Module - II
Regular Languages

Topic 1:

Regular Set: A Special set of words over input alphabet Σ is called a regular set, if it follows the following properties recursively.

(i) Every finite set of words over Σ including ϵ is a regular set

(ii) If R_1 and R_2 are regular sets over Σ, then $R_1 \cup R_2$, $R_1 R_2$ & also R_1^* are Regular sets

Regular Set can be ~~applico~~ obtained by finite application of 1 and 2

Example ①: Let $\Sigma = \{1\}$ then the set of strings $\{1, 11, 111 ---\}$ is a regular Set

Example ②: Let $\Sigma = \{a, b\}$ then the Set of strings $\{ab, ba\}$ is a regular set

So, In short we can say Regular sets are the sets which are accepted by finite automata

Example ③: Let $\Sigma = \{0, 1, 2\}$ then the set of strings that contain Even no. of 0's or Even no. of 1's or Even no. of 2's is a regular set.

Yes, it is a regular set, Even no. of 0's or Even no. of 1's or Even no. of 2's can be obtained by repeating the operations like union, containation and Kleen closure.

Topic 2:

Regular Expression: Let Σ be the alphabet. The RE over Σ and the sets that they denote are defined recursively as follows.

① Any symbols from Σ, $\{\emptyset\}$ and ϵ are RE

null

'0' length

(ii) If R_1 & R_2 are RE, then R_1+R_2, R_1R_2 is RE that denote R_1 Or R_1R_2 and R_1^* respectively

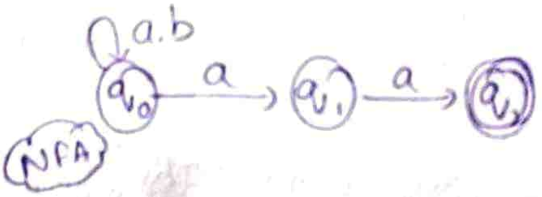
(iii) A string is a RE if it can be derived from Primitive RE by finite application of 2.

→ RE are means to represent certain sets of strings in some algebraic manner.

→ RE described in the language accepted by FA.

__Example:__ {Set of strings of a's and b's ending with aa}

L = {aa, aaa, baa, bbaa, babaa ...} → Regular set.



$(a+b)^*aa$.
↓ Algebraic Expression.
Regular Expression.

__Examples:__ Design a RE over {1} that accepts all string of any length

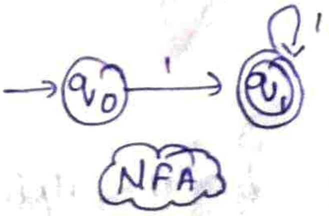
The string of the language are

__Sol:__ ε
1
11
111
1111
-- ...

RE = $(1)^*$

② Design a RE over {1} that accepts all the string of length @least 1

Sol:
1
11
111

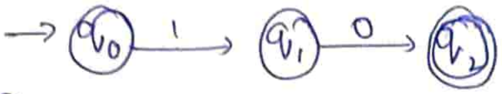


Positive Klean closure

RE = $(1)^+$

③ Design a R.E over {0,1} that accepts string 10

sol: {10, 010, 110, 0100, 1101 ---}



10 ⇒ Concatenation operation

(Q) Design RE over {0, 1} that accepts {010, 111, 100}

any one.

sol:- RE = 010 + 111 + 100

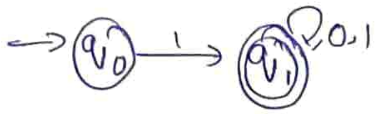
010 OR 111 OR 100

'+' Union Operations

In RE we use '+' Symbol

(Q) Design a RE over {1} that starts with 1

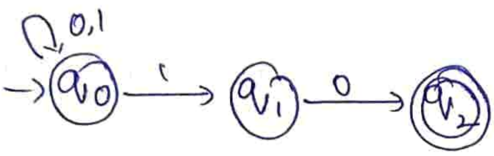
sol: 1, 10, 11, 101 --- 110 ---



$1(0+1)^* = R\cdot E$

(Q) Design a R.E that Ends with 10

sol: 10, 010, 110 ---- 10



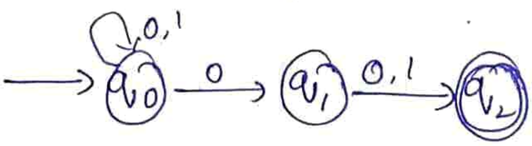
$(0+1)^* 10 \Rightarrow RE$

(Q) Design a R.E over Σ such that automata accept strings in which second character from right is 0

sol: 01, 00 ----

00(0|1)

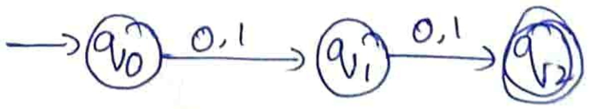
10(0|1)



$(0+1)^* 0 (0+1) \Rightarrow RE$

(Q) Design a RE over Σ such that automata accepts string length Exactly 2

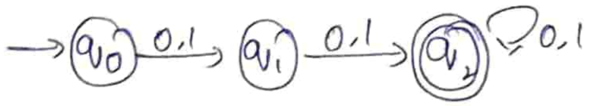
sol: 00, 11, 01, 10,



$(0+1) \cdot (0+1) \Rightarrow RE$

(Q) Design a RE over {0, 1} such that the automata accepts string length atleast 2 (3)

sol: 01, 10, 00, 01, 000, 111 - - - -



$(0+1)(0+1)(0+1)^* \Rightarrow RE$

(Q) Design a RE over {0, 1} such that the automata accept string length atmost 2

sol: ϵ, 0, 1, 00, 01, 10, 00 ~~or~~

$(\epsilon+0+1)(\epsilon+0+1) \Rightarrow RE$

Topic 3 : Identity rules.

To Simplify Regular Expression. If p, q, r are RE then.

$\phi \Rightarrow$ null set $\quad \epsilon \Rightarrow$ null string.

Rule 1 : $\phi + r = r$. Rule 2 : $\phi \cdot r = r \cdot \phi = \phi$

Rule 3: $\epsilon \cdot r = \epsilon \cdot r = r$ Rule 4 : $r + r = r$. Rule 5: $r^* \cdot r^* = r^*$

Rule 6 = $r^* r = r \cdot r^* = r^+$

Rule 7 : $(r^*)^* = r^*$. Rule 8: $\epsilon + r r^* = \epsilon + r^* r = r^*$

Rule 9 : $(p \cdot q)^* p = p(q p)^*$

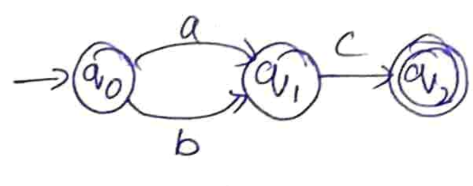
Topic 4 : Construct a finite automata for a given Regular Expression

① ba^*b

sol: $L = \{bb, bab, baab ---\}$

② $(a+b)c$ (OR) $\nearrow$ +

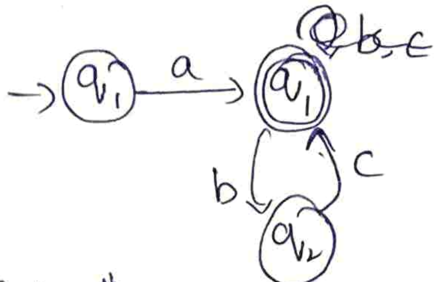
sol:



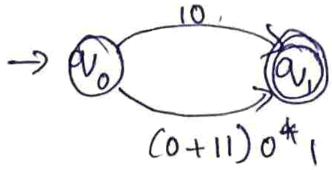
$L = \{ac, bc\}$

③ $a(bc)^*$

sol: $L = \{ a, abc, abcbc, \ldots \}$

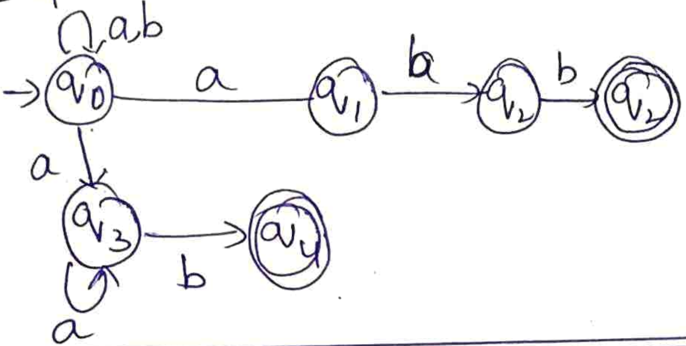


⑤ $10 + (0+11)0^*1$



④ $(a|b)^* (abb | a^+b)$

sol: / = +



$a^+ = \{a, aa, aaa \ldots\}$

$a^* = \{\epsilon, a, aa, aaa \ldots\}$

1st May

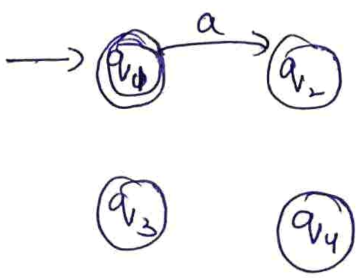
Rough

G0, G1, G2, G3, G4, G7, G8, G9, H0, H1, H2, H3, H8, H7, J0, J2, J3, K2, K6, K7, K8

L1, L3, L4, L5, L6, L7, M1, M12, M3, M4, M6, M8, M9, N0, N4, N7, N8, N9, P1, P2, P3, P5, P6, P8

Le, 14, 15, 16, 17, 18, 19

Topic : 5 : Conversion of finite automata to Regular Expression



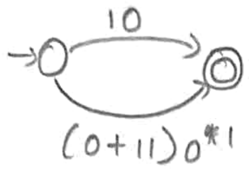
In next page

Conversion of Regular Expression to finite automata

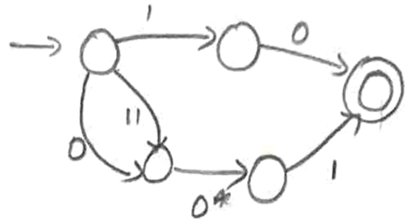
① Convert the following RE to its Equivalent finite automata

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

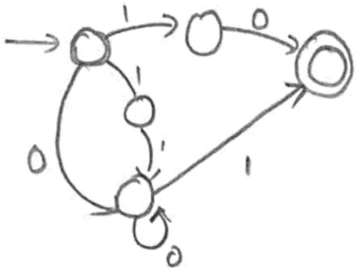
Step 1:



Step 2:

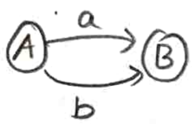


Step 3:

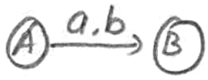


② Rules

$(a+b)$



(or)



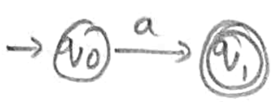
$(a \cup b)$

$(a \cdot b)$

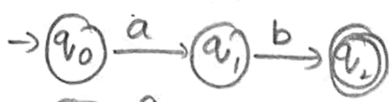
a^*

Direct Method

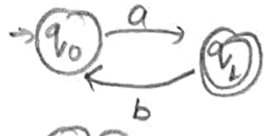
RE = a ⇒



RE = a • b ⇒



RE = a+b ⇒

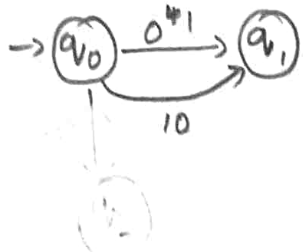


RE = a^+ →

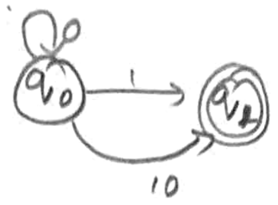
RE = $(a+b)^*$ →

① $0^*1 + 10$

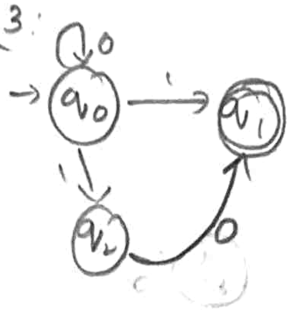
Step 1



Step 2:

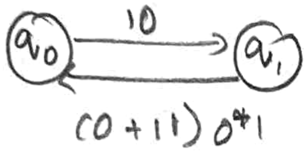


Step 3:

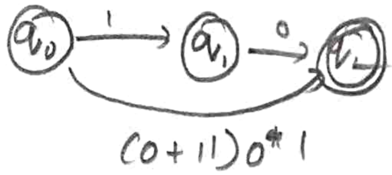


② $10 + (0+11)0^*1$

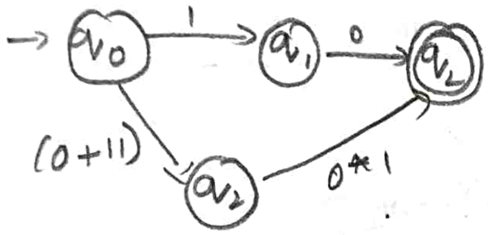
Step 1:



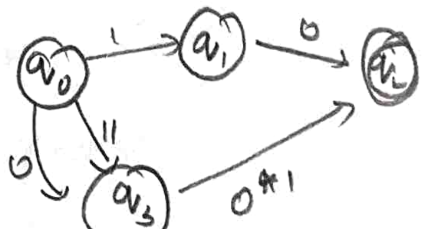
Step 2:



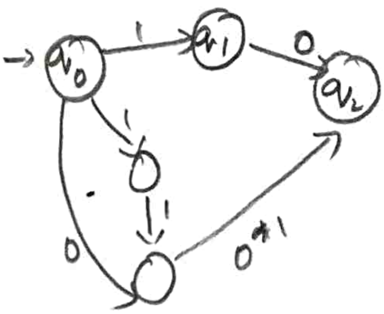
Step 3:



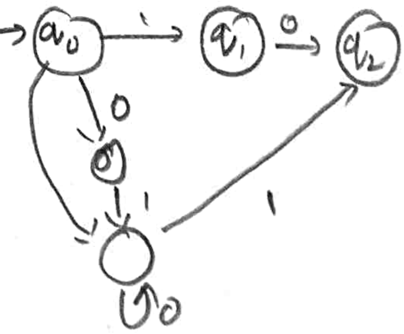
Step 4:



Step 5

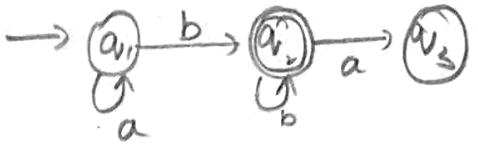


Step 6:



Using Arden's theorem

① construct RE from given DFA



Sol: lets write down the Equations for Each state

how. state = input coming to it * source state of i/p.

$q_1 = q_1 a + \epsilon$

Since q_1 is starting state ϵ will be added.

$q_2 = q_1 b + q_2 b$

Let us simplify q_1

$q_1 = q_1 a + \epsilon$ it can also be written as.

$q_1 = \epsilon + q_1 a$

it is in the form of. $R = Q + RP$ which can be further reduced to $R = QP^*$

assuming $R = q_1 \quad Q = \epsilon, P = a$

we get.

$q_1 = \epsilon \cdot a^* \quad q_1 = a^* \quad \{\epsilon \cdot R = R\}$

sub value of q_1 in q_2 we get

$q_2 = q_1 b + q_2 b$

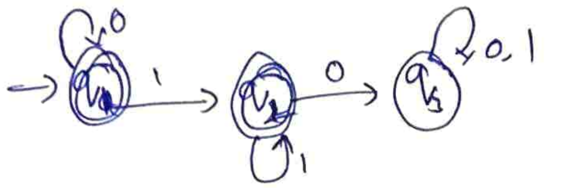
$q_2 = a^* b + q_2 b$

we can compare this Equation with $R = Q + RP$ assuming $R = q_2 \quad Q = a^* b, \ P = b$. which gets reduced to $R = QP^*$

$q_2 = a^* b \cdot b^* \quad \{R \cdot R^* = R^+\}$

$q_2 = a^* \cdot b^+$

construct RE for the given DFA



$q_1 = q_1 0 + \epsilon$

$q_2 = q_1 1 + q_2 1$

$q_3 = q_2 0 + q_3 (0+1)$.

final states are. q_1 q_2 we concentrate on those states.

$q_1 = q_1 0 + \epsilon$

$q_1 = \epsilon + q_1 0$

Similary to $R = Q + RP$. $R = q_1$

$q_1 = \epsilon (0)^*$. $q_1 = 0^*$. $\epsilon \cdot R^* = R$

$\hookrightarrow QP^*$

sub q_1 in q_2 we get

$\cancel{QP = \epsilon \cdot 0^*}$ $q_2 = 0^* 1 + q_2 1$

$R = Q + RP \Rightarrow QP^*$ $q_2 = 0^* 1 (1)^*$.

Regular Expression is given by

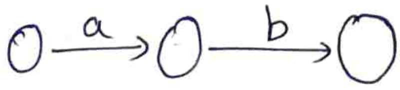
$r = q_1 + q_2$.

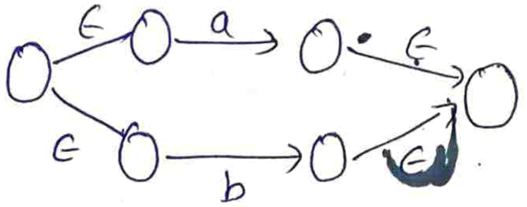
$= 0^* + 0^* 1 \cdot 1^*$

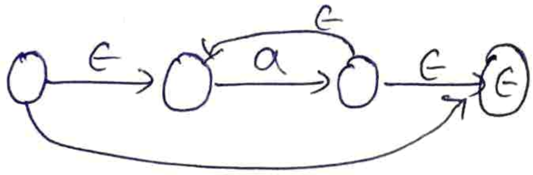
$r = 0^* + 0^* 1^+$ $\therefore 1 \cdot 1^* = 1^+$

Convertion of the Regular Expression to DFA

→ RE to ∈-NFA ~~to DFA~~

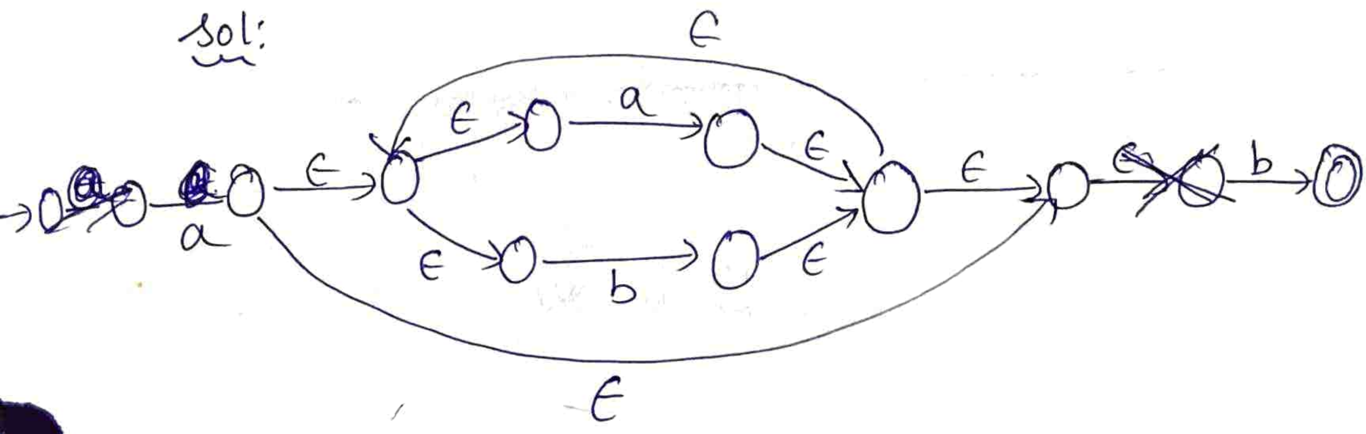
ab ⇒  (or) 

a+b ⇒ 

a^* → 

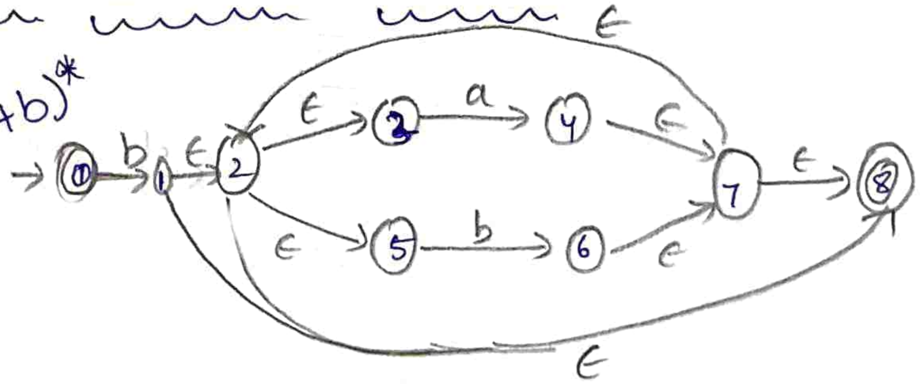
Ex: $a(a+b)^*b$

Sol:



RE to ∈-NFA to DFA

$b(a+b)^*$



Assume A as Starting State of DFA.

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

Equate $A = \epsilon\text{-closure}(0) = \{0\}$

New state is A in DFA

so,

$\delta(A, a) = \epsilon\text{-closure}(\delta(0, a)) = \phi \Rightarrow$ Dead/Empty State

$\delta(D, a) = \delta(D, b) = \phi \Rightarrow \emptyset$

$\delta(A, b) = \epsilon\text{-closure}(\delta(0, b)) = 1$

$= \epsilon\text{-closure}(1) = \{1, 2, 5, 3, 8\} \Rightarrow B$

$\delta(B, a) = \epsilon\text{-closure}(\delta(1, a) \cup \delta(2, a) \cup (5, a) \cup (3, a) \cup (8, a))$

$= \epsilon\text{-closure}(\phi \cup \phi \cup \phi \cup 4 \cup \phi)$

$= \epsilon\text{-closure}(4) = \{4, 7, 8, 2, 3, 5\} \Rightarrow C$

$\delta(B, b) = \epsilon\text{-closure}(\delta(1, b) \cup \delta(2, b) \cup (5, b) \cup (3, b) \cup (8, b))$

$= \epsilon\text{-closure}(\phi \cup \phi \cup 6 \cup \phi \cup \phi)$

$= \epsilon\text{-closure}(6) = \{6, 7, 8, 2, 3, 5\} \Rightarrow E$

$\delta(C, a) \Rightarrow \epsilon\text{-closure}(\delta(4, a) \cup (7, a) \cup (2, a) \cup (3, a) \cup (5, a))$

$= \epsilon\text{-closure}(\phi \cup \phi \cup \phi \cup 4 \cup \phi)$

$= \epsilon\text{-closure}(4) = \{4, 7, 8, 2, 3, 5\}$

$\delta(C, b) \Rightarrow \epsilon\text{-closure}(\delta(4, b) \cup (7, b) \cup (2, b) \cup (3, b) \cup (5, b))$

$= \epsilon\text{-closure}(\phi \cup \phi \cup \phi \cup \phi \cup 6)$

$= \epsilon\text{-closure}(6)$

$\delta(E, a) = \epsilon\text{-closure}(\delta(6,a) \cup \delta(7,a) \cup \delta(8,a) \cup \delta(2,a) \cup \delta(3,a) \cup \delta(5,a)\}$ ③

$= \epsilon\text{-closure}(\cancel{\delta} \phi \cup \phi \cup \phi \cup \phi \cup 4)$

$= \epsilon\text{-closure}(4)$

$\delta(E, b) = \epsilon\text{-closure}(\delta(6,b) \cup \delta(7,b) \cup \delta(8,b) \cup \delta(2,b) \cup \delta(3,b) \cup \delta(5,b))$

$= \epsilon\text{-closure}(\phi \cup \phi \cup \phi \cup \phi \cup \phi, \cup 6)$

$= \epsilon\text{ closure}(6)$

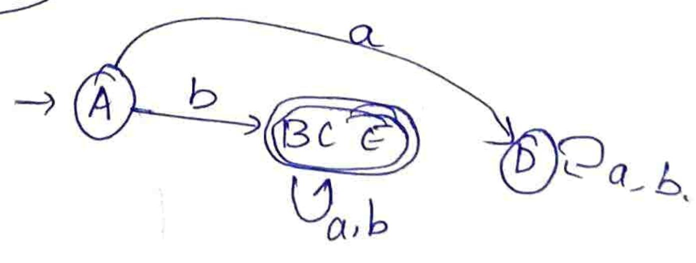
~~See~~, * final state → which ever state contains final state & then that state/set becomes final set

Transition table.

	a	b
→ A	D	B
*B	C	E
C*	C	E
D	D	D
E*	C	E

⟸

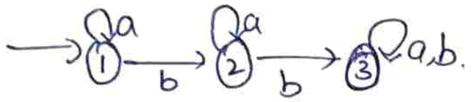
	a	b
A	D	{B, C, E}
D	D	D
B, C, E	BCE	BCE.



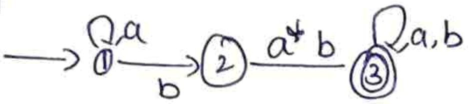
DFA to Regular Expression:

$1 \xrightarrow{a} 2 \xrightarrow{b} 3$ \ or $0 \xrightarrow{ab} 0$.

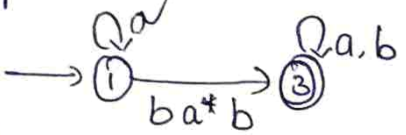
Ex:



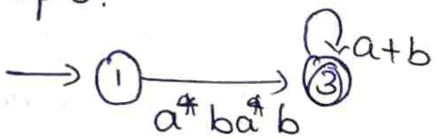
Step 1:

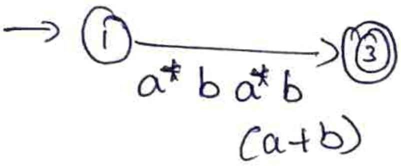


Step 2:



Step 3:

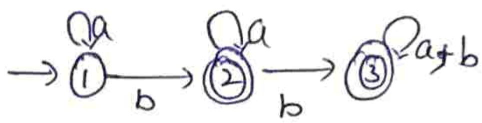


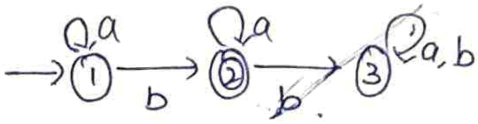


Re : $a^* b a^* b (a+b)^*$

Two final state

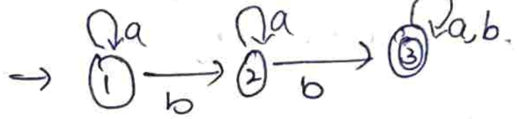
(2)





$a^* b a^*$

+



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

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

Pumping Lemma: is used to check Language is Regular or Not

Language < finite always Regular.
infinite may be Regular or Not.

↓ PLT.

(2)

Pass — undecidable

fail — Not Regular Language

Pumping Lemma:

If L is an infinite language then there Exits some Positive integer 'n' (pumping lemma) such that any string $w \in L$ has length greater than Equal to 'n' i.e., $|w| \geq n$ then w can be divided into three parts, $w = xyz$ satisfy following conditions (i) for each $i \geq 0$, $xy^iz \in L$ (ii) $|y| > 0$ (iii) $|xy| \leq n$

Ex: $a^n b^{2n}$ where $n \geq 0$

suppose if $n=2$ then

$xy^i z \xrightarrow{Pump} xy^{i+1}z$.

$\underset{x}{aa}\ \underset{y}{bb}\ \underset{z}{bb}$ Assume & divide. $\in L$

$w = xyz$

now just pump y

$\underset{x}{aa}\ \underset{y}{bbbb}\ \underset{x}{bb}$ $\notin L$

if it is not belonging to language then we can say that PLT is failed and Languag is not Regular.

Ex2

$\underset{x}{a}\underset{y}{a}\ \underset{z}{bbbb}$ $\in L$.

$\underset{x}{a}\ \underset{y}{abab}\ \underset{z}{bbb}$ $\notin L$ Not a regular language.

Pumping Lemma for Regular sets

Ex 3: Using pumping lemma Prove that language $A = \{a^n b^n / n \geq 0\}$ is not regular

Sol: Assume that A is regular

Pumping length = p

$S = a^p b^p$

x y z

Assume $P = 7$

= aaaaaaa bbbbbbb

Case (i) The Y is in the 'a' part.

$\underset{x}{aa}\underset{y}{aaaa}\underset{z}{abbbbbbb}$.

Case (ii) The y is in the 'b' part

$\underbrace{aaaaaaa}_{x}\underbrace{bbb}_{y}\underbrace{bbb}_{z}$.

Case (iii) The y is the 'a' and 'b' part

$\underbrace{aaaaaa}_{x}\underbrace{abb}_{y}\underbrace{bbbbb}_{z}$.

xy^iz

① xy^2z aa aaaa aaaa a bbbbbbb

$11 \neq 7$

② $xy^iz \Rightarrow xy^2z$

aaaaaaa bb bbbb bbbbb

$7 \neq 11$

③ $xy^iz \Rightarrow xy^2z$

aaaaa aabbaabb bbbbb

~~∉~~ pattern is not followed.

Case ①

$|xy| \leq P$ $6 \leq 7$ ✓

Case ②

$|xy| \leq P$ $13 \leq 7$ ✗

Case ③ $|xy| \leq P$ $9 \leq 7$ ✗

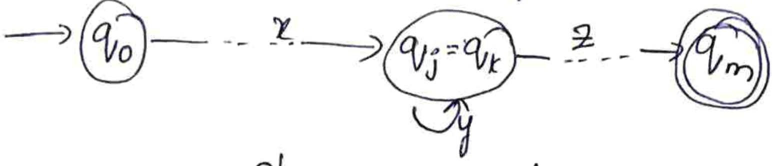
Pumping Lemma for regular sets

Theorem: Let $M = (Q, \Sigma, \delta, q_0, F)$ be a finite automata with n states. Let L be the regular set accepted by M. Let $w \in L$ and $|w| \geq m$. If $m \geq n$, then there exists x, y, z such that $w = xyz$, $y \neq$ null and $xy^iz \in L$ for each $i \geq 0$

Step 1: Assume L is regular. Let n be the number of states in the corresponding FA.

Step 2: Choose a string w such that $|w| \geq n$, use pumping lemma to write $w = xyz$, with $|xy| \leq n$ and $|y| > 0$

Step 3: Find a suitable integer i such that $xy^iz \notin L$. This contradicts our assumption. Hence L is not regular.



String Accepted by M.

Problem: Show that the set $L = \{b^{i^2} / i \geq 1\}$ is not regular

Step 1: Suppose L is regular and let there are n states in FA accepting L.

Step 2: Let $w = b^{n^2}$ then $|w| = n^2 > n$. By step 2 we can write $w = xyz$ with $|xy| \leq n$ and $|y| > 0$.

Step 3: Consider xy^2z so $|xy^2z| = |x| + 2|y| + |z|$ and it should be greater than $|x| + |y| + |z|$ ie. $|x| + 2|y| + |z| > |x| + |y| + |z|$ as $|y| > 0$

This means $n^2 = |xyz| = |x| + |y| + |z| < |xy^2z|$

As $|xy| \leq n$, $|y| \leq n$ therefore.

$|xy^2z| = |x| + 2|y| + |z| \leq (n^2 + n)$, ie.

$n^2 < |xy^2z| \leq n^2 + n < n^2 + n + n + 1$

So $n^2 < |xy^2z| < (n+1)^2$

then $|xy^2z|$ is not a perfect square so $xy^2z \notin L$

But by pumping lemma $xy^2z \in L$

Thus this a contradiction.

Properties of Regular Sets

① The union of two regular set is regular.

$RE_1 = a(aa)^*$ $RE_2 = (aa)^*$.

$L_1 = \{a, aaa, aaaa \ldots\}$

$L_2 = \{\epsilon, aa, aa, aaa \ldots\}$

$L_1 \cup L_2 = \{\epsilon, a, aaa, aaaa . aaaaa \ldots\}$.

(strings of all possible length including null.

$RE \{L_1 \cup L_2\} = a^*$ (which is a regular Expression itself)

② The intersection of two regular set is regular.

$RE_1 = a(a^*)$ and $RE_2 = (aa)^*$

$L_1 = \{a, aa, aaa, aaaa \ldots\}$.

$L_2 = \{\epsilon, aa, aaaa, aaaaaa \ldots\}$

$L_1 \cap L_2 = \{aa, aaaa, aaaaaa \ldots\}$ Even length

$RE(L_1 \cap L_2) = aa(aa)^*$. which is a regula expression itself

③ The compliment of regular set is regular.

$RE_1 = (aa)^*$

$L = \{\epsilon, aa, aaaa, aaaaaa \ldots\}$.

compliment of L is all strings not in L.

$L' = \{a, aaa, aaaaa \ldots\}$.

$RE(L') = \{ a(aa)^*$.

Grammar

{V, T, P, S} → Quadraple.

V → Set of Non-terminals / Variables {upper case} similar to states.

T → Set of terminals (symbols) {Lower case}.

* P → Production Rules

S → start symbol

→ Represented as LHS → RHS

LHS ↓ Single Non terminals

RHS → ε / string of terminals of non terminals.

A → aB → Production Rule

B → ε / aB

Grammar

Grammar

- Regular Grammar
  - LHS → RHS
    - LHS ↓ Non terminal
    - RHS ↓ ε, terminal, terminal followed by NT | NT " " T
  - Eg
    - left linear Grammar
      - D → Ab.
      - ↓ Non Terminal should be on left side
    - Right linear Grammar
      - C → aA
      - ↳ Right most.

Ex:

A → ε

B → a

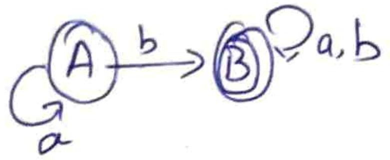
C → aA

D → Ab

Regular Grammar

A → aBb

↳ Not a Regular Grammar



Construct Regular Grammar.

$V = \{A, B\}$ $T = \{a, b\}$ $P = \{?\}$ $S = \{A\}$

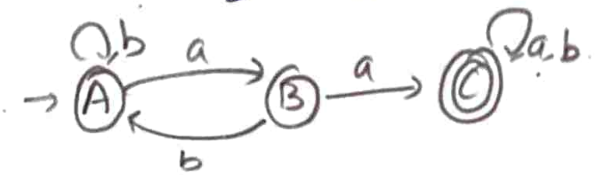
$P = \{ A \rightarrow aA$ → Right Linear Grammar.

$A \rightarrow bB / b$

$B \rightarrow aB / bB$ Regular Grammar.

$B \rightarrow a$ → Terminal.

$B \rightarrow b$

(Q) 

$V = \{A, B, C\}$ $T = \{a, b\}$ $P = \{?\}$ $S = \{A\}$

$P = \{ A \rightarrow aB,$

$A \rightarrow bA,$

$B \rightarrow aC,$

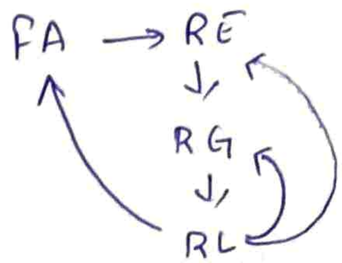
$B \rightarrow a,$

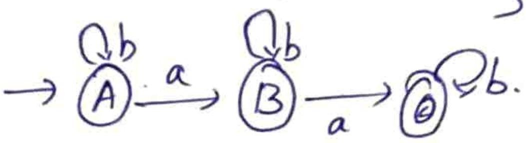
$C \rightarrow aC,$

$C \rightarrow a,$

$C \rightarrow bC,$

$C \rightarrow b. \}$



(Q) 

* Right Linear Grammar is Regular Grammar

Sol:

$A \rightarrow aB$

$A \rightarrow bA$

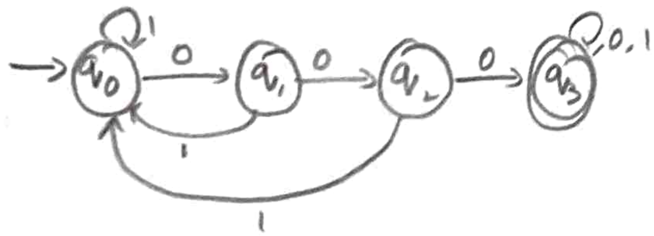
$B \rightarrow aC$

$B \rightarrow a$

$B \rightarrow bB$

$C \rightarrow bC$

$C \rightarrow b$



G0, G1, G3, G8, G9, H0, 4,

H → 0, 1, 2, 6, 7, 9, 4

J → 2, 4, 5, 6, 8, 9, 3

K → 1, 2, 3, 4, 6, 7, 8, 9

L → 0, 1, 3, 4, 6, 8

M → 0, 1, 2, 6, 8, 9

N → 0, 1, 6, 7, 8, 9

P → 1, 2, 3, 6, 7, 8

Le – 13, 14, 15, 16, 17, 18,

Conversion of right linear grammar to left linear grammar

$S \to bB$

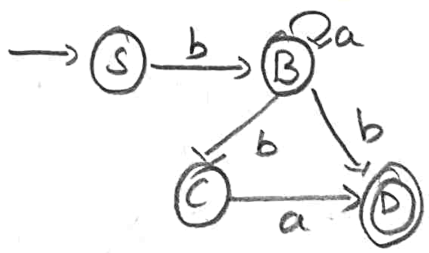
$B \to bC$

$B \to aB$

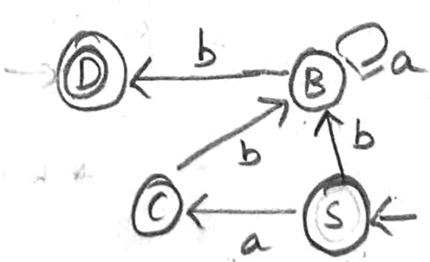
$C \to a$

$B \to b$

Step 1: Construct finite automata



Step 2: Reverse finite automata (interchange initial & final state and reverse edges)



Step 3: Write a Right linear grammar.

$S \to bB$

$S \to aC$

$C \to bB$

$B \to aB$

$B \to bD$

Step 4: write left linear Grammar

$S \to Bb$

$S \to Ca$

$C \to Bb$

$B \to Ba$

$B \to Db$

Construction of LLG to RLG.

$S \rightarrow Sa / Sb / Ab$

$A \rightarrow Aa / Ba$

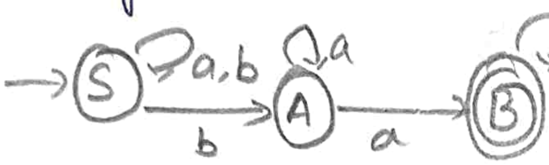
$B \rightarrow Bb / \epsilon$

Step 1: Design FA

step 2: Interchange start state and final state

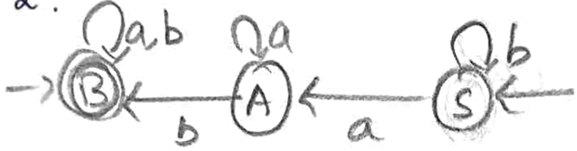
Step 3: Reverse the directions

Sol: Design finite Automata



(as we have epsilon)

step 2:



step 3:

$S \rightarrow bS / aA$

$A \rightarrow aA / bB$

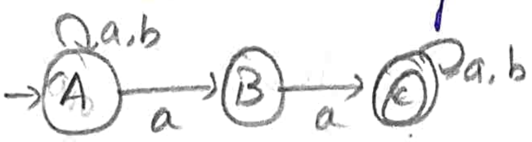
$B \rightarrow aB / bB / \epsilon$

Equivalence of Regular Grammar and Finite Automata

Yes. we have Equivalence b/w RG + FA

Ex:

(Q) "aa" as substring



$A \rightarrow aB$

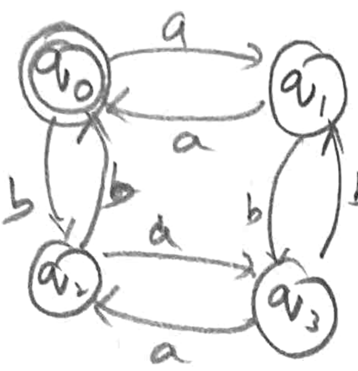
$A \rightarrow aA / bA$

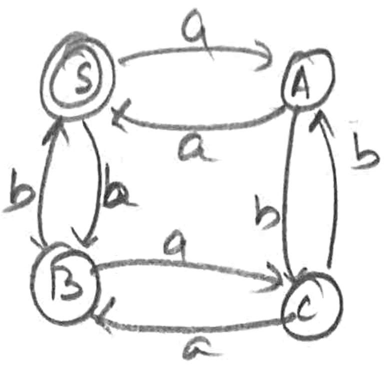
$B \rightarrow aC$

$C \rightarrow aB / bC / \epsilon$

as it is final state

Even a's and Even b's.





$S \rightarrow aA / bB / \epsilon$

$A \rightarrow bC / aS / \phi$

$B \rightarrow aC / bS$

$C \rightarrow bA / aB$.

Language is said to be Regular ~~Grammar~~ iff ∃ Regular ~~Language~~ Grammar generates it

Language is said to be Regular iff ∃ FA accepting it

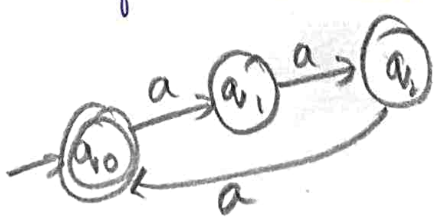
Regular Grammar

LHS → RHS

↓ ↓

1 NT $T^* / T^* N$ (OR) NT^* / T^*

No. of a's divisible 3



$S \rightarrow \underline{aaa}\, S / \epsilon$

$S \rightarrow aA / \epsilon$

$S \rightarrow aB$

$S \rightarrow aS$

$G_0, G_2, G_5, G_7, G_8, H_1, H_4, H_9$, H_0, H_7, H_8

$J_4, J_6, J_7, J_8, K_1, K_2, K_3, K_4, K_6, K_7, K_8, K_9$

$L_1, L_4, L_8, L_5, L_6,$

$M_1, M_2, M_3, M_6, M_8, M_9, N_0, N_4, N_6, N_7, N_8, N_9$

P_6, P_3, P_{48}

Le 13, 14, 18,

Module - III

Context Free Grammar

Formal Defination : Rules to form a string, inturn forms a language

CFG : (V, T, P, S)

- finite of Non-Terminals (captial)
- finite set of Terminals (Lower) $(V \cap T = \phi)$
- Production rules. $\alpha \to \beta$

  L.H.S R.H.S

  Type 2 { only 1 NT $(V \cup T)^*$
- start symbol. (in left hand side)

Ex: $S \to 0S1$

$S \to \epsilon$

start symbol $= \{S\}$

$T = \{0, 1\}$ $P = 2$ (or) 1

Derivation Tree: $\{V, T, P, S\}$

$G : E \to E+E \,/\, E*E \,/\, E=E \,/\, id$

$W : id + id * id$

whether a word belong to Grammar or not