→ S → 0A
A → 1A
A → 1

→ (3) (A)

Chapter-5
Regular Set & Regular Expression

- Verbally with the help of mathematical operation.

$L(G) = \{0 1^n \mid n > 0\}$

in set form $= \{01, 011, 0111 \ldots\}$

$L(G)$ = set of all strings over $\Sigma = \{a, b\}$ which contains even no's a's }

→ Regular Expression :-

It is way of representing certain set of string in algebraic fashion.

(i) Any terminal symbol including $\wedge$ & ϕ are regular expression.

(ii) Union of two regular expression R_1 and R_2 $(R_1 + R_2)$ is also RE

(iii) The concatenation of two regular expression $(R_1 . R_2)$ is RE.

(iv) The iteration of RE R^* is also a RE

(v) If R is RE then (R) is also a RE

(vi) The RE over Σ are those RE which are obtained from above.

• NOTE

Parenthesis in Rule 5 used to influence the order of evaluation of given RE. In absence of parenthesis its highest priority iteration → concatenation → union

(a) $\{101\} = 101 = 1.0.1$

(b) $\{abab\} = abab = a.b.a.b$

(c) $\{01,10\} = 01+10 = 0.1+1.0$

(d) $\{\wedge, ab\} = \wedge + ab = \wedge + a.b$

(e) $\{abb, a, b, bba\} = abb + a + b + bba = a.b.b + a + b + b.b.a$

→ +, *, ., ( ) are used to write the regular expression.

→ Set of all strings over $\{0,1\}$ ending with 00.

$$L(r) = (0+1)^* 00$$

→ Set of all strings over $\{0,1\}$ begins with 0 and ends with 1

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

→ Find RE over $\{a,b\}$ the set of strings containing exactly two a's.

$$= b^* a a b^* + a a b^*$$

$$L(r) = b^* a b^* a b^* + b^* a b^*$$

→ Write an RE for set of all strings containing atleast two a's

$$L(r) = (a+b)^* a (a+b)^* a$$

→ Atmost two a's

$$L(r) = b^* a b^* a b^* + b^* a b^*$$

→ Contain 'aa' as substring

$$L(r) = (a+b)^* aa (a+b)^*$$

→ Write an R.E for starting with any no. of a, followed by one or more b's, followed by one or more a's, followed by single b, followed by any no. of a, followed by single b and ending with the any no. of a's & b's

$$L(r_1) = a^* bb^* aa^* b(a+b)^*$$

string $(\wedge, \phi)$ alphabet of

Identity for RE :-

1. $\phi + R = R$
2. $\phi R = R\phi = \phi$
3. $\wedge R = R\wedge = R$
4. $\wedge^* = \wedge$ and $\phi^* = \wedge$
5. $R + R = R$
6. $R^* R^* = R^*$
7. $RR^* = R^* R$
8. $(R^*)^* = R$
9. $\wedge + RR^* = R^* = \wedge + R^* R$
10. $(PQ)^* P = P(QP)^*$
11. $(P+Q)^* = (P^* Q^*)^* = (P^* + Q^*)^*$
12. $(P+Q)R = PR + QR$ & $R(P+Q) = RP + RQ$

Prove: $(1+00^*1) + (1+00^*1)(0+10^*1)^*(0+10^*1)$
$= 0^*(0+10^*1)^* \;/\; 0^*1(0+10^*1)^*$

Date - 13/9/25

Regular Expression to Finite Automata

FA
- NDFA
- DFA

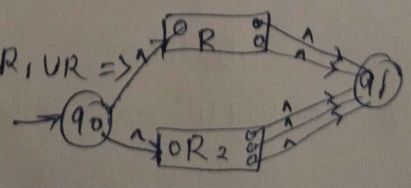
$R_1 = (0^*1)$
$R_2 = (1^*0) = R_2^*$

① $\delta(Q, a) =$ no unique ~~$R_1 \cup R_2 =$~~
② Σ
③ $\wedge$- moves

→ (0) $R = \wedge$ • —(0) $R = \phi$ • R_1 / R_2 $= R_1 + R_2 =$

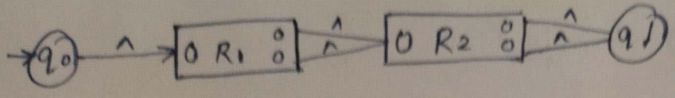
- Create new initial & final state.
- Connect them and moves with $\wedge$

$R_1 \cup R =$

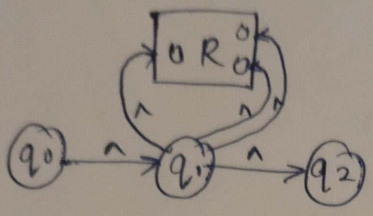


[Concatination two FA]

$R_1 \quad R_2 = R_1 \cdot R_2$

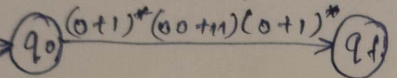


→ $R = R^*$

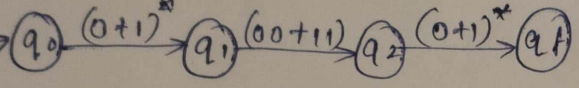


→ $(0+1)^* (00+11) (0+1)^*$ FA

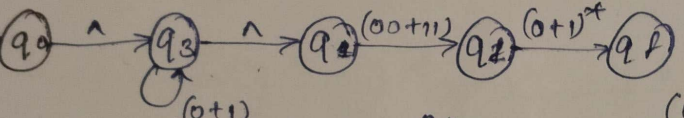
1]



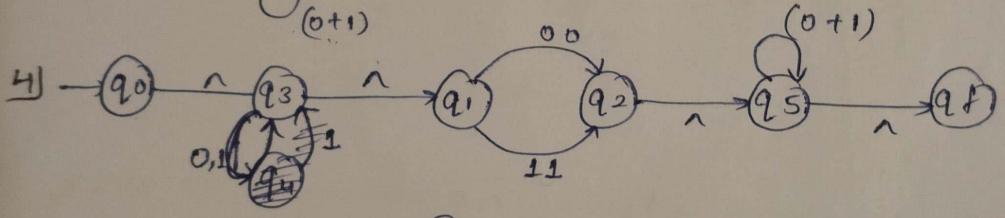
2]



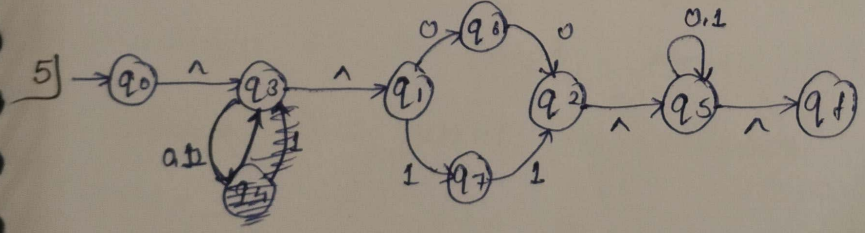
3]



4]

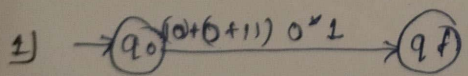


5]

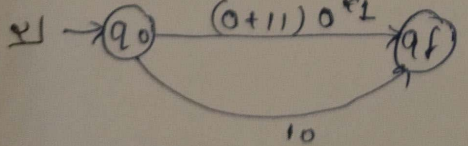


→ $10 + (0+11)\ 0^*1$

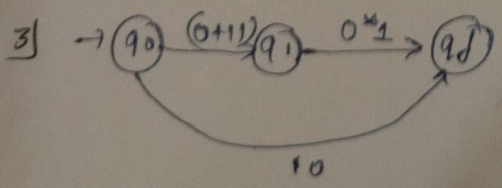
1]



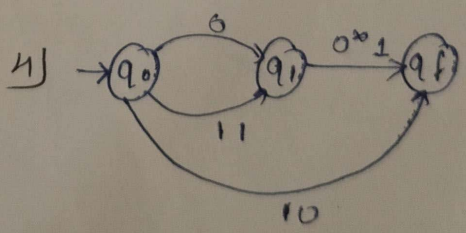
2]



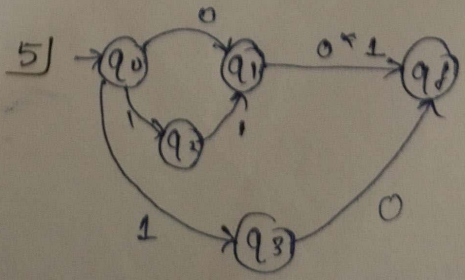
3]

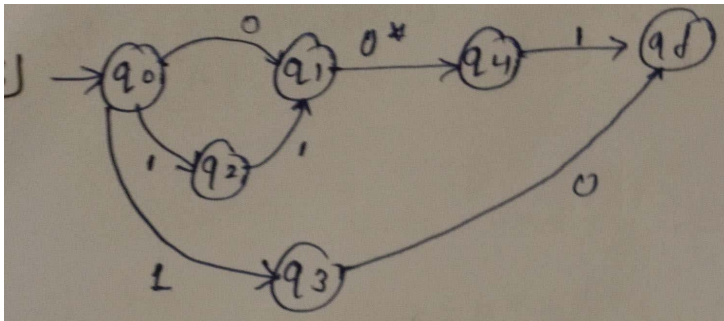


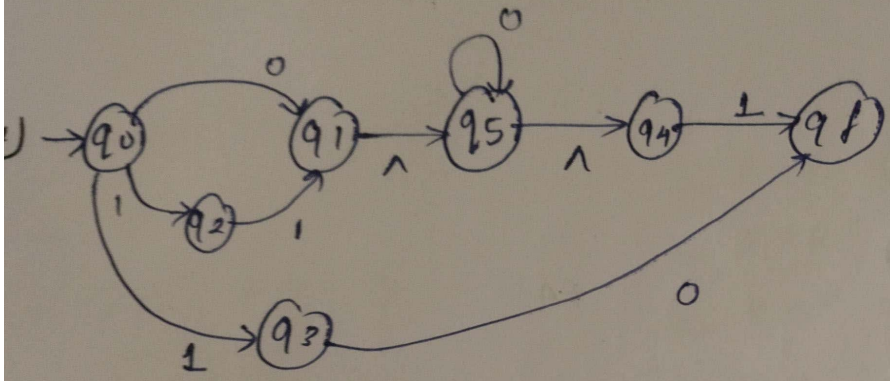
4]



5]



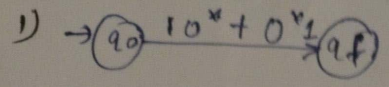


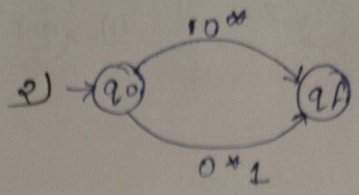


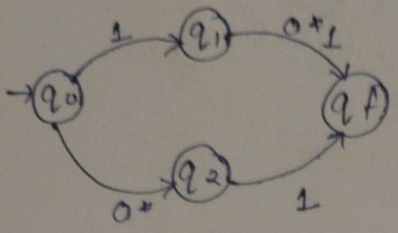
→ $00 + 11^* + 00(11^*)^*$

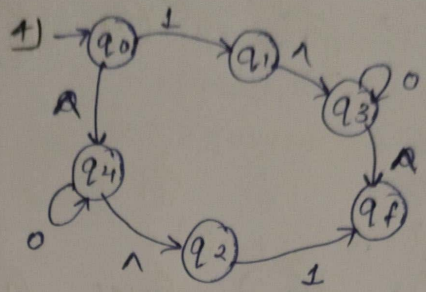
Date - 17/9/25

→ $10^* + 0^*1$

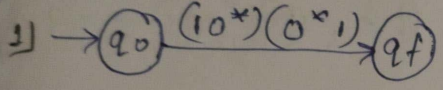
1) 

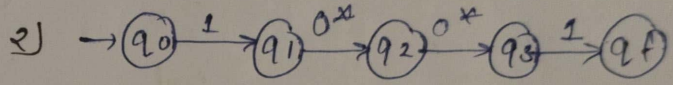
2) 

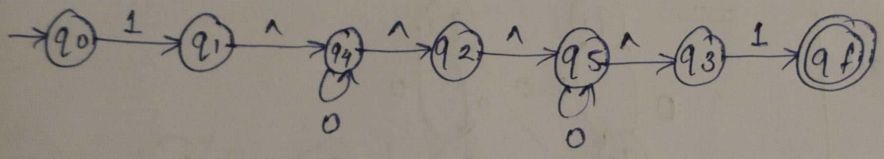
3) 

4) 

→ $(10^*)(0^*1)$

1) 

2) 

3) 

(empty moves is called NDFA)

How to Remove an empty moves in a FA.

1) Conversion of FA empty moves to FA w/o empty moves

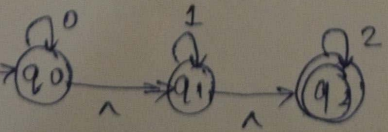
- Suppose we want to remove empty moves vertex V_1 and V_2.

S-1 Find all the edges starting from V_2.

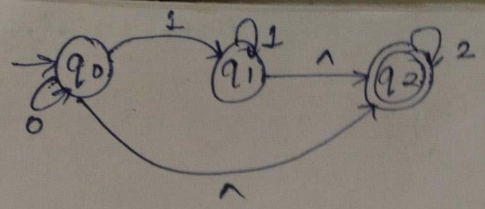
S-2 Duplicate all this edges starting from V_1, without changing the labels of edges.

S-3 If V_1 is initial state make V_2 also a initial state.

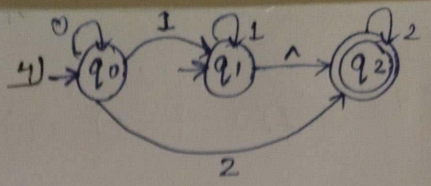
S-4 If V_2 is final state make V_1 also a final state.

$V_1 \xrightarrow{\wedge} V_2$ 1) 

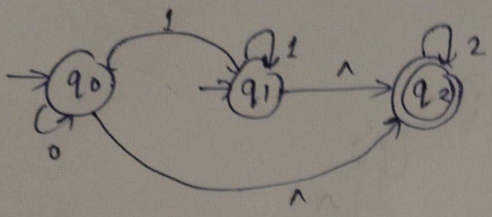
2)



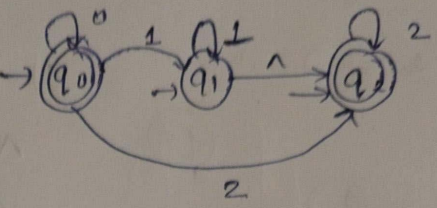
4)



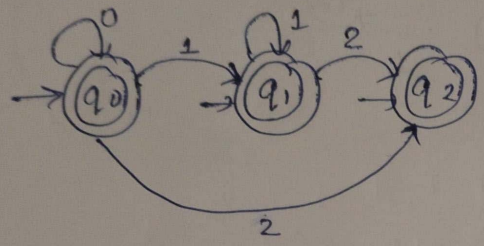
3)



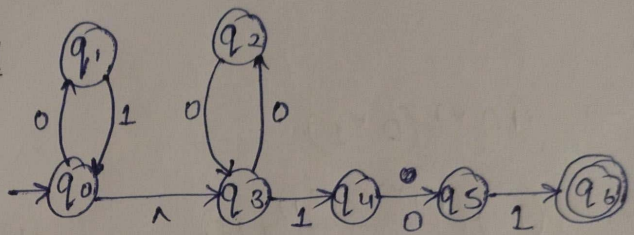
5)



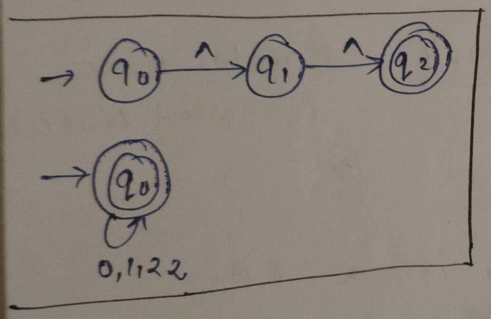
6)

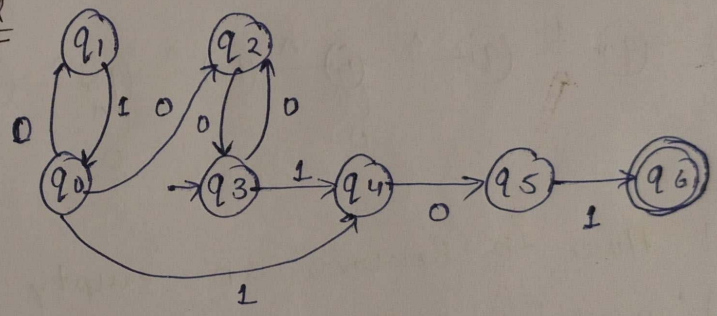


Ques

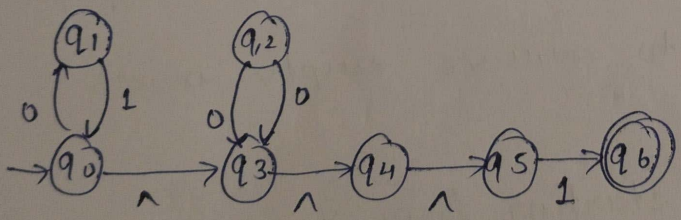


Sol



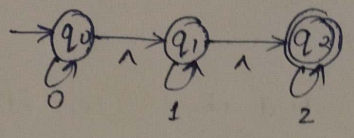


Ques



1) 

2) Equivalence Method



A • λ- Equivalence of $(q_0) = (q_0, q_1, q_2)$

B • λ " " $(q_2) = (q_1, q_2)$

C • λ " " $(q_2) = (q_2)$

$\Sigma(0,1,2)$ (A)

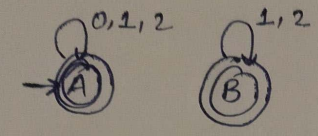
- $(q_0, q_1, q_2, 0) = (q_0,0) \cup (q_1,0) \cup (q_2,0)$
- $(q_0, q_1, q_2, 1) = (q_0,1) \cup (q_1,1) \cup (q_2,1)$
- $(q_0, q_1, q_2, 2) = (q_0,2) \cup (q_1,2) \cup (q_2,2)$

(B)

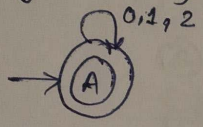
- $(q_1, q_2) \xrightarrow{\lambda} ,0 = (q_1,0) \cup (q_2,0)$
- $(q_1, q_2, 1) = (q_1,1) \cup (q_2,1)$

(C) $(q_2, 0)$

$(q_2, 1)$

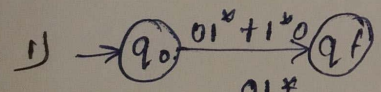


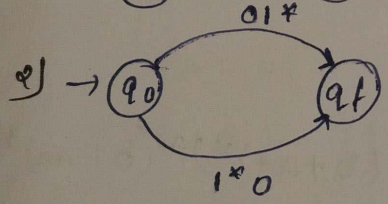
- Since B and C are not reaching to any one

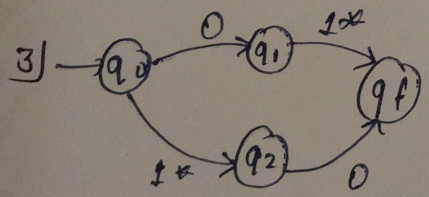


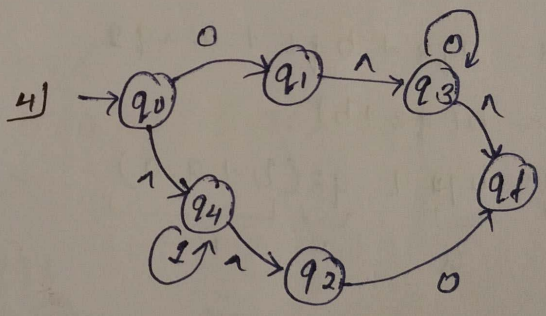
19/9/25

→ $01^* + 1^*0$

1) 

2) 

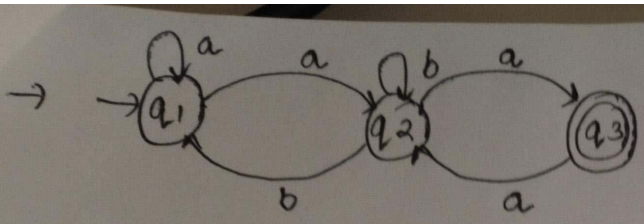
3) 

4) 

- λ- Equivalence of $q_0 = (q_0, q_1, q_2, q_3, q_4, q_f)$

" " of $q_2 = (q_2, q_f)$

20/9/25



$q_1 = a \cdot q_1 + bq_2 + \Lambda \rightarrow (1)$

$q_2 = aq_1 + bq_2 + aq_3 \rightarrow (2)$

$q_3 = aq_2 \rightarrow (3)$

$q_2 = aq_1 + bq_2 + a \cdot aq_2$

$\underbrace{q_2}_{R} = \underbrace{aq_1}_{Q} + \underbrace{q_2}_{R}\underbrace{(b + a \cdot a)}_{P}$

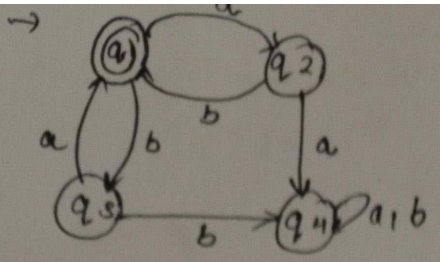
$R = QP^*$

$q_2 = aq_1 (b + a \cdot a)^*$

$q_1 = aq_1 + b(aq_1(b+a \cdot a)^*) + \Lambda$

$q_1 = \underbrace{q_1}_{R}\underbrace{(a + b(aq_1(b+a \cdot a)^*))}_{P} + \underbrace{\Lambda}_{Q}$

$q_1 = (a + b(aq_1(b+a \cdot a)^*))$



$q_1 = b.q_2 + a.q_3 + \wedge \rightarrow (1)$

$q_2 = a.q_1 \rightarrow (2)$

$q_3 = b q_1 \rightarrow (3)$

$q_4 = a.q_4 + b.q_4 + b q_3 + a.q_2 \rightarrow (4)$

$q_1 = b.q_2 + a.q_3 + \wedge$

$q_1 = b\,a.q_1 + a.b.q_1 + \wedge$

$q_1 = \overset{R}{q_1}\overset{P}{(ba + ab)} + \overset{a}{\wedge}$

$q_1 = (ba + ab)^*$

$q_2 = a.q_1$

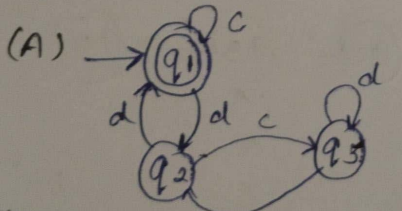
$= a(ba + ab)^*$

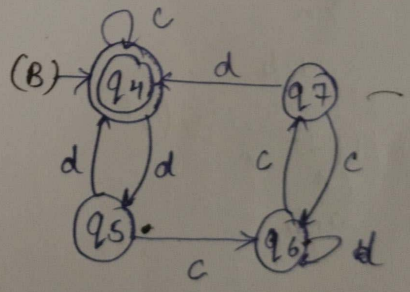
$q_3 = bq_1$

$= b(ba + ab)^*$

Equivalence b/w Two FA

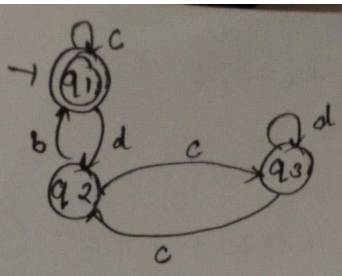
- Comparison ~~Table~~ method :- $\Sigma = \{c, d\}$
- Given with two FA.

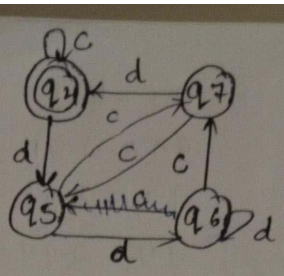




P. state	Next state	
	c	d
(q1, q4)	(q1 q4)	(q2, q5)
(q2, q5)	(q3, q6)	(q1, q4)
(q3, q6)	(q2, q7)	(q3, q6)
(q2, q7)	(q3, q6)	(q1, q4)

Hence the machine has equivalence.





P-state	Next state	
	c	d
(q1, q4)	(q1, q4)	(q2, q5)
(q2, q5)	(q3, q7)	(q1, q6) ✗

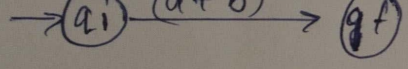
Not a equivalence.

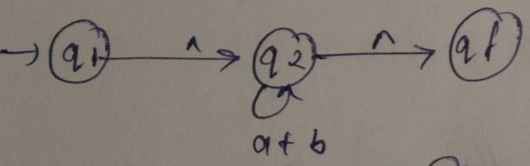
Ques Prove that $(a+b)^* = a^*(ba^*)^*$

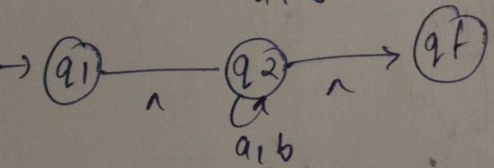
~~$(a^*b^*)^*$~~

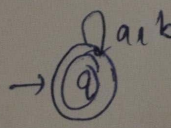
~~a^*~~

$(a+b)^*$

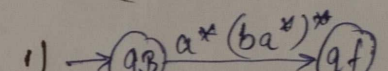
1] 

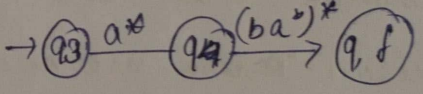
2] 

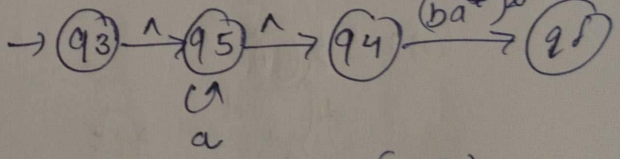
3] 

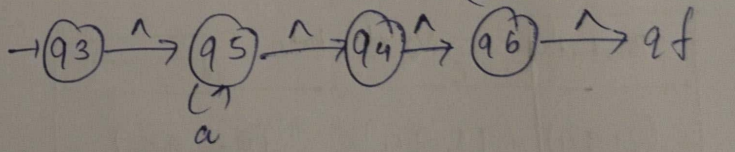


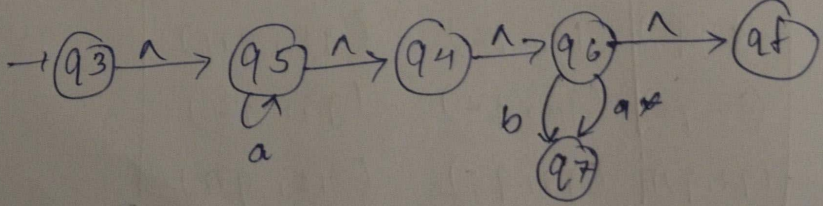
$a^*(ba^*)^*$

1] 





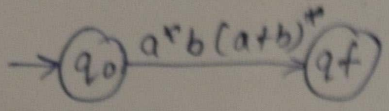


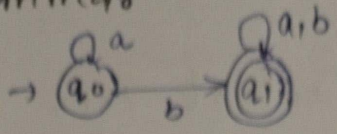


R.E to Regular

R.E to Regular Grammar

→ $P = a^*b(a+b)^*$





$S \to aS \quad A \to aa$

$S \to bA \quad A \to bA$

$S \to b \quad A \to a/b$

$L = 0^n 1^{2n}$

21/4/25

Chapter-6

6-7 → Pull down automata

Context-Free Languages

- CFL it is used to design block structure of programming language with the help of parser (Pase tree)
  G is said to be content free grammar if every production is in the form of ~~A→~~ A → α where α is element from $\alpha \in (V_N \cup E)^*$. The language generated by CFG is called as CFL.
- This languages can be derived with the help of pase tree or derivation tree.
- four tuple quantity $G = \{V_N, \Sigma, S, P\}$
  V_N = Capital (Variable are represent in single Capital letter)
  Σ = terminal
  S → Start symbol

<sign> → A

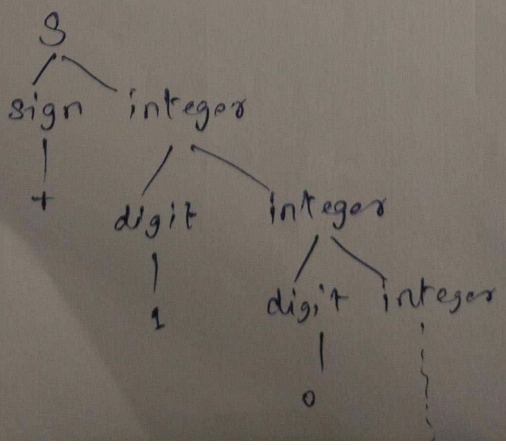
S → <sign><integer>
sign → +/-
integer → <digits><integer> / <digits>
<digits> → 0/1/2/..../9

S → AB
A → +/-
B → CB/C
C → 0/1/2/... 9

L(G) = set of all integer that can derived P.

→ Find 101

S →



Derivation Tree:- It is also called pase tree for a CFG. the tree must satisfy following condition

(i) Every vertex has a label which can be variable or terminal symbol or $\wedge$ (empty).

(ii) The root of the tree always going to be S. (start symbol).

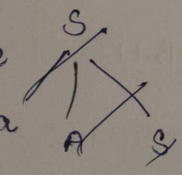
(iii) The label of internal vertex is going to be a variable.

(iv) A vertex n is a leaf if it is an element of Σ or empty.

Ques $S \to aAS|a$
$A \to SbA|SS|ba$

aabbaa. Check this it is generated or not using pase tree.

aAS
↓ ↳ variable
Terminal



$S \to aAS \to aSbAS \to aabAS \to aabbaS \to aabbaa$

$S \to a$

$aAa \to aSbAa \to aabbaa$

#leftmost Derivation Tree :-

A Derivation of the form $A \overset{*}{\to} w$ is called leftmost derivation if we apply a production only to the leftmost variable at every step.

~~Replaced with~~ Same → Right most Derivation.

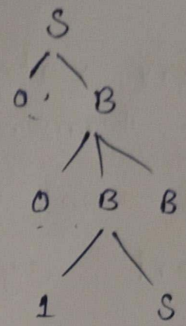
Note:- For a given language the LDT is used to generate that language but RDT is not always successful to generate that languages.

S → 0B / 1A

A → 0 / 0S / 1AA

B → 1 / 1S / 0BB

00110101 Find LDT or RDT.



~~S → 0B → 00BB → 001B~~

x ↓ 1A

↓ ~~001SB~~

Left:-

S → 0B → 00BB → 001B → 0011S → 00110B → 001101S → ~~0B~~ →

↓x 1A

↓ 001SB ↓ 0011AB

↓ 0011010B ↓ 00110101

Right:-

S → 0B → 00BB ~~→ 00B1S → 00B11A → 00B110 → 00B1S110~~

↓ 00B1 → 001S1 → 0011A1 → 00110S1

↓ 001101A1

↓ 00110101

Ambiguity in

A terminal string w which can be element of w ∈ (LG) is said to be ambigious if w has more than two derivation tree. (2 or more LDT).

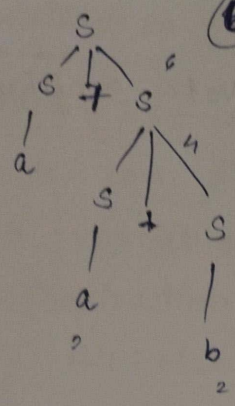
w = a + ab

S → S+S / S*S / a / b

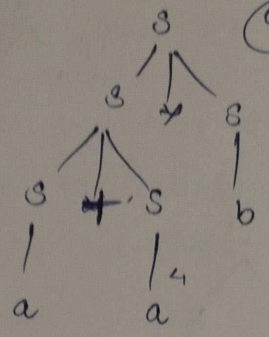
S → S+S → a+S → a+S*S → a+a*S → a+a×b

S → S*S → S+S*S → a+S*S → a+a*S → a+a*b

(6)



(8)

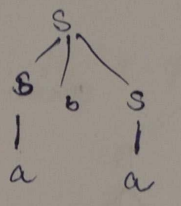


a = 2
b = 2

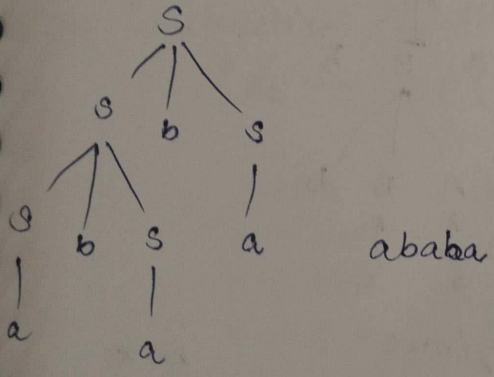
Q7/9/25

S → SbS/a . Prove that the grammar is ambigious.

S → SbS
S → a



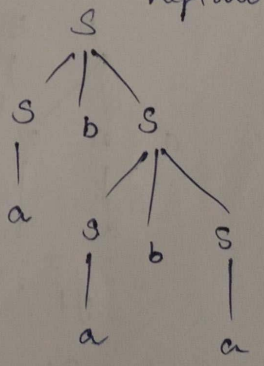
aba



ababa

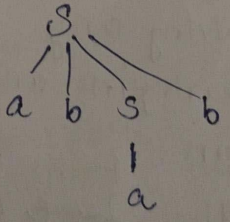
• Leftmost Tree

S → S

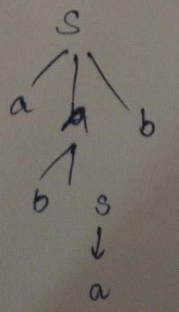


→ S → a/abSb/aAb
A → bS/aAAb

S → abSb → abab



=abab

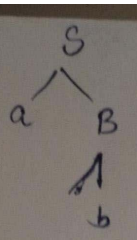


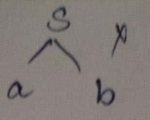
abab

• Check CFG (Type-2)

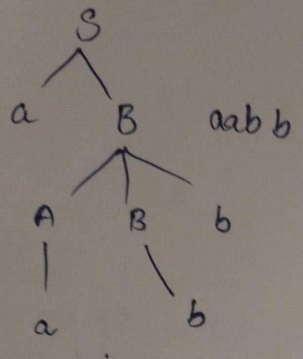
S

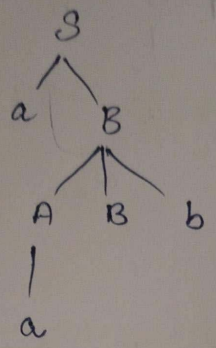
→ S → aB | ab
A → aAB | a
B → ABb | b

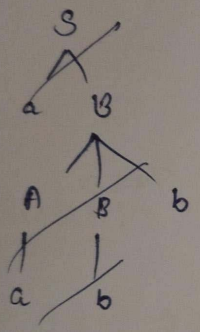


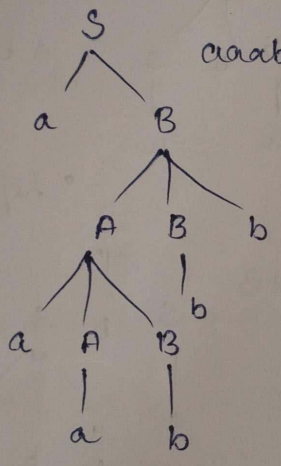


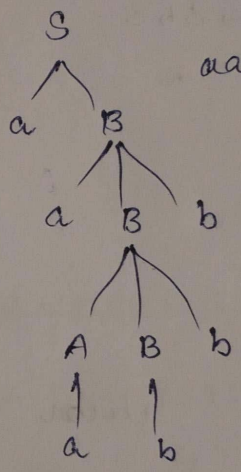
Not a ambigious. aabab











It is a ambigious.

A → ∧

Simplification of CFGs:

1. Variable not deriving any terminal string
2. Symbol not appearing in any sentential form.
3. null production
4. Production of the ~~form~~ A → B.

S → AB
A → a
B → b
B → C
E → C | ∧

S → aB
S → (ab) sentential form

Construction of reduced grammar

Theorem 1 :- G_1 is a CFG such that $L(G) \neq \phi$ which can find an equivalent grammar G_1' such that each variable in G_1' derives some terminal string.

(a) Construction of V_N'

W_1 = element of V_N such that there is a production $A \to w$, where w in an element of Σ^*

$\{A \in V_N \mid$ there is a production $A \to w$, where $w \in \Sigma^*\}$

$W_2 = W_1 \cup \{A \in V_N \mid$ there exists a production $A \to \alpha$ with $\alpha \in (\Sigma \cup W_i)^*\}$

(b) Construction of P'

$P' = \{A_1 \to \alpha \mid \alpha \in (V_N' \cup \Sigma)^*\}$

$S \to AB$
$A \to a$ ✓
$B \to b$ ✓
$B \to C$ ✗
$E \to c$ ✓

$W_1 = \{A, B, E\}$
$W_2 = \{A, B, E\} \cup \{S\}$
$W_2 = \{A, B, E, S\}$
$W_3 = W_2$

$V_N' = \{S, A, B, E\}$

P'

$S \to AB$
$A \to a$
$B \to b$
$E \to c$

$S \to aAB$
$A \to aAla$
$B \to bBlb$
$B \to clb$
$C \to D$ ~~B~~

Find V_N' & P'

$W_1 = \{$ ~~S A B C~~ AB $\}$
$W_2 = \{AB\} \cup \{S\}$
$W_2 = \{S, A, B\}$
$W_3 = W_2$
$V_N' = \{S, A, B\}$

P'

$S \to aAB$
$A \to aAla$
$B \to bBlb$