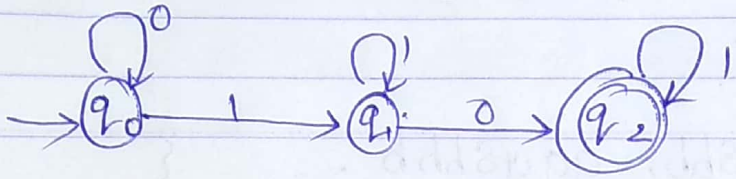
f - set of final states.

Finite automation (FA)

FA is a abstract computing device. It is a mathematical model of a system with discrete inputs, outputs states and set of transitions from state to state that occurs on ~~out~~ input symbols from alphabet Σ.

Its representations :-
- Graphical (Transition Diagram)
- Tabular (Trasition Table)
- Mathematical (Transition function)

① Transition Diag :-
It is a directed graph associated with vertices of the graph corresponds to the state of finite automata

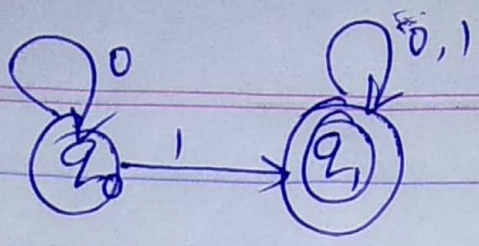


$\{0, 1\}$ – inputs
q_0 – initial state
q_1 – intermediate
q_2 – final.

② Transition Table — Tabular form

	Σ	Σ	Σ	Σ
q →				
q →				
q →				

$$\delta = Q \times \Sigma \rightarrow Q$$



.	0	1
q_0	q_0	q_1
q_1	q_1	q_1

Transition function. (δ)

- Two parameters are passed this function:-
  1. current state
  2. Input symbols
- The transition function returns a state which can be called as next state

δ(current state, current i/p) = next state

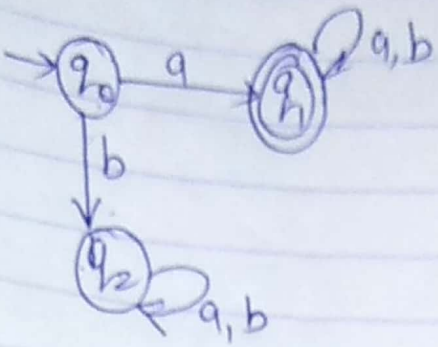
eg $\delta(q_0, 0) = q_1$

or

$\delta(q_0, 1) = q_1$

L = String starting with a for $\Sigma^*(a, b)$

$L = \{a, aa, abb, aaa, abbb \ldots, ba\}$



Σ / q_0	a	b
$\rightarrow q_0$	q_1	q_2
q_1	q_1	q_1
q_2	q_2	q_2

abb

$\delta(q_0, a) = q_1$

$\delta(q_1, b) = q_1$

$\delta(q_1, b) = (q_1)$ ← final state

String abb belongs to L

bab

$\delta(q_0, b) = q_2$

$\delta(q_2, a) = q_2$

$\delta(q_2, b) = (q_2)$ ← non final state

bab does not belong to L

Questions (Practice)

1. String containing 'a'
2. string ends with 'a'
3. String starting with 'a' & ends with 'b'
4. " must start with substring w
   1. w = ba 2. w = abb
5. every string must end with substring w
   1. w = bb 2. ab 3. bab
6. Design a DFA over $\Sigma = \{a, b\}$ such that every string accepted must contain a substring w
   1. w = ad 2. w = ba 3. w = abb
7. Design a DFA over $\Sigma = (a, b)$ such that every string
   1. must start & end with same symbol
   2. must start & end with diff. symbol
   3. must start & end with w = ab

(8) Design DFA over $\Sigma = (a, b)$ such that every string accepted must

(a) $|w| = 2$

(b) $|w| \geq 2$

(c) $|w| \leq 2$

(d) $|w|_a = 2$

(e) $|w|_a \geq 2$

(f) $|w|_a \leq 2$

To design FSM/FA we always following steps:-

1. Logic
2. Formal Notations
3. Transition Function
4. Testing the automata.

Q1 Design a FSM to check if the given number (decimal) is divisible by 3 or

⇒ ~~FSM consists of~~ Definition of FSM

1. Logic : $I = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

$O = \{Y, N\}$

$S = \{q_0, q_1, q_2\}$.

(n mod 3) $q_0 \rightarrow n \bmod 3 = 0$

$q_1 \rightarrow n \bmod 3 = 1$

$q_2 \rightarrow n \bmod 3 = 2$

$0/3 = 3/3 = 6/3 = 0$ remainder $\rightarrow q_0$

$1/3 = 4/3 = 7/3 = 1$ " $\rightarrow q_1$

$2/3 = 5/3 = 8/3 = 2$ " $\rightarrow q_2$

Q = Set of all states = $\{q_0, q_1, q_2\}$

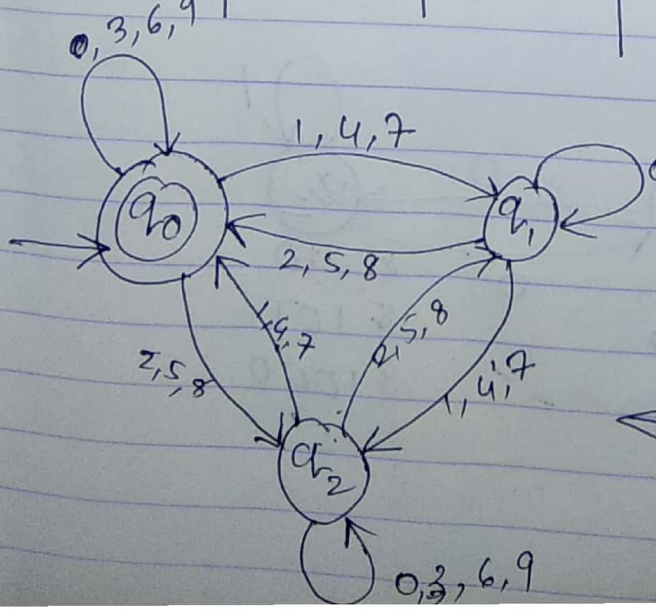
Σ = Set of all I/P symbols = $\{0,1,2,3,4,5,6,7,8,9\}$

δ = Delta = step 3 = Transition function (table/diag)

q_0 = start state = q_0

F = Set of all final states = $\{q_0\}$.

Q \ Σ	0, 3, 6, 9	1, 4, 7	2, 5, 8
rem 0 $*(q_0)$	q_0	q_1	q_2
rem 1 q_1	q_1	q_2	q_0
rem 2 q_2	q_2	q_0	q_1



Transition Diagram

$(q_0, 369)$

$\vdash (q_0, 69)$

$\vdash (q_0, 9)$

$\vdash (q_0, 0)$

$\vdash (q_0, \epsilon) = y$

$(q_0, 247)$

$\vdash (q_2, 47)$

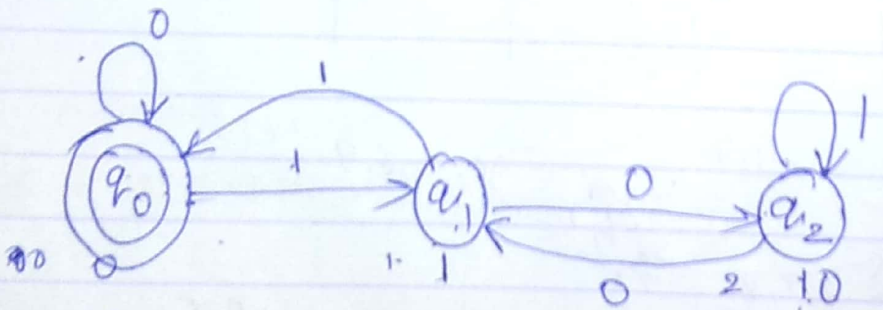
$\vdash (q_0, 7)$

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

0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10

q_0

10 100



0 1 10

3 11 4 100 5 101

6 110 7 111 8 1000

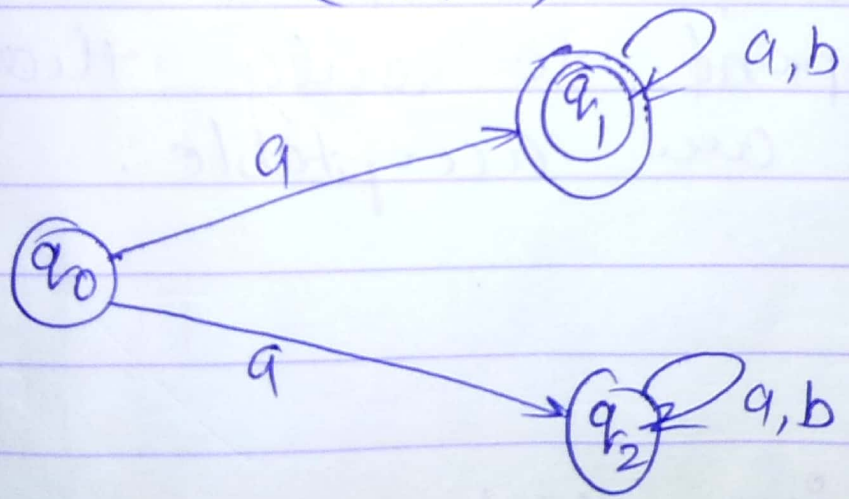
DFA = $\delta(q_0, a) = q_1$ but

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

NFA

Acceptance of NFA :- A string ω is said to be accepted by a NFA if there exist at least one transition path on which we start at initial state and ends at final state.

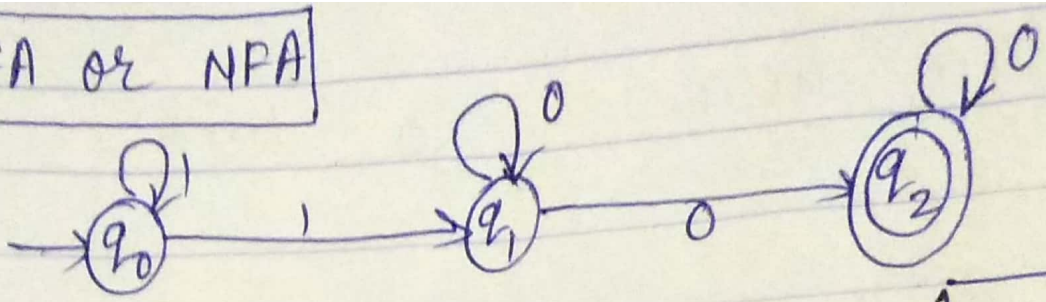
$$\delta^*(q_0, \omega) = F$$



properties of NFA

1. can have more than one transition for one symbol
2. Not necessary to do transition for every symbol
3. NO dead state.
4. Practical Implementation is not possible
5. No need to complete NFA
6. ~~Ate~~ Need to respond to only those string which are acceptable.

NDFA or NFA



$\Sigma = (0,1)$

DFA

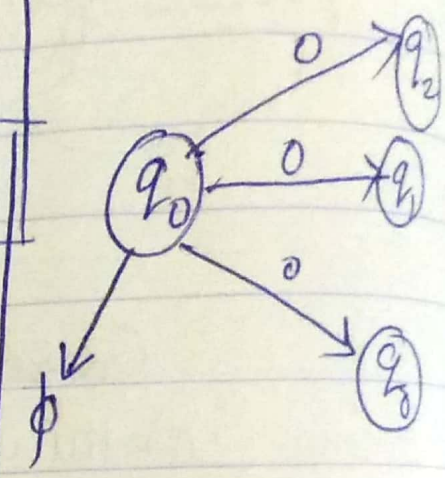
$\delta : Q \times \Sigma = Q$

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

NFA

$\delta : Q \times \Sigma = 2^Q = 2^3 = 8$

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



Q1. Design a FSM to check whether a given decimal number is divisible by 4.

Solution:- step 1: Defination & fram behind.

~~step 2~~: Theory

Step 2:- logic

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

Output $O = \{Yes, No\}$.

		Remainder.	valid inputs.
inputs with remainder 0,	q_0	0	$\{0,4,8\}$.
	q_1	1	$\{1,5,9\}$.
	q_2	2	$\{2,6\}$.
	q_3	3	$\{3,7\}$.

Step 3:- State Transition Function :- (In tabular form)

	q \ Σ	0,4,8	1,5,9	2,6	3,7
		0	1	2	3
E	q_5	q_0	q_1	q_2	q_3
0	q_0	q_0	q_1	q_2	q_3
remainders ⇒ 1	q_1	q_2	q_3	q_0	q_1
2	q_2	q_0	q_1	q_2	q_3
3	q_3	q_2	q_3	q_0	q_1

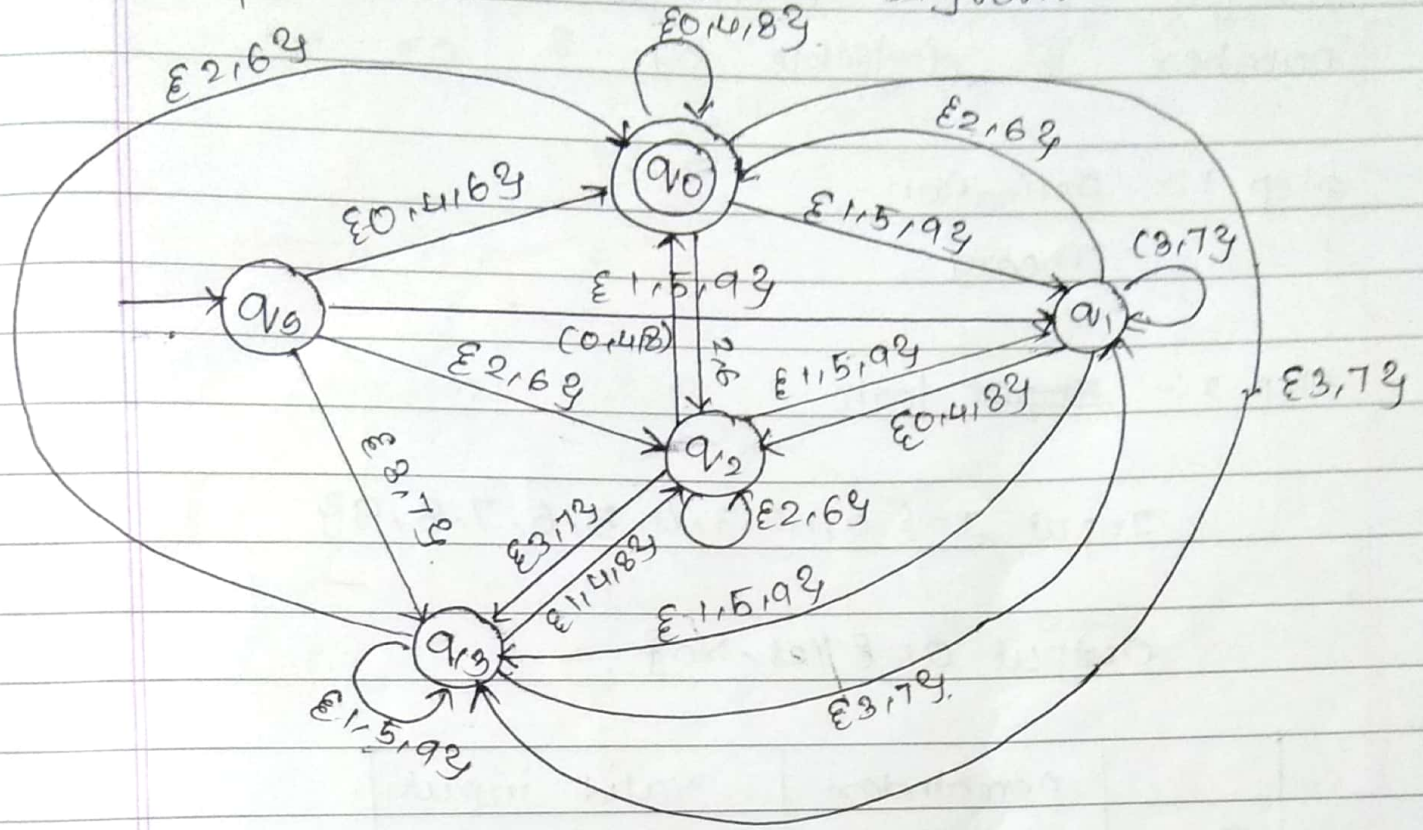
for 0 $00\%4=0$ $01\%4=1$ & so on
for 1 $10\%4=2$ $11\%4=1$ & so on.
for 2. $20\%4=0$ $21\%4=1$ & so on

Step 4:- Where the ~~state~~ given condition satisfies put Yes else No.
in this case put Yes for q_0 only.

MAF

q \ Σ	0,4,8	1,5,9	2,6	3,7
q_5	Yes	No	No	No
q_0	Yes	No	No	No
q_1	No	No	Yes	No
q_2	Yes	No	No	No
q_3	No	No	Yes	No.

Step 5 :- State Transition Digram



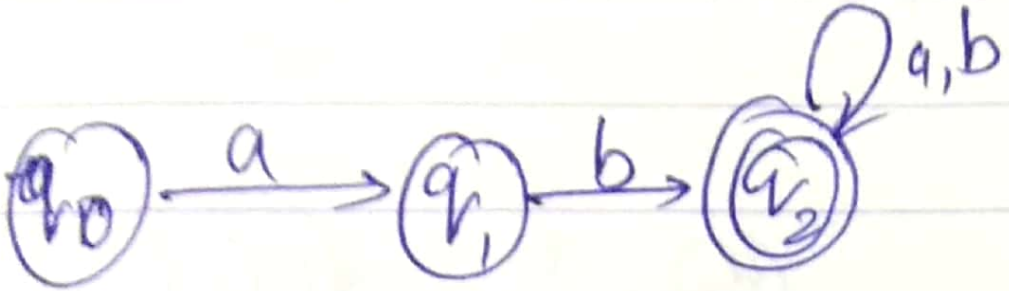
Step 6 :- Example.

$\delta(q_{vs}, 52)$

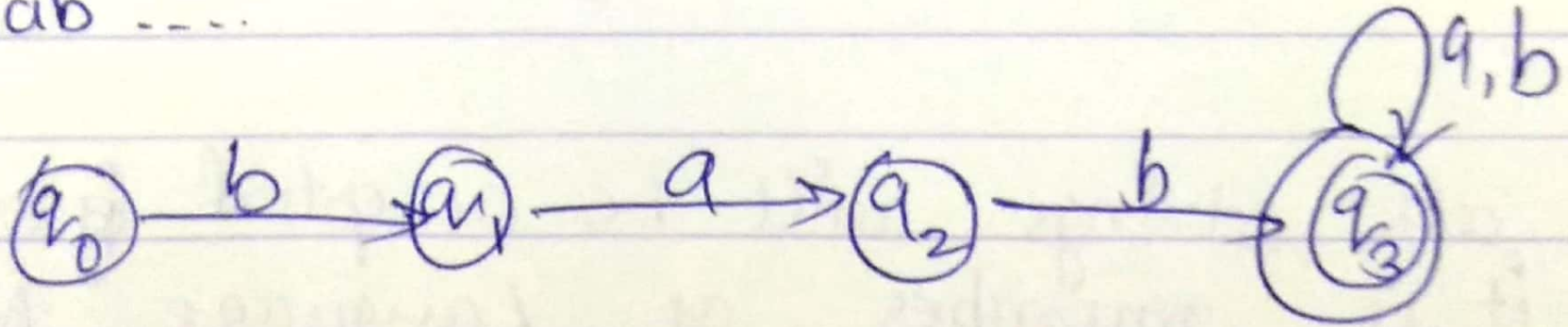
$\delta(q_1, 2)$

(q_0, ϵ) Hence the number is divisble.

① ab* ---- bab ---

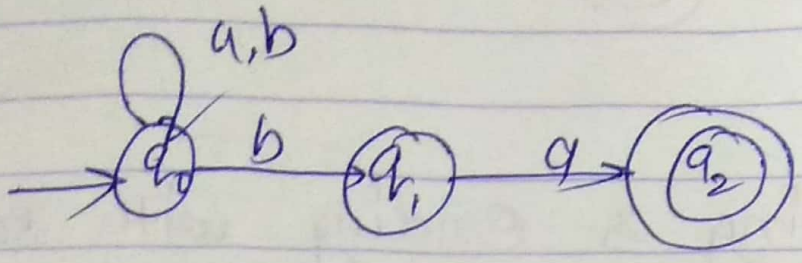


② bab ----

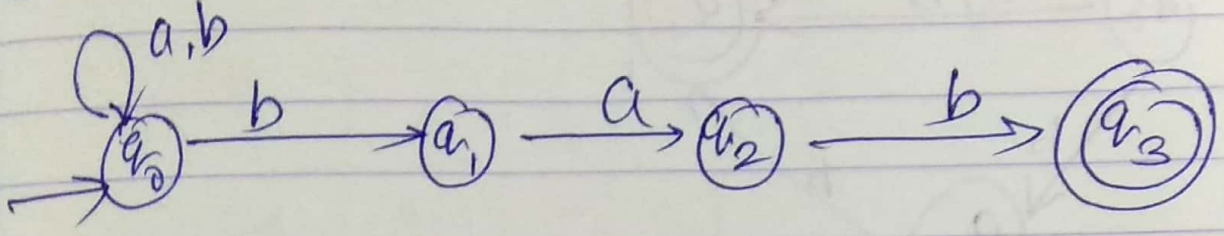


Design NFA over $\Sigma = \{a, b\}$ such the every string ends with

① ... ba

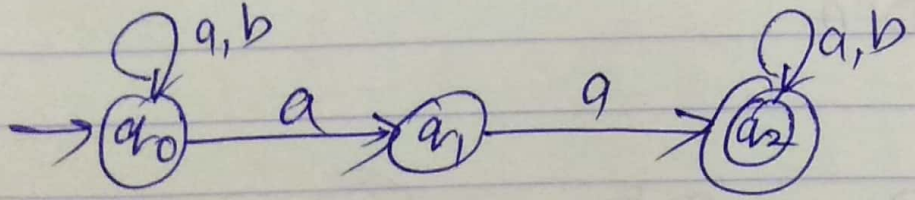


② .... bab



Design NFA $\Sigma = \{a, b\}$ such that every string accepted contain a substring:-

① ....aa....



② .... abb...

* Rules to convert RE into NFA (Thompson's Rule)

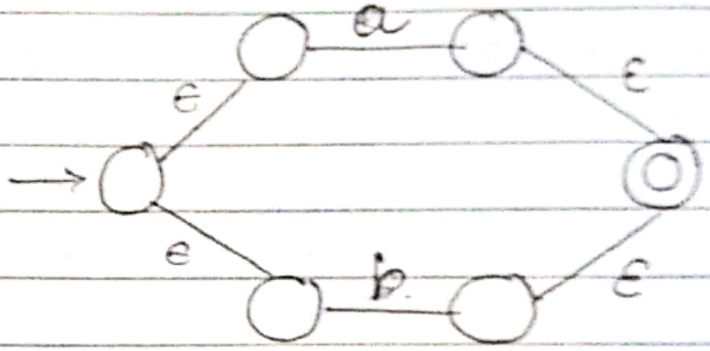
1. $r = \phi$ ⟶ 

2. $r = \epsilon$ ⟶ 

3. $r = a$ ⟶ 

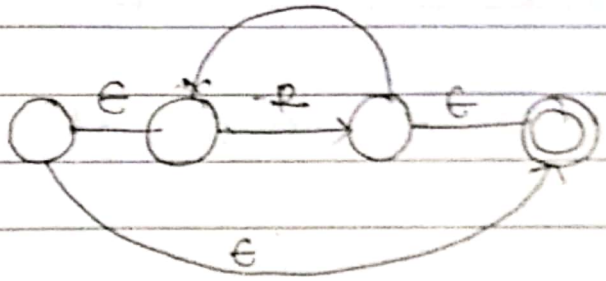
4. $r = R|S$ or $R+S$

$r = a+b$

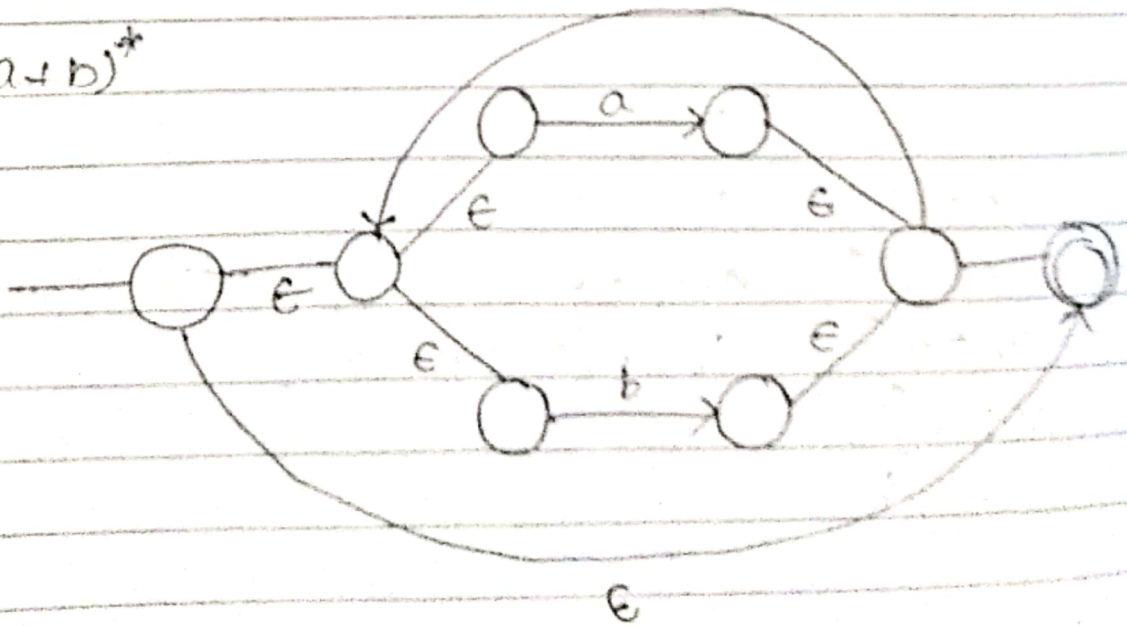


5. $r = R^*$

eg: $r = a^*$

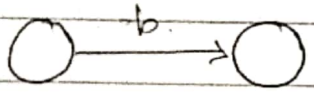


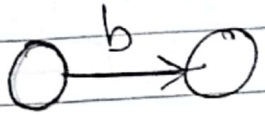
6. $r = (a+b)^*$



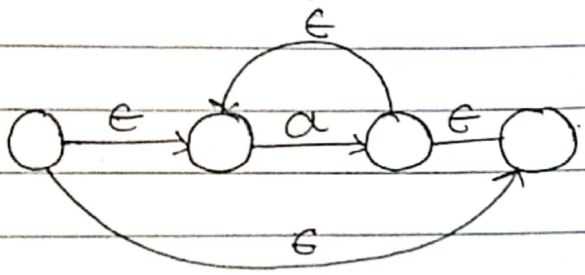
Q.1. Construct NFA for RE = b + ba*

Solution :- Step 1 :- Representing b.



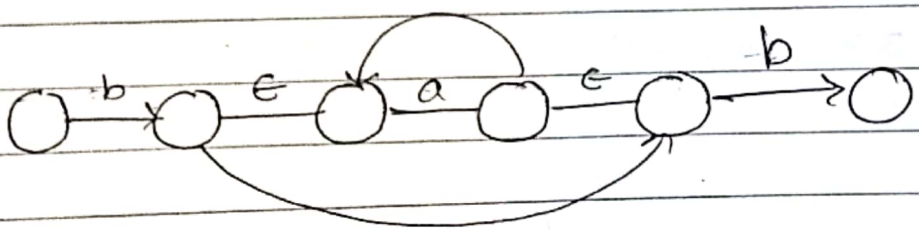


Step 2 :- Representing a*

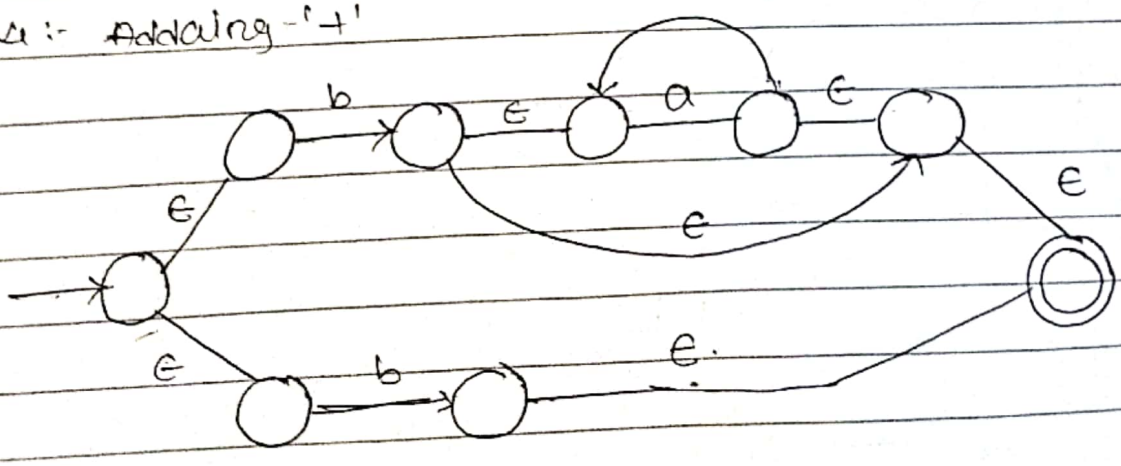


Step 3 :- Representing ba*

a*b.

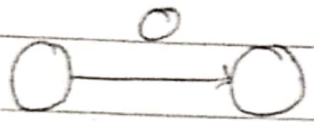


Step 4 :- Adding '+'

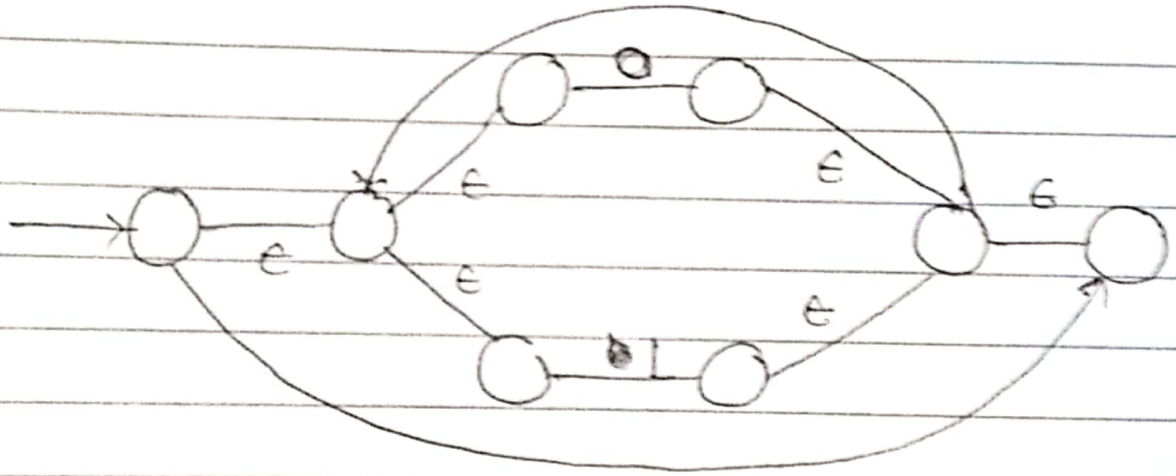


Q.2. Construct NFA for $r = 0 \cdot (0+1)^* \cdot 10$

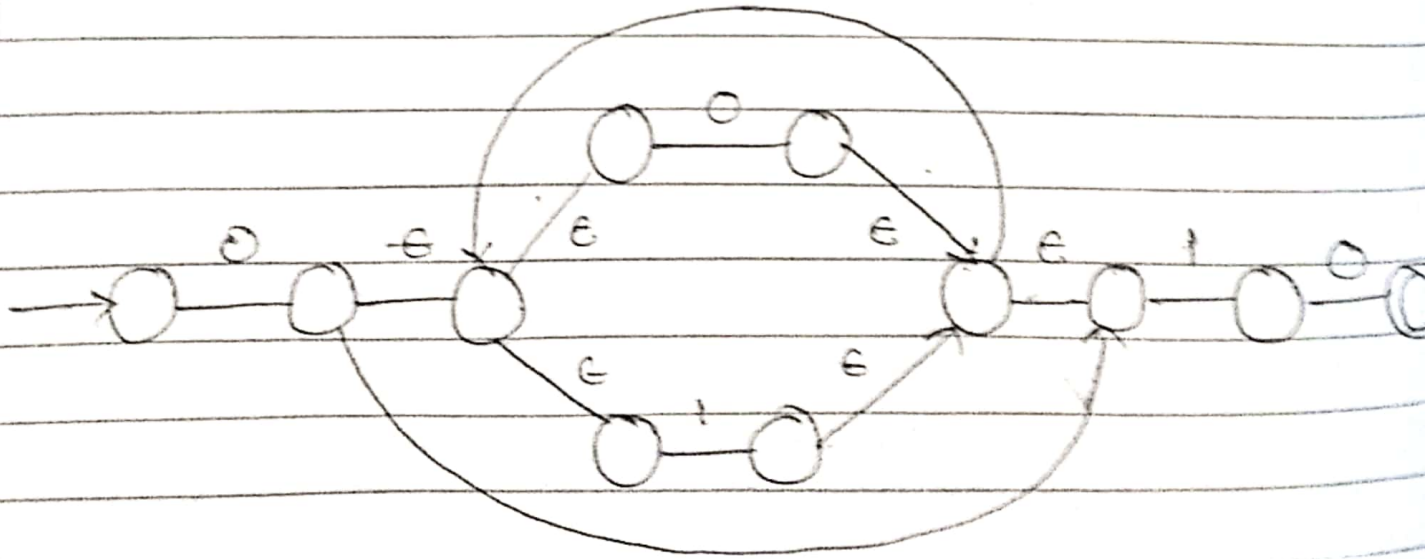
sol: step 1:- representing 0.



step 2:- Representing $(0+1)^*$

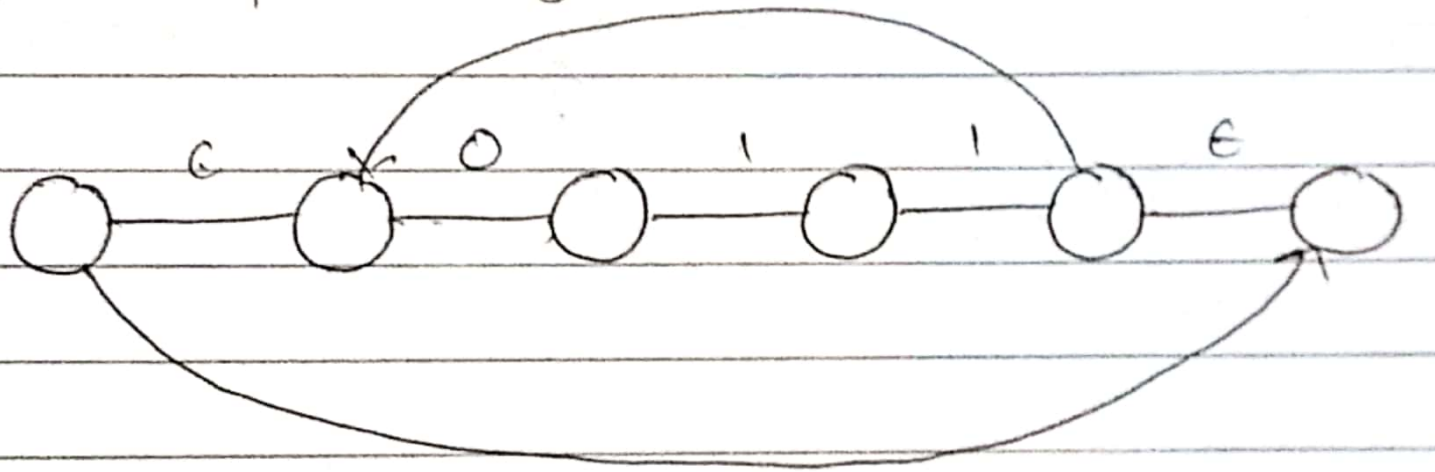


step 3:- Representing $0 \cdot (0+1)^* \cdot 10$

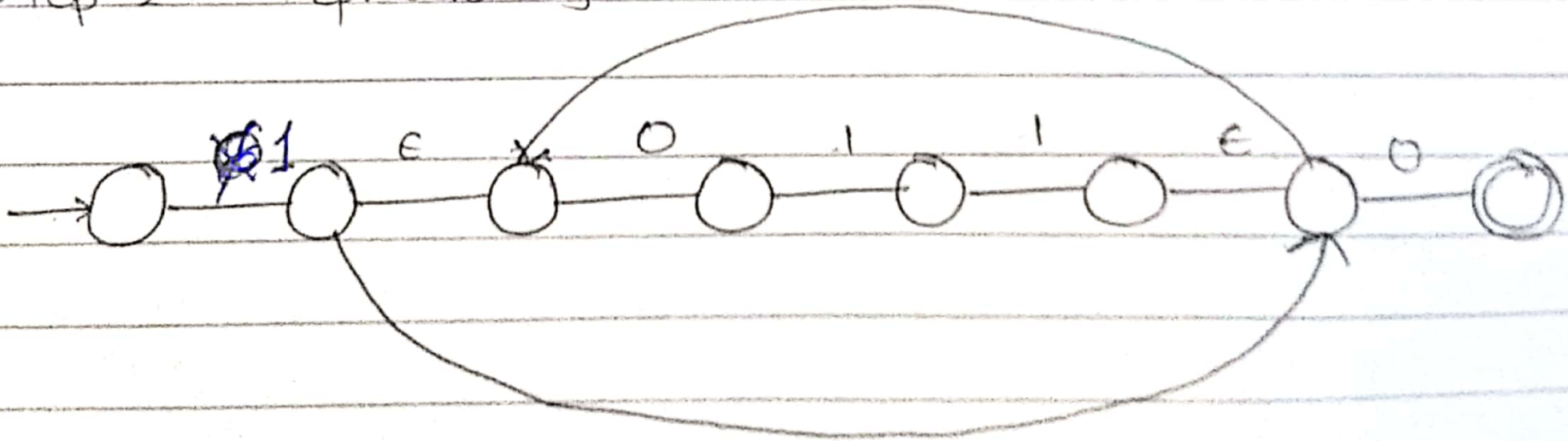


3. Design an NFA RE 1(011)*0

: step 1:- Representing (011)*1

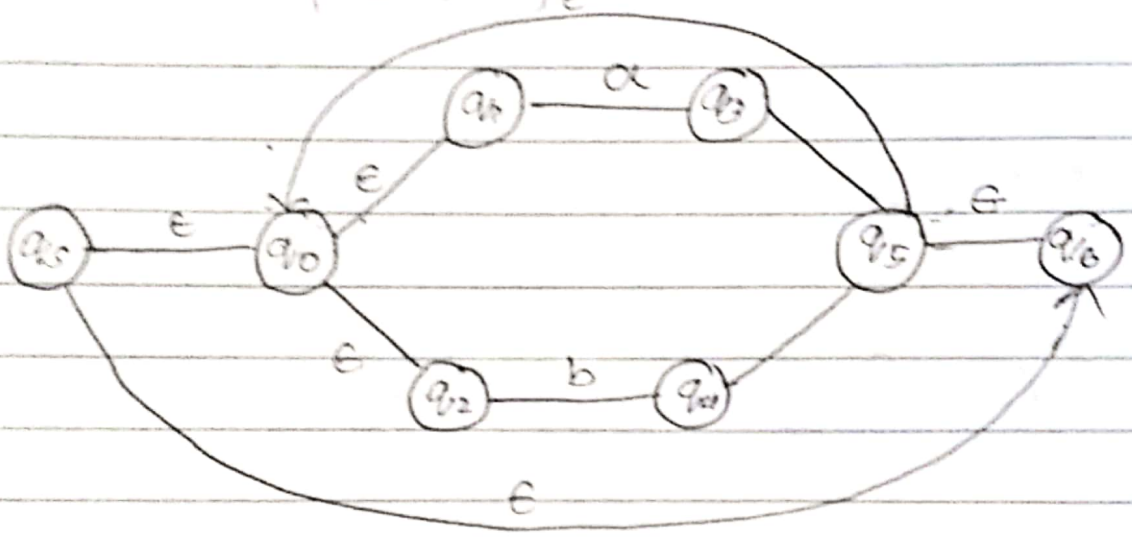


step 2:- Representing 1(011)*0

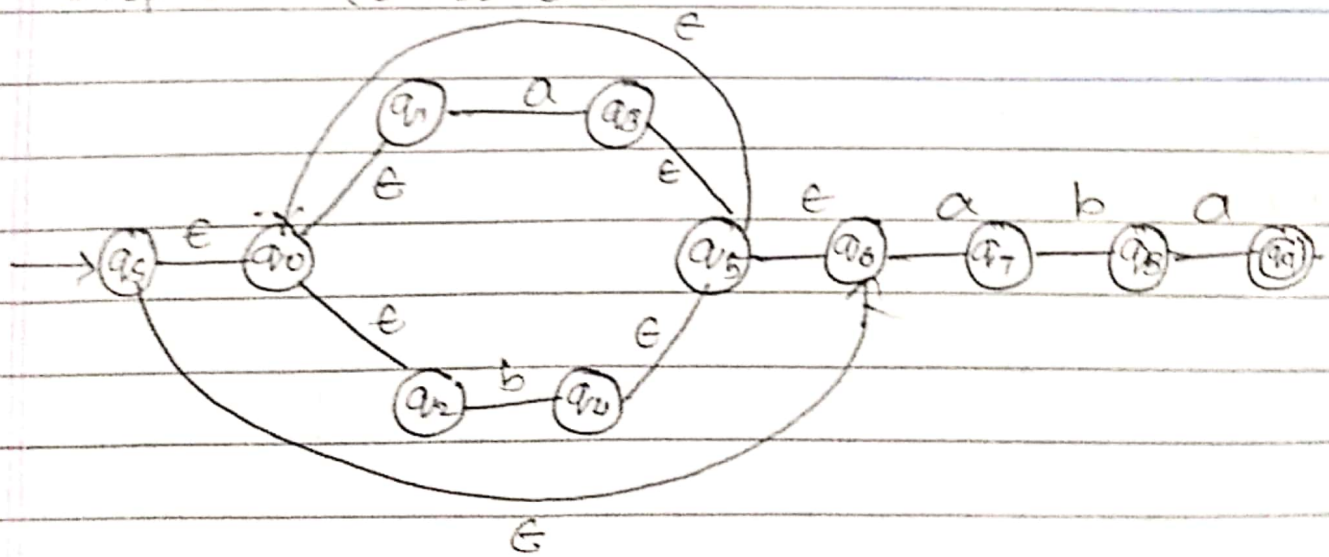


Q.3 Design an NFA for RE $(a+b)^* abb$.

Ans: Step 1:- Representing $(a+b)^*$.



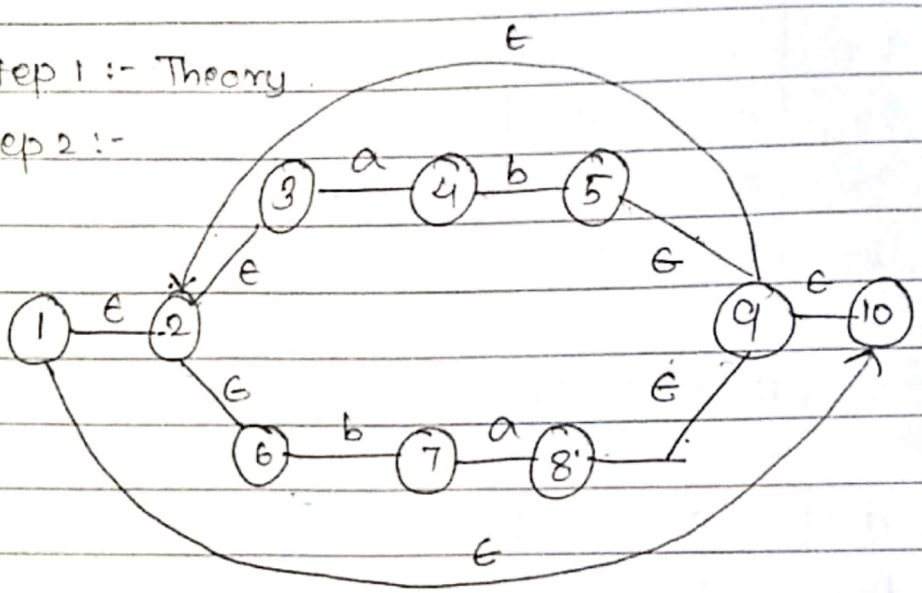
Step 2:- $(a+b)^* abb$.



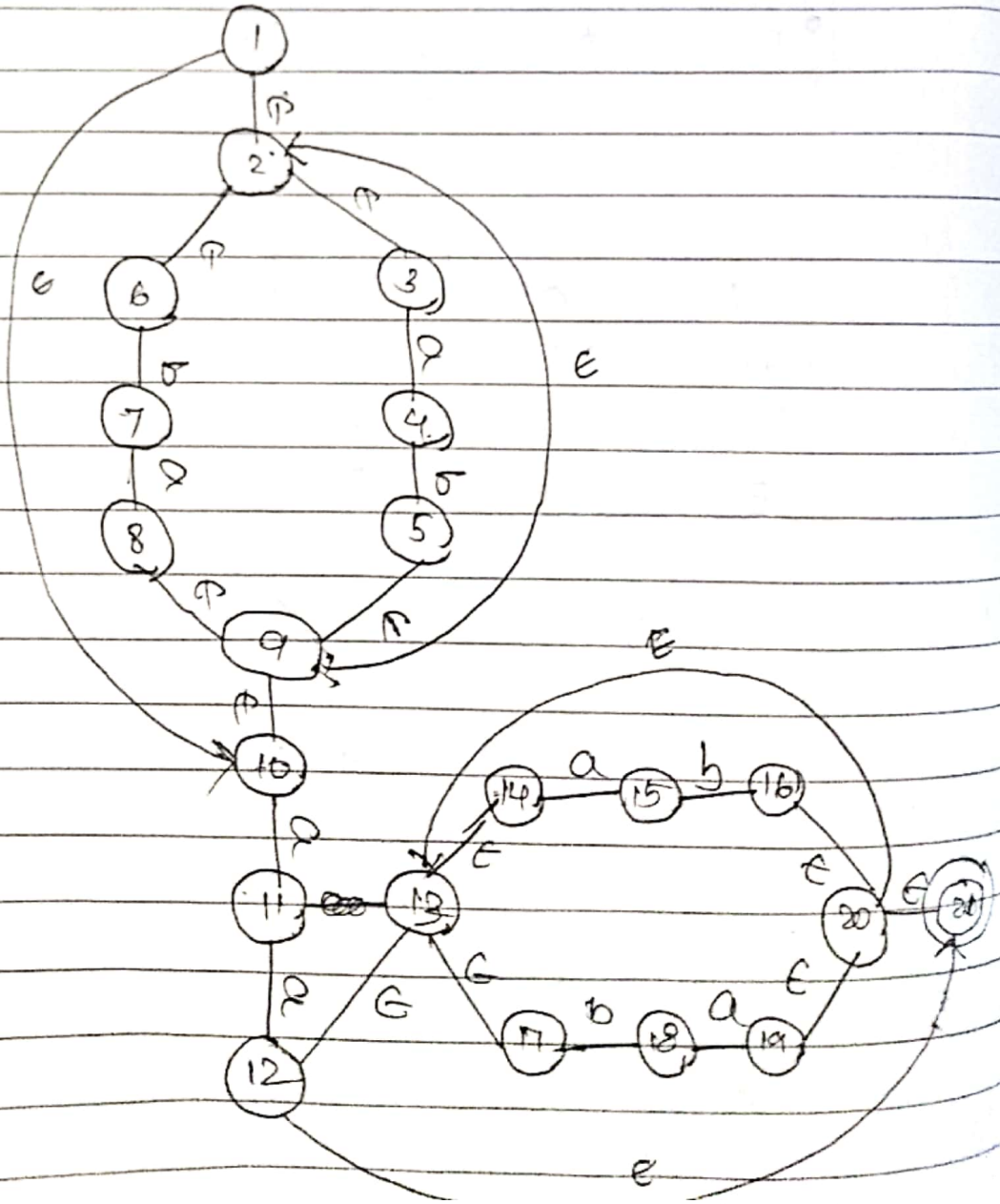
by (a|ba)* aa (ab|ba)*

Solution :- step 1 :- Theory

step 2 :-

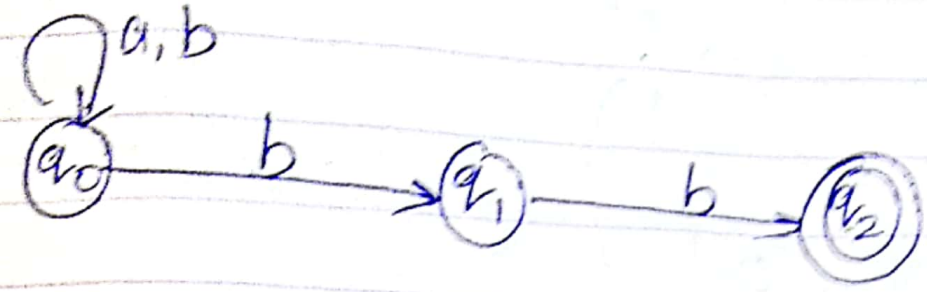


Representing (ab+ba)*aa(ab+ba)*



NFA to DFA conversion

1)



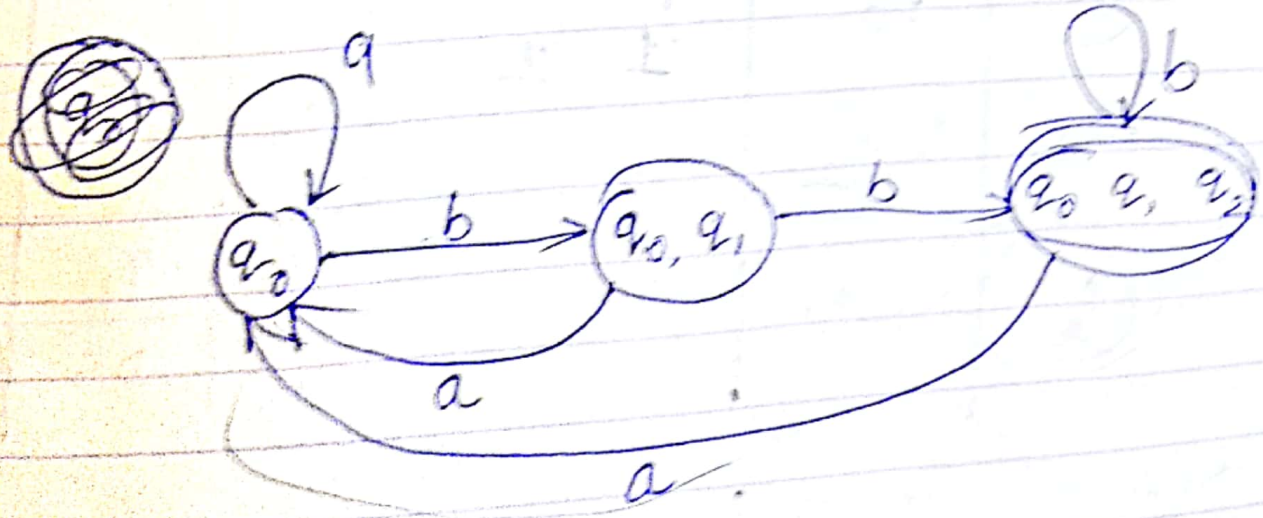
NFA

	a	b
$\rightarrow q_0$	q_0	$q_0\ q_1$
q_1	—	q_2
(q_2)	—	—

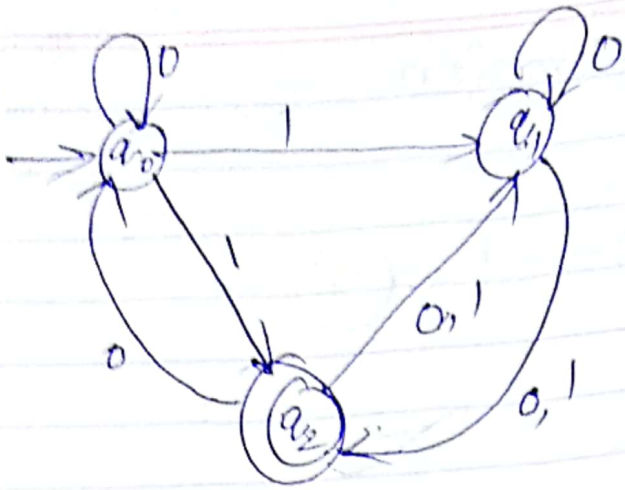
↓

DFA

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



(i)



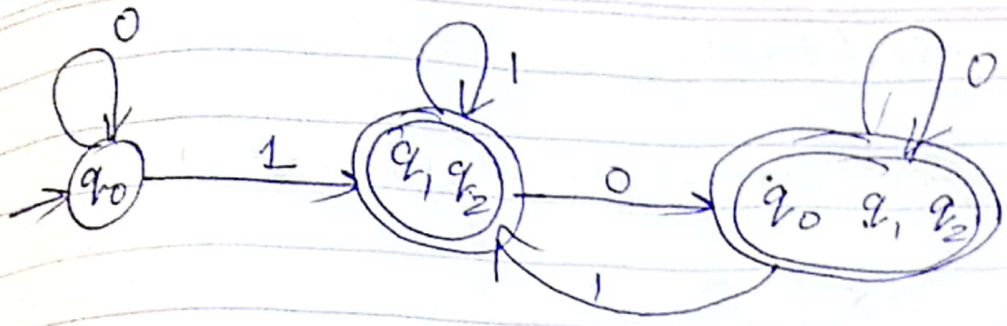
	0	1
q_0	q_0	q_1, q_2
$\{q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_0, q_1, q_2\}$
$\{q_0, q_1, q_2\}$		

NFA

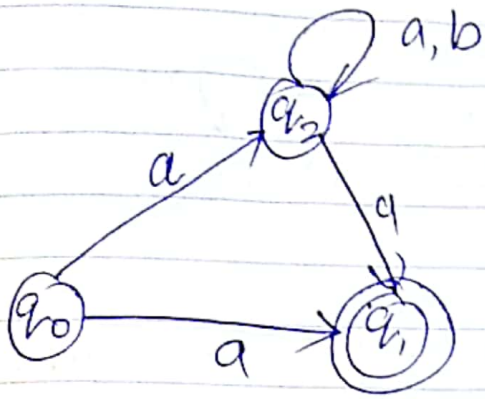
	0	1
q_0	q_0	q_1 q_2
q_1	q_1 q_2	q_2
q_2	q_0 q_1	q_1

DFA

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

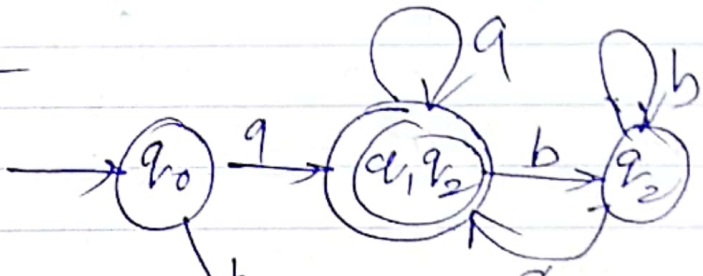


(3)



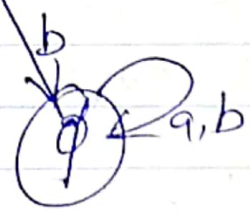
NFA

	a	b
q_0	q_1, q_2	—
q_1	—	—
q_2	q_2, q_1	q_2

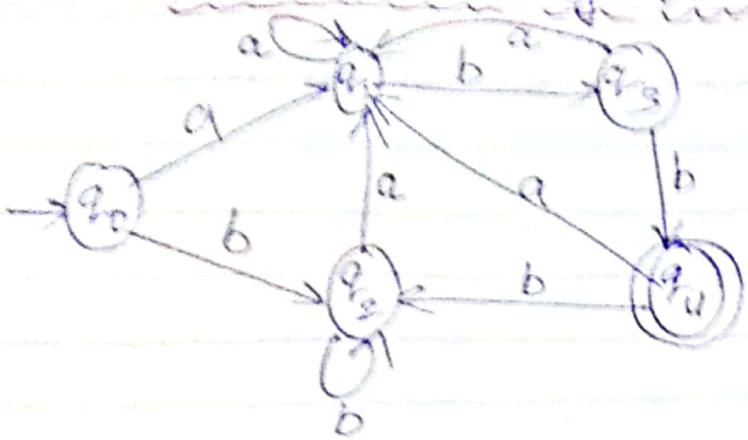


DFA

	a	b
q_0	$[q_1 q_2]$	$\{\phi\}$ ← dead state
$[q_1 q_2]$	$[q_1 q_2]$	$[q_2]$
$[q_2]$	$[q_1 q_2]$	q_2
ϕ	ϕ	ϕ



Minimization of DFA



① Delete all states which cannot be reached from initial state ② Divide in to final & non final states

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

0 equivalent state

$[q_0 \; q_1 \; q_2 \; q_3]$ $[q_4]$

1 equivalent

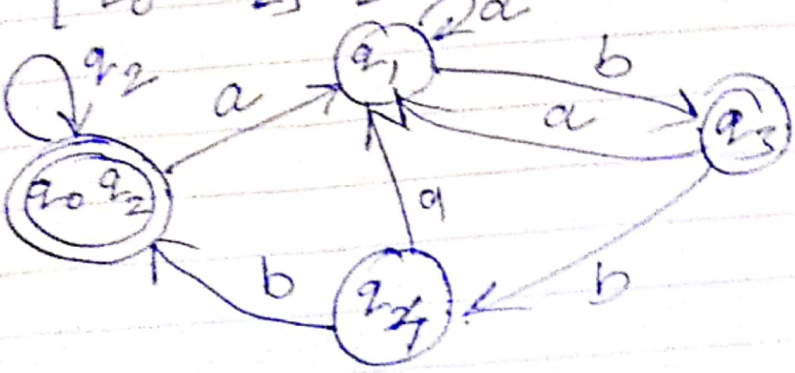
$[q_0 \; q_1 \; q_2]$ $[q_3]$ $[q_4]$

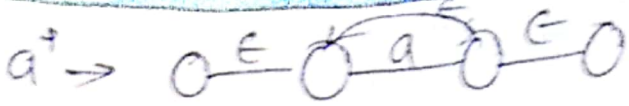
2 equivalent

$[q_0 \; q_2]$ $[q_1]$ $[q_3]$ $[q_4]$

3 equivalent

$[q_0 \; q_2]$ $[q_1]$ $[q_3]$ $[q_4]$





Assignment -3

① Design NFA $\Sigma = (a, b)$ such that every string accepted contains a substring 'abb'

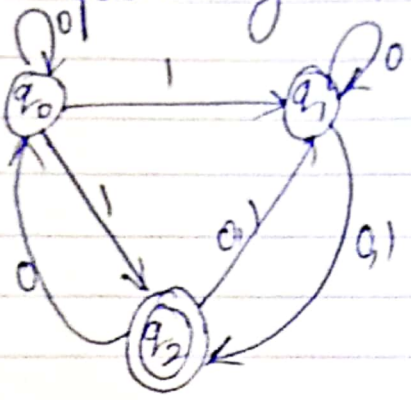
② Design NFA $\Sigma = (0, b)$ such that every string starts and ends with same symbol

③ Convert the following RE to ε-NFA

1. $(00)^* \cdot 1 \cdot (11)^*$
2. $(00+11)^*$
3. $a^* + b^*$
4. $b + ba^*$
5. $(0+1)^* \cdot (00+11)$
6. $(a+b)^* \cdot b \cdot c^+$

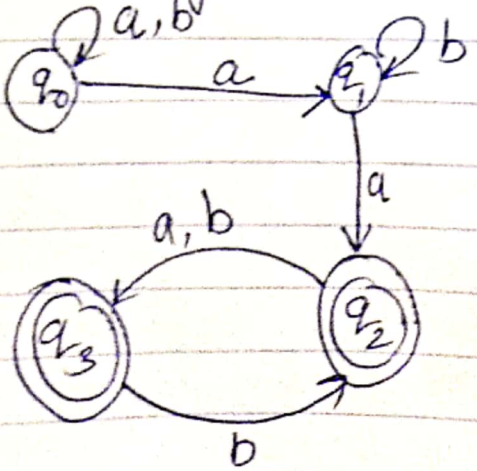
④ Convert following NFA to DFA

①

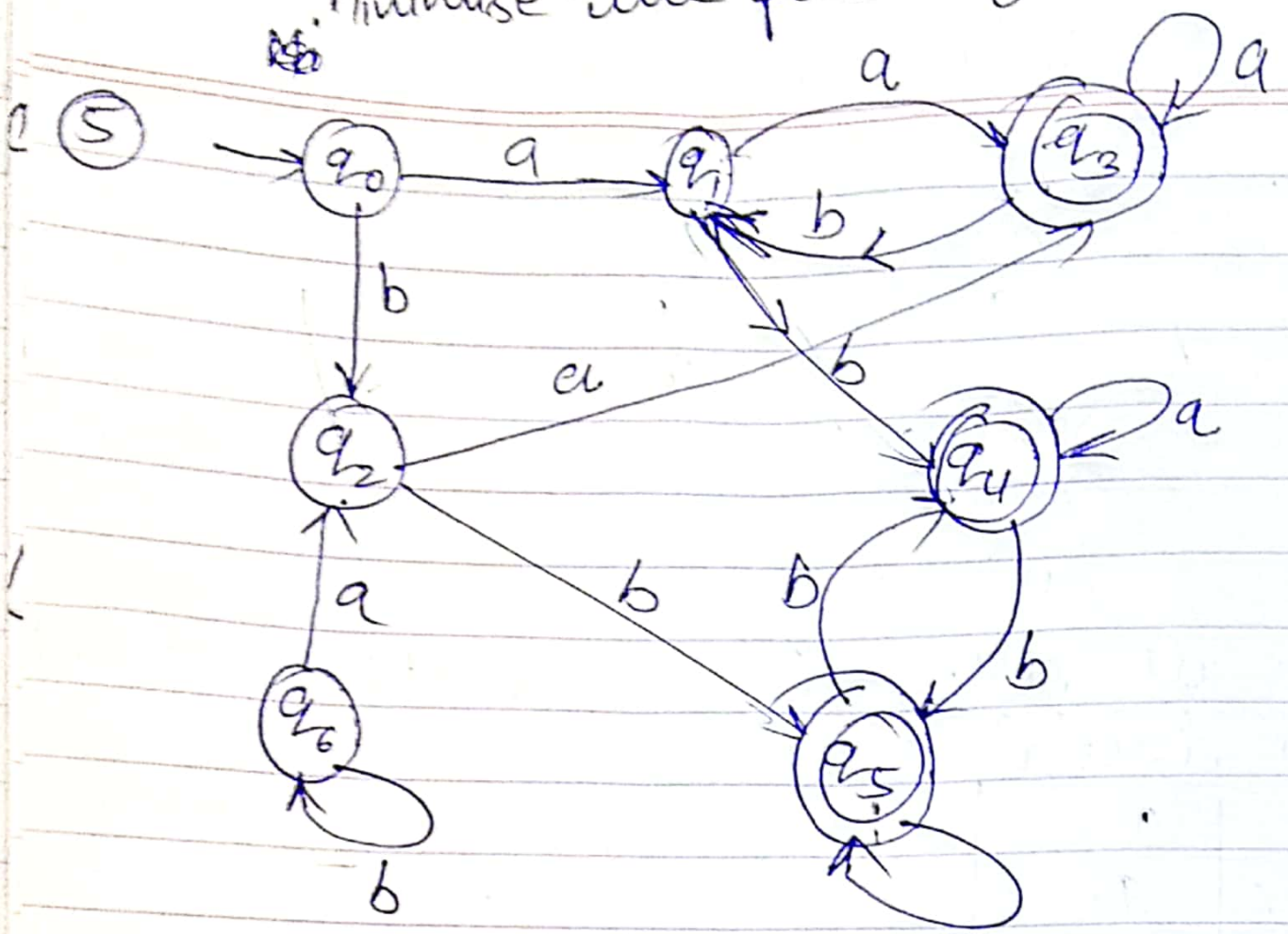


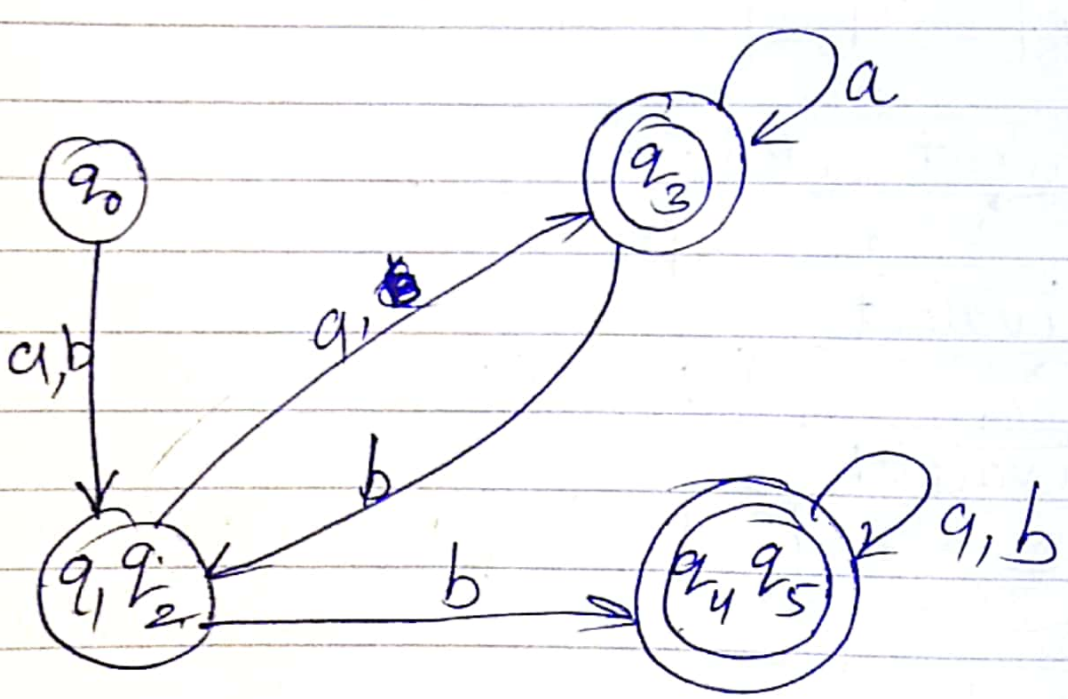
② L = {string starts with a}

③



Minimise the following DFA

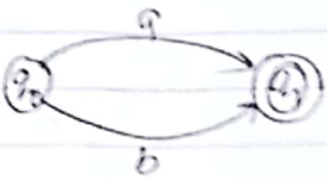




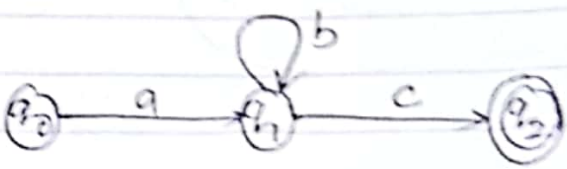
FA → RE

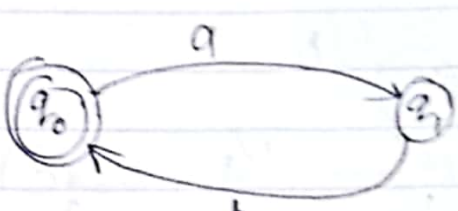
(1) 

⇒

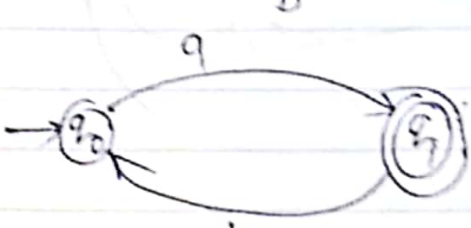
(2) 

⇒

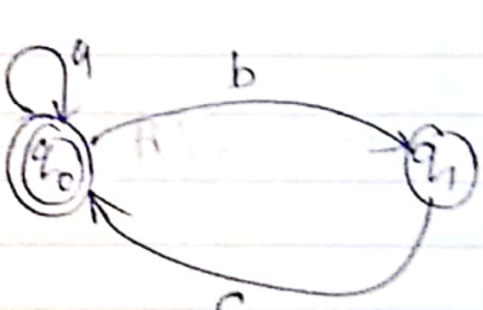
(3) 

(4) 

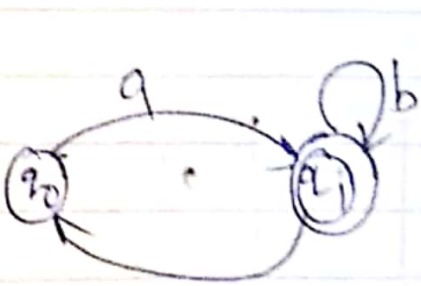
⇒

(5) 

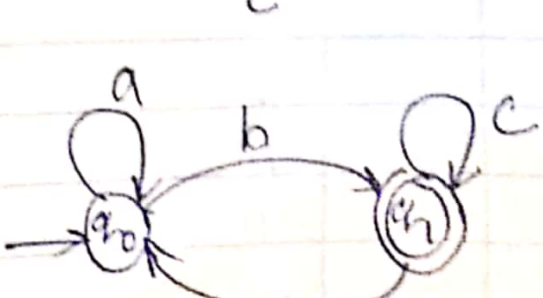
⇒

(6) 

⇒
⇒

(7) 

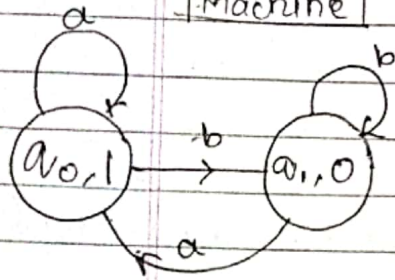
⇒

(8) 

* Finite Automata with output.

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

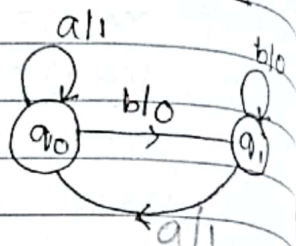
Moore Machine



Q - finite set of states.
Σ - input alphabet.
δ - Transition function
q_0 - initial state
Δ - output alphabet
λ - output function.

$\lambda : Q \rightarrow \Delta$
$q_0 \rightarrow 1$
$q_1 \rightarrow 0$

Mealy Machine



$\lambda : Q \times \Sigma \rightarrow \Delta$
$(q_0, a) \rightarrow 1$
$(q_0, b) \rightarrow 0$
$(q_1, b) \rightarrow 0$
$(q_1, a) \rightarrow 1$

Q.1. Difference between

Ans

Moore	Mealy
1. In moore machine state gives the output.	In Mealy machine transition gives the output.
2. Output is dependent on the state	Output depends on state and the input
3. Ouput mapping is defined as, $Q \rightarrow \Delta$	Output mapping is defined by $Q \times \Sigma \rightarrow \Delta$.
4. If length of the input sequence/symbol is 'n' then the length of oupt sequence is 'n+1'	If length of input sequence is then length of output sequence is 'n'.
5. Output can be obtained on ϵ	Output cannot be obtained on ϵ

Mealy & Moore

baa

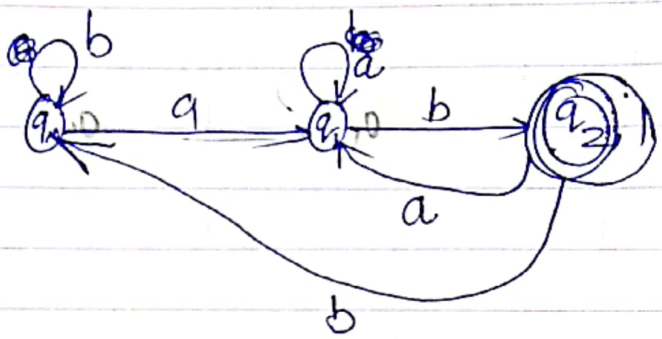
Construct a moore m/c that takes set of all string over {a, b} as i/p & prints '1' as o/p for every occurance of (ab) as a substring.

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

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

ab

ab ab ab

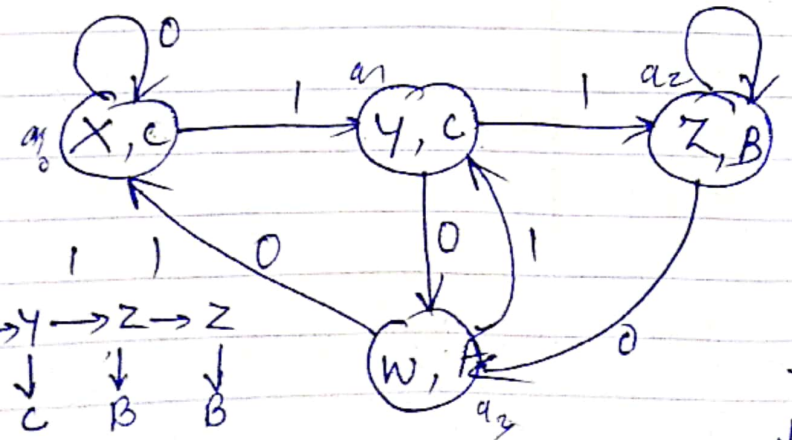


Q Construct a moore m/c that takes set of string over {0, 1} & produces A as o/p if i/p ends with '10' or produces 'B' as if i/p ends with '11' otherwise produces C

$\Sigma = \{0, 1\}$ $\Delta = \{A, B, C\}$ 10 → A

11 → B

$Q = \{X, Y, Z\}$



1 1 1 1 0

X → Y → Z → Z → Z

C C B B

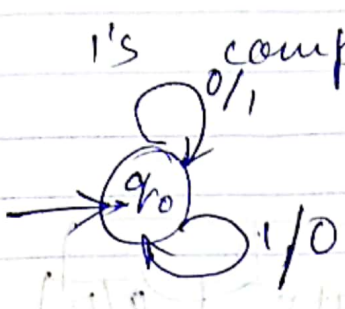
1 1 0

X → Y → Z → W

C B A

Mealy M/c

Construct a mealy m/c that takes binary number as i/p & produces 2's complement of that number as o/p Assume the string is read from LSB to MSB & end carry is discarded.

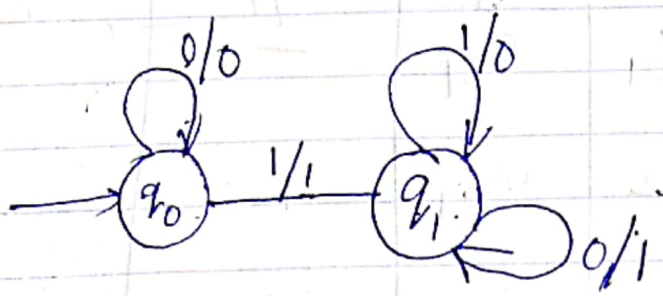


eg

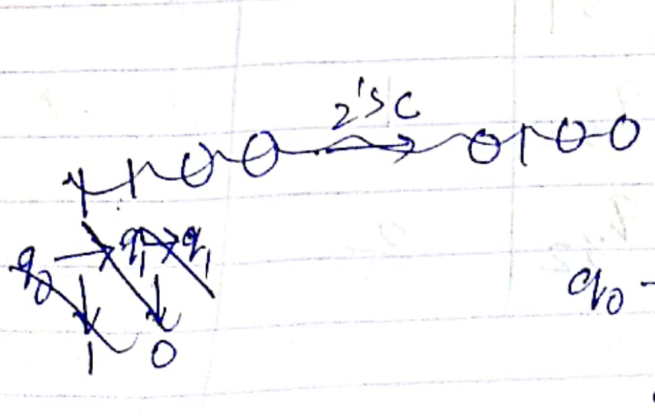
10 11
01 00
1
01 01

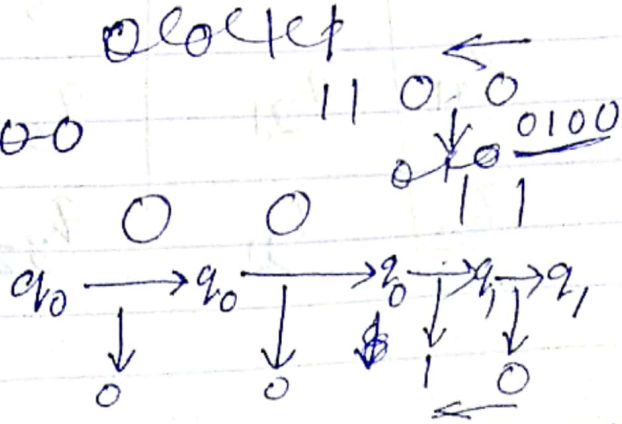
11 00
00 11
1
0100

11101 00
000100 11
1
00010 100

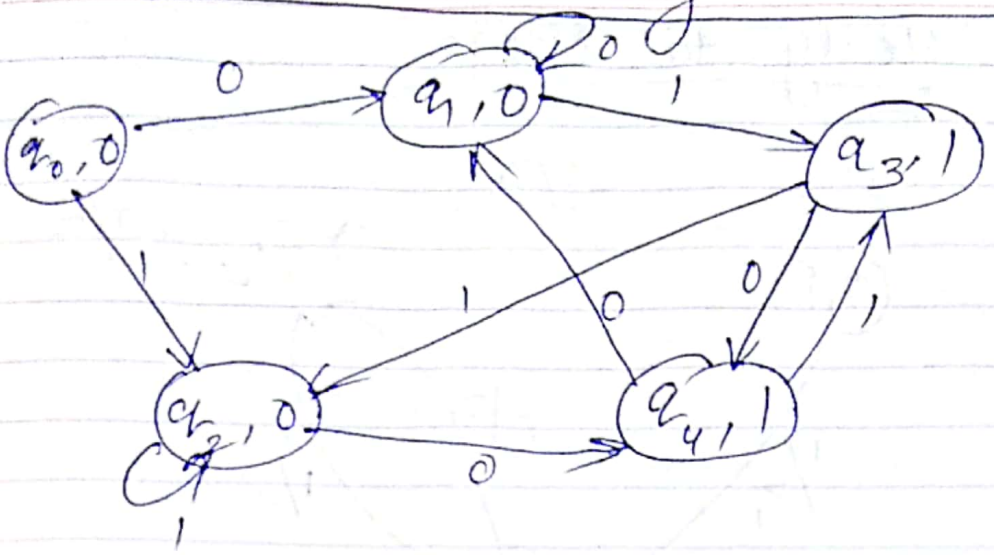


	0		1	
	State	o/p	State	o/p
q_0	q_0	0	q_1	1
q_1	q_1	0	q_1	0



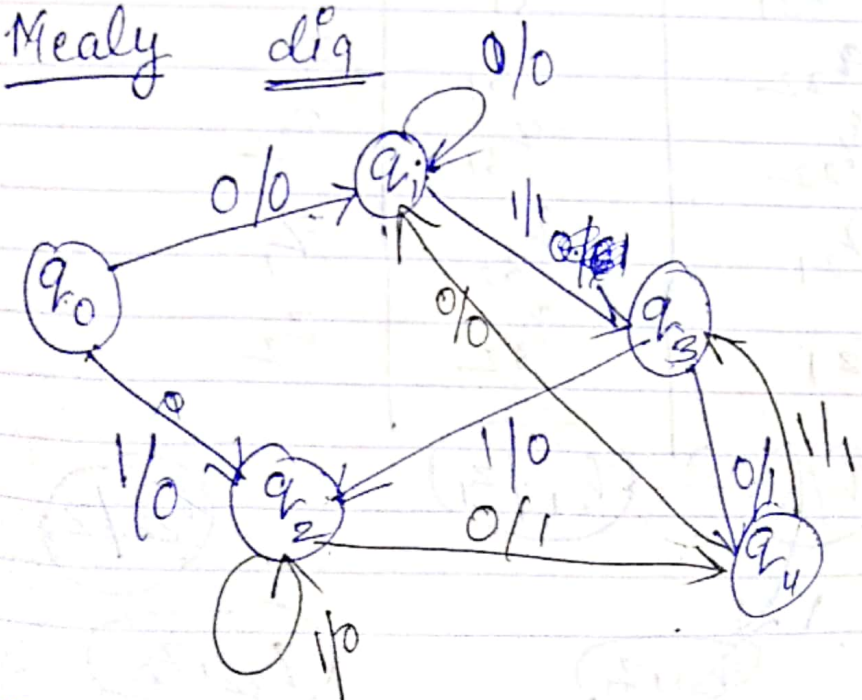


Moore to Mealy conversion

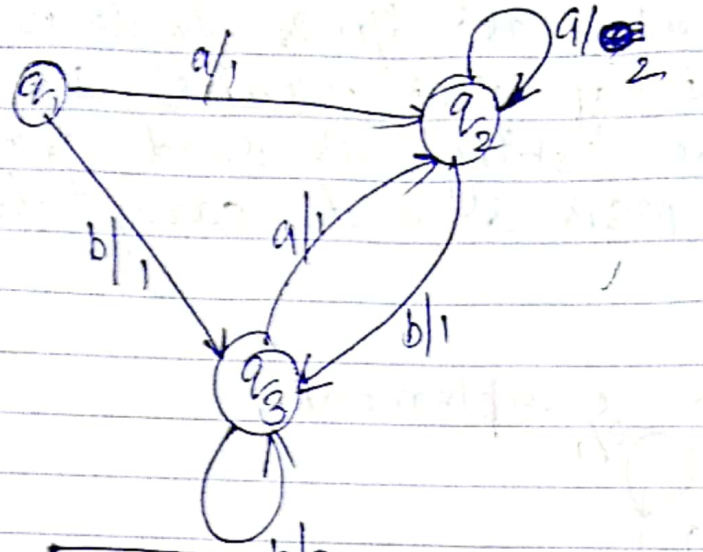


State	0	1	o/p
q_0	q_1	q_2	0
q_1	q_3	q_1	0
q_2	q_2	q_4	0
q_3	q_4	q_2	1
q_4	q_1	q_3	1

Mealy dig



Mealy to Moore M/C

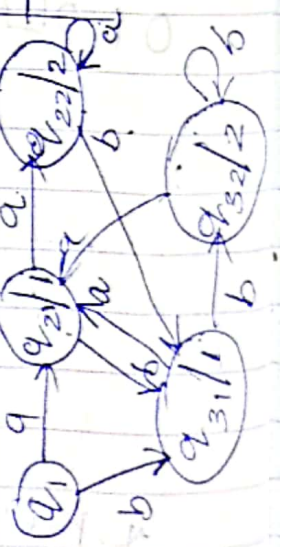


	a		b	
	state	o/p	state	o/p
q_1	q_2	1	q_3	1
q_2	q_{21} q_{22}	2	q_3	1
q_3	q_{31} q_{32}	1	q_3	2

$q_2 \to q_{21}-1$, $q_{22}-2$

$q_3 \to q_{31}-1$, $q_{32}-2$

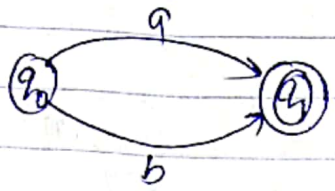
	a	b	o/p
q_1	q_{21}	q_{31}	—
q_{21}	q_{22}	q_{31}	1
q_{22}	q_{22}	q_{31}	2
q_{31}	q_{21}	q_{32}	1
q_{32}	q_{21}	q_{32}	2



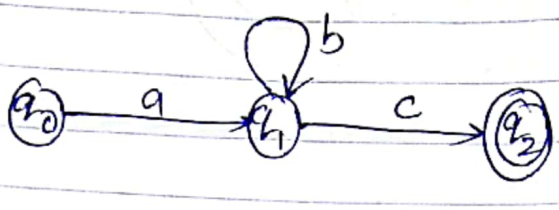
FA → RE

① 

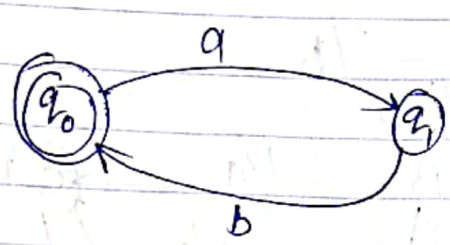
⇒ ab

② 

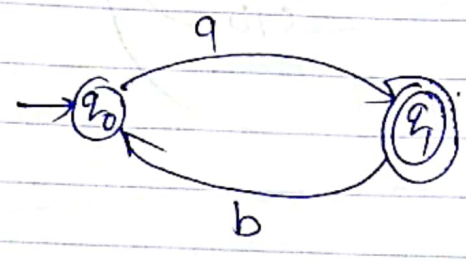
⇒ a+b

③ 

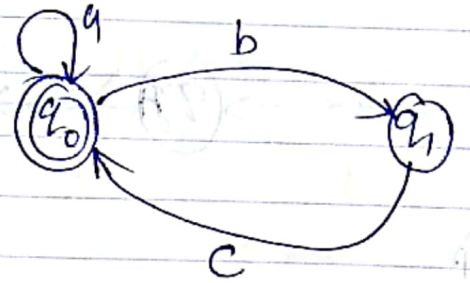
⇒ ab^*c

④ 

⇒ $(ab)^*$

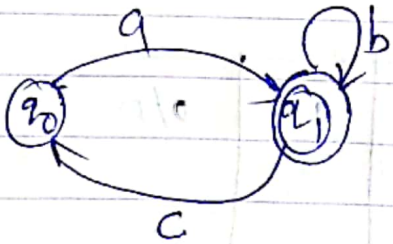
⑤ 

⇒ $a(ba)^*$

⑥ 

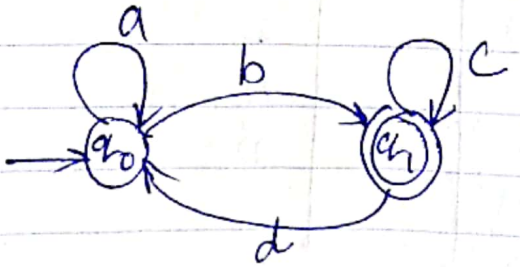
⇒ $(a^*+(bc)^*)^*$

⇒ $(a+bc)^*$

⑦ 

⇒ ~~$a(b^*(ca)^*)^*$~~

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

⑧ 

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

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

CFL → CFG

1) $a^n b^n$, $n \geq 0$ — $S \to aSb/\epsilon$

$a^n b^n$, $n \geq 1$ — $S \to aSb/ab$

2) $a^n b^{n+2}$, $n \geq 0$ — $S \to aSb/bb$

3) $a^{2n} b^n$, $n \geq 0$ — $S \to aaSb/\epsilon$

$n \geq 1$ — $S \to aaSb/aab$

4) $a^{2n+3} b^n$ $n \geq 0$ — $S \to aaSb/aaa$

5) $a^m b^n$, $m > n$, $n \geq 0$

$S \to AS_1$
$S_1 \to aS_1b/a\epsilon$
$A \to aA/a$

6) $L = \{w / n_a(w) = n_b(w)\}$

$S \to aSb/bSa/\epsilon$
$S \to aSb/bSa/SS/\epsilon$

7) $L = \{ww^R\}$ — $S \to aSa/bSb/\epsilon$

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

$S \to S_1C$
$S_1 \to aS_1b/\epsilon$
$C \to cC/\epsilon$

CFG
↓
CFL
↓
PDA

Context-free Grammar

① Construct CFG for the language having any no. of a's over the set $\Sigma = \{a\}$.

$G\{V, T, P, S\}$

$\Sigma = \{a\}$

$L = \{\epsilon, a, aa, \ldots\}$.

$RE = a^*$

Production Rule :-

$S \rightarrow aS$ — ①

$S \rightarrow \epsilon$ — ②

derive i/p = "aaaaa"

S

$\Rightarrow$ aS $S \rightarrow aS$

$\Rightarrow$ aaS $S \rightarrow aS$

$\Rightarrow$ aaaS $S \rightarrow aS$

$\Rightarrow$ aaaaS $S \rightarrow aS$

$\Rightarrow$ aaaa $S \rightarrow \epsilon$

CFL → CFG

② Construct a CFG for language string $L = \{w c w^R \mid \text{where } w \in (a,b)^*\}$

$L = \{aacaa, bcb, abcba, \ldots\}$

Grammar

$S \rightarrow asa$ — ①

$S \rightarrow bsb$ — ②

$S \rightarrow c$ — ③

derive i/p ⇒ "abbcbba"

S

⇒ asa — ①

⇒ abSba — ②

⇒ abbSbba — ②

⇒ abbcbba — ③

* Leftmost Derivation :-

Left most derivation of a gra is obtained by replacing the leftmost variab by its production.

Right most Derivation :-

Right most Derivation of a gra is obtained by replacing the right most variable by its production.

Q1. Derive the string aba for leftmost and rightmost derivation using clg given by.

$S \rightarrow XYX$.

$X \rightarrow a$

$Y \rightarrow b$.

Sol: $V = (X, Y, S)$.

$T = (a, b)$

leftmost derivation

	Production
S	XYX
aYX	$X \rightarrow a$
abX	$Y \rightarrow a$
aba	$X \rightarrow a$

Right most derivation.

XYX	$S \rightarrow XYX$
XYa	$X \rightarrow a$
Xba	$Y \rightarrow b$
aba	$X \rightarrow a$

Q.2. Derive the string aabbabba for leftmost and right most derivation using cfG

$S \rightarrow aB / bA$

$A \rightarrow a / aS / bAA$

$B \rightarrow b / bS / aBB$

Solution: Note start with start Variable.

case 1:- For left most Derivation

S	Production
aB	$S \rightarrow aB$
aaBB	$B \rightarrow aBB$
aabB	$B \rightarrow b$
aabbS	$B \rightarrow bS$
aabbaB	$S \rightarrow aB$
aabbabS	$B \rightarrow bS$
aabbabbA	$S \rightarrow bA$
aabbabba	$A \rightarrow a$

case 2:- for right most Derivation.

S	Production
aB	$S \rightarrow aB$
aaBB	$B \rightarrow aBB$
aaBbS	$B \rightarrow bS$
aaBbbA	$S \rightarrow bA$
aaBbba	$A \rightarrow a$
aabSbba	$B \rightarrow bS$
aabbAbba	$S \rightarrow bA$
aabbabba	$A \rightarrow a$

Q.3 Consider the grammer $S \rightarrow 0B \mid 1A$

$A \rightarrow 0 \mid 0S \mid 1AA$

$B \rightarrow 1 \mid 1S \mid 0BB$

derive the string 00110101 find the follo

1) left most.
2) Right most.

Sol: Left most derivation.

S	Production.
0B	$S \rightarrow 0B$
00BB	$B \rightarrow 0BB$
001SB	$B \rightarrow 1S$
0011S	$B \rightarrow 1S$
00110B	$S \rightarrow 0B$
001101S	$B \rightarrow 1S$
0011010B	$S \rightarrow 0B$
00110101	$B \rightarrow 1$

Right most Derivation.

S	Production
0B	$S \rightarrow 0B$
00BB	$B \rightarrow 0BB$
00B1S	$B \rightarrow 1S$
00B10B	$S \rightarrow 0B$
00B101	$B \rightarrow 1$
001S101	$B \rightarrow 1S$
0011A101	$S \rightarrow 1A$
00110101	$A \rightarrow 0$

Derivation Tree / Parse Tree

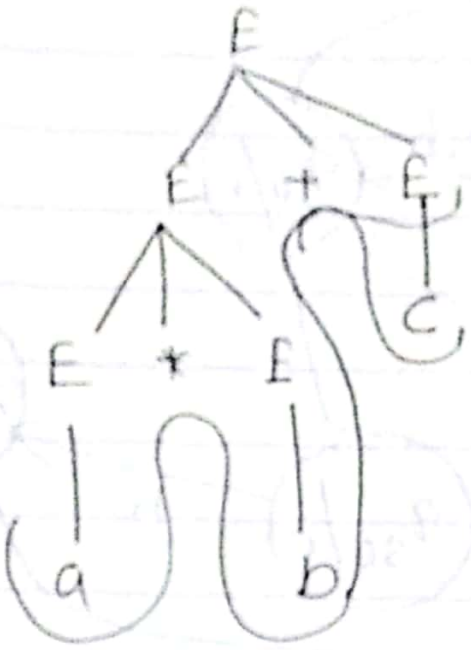
(1) Product^{n} rule :

$E = E+E$

$E = E*E$

$E = a/b/c$

i/p derivation — a* b+c.



Ambiguity in Grammer

- more than one LMD
- " " RMD
- " " parse tree
- remove the ambiguity by rewriting grammer.

eg ①

E = I

E = E + E

E = E * E

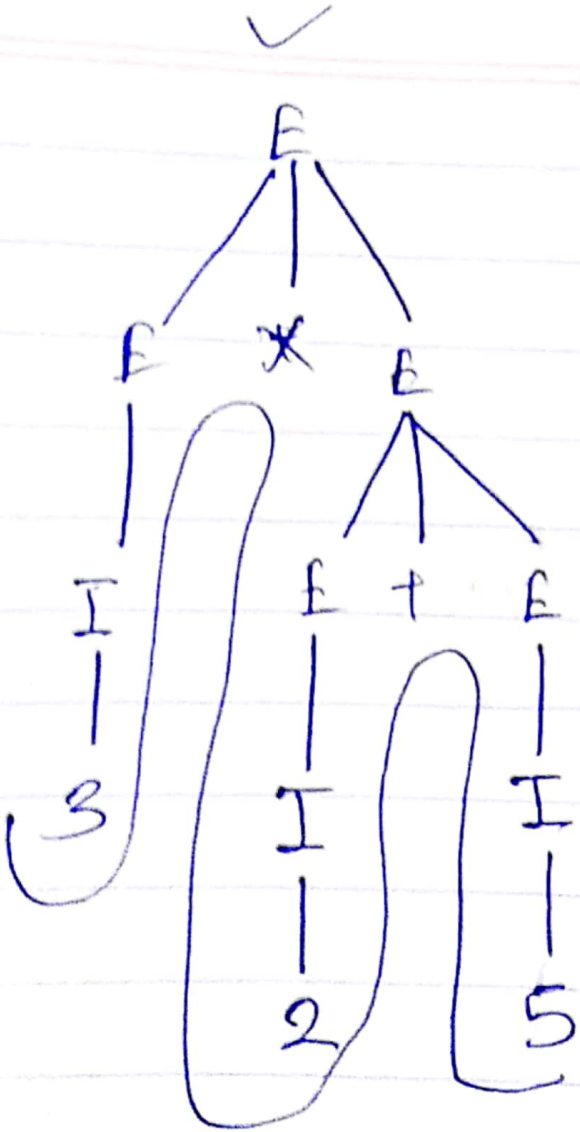
E = (E)

I = 0 | 1 | 2 ... 9

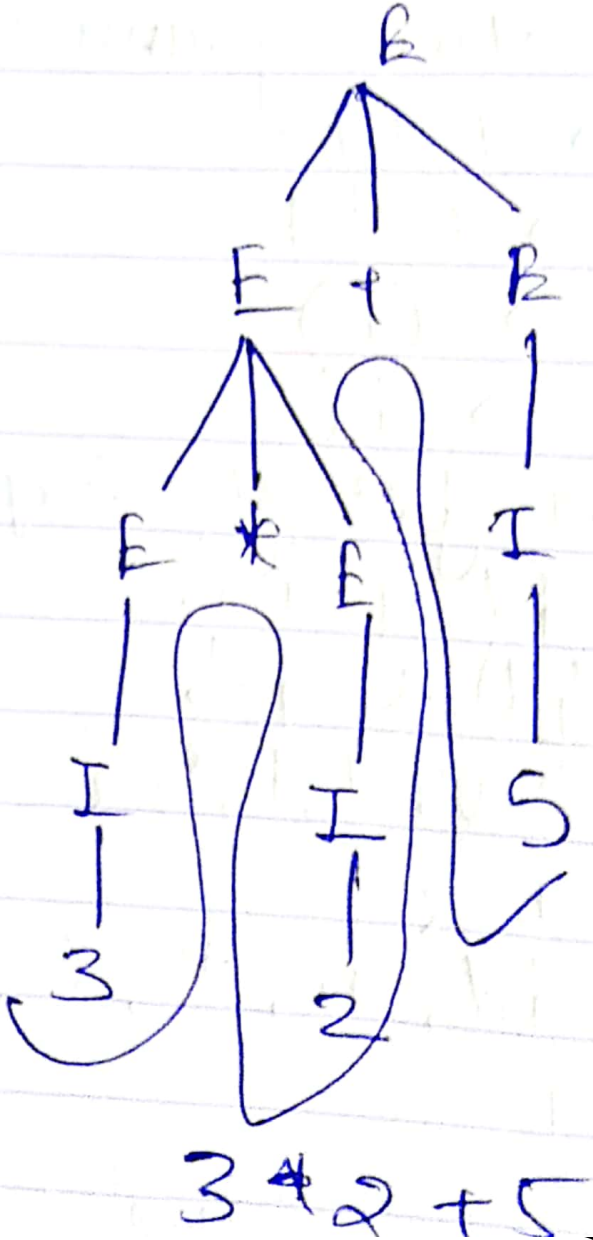
V = { I, E }

T = { + * ( ), ϵ, 0, 1, 2 ... 9 }

I/p derivat^{n} - 3 * 2 + 5



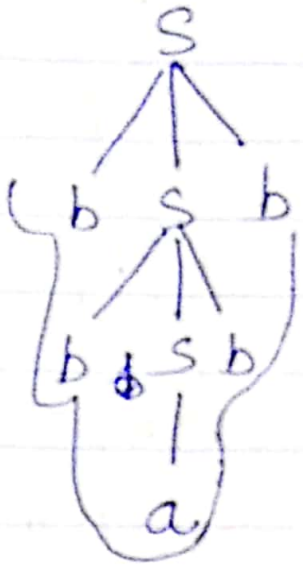
3 * 2 + 5



3 4 2 + 5

Q) Construct derivation tree for the string 'bbabb' from CFG given by

$S \rightarrow bSb / a / b$



Q) Construct DT for string aabbabba for CFG

$S \rightarrow aB / bA$

$A \rightarrow a / aS / bAA$

$B \rightarrow b / bS / aBB$

Q. Consider the grammer

$E \rightarrow E + E$

$E \rightarrow E * E$

$E \rightarrow (E)$

$E \rightarrow id$

obtain DT of expression

(1) id * id + id

(2) (id + id) * id

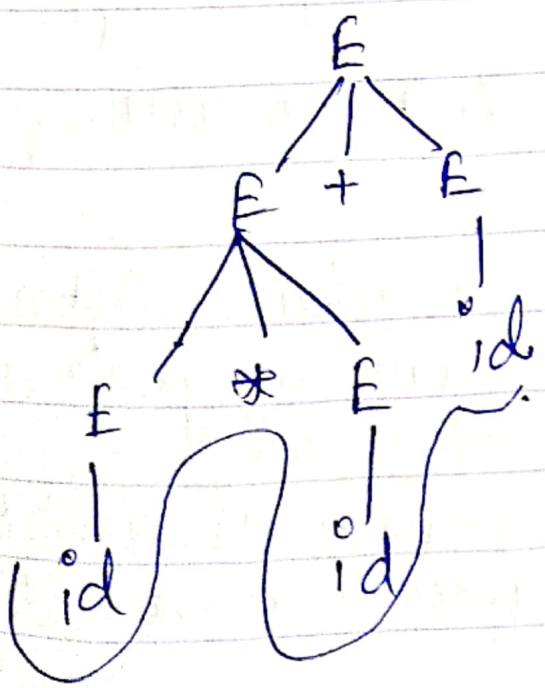
$G = \{V, T, P, S\}$

$V = \{E\}$

$T = \{id, +, *, (,)\}$

$S = E$

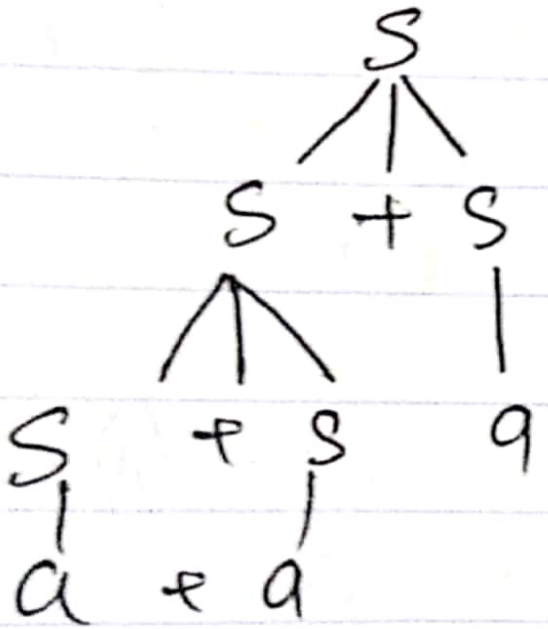
(1) id * id + id

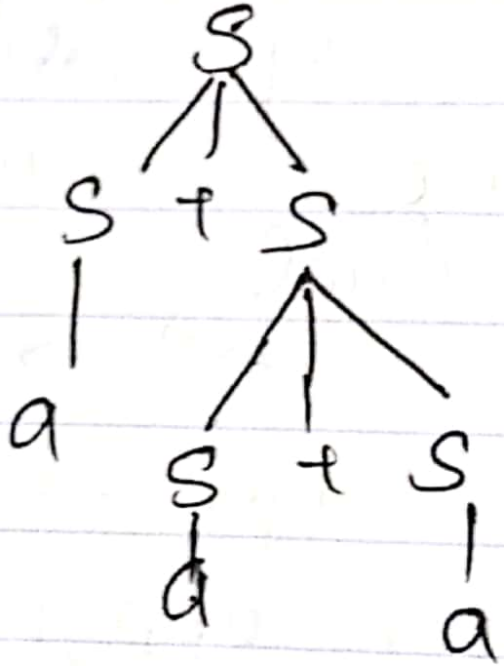


Scanned with Cam

① $S \rightarrow S+S/a$

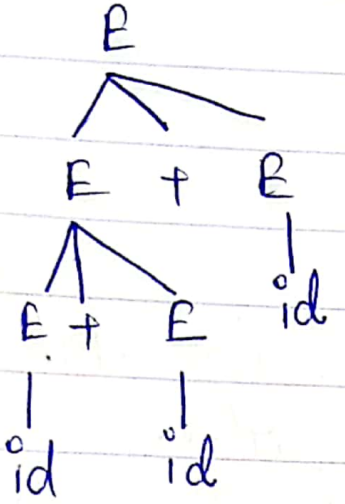
$w = a+(a+a)$

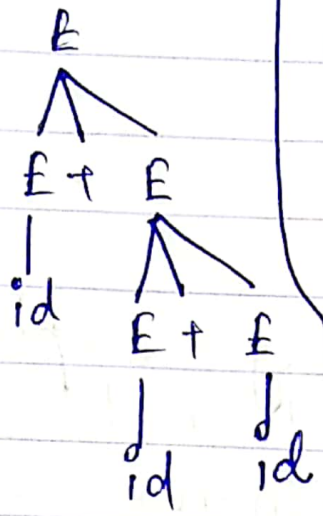




Converting ambiguous to unambiguous grammer

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





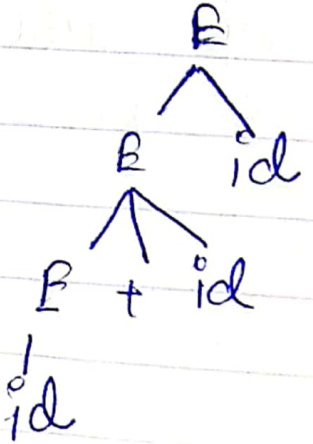
① $id + id + id$

Left associative $(id + id) + id$
Right " $id + (id + id)$

Left recursive

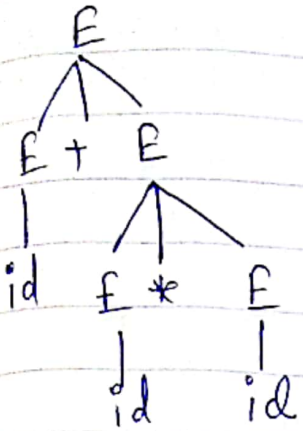
To achieve left associativity we have to grow in left direction only.
So now we restrict the growth in right direct^{n}

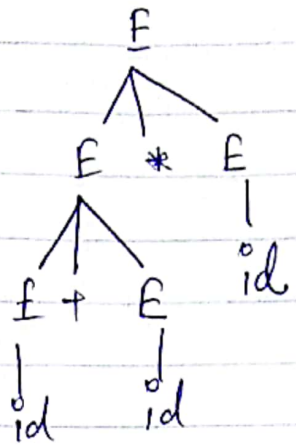
$E \rightarrow E + id$
$E \rightarrow id$



(2) id + id * id

precedence is not taken care





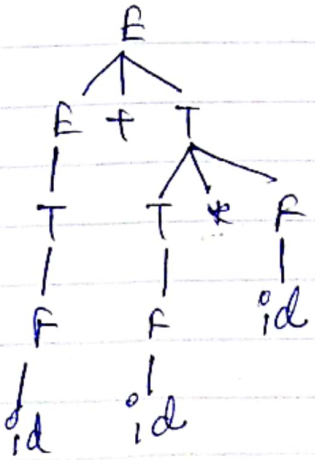
id + (id * id) (id+id) * id

For precedence problem :-

Highest precedence operator should be at the least level

E → E+T
E → T
T → T*F/F
F → id

⇒ id + id * id



Simplification of CFG

- Extra Symbol
  - remove useless Symbol
  - " epsilon production
  - " Unit production.

Removing Unit Production

NT → NT

eg X → Y

S → Y

To remove X → Y, add production X → a to the grammer whenever Y → a occurs in grammer

- then delete X → Y

eg S → OA/1B/C
A → OS/00
B → 1/A
C → 01

⇒ S → OA/1B/01
A → OS/00
B → 1/OS/00
C → 01

· Unit Productn removal

M → N/QMnqQ
N → nqN/n
Q → qQ/∈

M → N

A → B → C
A → C

M → nqN/n/QNnqQ

(2) S → Aa/B/C
B → A/bb
A → a/bc/B

Removal of ϵ productions

eg

$S \rightarrow XYX$

$X \rightarrow 0X / \epsilon$

$Y \rightarrow 1Y / \epsilon$

$S \rightarrow XYX / YX / XY / XX / Y / X$

$X \rightarrow 0X / 0$

$Y \rightarrow 1Y / 1$

① $S \rightarrow ABC$
$A \rightarrow a/bbD$
$B \rightarrow a/\epsilon$
$C \rightarrow b/\epsilon$
$D = C/d$

$\Rightarrow$

Remove $B \rightarrow \epsilon$

$S \rightarrow ABC/AC/AB$
$A \rightarrow a/bbD$
$B \rightarrow a$
$C \rightarrow b$
$D \Rightarrow C/d$

2) $S \to ABC \mid CbB \mid Ba$
$A \to da \mid BC$
$B \to gC \mid \epsilon$
$C \to ha \mid \epsilon$

$\Rightarrow$

$S \to ABC \mid \epsilon bB \mid Ba \mid \epsilon$
$AC \mid AB \mid bB \mid \epsilon$
$A \to da \mid BC \mid B \mid C$
$B \to gC \mid g$
$C \to ha \mid \epsilon$

Remove Useless production

eg
$S \to AB/a$
$A \to BC/b$
$B \to aB/C$
$C \to aC/B$

Derivability
Reachability

Useful symbols = { a, b, S, A, }

$S \to AB/a$
$A \to BC/b$

$\Rightarrow S \to a$
$A \to b$ } checked Reachability.

eg
$S \to AB/AC$
$A \to aAb/bAa/a$
$B \to bbA/aaB/AB$
$C \to abCA/aDb$
$D \to bD/aC$

Useful = {a, b, A, B, S}

$S \to AB$
$A \to aAb/bAa/a$
$B \to bbA/aaB/AB$

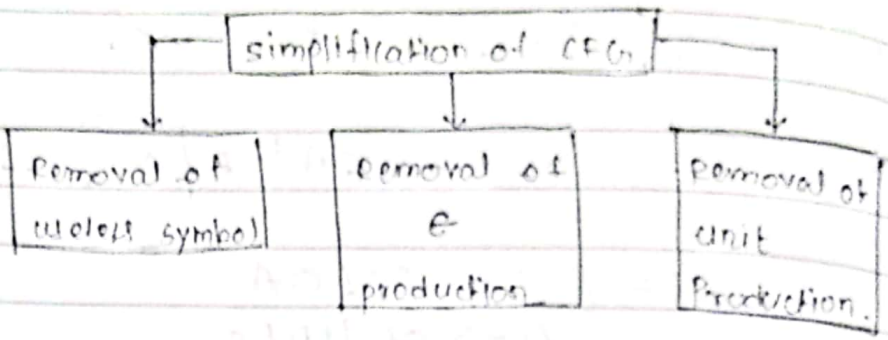
Remove Useless Production

eg- $S \rightarrow aS/A/C$
$A \rightarrow a$
$B \rightarrow aa$
$C \rightarrow aCb$

eg $S \rightarrow aA/a/Bb/cC$
$A \rightarrow aB$
$B \rightarrow a/Aa$
$C \rightarrow cCD$
$D \rightarrow ddd$

7

Simplification of CFG



* Removal is useless Symbol
* USE full symbol.

1. The symbol is useful if it appears on the left side of production.
2. It generates some terminal symbols.
3. A useful symbol is of the form

$$S \longrightarrow aP|\beta$$
$$P \longrightarrow a$$

then P is a useful symbol.

Q1. $S \longrightarrow aA|bB$
$A \longrightarrow aA|a$
$B \longrightarrow bB$ ✗
$D \longrightarrow ab|Ea$ ✓
$E \longrightarrow ac|d$ ✗

Ans. B → does not gives any Terminal.
D → not reachable from S i.e start state.
E → not reachable from S i.e start state.

1. B is a useless symbol as it does not produce any terminal symbol.
2. D and E cannot be reached through any non-terminal symbol so we remove it.

Thus the clean grammer is,

$$S \rightarrow aA$$
$$A \rightarrow aA \mid a$$

Q.2. $S \rightarrow AB \mid CA$ find the clean grammer.
$A \rightarrow a$
$B \rightarrow BC \mid AB$
$C \rightarrow aB \mid b$.

Ans. The clean gravomer is.

$S \rightarrow CA$
$A \rightarrow a$
$C \rightarrow b$

* Removal of ϵ production.

1. A production of the form

$A \rightarrow \epsilon$ is called as a null production.

2. A simplified grammer should be without these production.

eg:- $A \rightarrow aA$.
$A \rightarrow \epsilon$

substituting ϵ in place of.

$\therefore A \rightarrow a(\epsilon)$

$\therefore A \rightarrow a$.

* Steps to eliminate ε production.

1. Delete all the ε production.
2. Identify nullable non terminals.
3. Start eliminating all subsets on nullable & non terminal.

Q-1. $S \rightarrow XY$
$X \rightarrow Zb$
$Y \rightarrow bW$
$Z \rightarrow AB$
$W \rightarrow Z$
$A \rightarrow aA \mid bA \mid \epsilon$
$B \rightarrow Ba \mid Bb \mid \epsilon$

Solution:- Step 1:- Removal of ε production.

$A \rightarrow \epsilon$, $B \rightarrow \epsilon$

∴ New production are.

$A \rightarrow aA \mid bA \mid a \mid b$.
$B \rightarrow Ba \mid Bb \mid a \mid b$

Step 2:- $Z \rightarrow AB$
∴ $\boxed{Z \rightarrow \epsilon}$ [∵ $A \rightarrow \epsilon, B \rightarrow \epsilon$]

$W \rightarrow Z$.
∴ $\boxed{W \rightarrow \epsilon}$

$Y \rightarrow bW$
∴ $\boxed{Y \rightarrow b}$ [∵ $W \rightarrow \epsilon$]

$X \rightarrow Zb$.
∴ $X \rightarrow b$ [∵ $Z \rightarrow \epsilon$]

Step 3:- CFG.

$S \longrightarrow XY$

$X \longrightarrow b$

$Y \longrightarrow b$

$A \longrightarrow a \mid b$

$B \longrightarrow a \mid b$ ⟩ [$\because Ba \to \epsilon(a) = a$ & $bA \to \epsilon(a) = b$]

Step 4 :- Clean CFG.

since A and B are not ~~remove~~ reachable from start state

$\therefore$ A and B are useless symbols.

Final clean CFG :- $S \longrightarrow XY$

$X \longrightarrow b$

$Y \longrightarrow b$.

Q.2. $S \longrightarrow 0S$

$S \longrightarrow 1S$

$S \longrightarrow \epsilon$

Ans. Step 1 :- $S \longrightarrow 0S \mid 1S \mid 0 \mid 1$.

Step 2 :- $S \to 0$

$S \to 1$

Q.3 $S \rightarrow XYX$
$X \rightarrow 0X \mid \epsilon$
$Y \rightarrow 1Y \mid \epsilon$

Ans Step 1:- Removal of ϵ production.

$X \rightarrow \epsilon$, $Y \rightarrow \epsilon$

∴ New productions

$S \rightarrow XY \mid YX \mid XX \mid X \mid Y$
$X \rightarrow 0X \mid 0$
$Y \rightarrow 1Y \mid 1$

Step 2:- $Y \rightarrow 1Y$
∴ $Y \rightarrow 1$ [$\because Y \rightarrow \epsilon$]

$X \rightarrow 0X$
∴ $X \rightarrow 0$ [$\because X \rightarrow \epsilon$]

A unit production of the form $A \rightarrow B$
i.e Non-terminal 1 $\rightarrow$ Non-terminal 2

* Removal of unit production

A unit production is of the form $A \rightarrow B$
i.e non-terminal 1 $\rightarrow$ non-terminal 2

$S \rightarrow A$
$A \rightarrow a$
$S \rightarrow a$

Q1. $S \rightarrow aA \mid 1B \mid c$
$A \rightarrow 0S \mid 00$
$C \rightarrow 01$
$B \rightarrow 1 \mid A$

Sol: Step 1 :- $S \rightarrow C$

but $C \rightarrow 01$

∴ replace $S \rightarrow 0A \mid 1B \mid 01$

Step 2 :- $B \rightarrow 1 \mid A$.

but $A \rightarrow 0S \mid 00$

∴ replace $B \rightarrow 1 \mid 0S \mid 00$

∴ final production

$S \rightarrow 0A \mid 1B \mid 01$

$A \rightarrow 0S \mid 00$

$B \rightarrow 1 \mid 0S \mid 00$

Q.2. $S \rightarrow A \mid 0C1$

$A \rightarrow B \mid 01 \mid 10$

$C \rightarrow \epsilon \mid CD$.

Sol: step 1 :- $C \rightarrow \epsilon$ ∴ $C \rightarrow \epsilon D$.

But C and D does not gives any Terminal symbol ∴ remove C and D.

∴ $S \rightarrow A \mid 0C1$

$A \rightarrow B \mid 01 \mid 10$.

Now $A \rightarrow B$ but B does not generate any terminal symbol ∴ remove B.

∴ $S \rightarrow A \mid 01$

$A \rightarrow 01 \mid 10$.

Now $A \rightarrow 01 \mid 10$.

∴ $S \rightarrow 01 \mid 10 \mid 01$.

∴ $\boxed{S \rightarrow 01 \mid 10}$.

* Chomsky's Normal form. (CNF).

1. Any CFG without 'ϵ' can be generated by the grammer in which all the production's are of the form

$$A \longrightarrow BC \text{ or}$$
$$A \longrightarrow a$$

where B and C are non terminal. and a is a terminal symbol.

2. In CNF on the right hand side of production either two non-terminals are allowed or only one terminal symbol is allowed.

① Convert the given CFG to CNF grammar G_1 as.

$S \rightarrow a/aA/B$
$A \rightarrow aBB/\epsilon$
$B \rightarrow Aa/b$

---

① $S \rightarrow bA/aB$
$A \rightarrow bAA/aS/a$
$B \rightarrow aBB/bS/b$

Sol^n ① Rule of CNF
$NT \rightarrow NT.NT$
$NT \rightarrow T$

② Given Prod^n Rule

③ Simplify CFG
① Eliminate ϵ — NO
② Eliminate Unit — NO
③ Eliminate useless Symbol — NO

④ CFG → CNF
$S \rightarrow bA$
$S \rightarrow aB$
$A \rightarrow bAA/aS/a$
~~$B \rightarrow aBB/bS/b$~~
$A \rightarrow aS$
$A \rightarrow a$
$B \rightarrow aBB$
$B \rightarrow bS$
$B \rightarrow b$

Rule ① $S \rightarrow bA$

$S \rightarrow C_bA$

$*C_b \rightarrow b$

Rule ② $S \rightarrow aB$

$S \rightarrow C_aB$

$*C_a \rightarrow a$

Rule ③ $A \rightarrow bAA$

$A \rightarrow C_bD$

$C_b \rightarrow b$

$*D_1 \rightarrow AA$

Rule ④ $A \rightarrow aS$

$A \rightarrow C_aS$

Rule ⑤ $B \rightarrow aBB$

$B \rightarrow C_aD_2$

$*D_2 \rightarrow BB$

Rule ⑥ $B \rightarrow bS$

$B \rightarrow C_bS$

⑤ Resultant Productn

$S \rightarrow C_bA / C_aB$

$A \rightarrow C_bD / C_aS / a$

$B \rightarrow C_aD_2 / C_bS / b$

$C_a \rightarrow a$

$C_b \rightarrow b$

$D_1 \rightarrow AA$

$D_2 \rightarrow BB$

② Q. → CFG → CNF

q_1 → $S \to a/aA/B$
$A \to aBB/\epsilon$
$B \to Aa/b$

$S \to \epsilon$
$A \to AB$
$A \to Ba$

② Remove ε production

$S \to a/aA/B$
$A \to aBB$
$B \to Aa/b/a$

③ Remove Unit Productn

$S \to a/aA/b$
$A \to aBB$
$B \to Aa/b/a$

④ Remove Useless Productn. — NO

⑤ CFG → CNF

$S \to a$ ✓
$S \to aA$
$S \to b$ ✓
$A \to aBB$
$B \to Aa$
$B \to b$ ✓
$B \to a$ ✓

Rule ① $S \to aA$

$S \to C_aA$
$C_a \to a$

Rule ② $A \to aBB$

$A \to C_aD_1$
$C_a \to a$
$D_1 \to BB$

Rule ③ $B \to Aa$

$B \to AC_a$

Rule ④

(5) Resultant Production :-

$S \rightarrow C_a A \mid a \mid b$

$A \rightarrow C_a D_1$

$B \rightarrow A C_a \mid b \mid a$

$C_a \rightarrow a$

$D_1 \rightarrow BB$

Q1. $S \rightarrow asa \mid bsb \mid a \mid b \mid aa \mid bb$ convert into CNF

Ans.

$S \rightarrow a$ —(i)

Already in production.

$S \rightarrow b$ —(2)

Already in production.

$S \rightarrow aa$.

To convert into CNF

Add $A \rightarrow a$

& $S \rightarrow AA$. —(3)

$S \rightarrow bb$.

To convert into CNF

Add $B \rightarrow b$

& $S \rightarrow BB$ —(4)

$S \rightarrow aSa$.

$A \rightarrow a$.

$\therefore S \rightarrow ASA$. still not in CNF

$\therefore R_1 \rightarrow AS$

$\therefore S \rightarrow R_1 A \quad -(5)$

$S \rightarrow bSb$.

$B \rightarrow b$.

$\therefore S \rightarrow BSB$ still not in CNF

$\therefore R_2 = BS$.

$\therefore S \rightarrow R_2 B \quad -(6)$

final CNF = $S \rightarrow a$

$S \rightarrow b$

$A \rightarrow a$

$B \rightarrow a$

$S \rightarrow AA$

$S \rightarrow BB$

$S \rightarrow R_1 A$

$S \rightarrow R_2 B$.

$R_1 \rightarrow AS$

$R_2 \rightarrow BS$.

Q.2. Express CFG in CNF

$S \to ABA$

$A \to aA \mid \epsilon$

$B \to bB \mid \epsilon$

Sol: Step 1:- Removal of ϵ production.

$A \to \epsilon$, $B \to \epsilon$.

$\therefore S \to AB \mid BA \mid AA \mid A \mid B \mid ABA$

$A \to aA \mid a$

$B \to bB \mid b$

Step 2:- $A \to aA$.

$\therefore A \to a \quad [\because A \to \epsilon]$

$B \to bB$

$\therefore B \to a$

Step 3 :- $S \to ABA$.

$\therefore S \to SA \quad -(1) \quad [\because S \to AB]$.

$S \to AB \quad -(2)$

in CNF

$S \to BA \quad -(3)$

in CNF

$S \to AA$ in CNF $-(4)$.

$S \to A$.

$\therefore S \to a \quad -(5)$

$S \to B$

$S \to b \quad -(6)$

$A \rightarrow aA$

$A \rightarrow SA$.

$B \rightarrow bB$..

$\therefore B \rightarrow SB$

Final production $S \rightarrow SA|AB|BA|AA|a|b$

$A \rightarrow SA$

$B \rightarrow SB$.

Q.3. $S \rightarrow bA|aB$ Express CFG in CNF

$A \rightarrow bAA|aS|a$.

$B \rightarrow aBB|bS|b$.

Sol:

$\boxed{B \rightarrow b}$ in CNF

$\boxed{A \rightarrow a}$ in CNF

$S \rightarrow bA$. $\therefore \boxed{S \rightarrow BA}$

& $S \rightarrow aB$. $\boxed{S \rightarrow AB}$

$A \rightarrow bAA$.

$\therefore A \rightarrow BAA$

$S \rightarrow BA$

$\therefore A \rightarrow BA$

$\boxed{A \rightarrow SA}$

$B \rightarrow aBB$

$A \rightarrow a$.

$\therefore B \rightarrow ABB$.

$\therefore \boxed{B \rightarrow SA}$

$A \rightarrow aS$.

$A \rightarrow a$..

$\therefore \boxed{A \rightarrow AS}$

$B \rightarrow bS$.

$B \rightarrow b$.

$\therefore \boxed{B \rightarrow BS}$

$\therefore S \rightarrow BA|AB$

$A \rightarrow SA|AS|a$

$B \rightarrow SA|BS|b$.

Q.3. $S \rightarrow aAD$

$A \rightarrow aB \mid bAB$

$B \rightarrow b$

$D \rightarrow d.$

Sol. B not rechable from S ∴ B is useless symbol.

∴ remove B.

∵ B is removed A will also be removed.

∴ $S \rightarrow aAD$

$D \rightarrow d.$

∵ A is eliminated and grammer

∴ The whole production's will be eliminates

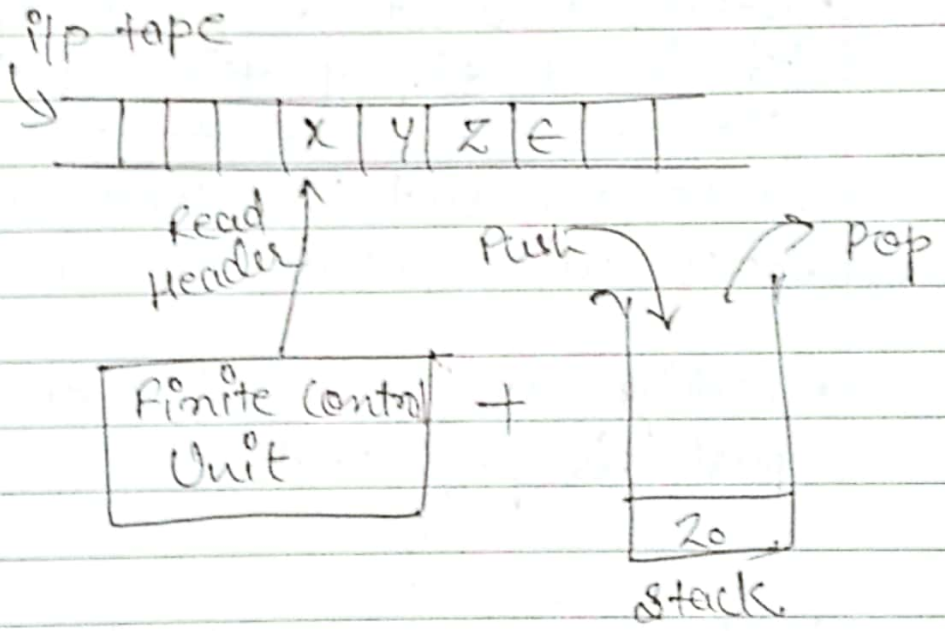
DOMS | Page No. | Date

Push Down Automata

CFG —generates→ CFL —Accepted by→ PDA

PDA = $\{Q, \Sigma, q_0, F, Z_0, \Gamma, \delta\}$

Q = Finite set of states
Σ = input alphabet
q_0 = initial state
F = set of final states.
Z_0 = Bottom/initial stack symbol
Γ = stack alphabet.
δ = transition function.

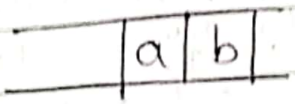


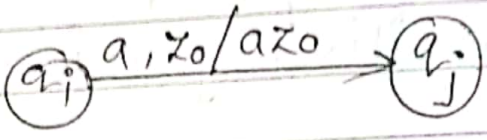
* Advantage of Stack :-
- Stack is zero address DS
- No need to give address to complete push & pop operation
- We assume that stack is of infinite length. So no need to look for overflow condition.

- Z_0 is used to indicate that stack is empty.

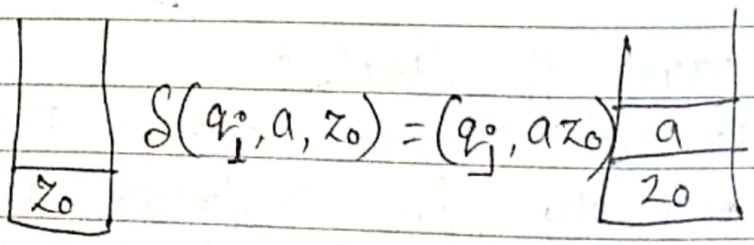
Operations on PDA

① PUSH :-



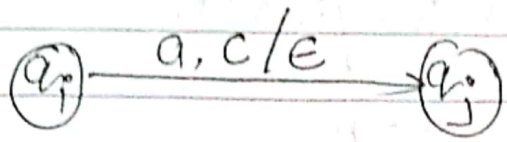


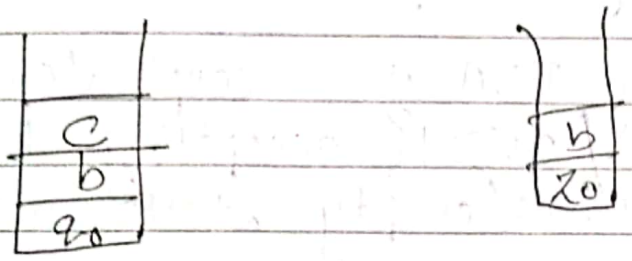
- When you read 'a' from (q_i) and 'Z_0' is at top of the stack then we will go to the new state (q_j) and stack will have new status as 'aZ_0'
- Looking aZ_0 you will know that push is performed.



DOMS | Page No.
Date / /

② POP :-

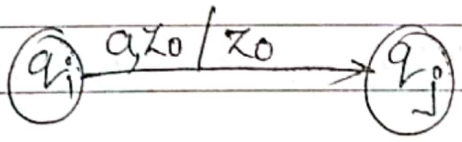


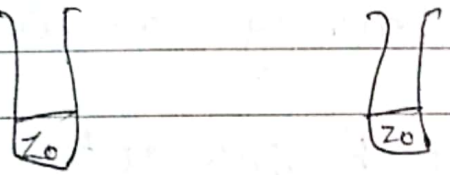


$\delta(q_0, a, c) = (q_j, \epsilon)$

ϵ represents pop operation.

③ Skip





$\delta(q_i, a, z_0) = (q_j, z_0)$

Representation of PDA

1. Transition Diagram
2. Transition fun^n/moves.

Designing PDA

Q1 Design PDA to accept language $L = \{a^n b^n / n \geq 1\}$ accepting by final states/empty stack.

or

$L = \{0^n 1^n / n \geq 1\}$

Sol^n (1) $L = \{a^n b^n | n \geq 1\}$

$L = \{ab, aabb, aaabbb. - - -\}$

(2) Algo (Logic)

Consider $n = 3$, $\omega = aaabbb$

1. Push 'n' no. of 'a' into the stack
2. For every 'b' pop out an 'a' from the stack
3. At the end of the string, the m/c stops as it reaches the final states.

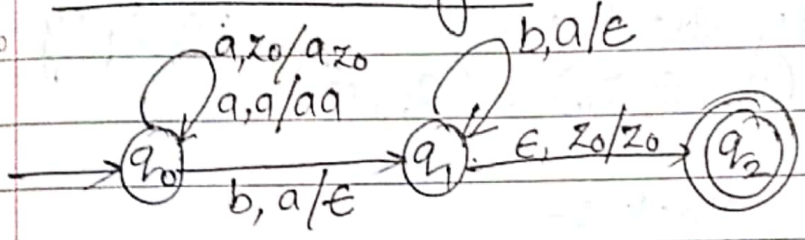
DOMS Page No.
Date / /

(3) Implementation :-

PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$

$Q = \{q_0, q, q_2\}$ $\quad q_0 = q_0$
$\Sigma = \{a, b\}$ $\quad z_0 = z_0$
$\Gamma = \{a, b, z_0\}$ $\quad F = \{q_2\}$

(4) Transition Diagram :-



(5) Transition functions :-

Push 'a' :-

$\delta(q_0, a, z_0) = (q_0, az_0)$

↑ top of stack

Push 'a' :-

$\delta(q_0, a, a) = (q_0, aa)$

Push 'a' :-

$\delta(q_0, a, a) = (q_0, aa)$

push 'b'

$\delta(q_0, b, a) = (q_1, \epsilon)$ — pop

push 'b'

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

push 'b'

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

Empty stack :-

$\delta(q_1, \epsilon, Z_0) = (q_1, \epsilon)$ — stack empty

Final stack :-

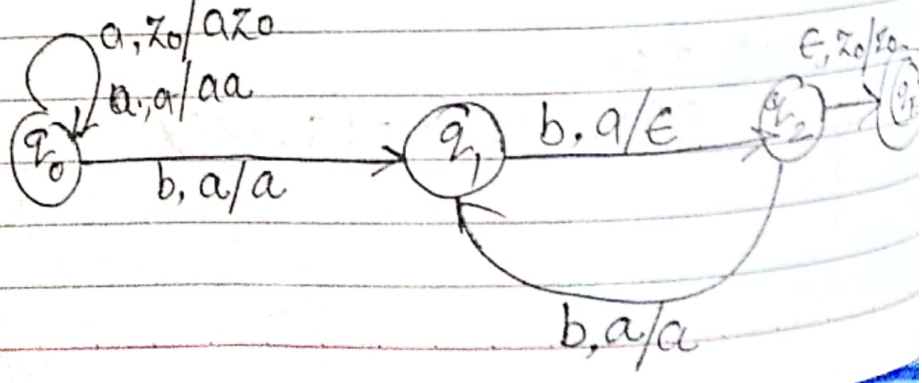
$\delta(q_1, \epsilon, Z_0$

Q2 $L = \{a^n b^{2n} \mid n \geq 1\}$

a	a	a	b	b	b	b	b	b	ϵ

∵ a = 2 times of b

∵ we cannot push 2 b's at a time to pop 1 a

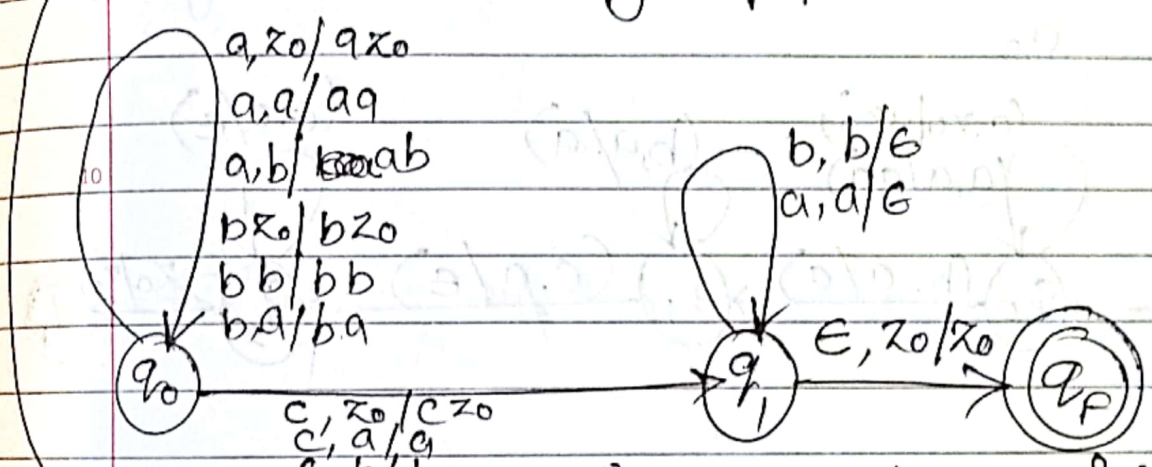


Q3 $L = \{wcw^R / w \in (a,b)^*\}$

a	b	b	c	b	b	a	∈

string seperator

Logic:
- c will always tell that w is finished
- stack is always poped in reverse.



If c is scanned q will skip & change to remember that the string w is finished.

- when reverse string starts, the scanned symbol is matched with stack top that only the pop operation is performed.

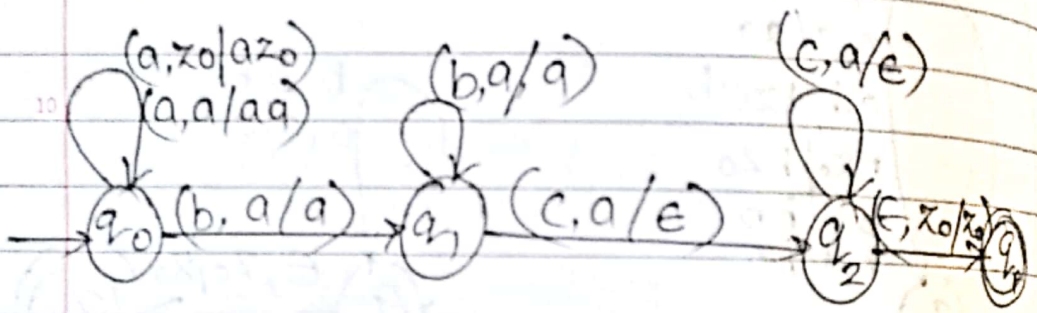
DOMS Page No. Date / /

Q4 $L=\{a^n b^m c^n \mid n \geq 1\}$

Logic :- since the sequence abc

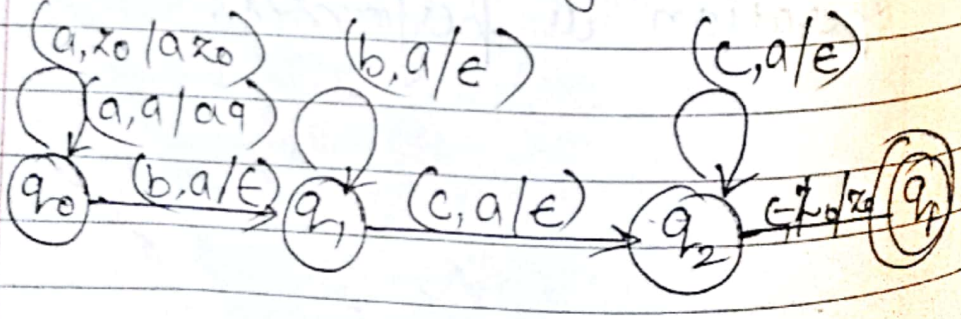
so after 'a' if 'b' comes simply skip the operation (NO PUSH/POP)

- we need to match no. of cs with as



Q5 $L=\{a^{m+n} b^m c^n \mid m,n \geq 1\}$

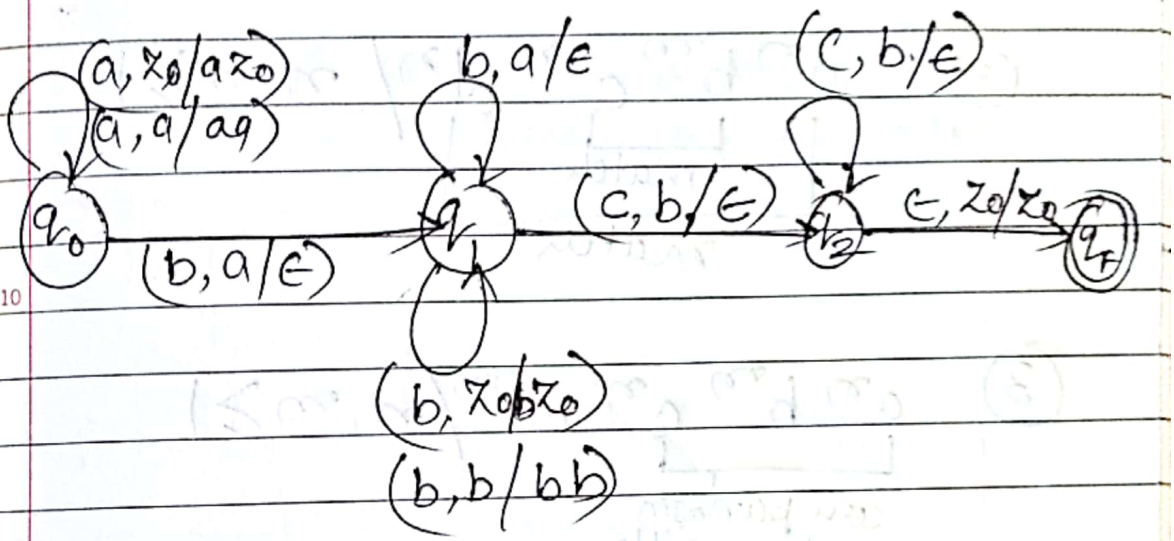
Logic ⇒ we have to save & store all a's in stack compare as count with bs and cs together.



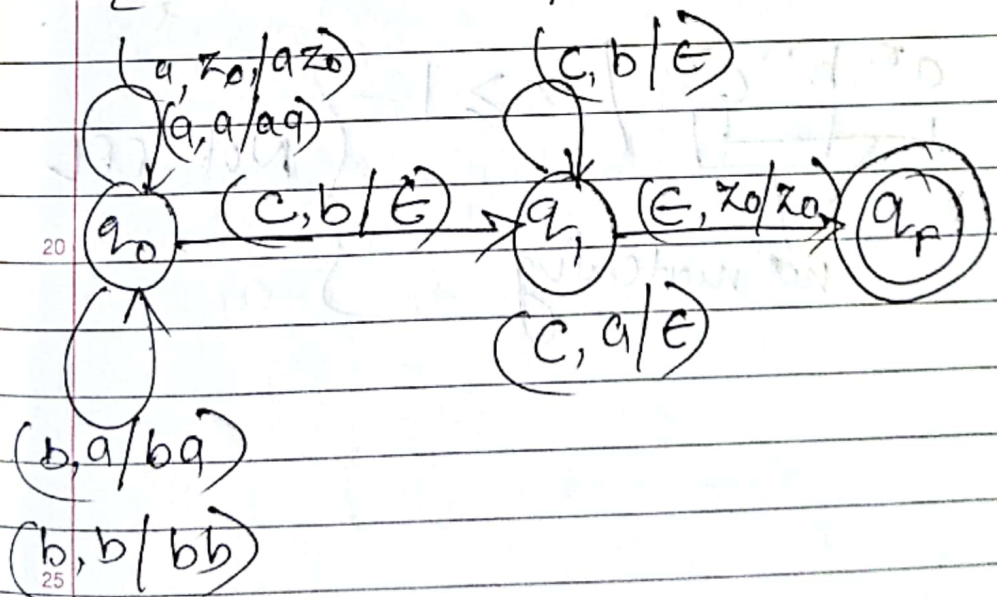
Q6 $L = \{a^n b^{m+n} c^m / n, m \geq 1\}$

$a^n b^n \; b^m c^m$

compare & pop — compare & pop



Q7 $L = \{a^n b^m c^{n+m} / n, m \geq 1\}$



even if a comes after b there will not be any transitions

DOMS
Date / / Page No. / /

Practice

(1) $a^n b^n c^m d^m \mid n, m \geq 1$

$a^n b^n$ — Match & pop; $c^m d^m$ — match & pop

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

$b^m c^n$ — match; $a^m \ldots d^n$ — match

(3) $a^n b^m c^n d^m \mid n, m \geq 1$

$a^n \ldots c^n$ — comparison not possible

Hence it is not CFL

(4) $a^n b^n c^n \mid n \geq 1$

not matching

Not CFL

DOMS Page No. Date / /

Q $L = \{ww^R / w \in (a,b)^+\}$

eg abaabq

↑

Here to find centre (string separator)

The possibility that the symbol of centre is when top of the stack and input symbol must match.

eg aa|aa

↑↓

So for this situation we need NPDA which will consider both the cases (if two symbol repeat then either centre has come or it is not)

aaaa

$(q_s, aaaa, z_0)$

↓

(q_s, aaa, qz_0)

No centre — Centre has come

↓ ↓

(q_s, aa, aaz_0) centre — (q_s, aa, z_0) ✗ stops

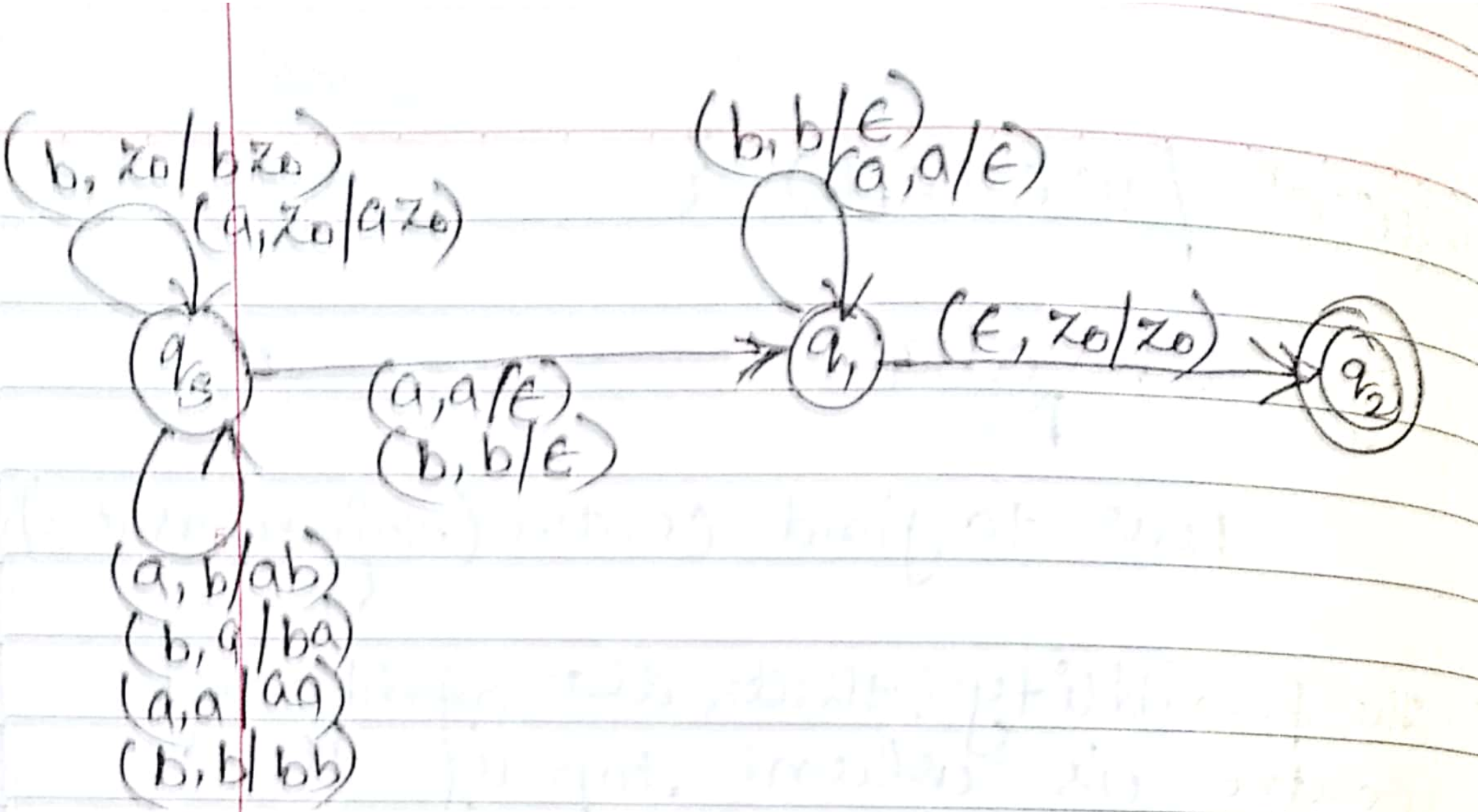
↓ ↓

(q_s, a, aaz_0) (q_1, a, az_0)

↓ ↓

$(q_s, \epsilon, aaaz_0)$ (q_1, ϵ, z_0)

(q_1, ϵ, qz_0)



Q. CFG → PDA

$S \to aSa$

$S \to bSb$

$S \to c$

Steps :- To convert CFG to PDA

(1) Write ~~production rules~~ transition rules

(2) Pop rules.

(1) $\delta(q_0, \epsilon, Z_0) = (q_0, Z_0)$

(2) $\delta(q_0, \epsilon, S) = (q_0, aSa)$

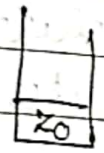
(3) $\delta(q_0, \epsilon, S) = (q_0, bSb)$

(4) $\delta(q_0, \epsilon, S) = (q_0, c)$

(5) $\delta(q_0, a, a) = (q_1, \epsilon)$

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

(7) $\delta(q_2, c, c) = (q_3, \epsilon)$



transition ~~Production~~ rules.

Pop rules.

2015 Page No. Date / /

Transition Table

Sr No	State	unread i/p	top of stack	transition
1	q_0	abbcbba	z_0	1
2	q_0	abbcbba	a	1
3	q_0	bbcbba	asa	2
4	q_1	bbcbba	sa	5
5	q_0	bbcbba	bsba	3
6	q_2	bcbba	sba	6
7	q_0	bcbba	bsbba	3
8	q_2	cbba	sbba	6
9	q_0	bba	bbba	4
10	q_3	bba	bba	7
11	q_2	ba	ba	6
12	q_2	a	a	6
13	q_1	z_0	z_0	5

Q Construct a PDA A which is equivalent to the following given CFG

$S \rightarrow OCC$

$C \rightarrow OS$

$C \rightarrow 1S$

$C \rightarrow 0$

Test whether 010000 is accepted PDA A

Sol^{n} PDA is as follow

$A = \{Q, \{0,1\}, \{S, C, 0, 1\}, \delta, q_0, Z_0, F\}$

δ is defined by following rules:-

(1) $\delta(q_0, \epsilon, Z_0) = (q_0, Z_0)$

(2) $\delta(q_0, \epsilon, S) = (q_0, OCC)$

(3) $\delta(q_0, \epsilon, C) = (q_0, OS)$

(4) $\delta(q_0, \epsilon, C) = (q_0, 1S)$

(5) $\delta(q_0, \epsilon, C) = (q_0, 0)$

} transition rules

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

(7) $\delta(q, 1, 1) = (q_2, \epsilon)$

} Pop rules.

Sr No.	State	Unread i/p	top of stack	transition No.
1	q_0	010000	Z_0	1
2	q_0	010000	S	1
3	q_0	ø10000	øC.C	2
4	q_1	10000	C C	6
5	q_0	X0000	X.S.C	4
6	q_1	0000	S C	7
7	q_0	ø000	ø.CCC	2
8	q_1	000	CCC	6
9	q_0	ø00	øCC	5
10	q_1	00	CC	6

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

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

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

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

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

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

DOMS | Page No. | Date / /

⑧ Production Rules

① S production

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

$\delta(q_0, a, z_0) \longrightarrow (q_0, a z_0)$ ← Push operation

$[q_0, z_0, q_0] \longrightarrow a\,[q_0, a, q_0]\;[q_0, z_0, q_0]$
$[q_0, z_0, q_0] \longrightarrow a\,[q_0, a, q_1]\;[q_1, z_0, q_0]$
$[q_0, z_0, q_1] \longrightarrow a\,[q_0, a, q_0]\;[q_0, z_0, q_1]$
$[q_0, z_0, q_1] \longrightarrow a\,[q_0, a, q_1]\;[q_1, z_0, q_1]$

$\delta(q_0, a, a) \longrightarrow (q_0, aa)$

$[q_0, a, q_0] \longrightarrow a\,[q_0, a, q_0]\;[q_0, a, q_0]$
$[q_0, a, q_0] \longrightarrow a\,[q_0, a, q_1]\;[q_1, a, q_0]$
$[q_0, a, q_1] \longrightarrow a\,[q_0, a, q_0]\;[q_0, a, q_1]$
$[q_0, a, q_1] \longrightarrow a\,[q_0, a, q_1]\;[q_1, a, q_1]$

$\delta(q_0, b, a) \longrightarrow (q_1, a)$ ← (skip) no operation

$[q_0, a, q_0] \longrightarrow b\,[q_1, a, q_0]$
$[q_0, a, q_1] \longrightarrow b\,[q_1, a, q_1]$

DOMS Page No.
Date / /

$\delta(q_0, b, a) \longrightarrow (q_1, a)$

$[q_1, a, q_0] \longrightarrow b\,[q_1, a, q_0]$

$[q_1, a, q_1] \longrightarrow b\,[q_1, a, q_1]$

$\delta(q_1, a, a) \longrightarrow (q_1, \epsilon)$ — Pop oper^n

$[q_1\ a, q_1] \longrightarrow a$

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

$[q_1, z_0, q_1] \longrightarrow \epsilon$

DOMS | Page No.
Date / /

Tutorial Questions

(1) Design PDA to accept language $L = \{a^n b^n / n \geq 1\}$ accepting by final states / empty stack.

(2) Design PDA to accept language $L = \{a^n b^{2n} / n \geq 1\}$ accepting by final states or empty stack.

(3) Design PDA to accept language $L = \{wcw^R / w \in (a,b)^*\}$

(4) $L = \{w@w^R / w \in (a,b)^+\}$
(Non-deterministic PDA)

(5) Construct a PDA which is equivalent to the following given CFG

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

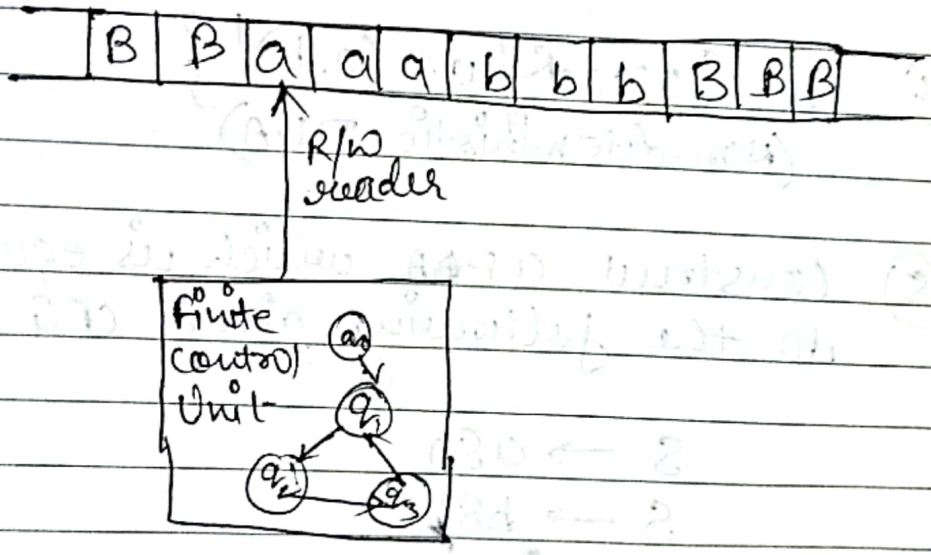
(6) Construct a PDA accepting $\{a^n b^m a^n / m,n \geq 1\}$ by null store & construct CFG accepting same.

DOMS Page No. Date / /

Turing Machine (TM)

Introduction

- A Turing m/c is capable of performing computations on input & producing new results.
- TM has infinite size tape and it is used to accept recursive enumerable languages.



- Input to TM is provided through long tape
- It has R/W header
- Tape :- each cell is capable to hold single symbol
- Blank Square holds 'B'

DOMS | Page No. | Date / /

Readers :-
- can read a symbol
- can modify symbol
- shifting Left/Right

- If the string is not in language, m/c will halt

* Mathematical Representation of TM

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

where,

Q = finite set of states
Σ = Finite set of i/p alphabets not containing B
Γ = Finite set of tape symbols includes B
q_0 = initial state
B = represents all empty cells
F = final state

$$Q \times \Sigma \longrightarrow Q \times \Gamma \times (L/R)$$

- After reading an i/p symbol, it is replaced with another symbol, its internal state is changed & it moves from one cell to the right or left.
- If TM reaches final state the i/p string accepted otherwise rejected.

Q1 Design a TM which recognizes the language $L = \{a^n b^n \mid n \geq 1\}$

(a) Definition

Step (1) Definition of TM:-

A TM is capable of performing computations on input & producing new results.

Mathematical Representation:-

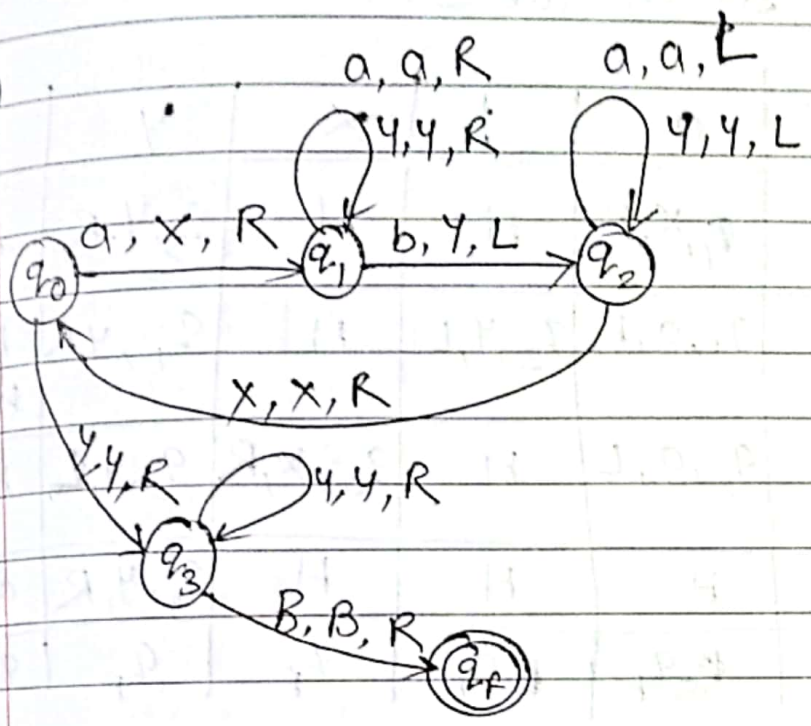
Refer previous page

Step (2) Logic:-

(1) Replace first 'a' with 'X' & keep going right direction until you encounter 'b'
(2) As you encounter 'b', replace it with 'Y' and move left
(3) Move left until you encounter 'X'
(4) 'Y' is encountered, move right until 'b' encountered.
(5) if 'b' encounter, follow step (2)
(6) Keep moving left until X is encountered

B	B	a	a	a	b	b	b	B

Step (1)



Step (2) Logic :-

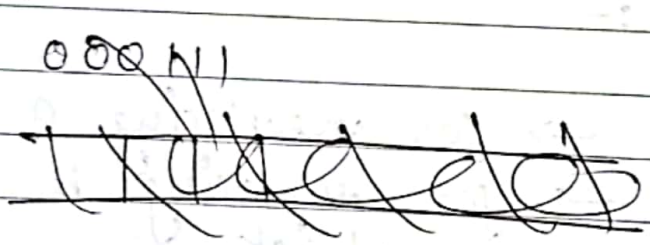
q_0 → for replacing a with X
q_1 → for replacing b with Y
q_2 → come back
q_3 → check if any b is left or blank (B) is there.

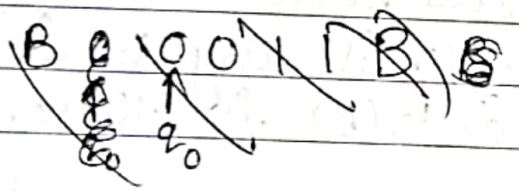
$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$
$Q = \{q_0, q_1, q_2, q_3, q_f\}$
$\Sigma = \{a, b\}$
$\Gamma = \{B, a, b, X, Y\}$

Step (4) Transition Table

Q | Γ	a	b	X	Y	B
q_0	q_1, X, R	H	H	q_3, Y, R	H
q_1	q_1, a, R	q_2, Y, L	H	q_1, Y, R	H
q_2	q_2, a, L	H	q_0, X, R	q_2, Y, R	H
q_3	H	H	H	q_3, Y, R	q_f, B, S
q_f	q_f	q_f	q_f	q_f	q_f

eg





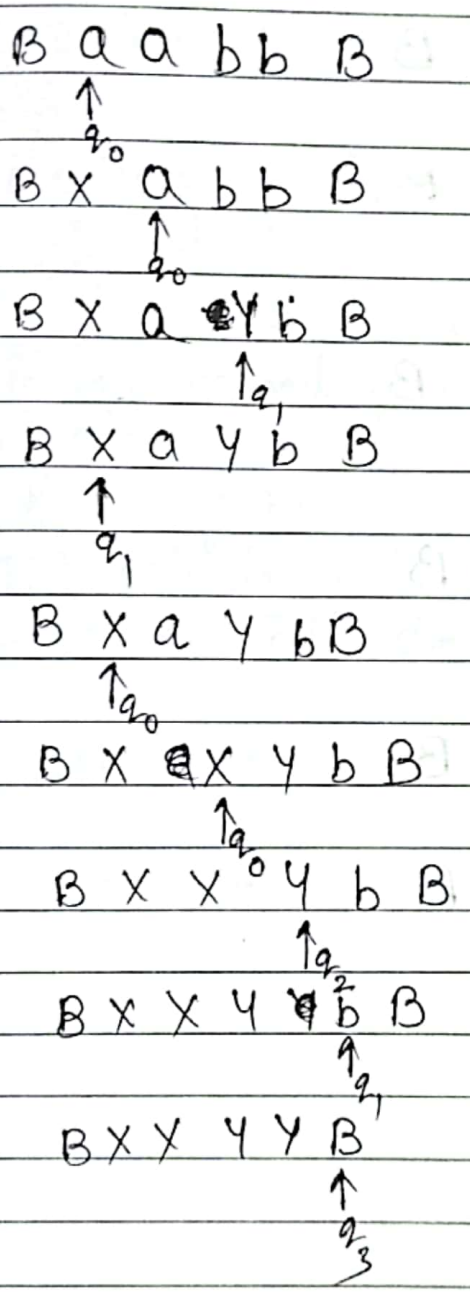
B X

Transition Function

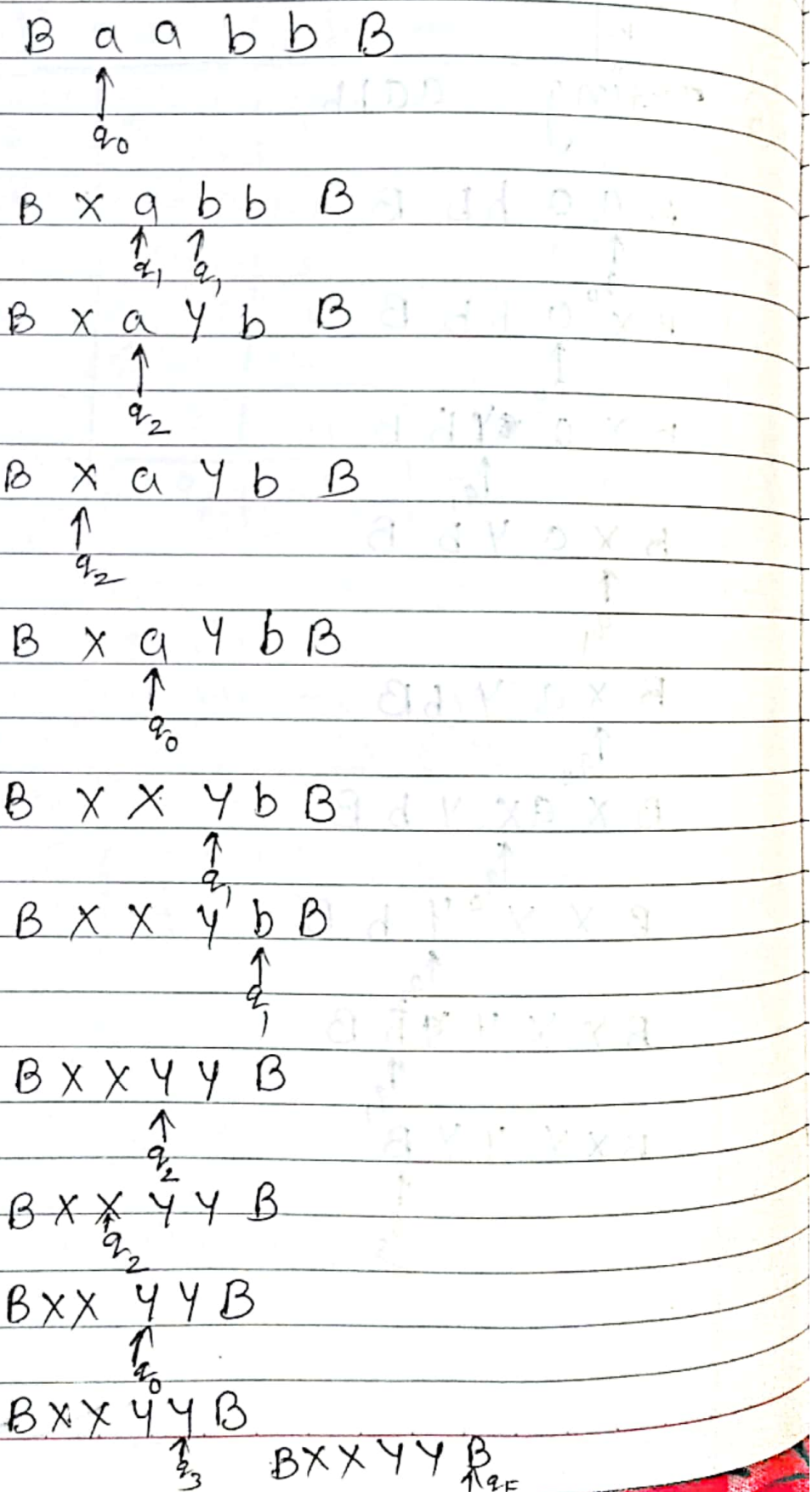
$\delta(q_0, a) = (q_1, X, R)$

B

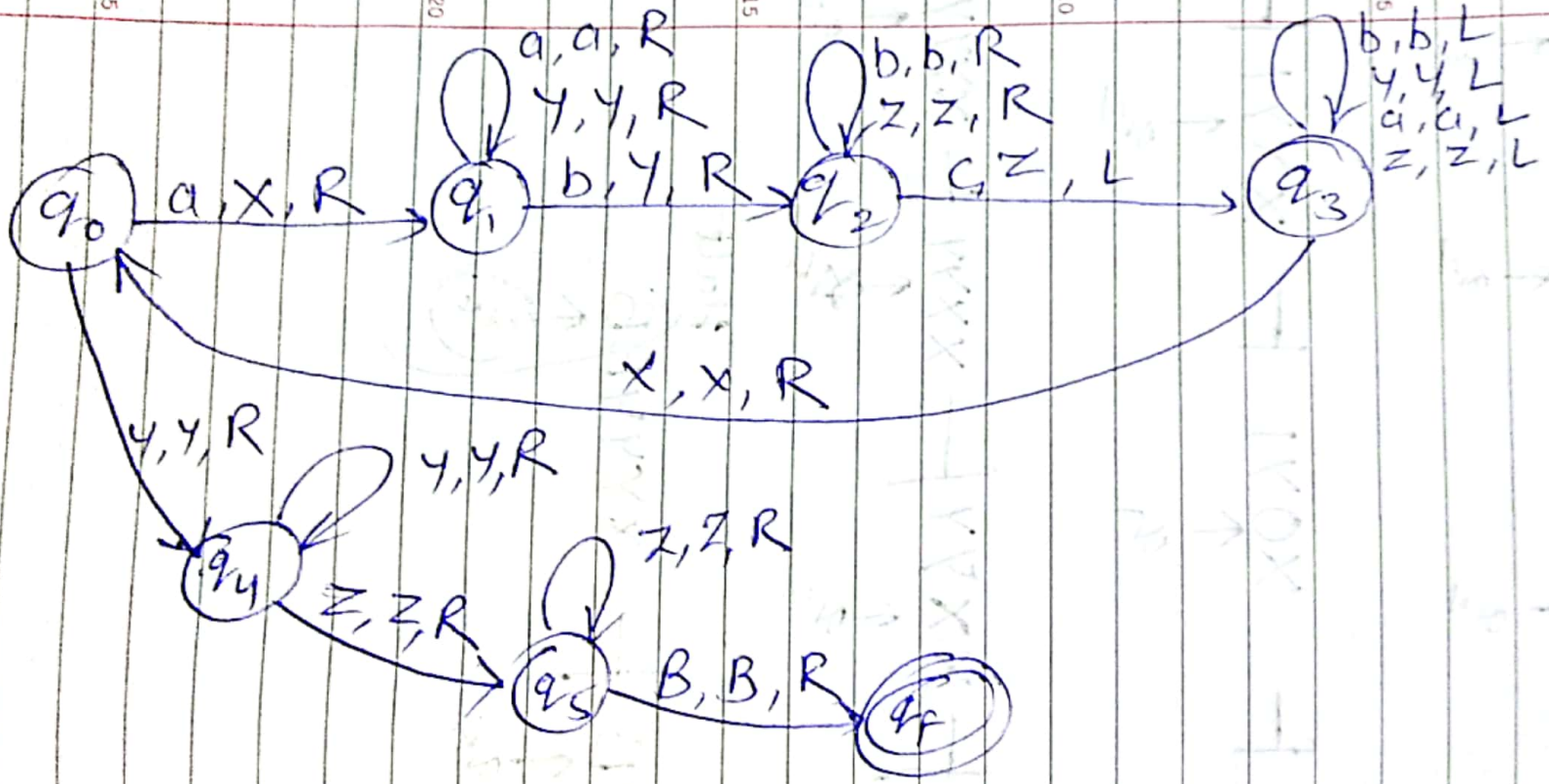
Step (5) Testing aabb



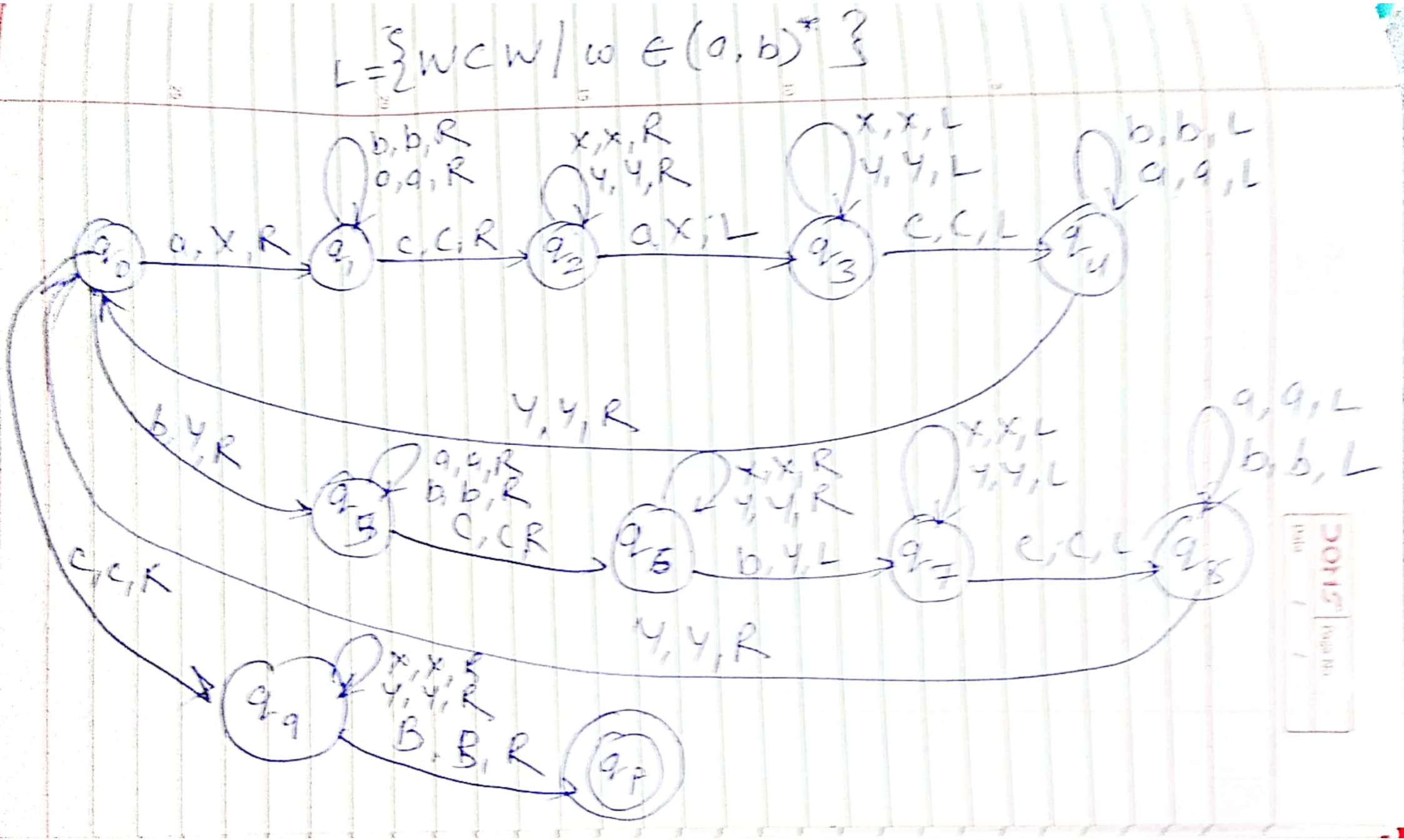
Step 5 Testing → a a b b



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



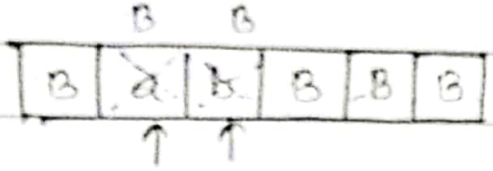
$L = \{ wcw \mid w \in (a,b)^* \}$



Q.1 Design a Turing machine that Erases all the Non-Blank symbols on the tape where the sequency of non-blank symbol does not contain any Blank symbol in between.

Sol: Step 1:- Theory

Step 2:- logic



$\delta(q_0, a) = (q_0, B, R)$ → replace/erase a and replace it with B, move towards Right

$\delta(q_0, b) = (q_0, B, R)$ → erase b and replace it with B & move towards right.

$\delta(q_0, B) = (q_1, N)$ → When all a's and b's are over

Step 8:- Transition Table.

	a	b	B
→ q_0	(q_0, B, R)	(q_0, B, R)	(q_1, B, N)
* q_1	q_1	q_1	q_1

Halting state or Final state.

Imp

Q.2. Design a Turing machine which accepts all the string of the form $a^n b^n$ for $n >= 1$ and rejects all the other strings. OR.

for $0^n 1^n$ $n >= 1$

Solution:- Step 1:- Theory

Step 2:- Logic.

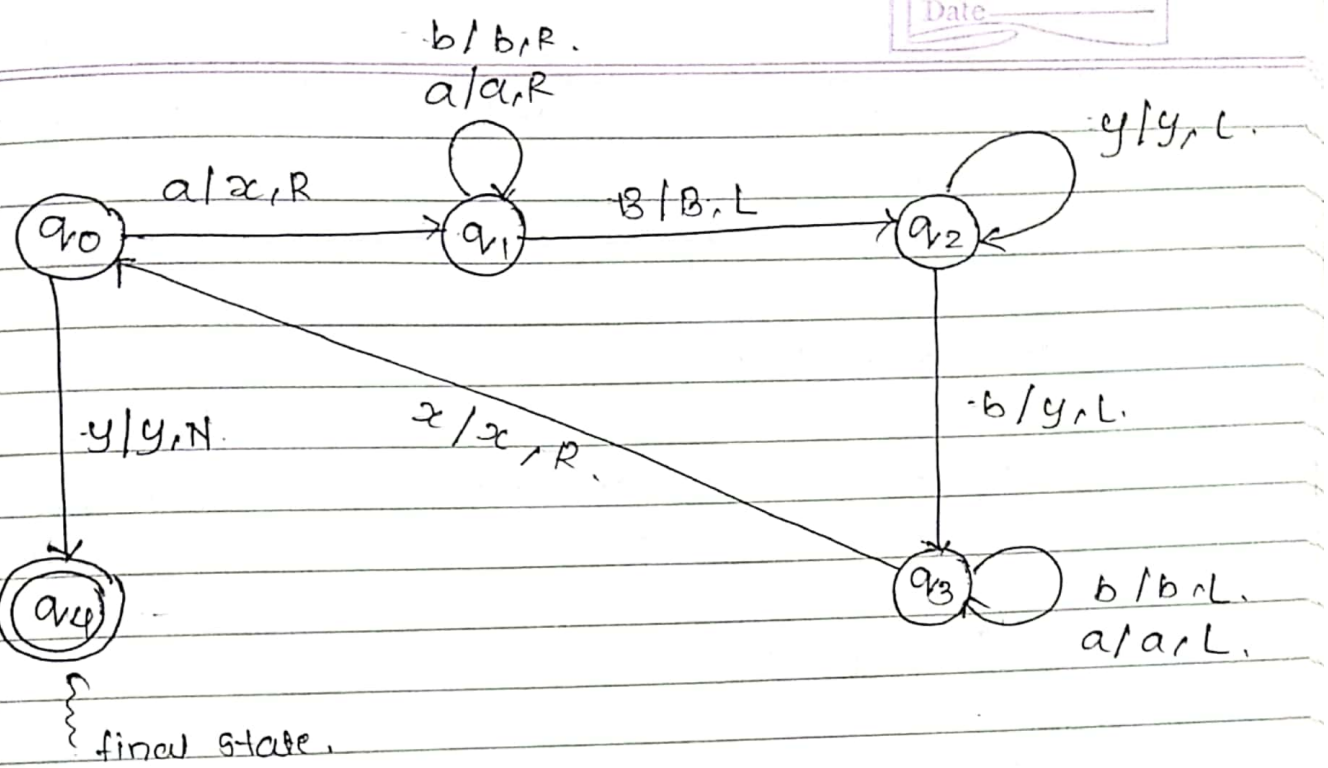
left most 'a' is changed to 'x'
right most 'b' is changed to 'y'
Head comes back to first 'a'

cycles :- Form mapping x & y.

B	a	a	a	b	b	b	B	⇒	B	x	a	a	b	b	y	B

B	a	a	a	b	b	b	B	⇒	B	x	x	a	b	y	y	B

B	a	a	a	b	b	b	B	⇒	B	x	x	x	y	y	y	B



1. q_0 → leftmost a has to be replaced with x.
2. q_1 → state q_1 locates the last B by skipping a's & b's & taking 'a' left turn
3. q_2 → right most b is changed to 'y'.
4. q_3 → located the first 'a' by skipping a's & b's by moving left & taking a right turn on first 'x' and entering q_0
5. q_4 → q_4 is a halt state.

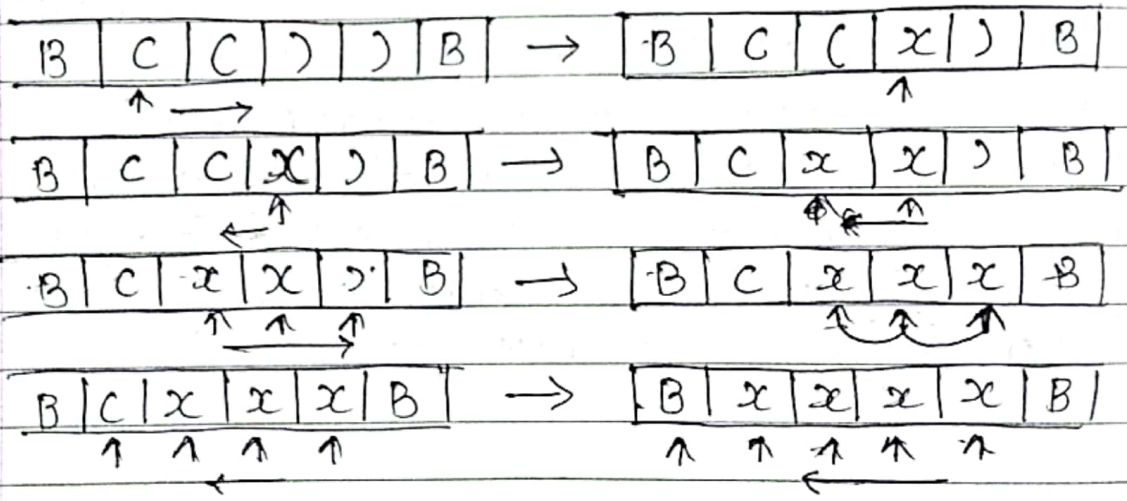
Q.3. Construct a Turing m/c to check well formedness of parenthesis.

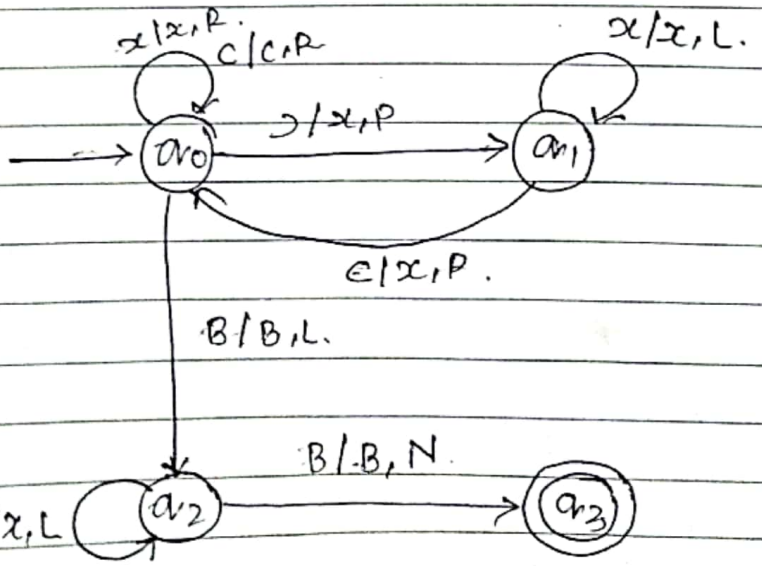
Solution:- Step 1:- Theory.

Step. 2:- logic.

In each cycle leftmost closing braket is written as x, then head moves towards left to locate the nearest openning bracket & it is changed to x'.

Step 3:- Cycles of computation



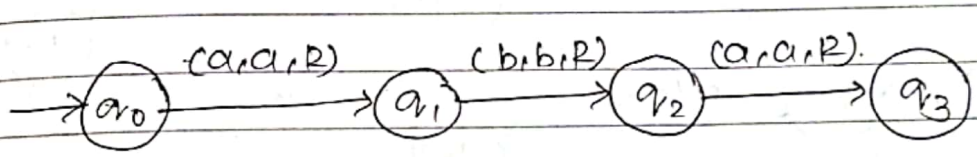


Q.4. Design a Turing M/C that recognized a palindrome over a variable a, b.

Q.5. Design a Turing M/C that recognized aba ~~substring~~ over (a, b).

Q.6. Design a TM that recognized words of form $a^n b^n c^n$.

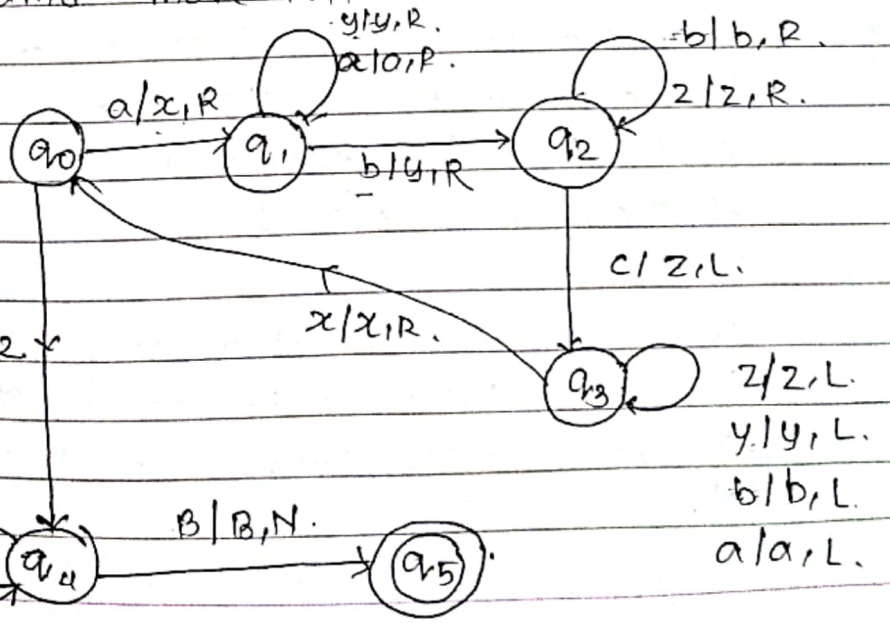
Q.5.
Sol:



Q.6. $a^n b^n c^n$.

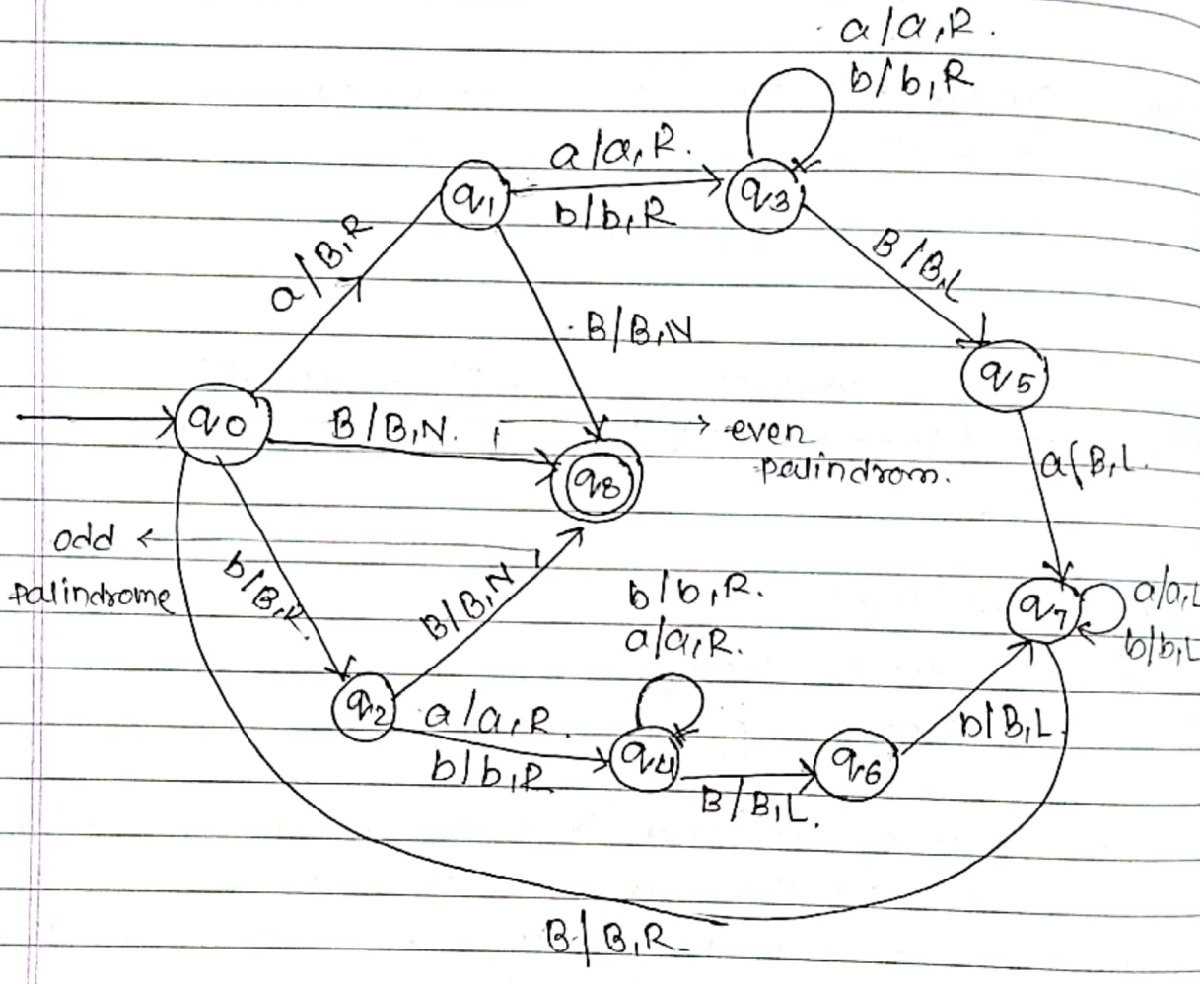
Solution: Step 1 :- Theory.
Step 2 :- Logic.

1. convert the first 'a' to 'x' and move towards right.
2. convert the ~~first~~ 'b' to 'y' and move right.
3. Skip the no of a's & b's and convert c to z and move left.



Q.4. Palindrome.

solution:- Palindrome type. i) ww^R.
ii) waw^R.
iii) wbw^R.

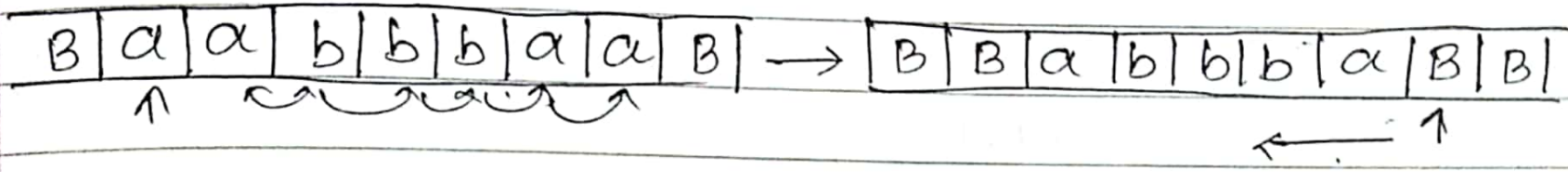


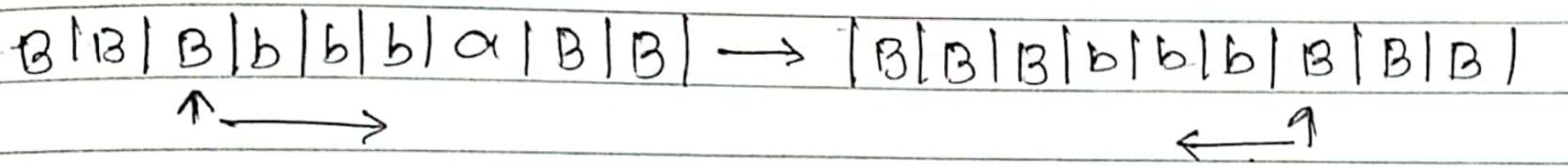
Iggic :- The algorithm requires n cycles in each cycle the first character is matched with the last character & both of them are erased

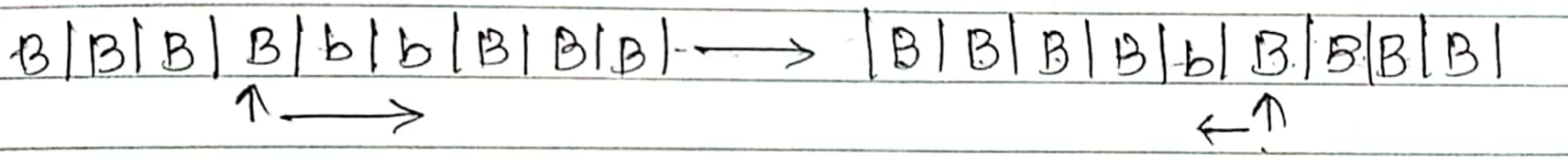
If the leftmost character is 'a' machine takes path from q_0, q_1, q_3, q_5, q_7, looking for the last character 'a'.

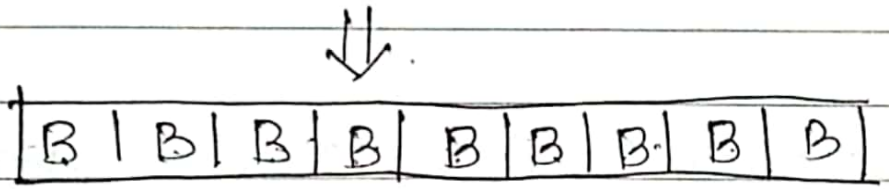
If the leftmost character is b the machine takes path from q_0, q_2, q_4, q_7 look for last character as 'b'.

cycles :-









Module No: 4 Push Down Automata.

1. Push Down Automata (PDA) is used for recognizing context free grammer.
2. PDA is more powerful than FA because it contains a stack which can be used for remembering some information.

3. Components of PDA.

a. PDA consists of finite set of states, ~~tape~~ input tape, read head and a stack.

4. Working of PDA

a. PDA can change the state, or remain in the same state.
b. PDA can perform some stack operation.

* Defination of PDA

$$M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, f).$$

where,

$Q \rightarrow$ finite set of set.
$\Sigma \rightarrow$ finite set of input symbols.
$\Gamma \rightarrow$ stack alphabet.
$\delta \rightarrow$ Transition function $Q \times \Sigma \cup (\Sigma) \times \Gamma$
$q_0 \rightarrow$ start state
$z_0 \rightarrow$ initial stack symbol
$f \rightarrow$ finite set of final state.

$q_1, R, \epsilon \quad f, R$

Q1. Design a PDA for recognizing $L = \{a^n b^n / n >= 1\}$

Sol: step 1: Theory.

$L = \{ab, aabb, aaabbb, \ldots\}$

step 2: logic

for each a push 1 x

for each b pop 1 x

Implementation. (i.e combination's)

↳ Top

$\delta(q_0, a, R) = (q_0, xR)$

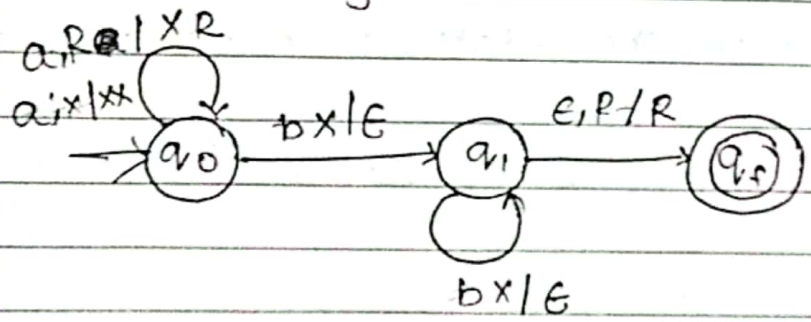
$\delta(q_0, a, x) = (q_0, xx)$

$\delta(q_0, b, x) = (q_1, \epsilon)$ $\epsilon \rightarrow$ indicates pop

$\delta(q_\phi, \epsilon, R) = (q_5, R) \rightarrow$ final state

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

Transition digram.



$\delta(q_0, aabb, R)$

$\delta(q_0, abb, xR)$

$\delta(q_0, bb, xxR)$

$\delta(q_1, b, xR)$

$\delta(q_1, \epsilon, R)$

$\{q_f, R\}$

Q2. Design a PDA for recognizing $L = \{0^n 1^n \mid n > 0\}$.

Sol: Step 1:- Theory.

Step 2:- logic.

i) For each '0' push 1x

ii) For each '1' pop 1x.

Step 3:- Implementation.

$\delta(q_0, 0, R) = (q_0, xR)$.

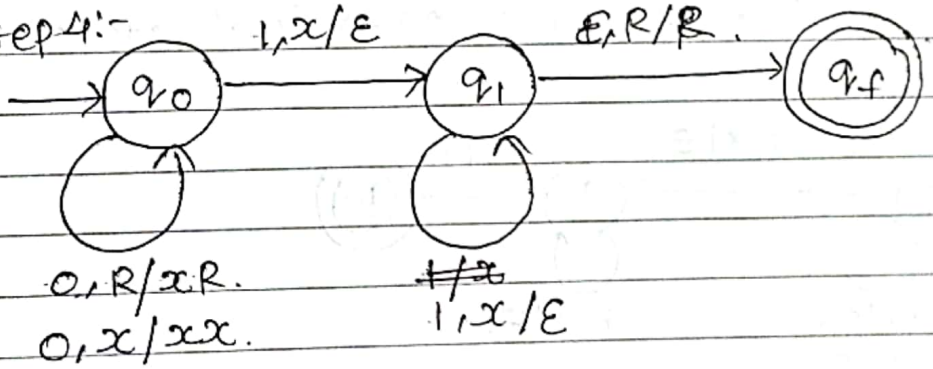
$\delta(q_0, 0, x) = (q_0, xx)$.

$\delta(q_0, 1, x) = (q_1, \varepsilon)$.

$\delta(q_1, 1, x) = (q_1, \varepsilon)$.

$\delta(q_1, \varepsilon, R) = (q_f, R)$.

Step 4:-



Step 5:- Simulation.

$\delta(q_0, 000111, R)$.

$\delta(q_0, 00111, xR)$.

$\delta(q_0, 0111, xxR)$.

$\delta(q_0, 111, xxxR)$.

$\delta(q_1, 11, xxR)$.

$\delta(q_1, 1, xR)$.

$\delta(q_1, \varepsilon, R)$.

$\longrightarrow (q_f, R)$.

Q.3. Design a PDA for recognizing $L = \{a^n b^{2n} \mid n \neq 0\}$

Sol: Step 1 :- Theory.

Step 2 :- logic.

for each a push 2 X ∵ b has power 2n
for each b pop 1 X.

Implementation.

↱ Top

$\delta(q_0, a, R) = (q_0, XX)$

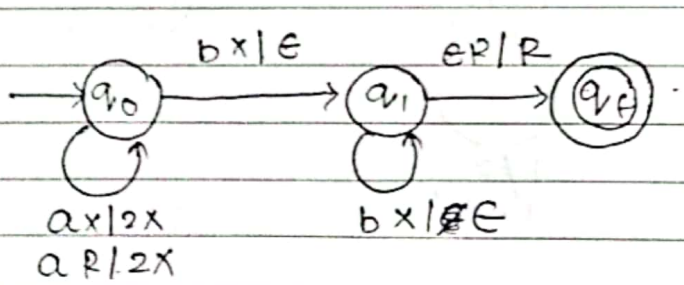
$\delta(q_0, a, X) = (q_0, XXX)$

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

~~$\delta(q_1, \epsilon, R)$~~

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

$\delta(q_1, \epsilon, R) = (q_f, R)$ → final state



$\delta(q_0, aabbbb)$ R.

$\delta(q_0, abbbb)$ XXR.

$\delta(q_0, bbbb)$ XXXXR.

$\delta(q_1, bbb)$ XXXR.

$\delta(q_1, bb)$ XXR

$\delta(q_1, b)$ XR.

$\delta(q_1, \epsilon)$ R.

q_f, R.

Q4. Design a PDA for recognizing $L = \{ c^n b^{3n} \mid n \geq 1 \}$.

sol: step 1: $L = \{ ccbbb, ccbbbbbb, \ldots\ldots \}$

step 2:- logic.

for each c push 3 x

for each b pop x

same implementation as of Q 3.

step 3:- Implementation:-

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

$q_0 = q_0$.

$z_0 = R$.

$\Gamma = \{ x \}$.

$f = q_f$.

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

$\delta(q_0, c, R) = (q_0, xxxR)$.

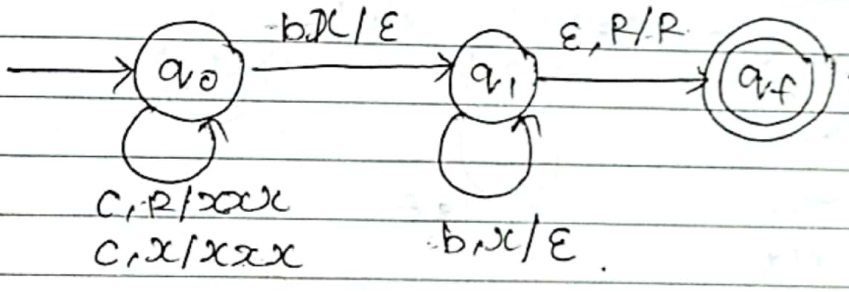
$\delta(q_0, c, x) = (q_0, xxxx)$.

$\delta(q_0, b, x) = (q_1, \varepsilon)$.

$\delta(q_1, b, x) = (q_1, \varepsilon)$

$\delta(q_1, \varepsilon, R) = (q_f, R)$.

Step 4:- State Transition Diagram



Step 5:- Simulation:

$\delta(q_0, ccbbbbbb, R)$.

$\delta(q_0, cbbbbbb, xxR)$.

$\delta(q_0, bbbbbb, xxxxxR)$.

$\delta(q_1, bbbbb, xxxxR)$.

$\delta(q_1, bbbb, xxxR)$

$\delta(q_1, bbb, xxR)$.

$\delta(q_1, bb, xR)$.

$\delta(q_1, b, R)$

$\delta(q_1, \varepsilon, R) \longrightarrow (q_f, R)$.

Q5. Design a PDA to recognize $L = \{0^m 1^n 2^{m+n} \mid m,n >= 1\}$

Sol: Step 1:- Theory.

Step 2:- logic

for each '0' push 1 X

for each '1' push 1 X

for each '2' pop 1 X ∵ m+n is power of 2.

Implemental

$\delta(q_0, 0, R) = (q_0, \underline{X}R)$

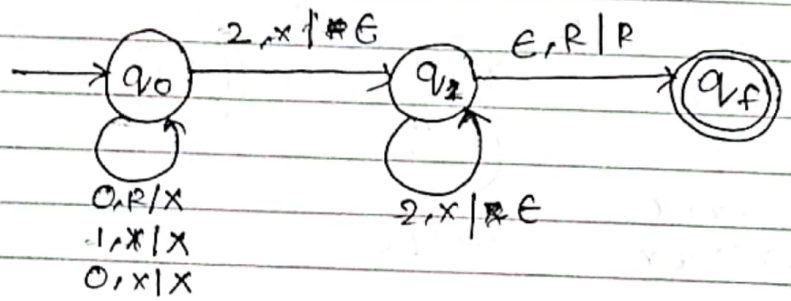
$\delta(q_0, 0, X) = (q_0, X\underline{X})$

$\delta(q_0, 1, X) = (q_0, X\underline{X})$

$\delta(q_0, 2, X) = (q_1, \epsilon)$

$\delta(q_1, 2, X) = (q_1, \epsilon)$

$\delta(q_1, \epsilon, R) = (q_f, R)$



$\delta(q_0, 0011 2222, R)$.

$\delta(q_0, 0011 2222, \underline{X}R)$

$\delta(q_0, 11 2222, \underline{X}XR)$

$\delta(q_0, 1 2222, \underline{X}XXR$

$\delta(q_0, 2222, \underline{X}XXXR)$

$\delta(q_1, 222, XXXR)$

$\delta(q_1, 22, XXR)$

$\delta(q_1, 2, XR)$

$\delta(q_1, R)$.

$\delta(q_1, \epsilon, R) \rightarrow (q_f, R)$

$q_0 \longrightarrow \phi$!

Q.6. Design a PDA to recognize $L = \{a^n b^m a^n \mid m, n \geq 1\}$.

Ans. $L = \{aba, abba, aabaa, aabbbaa, \ldots\}$.

Step 2 :- logic.

for each a push x.

for ~~each~~ sequence of b's → bypass all b's.

i.e (moment b's ↑ → for the final sequence of a's

pop x

(for each a's later).

Step 3 :- Implementation.

Here '—' indicates no change in stack. ~~tack~~ place.

$\delta(q_0, a, R) = (q_0, xR)$.

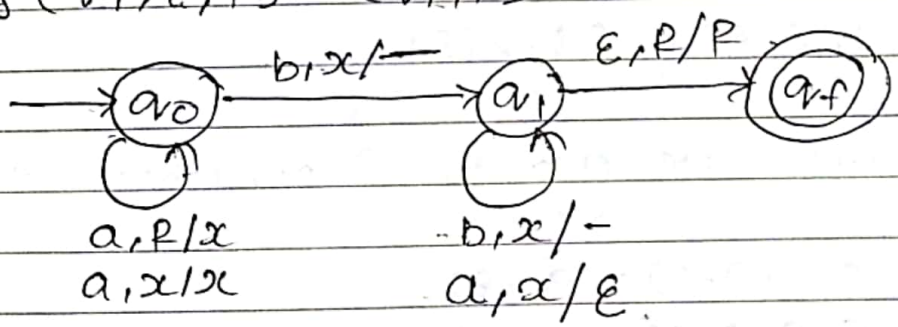
$\delta(q_0, a, x) = (q_0, xx)$.

$\delta(q_0, b, x) = (q_1, \overline{xx})$.

$\delta(q_1, b, x) = (q_1, \overline{xx})$

$\delta(q_1, a, x) = (q_1, \varepsilon)$

$\delta(q_1, \varepsilon, R) = (q_f, R)$.



$\delta(q_0, abba, R)$.

$\delta(q_0, bba, xR)$.

$\delta(q_1, ba, xR)$.

$\delta(q_1, a, xR)$.

$\delta(q_1, \varepsilon, R)$

(q_f, R)

Imp ***

Q.7. Design a PDA to accept a string of balanced parenthesis.

Solution:- Step 1:- Defination / Theory.

Step 2:- logic

1. for every 'c' push into stack. (opening brace)
   and for every ')' pop from stack. (close brace)
2. for every 'ε' push in stack
   & for every 'ε' pop from stack.

Implementation:-

$\delta(q_0, c, z_0) = (q_0, c, z_0)$
$\delta(q_0, \varepsilon, z_0) = (q_1, \varepsilon, z_0)$.
$\delta(q_0, c, c) = (q_0, cc)$.
$\delta(q_1, \varepsilon, c) = (q_1, \varepsilon c)$.
$\delta(q_1, \varepsilon, \varepsilon) = (q_1, \varepsilon\varepsilon)$

Now when we encounter) parenthesis then pop from stack.

$\delta(q_1,), c) = (q_1, \varepsilon)$
$\delta(q_1, \varepsilon, \varepsilon) = (q_1, \varepsilon)$

When we have read all symbols we read as ε.
$\delta(q_1, \varepsilon, z_0) = (q_f, z_0)$ → final state.

Simulation:- $\delta(q_0, (\varepsilon\varepsilon\varepsilon)\varepsilon, z_0)$
$= \delta(q_0, \varepsilon\varepsilon\varepsilon), cz_0)$
$= \delta(q_1, \varepsilon\varepsilon), \varepsilon cz_0)$
$= \delta(q_1, \varepsilon), \varepsilon cz_0)$
$= \delta(q_1,), \varepsilon cz_0)$
$= \delta(q_1,), cz_0) = \delta(q_1, \varepsilon, z_0) = \delta(q_f, z_0)$.

Q-8. Design a PDA for recognizing $\{L = a^m b^n c^n d^m \mid m, n \geq 1\}$.

Solution:- Step 1:- Defination / Theory.

$L = \{abcd, aabbccdd, abbccd, \ldots\}$

Step 2:- Logic.

for each 'a' push 1X.
for each 'b' push 1Y.
for each 'c' pop 1Y
for each 'd' pop 1X

Implementation :- $\delta(q_0, a, R) = \{q_0, XR\}$.
~~$\delta(q_0, b, R) = \{q_1, YR\}$~~
$\delta(q_0, a, X) = \{q_0, XX\}$
$\delta(q_0, b, X) = \{q_1, YX\}$.
$\delta(q_0, b, Y) = \{q_1, YY\}$

~~$\delta(q_0, c, Y) = \{\}$~~
$\delta(q_1, c, Y) = (q_2, \varepsilon)$
$\delta(q_2, c, Y) = (q_2, \varepsilon)$
$\delta(q_2, d, X) = (q_3, \varepsilon)$
$\delta(q_3, d, X) = (q_3, \varepsilon)$
$\delta(q_3, \varepsilon, R) = (q_f, R)$
↑ final state.

Simulation:-

$(q_0, aabbccd, R)$
$(q_0, abbccd, XR)$
$(q_1, bccd, YX)$
$(q_1, ccd, YYXR)$
(q_2, cd, YXR)
(q_2, d, XR)
(q_3, ε, R)
(q_f, R).

Q.a Desig a PDA for $L = wcw^R / w \in (a+b)^*$

c → constant

w^R → reverse of w)

sol :- step 1 :- Theory.

step 2 :- logic.

for each a push 1X & on a pop 1X.

for each b pop 1Y & on

when c comes bypass.

Implementation :- $\delta(q_0, a, R) = (q_0, XR)$.

$\delta(q_0, b, R)$

Q.a. Design a PDA for $L = wcw^R / w \in (a+b)^*$

c → constant.

w^R → reverse.

sol:- step 1 :- Theory

step 2 :- logic

for each a pop push 1X.

—11— b push 1Y.

when c encounterd bypass.

for each a pop 1X.

—11— pop 1Y.

implementation :- $\delta(q_0, a, R) = (q_0, XR)$

$\delta(q_0, b, R) = (q_0, YR)$.

push.

$\delta(q_0, a, X) = (q_0, XX)$.

$\delta(q_0, b, X) = (q_0, YX)$.

$\delta(q_0, a, Y) = (q_0, XY)$.

$\delta(q_0, b, Y) = (q_0, YY)$.

Bypass $\delta(q_0, C, R) = (q_1, R)$.

$\delta(q_0, C, X) = (q_1, X)$.

$\delta(q_0, C, Y) = (q_1, Y)$.

pop $\delta(q_1, a, X) = (q_1, \varepsilon)$

$\delta(q_1, b, Y) = (q_1, \varepsilon)$.

$\delta(q_1, \varepsilon, R) = (q_f, \varepsilon)$.

Simulation :- $\delta(q_0, abcba, R)$.

$\delta(q_0, bcba, XR)$

$\delta(q_0, cba, YXR)$

$\delta(q_0, ba, YXR)$.

$\delta(q_0, a, XR)$.

$\delta(q_1, \varepsilon, R)$.

(q_f, R).

Test :-

Q.1. Design a PDA $L = \{a^m b^n c^p \mid p = m+n ;\ m, n, p$ are integers greater than 0$\}$

Solution :- Step 1 :- theory.

Step 2 :- logic

for each a push 1X.

for each b push 1X.

for each c ~~push~~ pop 1X.

Implementation :- $\delta(q_0, a, R) = (q_0, XR)$

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

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

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

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

$\delta(q_1, \epsilon, R) = (q_f, P)$

simulation :- $\delta(q_0, abcc, P)$

$\delta(q_0, bcc, XR)$

$\delta(q_0, cc, XXR)$

$\delta(q_1, c, XR)$

$\delta(q_1, \epsilon, R)$

(q_f, R)

5/5

Q-2. Design a PDA for $L = \{a^n c b^{2n} \mid n > 1\}$ over the alphabets $\Sigma(a, b, c)$.

Solution :- step 1 :- Theory.

step 2 :- logic.

for each a push 2 x

for c bypass

for each b pop 1 x

Implementation $\delta(q_0, a, P) = (q_0, xxP)$

$\delta(q_0, a, x) = (q_0, xxx)$

$\delta(q_0, c, x) = (q_1, x)$

~~$\delta(q_1, \epsilon$~~ $\delta(q_0, c, x) = (q_1, xx)$

$\delta(q_1, b, x) = (q_1, \epsilon)$.

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

$\delta(q_1, \epsilon, P) = (q_f, P)$.

q_f final state

simulation :- $\delta(q_0, aacbbbbb, P)$

$\delta(q_0, acbbbb, xxP)$

$\delta(q_0, cbbbb, xxxP)$

$\delta(q_1, bbbb, xxxP)$

$\delta(q_1, bbb, xxP)$

$\delta(q_1, bb, xP)$

$\delta(q_1, b, P)$

$\delta(q_1, \epsilon, P)$

(q_f, P)

on non-terminal write its all production with ?

~~on terminal write $(q, a, a) = \{q\}$~~

* ~~PDA into~~ CFG. into PDA.

Rules.

1. For each non-terminal 'A' include transition.

$\Rightarrow \delta(q, \varepsilon, A) = \{(q, \alpha), (q, \beta), (q, \gamma)\}$

where α, β, γ are productions of non-terminal 'A' such that.

$A \longrightarrow \alpha \mid \beta \mid \gamma$.

2. For each terminal 'a' include the transitions

$\Longrightarrow \delta(q, a, a) = \delta(q, \varepsilon)$

Q1. Construct a PDA equivalent to a given CFG

$S \rightarrow OBB$

$B \rightarrow OS \mid 1S \mid 0$

$\delta(q, \varepsilon, S) = (q, OBB)$

$\delta(q, \varepsilon, O) = (q, OS)$

Solution:-

$\delta(q, \varepsilon, S) = \{(q, OBB)\}$.

$\delta(q, \varepsilon, B) = \{(q, OS)\}$.

$\delta(q, \varepsilon, B) = \{(q, 1S)\}$.

$\delta(q, \varepsilon, B) = \{(q, 0)\}$.

$\delta(q, 0, 0) = \{(q, \varepsilon)\}$.

$\delta(q, 1, 1) = \{(q, \varepsilon)\}$.

simulation :- $\delta(q_0, 0100, OBB)$.
$\delta(q_0, 100, BB)$.
$\delta(q_0, 100, 1SB)$
$\delta(q_0, 000, SB)$
$\delta(q_0, 000, OBBB)$
$\delta(q_0, 00, BBB)$.
$\delta(q_0, 00, 0BB)$
$\delta(q_0, 0, BB)$.
$\delta(q_0, 0, 0B)$.
$\delta(q_0, 0, B)$
$\delta(q_0, 0, 0)$
(q_0, ε)

Q.2. Construct a PDA equivalent to given CFG
$S \rightarrow aABC$
$A \rightarrow aB \mid a$.
$B \rightarrow bA \mid b$.
$C \rightarrow a$.

Sol: $\delta(q_0, \varepsilon, S) = \{(q_0, aABC)\}$.
$\delta(q_0, \varepsilon, A) = \{(q_0, aB)\}$.
$= \{(q_0, a)\}$.

$\delta(q_0, \varepsilon, B) = \{(q_0, bA)\}$.
$= \{(q_0, b)\}$.

$\delta(q_0, \varepsilon, C) = \{(q_0, a)\}$.

$\delta(q_0, a, a) = \{(q_0, \varepsilon)\}$.

$\delta(q_0, b, b) = \{(q_0, \varepsilon)\}$.

~~$\delta(q_0, BC, C) = \{(q_0, \varepsilon)\}$~~

simulation :- $\delta(q, aababa, S)$

$\delta(q, aababa, aABC)$.

$\delta(q, ababa, ABC)$

$\delta(q, ababa, aBBC)$

$\delta(q, baba, BBC)$.

$\delta(q, baba, bABC)$

$\delta(q, aba, ABC)$

$\delta(q, aba, aBC)$

$\delta(q, ba, BC)$.

$\delta(q, ba, bC)$

$\delta(q, a, C)$.

$\delta(q, a, a)$

(q, ϵ)

* N- PDA — Non-Deterministic Pushdown Automata.

1. The Non-Deterministic Pushdown automata is very much similar to NFA except that it has a stack for recording large amount of information in Last IN First out operation.

DPDA.

A PDA with restriction's that atmost one move is possible in any configuration is called Diterministic PDA

Q: Design a NPDA $L = ww^R / w \in (a,b)$.

w^R - reverse of w.

solution:-

case 1:- The middle of string is not reached hence we continue to push repetitive symbols on the stack.

case 2:- Middle of string is reached & hence we start poping the symbols from the stack.

Implementations :- start.

$\delta(q_0, \varepsilon, R) = \{q_0, \varepsilon)\}$.

$\delta(q_0, a, R) = \{q_0, XR)\}$.

$\delta(q_0, b, R) = \{(q_0, YR)\}$.

$\delta(q_0, b, X) = \{(q_0, YX)\}$.

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

$\delta(q_0, a, Y) = \{(q_0, XY)\}$.

$\delta(q_0, b, Y) = \{(q_0, YY), (q_1, \varepsilon)\}$

pop.

$\delta(q_1, a, X) = (q_1, \varepsilon)$.

$\delta(q_1, b, Y) = (q_1, \varepsilon)$

$\delta(q_1, \varepsilon, R) = \{q_f, \varepsilon)\}$

Simulation :- $(q_0, abbbba, R)$.

$(q_0, bbbba, xR)$.

$(q_0, bbba, yxR)$.

$(q_0, bba, yyxR)$ (q_1, bba, xR).

$(q_0, bca, yyyxR)$ (q_1, ba, yxR).

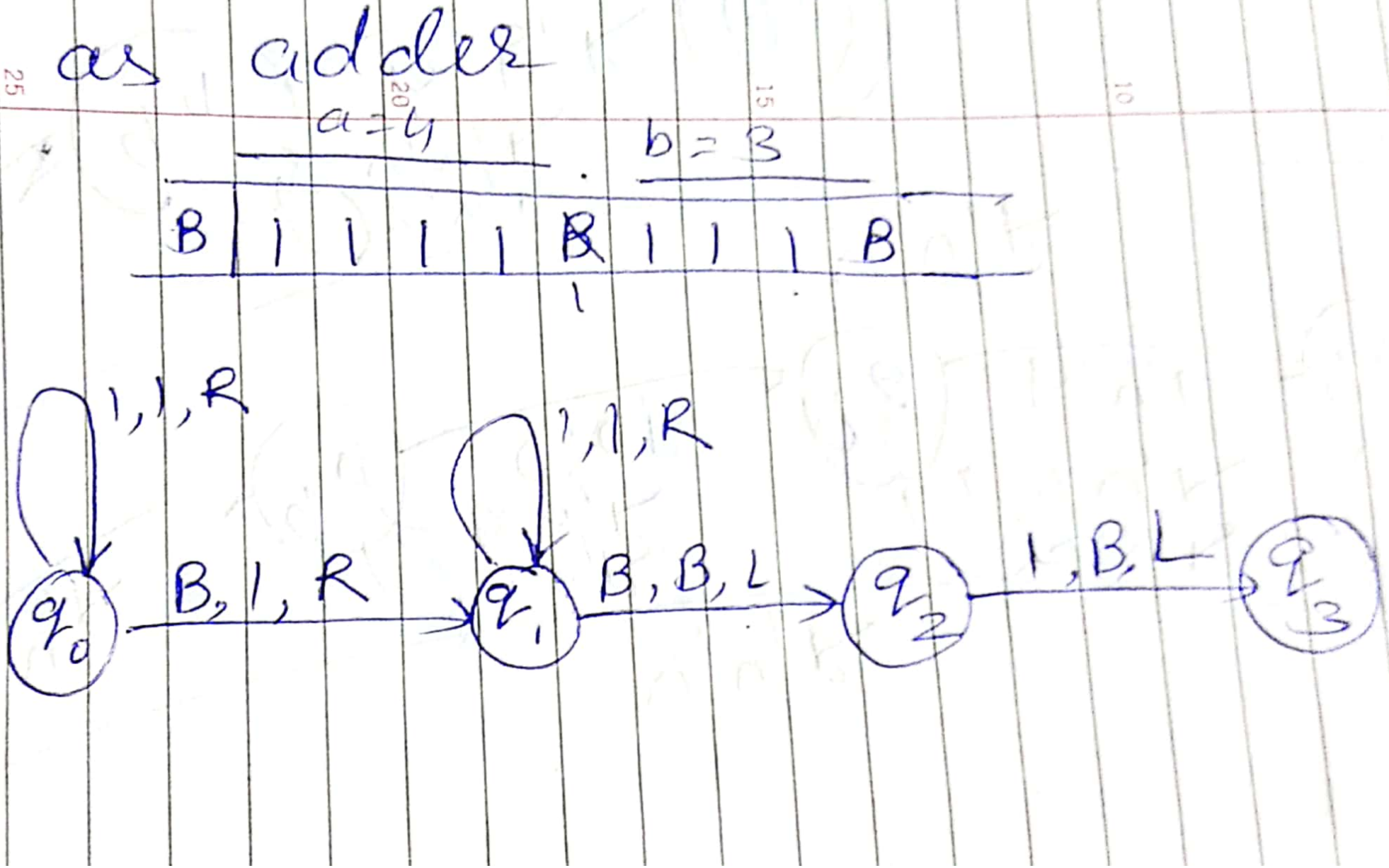
$(q_0, a, yyxxR)$ (q_1, a, xR).

(q_1, ε, R)

(q_f, R).

accepted.

TM as adder



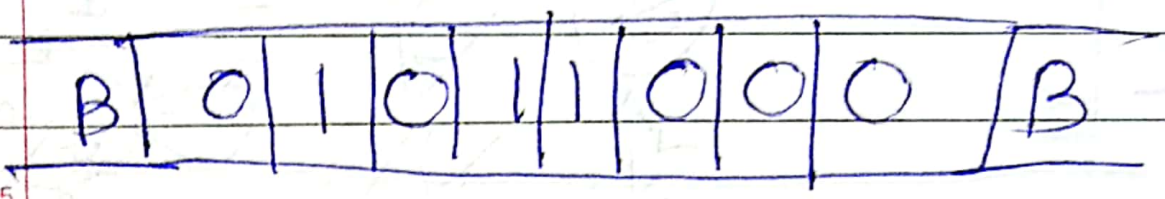
TM as 2's complement

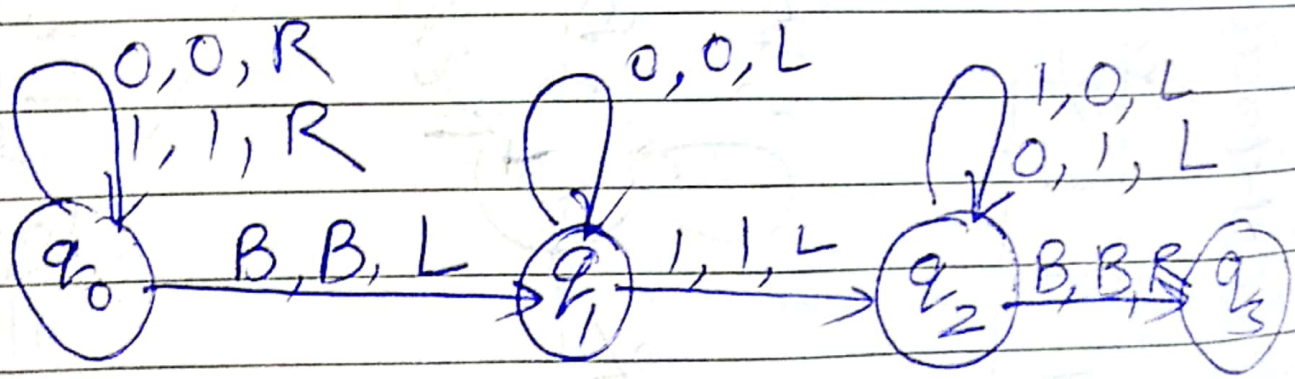
01011000

1's 10100111

+ 1

10101000





TM as Subtractor