TCS

28 Gauss

15/7/22

Alphabet Alphabet is a finite set of symbols.
Alphabet is denoted by 'Σ'

$\Sigma_1 = (a, b, c \cdots, z)$ set of letters in English

$\Sigma_2 = (0, 1 \cdots 9)$ set of (base 10) (digit)

$\Sigma_3 = (a, b \cdots, z, =)$: set of letters plus the special Symbols

$\Sigma_4 = (\ (\ )\ )$ set of open & close parantheses

String A string over alphabet is a finite sequence of symbol Σ.
The empty string will be denoted as ε

abtbz is a string over $\Sigma_1 = (a, b, c, d \cdots z)$
9021 is a string over $\Sigma_2 = (0, 1 \cdots 9)$

Language Language is a set of strings over an alphabet
It describe in yes / no. ~~Denotes~~ by L

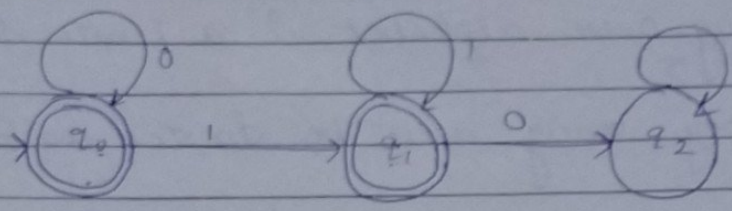
L_1 = The set of all string over Σ_1 that contains the substring 'tool'

L_2 = The set of all string over Σ_2 that contains

Inputs x states = no. of ways

DFA ⇒ deterministic finite automaton (DFA) is 5 tuple $(Q, \Sigma, \delta, q_0, F)$ where

- Q is a finite set of states
- Σ is a alphabet [Input]
- δ $Q \times \Sigma \to Q$ is a trasition function.
- $q_0 \in Q$ is a initial state
- $F \subseteq Q$ is a set accepting states (or final states)



alphabet = Σ (0, 1)

Total no of states

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

Initial State = q_0

Accepting States $F = \{q_0, q_1\}$

States	Input 0	Input 1
q_0	q_0	q_1
q_1	q_2	q_1
q_2	q_2	q_2

transition States

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

$\delta = (q_0, 1) = q_1$

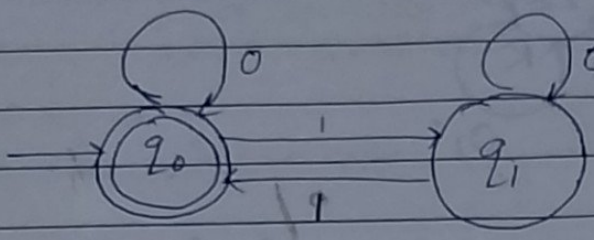
$(q_1, 0) = q_1$

$(q_1, 1) = q_2$

$(q_2, 0) = q_2$

$(q_2, 1) = q_2$

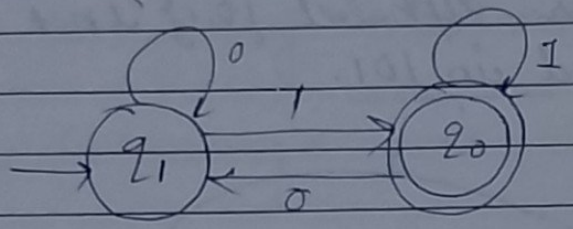
// the starting and ending state is same then ε is taken.

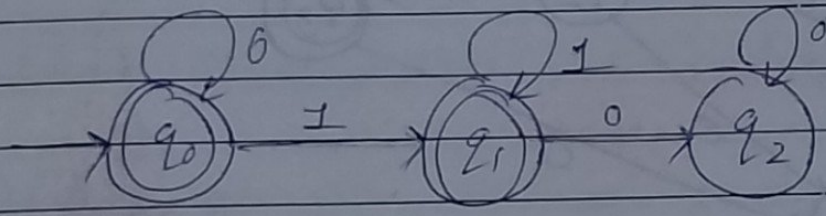


$L = \{ \varepsilon, 0, 00, 000, \ldots \}$

0^n is even odd

L = language where no. odd no. of 1 are not accepted.



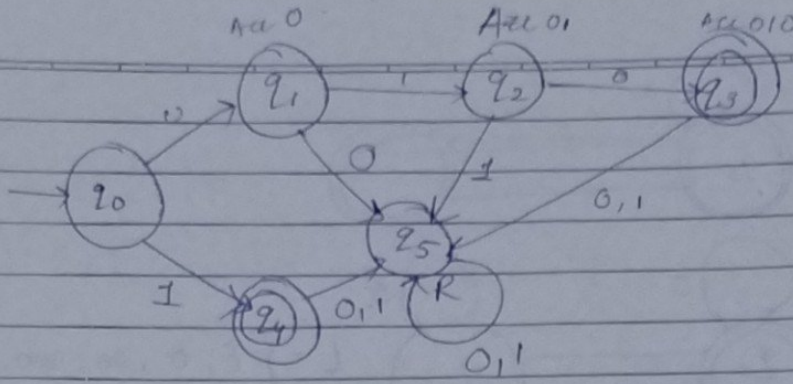


$L = \{ \varepsilon,$

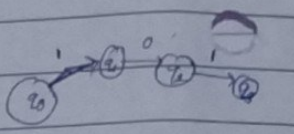
what are the languages of these DFAs

Construct a DFA that accepts the language.

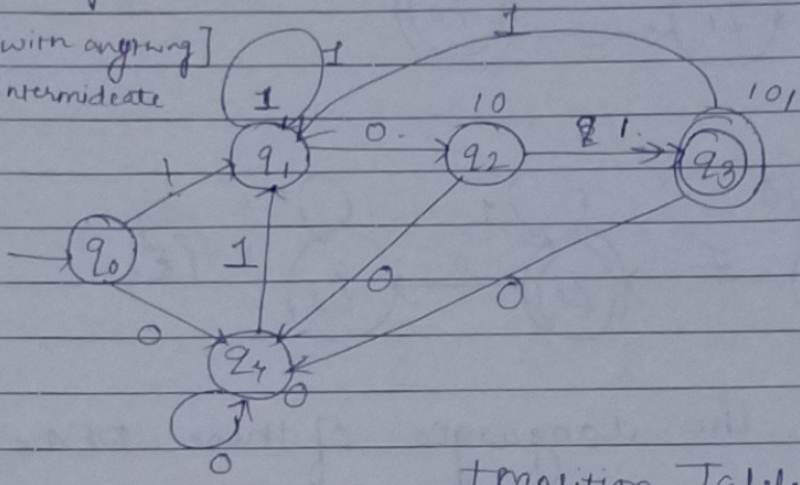
$L = \{010, 1\}$ $(\Sigma = \{0, 1\})$



- Construct a DFA over alphabet {0,1} that accepts all strings that end in 101.

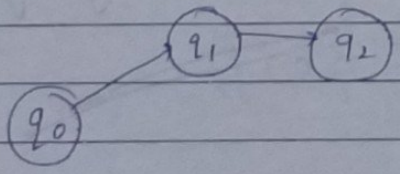


End with [Start with anything]
contains with [Intermidiate
Start with ξ



transition Table.

	Inputs	
	0	1
q_0	q_4	q_1
q_1	q_2	q_1
q_2	q_4	q_3
q_3	q_4	q_1
q_4	q_4	q_1



i) $S \times I \to S$
ii) $S \times I \to O$

i) Design a machine which checks whether a given decimal number is even or odd.

ii) String is acceptable if it ends with 'aba'.

iii) String is acceptable if it ends with '00' or '11'.

iv) String containing '1011' has a substring.

v) String Containings even no. of zeros & odd no. of ones.

vi) String ends with '00'.

vii) String starts with 0 & end with 1.

viii) Set of all the strings ending with 101

ix) String with exactly Two zeros any where.

x) Design a machine to check whether a given decimal no. is divisible by 3.

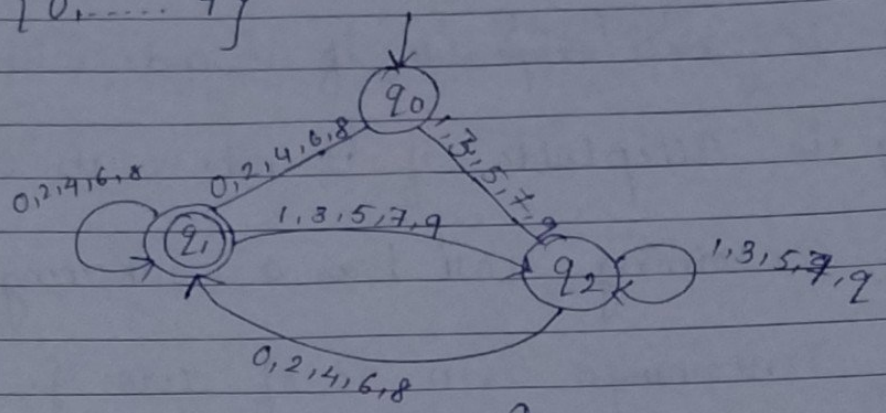
divide by 5 & 7

xi)

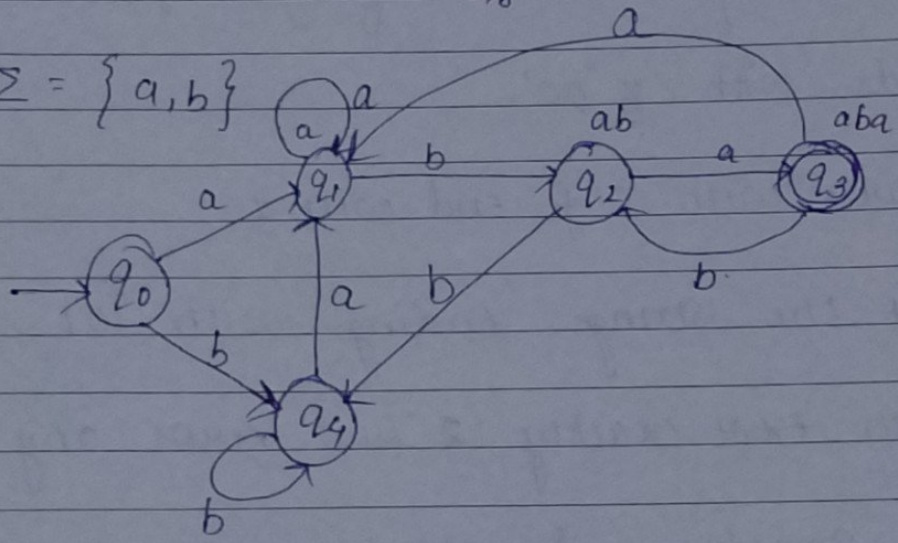
Solution

ii) $\Sigma = \{0, \ldots 9\}$

5mks / 10mks

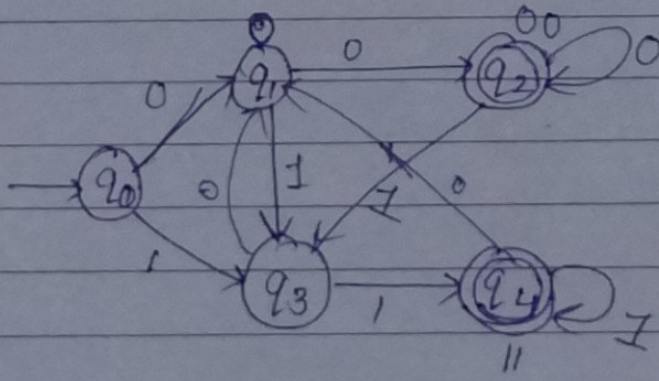


iii) $\Sigma = \{a, b\}$

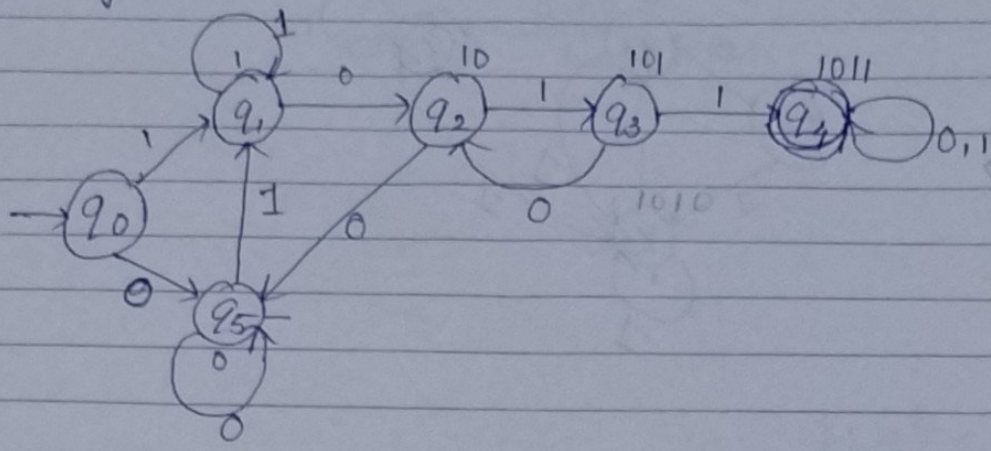


aba a/b.

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

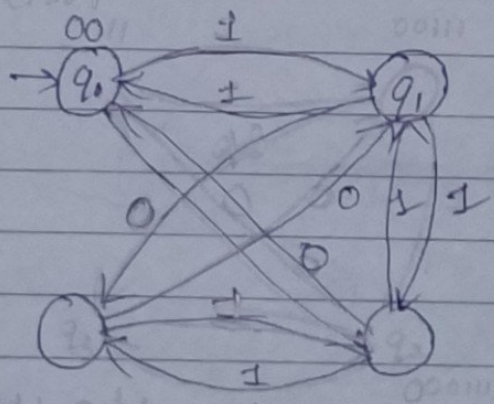


iv) Containing '1011'. $\Sigma = \{0, 1\}$

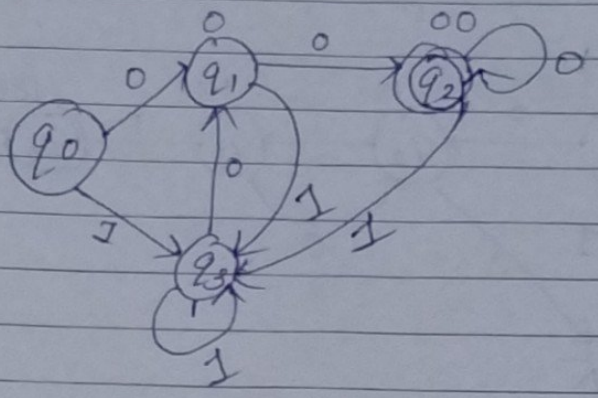


v) 00, 0000, 000000

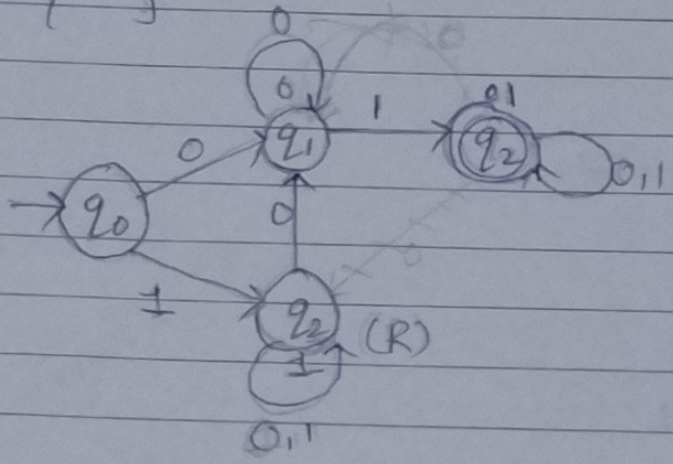
even 0	even 1
even 0	odd 1
odd 0	~~odd~~ even 1
odd 0	even 1



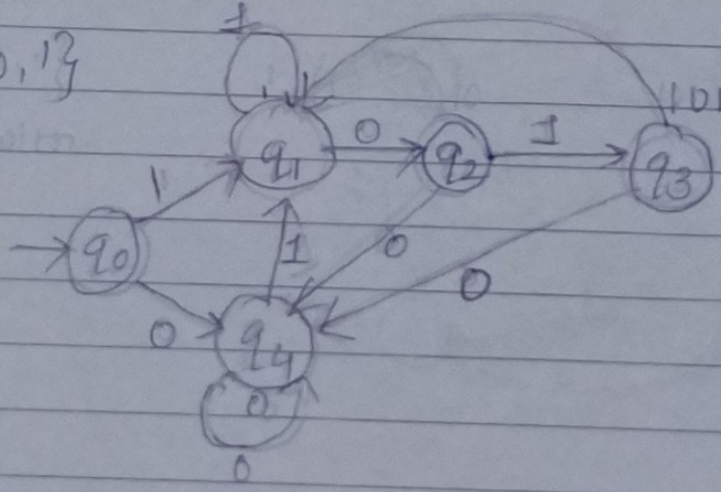
(vi)



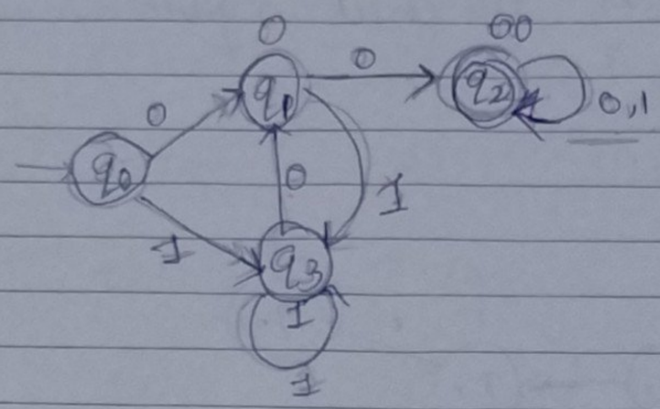
(vii) $\Sigma = \{0, 1\}$



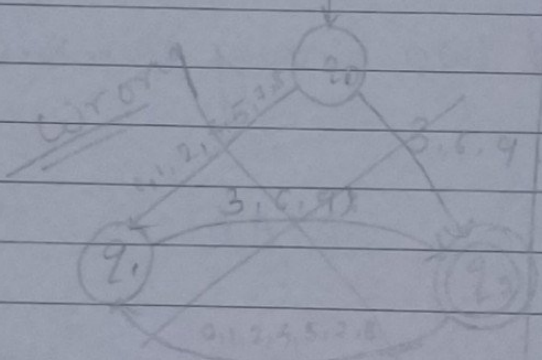
(viii) $\Sigma = \{0, 1\}$



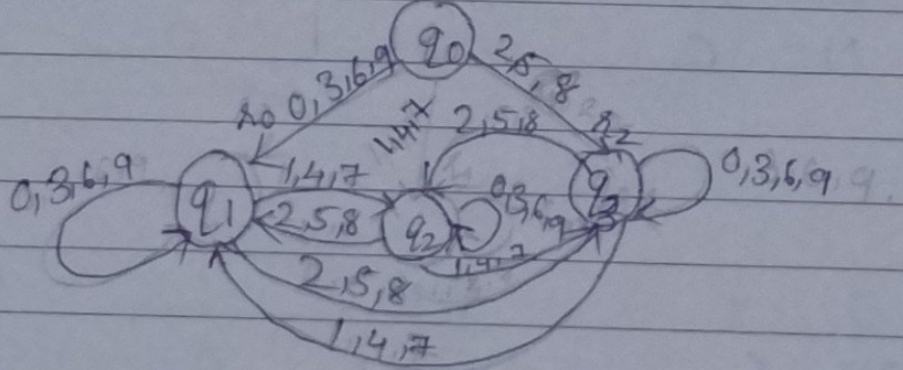
ix)



x) $\Sigma = \{0, 1, \ldots\ldots 9\}$



division by 3
→ so the
see the remainder



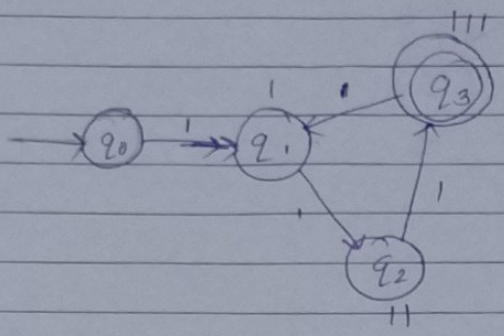
Imp 14 to 15 times in SEM

Q.11) Design ~~FA~~ FSM to chuk a given Unary no. ~~divisib~~ divisible by 3.

$\Sigma = \{1\} = 1!!!11$

1 1 1

Starts & ends with 1.



$S \times I \Rightarrow O$

Extreme Imp

Q.12 Design FSM to add two binary numbers of Equal length

STF MAF

STF MAF

State machine

fun fun

NC

$\frac{I_1}{I_2}$	0	1
0	NC	NC
1	NC	C

$\frac{IK}{S_2}$	0	1
0	0	1
1	1	0

$\frac{S_1}{I_2}$	0	1
0	NC	C
1	C	C

$\frac{I_1}{I_2}$	0	1
0	1	0
1	0	1

1+1+1

1+1+0

1+0+0.

1+0+11

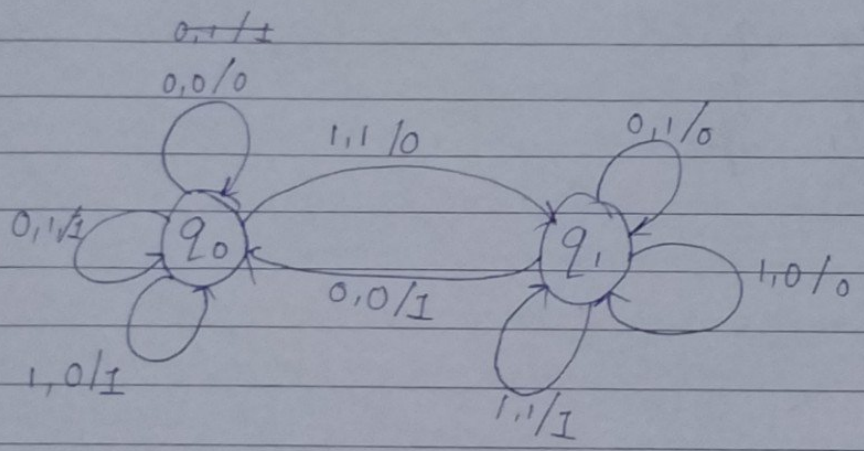
1 1 1

+ 0 1 0 1

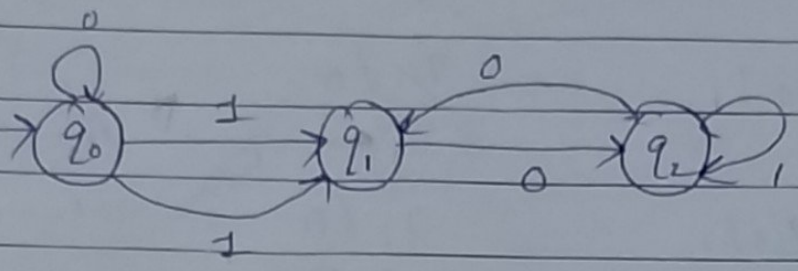
1 0

Transition table.

i_1	i_2	NC (q_0) ST (o/p)	NC (q_1) ST/o/p
0	0	q_0/0	q_0/1
0	1	q_0/1	q_1/0
1	0	q_0/1	q_1/0
1	1	q_1/0	q_1/1



binary number divisible by 3. (Type 1)

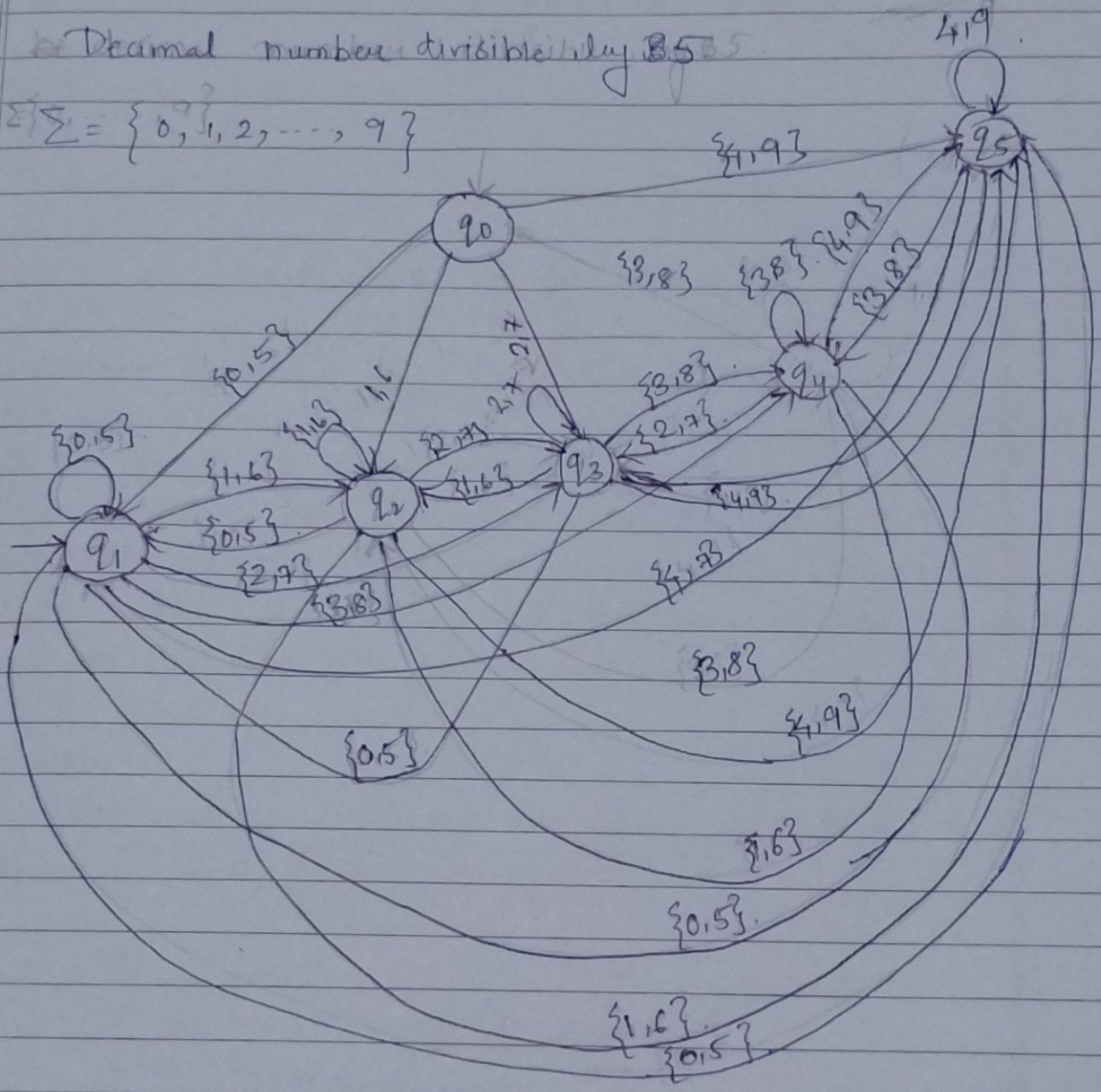


transition table

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

Decimal number divisible by 5

$\Sigma = \{0, 1, 2, \cdots, 9\}$



NFA

Non-deterministic finite Automata - [NFA]

NFA has two type
NFA with ε
NFA without ε

NFA can reside in multiple states at the same time.

★ Important Points

- NFA can be in several states at any time. Thus NFA can guess about an input sequence.
- Any NFA can be converted into Equivalent DFA.
- An NFA can be design in fewer States compared to DFA.
- DUE To Non-deterministic. NFA takes more time to Recognized the String.

Definiation.

Non-deterministic finite Automata is a five tuple

$M = \{ Q, \Sigma, \delta, q_0, F \}$

Where Q = finite set of States
Σ = finite set of Inputs
δ = transition fun" from $Q \times \Sigma = 2^Q$ (power set of Q)

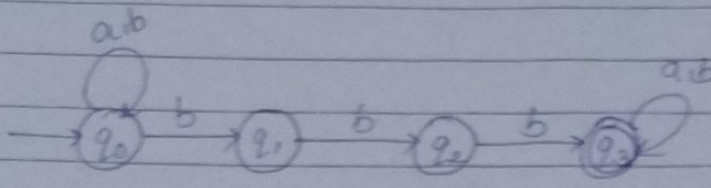
- Then $q_0 = q_0 \in Q$
  $F = F \subseteq Q$

difference of NFA & DFA

DFA is more faster

Design NFA which accepts strings Contains one occurance of 'bbb'.

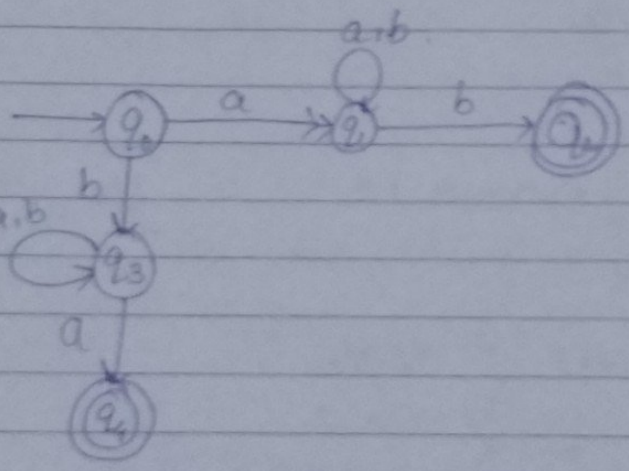
min cond^n → bbb



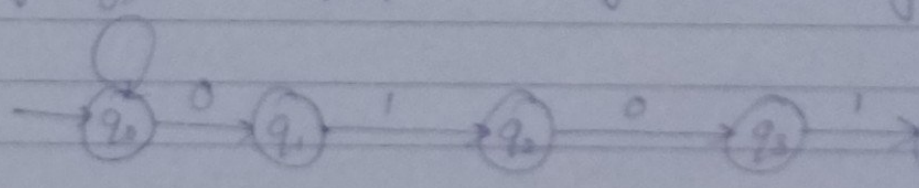
$\delta(q_0, a) = q_0$

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

Design NFA which accepts string starting 'a' & ending with 'b' or starting 'b' and ending with 'a'



Design NFA which accepts sub string containing 0101

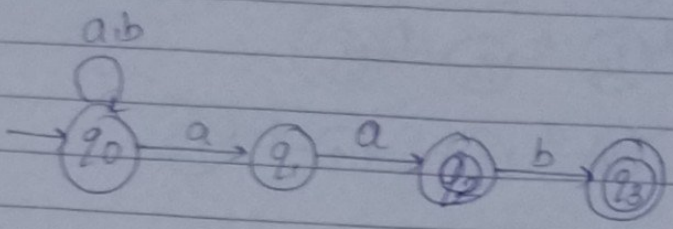


transition diagram + Table

Prog Design NFA that accepts any positive number of occurance substring 'ab' is given has $\{x \in \{a,b\}^* / x \text{ ends with } aab\}$

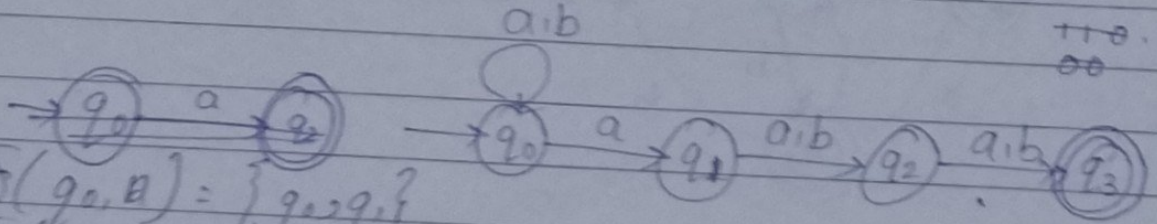
$\Sigma = \{a, b\}$

End



	a	b
q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\{q_2\}$	ϕ
q_2	ϕ	$\{q_3\}$
q_3	ϕ	ϕ

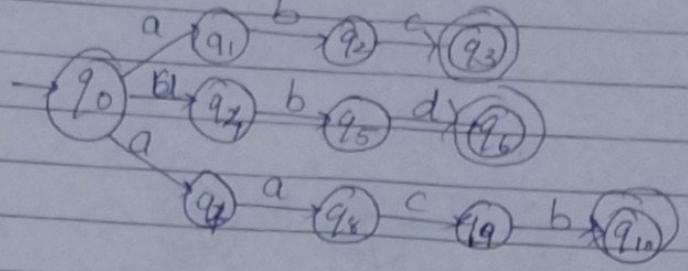
$\Sigma = \{0, 1\}$ and the String Contains 3rd Symbol from the right end is zero.



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

	a	b
q_0	$\{q_0, q_1\}$	q_0
q_1	q_2	q_2
q_2	q_3	q_3
q_3	ϕ	ϕ

$\Sigma = \{a, b, c, d\}$ Design NFA to recognized for strings
abc, abd, aacb
$L = \{$



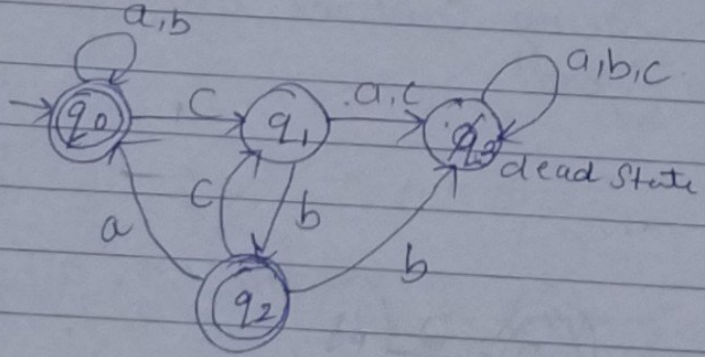
- Construct NFA and DFA

1) $L = \{x \in \{a, b, c\}^* : x$ contains exactly one b immediately follows c$\}$.

2) $L = \{x \in \{0, 1\}^* : x$ is starting with 1 & $|x|$ is divisible by 3.

3) $L = \{x \in \{a, b\}^* : x$ is contains any no of a's followed by atleast one b$\}$.

i)

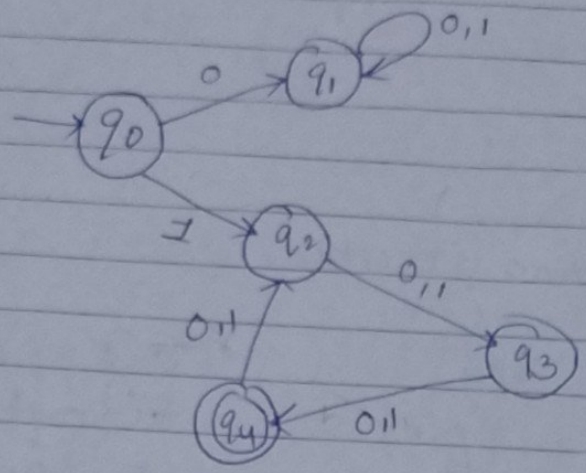


DFA & NFA

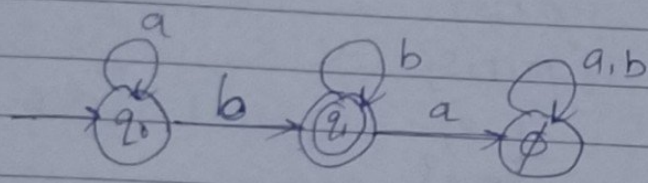
'|x| means length of x'.

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

{ x start with 1 Can't start with 0



iii)



* ⇒ final state

Convertion Problem

Given NFA convert it into Equivalent DFA.

	0	1
→p	{p,q}	{p}
q	{r}	{r}
r	{s}	∅
s*	{s}	{s}

Set of States {p,q} consider as 1 state

Subset Construction method.

	0	1
→ p	{p,q}	{p}
{p,q}	{p,q,r}	{p,r}
{p,q,r}	{p,q,r,s}	{p,r}
~~{p,q,r,s}~~ {p,q,s}	{p,q,s}	{p}
{p,r,s}	{p,q,r,s}	{p,r,s}
{p,q,s}	{p,q,r,s}	{p,r,s}
{p,r,s}	{p,q,s}	{p,s}
{p,s}	{p,q,s}	{p,s}

$\delta(\{p,q\},0) = \delta(p,0) \cup \delta(q,0)$

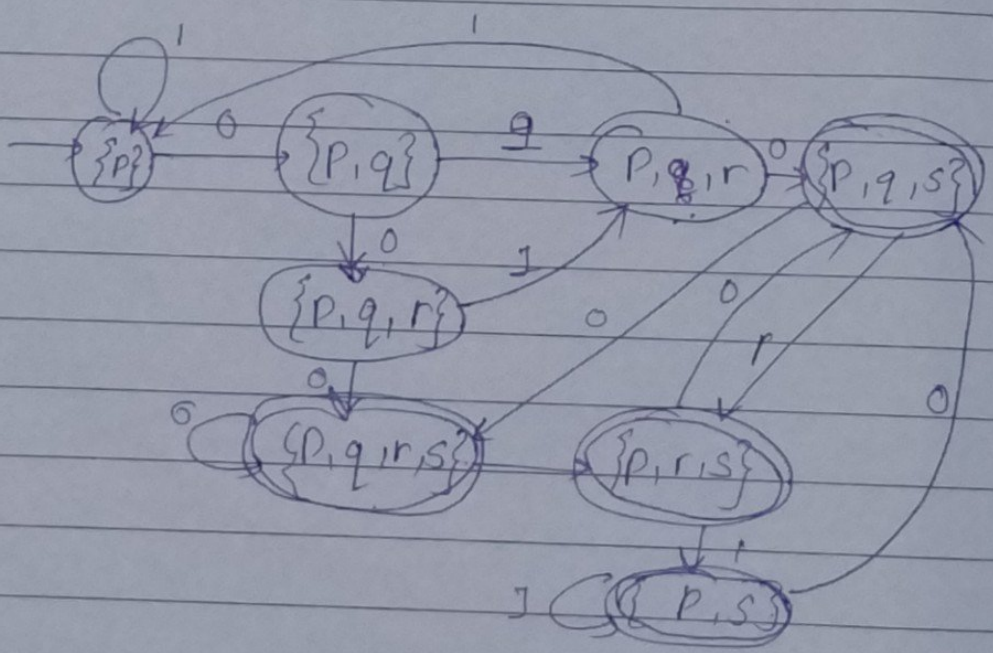
$\delta(\{p,q\},1) = \delta(p,1) \cup \delta(q,1)$

$= \{p,q\} \cup \{r\} = \{p,q,r\}$

$\delta(\{p,q\},1) = \delta(p,1) \cup \delta(q,1)$

$= \{p,$

$P \cup \phi = P$

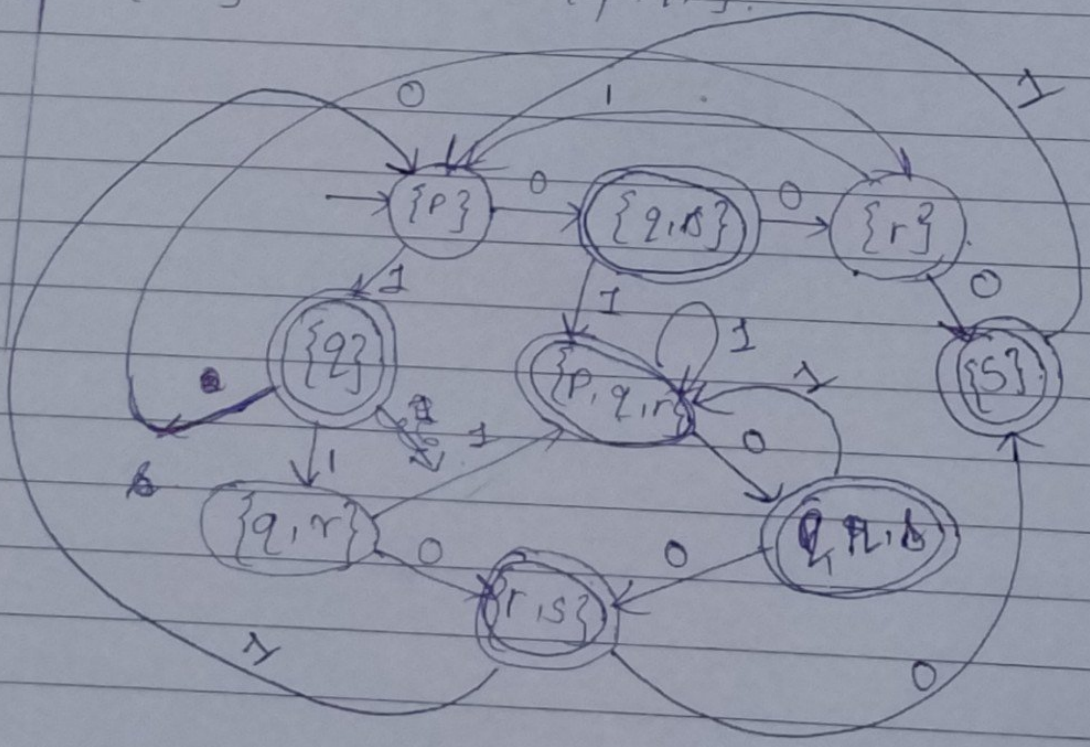


25/9/22.

	0	1
→p	{q,s}	{q}
q*	{r}	{q,r}
r	{s}	{p}
s*	∅	{p}

	0	1
→p	{q,s}	{q}
{q,s}*	{r,s}	{p,q,r}
{q}*	{r}	{q,r}
{r}	{s}	{p}
{q,r}*	{r,s}	{p,q,r}
{p,q,r}*	{q,r,s}	{p,q,r}
{s}*	∅	∅
{r,s}*	{s}	{p}
{q,r,s}*	{r,s}	{p,q,r}

δ({q,s},0) = δ(

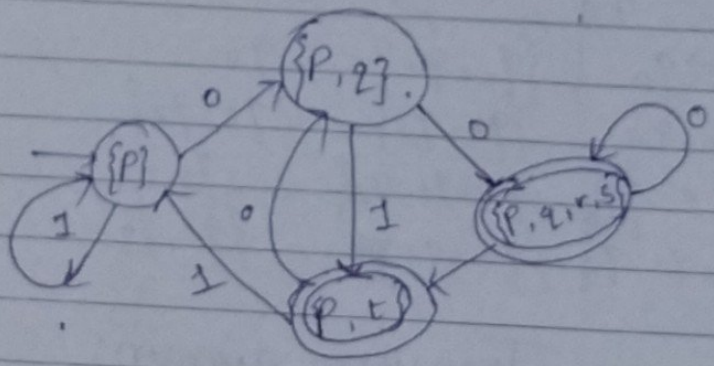


Conversion NFA to DFA

3)

	0	1
→ p	p,q	p
q	r,s	t
r	p,r	t
s*	∅	∅
t*	∅	∅

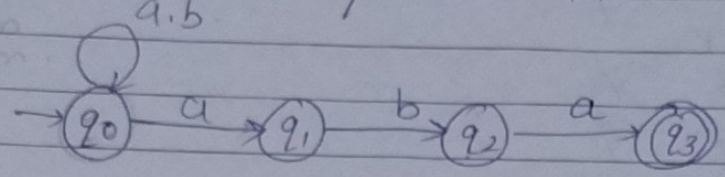
	0	1
→ p	{p,q}	p
p,q	p,q,r,s	p,t
{p,q,r,s}*	p,q,r,s	p,t
{p,t}*	p,q	p



tof
times

• NFA

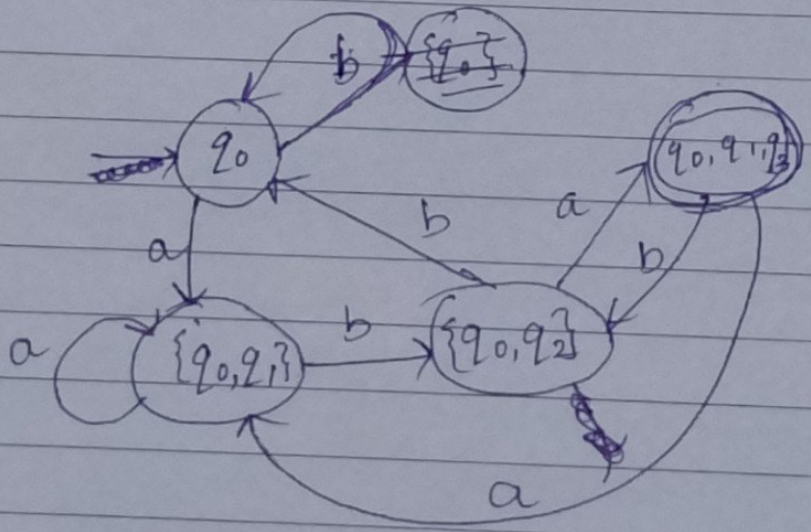
Construct NFA that accepts set of all string over a,b ending with aba. Use this NFA to construct an Equivalent DFA.



Transition Table

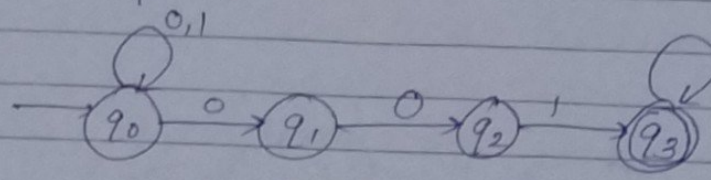
	a	b
q_0	q_0, q_1	q_0
q_1	ϕ	q_2
q_2	q_3	ϕ
q_3	ϕ	ϕ
$\rightarrow q_0$	$\{q_0, q_1\}$	q_0
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	q_0
$\{q_0, q_1, q_3\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$

Transition digram:



phi 5.

Construct NFA $\Sigma = \{0,1\}$ each string ~~has~~ which has 2 consecutive 0's followed by 1. is accepted.

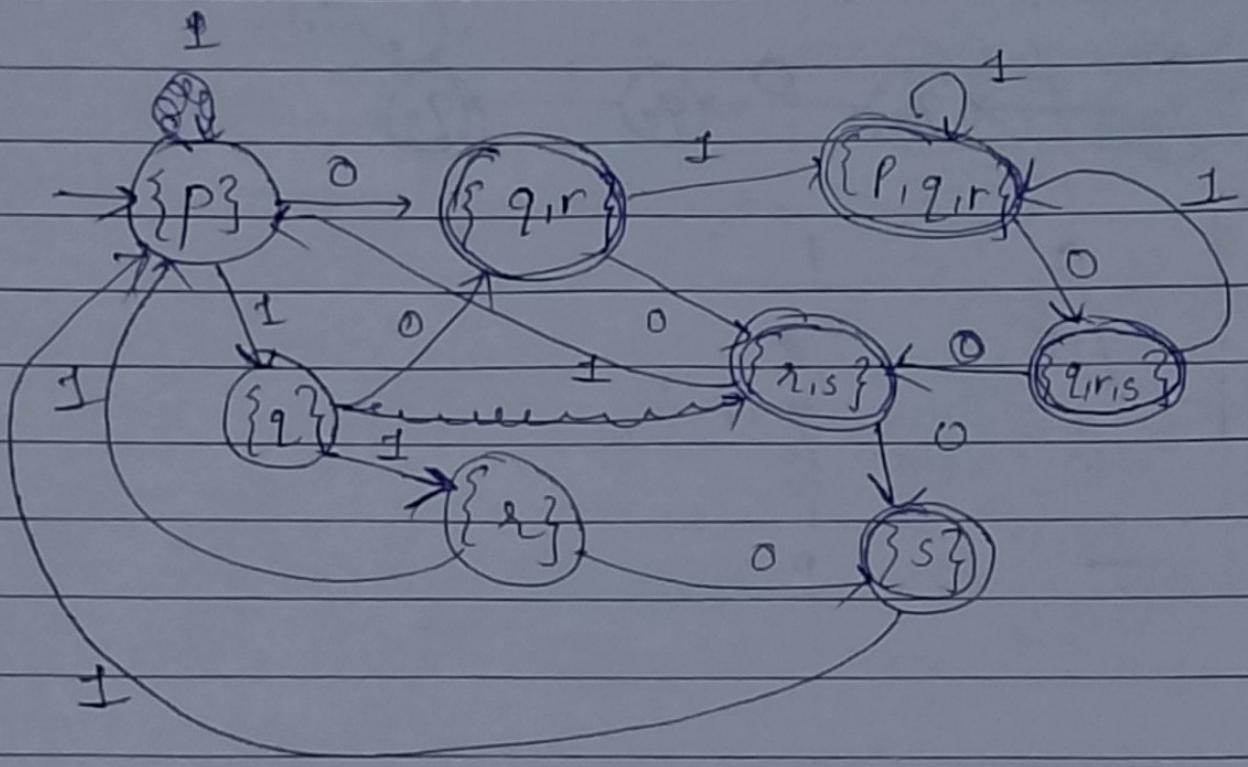


	0	1
→ p	q, r	q
q*	r	q, r
r	s	p
s*	—	p

Transition table:-

	0	1
→ p	{q, r}	{q}
*{q, r}	{r, s}	{p, q, r}
{q}	{r}	{q, r}
*{r, s}	{s}	{p}
*{p, q, r}	{q, r, s}	{p, q, r}
{r}	{s}	{p}
~~*{q, r}~~ {s}	~~{s}~~	{p}
*{q, r, s}	{r, s}	{p, q, r}

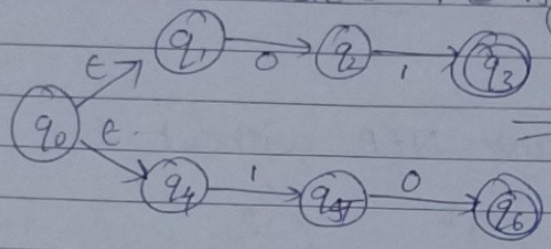
Transition diagram:



NFA with ϵ-transitions.

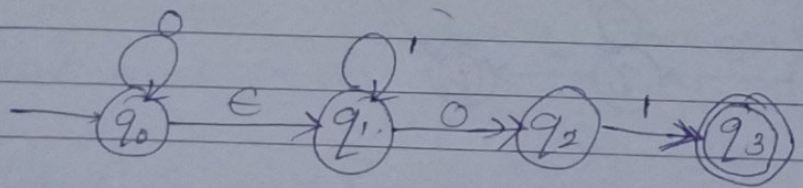
ϵ stands for null symbol
ϵ transition allows transition of ϵ

This implies that the machine can make ~~any~~ a transition without any input. When finding the string ~~desrip~~ decribe by ~~a part~~ path containing arc ~~table~~ with table ϵ, ϵ symbol is discarded.

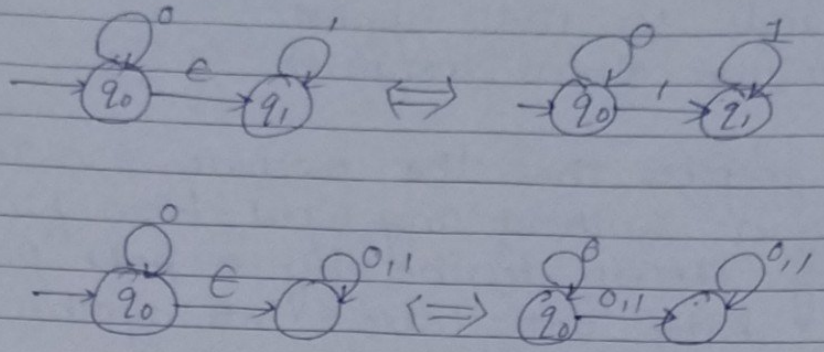


⇒ Ends with.
01 or 10.

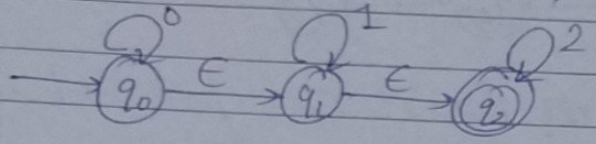
1. String starting with 0's. ~~or~~ more 0's. followed by 0 or. or more 1's if ending with 01.



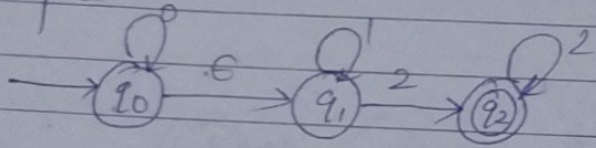
Equivalence of ∈NFA and NFA.



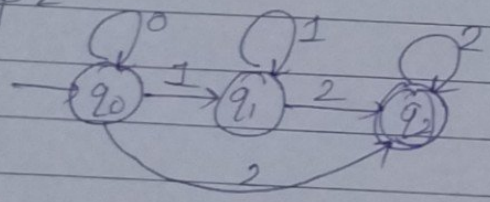
1 Find Equivalent NFA without ∈



Step 1.



Step 2:

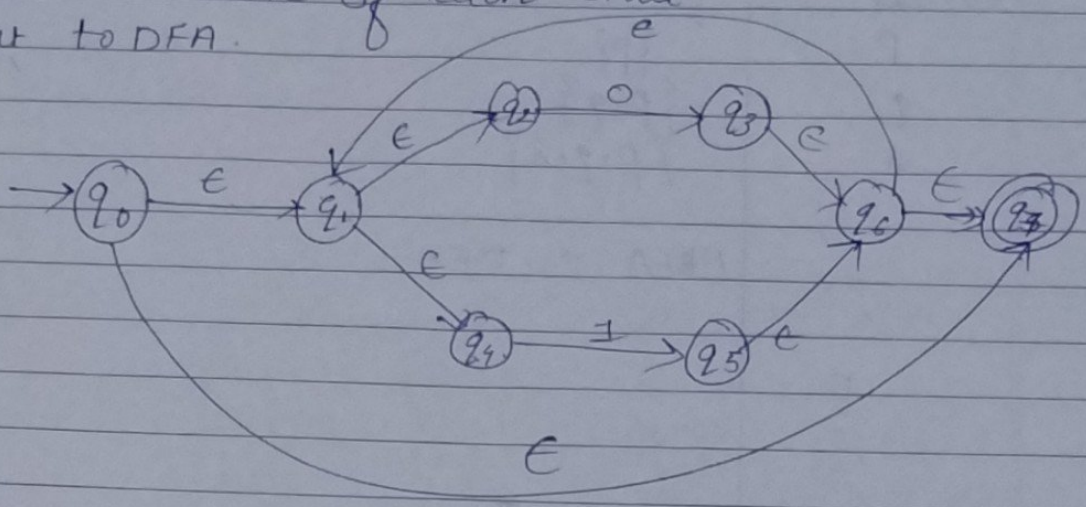


2 mks. SEM.

	∈	a	b	c
→p	∅	p	q	r
q	p	q	r	∅
r*	q	q	∅	p

a) Compute ϵ-closure of each state
b) Convert to DFA.

a)



* ϵ-closer of state q_i
i) ϵ-closer of state q_i include q_i itself.
ii) Set of states ~~reachable~~ reachable to q_i on ϵ move.
iii) set of states reacheable from existing states using ϵ-move and so-on.

ϵ-closer of $q_0 = \{q_0, q_1, q_2, q_4, q_7\}$.
ϵ-closer of $q_1 = \{q_1, q_2, q_4\}$.
ϵ-closer of $q_2 = \{q_2\}$
ϵ-closer of $q_3 = \{q_3, q_6, q_7\} q_1, q_2\}$
ϵ-closer of $q_4 = \{q_4\}$
ϵ-closer of $q_5 = \{q_5, q_6, q_7\} q_1, q_2, q_4\}$.
ϵ-closer of $q_6 = \{q_6, q_7\} q_1, q_4, q_2\}$
ϵ-closer of $q_7 = \{q_7\}$.

Moore Machine:

i) Collection of 5 things
ii) Set of states (Q)
iii) Set of i/p alphabets (Σ)
iv) Set of o/p symbols
4) Transition table
5) o/p table which shows that what o/p char is printed when each state is reached.

State gives o/p in Moore

Mealy machine.

i) Set of states.
ii) i/p alphabets
iii) Set of o/p char.
4) Pictorial representation as a graph, each state is vertex and each edge is labelled as i/p, o/p.

Input is associated with o/p in mealy.

1) Design Moore Machine to get 1's complement of given binary string.

0/1 0/1 0/1 ----

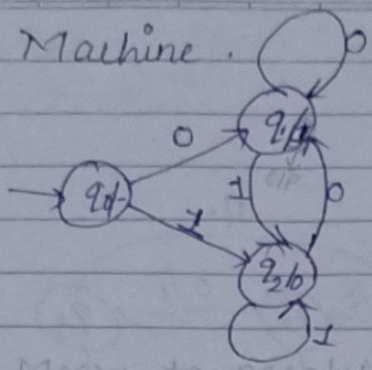
Nothing like a final state in Moore & Mealy m/c

1 continue Moore Machine.

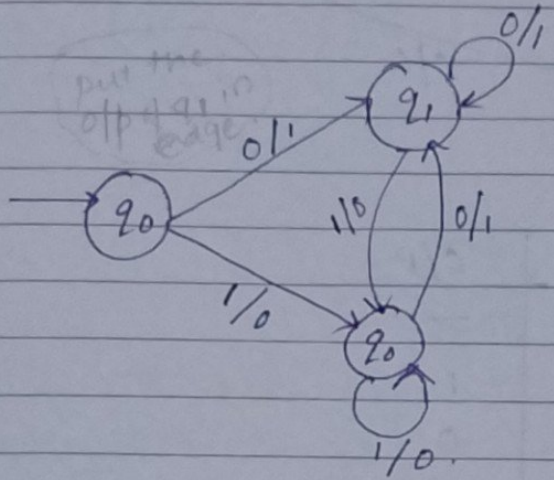
Just like DFA

5 times

5 yrs. [3 times in SEM]

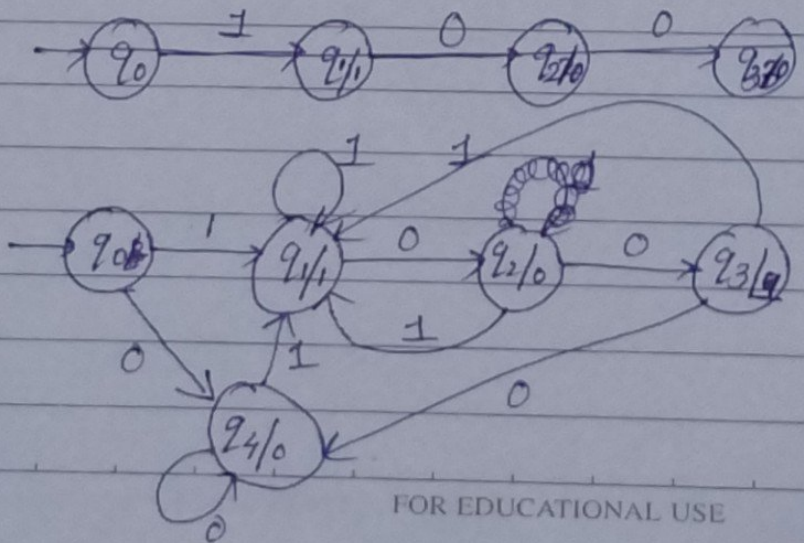


Convert Moore to Mealy machine



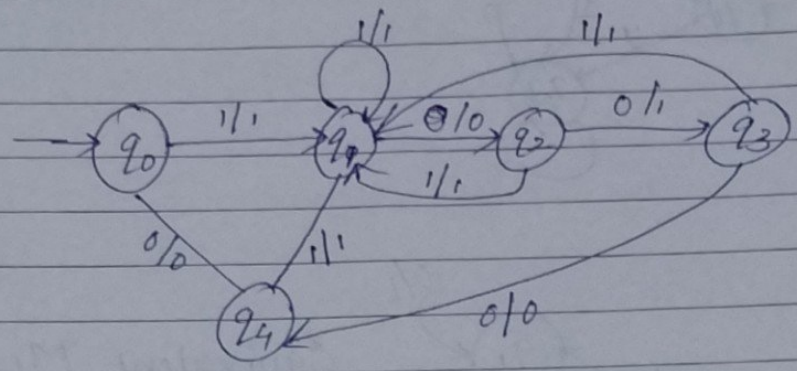
Equivalent Mealy Machine.

2) Design Moore and Mealy machine to convert each occurance of substring 100 by 101.



Mealy Machine.

1 anywhere
& 0

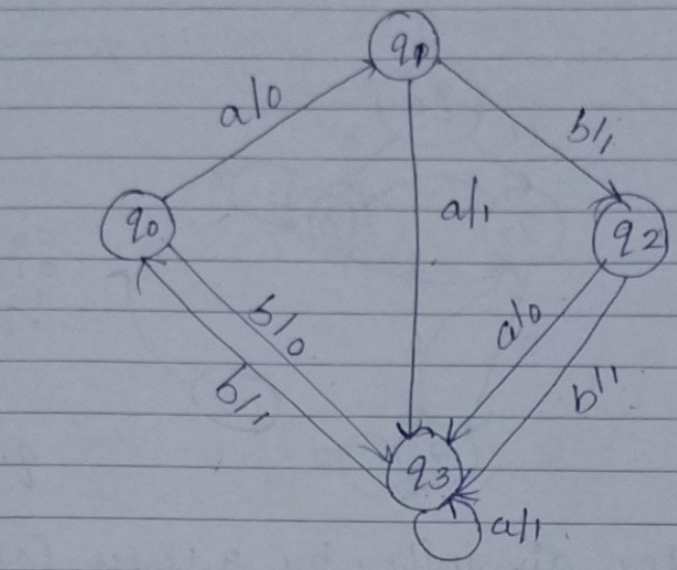


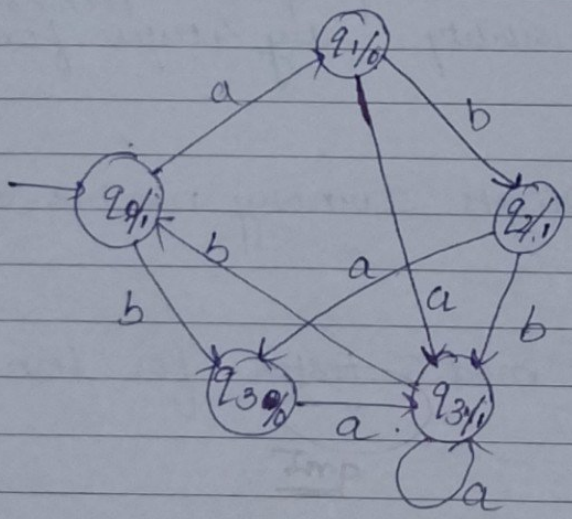
transition Table.

q	0	1	O/P
q_0	q_4	q_1	—
q_1	q_2	q_1	1
q_2	q_3	q_1	0
q_3	q_4	q_1	1
q_4	q_4	q_1	0

[See the o/p associated with it]

3. Convert the following Mealy machine to Equivalent Moore machine.





00
01 → 1.
10 → 2
11 → 3

10 0/1
100/101
4 5

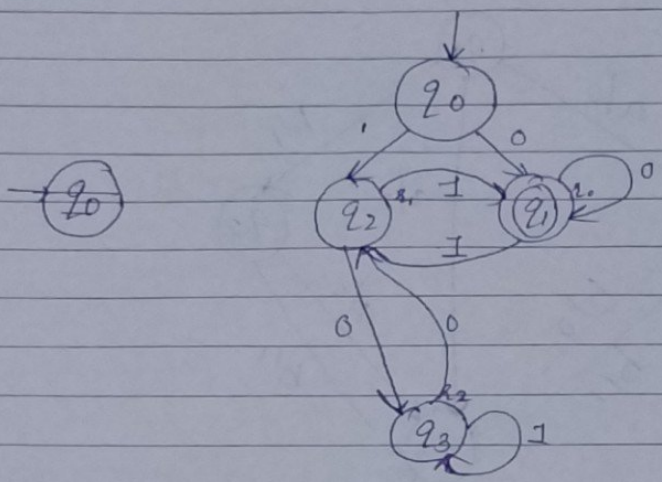
28/7/22

1 Design FSM for the binary number divisible by 3.

00/1 = 0×2+0 | 0×2+1.

101 0/1 → 5×2+0 | 5×2+1

10 / 11.

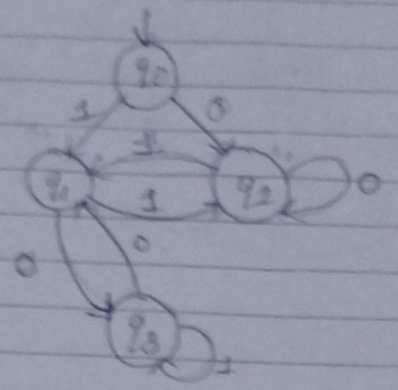


00 0×2+0=0
01 = 0×2+1=1
10 = 1×2+0=2
11 = 1×2+1=3
1
3

q_1 → Exactly divisible by 3.

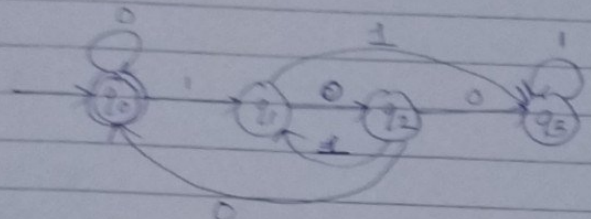
1. Design DFA for divisibility by 3 tester. [same as above]
2. Design DFA for divisibility by 4 tester for a binary number. (for binary number)
3. Design DFA that accepts ternary number divisibility by 4.
4. Design DFA for a mod 5 tester for ternary input.
5.

(ii)



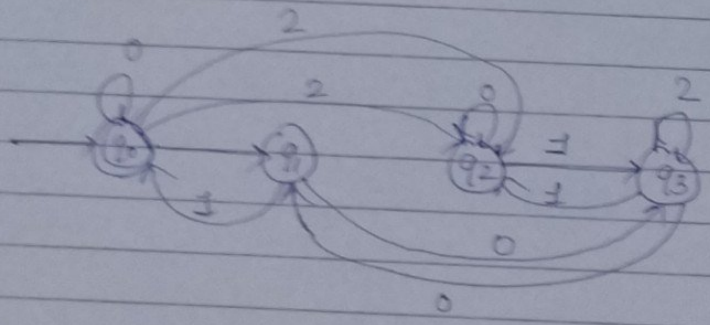
	0	1
q_0	q_1	q_2
q_1	q_3	q_2
q_2	q_2	q_1
q_3	q_1	q_3

(iii)



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

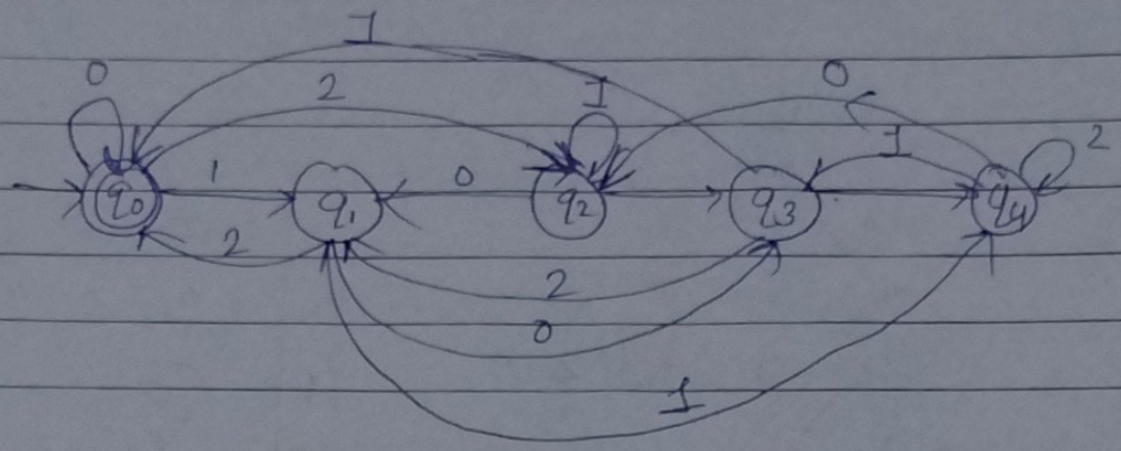
(iii)



	0	1	2
q_0	q_0	q_1	q_2
q_1	q_3	q_0	q_1
q_2	q_2	q_3	q_0
q_3	q_1	q_2	q_3

Regular Expression:-

4)



28/01/21

Regular Expression:-

It uses set of operators as follows:-

i) + ⇒ Union operator
• ⇒ concatenate operator
* ⇒ known as 'star' or closer operator.

*) Write regular expression for the following languages.

1) set of 1010.
RE = set of {1010}

2) set of 10, 1010.
RE = {10, 1010} = 10 + 1010

3) ∈, 10, 01
RE = {∈, 10, 01} = ∈ + 10 + 01

4) ∈, 0, 00, 000, 0000, ----
RE = 0^*
0^* = Empty string
0^+ = {∈, 0, 00, 000, 0000 ---}
0^+ = {0, 00, 000 ---}

5) 0, 00, 000 ---
RE = 0^+

6) Σ = {0, 1} strings starts with 0
RE = 0(0+1)*
{∈, 0, 1, 01, ---- }

$0^+ = 00^*$

$0^+ = 0\{∈, 0, 00, 000 ---$

7) $\Sigma = \{0,1\}$ string should end with 1.
RE = $(0+1)^*1$

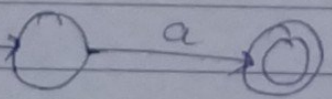
8) $\Sigma = \{a,b\}$ string starts with 'a' and ends with 'b'
RE = $a(a+b)^*b$

* Recursive Nature of the Regular Expression.

1. $R = (1+0(10)^*)^*$, R can be written as
$R = P^*$
where $P = 1+0(10)^*$
P can be written as.
$P = P_1 + P_2$
where $P_1 = 1$ and $P_2 = 0(10)^*$
P_2 can be written as.
$P_2 = P_3 \cdot P_4$
where $P_3 = 0$ & $P_4 = (10)^*$
P_4 can be written as
$P_4 = P_5^*$
where $P_5 = 10$
P_5 can be written as.
$P_5 = P_6 \cdot P_7$
where $P_6 = 1$ and $P_7 = 0$

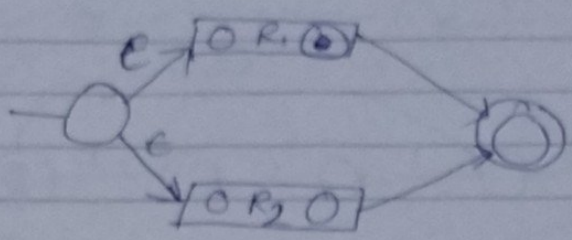
For Regular expression R_1 and R_2

$R_1 = a$

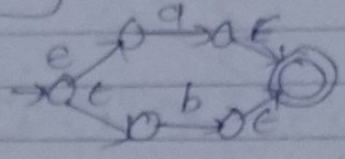


FA for
$R_1 + R_2$

i) $R_1 + R_2$

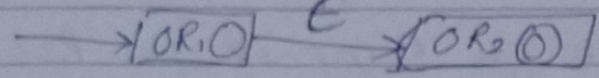


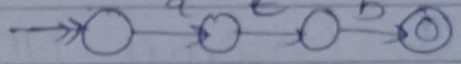
a+b



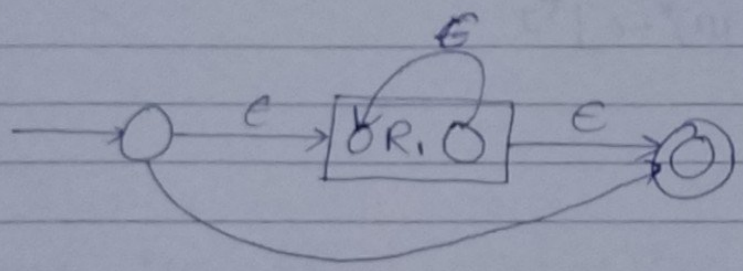
ii) $R_1 \cdot R_2$

a·b

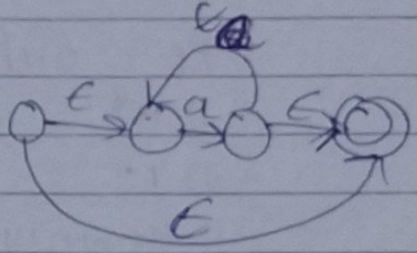




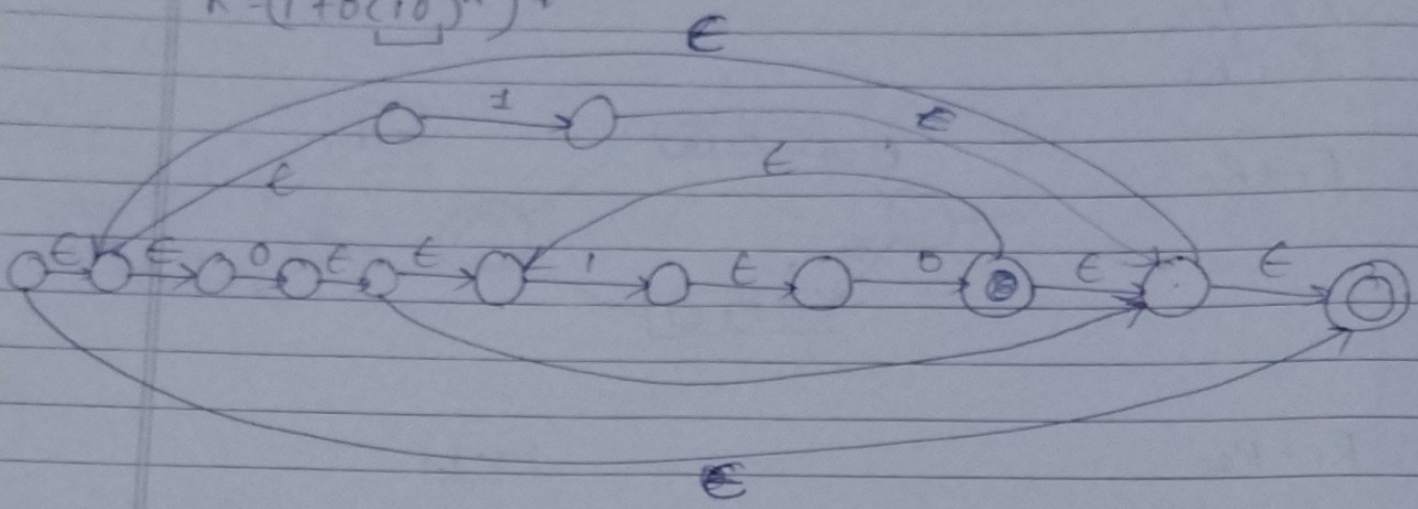
iii) R_1^*



a^*



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



① Construct NFA with ε-moves for RE

RE : $0 \cdot 1[(10^* + 111)^* + 0]^* 1$

i) 0^*

ii) 1

iii) Add with 111

iv) $(10^* + 111)^*$

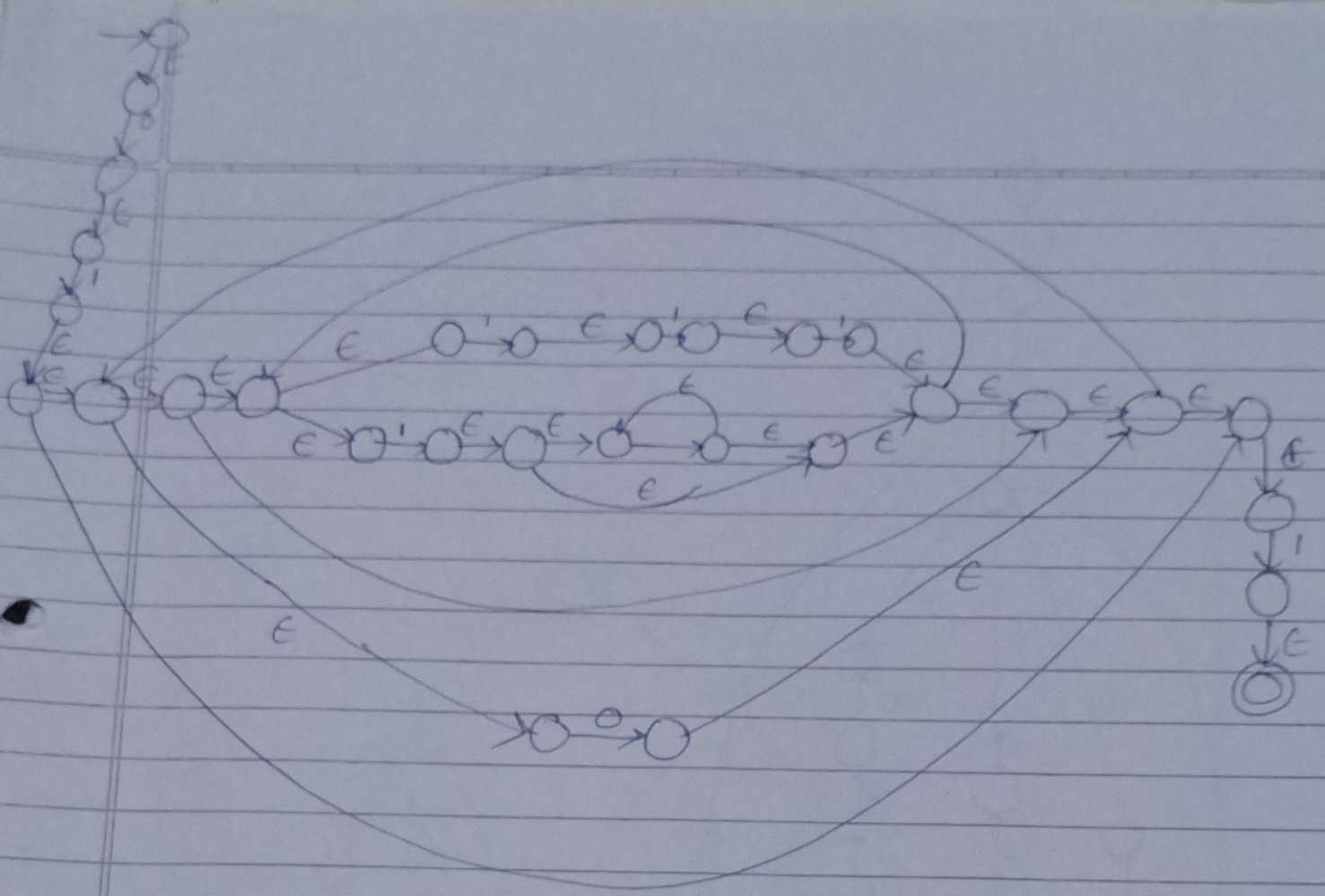
v) Add 0

vi) $[(10^* + 111)^* + 0]^*$

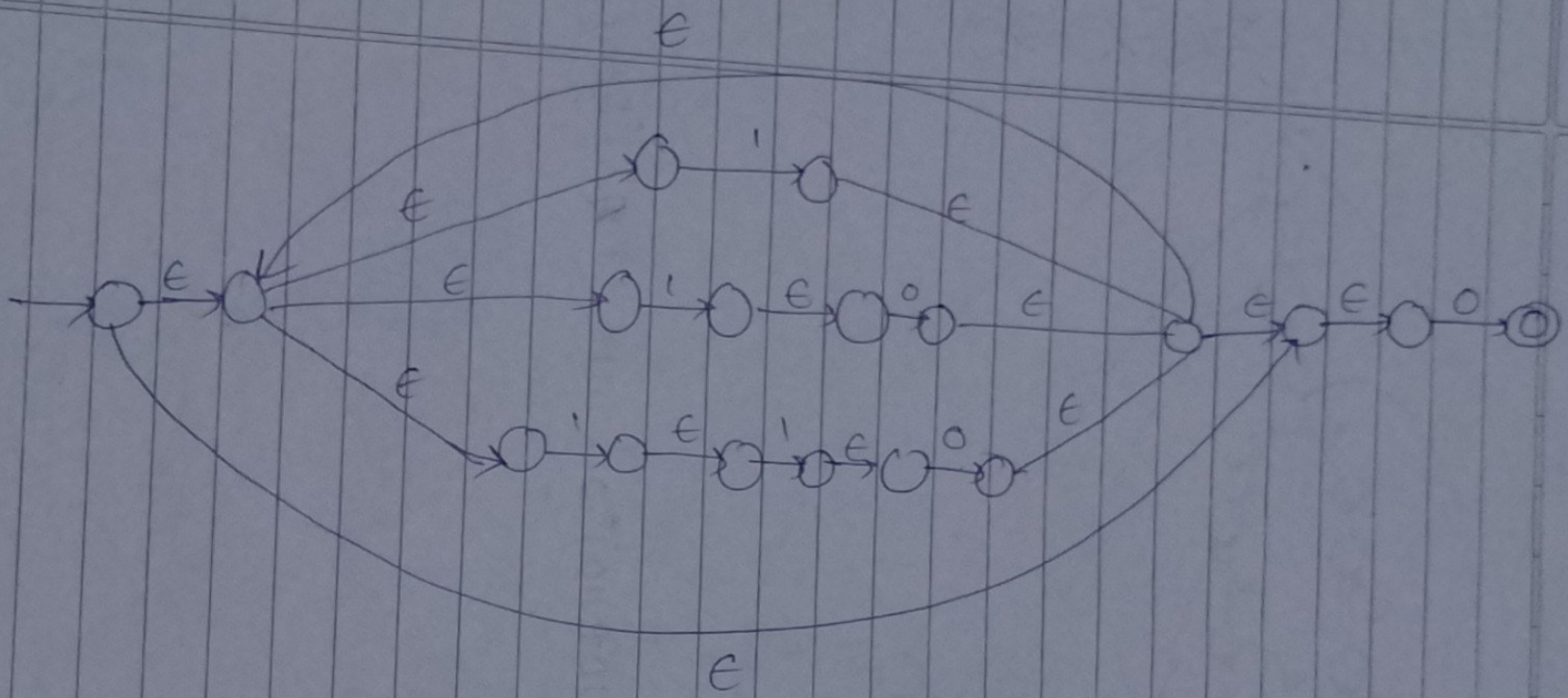
vii) Multiply $0 \cdot 1[(10^* + 111)^* + 0]$

viii) Whole *

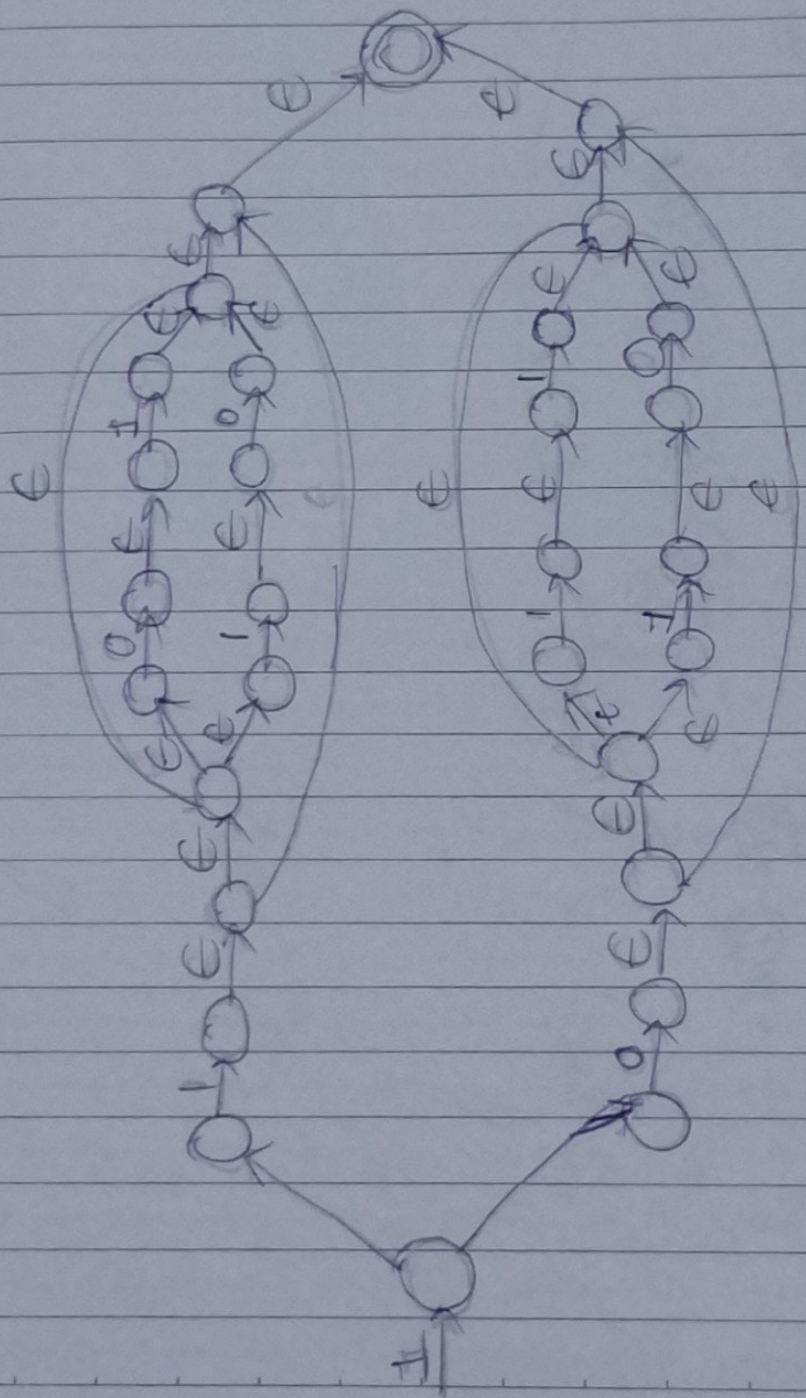
ix)



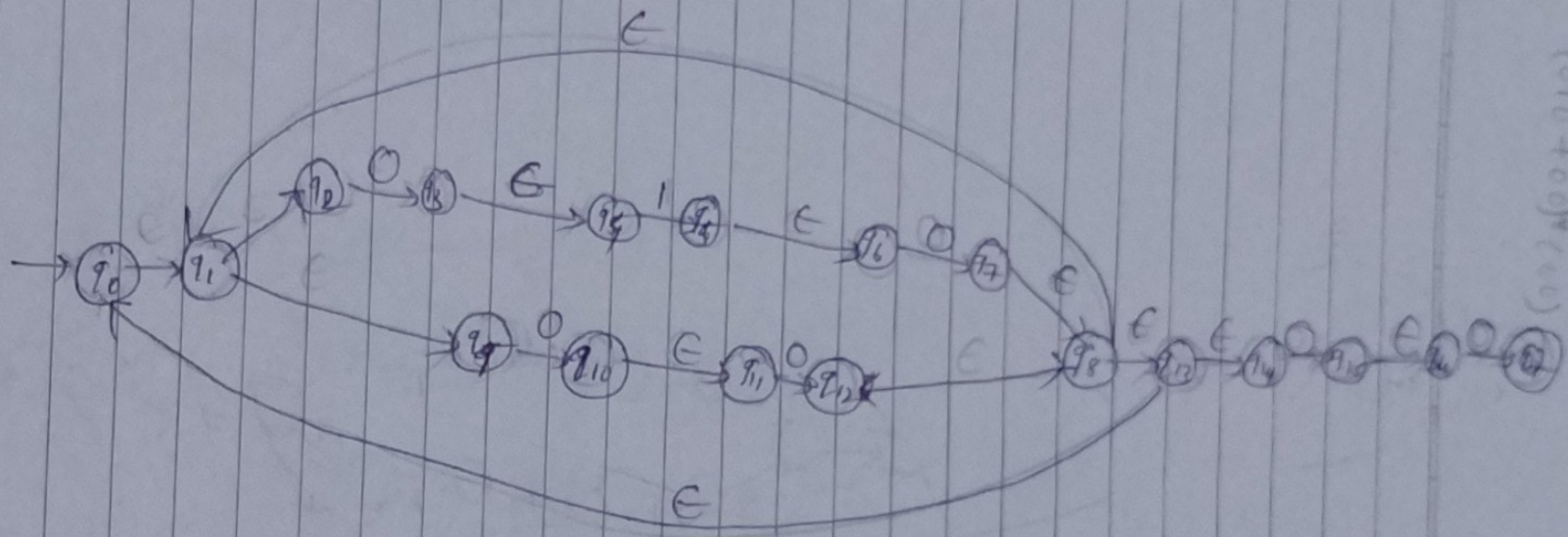
Convert the RE $(1+10+110)^*0$ into FA (DFA/NFA)


ε
ε
1
ε
ε
ε
1
ε
0
ε
ε
ε
0
ε
1
ε
1
ε
0
ε
ε

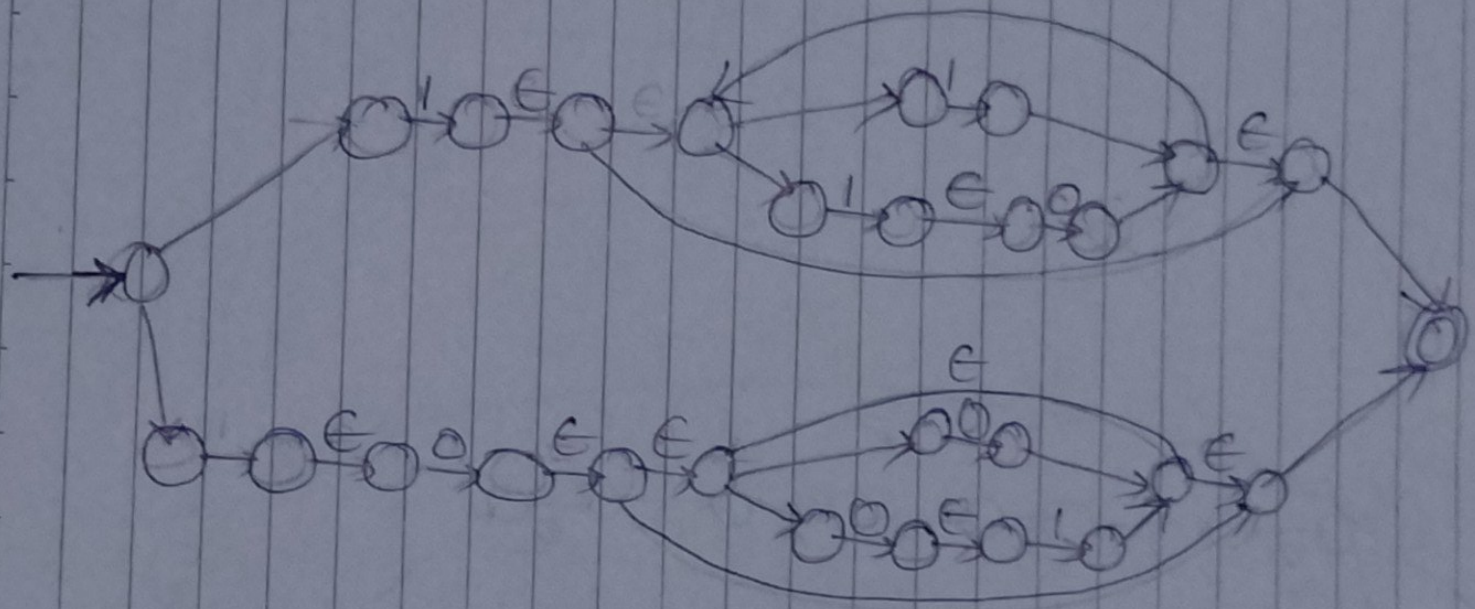
Hw $1(01+10)^* + 0(11+10)^*$
$(010+00)^* (00)$
$1(1+10)^* + 10(0+01)^*$



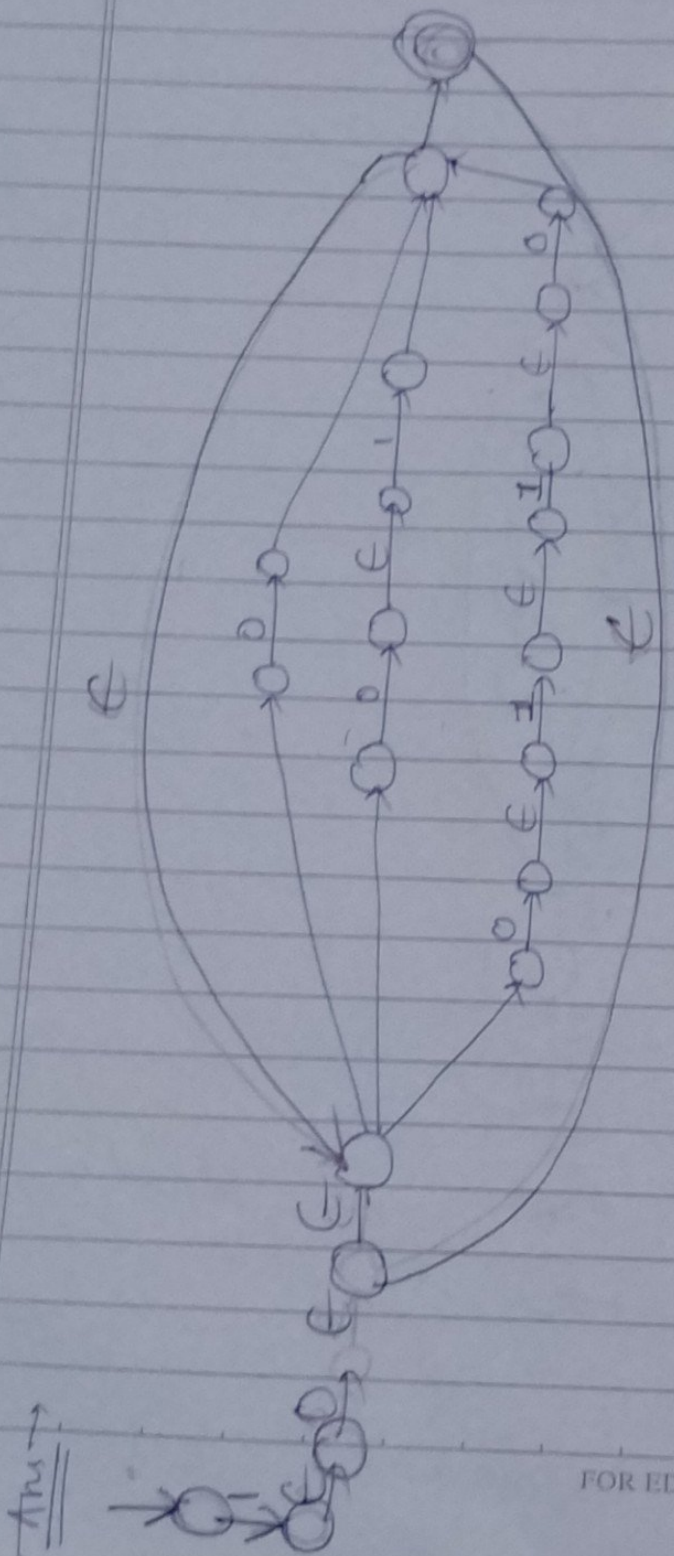
② $(01|0+00)^*(00)$



③ $1(1+10)^* + 10(0+01)$



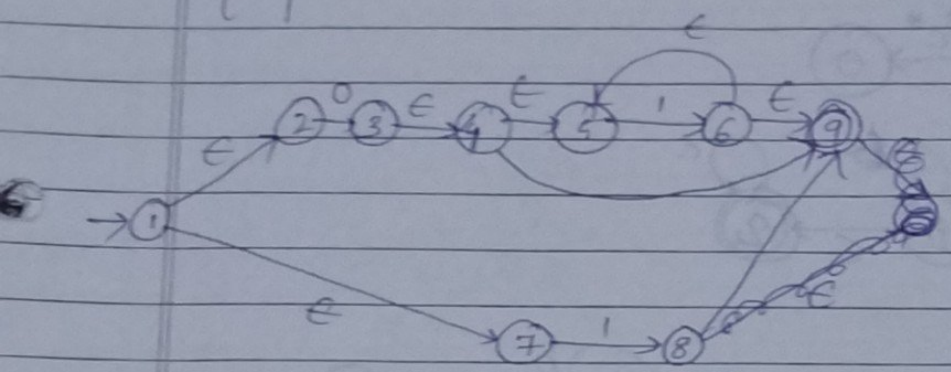
Convert NFA with ε more for R.E = 10(0+01+011)*



Convert RE into equivalent DFA = (01* + 1)

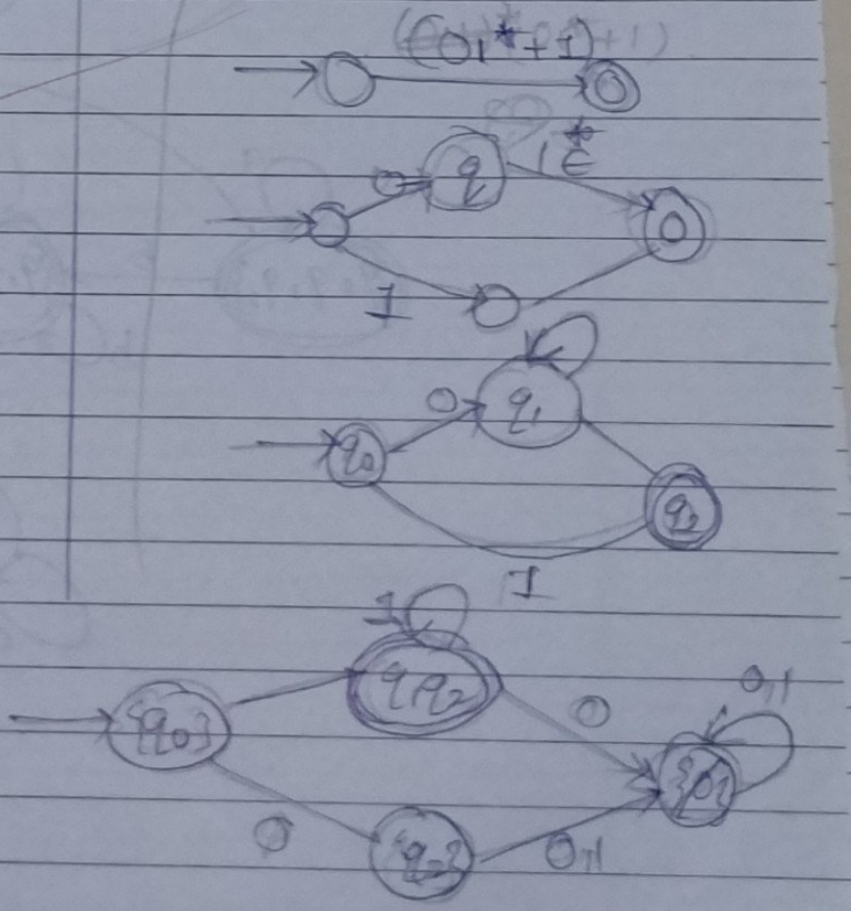
Ans RE to ε-dfa

Step 1. RE to ε-nfa



Step 2: ε-nfa to dfa by ε closer method.

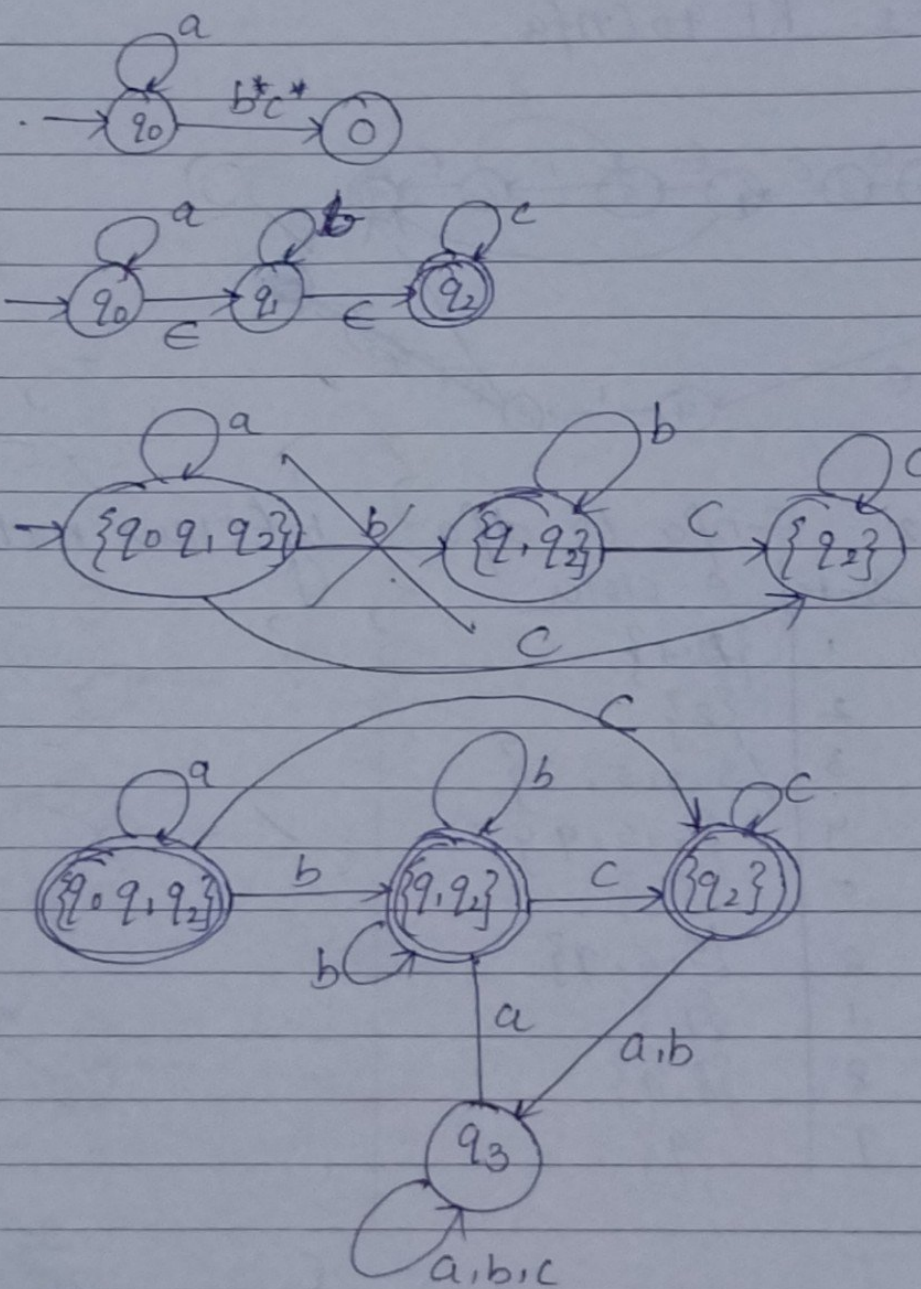
State	ε - closer
1	{1,7}
2	{2}
3	{3,4,5,9}
4	{4,5,9}
5	{5}
6	{5,6,9}
7	{7}
8	{8,9}
9	{9}



* means loop itself + ε

8/22

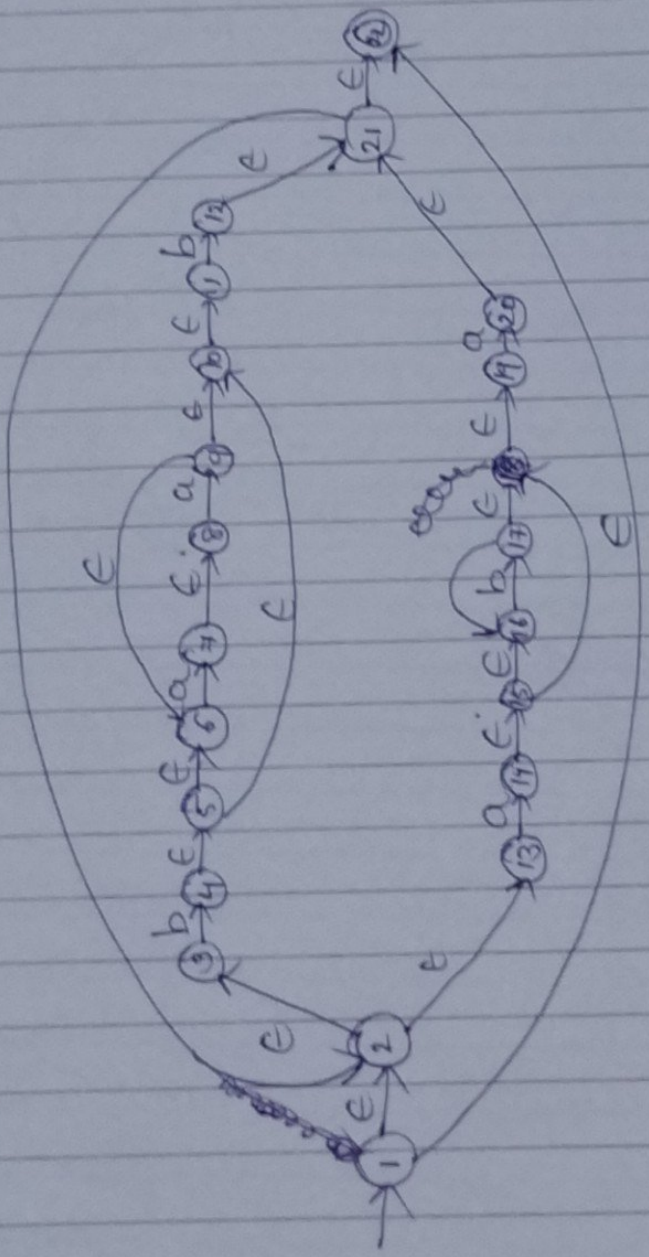
Construct NFA to accept a language $a^*b^*c^*$ Represented by $a^*b^*c^*$. Convert this NFA to equivalent DFA.

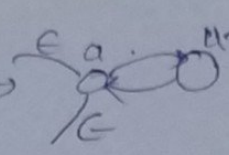


State	ϵ-closure
q_0	q_1, q_2, q_4, q_7
q_1	q_2, q_4

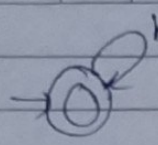
ii) $(b(aa)^* b + ab^* a)^*$

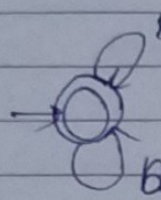
Convert it into ϵ moves. and then convert it to Equivalent DFA.

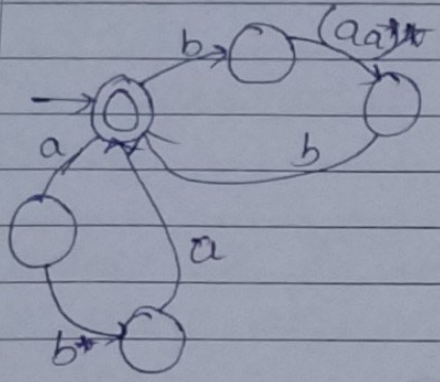


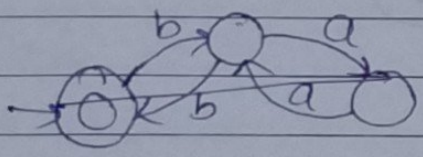
$(aa)^*$ →  Direct Method

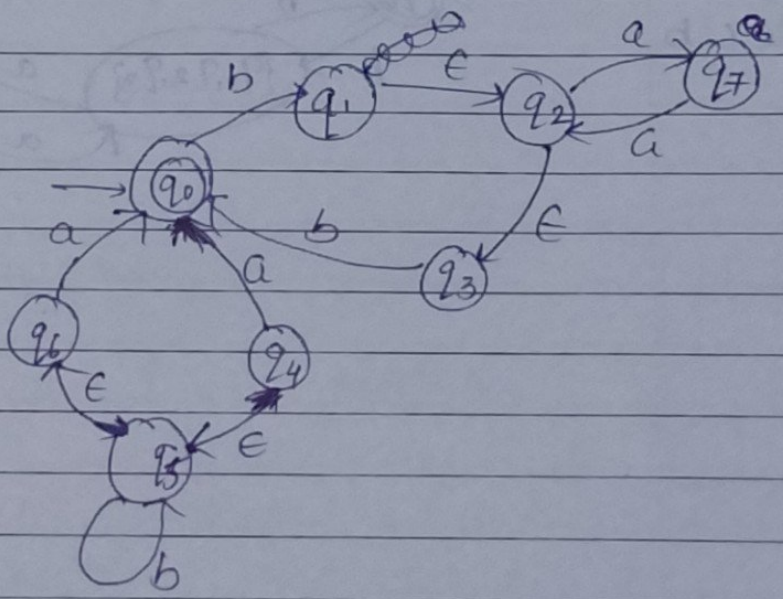
ε NFA to DFA











State	ε - closer.
q_0	q_0.
q_1	q_1, q_2, q_3
q_2	q_2, q_3.
q_3	q_3
q_4	q_4, q_5, q_6.
q_5	q_5, q_6.
q_6	q_7.

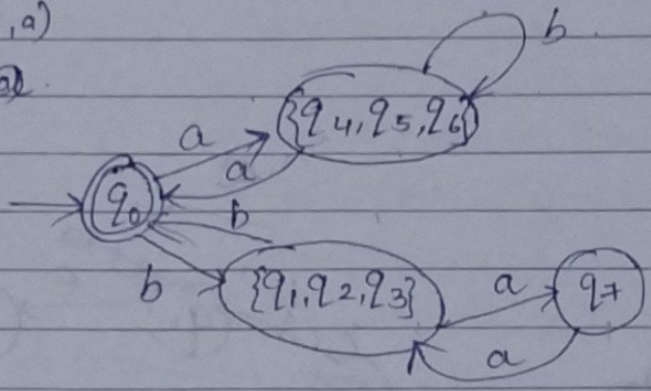
$\delta(q_4, a) \cup (q_5, a) \cup (q_6, a)$

$= q_6 \cup \epsilon \cup q_0(a)$.

$= q_6 a$.

δ

$\epsilon \cup \epsilon \cup b$.



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

5/8/22

DFA to RE

If $L = L(A)$ for some DFA 'A' then their is a regular expression R, then $L = L(R)$

Let Set of $A = \{1, 2, \cdots n\}$ for some integer N.

This always

Let us use R_{ij}^k has a name of RE whose language is set of w, such that 'w' is a label of a path from state 'i' path and the path as no intermediate node whose number is greater that k

Q.1 Construct RE R_{ij}^k ~~start~~ start at inductive defination and We start at inductive defination $k=0$ and finding reaching.

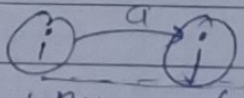
$k=0$ ---- $k=0$

Basis $k=0$

1) i to j

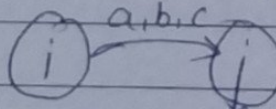
2) node i ie path of length 0.

If $i \neq j$ then only 'case I' is possible.



i) If their is no such symbol 'a' then $R_{ij} = \emptyset$

ii) If their is actual exactly 1 such symbol 'a' the $R_{ij} = a$

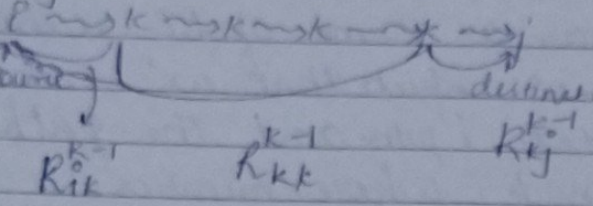


If their are more then 1 inputs then $R_{ij} = a_1 + a_2 + a_3 \cdots + a_k$

* Let thair is a path that goes through no. state higher than k.

Their are two possible cases in induction.

i) The path does not go through k state
R_{ij}^{k-1}

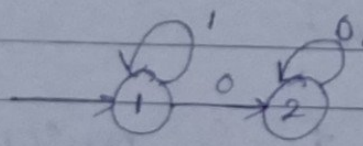
ii) 

Now we have to combine the expression of two types

∴ We get the express.

$$R_{ij}^{(k)} = R_{ij}^{k-1} + R_{ik}^{(k-1)} \cdot (R_{kk}^{k-1})^* \cdot R_{kj}^{k-1}$$

Convert the DFA to Equivalent Regular expression.



R_{11}^0	$\epsilon+1$
R_{12}^0	0
R_{21}^0	$\emptyset$
R_{22}^0	$(\epsilon+0+1)$

	By direct Substitution	Simplified
R'_{11}	$(\epsilon+1)+(\epsilon+1)^*(\epsilon+1)$	$(\epsilon+1)^* = 1^*$
R'_{12}	$0+(\epsilon+1)(\epsilon+1)^*0$	1^*0
R'_{21}	$\phi+\phi(\epsilon+1)^*(\epsilon+1)$	ϕ
R'_{22}	$(\epsilon+0+1)+\phi$	$\epsilon+0+1$

~~$R'_{ij} = R^0_{ij} + R^0_{ik}(R^0_{kk})^* R^0_{kj}$~~

$R'_{ij} = R^0_{ij} + (R^0_{ii})^* R^0_{ij}$

$R'_{11} = R^0_{11} + (R^0_{11})^* R^0_{11}$

$R'_{12} = R^0_{12} + (R^0_{12})^* R^0_{12}$

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

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

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

$0(\epsilon+1)^*$

$R'_{21} = R^0_{21} + R^0_{21}(R^0_{11})^* R^0_{12}$

$\phi = \phi + \phi(\epsilon+1)^*(\epsilon+1)$

$R'_{22} = R^0_{22} + R^0_{21}(R^0_{11})^* R^0_{12}$

$= (\epsilon+0+1) + \phi \cdot (\epsilon+1)^* \cdot 0$

R^2_{11}	$1^* + 1^*0(\epsilon+0+1)^* \cdot \phi$	1^*
R^2_{12}	$1^*0 + 1^*0(\epsilon+0+1)^*(\epsilon+0+1)$	$1^*0(0+1)^*$
R^2_{21}		
R^2_{22}		

$R^2_{12} = R^1_{12} + R'_{12}(R'_{22})^* R'_{22} = 1^*0 + 1^*0(\epsilon+0+1)^*(\epsilon+0+1)$

$R^2_{11} = R'_{11} + R'_{12}(R'_{22})^* R'_{21} = 1^* \quad 1^*0(\epsilon+(\epsilon+0+1)^*)$

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

2 times IN SEM

Arden's Theorem

In order to find out a regular expression of a finite automaton we use Arden's Theorem along with the properties of regular expression.

- Statement

Let P and Q be two regular expression.

If P does not contain null string then $R = Q + RP$ has a unique solution that is $R = QP^*$

- Proof -

$R = Q + (Q + RP)P$ [After putting the value $R = Q + RP$]

$= Q + QP + RPP$

when we put the value of R recursively again and again, we get the following equation -

$R = Q + QP + QP^2 + QP^3 \ldots$

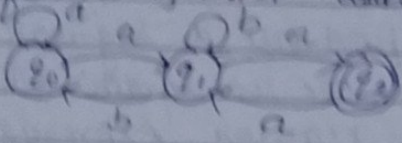
$= Q(\epsilon + P + P^2 + P^3 + \ldots)$

$R = QP^*$ [As P^* represents $(\epsilon + P + P^2 + P^3 + \ldots)$

- Hence proved.

- An another method for converting any FA into to

Convert following NFA using to Eq. RE using Arden's Thm.



$q_2 = q_1 \cdot a$ — (1)

$q_1 = q_0 \cdot a + q_1 \cdot b + q_2 \cdot a$ — (2)

$q_0 = q_0 \cdot a + q_1 \cdot b + \epsilon$ — (3)

Solving Equations by Ardenin's Theorem.

$q_1 = q_0 \cdot a + q_1 b + q_1 \cdot a \cdot a$

$q_1 = q_0 \cdot a + q_1 (b + aa)$

This Equ^n of the form $R = Q + RP$
where $Q = q_0 a$, $R = q_1$; $P = (b + aa)$

Using Ardem's Thm.
It's solution is given by $R = QP^*$
Thus $q_1 = q_0 \cdot a (b + aa)^*$ — (4)

Substitute q_1 in 3rd Equation.

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

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

This Equ^n of the form $R = Q + RP$
where

$Q = q_0 a \epsilon$, $R = q_0$, $P = a + a(b + aa)^* b$

Using Arden's Theorem
It's solution is given by

$R = QP^*$

$q_0 = q_0 a$

$q_0 = \epsilon \cdot (a+a(b+aa)^*b)^*$ — (5)

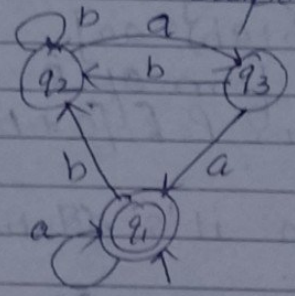
Put 5th Equ in 4th

$q_1 = \epsilon (a+a(b+aa)^*b)^* \cdot a(b+aa)^*$ — (6)

∴ Put 6 in (1) Equ

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

Construct RE corresponding



Here initial and final state is q_1

The Equ for the 3 states for q_1, q_2, q_3 are as follows.

$q_1 = q_1 a + q_3 a + \epsilon$ — (1)

$q_2 = q_1 b + q_2 b + q_3 b$ — (2)

$q_3 = q_2 a$ — (3)

$q_2 = q_1b + q_2b + q_3b$

$= q_1b + q_2b + (q_2 \cdot a)b$

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

This eqn is in the form of $R = Q + RP$

where $Q = q_1b$, $R = q_2$, $P = (b + ab)$

Using Arden's Theorem it's solution is given as.

$R = QP^*$

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

$q_1 = q_1a + q_3a + \epsilon$

$= q_1a + (q_2a)a + \epsilon$

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

$q_1 = q_1[a + b(b + ab)^* aa] + \epsilon$

$q_1 = \epsilon + q_1[a + b(b + ab)^* aa]$

This eqn is in the form of $R = Q + RP$

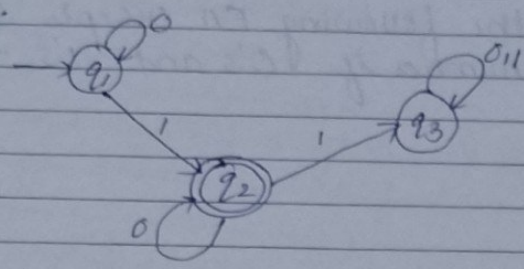
where $Q = \epsilon$, $R = q_1$, $P = [a + b(b + ab)^* aa]$

Using Arden's Theorm it's solution is given by as $R = QP^*$

$q_1 = [a + b(b + ab)^* aa]^*$

As q_1 is the final state in the given NFA the Equivalent regular expression is $[a + b(b + ab)^* aa)^*$

(2)



Here the initial state is q_1 and final state is q_2. following are the equivalent regular expression

$$q_1 = q_1 0 + \epsilon \quad - (1)$$
$$q_2 = q_1 1 + q_2 0 \quad - (2)$$
$$q_3 = q_2 1 + q_3 0 + q_3 1 \quad - (3)$$

Since 1st Equation is already in Ardem's Thm.

$$q_1 = \epsilon \cdot 0^* \Rightarrow 0^*$$
$$q_1 = 0^* \quad - (4)$$

put 4 in (2) Eqn.

$$q_2 = 0^* \cdot 1 + q_2 0$$

This Eqn is in the form of $R = Q + RP$

$R = q_2$, $Q = 0^*1$, $P = 0$

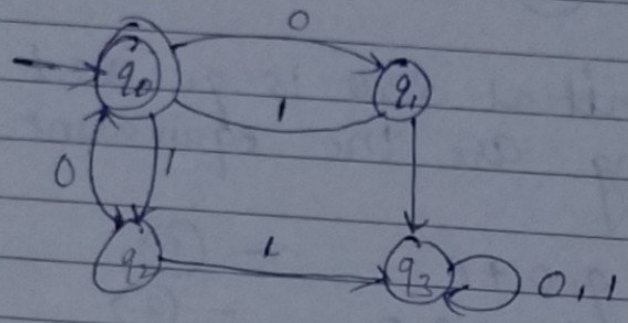
Using Ardem's Thm it's soln is given.

$R = QP^*$

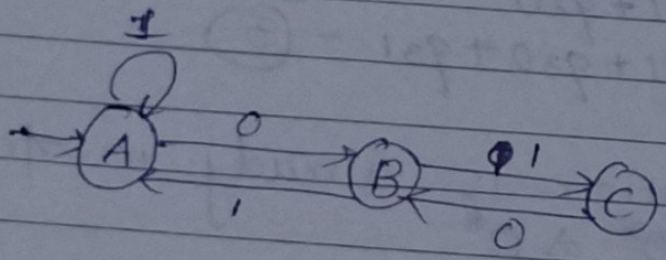
$q_2 = 0^*1(0)^*$

As q_2 is the final state.

1) Prove that the following FA accepts strings with equal number of 0's and 1's such has each.



2)



1 Ans The state q_3 is a dead. It is not required to equate for state q_3.

Set of equation for regular sets represented by the state q_0, q_1 and q_2 are given by:

$$q_0 = \varepsilon + q_1 \cdot 1 + q_2 \cdot 0$$
$$q_1 = q_0 \cdot 0$$
$$q_2 = q_0 1$$

∴ put q_1 & q_2 in q_0.

$$\therefore q_0 = \varepsilon + q_0 \cdot 0 \cdot 1 + q_0 \cdot 1 \cdot 0$$
$$q_0 = \varepsilon + q_0 (0 \cdot 1 + 1 \cdot 0)$$

It is of the form $R = Q + RQP$

∴ Where $Q = \varepsilon$, $R = q_0$, $P = 0 \cdot 1 + 1 \cdot 0$

By using Arden's theorem.

∴ It's solution of given $R = QP^*$

$$\therefore q_0 = \varepsilon (0 \cdot 1 + 10)^*$$
$$q_0 = (0 \cdot 1 + 10)^*$$

2 Ans The set of equations of regular sets represents by the states A, B, C are given as.

$A = \varepsilon + A1 + B1$

$B = A0 + C0$

$C = B1$

Rewriting A Equation.

$\therefore A = (\varepsilon + B1)1 + A1$

$\therefore$ It is of the form $R = Q + RP$

$R = A, Q = \varepsilon + B1$ & $P = 1$

By using Arden's Equation.

$\therefore$ It solution is $R = QP^*$

$\therefore A = (\varepsilon + B1)1^*$

$\therefore B = (\varepsilon + B1)1^*0 + B10$ $\therefore$ {Substituting A Eq^n & C Eq^n}

$= 1^*0 + B11^*0 + B10$

$= 1^*0 + B(11^*0 + 10)$ $\therefore (R = Q + RP)$ $Q = 1^*0$, $R = B$, $P = (11^*0 + 10)$

$\therefore$ By Arden's theorem. $\therefore$ It's solution $R = QP^*$

$\therefore B = 1^*0(11^*0 + 10)$

$\therefore$ Put B in C

$C = 1^*0(11^*0 + 10)^*1$

State C is a final state, the regular expression represented by C

$\therefore RE = 1^*0(11^*0 + 10)^*1$

8mks Closure Properties of Regular Expression.
Explain Decision properties of Regular expression.
Application of RE

* Properties for R.E

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

1 $\phi + R = R$.

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

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

4 $\epsilon^* = \epsilon$

5 $\phi^* = \epsilon$

6 $R + R = R$

7 $PQ + PR = P(Q+R)$

8 $QP + RP = (Q+R)P$

9 $R^*R^* = R^*$

10 $RR^* = R^*R$

11 $(R^*)^* = R^*$

12 $\epsilon + RR^* = RR^* = R^*$

13 $(PQ)^*P = P(QP)^*$

14 $(P+Q)^* = (P^*Q^*)^* = (P^*+Q^*)^*$

15 $(P^*Q^*) = \epsilon + (P+Q)^*Q$.

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

$= R^*$

UT 1: 1) FA → DFA → NFA → ε-closure. 2) Moore, Mealy 3) Minimization 4) RE → Proves. conversion. from RE - ε-NFA - DFA ε-NFA / DFA → RE Arden's with. & closure properties. 5) Context free Grammar

Prove that:

1. $\epsilon + 1^*(011)^*(1^*(011)^*)^* = (1+011)^*$

LHS $= \epsilon + RR^*$ where $R = 1^*(011)^*$

$= \epsilon + R^*$ $\quad a \cdot a^* = a^*$

$= R^*$

$= (1^*(011)^*)^*$

$= (1^* + (011)^*)^* \because (P^*Q^*)^*$ by 14th property

2. $(1+00^*1) + (1+00^*1)(0+10^*1)^* \cdot (0+10^*1) = 0^*1(0+10^*1)^*$

~~LHS = (0+00^*1)~~

LHS $= (1+00^*1)[\epsilon + (0+10^*1)^*(0+10^*1)]$

AB^* $= (1+00^*1)[0+10^*1)^*$ $\quad RR^* = R^*$

$= (1[\epsilon + 00^*])(0+10^*1)^*$

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

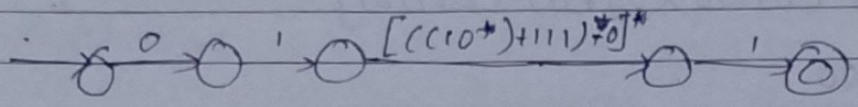
Regular Expression to

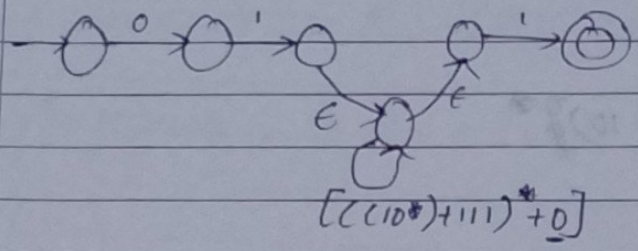
6) Construct finite Automata for the following Regular expressions.

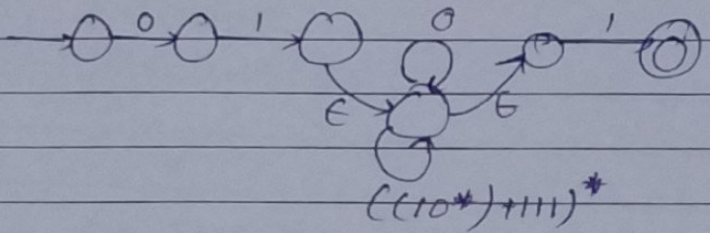
1) $10 + (0+11)0^*1$

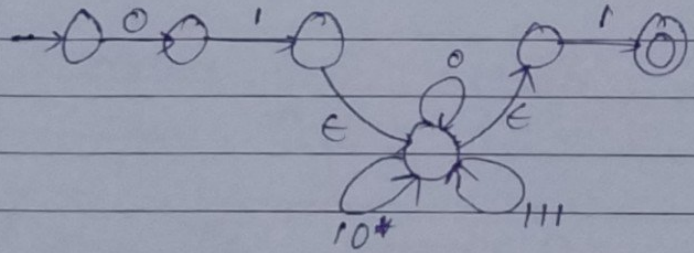
2) $01[((10^*)+111)^* + 0]^*1$

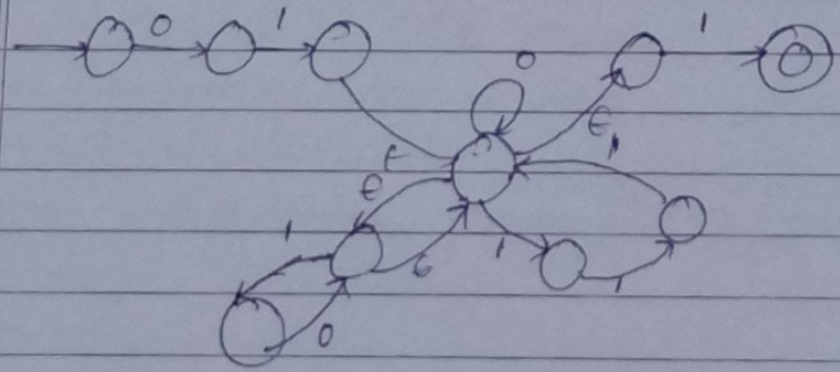
'*' connect ϵ to ϵ
'+' two diff. loops.
'.' → Seriers.

i) 

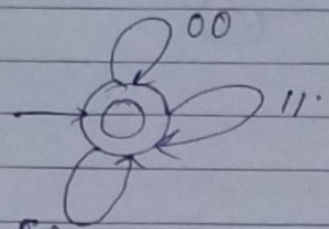
ii) 

iii) 

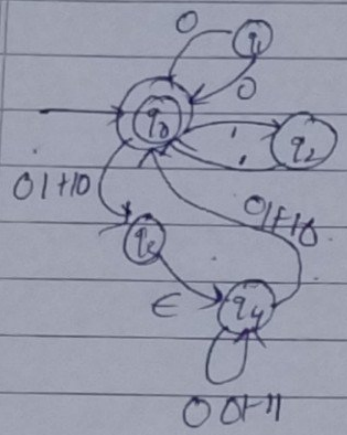
iv) 

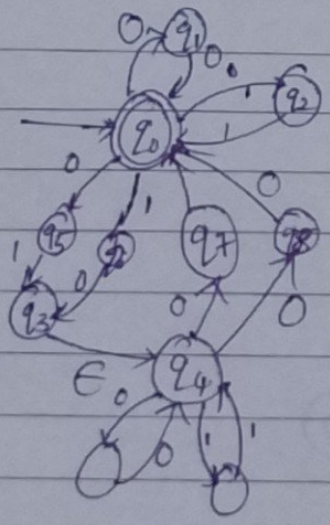
v) 

3) $[00+11+(01+10)(00+11)^*(01+10)]^*$



$[(01+10)(00+11)^*(01+10)]^*$





It can be asked → "don't forget FA, RE Application"

RE
RL FA

18/8/22

Context free language.

CFL.

A grammar G(V, T, P, S) is said to be context free grammer if Rules in production set P are in the form $A \to \alpha$, $A \in V$, $\alpha \in (V \cup T)^*$

Context free Gramer.

Variables → Captical letter.

A language L is said to be CFL if there. (A-Z).

terminators → Small letters.

$A \to Aa$

It derives Aa.

Notation.

Production.

Variable → Terminates.

Grammar: ① $S \to aSb$ (Production no. 1)

② $S \to \lambda$. (production no. 2)

Derive the sentence 'ab' from above grammar.

$S \Rightarrow aSb$ ~~$\Rightarrow ab$~~.

By using Production 1 we get $S \to aSb$.

$S \Rightarrow ab$.

By using production 2 we get. $S \to \lambda$.

Grammar: $S \to aSb$ (Production 1).
$S \to \lambda$. (Production 2)

Derive aabb.

$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaabbb$

$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaaaSbbbb$
$\Rightarrow aaaaabbbbb$

* Language of the Grammar

$S \to aSb$
$S \to \lambda$.
$L = \{a^n b^n : n \geq 0\}$

More No-

More Notations:

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

V: set of variables.

T: set of terminal symbols. { Small char a,b,c, 0,9,(,) }

S: Start variable.

P: Set of production rules.

Gammer G:

Sentential form:

A Sentence that contains variables & terminal

$S \rightarrow aSb \rightarrow aaSbb \rightarrow aaaSbbb \rightarrow aaaabbbb.$

Sentential form.

* → these cane written in sentential form.

we write $\cdot S^*$

Grammar G : S → Ab
A → aAb
A → λ

- Always start with 'S' variable.

Derivation.

S ⇒ Ab ⇒ b.
S ⇒ Ab ⇒ aAbb ⇒ abb
S ⇒ Ab ⇒ aAbb ⇒ aaAbbb ⇒ aabbb

language : $a^n b^{n+1}$

for Grammar G : S → Ab
A → aAb
A → λ.

$L(G) = \{a^n b^n b : n \geq 0\}$.
$S^* = a^n b^n b$

A Convenient Notation.

A → aAb. ⇒ A → aAb | λ.
A → λ

⟨article⟩ → a
⟨artide⟩ → the ⇒ ⟨article⟩ → a | the.

LC

$S \to aSa$.
$S \to bSb$
$S \to \lambda$.

$L(G) = ww^R$

CFG G: $S \to aSb$
$S \to SS$
$S \to \lambda$.

$S \to SS \Rightarrow aSbS \Rightarrow \cdots abab$.

$L(G) = \{w : n_a(w) = n_b(w), n_a(v) \geq n_b(v)$ in any prefix $v\}$.

Imp * Derivation Order.

1. $S \to AB$ 2. $A \to aaA$ 4. $B \to Bb$
3. $A \to \lambda$ 5. $B \to \lambda$.

Leftmost derivation.

$S \overset{1}{\Rightarrow} AB \overset{2}{\Rightarrow} aaAB \overset{3}{\Rightarrow} aaB \overset{4}{\Rightarrow} aaBb \overset{5}{\Rightarrow} aab$.

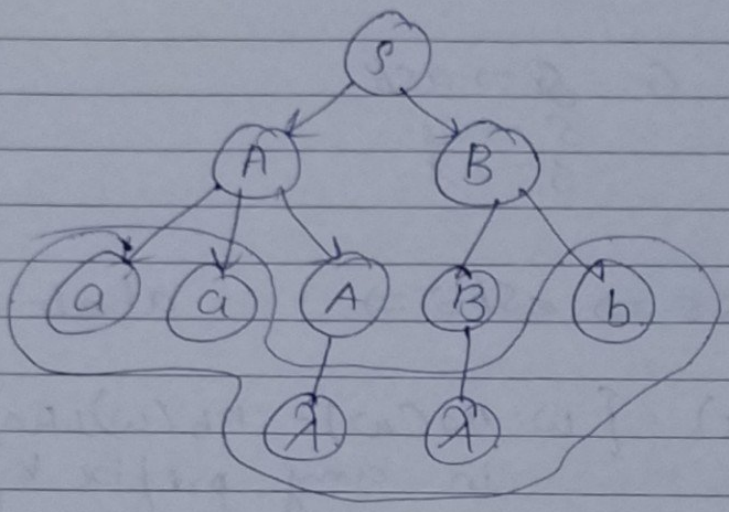
Right most derivation.

$S \overset{1}{\Rightarrow} AB \overset{4}{\Rightarrow} ABb \overset{3}{\Rightarrow} Ab \overset{2}{\Rightarrow} aaAb \overset{3}{\Rightarrow} aab$.

Derivation tree.

$S \to AB \qquad A \to aaA|\lambda \qquad B \to Bb|\lambda$.

$S \Rightarrow AB \Rightarrow aaAB$.



Yield.
aaλλb.
= aab.

$S \Rightarrow AB \Rightarrow aaAB \Rightarrow aaB \Rightarrow aaBb \Rightarrow aab$.

* → Can be empty. + means → ∈ (Not allowed $S \to \lambda$ (∈ is excepted)) restriction only single vari. at left.

V. Imp terms with Egs in M

Chomsky Hierarchy

Noam Chomsky ~~Hierarchy~~ introduced the classification scheme for

Type 0

A grammar $G=(V,T,SP)$ is said to be unrestricted or type 0 if all production are the form.

$u \to v$, where $u \in (V \cup T)^+$ & $V \in (V \cup T)^*$

left → (Any combination) → right

Definition.

A grammar $G=(V,T,S,P)$ is said to be context sensitive or type 1 if all productions are of the form.

$x \to y$, where $x, y \in (V \cup T)^*$ and $|x| \leq |y|$ (length)

Definition:

A grammar $G=(V,T,S,P)$ is said to be context free gramer. or type 2. If all production are of the form.

$A \to B$, where $A \in V$, $B \in (V \cup T)^*$

must start with terminal

var → a

↑ var.

There ~~must~~ be will be only an only variable.

Any combination of var. + ~~&~~ terminal ∈

Definition:

A grammar $G=(V,T,S,P)$ is said to be regular grammar. or type 3. If its production is

↑ var →

$A \to xB$ (or) $A \to x$, where $A \in V$, $B \in (V \cup \lambda)$, $x \in \Sigma^*$

Given a
for the Grammar Given below.

S → A | B S → A | B
A → OA/E , B → OB | 1B | E

Solve the above Grammar. find out the leftmost derivation & Right most derivation of the String 1001 also give parse tree for leftmost and rightmost derivations.

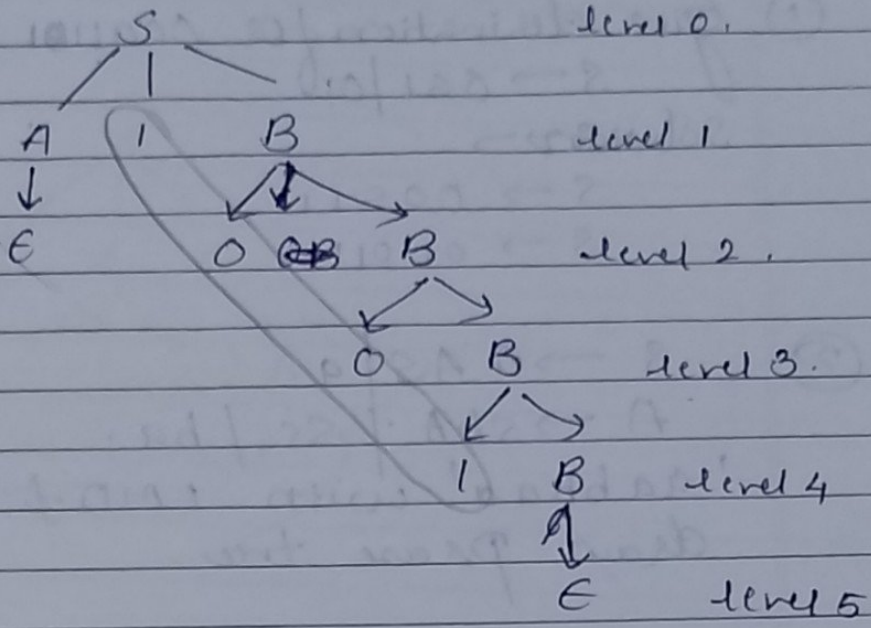
LMD :-

S → A | B
→ 1B (using A → E)
→ 10B (using B → 0B)
→ 100B (using B → 0B)
→ 1001B (using B → 1B)
→ 1001 (using B → E)

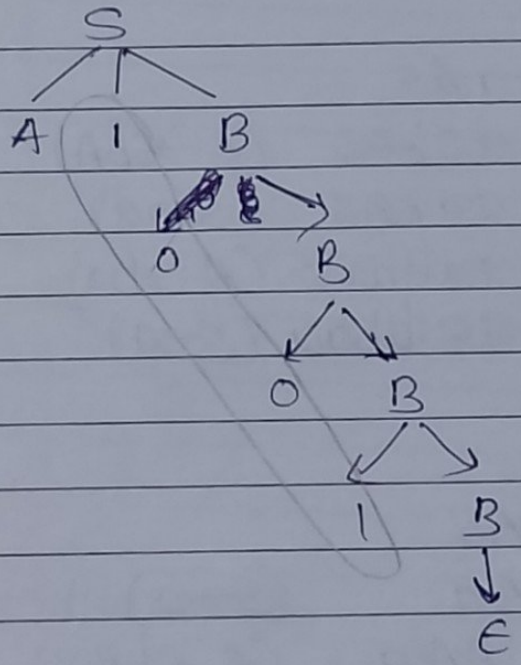
RMD :-

S → A | B
S → A10B (using B → 0B)
S → A100B (using B → 0B)
S → A1001B (using B → 1B)
S → A1001 (using B → E)
S → 1001 (using A → E)

LMD:



RMD:



① Give derivation for 0001101
$S \rightarrow 0S1 | 01$

Solution →

$S \rightarrow 00S11$

$S \rightarrow 000111$

② $S \rightarrow aAS | a$.

$A \rightarrow SbA | SS | ba$.

'aabbaa' with LMD & RMD draw parse tree.

LMD :-

$S \rightarrow aAS$

$S \rightarrow aSbAS$ $(A \rightarrow SbA)$

$S \rightarrow aabAS$ $(S \rightarrow a)$

$S \rightarrow aabbaSS$ $(A \rightarrow ba)$

$S \rightarrow aabbaa$ $(S \rightarrow a)$

RMD

$S \rightarrow aAS$

$S \rightarrow aAa$ $(S \rightarrow a)$

$S \rightarrow aSbAa$ $(A \rightarrow SbA)$

$S \rightarrow aSbbaa$ $(A \rightarrow ba)$

$S \rightarrow aabbaa$ $(S \rightarrow a)$

LMDPT

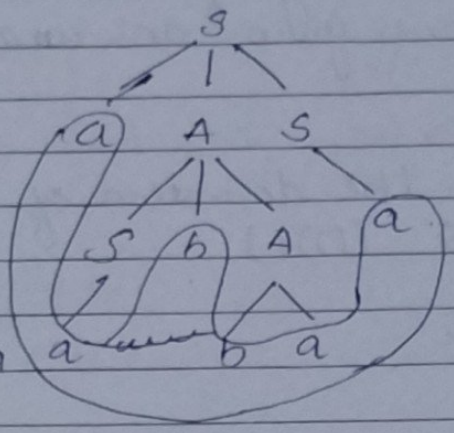
~~S → aAS~~

~~S → aAS~~

S → aAS

~~A~~ → SbA

S → a ; A → ba

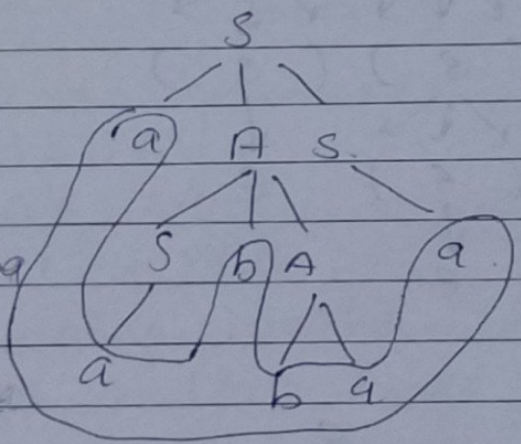


RMDPT

~~S → aAS~~

~~S → SbA~~ , S → a

S → a ; A → ba



1) $E \rightarrow E+T \mid T$
2) $T \rightarrow T*F \mid F$
3) $F \rightarrow (E) \mid a \mid b$.

find out sentential form & parse Tree.

Ambiguous & Unambiguous Grammar.

→ Two different parse trees for the same Grammar is called Ambiguous Grammar.

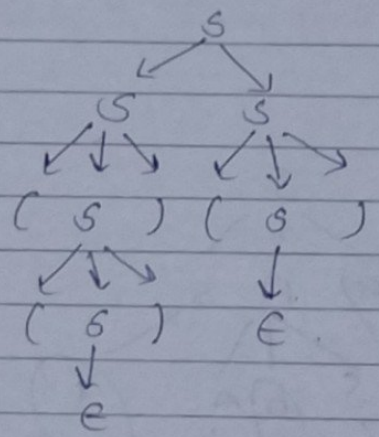
Q. Show that the given cfg which generates all strings of balanced parentesis is ambigous. Given an unambigous Grammar.

Let us Consider the derivation of string.

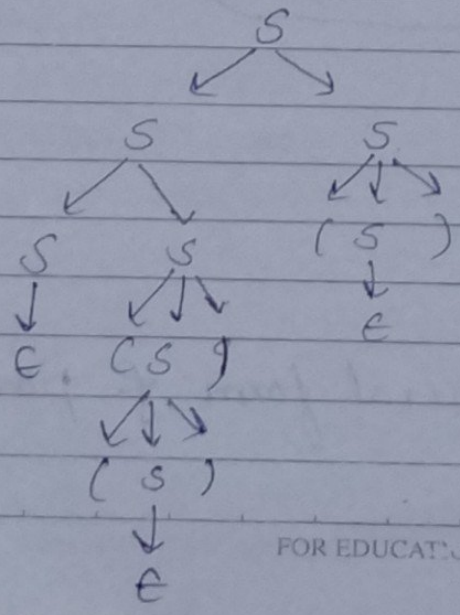
~~S → SS | (S) | ∈~~ (())()

S → SS | (S) | ∈

(1)



(2)



~~It is~~.

Since there are two different derivation trees the grammar is ambiguous. And unambiguous which is equivalent to given cfg

To remove Ambiguity, we need to make the production rules.

$S \rightarrow ST/T$
$T \rightarrow (S)/()$

So by using this production rules we will get only one parse tree.

4 times in sem for 10mks each carries 2½ mks.

Let G be the grammar.

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

for the string aaabbabbba
find LMD, RMD, Parse tree, ambiguous.

Leftmost derivation:

$S \rightarrow aB$
$S \rightarrow aaBB$ (using $B \rightarrow aBB$).
$S \rightarrow aaaBBB$ (using $B \rightarrow aBB$)
$S \rightarrow aaabBB$ (using $B \rightarrow b$))
$S \rightarrow aaabbB$ (using $B \rightarrow b$)
$S \rightarrow aaabbaBB$ (using $B \rightarrow aBB$)
$S \rightarrow aaabbabB$ (using $B \rightarrow b$)
$S \rightarrow aaabbabbS$ (using $B \rightarrow bS$).
$S \rightarrow aaabbabbbA$ (using $B \rightarrow bA$)
$S \rightarrow aaabbabbba$. (using $S \rightarrow a$)

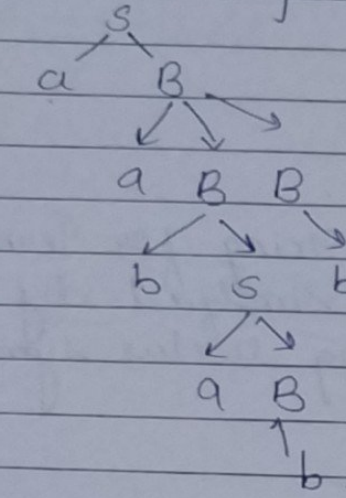
Right most derivation.

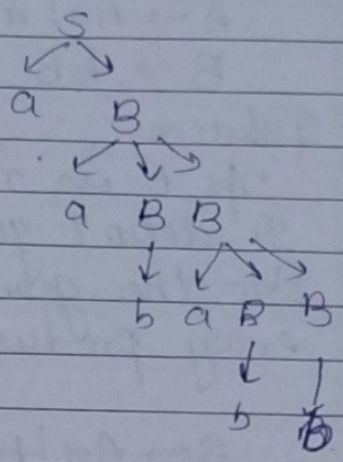
$S \rightarrow aB$
$S \rightarrow aaBB$ using $B \rightarrow aBB$.
$S \rightarrow aaBaBB$ using $B \rightarrow aBB$.
$S \rightarrow aaBaBbS$ using $B \rightarrow bS$.
$S \rightarrow aaBaBbbA$ using $S \rightarrow bA$.
$S \rightarrow aaBaBbba$ using $A \rightarrow a$.
$S \rightarrow aaBabbba$ using $B \rightarrow b$.
$S \rightarrow aaaBBabbba$ using $B \rightarrow aBB$
$aaaBabbba$ using $B \rightarrow b$
$aaabbabbba$ using $B \rightarrow b$.

For Ambiguous Check.

taking aababb

~~LMD~~. Taking String aababb ~~RMD~~.





* Simplification of CFG.
(Sequence is important)

1) Useless Symbol.
2) ε - production.
3) Unit production

Eg: $S \to a$, $S \to A$, $A \to a$, $B \to a$.
Generative.
$B \to a$.

11/9/22 1) Useless.

1) Elimination of useless symbols.
   a) Generative.
   b) Reachable

Two types of Symbols are useless Symbol.
1) Non-Generative
2) Non-Reachable.

a → Generative.

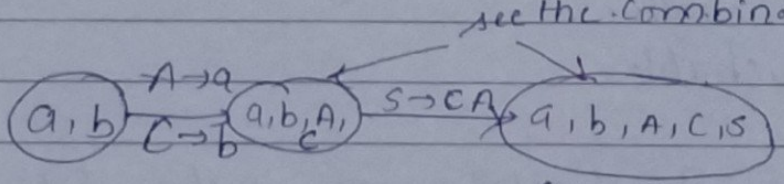
$S \rightarrow Aa \mid Bb \mid a \mid b$
$A \rightarrow Aa \mid a$.
$B \rightarrow bB$

Solution:
As B is non generating.
It doesn't generate any String for Terminals.
So, the grammer can be simplified by deleting every production containing useless symbol.

~~$S \rightarrow Aa \mid Bb \mid$~~
$S \rightarrow Aa \mid a \mid b$
$A \rightarrow Aa \mid a$

Q. Remove the non-Generating Symbols.

$S \rightarrow AB \mid CB$
$B \rightarrow BC \mid AB$
$A \rightarrow a$
$C \rightarrow aB \mid b$

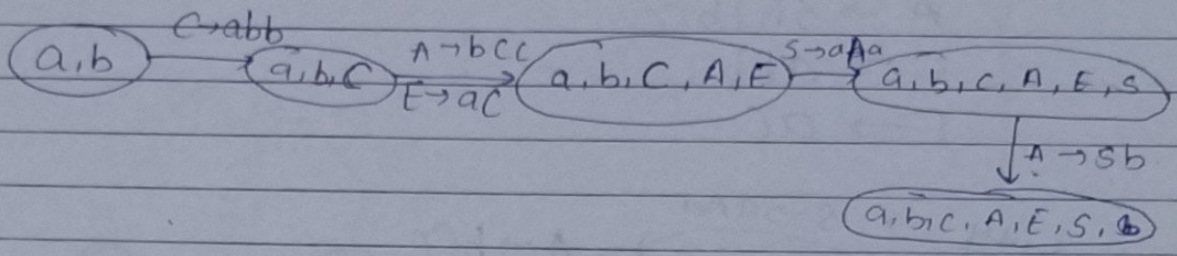


Set of generating symbol.

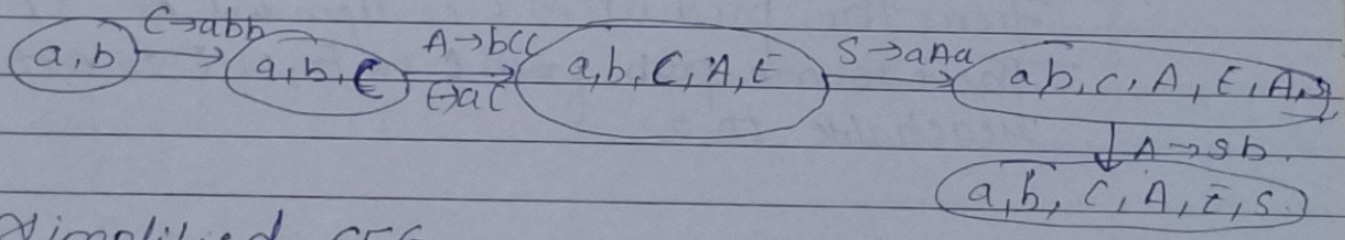
Reduced CFG is.

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

Q. $S \to aAa$
$A \to Sb \mid bCC$
$C \to abb$
$E \to aC$



$S \to aAa$
~~$A \to Sb$~~ $A \to Sb \mid bCC \mid DaA$
$C \to abb \mid DD$
$E \to aC$
$D \to aDA$



Simplified CFG.

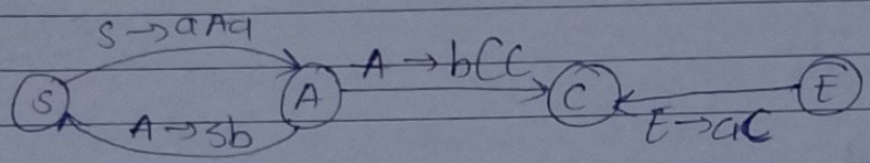
$S \to aAa$
$A \to Sb \mid bCC$
$C \to abb$
$E \to aC$

Non-reachable

Simified the Grammer using non-reachable test.

$S \rightarrow aAa$
$A \rightarrow Sb \mid bCC$
$C \rightarrow abb$
$E \rightarrow aC$

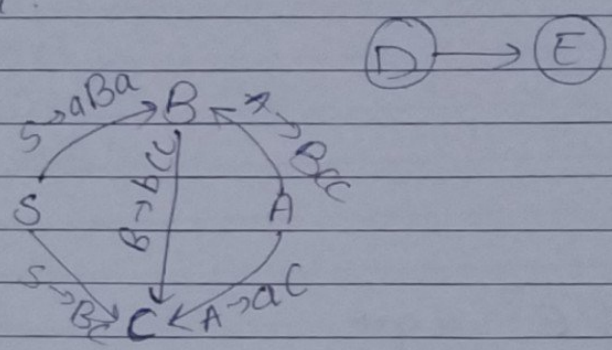
dependency graph.



From the production aAa A is dependent on S.
From the production C is dependent A. (A→bCC
From the production S is dependent A (A→Sb).
From the production C is dependent E (E→aC).
There is no path from S to E, so E is not reachable to S.

Simplified CFG: $S \rightarrow aAa$, $A \rightarrow Sb \mid bCC$ $C \rightarrow abb$

Q. $S \rightarrow aBa \mid BC$
$A \rightarrow aC \mid BCC$
$C \rightarrow a$
$B \rightarrow bCE$.
$D \rightarrow E$
$E \rightarrow d$.



A, D and E are non-reachable.

Simplified CFG.

$S \rightarrow aBa \mid BC$.
$A \rightarrow aC \mid BCC$.
$C \rightarrow a$.
$B \rightarrow bCC$.

* Elimination of ϵ-production.

$L(G') = L(G) - \{\epsilon\}$

Procedure for finding G'
i) find Nullable variables.
ii) Addition of production variable. Remove all the ϵ productions.

A is nullable as $A \to \epsilon$
B is nullable as $B \to \epsilon$
As A and B are nullable, S is also nullable.

$\{S, A, B\}$

$S \to ABA$
$A \to \epsilon$
$B \to \epsilon$

removal of ϵ we get,
simplified Grammer.
$S \to ABA | BA | AA | AB | A | B$

Q. $S \to ABA$ remove all the ϵ production.
$A \to aA | \epsilon$
$B \to aB | \epsilon$

$S \to ABA$
$S \to AB$ (use $A \to \epsilon$)
$S \to BA$ use $A \to \epsilon$.
$S \to AA$ $B \to \epsilon$
$S \to A$ $AB \to \epsilon$.
$S \to B$ $AA \to \epsilon$.

$A \to aA$ $B \to bB$.
$A \to a$ (use $A \to \epsilon$) $B \to b$ (use $B \to \epsilon$.

$S \to ABA|AB|BA|AA|A|B$

$A \to aA|a$

$B \to bB|b$

Q Remove all ϵ production.

$S \to AB$

$A \to aAA|\epsilon$

$B \to bBB|\epsilon$

$S \to AB$

$S \to A$ ~~$B \to \epsilon$~~ $S \to AB|A|B$

$S \to B$ use ~~$A \to \epsilon$~~

$A \to aAA$ ~~use $A \to \epsilon$~~

$A \to aA$ use $A \to \epsilon$. $A \to aAA|aA|a$

$A \to a$ use $A \to \epsilon$.

$B \to bBB$

$B \to bB$ $B \to \epsilon$. $B \to bBB|bB|b$

$B \to b$ $B \to \epsilon$.

* Elimination of unit production.

Eg :- $A \to B$

Q. $S \to ABA \mid BA \mid AA \mid AB \mid A \mid B$
$A \to aA \mid a$
$B \to bB \mid b$.

Solution.

Step 1: Till all non-unit production of Grammar which are

$S \to ABA \mid BA \mid AA \mid AB$
$A \to aA$
$B \to bB$.

Step 2: Locate every pair of variables such that $A_i \overset{\text{Generates}}{\Rightarrow} A_j$

$\{S, A\}$
$\{S, B\}$

Step 3: Unit production $S \to A$ can be removed by expanding A.

$A \to aA \mid a$

In the same way unit production $S \to B$ can be removed by expanding B.

$B \to bB \mid b$

pair $(S, A) \Rightarrow S \to aA \mid a$
pair $(S, B) \Rightarrow S \to bB \mid b$
$S \to ABA \mid BA \mid AA \mid AB \mid aA \mid a \mid bB \mid b$.

$P_1 \Rightarrow A \to aA \mid a$.
$B \to bB \mid a$

8. Eliminate unite production from the Given Grammar.

Sometimes

$E \to E+T \mid T$
$T \to T*F \mid F$
$F \to (E) \mid I$
$I \to a \mid b \mid Ia \mid Ib \mid I0 \mid I1$

Step 1: list all the non unit production.

$E \to E+T$
$T \to T*F$
$F \to (E)$
$I \to a \mid b \mid Ia \mid Ib \mid I0 \mid I1$

Step 2: locate Every pair of vari ables such that Generates $A_i \Rightarrow A_j$

$(E, T) \to E \to T$
$(T, F) \to T \to F$
$(F, I) \to F \to I$
$(E, F) - E \to T \& T \to F$
$(E, I) - E \to T, T \to F \& F \to I$
$(T, I) - T \to F \& F \to I$

Step 3: unit production can be remove by non-unit production.

$E \to T * f$

$T \to (E)$.

$f \to a|b|Ia|Ib|I0|Ib$

$E \to (E)$.

$E \to a|b|Ia|Ib|I0|II$

$T \to a|b|Ja|Ib|J0|J1$

CNF (Chomsky Normal form)

The CFG without ϵ production is said to Chomsky Normal form. If every production is of the form ~~$A \to Bc$ if~~.

$A \to BC$ { pair of variable

$A \to a$ or 1 terminal }

where $A, B, C \in \{V\}$.

$A \in \{V\}$

$a \in \{T\}$

find the CNF Equivalent to

~~$A \to BC$~~ $S \to aAbB$

~~$A \to a$~~ $A \to aA$

$B \to bB|b$

ϵ production. unproduction

variables terminal.

Step 1: Simplification of Gammer.

i) ϵ production.

2) Unit production

3) Useless pro

$A \xrightarrow{a}$
$B \to b$ (will not touch it hangs it)

The Grammer is with three production.

Step 2: Every Symbol in α in the production of the form $A \to \alpha$ where $|\alpha| \geq 2$
Every should be a variable
This can be achieved with the help of by adding two production,
$C_a \to a$
$C_b \to b$

Then the production will be
$S \to C_a A C_b B$ {change}. → {Can have more than 2 variable}

All in CNF

$A \to C_a A$
$B \to C_b B \mid b$
$C_a \to a$
$C_b \to b$

where $S \to C_a A C_b B$ {must have only 2 variables}
So, $S \to C_a C_1$ $\{C_1 \to A C_b B\}$
$C_1 \to A C_2$ $\{C_2 \to C_b B\}$
$C_2 \to C_b B$
$A \to C_a A$
$B \to C_b B \mid b$
$C_a \to a$
$C_b \to b$

This is Equivalent form.

Q* $S \to PQP$
$P \to OP/\epsilon$
$Q \to 1Q/\epsilon$
Soln: removal of ϵ ← step 1
Soln: $S \to PQP/OP/QP/PQ/P/Q/PP$
$P \to OP/0$
$Q \to 1Q/1$

Step 2: Here we got 2 unit productions which we have to remove.
ie $S \to P$ & $S \to Q$.
replace with $P \to OP/0$ & $Q \to 1Q/1$.

So we get $S \to PQP/QP/PQ/OP/0/1Q/1/PP$
It is in Normal form.

Step 3: Normal form to CNF

$S \to PQP/QP/PQ/C_0P/0/C_1Q/1/PP$ { do not change the CNF which are Exact 2 variable & 1 terminal }
$P \to C_0P/0$
$Q \to C_1Q/1$
Additional Productions have to used
$C_0 \to 0$
$C_1 \to 1$

Exact accept $S \to PQP$
$S \to PC_2$ { where $C_2 \to QP$ }
$C_2 \to QP$

$S \to PC_2 \mid QP \mid PQ \mid PP \mid C_0P \mid 0 \mid C_1Q \mid 1$

$C_2 \to QP$

$P \to C_0P \mid 0$

$Q \to C_1Q \mid 1$

$C_0 \to 0$

$C_1 \to 1$

Q. $S \to AACB$ $S \to AD \mid CD$

$A \to aAb \mid \epsilon$

$C \to aC \mid a$

$A \to aDa \mid bDb \mid \epsilon$

Soln:

Step 1: remove ϵ production.

$S \to AACD \mid ACD \mid CD$

Step 2: Non Generating:

where D is non Generating.

Since the Grammer is non-Generating.

Since the Start Symbol 'S' is non-Generating.

Hence it is invalid Grammer.

Q. Gammer:-

$S \to PQPS \to ASB/\epsilon$.

$A \to Aa\,s/a$

$B \to SbS/A/bb$

remove ϵ production.

$S \to ASB/AB$ { replace it once }.

$A \to Aas/a/Aa$.

$B \to Sbs/A/bb/bs/sb/b$

~~if~~ in B it is A Step 2:

So in $B \to A$ is a unit production. where $A \to Aa\,s/a/Aa$

~~$S \to ASB/AB$~~

~~$AB \to Sbs/Aas/a/Aa/bb/bs/sb/b$~~

* PDA

PDA involves seven components.

$P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ { Add memory to DFA by Add stack → (Infinite) is PDA }.

where Q = finite set of states like in FA

Σ = finite set of i

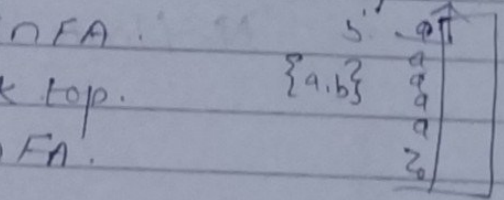
Γ = finite stack alphabet. { contents of stack }

δ = transition function As in FA. Stack top is Z_0.

q_0 = State Start State of A s in FA.

Z_0 = Start State of Stack top.

F = Subset of Q. as in FA.



· many time in SEM

w.w Reverse

Q.1) PDA for L = $\{ww^R\}$ — $\underbrace{0111}_{w}|\underbrace{1110}_{w^R}$ {match}

$P = (\{q_0, q_1, q_2\}, \{0,1\}, \{z_0, 0, 1\}, \delta, q_0, z_0, \{q_2\})$

where δ is defined by following rules.

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

1) $\delta(q_0, 0, z_0) = \{(q_0, 0z_0)\}$; $\delta(q_0, 1, z_0) = \{(q_0, 1z_0)\}$

2) $\delta(q_0, 0, 0) = \{(q_0, 00)\}$; ~~$\delta(q_0, 1, 1$~~ (old stack top)

Push (1),(2),(3)

$\delta(q_0, 0, 1) = \{(q_0, 01)\}$;

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

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

3) $\delta(q_0, \epsilon, z_0) = (q_1, z_0)$ {change state}, at center of ww^R

w is empty Accepted

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

$\delta(q_0, \epsilon, 1) = (q_1, 1)$

4) middle of the string is encounted. So matching is start & pop. Elt from the stack.

Poop

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

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

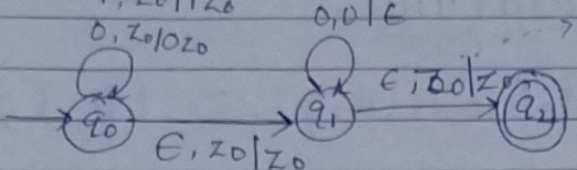
5) last — machine is has match

$\delta(q_1, \epsilon, z_0) = (q_2, z_0)$

PDA transition diagram.

Input, current state, top

1,1/11
1,0/10
0,1/01
0,0/00
1,Z0/1Z0
0,Z0/0Z0



Q2. $L = \{0^n 1^m : n \le m\}$

∵ no. of pop's are more. not match then ignore.

ϵ 0011 ϵ

n m

00011 X

$\delta(q_0, 0, Z_0) = \{(q_0, 0Z_0)\}$;

Start:

Push 1st zero

A) $\delta(q_0, 0, Z_0) = \{(q_0, 0Z_0)\}$; 2) $\delta(q_0, 0, 0) = \{(q_0, 00)\}$

last

Read 1st one & pop zero.

Push consecutive zero at stack.

pop

B) $\delta(q_0, 1, 0) = \{(q_1, \epsilon)\}$.

(current, input)

matching

4) $\delta(q_1, 1, 0) = \{(q_1, \epsilon)\}$.

Read

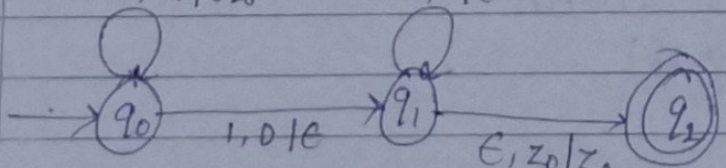
all the zeros are pop. but 1 is remain.

C) 5) $\delta(q_1, 1, Z_0) = \{(q_1, Z_0)\}$.

Ignore condn

A) $\delta(q_1, \epsilon, Z_0) = \{q_2, Z_0\}$.

0,0/00
0,Z0/0Z0
1,Z0/Z0
1,0/ϵ



Show the processing of string '00111' by PDA.

$(q_0, 00111, Z_0) \xrightarrow{R_1} (q_0, 0111, 0Z_0)$

$\xrightarrow{R_2} (q_0, 111, 00Z_0)$

$\xrightarrow{R_3} (q_1, 11, 0Z_0)$ 1 match with 0. 0 is pop.

$\xrightarrow{R_4} (q_1, 1, Z_0)$ 1 match with 0.

$\xrightarrow{R_5} (q_1, \epsilon, Z_0)$

$\xrightarrow{R_6} (q_2, \epsilon, Z_0)$

Q3) $L = \{a^m b^n : n < m\}$. {no. of a are more than b.}

Soln:

A) 1) $\delta(q_0, a, Z_0) = \{(q_0, aZ_0)\}$

2) $\delta(q_0, a, a) = \{(q_0, aa)\}$

B) 3) $\delta(q_0, b, a) = \{(q_1, \epsilon)\}$ 1st.

C) 4) $\delta(q_1, b, a) = \{(q_1, \epsilon)\}$ no. of b's

D) 5) $\delta(q_1, \epsilon, a) = \{(q_1, \epsilon)\}$.

E) 6) $\delta(q_1, \epsilon, Z_0) = \{q_2, Z_0\}$

16/9/22 GNF (Greibach Normal form)

CNF CFG $G = (V, T, P, S)$ is said to be in CNF.

The context free Grammer. If every production is said to be in form of $A \to a\alpha$ where $a \in T$ and $\alpha \in V^*$

ie GNF should start with terminal at RHS.

$A \to A\alpha$ {problem} $\longrightarrow$ $\{A \to a\alpha\}$

Left recursion $\{A \to A\alpha\}$

- Removing Left Recursion.

Language generated by left recursion.

i) $A \to A\alpha$ — Left recursive.

ii) $A \to \beta$ — for termination of recursion.

Then the language generated by the above production is $A \to A\alpha$

$A \to AA\alpha$

$A \to A\alpha\alpha$

$A \to A\alpha^n$

$A \to \beta\alpha^n$ $\longrightarrow$ from (ii)

Grammer

& Right recursion for $\beta\alpha^n$

~~This recursion~~.

The right recursive Grammer for $\beta\alpha^n$ can be written as

$A \to \beta B | \beta$ [where βB generates string α^n & production $A \to \beta$ is for the termination of recursion]

$B \to \alpha B | \alpha$

At last left recursive Grammer $A \to A\alpha$

$A \to A\alpha | \beta$

can be written as

$A \to \beta B | \beta$

$B \to \alpha B | \alpha$ ..... [explanation: $B \to \alpha^n$, $B \to \alpha B | \alpha$]

Q. Grammar with left recursion.

i) $A \to Aa | b$ ... [compare $A \to A\alpha | \beta$]

$A \to bB | b$

$B \to aB | a$.

ii) $A \to Aa | b | c$.

$A \to bB | cB | b | c$

$B \to aB | a$.

3) $A_1 \to A_1 A_2 A_3 | A_2 A_3$.

$A_1 \to A_2 A_3 B_1 | A_2 A_3$

$B_1 \to A_2 A_3 B_1 | A_2 A_3$

4) $A_1 \to A_1 A_2 A_3 | A_4 A_1 | A_5 A_3$

$A_1 \to A_4 A_1 | A_5 A_6 B_1 | A_4 A_1 A_6 B_1$

$B_1 \to A_2 A_3 B_1 | A_2 A_3$

5) $S \to S10 | 0$.

$S \to 0B_0 | 0$

$B \to 10B | 10$

Q.i) Construct Grammer in GNF which is equivalent to grammer.

$S \rightarrow AA|a$

$A \rightarrow SS|b$

1 step Grammer is in the simple form without ϵ production, Useless production.

~~Remar~~ Renaming variables.

S as A_1 & A & A_2.

$A_1 \rightarrow A_2A_2 \mid a$

$A_2 \rightarrow A_1A_1 \mid b$

Step 2:

19/3/22 Lemma

Q2.* Let $A \to A\alpha_1 | A\alpha_2 | A\alpha_3 | \ldots | A\alpha_r$.

Let

$A \to \beta_1 | \beta_2 | \beta_3 | \ldots | \beta_s$ are remaining productions

then.

By adding variables B.

1) $A \to \beta_i$
$A \to \beta_i B$ } $1 \le i \le s$

2) $B \to \alpha_i$
$B \to \alpha_i B$ } $1 \le i \le r$.

Convert following grammer to Greibach Normal form

$G = \langle V, T, P, S \rangle$

Q. $G = (\{A_1, A_2, A_3\}, \{a, b\}, P, A_1)$.

where P is.

when [$A_1 \to A_2 A_3$
$A_2 \to A_3 A_1 | b$

Hiolus man RHS (A2 > A1) $A_3 \to A_1 A_2 | a$.

Step2: Since the RHS of the productions of A_1 & A_2 Start with terminals or higher numbered variables. We begin with production $A_3 \to A_1 A_2$.

and Subtitute the String

$A_1 \to A_2 A_3$, $A_2 \to A_3 A_1 | b$

$A_3 \to A_2 A_3 A_2 | a$

Since the ~~two~~ R.H.S of production $A_3 \rightarrow A_2 A_3 A_2 b$ begins with lowered numbered variable. We substitute for the ~~first~~ 1st Occurance of the A_2.

$A_1 \rightarrow A_2 A_3$.

$A_2 \rightarrow A_3 A_1 / b$.

$A_3 \rightarrow \underbrace{A_3 A_1 A_3 A_2}_{A} / \underbrace{b A_3 A_2}_{B_1} / \underbrace{a}_{B_2}$.

Apply the lemma 2:

~~$A_3 \Rightarrow A_3 A_1 A_3 A_2$~~

$A_1 \rightarrow A_2 A_3$

$A_2 \rightarrow A_3 A_1 / b$

$A_3 \rightarrow b A_3 A_2 B_3 / a B_3 / b A_3 A_2 / a$

$B_3 \rightarrow A_1 A_3 A_2 B_3 / A_1 A_3 A_2 B_3$

Step 2:- Now all the productions with A_3 on the left have right hand sides that start with terminal or higher numbered variables. Those which starts with terminal are used to replace A_3 in the production $A_2 \rightarrow A_3 A_1$

$A_3 \rightarrow b A_3 A_2 B_3$

$A_3 \rightarrow a B_3$.

$A_3 \rightarrow b A_3 A_2$

$A_3 \rightarrow a$.

- $A_2 \rightarrow A_3 A_1 B_3$.
  replace A_3 pro by A_3 production.

  $A_2 \rightarrow bA_3A_2B_3A_1$
  $A_2 \rightarrow aB_3A_1$
  $A_2 \rightarrow bA_3A_2A_1$
  $A_2 \rightarrow aA_1$
  $A_2 \rightarrow b$

- $A_1 \rightarrow A_2 A_3$.
  Replace A_2 by its production at above.
  $A_1 \rightarrow bA_3A_2B_3A_1A_3$
  $A_1 \rightarrow aB_3A_1A_3$
  $A_1 \rightarrow bA_3A_2A_1A_3$
  $A_1 \rightarrow aA_1A_3$
  $A_1 \rightarrow bA_3$.

$B_3 \rightarrow A_1A_3A_2 B_3$
$B_3 \rightarrow A_1A_3A_2B_3$.

The B_3 prod^n has to be converted in normal form by replacement of production.

$B_3 \rightarrow A_1A_3A_2$.

$B_3 \rightarrow bA_3A_2B_3A_1A_3A_3A_2$.
$B_3 \rightarrow aB_3A_1A_3A_3A_2$
$B_3 \rightarrow bA_3A_2A_1A_3A_3A_2$
$B_3 \rightarrow aA_1A_3A_3A_2$
$B_3 \rightarrow bA_3A_3A_2$

$A \rightarrow \alpha A$

$\alpha \rightarrow$ all variables

$B_3 \rightarrow bA_3$

$B_3 \rightarrow A_1 A_3 A_2 B_3$

A_1 replaced by A_1 production's to get terminal at first in RHS.

$B_3 \rightarrow bA_3A_2B_3A_1A_3A_3A_2B_3$

$B_3 \rightarrow bB_3A_1A_3A_3A_2B_3$

$B_3 \rightarrow bA_3A_2A_1A_3A_3A_2B_3$

$B_3 \rightarrow aA_1A_3A_3A_2B_3$

$B_3 \rightarrow bA_3A_3A_2B_3$

Pumping Lemma

In pumping lemma we use pigeon hole principle

If Language 'L' is infinite regular language then their exist some positive integer 'm' such that, any string $w \in L$ whose length is m or greater than m, can be decomposed into three parts $w = xyz$. Where

Their are three must conditions:-

i) length of $|xy| \leq m$

ii) $|y| > 0$

iii) $w_i = xy^iz$ is also in L for all $i = 0, 1, 2, 3 \ldots$

for eg: $L = \{a^n b^n : n \geq 0\}$

$w = xyz$ — x | y | z

aaaabbbbbb

$|xy| = 5$ a's.
$|z| = 5$ b's.
$|x| = 4$ a's.
$|y| = 1$ a

for

Steps for Proofing non-regular Grammer.

i) We need to show, for any choice m.
ii) for some $w \in L$ that we get to choose ($w = xyz$)
iii) $w = xyz$ decompose in, so $|xy| \leq m$ & y is not empty.
iv) we choose i such that xy^iz is not in L.

1) $L = \{a^n b^n : n \geq 0\}$ prove that it is not regular by using pumping lemma.

Pumping lemma + problem

Def^n + cond^n + def^n

i) we don't know m let us consider m.
ii) Choose string $w = a^n b^n$ where $n \geq m$
iii) So that, any prefix of length m consist only a's.

iii) We don't know the decomposition of w into xyz but since length of xy $|xy| \leq m$.

FA, RE, Turing, PDA }

$|xy|$ must consist entirely of a's. Also y cannot be empty.

iv) Choose $i = 0$, there has an effect of droping $|y|$ a's out of the string without affecting the number of b's. The resultant string has fewer a's than b's. Hence $a^n b^n$ does n't belongs to L.

So L is not regular.

2) $L = \{a^n b^k : n > k \text{ & } n \geq 0\}$ is not regular.

$a^n b^{n-1} \Rightarrow a^{n-1} b^{n-1}$ not regular.

$x \; y$

3) $L = \{a^n$ where n is a prime number$\}$.

Turing Machines

- It is finite automaton with a twist

A Turing machine (tm)
$M = (Q, \Sigma, \Gamma, \delta, \square, q_0, F)$ where
Q is a finite set of states
Σ is a input alphabet
Γ is a finite set of symbol (the tape alphabet)
δ is the transition
$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$
$\square \in \Gamma$ is a special symbol called the blank
$\Sigma = \Gamma - \{\square\}$
q_0 is the ~~tm~~ Initial State
F is the set of final (accepting) States.

- A tm starts in the initial state, generally looking at the leftmost non-blank symbol on the tap.

- Acceptor : accept or reject.

- Transducer

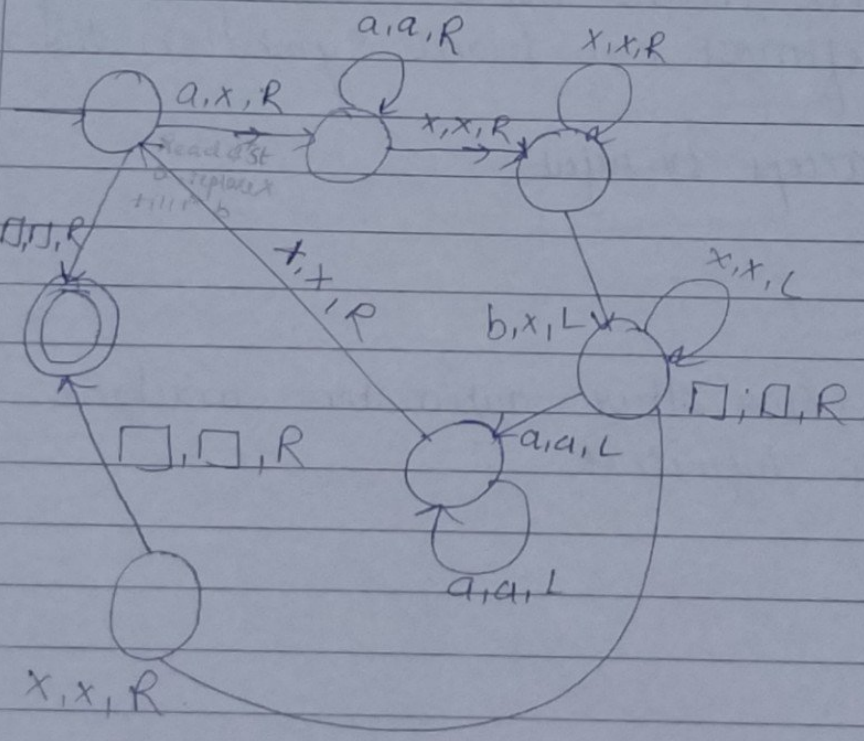
Note unlike our other automaton. machine can go in into infinite

aaabbb
x

Q. Can it accepts the language $\{a^n b^n\}$? How
→ Lay out a Strategy.

- Let's mark off each a & matching b with an x.
- Let's work with the a's and b's from left to right
- Whatever we mark off an a, move right until we find a b. Mark off the b & head back left looking for the left-most remaining a. Repeat as long as you can.
- If there are no more a's, go to the right and check that there are no more b's.
- Let machine die if there is no

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



aaaxbbb
aaaxbbbb □□,R

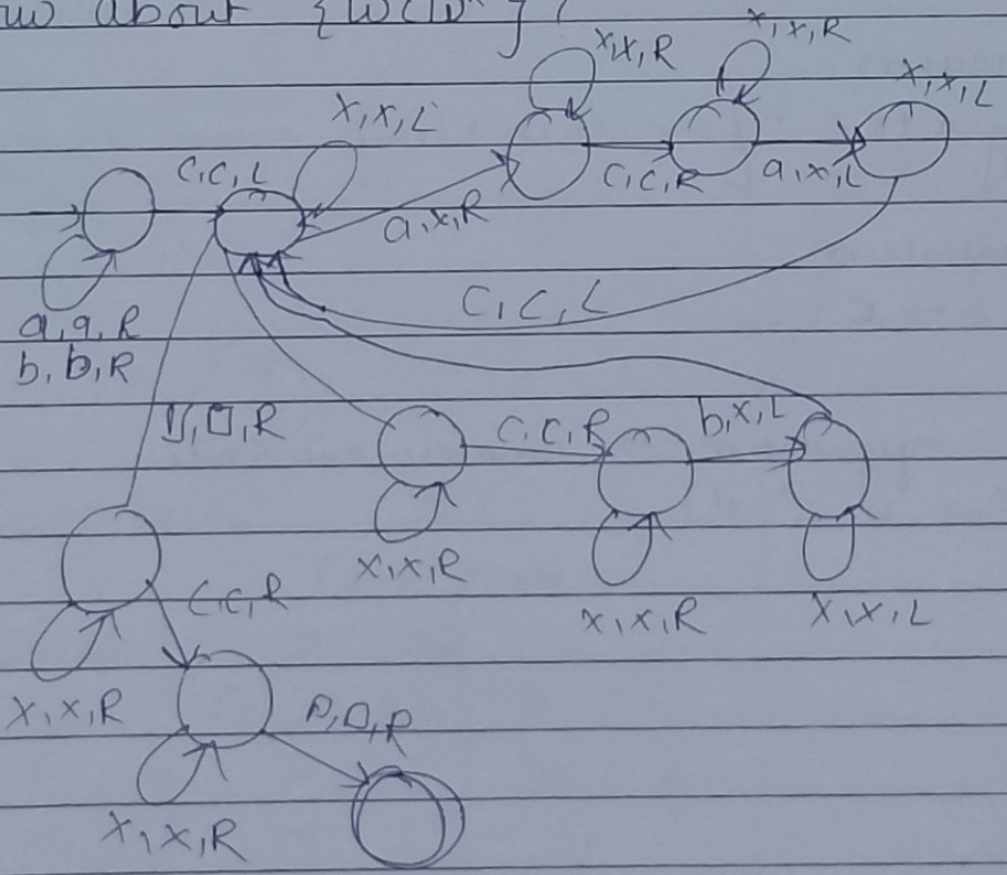
Tm Can Start anyway

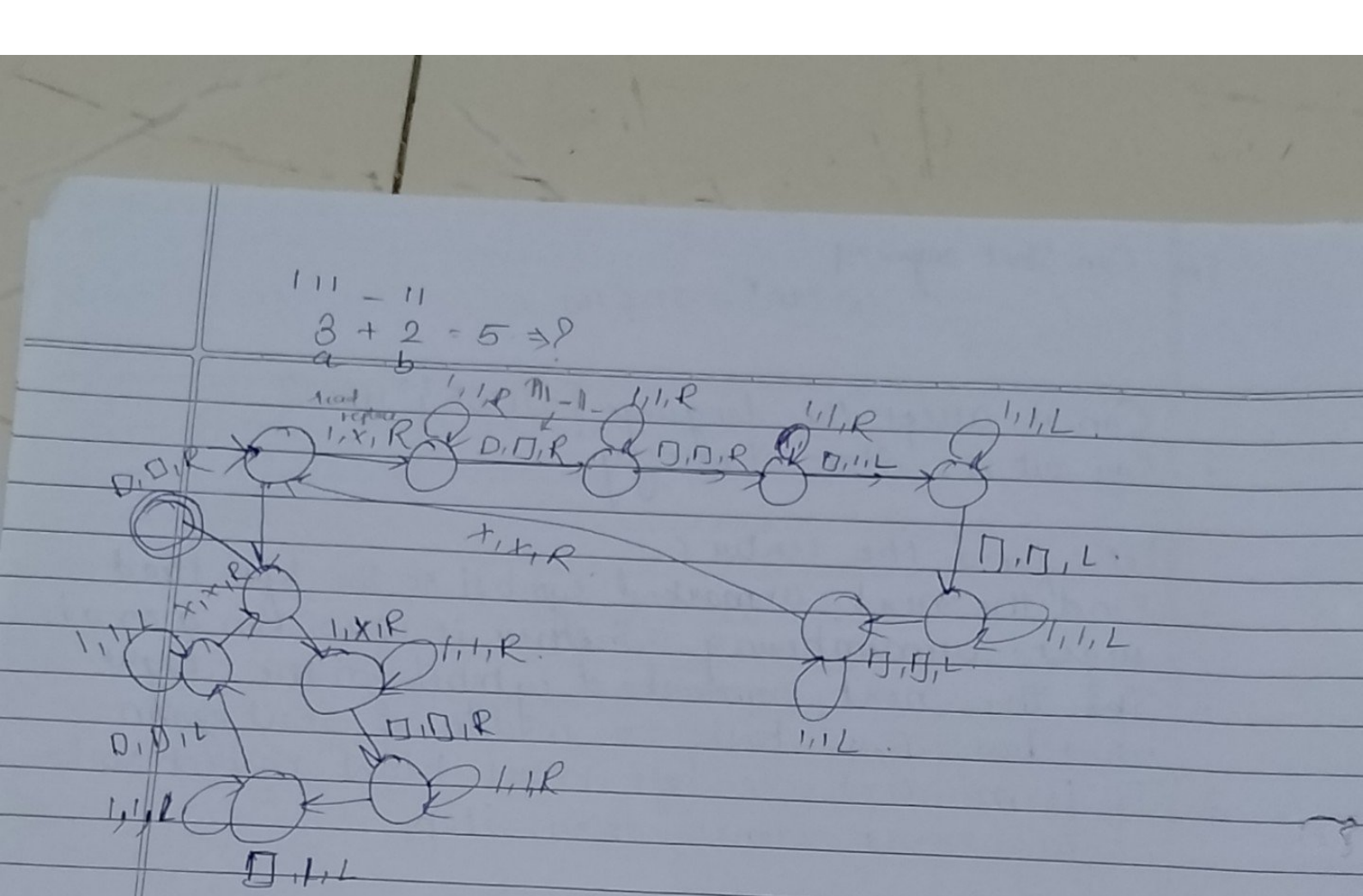
wcw^R - 011C110

Can it accept the language $\{wcw^R\}$? How?

- Lay out a ~~Stat~~ strategy.
- Let's find the centre c
- Find the next unmarked symbol on the left. Head right. remembering whether it was an a or a b
- If the next unmarked symbol on the right matches, head back to middle & Start again.
4. If from centre c left most blank match with right most blank then Stop.

How about $\{wcw^R\}$?





~~Transducer~~ Strategy: mark off each 1 in a and copy after the blank to the right

Subtraction.
copy a to c. for each 1 in b erase a 1 in c.

Multiplication:
Copy a to c.

TM configuration or instantaneous state
(s, a, b, s', d)

find a TM that recognizes the set
$L = \{x \in \{1\}^* , |x| = \text{even}\}$

Soln:

TM start from s_0

$R_1 : (s_0, B, B, s_2, R)$

$R_2 : (s_0, 1, 1, s_1, R)$

$R_3 : (s_1, 1, 1, s_0, R)$

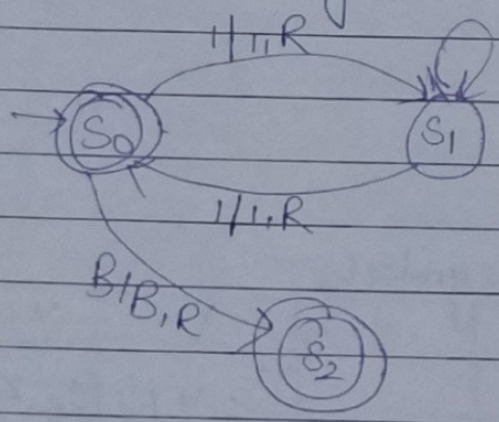
verification - Input $x = 1111$

$s_0 1111 B \vdash 1 s_1 111 B$

$\vdash 11 s_0 11 B \vdash 111 s_1 1 B$

$\vdash 1111 s_0 B \vdash$

transition diagram.



transition table :-

State	Tape symbol	
	B	1
s_0	(s_2, B, R)	$(s_1, 1, R)$
s_1	-	$(s_0, 1, R)$
s_2	-	-

Q* Construct a TM that recognizes the language $\{a^n b^n c^n \mid n \geq 0\}$.

Strategy =

1. If the current cell is empty, TM halts with success.
2. If the current cell is 'a', the tape head replaces it by symbol 'x' and tape head moves to the right looking for a corresponding b. the right of any a's and when found, the tape head replaces b with another symbol 'y' and continue scanning to the right, now looking for a symbol 'y' and corresponding c to right

Transition table -

State	Tape symbols						
	a	b	c	d	x	y	z
S_0	(S_1, x, R)	–	–	–	(S_0, y, R)	(S_4, z, R)	(H, B, N)
S_1	(S_1, a, R)	(S_2, y, R)	–	–	(S_1, y, R)	–	–
S_2	–	(S_2, b, R)	(S_3, z, L)	–	–	S_2	
S_3	(S_3, a, L)	(S_3, y, L)	–	(S_0, x, R)	(S_1, y, L)	$(S_3, z, L$	–
S_4	–	–	–	–	–	(S_4, z, R)	(H, B, N)

• Church-Turing Thesis:-
• Write SN on The universal turning machine
q_n as m

1 → L , 11 → R , 111 → q_0 Imp. Question

- Church Turing machine.
- Universal Turing machine. — Every SEM 5mks. Possible Or 10mks.

Possible Question [+ 1 Eg compulsory]

1) Recursive & ~~Recurlivity~~ Recursively ~~to~~ Enumeriable ~~tab~~ lagnages.
2) ~~Intav~~ Intractable Theory.
3) Simplification of ~~CFG~~ CFGs
4) decidation properties of regular grammer.
5) Rice's Theorem.
6) Chomsky hierarchy.
7) ~~Gibe~~ Greibach Theorm
8) Haltingion problem.
9) Post correspondance problem
~~10~~)

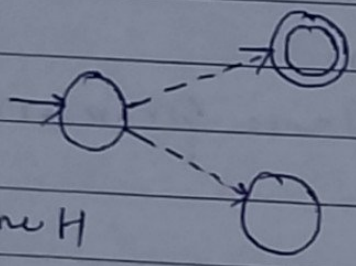
~~Intractable~~ Imp 10mks
~~NP~~ NP.
P. [which can be solved by deterministic TM]
NP complete poorbrems.]. → Exam.
NP hard.

~~Enune~~ Enumeriable → able to solve. [If the soln exist]
$a^n b^n \rightarrow a^2b^2, a^3b^3$

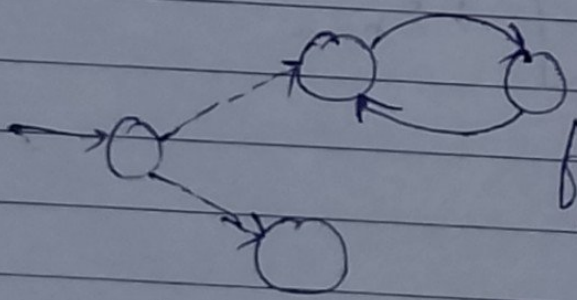
Definition

* Decidability: A problem is decidable if we can build a TM to provide a 'Y' or 'N' result in all cases.
* Computability
* A fun^n is computable if we can build a TM that computes it in all cases.
* Halting problem is undecidable.

Halting problem



→ Suppose H exists. Then we can clearly modify machine H into the following ~~machine~~.



Intractable problem.

- There exists problems which cannot be solved in polynomial time, called Intractable problems.

- A problem is tractable if it belongs to that class P. The problem can be solved by Non-deterministic TM in polynomial time complexity are denoted by NP-problems. There are some problems such has SAT. with property that of any R in P then all of NP is P. we call such problems to be NP complete problems.

- So we say that problem Q is NP complete if the following condition holds abt the Q.
  i) Q is in NP. and
  ii) For Every problem R in NP, R is polynomial reducible to Q.

- A problem Q with that property 2 is called NP hard problem. Thus we can say Q is the NP and Q is NP-hard.

Fomal def^n NP-completeness:
A decision problem is NP-complete if
i) C is in NP and
ii) Every problem in NP is reducible to C in polynomial time
iii) C can be shown.

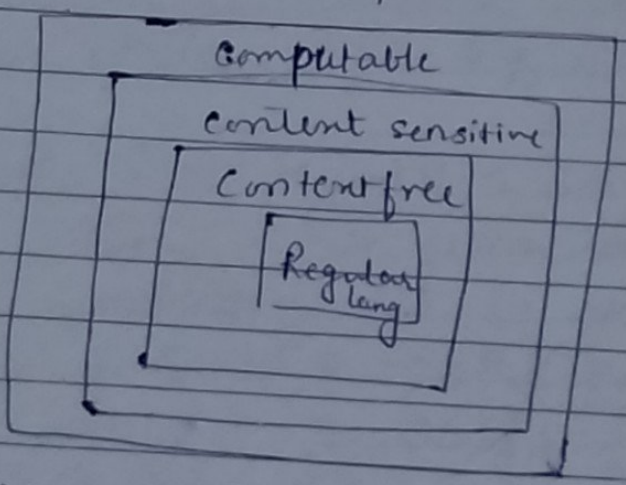
minsky
Expand

NP complete problem.
Eg i) graph Isomorphism problem.
ii) graph theory.
~~iii) graph Isomorphonphic.~~
~~iv)~~ iv) ~~graph theory.~~
iii) Boolean ~~satisf~~ satisfiability problem (sat.)
iv) N-puzzle.
v) knapsack problem
vi) Hamiltonian path problem.
vii) Travelling Sales-man problem.
viii) Subgraph isomorphic.

NP hard:
A problem H is ~~H~~ NP hard if and only if there is an NP-complete problem L that is polynomial time Turning ~~machine~~ reducible to H (ie $L \leq_T H$).
In other word L can be solved in polynomial Time
Eg: Subset sum problem.
Travelling salesman problem
~~decision~~ decision problem.

Chomsky Hierarchy represents the class of language that are excepted by different machines.

language class	language	Grammer	machines	Eg
Type 3	Regular	Regular	FSM DFA/NFA	a^*b^*
Type 2	context free	context free	PDA	$a^n b^n$
Type 1	decidable	Context sensitive	linear bounded automaton	$a^n b^n c^n$
Type 0	Computable	Unrestricted	TM	all



Every language of type 3 is also type 2, 1, 0
Every language of type 2 is also type 1, 0
~~Every~~

CNF eg
S→ABa
A→aab
B→Ac

Imp

Note: The RHS. starts with terminal only.

Initial z_0 is ~~Z_0~~ z_0

GNF form $[S \to aV^*]$ CNF form $\begin{bmatrix} S \to BC \\ S \to a \end{bmatrix}$

1/10/22

to be mentioned in SEM

Construct NPDA corresponding to grammar

$S \to aABB \mid aAA$

$A \to aBB \mid a$

$B \to bBB \mid A$

Solution

first Convert given Grammar into GNF

$S \to aABB \mid aAA$

$A \to aBB \mid a$

$B \to bBB \mid aBB \mid a$ $\quad B \to A$ [has to be removed].

~~It is now it~~ is Now is in GNF form.

~~Corre~~ The Corresponding Equivalent NPDA will required three States. $\{s, q_1, q_2\}$

First the Start symbol S is put on the Stack

The transition is given as $\delta(s, \lambda, z_0) = \{(q_1, Sz_0)\}$

Other productions can be given as

$\delta(q_1, a, S) \Rightarrow \{(q_1, ABB)(q_1, AA)\}$

$\delta(q_1, a, A) \Rightarrow \{(q_1, BB), (q_1, \lambda)\}$ $\quad \{\lambda$ or ϵ is same$\}$

$\delta(q_1, b, B) \Rightarrow \{(q_1, BB),$

$\delta(q_1, a, B) \Rightarrow \{(q_1, BB), (q_1, \lambda)\}$

Finaly NPDA is put in it's final States by transition:-

$\delta(q_1, \lambda, z_0) = \{(q_2, \lambda)\}$

* (Stating variable is 'S' But it

wcw^R / ww^R — 1 comes

10011,11001

abbba c abbba.

Imp Construct NDPA wcw that accepts the language language wcw^R

$L = (wcw^R \mid w \in (a,b)^*)$ give the o/p on the given string abbcbba.

Solution:

$M = \{(q_0, q_1, q_2), \{a, b, c\}, \{z_0, a, b\}, \delta, q_0, z_0, \{\ \}\}$

If w starts with a then a is push or b is push (a or b)

where δ is given as $\delta(q_0, \epsilon, z_0) =$

I) i) $\delta(q_0, a, z_0) = (q_0, a z_0)$ { It may start with
ii) $(q_0, b, z_0) = (q_0, b z_0)$. a or b }

for next aa, bb, ab, ba

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

$\delta(q_0, a, b) = (q_0, ab)$

$\delta(q_0, b, a) = (q_0, ba)$

$\delta(q_0, b, b) = (q_0, bb)$

II) $\delta(q_0, c, a) = (q_1, a)$

on current stack $\delta(q_0, c, b) = (q_1, b)$

It may end with a or b → $\delta(q_0, c, z_0) = \delta(q_1, z_0)$

If w contains nothing then w^R also contains nothing

III $(q_1, a, a) = (q_1, \epsilon)$.

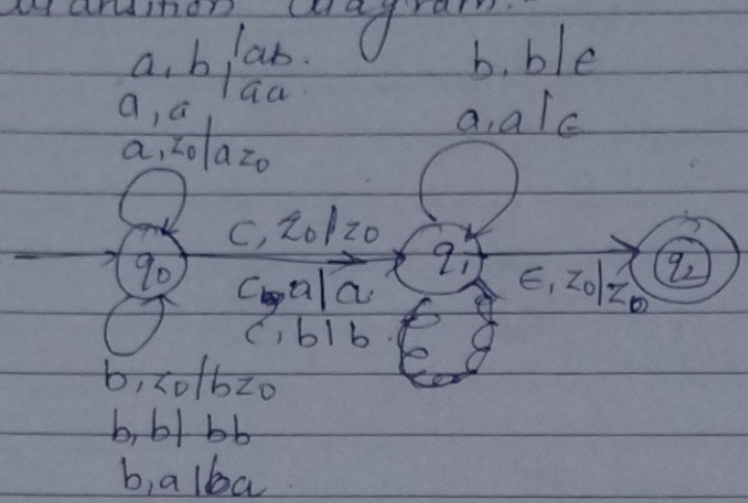
The current stack top must be a $(q_1, b, b) = (q_1, \epsilon)$

IV) $\delta(q_1, \epsilon, z_0) = \delta(q_2, z_0)$.

w contains uptill c it enters into q_0.

Has soon as w^R ends.

Transition diagram:-



for to test.

Let x = abbcbba.

$(q_0, abbcbba, z_0) \rightarrow (q_0, bbcbba, az_0)$.
$(q_0, bbcbba, az_0) \rightarrow (q_0, bcbba, baz_0)$.
$(q_0, bcbba, baz_0) \rightarrow (q_0, cbba, bbaz_0)$
$(q_0, cbba, bbaz_0) \rightarrow (q_1, bba, bbaz_0)$
$(q_1, bba, bbaz_0) \rightarrow (q_1, ba, baz_0)$
$(q_1, ba, baz_0) \rightarrow (q_1, a, az_0)$.
$(q_1, a, az_0) \rightarrow (q_1, e, z_0)$
$(q_1, e, z_0) \rightarrow (q_2, e, z_0)$