29/1

MODULE : 2

REGULAR EXPRESSION

- Regular expression is a string using a combination of ~~alpha~~ symbols from an alphabet Σ along with paranthesis and three operators : +, *, .

Here, + : union of two diff languages
. : concatenation of strings
* : kleen closure

- A regular expression can be recursively defined as follows

Primitive Regular ∈

1. ϕ is a regular expression denoting the regular language { }
2. ϵ is the regular expression denoting the regular language {e}
3. 'a' is a regular expression representing the regular language {a}
4. Let r_1 be a regular expression representing the language Lr_1 and r_2 be a regular expression representing the regular language Lr_2, then
   - $r_1 + r_2$ is a regular expression denoting $Lr_1 \cup Lr_2$
   - $r_1 \cdot r_2$ is a regular expression denoting $Lr_1 \cdot Lr_2$
   - $r_1 *$ is a regular expression denoting Lr_1*
5. Any strings generated by finite number of applications of rules $1 \rightarrow 4$ are regular expressions.

Q. Write the regular expression for the following languages.

1. Binary strings : $(0+1)^*$
2. All binary strings starting with 1 : $1 \cdot (0+1)^*$
3. All binary strings that starts and ends with same symbol : $1 \cdot (0+1)^* \cdot 1 + 0 \cdot (0+1)^* \cdot 0$
4. All strings over $\{a, b\}$ that ends with 'abb'
   $(a+b)^* \cdot a \cdot b \cdot b$
5. All strings that have a substring 'bab'
   $(a+b)^* \cdot b \cdot a \cdot b \cdot (a+b)^*$
6. Even number of a's followed by odd no. of b's
   $(aa)^* (bb)^* b$
7. All strings of even length : $((a+b)(a+b))^*$
8. strings with even no. of a's and any no. of b's
   $(b^* a b^* a b^*)^* + b^*$
9. Any strings of $\{0,1\}$ where 10th symbol from right end is 1 : $(0+1)^* \cdot 1 \cdot (0+1)^9$

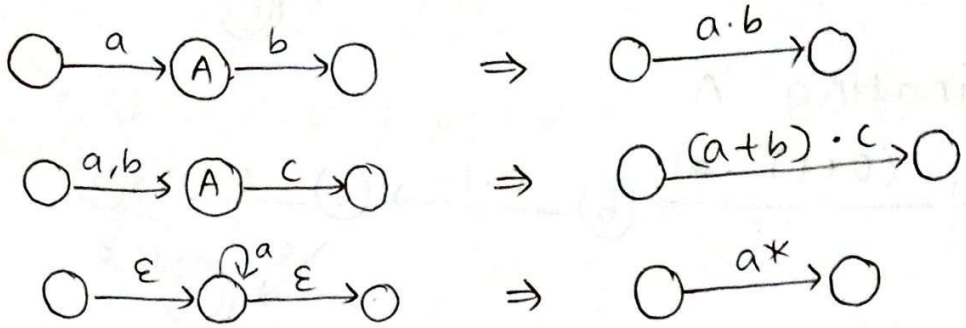
31/1 Converting Finite Automata to Regular Expression

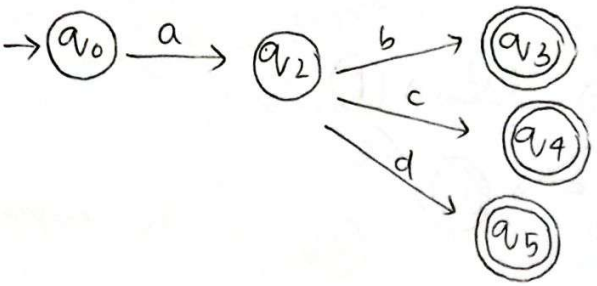
- Also known as State elimination

Step 1 : If there exists any incoming transitions to the initial state of given finite automata, then add a new initial state q_i with an ϵ-transition to the given initial state.

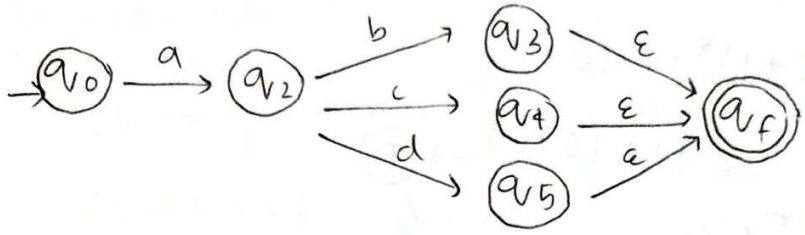
Step 2 : If there exists any outgoing transitions from final state or if there are multiple final states, add a new final state say q_f with ϵ-transitions from all final states.

Step 3 : Eliminate states except q_i and q_f one by one.

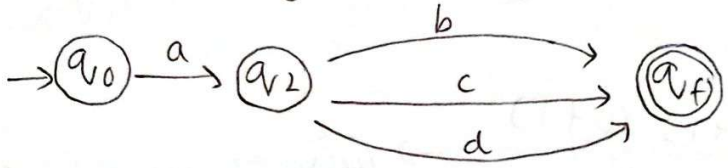


Q.1) 

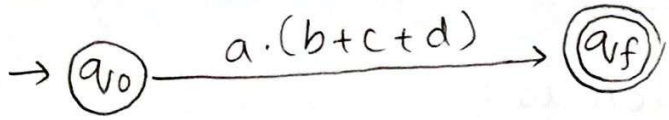
→ (Multiple final states)

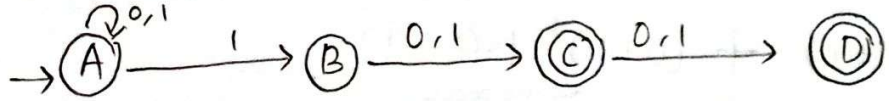


(eliminating q_3, q_4 and q_5) [do step by step]*

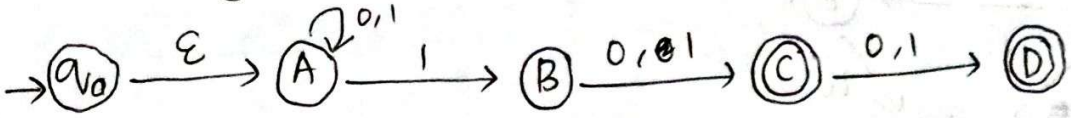


(eliminating q_2)

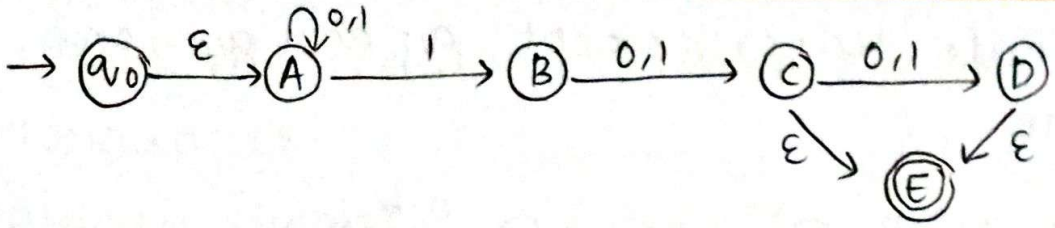


Q.2) 

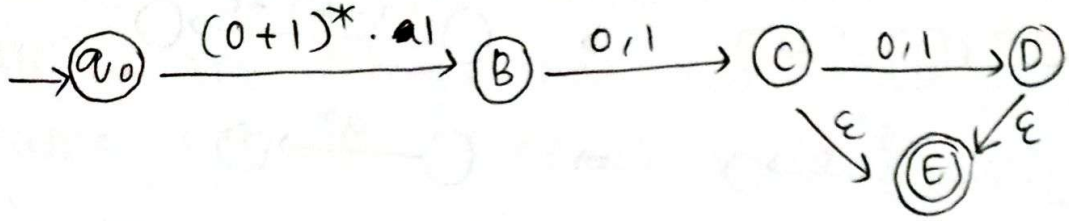
ans :- • incoming transition state to 'A'



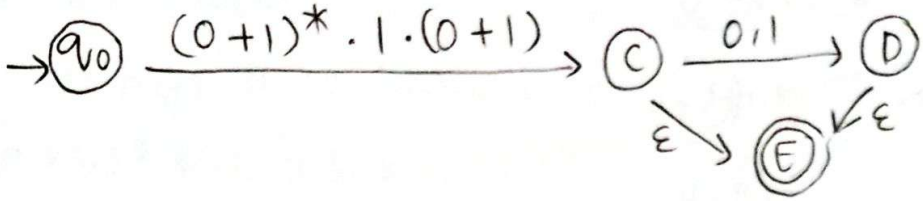
• multiple final states



- Eliminating 'A'



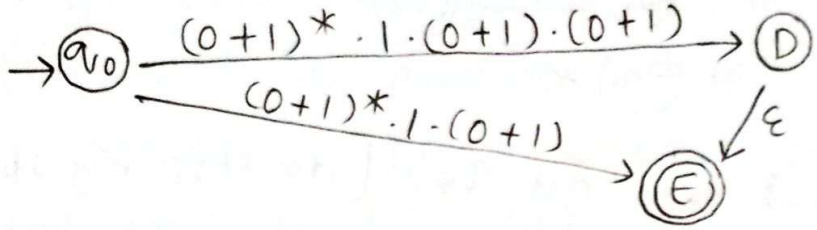
- Eliminating 'B'



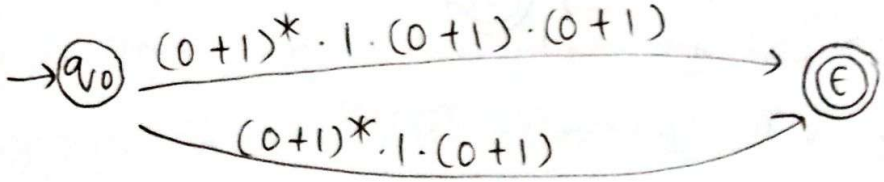
- Eliminating 'C'

q_0 to D : $(0+1)^* \cdot 1 \cdot (0+1) \cdot (0+1)$

q_0 to E : $(0+1)^* \cdot 1 \cdot (0+1)$



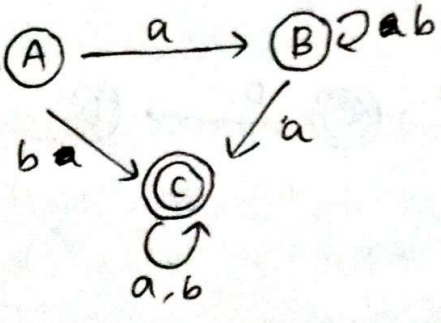
- Eliminating 'D'



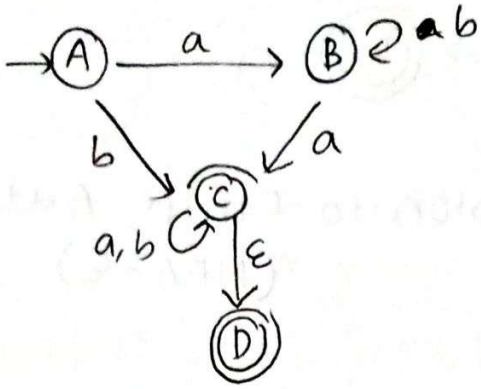
Hence Regular expression is :

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

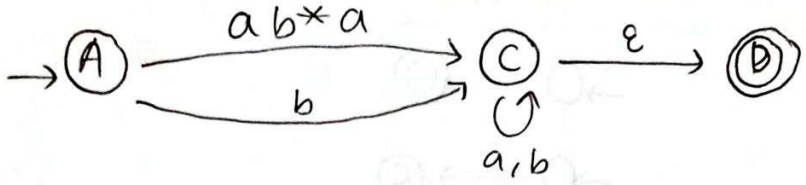
Q.3



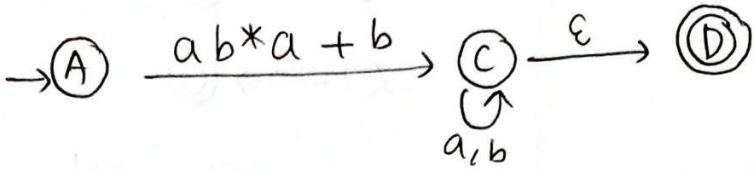
- Outgoing transitions exists



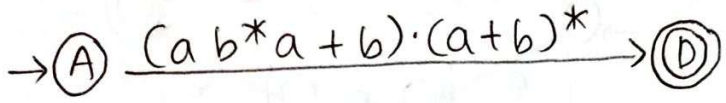
- Eliminating 'B'



~~Elim~~ or :

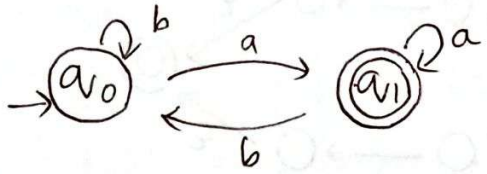


- Eliminating 'C'

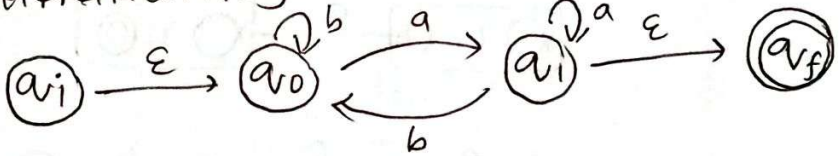


RE : $(ab^*a+b)(a+b)^*$

Q-4



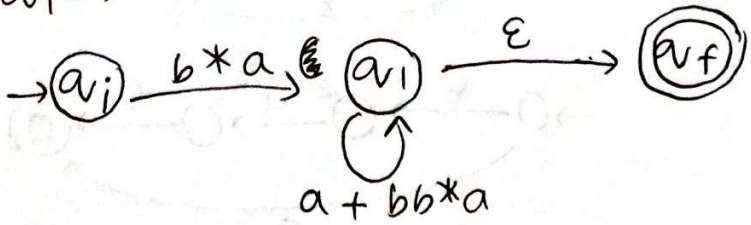
ans: • ~~Eliminating~~ Incoming and outgoing exists



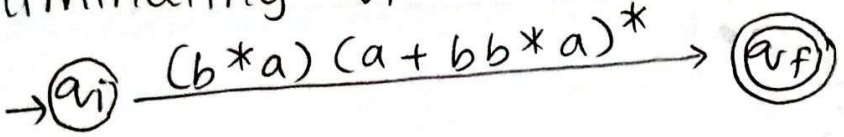
- Eliminating a_0

$a_i \rightarrow a_1 : b^*a$

$a_1 \rightarrow a_1 : bb^*a$

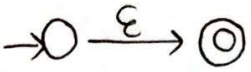
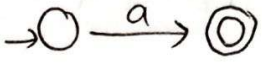
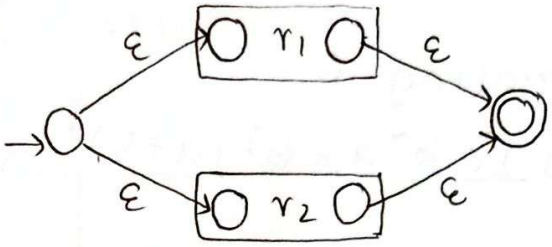
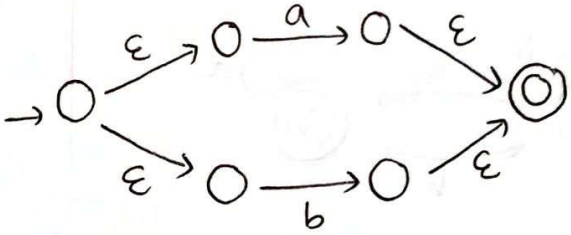
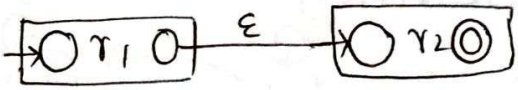
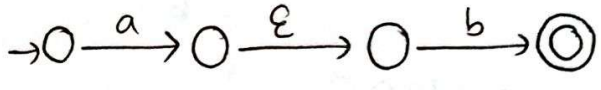
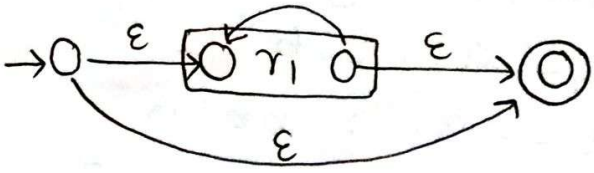
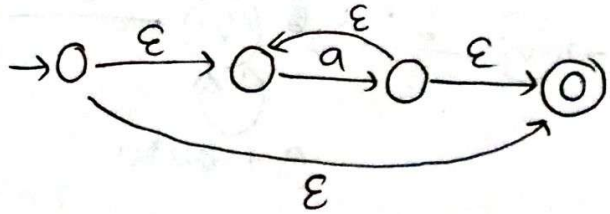


- Eliminating q_1



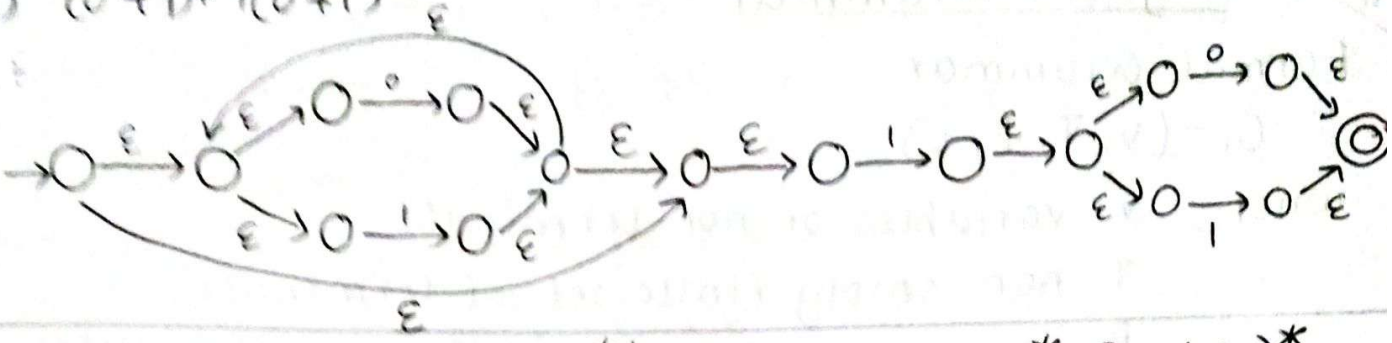
Converting Regular Expression to Finite Automata (NFA-ε)

- Thomson's Construction

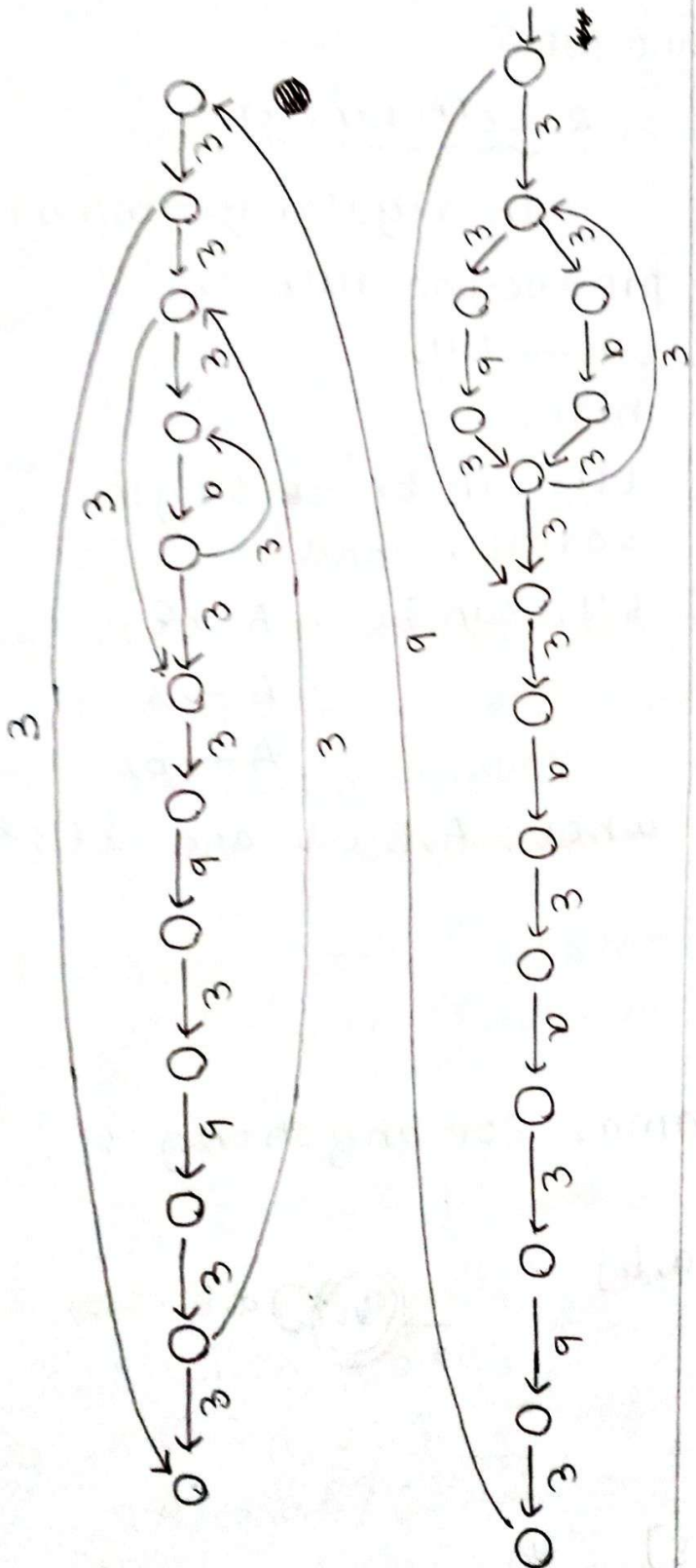
RE	F.A
1. ϕ	
2. ε	
3. a	
4. $r_1 + r_2$	
eg: $a+b$	
5. $r_1 \cdot r_2$	
eg: $a \cdot b$	
6. r_1^*	
eg: a^*	

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

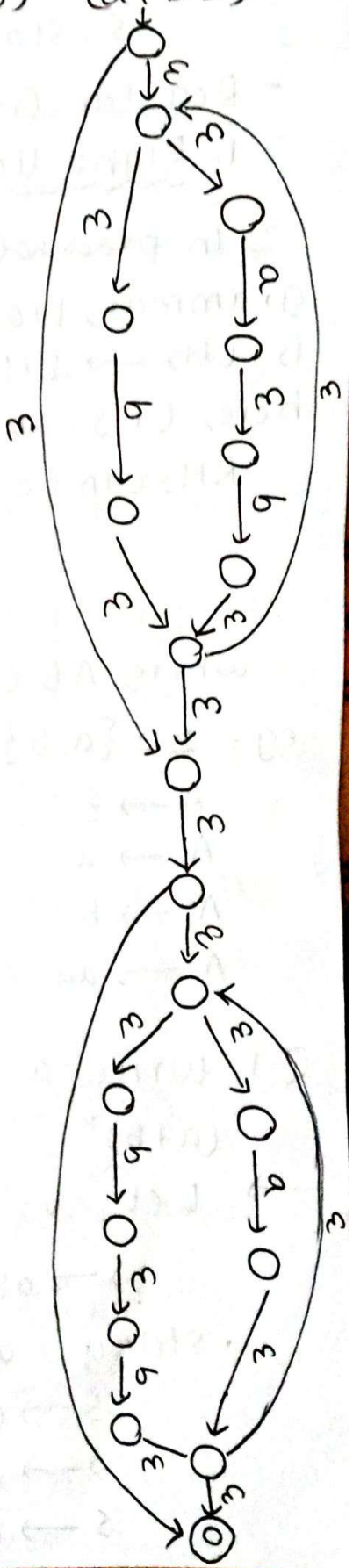
ans :



Q.2) $(a+b)^* aabb (a^* bb)^*$



Q.3) $(ab+b)^* (a+bb)^*$



1/8 # Regular Grammar

Formal Grammar

$G_1 = (V, T, P, S)$

where, V: variables or non-terminals
T: non-empty finite set of terminals
P: non-empty finite set of production rules
S: start symbol taken from V ($S \in V$)

- Regular Grammar is of two types:

1. Right linear

• In ~~production~~ regular grammar, production rule is LHS $\longrightarrow$ RHS.
here, LHS: a single variable
RHS can be • $A \to \varepsilon$
• $A \to x$
• $A \to xB$

where, $A, B \in V$, $x \in T^*$

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

$A \longrightarrow \varepsilon$ $\quad$ $A \longrightarrow aB$
$A \longrightarrow a$ $\quad$ $A \longrightarrow bA$
$A \longrightarrow b$ $\quad$ $A \longrightarrow aaB$
$A \longrightarrow aa$

2. Left Linear

• In regular grammar, production rule is
LHS $\to$ RHS.
here,
LHS can be a single variable and
RHS can be • $A \to \varepsilon$
• $A \to x$
• $A \to Bx$

where, $A, B \in V$ and $x \in T^*$

Q.1 Write a regular grammar for any string of $(a+b)^*$

$\to$ Let $V = \{S\}$, $T = \{a, b\}$

$S \longrightarrow aS/bS/\varepsilon$

• string : abab

$S \longrightarrow aS$
$\longrightarrow abS$ $\quad [S \to bS]$
$\longrightarrow abaS$ $\quad [S \to aS]$

$\rightarrow$ ababS [S $\rightarrow$ bS]

$\rightarrow$ abab [S $\rightarrow \varepsilon$]

$\Rightarrow (a+b)^+$

here, V = {S}, T = {a, b}

here S cannot be S $\rightarrow \varepsilon$

$\therefore$ S $\rightarrow$ aS/bS/a/b

for string abab :

S $\rightarrow$ aS

S $\rightarrow$ abS [S $\rightarrow$ bS]

S $\rightarrow$ abaS [S $\rightarrow$ aS]

S $\rightarrow$ abab [S $\rightarrow$ b]

} string derivation

5/8

Q.2 Write a regular grammar for strings using {x, y} that starts with y.

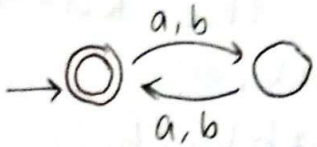
ans: S $\rightarrow$ ~~yS~~ yA

A $\rightarrow$ xA/yA/ε

Q3 Write a regular grammar for generating all even length string using {a, b}

ans: S $\rightarrow$ aA/bA/ε

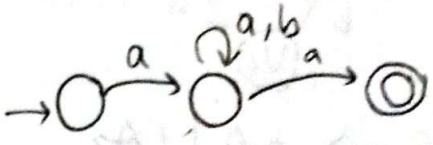
A $\rightarrow$ aS/bS ~~ε~~



Q.4 write regular grammar for generating all strings with {a, b} that start and end with 'a'

ans: S $\rightarrow$ aA/~~ε~~a

A $\rightarrow$ aA/bA/a~~ε~~

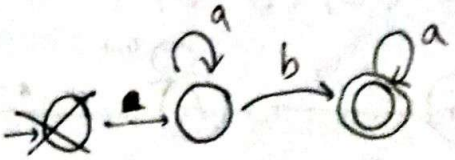


IMP

Q.5 Write a regular grammar for the regular expression (a*ba*)

ans: S $\rightarrow$ aS/bA

A $\rightarrow$ aA/ε



Equivalence of Regular Grammar & F.A

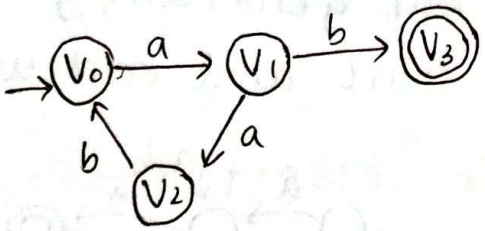
Converting R.G to F.A : Right Linear.

1. Creating states for each variable
2. Initial state of F.A will be the state corresponding to the start symbol S
3. $\Sigma = T$ (language is same)
4. For every production of the form $V_i \rightarrow aV_j$, add a transition $\delta(V_i, a) = V_j$
5. For every production of the form $V_i \rightarrow a$, add a transition $\delta(V_i, a) = V_f$ where V_f : final state.
6. For every production rule of the form $V_i \rightarrow \epsilon$, make V_i a final state.

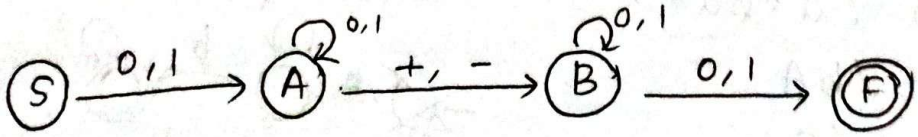
- For left linear grammar, follow all the above steps. Then reverse the directions of every transitions and swap initial and final states.
- The obtained will be NFA.

Q. Convert the RG → FA.

1. $V_0 \rightarrow aV_1$
   $V_1 \rightarrow abV_0 / b$



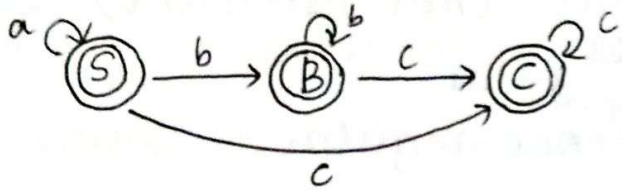
7/8
2. $S \rightarrow 0A \mid 1A$
   $A \rightarrow 0A / 1A / +B / -B$
   $B \rightarrow 0B / 1B / 0 / 1$



3. $S \rightarrow aS/bB/cC/\epsilon$

$B \rightarrow bB/cC/\epsilon$

$C \rightarrow cC/\epsilon$

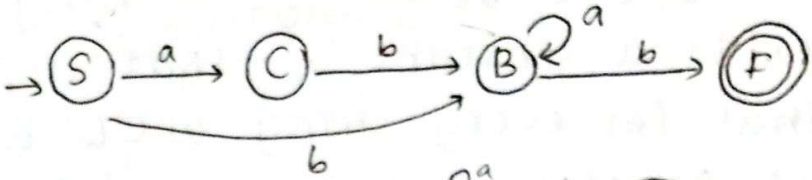


4. $S \rightarrow Ca/Bb$

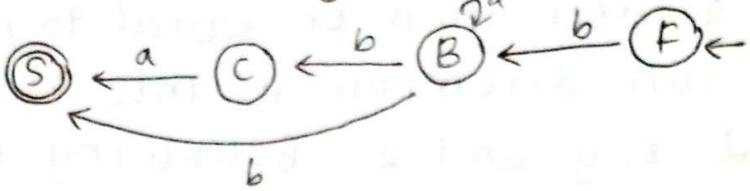
$C \rightarrow Bb$

$B \rightarrow Ba/b$

Right lineal :



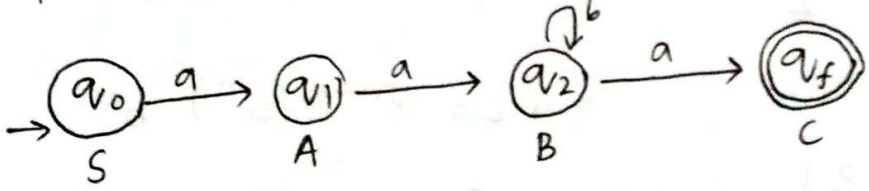
Left lineal :



Conversion of FA to RG

1. Each state in FA becomes a V (variable) in RG
2. Start symbol, S = the variable corresponding to the initial state.
3. T is same as Σ of FA
4. For every transition of the form $\delta(q_i, a) = q_j$ add a production $Q_i \rightarrow aQ_j$
5. For every final state of given FA, add a production $Q_i \rightarrow \epsilon$.

Q.1)



$S \rightarrow aA$

$A \rightarrow aB$

$B \rightarrow bB/aC$

Neccesary Condition for RL

- Pumping Lemma : ~~gives a~~ A gives a neccesary condition that (not sufficient) for a language to be not regular
- to prove a language is not regular.
- It is proof by contradiction.

Statement of pumping lemma:

Let L be a regular language, then there exists a constant n that depends on L such that for every string $w \in L$ with length of w greater than or equal to n ($|w| \geq n$), we can decompose w into three parts ~~x,y,z~~ and x, y and z, following the condition:

1. y cannot be NULL ($y \neq \varepsilon$)
2. $|xy|$ must be within the limit n
3. For all i, $i \geq 0$, xy^iz is in L

Q.1) Prove that $L = \{a^n b^n, n \geq 0\}$ is not regular.

→ Assuming that L is regular. Let m be a constant. And w is a string;

$w \in L$, $|w| \geq m$.

Considering, $w = a^m b^m$ $\quad |w| = 2m > m$

ie, $\overbrace{aa\ldots a}^{m}\overbrace{bb\ldots b}^{m}$

$\underbrace{\quad\quad}_{x\ y}\ \underbrace{\quad\quad}_{z}$

$X = a^{m-k}$

$Y = a^k$, $k \geq 1$

$Z = b^m$

$xy^iz = a^{m-k}(a^k)^i b^m$

when, $i = 0 \Rightarrow a^{m-k} a^0 b^m$

ie, $a^{m-k} b^m \notin L$

so our assumption is wrong and by contradict^n

L is not regular.

Q.2) $L = \{ ww^R, w \in (a,b)^* \}$

(w^R : reverse of w , ww^R : even palindrome)

→ Assume L is a regular language and m be the constant.

Let, $w \in L$, $|w| \geq m$

$w = a^m b^m b^m a^m$ $\quad |w| = 4m > m$

$\overbrace{aa\ldots a}^{m}\overbrace{aa\ldots a}^{m}\overbrace{bb\ldots b}^{m}\overbrace{bb\ldots b}^{m}$

$\underbrace{\quad\quad}_{x\ y}\quad\underbrace{\quad\quad\quad\quad}_{z}$

$x = a^{m-k}$

$y = a^k$, $k \geq 1$

$z = a^m b^m b^m$

$xy^iz = a^{m-k}(a^k)^i a^m b^m b^m$

$i=0 \Rightarrow xy^iz = a^{m-k} a^m b^m b^m$

ie, $a^{m-k} a^m b^m b^m \notin L$

Our assumption is false. Thus by contradiction, L is not regular.

Q.3) $L = \{ a^{n!}, n \geq 1 \}$. PT L is not regular.

→ Assume L is regular and m be the constant

Let $w \in L$, $|w| \geq m$

$w = a^{m!}$, $|w| = m! > m$

$\overbrace{aa\ldots\ldots a}^{m!}$

$x = a^{m-k}$

$y = a^k$, $k \geq 1$

$z = a^{m!-m}$

$$xy^iz = a^{m-k}(a^k)^i a^{m!-m}$$
$$= a^{m-k+ki+m!-m}$$
$$= a^{m!+ki-k}$$

when $i = 2 \Rightarrow a^{m!+k}$

Here, $m!+k$ must be a factorial of a number

- $m!$ if $k=0$, not possible
- $(m+1)! = m!(m+1)$
  $= m!m + m!$

  we know, $m!+k \leq m!+m < m!m+m!$

  Hence it is also not possible.

Q.4) $L = \{a^i, 'i' \text{ is a prime number}\}$

→ Assume L is regular and m be constant.

$w \in L$, $|w| \geq m$.

$w = a^m$, m is a prime number

$w \rightarrow |xyz| = m$

xy^iz, $i = m+1$

ie, $|xy^{m+1}z| = |xy^m yz|$
$= |xyz| + y^m$
$= m + m|y|$
$= m(1+|y|)$

for $m(1+|y|)$ to be prime, ~~either~~ $y=0$ which is not possible. Hence assumption is false. Thus by contradiction, L is not regular.

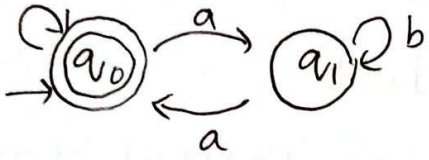
12/8 IMP

Closure Properties of RL

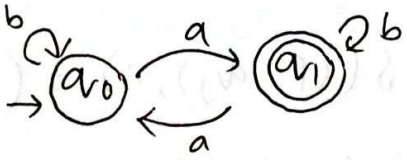
I Set operations

1. Union ($L_1 \cup L_2$)
2. Intersection ($L_1 \cap L_2$)
3. Difference ($L_1 - L_2$)
4. compliment ($\bar{L}_1$)
5. concatenation ($L_1 \cdot L_2$)
6. kleen closure (L_1^*)

Compliment

L: 

(even no. of a's)

$\bar{L}$: 

(odd no. of a's)

- If L is a regular language such that & $M(L) = (Q, \Sigma, d, q_0, F)$, then there exists $\bar{L}$ which is also regular, ie $\bar{L} = (Q, \Sigma, d, q_0, Q - F)$.

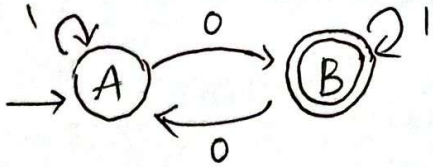
-

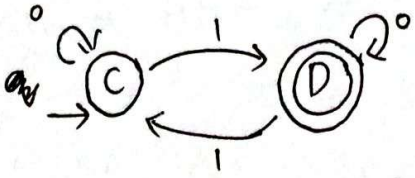
Intersection

- If L_1 and L_2 are two regular languages, then $L_1 \cap L_2$ is also regular.

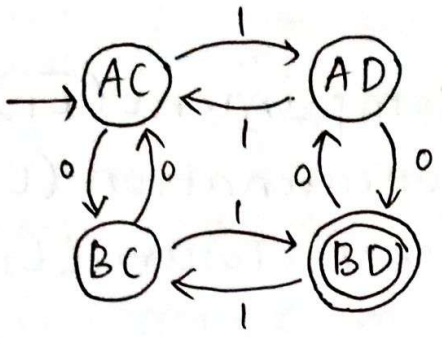
$L_1 = M_1 = (P, \Sigma, \delta_1, P_0, F_1)$
$L_2 = M_2 = (Q, \Sigma, \delta_2, q_0, F_2)$
} $L_1 \cap L_2 = (P \times Q, \Sigma, \delta, (P_0, q_0), \hat{F})$

eg: Let $\Sigma = \{0, 1\}$, L_1 : odd no. of 0s and L_2 : odd no. of 1s.

L_1: 

L_2: 

- States : $\{A, B\} \times \{C, D\} = \{AC, AD, BC, BD\}$



$\delta(AC, 0) = (\delta(A,0), \delta(C,0)) = (B,C)$
$\delta(AC, 1) = (AD)$
$\delta(AD, 0) = (B,D)$
$\delta(AD, 1) = (A,C)$
$\delta(BD, 0) = (AD)$, $\delta(BD, 1) = (BC)$
$\delta(BC, 0) = (AC)$, $\delta(BC, 1) = (BD)$

ie, $\delta((P_i, q_j), a) = (\delta(P_i, a), \delta(q_j, a))$

- Initial state : ~~combinat~~ contains initial state of both the languages.
- Final states : first digit is final state of L_1 and second digit is final state of L_2.

$$\hat{F} = \{(P_i, q_j) / P_i \in F_1, q_j \in F_2\}$$

Difference

- $L_1 - L_2 = L_1 \cap \bar{L}_2$

II Other operations

1. Reversal
2. Homomorphism
3. Inverse homomorphism.

Reversal

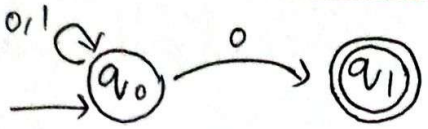
$L \in RL$, L^R is its reverse

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

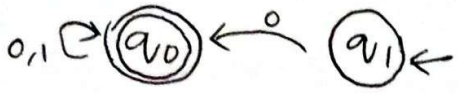
$L^R =$

eg: $\Sigma = \{0,1\}$

$L_1 = \{$ string ends with 0 $\}$



L^R = { string starts with 0 }



1. Swap initial and final states
2. Reverse every transitions.

14/8

Homomorphism

- It is a mapping function
- Considering Σ, Δ be two alphabets then, $\boxed{h: \Sigma \rightarrow \Delta^*}$

eg: $\Sigma = \{a, b\}$ $\Delta = \{0, 1\}$

$h(a) = 01$

$h(b) = 10$

then, string, s=abab : $h(s) = 01100110$

language, $L = \{ab, aba\}$ $h(L) = \{0110, 011001\}$

- If L is regular, h(L) is also regular.

eg: $\Sigma = \{a, b\}$ $\Delta = \{b, c, d\}$

$h(a) = dbcc$

$h(b) = bdc$

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

$h(L) = (dbcc + (bdc)^*)(dbccdbcc)^*$

Inverse Homomorphism

- Given a homomorsphism h which is defined from $\Sigma \rightarrow \Delta^*$ and a language L which is a subset of Δ^*, then $h^{-1}(L)$ is defined as,

$\boxed{h^{-1}(L) = \{ w \in \Sigma^*, h(w) \in L \}}$

eg: $\Sigma = \{x, y\}$, $\Delta = \{a, b\}$

Given, $h : \Sigma \rightarrow \Delta^*$ such that

$h(x) = ab$ and $h(y) = b$

$L \Rightarrow (ab)^*$

→ $L = \{\varepsilon, ab, abab \ldots\}$

$h^{-1}(L) \subseteq \Sigma^*$

- $\varepsilon \Rightarrow h(\varepsilon) = \varepsilon$ ✓ ---> belongs to L
- $x \Rightarrow h(x) = ab$
- $y \Rightarrow h(y) = b$ × ---> doesnot belong to L
- $xx \Rightarrow h(xx) = abab$
- $xy \Rightarrow h(xy) = abb$ ×
- $yx \Rightarrow h(yx) = bab$ ×
- $yy \Rightarrow h(yy) = bb$ ×
- $xxx \Rightarrow h(xxx) = ababab$ ✓

Hence, $h^{-1}(L) = \{\varepsilon, x, xx, xxx \ldots\}$

$h^{-1}(L) \Rightarrow x^*$

Type 2 Grammar

CONTEXT FREE GRAMMAR (CFG) / CFL

- P : LHS → RHS

$\boxed{V \rightarrow (VUT)^*}$

- LHS : single capital letter

RHS : can be any combination of variables and terminals.

eg: $S \rightarrow \varepsilon$ $S \rightarrow aaBa$

$S \rightarrow ab$

$S \rightarrow A$

- All regular grammar is CFG but not vice versa.

-

Q. $L = \{ a^n b^n , n \geq 0 \}$

$S \rightarrow aSb / \varepsilon$ ...aaabbb...

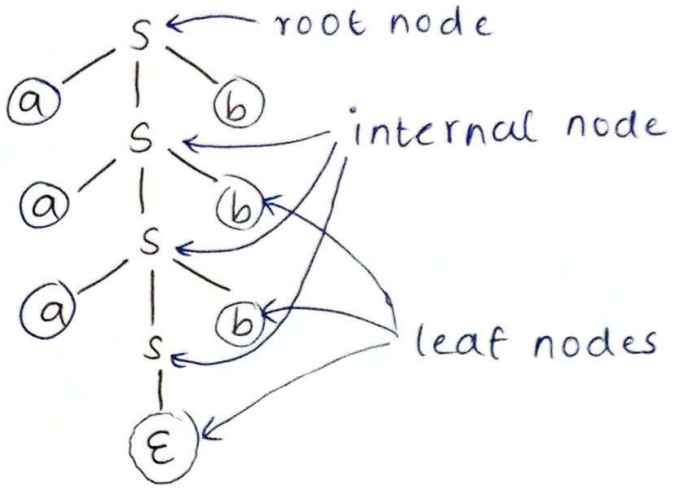
Derivation of :

- $S \rightarrow aSb \xrightarrow{S \rightarrow \varepsilon} ab$
- $S \rightarrow aSb \xrightarrow{S \rightarrow aSb} aaSbb \xrightarrow{S \rightarrow aSb} aaaSbbb \xrightarrow{S \rightarrow \varepsilon} aaabbb$

Derive the string a^3b^3:

$S \rightarrow aSb \xrightarrow{S \rightarrow aSb} aaSbb \xrightarrow{S \rightarrow aSb} aaaSbbb \xrightarrow{S \rightarrow \varepsilon} aaabbb$

Parse tree of a^3b^3:

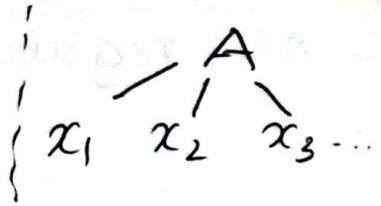


Parse Trees

- Parse trees of a grammar G are trees with following properties:

1. Root will be the start symbol
2. Each ~~node~~ internal node will be variable V
3. Each leaf node is labelled by a variable or terminal or ε. If ε is a node, that will be the only child of that parent.
4. If A is an internal node and its children $x_1, x_2, \ldots x_n$, then $A \rightarrow x_1 x_2 \ldots x_n$ will

be a production in grammar G



Yield of a Parse ~~Field~~ Tree

- word generated from a grammar
- Read from left to right

Sentential form

- can include variables and terminals
- intermediate ~~of~~ stage of derivation

eg:-