

UNIT-II

Regular Expressions:

- * Introduction
- * Examples
- * Components
- * Operations.

Introduction: Regular expressions are mathematical expressions describing a language which is accepted by FA.

* R.E describing a language called Regular language.

Definition:-

Let Σ be an alphabet then R.E over Σ is defined as follows.

- * \emptyset is a R.E then that describes an empty set. R.E = \emptyset .
- * ϵ is a R.E then that describes "NULL" string set. R.E = ϵ , R.E = { ϵ }
- * a is a R.E over Σ then that describes the set with a . R.E = { a }
- * Let r_1 and s are two R.E and L_1 and L_2 are two languages which are described by r_1 and s . Then,
 - $r_1 + s$ is equivalent to $L_1 \cup L_2$
 - $r_1 r_2$ is equivalent to $L_1 L_2$ (or) $L_1 \cap L_2$
 - r^* is equivalent to L^*

Components:-

union, concatenation, Kleene closure.

Examples:-

- 1) Write the R.E for the language accepting all combinations of a 's over $\Sigma = \{a\}$

Sol: $\Sigma = \{a\}$

$$L = \{\epsilon, a, aa, aaa, aaaa, \dots\}$$

$= a^*$

$\therefore R.E = a^*$

- 2) Write a R.E for the language accepting all combinations of a 's except empty string over $\Sigma = \{a\}$

Sol: $\Sigma = \{a\} \{a, aa, aaa, aaaa, \dots\}$

$$= a^*$$

$$\therefore R.E = a^*$$

- 3) Write a R.E for the language accepting any no. of a's and b's over $\Sigma = \{a, b\}$

Sol:-

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

$$\begin{aligned} L &= \{\epsilon, a'b', a^2, ab, ba, b^2, \dots\} \\ &= \{\epsilon, a, b, aa, ab, ba, bb, \dots\} \\ &= (a+b)^* \end{aligned}$$

$$\therefore R.E = (a+b)^*$$

- 4) Write a R.E for the language containing all strings which are ending with '00' over $\Sigma = \{0, 1\}$.

Sol:-

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

$$L = \{ \underbrace{(0+1)^*}_{+} 00 \}$$

$$\therefore R.E = (0+1)^* 00$$

- 5) Write a R.E for the language Accepting set of all string which are starting with '1' and ending with '0' over $\Sigma = \{0, 1\}$

Sol:-

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

$$1 \underbrace{(0+1)^*}_{+} 0$$

$$\therefore R.E = 1 \underbrace{(0+1)^*}_{+} 0$$

- 6) Write a R.E for the language accepting any no. of a's followed by any no. of b's, followed by any no. of c's over $\Sigma = \{a, b, c\}$

Sol:-

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

$$\therefore R.E = a^* b^* c^*$$

- 7) Write a R.E for the language accepting set of all strings which contains the third character from the right end of the string is always 'a' over $\Sigma = \{a, b\}$. $\therefore R.E = (a+b)^* a \underline{(a+b)} (a+b)$

Sol:-

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

$$(a+b)^* \underline{a} \underline{(a+b)} (a+b)$$

language Associated with RE:-

- * A language which is described by RE is called Regular language.
- * Let R be a RE then the language accepted by R is denoted by $L(R)$.

Ex:- If the RE $R = (ba)^*$ then $L(R) = ?$

$$\text{Sol: } R = (ba)^*$$

$$L(R) = \{ (ba)^0, (ba)^1, (ba)^2, (ba)^3, \dots \}$$

$$= \{ \epsilon, ba, baba, bababa, \dots \}$$

Properties of RE (Identity rules):

Let R, S be RE then the following properties are two

$$1) (R+S)+T = R+(S+T) \quad 11) R^* R^* = R^* = (R^*)^*$$

$$2) R+R = R.$$

$$12) RR^* = R^* R = R^*$$

$$3) R\emptyset = \emptyset + R = R$$

$$13) (R+S)^* = (R^* S^*)^* = (R^* + S^*)^*$$

$$4) R\emptyset = \emptyset R = \emptyset$$

$$14) (R \cdot S)^* = (R^* + S^*)^* = (R^* S^*)^*$$

$$5) R+S = S+R$$

$$6) RE = \epsilon R = R$$

$$7) R(ST) = (RS)T$$

$$8) R(S+T) = RS + RT$$

$$9) (S+T)R = SR + TR$$

$$10) \emptyset^* = \epsilon^* = \epsilon$$

Manipulation of RE (Basic operations)

- i) union
- ii) concatenation
- iii) Kleene closure.

UNION:

Let R, S be two RE then union of R, S is defined as

$$R \cup S = \{ z \mid z \in R \text{ or } z \in S \}$$

Ex:- If $R = \{ab, c\}$ and $S = \{d, ef\}$ then $R \cup S = ?$

$$R \cup S = \{ ab, c, def \}$$

concatenation:-

Let R, S be two RE then concatenation of R, S is defined as $RS = \{ xy \mid x \in R \text{ and } y \in S \}$

Ex: If $R = \{ab, c\}$ and $S = \{d, ef\}$ then $RS = ?$

Sol: $RS = \{ab, c\} \{d, ef\}$
 $= \{abd, abef, cd,cef\}$

Kleene closure:

Let 'R' be a R.E then the Kleene closure of R is denoted as R^* which contains set of all strings including null string.

Ex: If $R = \{ab\}$ then $R^* = ?$

Sol: $R^* = \{(ab)^0, (ab)^1, (ab)^2, (ab)^3, \dots\}$
 $= \{\epsilon, ab, abab, ababab, \dots\}$

examples:

construct a string set for the R.E given below.

i) ab^*a

Sol: $\{ab^0a, ab^1a, ab^2a, \dots\}$
 $\{\epsilon, aa, aba, abba, \dots\}$

ii) 1^*0

$$\begin{aligned} & \{1^0, 1^1, 1^2, 1^3, 1^4, \dots\}0 \\ &= \{\epsilon, 1, 11, 111, 1111, \dots\}0 \\ &= \{0, 10, 110, 1110, 11110, \dots\} \end{aligned}$$

iii) 00^*

$$\begin{aligned} &= \{00^0, 00^1, 00^2, 00^3, 00^4, \dots\} \\ &= \{0, 00, 000, 0000, 00000, \dots\} \\ &= 0^+ \end{aligned}$$

iv) $(100^+)^*$

$$\begin{aligned} &= ((100^0, 100^1, 100^2, 100^3, \dots))^* \\ &= (100, 1000, 10000, \dots)^* \\ &= ((100)^*, (1000)^*, (10000)^*) \\ &= (\{\epsilon, 100, 100100, 100100100, \dots\}, \{\epsilon, 1000, 10001000, 100010001000, \dots\}, \{\epsilon, 10000, 1000010000, \dots\}) \end{aligned}$$

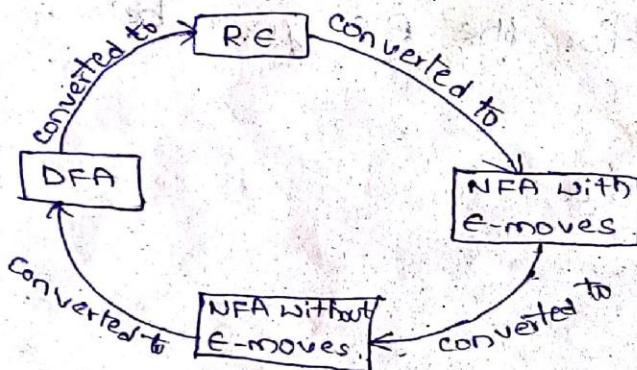
$$\text{v) } (0+1)^* = 0^* + 1^* \\ = \{\epsilon, 0, 00, 000, \dots\} \cup \{\epsilon, 1, 11, 111, 1111, \dots\} \\ = \{\epsilon, 0, 00, 000, \dots, 1, 11, 111, 1111, \dots\}$$

$$\text{vi) } (0+1)^* 011 \\ = (0^* + 1^*) 011 \\ = ((\{\epsilon, 0, 00, 000, \dots\} \cup \{\epsilon, 1, 11, 111, \dots\}) 011) \\ = ((\{\epsilon, 0, 00, 000, \dots, 1, 11, 111, \dots\}) 011) \\ = \{011, 0011, 00011, 000011, \dots, 1011, 11011, 111011, \dots\}$$

Equivalence of R.E and FA:

1. Conversion of R.E to FA
2. Conversion of FA to R.E

Relationship b/w R.E and FA:



1. Conversion of R.E to FA:

Basic notations used.

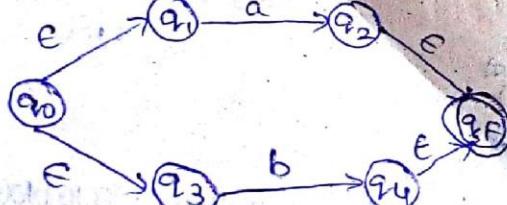
1. If the R.E is like $\emptyset = \epsilon$ then FA is



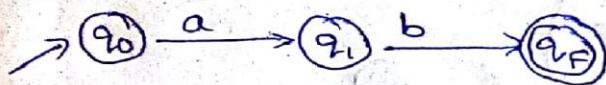
2. If the R.E is like $a = a$ then FA is



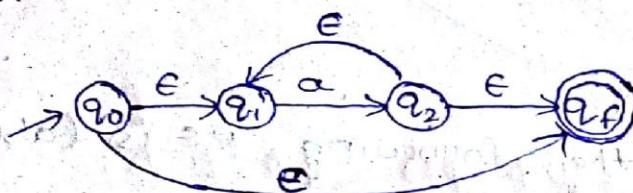
3. If the R.E is like $r = a+b$ then FA is



4. If the RE is like $\eta = ab$ then FA is



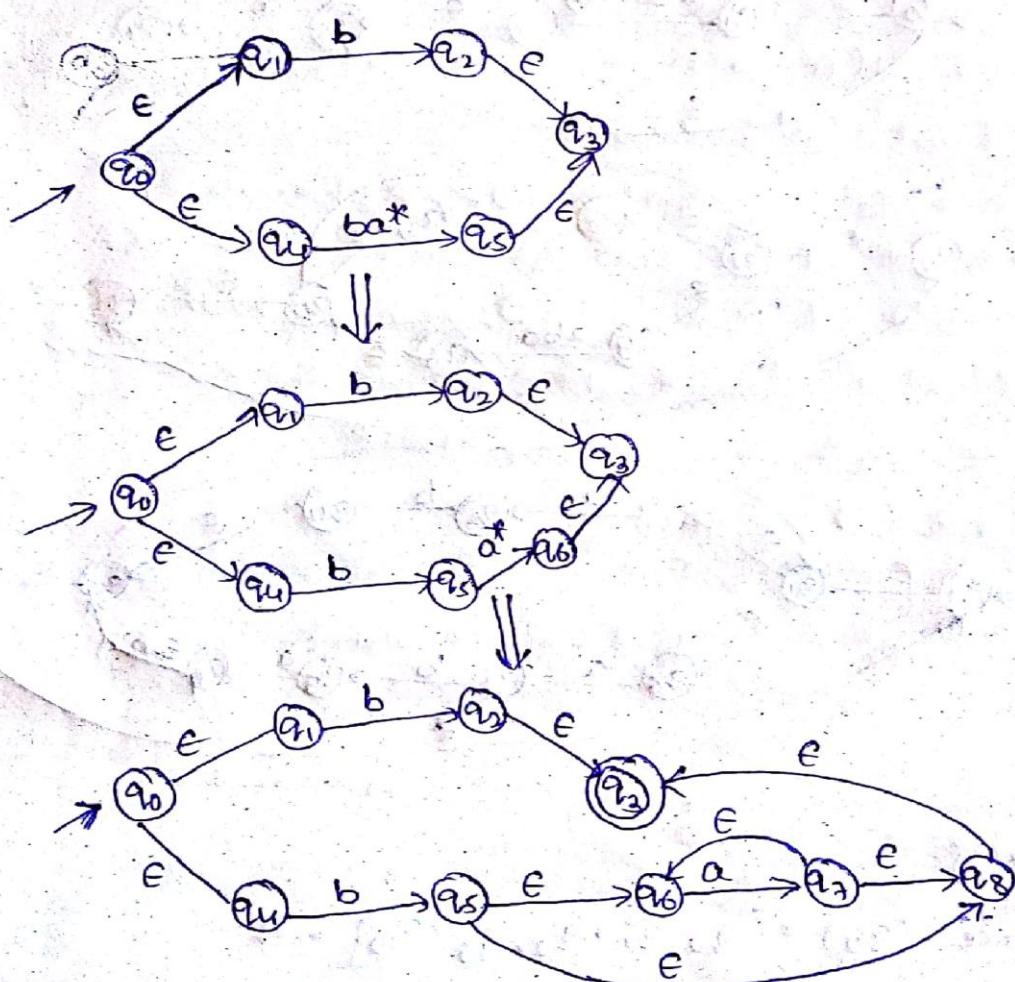
5. If the RE is like $\eta = a^*$ then FA is



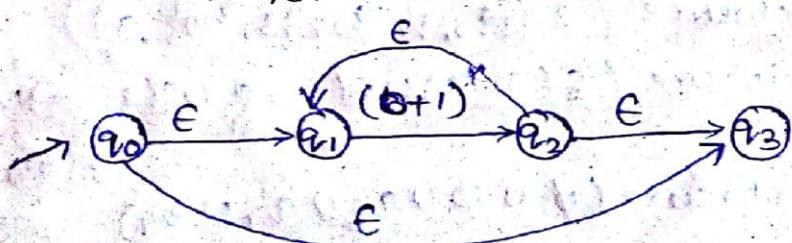
examples:

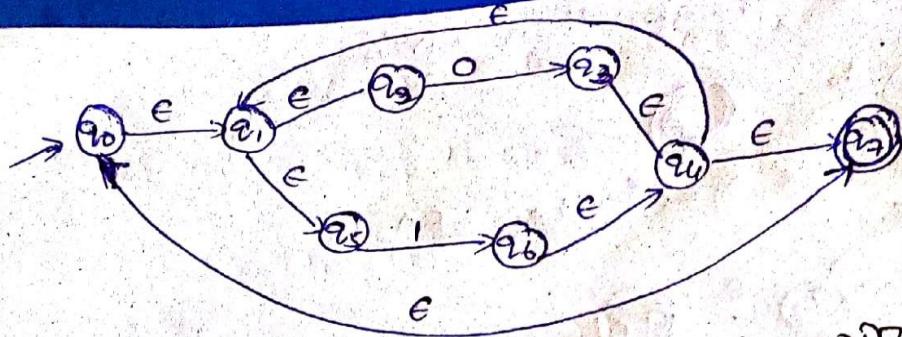
i) Construct an NFA with ϵ -moves for the RE $b + ba^*$

Sol:- The given RE $\tau = b + ba^*$

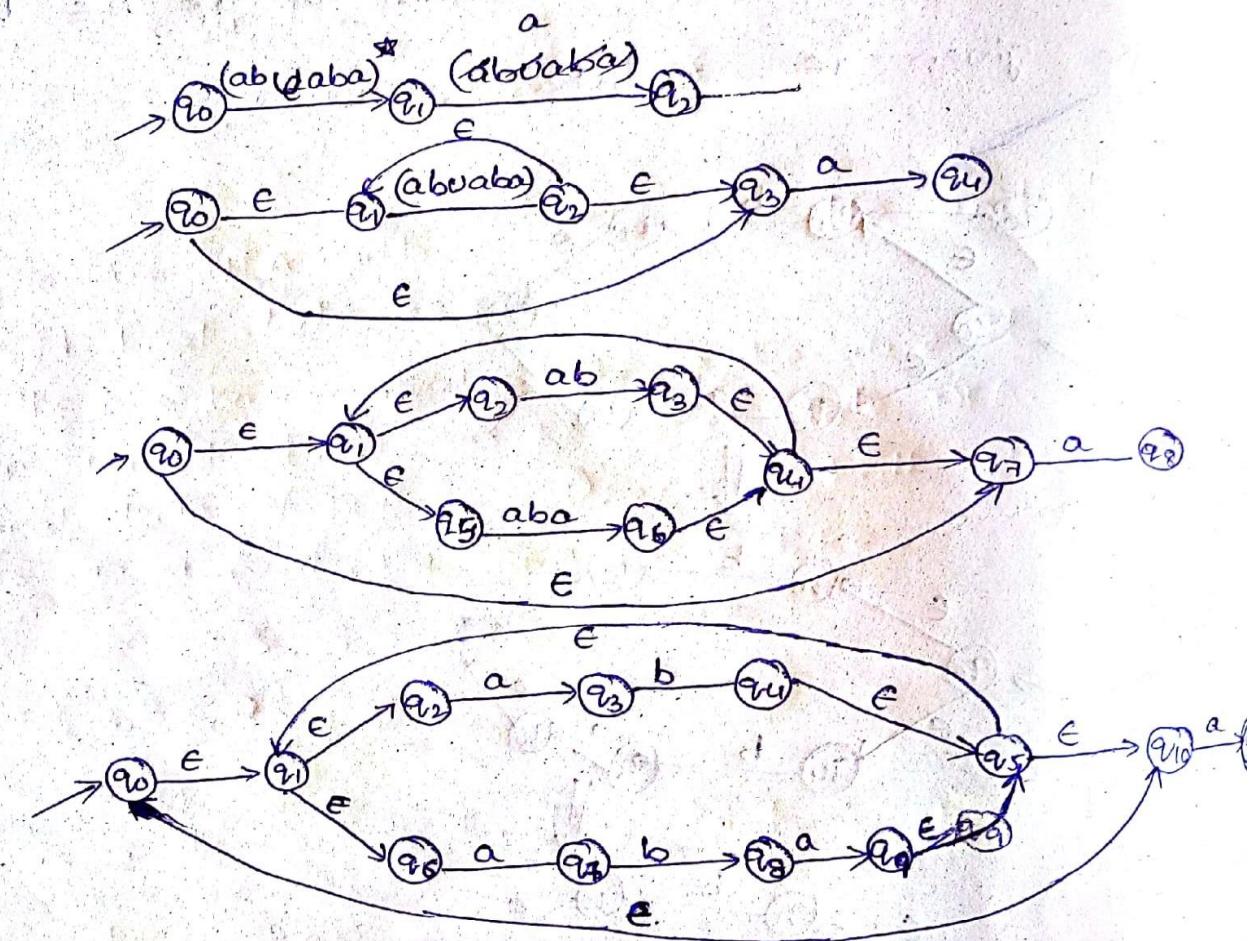


ii) construct an NFA with ϵ -moves for the RE is. (contd)





- 3) Construct a DFA for the following
the given RE is $(abuaba)^*a$



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

$$\epsilon\text{-closure } (\{q_0\}) = \{q_0, q_1, q_2, q_6, q_{10}\} = A$$

$$S'(A, a) = \epsilon\text{-closure } (S(A, a))$$

$$= \epsilon\text{-closure } (S(\{q_0, q_1, q_2, q_6, q_{10}\}, a))$$

$$= \epsilon\text{-closure } (S(q_0, a) \cup S(q_1, a) \cup S(q_2, a) \cup S(q_6, a) \cup S(q_{10}, a))$$

$$= \epsilon\text{-closure } (\emptyset \cup \{q_3\} \cup \{q_7\} \cup \{q_{11}\})$$

$$= \epsilon\text{-closure } (q_3) \cup \epsilon\text{-closure } (q_7) \cup \epsilon\text{-closure } (q_{11})$$

$$= \{q_3\} \cup \{q_7\} \cup \{q_{11}\}$$

$$= \{q_3, q_7, q_{11}\} = B$$

$$S^1(A, b) = \epsilon\text{-closure}(S, (A, b))$$

$$= \epsilon\text{-closure}(S(\{q_0, q_1, q_2, q_6, q_{10}\}), b)$$

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

$$= \emptyset$$

$$S^1(B, a) = \epsilon\text{-closure}(S, (B, a))$$

$$= \epsilon\text{-closure}(S(\{q_3, q_7, q_{11}\}), a)$$

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

$$= \emptyset$$

$$S^1(B, b) = \epsilon\text{-closure}(S(B, b))$$

$$= \epsilon\text{-closure}(S(\{q_3, q_7, q_{11}\}, b))$$

$$= \epsilon\text{-closure}(q_4 \cup q_8 \cup \emptyset)$$

$$= \epsilon\text{-closure}(q_4) \cup \epsilon\text{-closure}(q_8)$$

$$= \text{efecto. } \{q_4, q_5, q_{10}, q_6, q_2, q_8\} \cup \{q_8\}$$

$$= \{q_1, q_2, q_4, q_5, q_6, q_8, q_{10}\}$$

$$= C.$$

$$S^1(C, a) = \epsilon\text{-closure}(S(C, a))$$

$$= \epsilon\text{-closure}(S(\{q_1, q_2, q_4, q_5, q_6, q_8, q_{10}\}), a))$$

$$= \epsilon\text{-closure}(\emptyset \cup q_3 \cup \emptyset \cup \emptyset \cup q_7 \cup q_9 \cup q_{11})$$

$$= \epsilon\text{-closure}(q_3) \cup \epsilon\text{-closure}(q_7) \cup \epsilon\text{-closure}(q_9) \cup \epsilon\text{-closure}(q_{11})$$

$$= \{q_3\} \cup \{q_7\} \cup \{q_9, q_5, q_{10}, q_1, q_2, q_6\} \cup \{q_{11}\}$$

$$= \{q_1, q_2, q_3, q_5, q_6, q_7, q_9, q_{10}, q_{11}\}$$

$$= D$$

$$S^1(C, b) = \epsilon\text{-closure}(S(C, b))$$

$$= \epsilon\text{-closure}(S(\{q_1, q_2, q_4, q_5, q_6, q_8, q_{10}\}), b))$$

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

$$= \emptyset$$

$$S^1(D, a) = \epsilon\text{-closure}(S(D, a))$$

$$= \epsilon\text{-closure}(\emptyset \cup q_3 \cup \emptyset \cup \emptyset \cup q_7 \cup \emptyset \cup q_1 \cup \emptyset)$$

$$\begin{aligned}
 &= \epsilon\text{-closure}\{q_3\} \cup \epsilon\text{-closure}\{q_7\} \cup \epsilon\text{-closure}\{q_{11}\} \\
 &= \{q_3\} \cup \{q_7\} \cup \{q_{11}\} \\
 &= B.
 \end{aligned}$$

$$\begin{aligned}
 S'(D, b) &= \epsilon\text{-closure}(S(D, b)) \\
 &= \epsilon\text{-closure}(\phi \cup \phi \cup q_4 \cup \phi \cup \phi \cup q_8 \cup \phi \cup \phi \cup \phi) \\
 &= \epsilon\text{-closure}(q_4) \cup \epsilon\text{-closure}(q_8) \\
 &= \{q_4, q_5, q_1, q_{10}, q_2, q_6\} \cup \{q_8\} \\
 &= \{q_1, q_2, q_4, q_5, q_6, q_8, q_{10}\} \\
 &= C.
 \end{aligned}$$

The previous NFA with ϵ -moves has the initial state as q_0 and final state as q_{11} .

Now the DFA has initial state A because it contains q_0 as an element which can be initial state NFA.

The final state of DFA is B, D because both states contain q_{11} as an element. The DFA 'M' is like,

$$M = \{Q, \Sigma, S, Q_0, F\} \text{ where}$$

$$Q = \{A, B, C, D\}$$

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

S : is a transition function which is defined as:

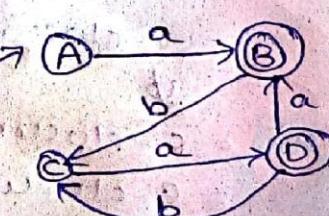
$S:$

	a	b
A	B	ϕ
B	ϕ	C
C	D	ϕ
D	B	C

$$q_0 = A$$

$$F = \{B, D\}$$

\therefore transition diagram for DFA is:



2nd method:

conversion of FA to R.E.

ARDEN'S Theorem:-

Let P and Q be two R.E. over the input alphabet Σ .

The R.E 'R' is given as $R = Q + RP$ which has a unique solution. $R = QP^*$

conversion algorithm:

* Let q_1 be a initial state.

* There are $q_2, q_3, q_4, \dots, q_n$ are no. of states. The final state may be some q_j where $j \leq n$.

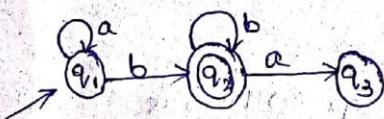
* Let α_{ij} be a transition from q_i to q_j state.

* calculate $q_j = \alpha_{ij} q_i$. If q_j is an initial state then

$$q_j = \alpha_{ij} q_i + \epsilon$$

* Similarly compute the final state equation which gives the R.E.

Ex:- Construct R.E. from given DFA.



Sol: $q_1 = aq_1 + \epsilon \rightarrow ①$

$$q_2 = bq_1 + bq_2 \rightarrow ②$$

$$q_3 = aq_2 \rightarrow ③$$

$$q_1 = aq_1 + \epsilon \quad \left(\begin{array}{l} R = Q + RP \\ R = QP^* \end{array} \right)$$

where

$$R = q_1$$

$$Q = \epsilon$$

$$P = \alpha$$

solution for above equ. is.

$$R = QP^*$$

$$q_1 = \epsilon a^*$$

$$q_1 = a^* \rightarrow ④$$

Substitute equ ④ in equ ②

$$q_2 = bq_1 + bq_2$$

$$q_2 = ba^* + bq_2$$

$$q_2 = ba^* b^*$$

$$= a^* bb^*$$

$$= a^* b^*$$

The R.E is $a^* b^*$

Construct R.E from given DFA



$$\text{Sd: } q_1 = 0q_1 + \epsilon \rightarrow ①$$

$$q_2 = 1q_1 + 1q_2 \rightarrow ②$$

$$q_3 = 0q_2 + 0q_3 + 1q_3 \rightarrow ③$$

$$q_1 = 0q_1 + \epsilon$$

$$q_1 = \epsilon + 0q_1 \quad (\therefore R = Q + RP \Rightarrow R = QP^*)$$

$$q_1 = \epsilon 0^*$$

$$q_1 = 0^* \rightarrow ④$$

QUBB equ ④ & equ ②

$$q_2 = 1q_1 + 1q_2$$

$$q_2 = 10^* + 1q_2$$

$$q_2 = 10^* 1^*$$

$$= 0^* 11^*$$

$$= 0^* 1^*$$

\therefore The regular expression for given DFA is

$$q_1 + q_2 = 0^* + 0^* 1^*$$

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

$$= 0^* 1^*$$

Construct R.E from given DFA.



Sol:

$$q_1 = \epsilon \rightarrow ①$$

$$q_2 = 0q_1 + 0q_3 \rightarrow ②$$

$$q_3 = 0q_2 \rightarrow ③$$

QUBB equ ① & equ ②

$$q_2 = 0q_1 + 0q_3$$

$$= 0 \cdot \epsilon + 0 \cdot q_3$$

$$q_2 = 0 + 0q_3 \rightarrow ④$$

sub E_{q2} ③ in E_{q2} ④

$$q_2 = 0 + 0 \cdot q_3$$

$$q_2 = 0 + 0 \cdot q_2 \quad (R = Q + RP \Rightarrow R = QP^*)$$

$$q_2 = 0(00)^*$$

The R.E for given DFA is

$$0(00)^*$$

//

Applications of R.E:-

* R.E are used for describing a language called Regular language.

* R.E are used to implement lexical analysis in Compiler design.

* R.E are used to represent a set of strings in UNIX programming.

Closure properties of Regular languages:-

* Regular languages are closed under Union.

*	"	"	"	"	"	Intersection
*	"	"	"	"	"	concatenation
*	"	"	"	"	"	Kleene closure
*	"	"	"	"	"	Difference
*	"	"	"	"	"	Complement
*	"	"	"	"	"	Reverse
*	"	"	"	"	"	Symmetric difference
*	"	"	"	"	"	Homomorphism
*	"	"	"	"	"	Inverse Homomorphism

Regular Grammar and FA:-

* Grammar

* Regular Grammar

1. left linear Grammar

2. Right linear Grammar

* FA from Regular Grammar

* RG from FA

* RE from RG.

Grammar:-

grammar is a set that contains four types like.

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

Regular grammar from FA:-

Let $M = (\mathcal{Q}, \Sigma, \delta, q_0, F)$ be a FA. It contains symbols.
no. of states like $\mathcal{Q} = \{q_1, q_2, \dots, q_n\}$ and $\Sigma = \{a_1, a_2, a_3, \dots, a_n\}$
Therefore the Regular grammar $G = (V, T, P, S)$ is defined as

$$V = \{q_1, q_2, \dots, q_n\}$$

$$T = \Sigma = \{a_1, a_2, \dots, a_n\}$$

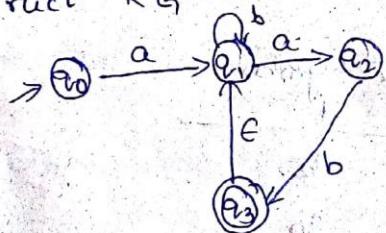
$$S = \emptyset = q_1$$

P = Transitions of FA.

Rules:-

- * If the transition of FA is like $q_i \xrightarrow{a} q_j$ then the production rule is $q_i \rightarrow a q_j$.
- * If there is a final state in FA is like q_i then the production rule is $q_i \rightarrow \epsilon$.
- * If the transition of FA is like $q_i \xrightarrow{a} q_j$ then the production rules are $q_i \rightarrow a q_j$, $q_i \rightarrow a$.

To :- construct RG from the given FA.



Sol:- The given FA is like $M = (\mathcal{Q}, \Sigma, \delta, q_0, F)$

$$\text{where } \mathcal{Q} = \{q_0, q_1, q_2, q_3\}$$

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

δ is a transition function is defined as.

δ : states	input symbols		
	a	b	ϵ
q_0	q_1	ϕ	ϕ
q_1	q_2	q_1	ϕ
q_2	ϕ	q_3	ϕ
q_3	ϕ	ϕ	q_1

$$q_0 = q_0$$

$$F = \{q_3\}$$

Now, therefore the equivalent regular grammar of M is defined as $G = (V, T, P, S)$ where

$$V = \{q_0, q_1, q_2, q_3\}$$

$$T = \{a, b, \epsilon\}$$

P is a set of production rules defined from transitions of M is like.

$P:$

$$\{q_0 \rightarrow aq_1\}$$

$$q_1 \rightarrow aq_2$$

$$q_1 \rightarrow bq_1$$

$$q_2 \rightarrow bq_3$$

$$q_2 \rightarrow b\cancel{q}_2$$

$$q_3 \rightarrow \epsilon q_1\}$$

$$S = \{q_0\}$$

2) construct RG from the given FA.

Sol: The given FA is like.

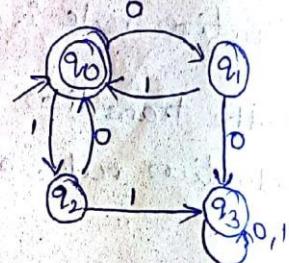
$$M = (Q, \Sigma, S, q_0, F)$$

$$Q = \{q_0, q_1, q_2, q_3\}$$

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

S:

states	input symbols	
	0	1
$\rightarrow q_0$	q_1 q_2	
q_1	q_3 q_0	
q_2	q_0 q_3	
q_3	q_3 q_3	



now, therefore the equivalent RG of M is defined as $G = (V, T, P, S)$ where

$$V = \{q_0, q_1, q_2, q_3\}$$

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

P is a set of production rules defined from transitions of M is like

$$P: \{q_0 \rightarrow 0q_1, q_1 \rightarrow 1, q_3 \rightarrow 1q_3, q_0 \rightarrow 1q_2, q_2 \rightarrow 0q_2, q_2 \rightarrow 0, q_1 \rightarrow 0q_3, q_2 \rightarrow 1q_3, q_1 \rightarrow q_0, q_3 \rightarrow 0q_3\}$$

Finite automata from RG:-

- Let $G = (V, T, P, S)$ be a RG. We can construct DFA 'M' whose
1. states corresponds to variables 'V'
 2. starting state corresponds to start symbol's
 3. Transitions in 'M' corresponding to production Rules in 'P'.
 4. Input symbols corresponding to terminals in 'T'.
 5. If there is a production is of the form $q_i \rightarrow a$ then the transition is terminate at a new state called final state

Rules:-

* If the production rule is of the form $A \rightarrow \epsilon$ then 'A' is the final state. (A)

* A production rule is of the form $A_i \rightarrow a$ then there is a transition from A_i to final state labelled with 'a'.

$$A_i \rightarrow a = (A_i) \xrightarrow{a} (N_f)$$

* A production rule is of the form $A_i \rightarrow aA_j$ then there is a transition from A_i to A_j labelled with 'a'.

$$(A_i) \xrightarrow{a} (A_j)$$

* If a production rule is of the form $A_i \rightarrow a_1 a_2 a_3 \dots a_m$ A_j then there is a transition from A_i to A_j and add intermediate states labelled by $a_1, a_2, a_3, \dots, a_m$.

$$A_i \xrightarrow{a_1} (x_1) \xrightarrow{a_2} (x_2) \xrightarrow{a_3} (x_3) \dots \xrightarrow{a_m} (x_m) \xrightarrow{} A_j$$

Ex:- Construct a FA from the RG.

$$S \rightarrow a A/B$$

$$A \rightarrow aAB$$

$$B \rightarrow bB/a$$

Sol:- The Given $G = (V, T, P, S)$ where

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

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

$$P = \left\{ \begin{array}{l} S \rightarrow aA \quad B \rightarrow a \\ S \rightarrow B \end{array} \right\}$$

$$A \rightarrow aAB$$

$$B \rightarrow bB$$

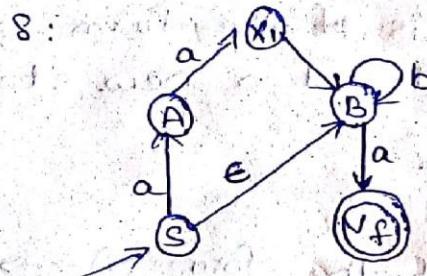
$$S = \{S\}$$

\therefore The equivalent DFA M is

$$M = (Q, \Sigma, S, q_0, F) \text{ where}$$

$$Q = \{S, A, B, V_f\}$$

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



$$Q = \{S, A, B, V_f, x_1\}$$

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

$$q_0 = \{S\}$$

$$F = \{V_f\}$$

2) Construct FA from the given RG.

$$S \rightarrow aA|\epsilon$$

$$A \rightarrow aA|bB|\epsilon$$

$$B \rightarrow bB|\epsilon$$

Sol: The given $G = (V, T, P, S)$

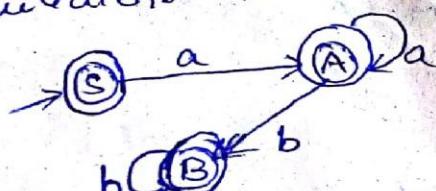
$$V = \{S, A, B\}$$

$$T = \{a, b, \epsilon\}$$

$$P = \left\{ \begin{array}{l} S \rightarrow aA \\ S \rightarrow \epsilon \\ A \rightarrow aA \\ A \rightarrow bB \\ A \rightarrow \epsilon \\ B \rightarrow bB \\ B \rightarrow \epsilon \end{array} \right\}$$

$$S = \{S\}$$

The equivalent DFA M is $M = (Q, \Sigma, S, q_0, F)$



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

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

$$q_0 = \{S\}$$

Regular Expression from Regular grammar:

Let $G = (V, T, P, S)$ be a R.G. Now the R.E 'R' is defined from 'G' by using the following rules.

* Replace the ' \rightarrow ' in the grammar productions with equal symbol ($=$) to get the set of equations.

* convert the equation of the form $A \rightarrow aA|ab$

\downarrow

$$A = a^*ab$$

* Repeat the step 2 until we get the R.E for the starting symbol. This gives the final R.E of given grammar 'G'.

To :- obtain the Regular Expression from the grammar given below.

$$S \rightarrow 0IB|0$$

$$B \rightarrow 1B|1$$

Sol :- The given Regular Grammar $G = (V, T, P, S)$

where

$$V = \{ S, B \}$$

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

$$P = \{ S \rightarrow 0IB \\ S \rightarrow 0 \}$$

$$B \rightarrow 1B$$

$$B \rightarrow 1 \}$$

$$S = \{ S \}$$

Replace arrow from set of productions with equal ($=$)

$$S = 0IB|0$$

$$B = 1B|1$$

\downarrow

$$B = 1^*1$$

$$B = 1$$

$$\text{Sub } B = 1 \text{ in } S = 0IB|0$$

$$S = 01^*1|0$$

The Final RE is

$$S = 01^*1 + 0$$

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

$$= 01^*$$

2)

$$S \rightarrow bas/aA$$

$$A \rightarrow bba/bb$$

Sol: The given RG $G = (V, T, P, S)$

where $V = \{S, A\}$, $T = \{a, b\}$

$$P = \{ S \rightarrow bas \}$$

$$S \rightarrow aA$$

$$A \rightarrow bba$$

$$A \rightarrow bb^*$$

Replace arrow from set of productions with '='

$$S = bas/aA$$

$$A = bba/bb$$

$$S = bas/aA$$

$$A = bb^*bb$$

$$A = bb^*$$

$$S = ba^*aA$$

$$\text{Sub } A = bb^*bb \text{ in } S = ba^*aA$$

$$S = ba^*abb^*bb$$

$$R.E = ba^*abb^*bb$$

$$= ba^*bb^*$$

Introduction:-

* Pumping Lemma is used for checking the given language Regular or not.

* Let $M = (Q, \Sigma, \delta, q_0, F)$ be a FA with n states.

* Let ' L ' be a Regular language accepted by ' M '

* Let ' w ' belongs to L ($w \in L$) and there exists x, y, z such that $w = xyz$ and $xyz \in L$ for each $|x|^i i \geq 0$.

Application:-

Step 1:- Assume ' L ' be regular language and ' n ' is the no. of states in the FA.

2:- choose the string ' w ' such that $|w| \leq n$. use pump lemma to write $w = xyz$. with the conditions

i) $|xy| < n$ ii) $|y| > 0$

3:- find a suitable integer 'i' such that $xy^i \notin L$. Hence, 'L' is not regular.

examples:-

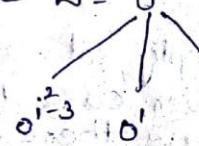
i) S.T the set $L = \{0^{i^2} \mid i \geq 1\}$ is not regular.

Sol:- Given $L = \{0^{i^2} \mid i \geq 1\}$

$$L = \{0^1, 0^4, 0^9, 0^{16}, \dots\}$$

$$L = \{0^1, 0^4, 0^9, 0^{16}, \dots\}$$

Assume $w = 0^{i^2}$



$\star y_3 = 3$

for $i=2$

$$\star y_3 = 0^{i^2-3} 0^2 0^2$$

$$= 0^{i^2+1}$$

$$= 0^{5+1}$$

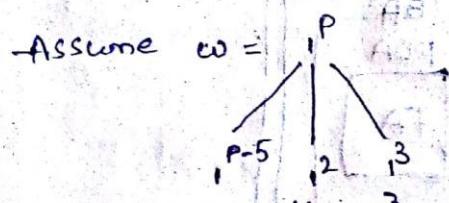
$$= 0^6 \notin L$$

The given set is not regular.

2) S.T the set $L = \{p^p \mid p \text{ is prime}\}$ is not regular.

Sol:- Given $L = \{p^p \mid p \text{ is prime}\}$

$$L = \{2^2, 1^3, 1^5, 1^7, 1^11, 1^13, \dots\}$$



for $i=2$

$$\star y_3 = 1^{p-5} (1^2)^2 (1^3)$$

$$= 1^{p-5} 1^4 1^3$$

$$\Rightarrow 1^{p+2} \notin L$$

$\therefore L$ is not regular.

INDEX

S.No	UNIT – II: Regular Expressions	Page No
1	Regular Expressions, Regular Sets, Properties of Regular Expressions	2
2	Identity Rules,	3
3	Manipulations of Regular Expressions (Algebraic laws)	4
4	Equivalence of two Regular Expressions,	4
5	Finite Automata, and Regular Expressions, Inter Conversion, Equivalence between Finite Automata and Regular Expressions	4
6	Pumping Lemma	5
7	Closure Properties	6
8	Applications of Regular Expressions	
9	Finite Automata and Regular Grammars	8
10	Regular Expressions and Regular Grammars	9

Regular Expressions :

Regular expressions are simple algebraic expressions that are used to represent a set of strings.

Recursive definition of regular expression:

For given input alphabet Σ , regular expression is defined by following rules:

1. Every character belonging to Σ , is a regular expression.
2. Null string ϵ is a regular expression.
3. If R_1 and R_2 are two regular expressions, then $R_1 + R_2$ is also a regular expression.
4. If R_1 and R_2 are two regular expressions, then $R_1 \cdot R_2$ is also a regular expression.
5. If R is a regular expression, then (R) is also a regular expression.

Formal Languages and Automata Theory – R16

6. If R is a regular expression, then R^* is also a regular expression.
7. Any combination of above rules is also a regular expression.

Regular Set or Regular Language :

Set of strings generated by regular expressions is known as regular set (or) regular language. It is the language accepted by Finite Automata.

Recursive definition of regular language from regular expression:

For given input alphabet Σ , regular expression is defined by following rules:

1. If Null string ϵ is a regular expression, then the regular language is $L=\{\epsilon\}$ which contains the empty string.
2. For every input symbol $a \in \Sigma$, a is a regular expression, then the regular set is $L=\{a\}$.
3. Let $R_1 = a$ and $R_2 = b$ and $R_1 + R_2$ is a regular expression, then regular expression is $a+b$ (means a or b) and regular set is $L=\{a,b\}$.
4. Let $R_1 = (a+b)$ and $R_2 = c$ and $R_1 \cdot R_2$ is a regular expression, then regular expression is $(a+b).c$ (means starting with a or b followed by c) and regular set is $L=\{ac,bc\}$.
5. Let $R=\{a\}$ and R^* is a regular expression (means any number of repetitions of a from 0 to ∞) then regular set is $L=\{\epsilon, a, aa, aaa, \dots\}$.
6. If ϕ is a regular expression, then regular language is $L=\{\}=\phi$ which denotes the empty set.

Properties of Regular Expressions:

If R_1 and R_2 are two regular expressions, then language generated by R_1 is $L(R_1)$ and language generated by R_2 is $L(R_2)$. The properties of regular expressions are

1. **Union:** It is a string from $L(R_1)$ or $L(R_2)$. It is denoted by $L(R_1 + R_2) = L(R_1) \cup L(R_2)$.
Example: let $R_1 = ab+c \Rightarrow L(R_1) = \{ab,c\}$
 $R_2 = d+ef \Rightarrow L(R_2) = \{d,ef\}$
Then $L(R_1 + R_2) = L(R_1) \cup L(R_2) = \{ab,c\} \cup \{d,ef\} = \{ab,c,d,ef\}$
2. **Concatenation:** It is a string from $L(R_1)$ followed by $L(R_2)$. It is denoted by $L(R_1 \cdot R_2) = L(R_1) \cdot L(R_2)$.
Example: let $R_1 = ab+c \Rightarrow L(R_1) = \{ab,c\}$
 $R_2 = d+ef \Rightarrow L(R_2) = \{d,ef\}$
Then $L(R_1 \cdot R_2) = L(R_1) \cdot L(R_2) = \{ab,c\} \cdot \{d,ef\} = \{abd,abef,cd,cef\}$
3. **Kleene Closure:** It is a string obtained by concatenating zero or more strings with repetitions. It is denoted by $L(R^*) = (L(R))^*$.
Example: let $R = ab+c \Rightarrow L(R) = \{ab,c\}$
Then $L(R^*) = (L(R))^* = \{ab,c\}^* = \{\epsilon, ab, c, abab, abc, cab, cc, ababc, \dots\}$
4. $L((R))=L(R)$ denotes same language as R .
5. $L(R).\epsilon = \epsilon \cdot L(R) = L(R)$.

Identity rules of Regular Expressions:

Two regular expressions R_1 and R_2 are equivalent if $L(R_1) = L(R_2)$. The identity rules of regular expressions are

1. $\epsilon + R = R + \epsilon$
2. $\epsilon \cdot R = R \cdot \epsilon = R$

Formal Languages and Automata Theory – R16

3. $R + R = R$
4. $\Phi + R = R + \Phi = R$
5. $\Phi \cdot R = R \cdot \Phi = \phi$
6. $\varepsilon^* = \varepsilon = \phi^*$
7. $R^* \cdot R^* = R^*$
8. $R \cdot R^* = R^* \cdot R$
9. $\varepsilon + R \cdot R^* = \varepsilon + R^* \cdot R = R^*$
10. $(R^*)^* = R^*$

PROOFS

	LHS	RHS	
$\varepsilon + R = R + \varepsilon$	$\begin{aligned} L(\varepsilon + R) &= L(\varepsilon) \cup L(R) \\ &= \{\varepsilon\} \cup L(R) \end{aligned}$	$\begin{aligned} L(R + \varepsilon) &= L(R) \cup L(\varepsilon) \\ &= L(R) \cup \{\varepsilon\} \end{aligned}$	Since $L(\varepsilon + R) = L(R + \varepsilon)$, therefore $\varepsilon + R = R + \varepsilon$
$\varepsilon \cdot R = R \cdot \varepsilon = R$	$\begin{aligned} L(\varepsilon \cdot R) &= L(\varepsilon) \cdot L(R) \\ &= \{\varepsilon\} \cdot L(R) \\ &= L(R) \end{aligned}$	$\begin{aligned} L(R \cdot \varepsilon) &= L(R) \cdot L(\varepsilon) \\ &= L(R) \cdot \{\varepsilon\} \\ &= L(R) \end{aligned}$	Since $L(\varepsilon \cdot R) = L(R \cdot \varepsilon) = L(R)$, therefore $\varepsilon \cdot R = R \cdot \varepsilon = R$
$R+R=R$	$L(R + R) = L(R) \cup L(R) = L(R)$		Since $L(R+R) = L(R)$, therefore $R+R=R$
$\Phi + R = R + \Phi = R$	$\begin{aligned} L(\Phi + R) &= L(\Phi) \cup L(R) \\ &= \{\}\cup L(R) \\ &= L(R) \end{aligned}$	$\begin{aligned} L(R + \Phi) &= L(R) \cup L(\Phi) \\ &= L(R) \cup \{\} \\ &= L(R) \end{aligned}$	Since $L(\Phi+R) = L(R+\Phi) = L(R)$, therefore $\Phi+R=R+\Phi=R$
$\Phi \cdot R = R \cdot \Phi = \phi$	$\begin{aligned} L(\Phi \cdot R) &= L(\Phi) \cdot L(R) \\ &= \{\} \cdot L(R) \\ &= \{\} \\ &= \Phi \end{aligned}$	$\begin{aligned} L(R \cdot \Phi) &= L(R) \cdot L(\Phi) \\ &= L(R) \cdot \{\} \\ &= \{\} \\ &= \Phi \end{aligned}$	Since $L(\Phi \cdot R) = L(R \cdot \Phi) = \Phi$, therefore $\Phi \cdot R = R \cdot \Phi = \phi$
$\varepsilon^* = \varepsilon = \phi^*$	$\begin{aligned} L(\varepsilon^*) &= (L(\varepsilon))^* \\ &= \{\varepsilon\}^* \\ &= \{\varepsilon, \varepsilon\varepsilon, \varepsilon\varepsilon\varepsilon, \dots\} \\ &= \{\varepsilon, \varepsilon, \varepsilon, \dots\} \\ &= \{\varepsilon, \varepsilon\varepsilon, \varepsilon\varepsilon\varepsilon, \dots\} \\ &= \{\varepsilon\} \\ &= L(\varepsilon) \end{aligned}$	$\begin{aligned} L(\phi^*) &= (L(\phi))^* \\ &= \{\}^* \\ &= \{\varepsilon\} \\ &= L(\varepsilon) \end{aligned}$	Since $L(\varepsilon^*) = L(\phi^*) = L(\varepsilon)$, therefore $\varepsilon^* = \varepsilon = \phi^*$
$R^* \cdot R^* = R^*$	$\begin{aligned} L(R^* \cdot R^*) &= L(R^*) \cdot L(R^*) \\ &= (L(R))^* \cdot (L(R))^* \\ &= (L(R))^* \cdot (\varepsilon, \varepsilon) \\ &= (L(R))^* \cdot (\varepsilon) \\ &= (L(R))^* \end{aligned}$		Since $L(R^* \cdot R^*) = (L(R))^*$ Therefore $R^* \cdot R^* = R^*$
$R \cdot R^* = R^* \cdot R$	$\begin{aligned} L(R \cdot R^*) &= L(R) \cdot L(R^*) \\ &= L(R) \cdot (L(R))^* \\ &= (R) \cdot \{\varepsilon, R, RR, \dots\} \\ &= \{R, RR, RRR, \dots\} \end{aligned}$	$\begin{aligned} L(R^* \cdot R) &= L(R^*) \cdot L(R) \\ &= (L(R))^* \cdot L(R) \\ &= \{\varepsilon, R, RR, \dots\}(R) \\ &= \{R, RR, RRR, \dots\} \end{aligned}$	Since $L(R \cdot R^*) = L(R^* \cdot R)$ Therefore $R \cdot R^* = R^* \cdot R$
$\varepsilon + R \cdot R^* = \varepsilon + R^* \cdot R = R^*$	$\begin{aligned} L(\varepsilon + R \cdot R^*) &= L(\varepsilon) \cup L(R \cdot R^*) \\ &= \{\varepsilon\} \cup \{R, RR, RRR, \dots\} \end{aligned}$	$\begin{aligned} L(\varepsilon + R^* \cdot R) &= L(\varepsilon) \cup L(R^* \cdot R) \\ &= \{\varepsilon\} \cup \{R, RR, RRR, \dots\} \end{aligned}$	Since $L(\varepsilon + R \cdot R^*) = L(\varepsilon + R^* \cdot R) = L(\varepsilon + R^*)$

Formal Languages and Automata Theory – R16

	$= L(R^*)$	$= L(R^*)$	$L(R^*)$ Therefore $\epsilon + R.R^* = \epsilon + R^*.R = R^*$
$(R^*)^* = R^*$	$\begin{aligned} L((R^*)^*) &= (L(R^*))^* \\ &= \{(L(R))^*\}^* \\ &= (R^*)^* \\ &= \{\epsilon, R, RR, \dots\}^* \\ &= \{\epsilon, R, RR, \dots\} \\ &= R^* \end{aligned}$	$L(R^*) = (L(R))^* = R^*$	Since $L((R^*)^*) = L(R^*)$ Therefore $(R^*)^* = R^*$

Algebraic laws of Regular Expressions:

1. Commutative law: $R + S = S + R$
2. Associative law: $(R+S)+T = R+(S+T)$
 $(R.S).T = R.(S.T)$
3. Distributive law: $(R+S)T = RT+ST$
 $(S+T)R = SR+TR$
4. Demorgan's law: $(R+S)^* = (R^*.S^*)^* = (R^*+S^*)^*$
 $(R.S)^* = (R^*.S^*)^* = (R^*+S^*)^*$
5. Other laws: $(RS)^*R = R(SR)^*$

Equivalence of two regular expressions:

For every regular expression, there is an equivalent finite automata. Two regular expressions are said to be equivalent, if same finite automata is constructed for both of the regular expressions.

Finite Automata, and Regular Expressions:

Finite Automata to Regular Expressions:

Arden's theorem: It is used to construct regular expression from finite automata.

Statement: If P, Q are two Res, and if P does not contain epsilon ϵ , then equation $R = Q + RP$ has a unique solution given by $R = QP^*$.

Steps to convert finite automata to regular expression:

1. Write the equation for each and every state using the following notation
Name of state = (state from which input is coming) (input symbol on the transition)
2. While writing the equation for starting state, include ϵ .
3. Make substitutions and apply ardens theorem wherever possible to obtain a regular expression.

Regular Expression to Finite Automata:

There are 2 ways to convert regular expression to finite automata. They are

1. Top-down approach
2. Bottom-up approach

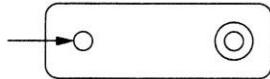
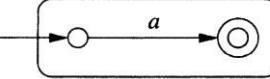
Top-down approach: FA construction starts from RE and ends by reaching a single element on transition.

Bottom-up approach: FA construction starts from basic element and ends on reaching RE.

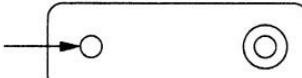
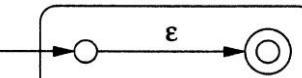
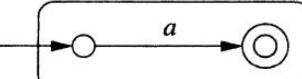
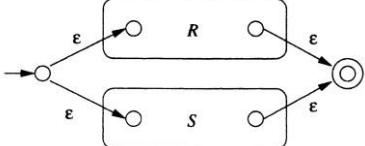
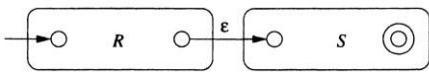
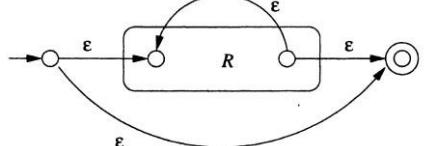
Steps to covert regular expression to finite automata:

Top-down approach:

Formal Languages and Automata Theory – R16

1	Take 2 states – start state q_0 and final state q_f .	
2	Make transition between these two states and place the given regular expression on the transition.	
3	If regular expression is in the form of $RE=R_1+R_2$, then finite automata is represented as	
4	If regular expression is in the form of $RE=R_1.R_2$, then finite automata is represented as	
5	If regular expression is in the form of $RE=R^*$, then finite automata is represented	
6	Make use of the above steps until the finite automata contains only single element on each transition.	

Bottom-up approach:

1	Divide the given regular expression into number of parts (Each part must be in the form of a RE.) and construct finite automata for each and every part separately using steps 2 to 7.	
2	Take 2 states – start state q_0 and final state q_f . Make transition between these two states and place the regular expression on the transition.	
3	If regular expression is in the form of $RE=\epsilon$, then finite automata is represented as	
4	If regular expression is in the form of $RE=a$, then finite automata is represented as	
5	If regular expression is in the form of $RE=R+S$, then finite automata is represented as	
6	If regular expression is in the form of $RE= R.S$, then finite automata is represented as	
7	If regular expression is in the form of $RE=R^*$, then finite automata is represented	
8	connect all the finite automata with an epsilon transition to obtain resultant finite automata.	

Pumping Lemma for regular languages:

Every language is not regular. To show that L is not regular, pumping lemma is used.

Pumping means “generating”. Pumping lemma means “it is a method of generating many input strings from a given string.” i.e., it is used to break a given long input string into several substrings.”

Statement of pumping lemma:

Formal Languages and Automata Theory – R16

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a finite automata with ‘n’ number of states.

1. Assume that the given language L is regular accepted by finite automata M.
2. Choose a string $w \in L$ such that $|w| \geq n$. i.e., length of string \geq number of states.
3. Break the string into 3 strings x,y,z, i.e., $w=xyz$ such that $|xy| \leq n$ and $y \neq \epsilon$.
4. By using pumping lemma, pump y to the suitable integer ‘i’ such that $xy^i z \notin L$ for $i \geq 0$. This contradicts our assumption.
5. Therefore L is not regular.

Closure Properties of Regular Languages:

Closure property: A set is closed under an operation if applying that operation to elements of the set results in an element of the set.

For example, the set of natural numbers is closed under addition and multiplication but not under subtraction or division because $2-3$ is negative and $1/2$ is not an integer. The set of integers is closed under addition, subtraction and multiplication but not under division.

Regular sets are closed under the following:

1. Union
2. Concatenation
3. Complementation
4. Intersection
5. Kleene closure
6. Substitution
7. Homomorphism
8. Inverse homomorphism
9. Quotient operation

Union: The class of regular languages is closed under union.

Statement: If L_1, L_2 are two regular languages, then $L_1 \cup L_2$ is also regular.

Proof: let r_1 is a regular expression denoting $L_1=L(r_1)$ and let
 r_2 is a regular expression denoting $L_2=L(r_2)$

$$\begin{aligned}L_1 \cup L_2 &= L(r_1) \cup L(r_2) \\&= L(r_1 + r_2)\end{aligned}$$

According to recursive definition of regular expression, if r_1 and r_2 are two regular expressions, then $r_1 + r_2$ is also a regular expression.

Since $r_1 + r_2$ is a regular expression and $L_1 \cup L_2 = L(r_1 + r_2)$, therefore $L_1 \cup L_2$ is also a regular.

Concatenation: The class of regular languages is closed under concatenation.

Statement: If L_1, L_2 are two regular languages, then $L_1 \cdot L_2$ is also regular.

Proof: let r_1 is a regular expression denoting $L_1=L(r_1)$ and let
 r_2 is a regular expression denoting $L_2=L(r_2)$

$$L_1 \cdot L_2 = L(r_1) \cdot L(r_2) = L(r_1 \cdot r_2)$$

According to recursive definition of regular expression, if r_1 and r_2 are two regular expressions, then $r_1 \cdot r_2$ is also a regular expression.

Since $r_1 \cdot r_2$ is a regular expression and $L_1 \cdot L_2 = L(r_1 \cdot r_2)$, therefore $L_1 \cdot L_2$ is also a regular.

Formal Languages and Automata Theory – R16

Complementation: *The class of regular languages is closed under Complementation.*

Statement: If $L(M)$ is a regular language, then $L(M')$ is also regular ($L = \Sigma^* - L$) .

Proof: Let $L(M)$ is regular and let $M=(Q,\Sigma,\delta,q_0,F)$ be a DFA for $L(M)$ and

$M'=(Q,\Sigma,\delta,q_0,Q-F)$ be a DFA for $L(M')$, complement of L .

If we construct a DFA for $L(M)$, then $w \in L$, if w leads to final state in M . This implies that in M' , w leads to the same state but this state is non-accepting state in M' . Therefore, M' rejects w . Similarly, if $w \notin L$, then M' accepts w . Therefore, $L(M') = L(M)$.

Since $L(M)$ is a regular language, therefore $L(M')$ is also regular

Intersection: *The class of regular languages is closed under Intersection.*

Statement: If L_1, L_2 are two regular languages, then $L_1 \cap L_2$ is also regular.

Proof: let r_1 is a regular expression denoting $L_1=L(r_1)$ and let

r_2 is a regular expression denoting $L_2=L(r_2)$

$$L_1 \cap L_2 = \overline{(\overline{t_1} \cup \overline{t_2})}$$

According to closure properties of regular languages, since L_1 and L_2 are two regular languages, its complement $\overline{t_1}$ and $\overline{t_2}$ are also regular. since $\overline{t_1}$ and $\overline{t_2}$ are two regular languages, its union $\overline{t_1} \cup \overline{t_2}$ is also regular. since $\overline{t_1} \cup \overline{t_2}$ are two regular languages, its complement $\overline{\overline{t_1} \cup \overline{t_2}}$ is also regular. Therefore $L_1 \cap L_2$ is also regular.

Kleene Closure: *The class of regular languages is closed under kleene closure.*

Statement: If L_1 is a regular language, then L_1^* is also regular.

Proof: let r_1 is a regular expression denoting $L_1=L(r_1)$ and

$$\begin{aligned} L_1^* &= (L(r_1))^* \\ &= L(r_1^*) \end{aligned}$$

According to recursive definition of regular expression, if r_1 is a regular expression, then r_1^* is also a regular expression.

Since r_1^* is a regular expression and $L_1^* = L(r_1^*)$, therefore L_1^* is also a regular.

Substitution: *The class of regular languages is closed under substitution.*

Statement: If L_1 is a regular language with some alphabet Σ , then L_1 is also regular with a unique replacement of Σ with Σ_1 .

Proof: let $\Sigma=\{a,b\}$, then a is a regular expression denoting $L_1=\{a\}$ and b is a regular expression denoting $L_2=\{b\}$ therefore $L=L_1.L_2=ab$ is a regular language over $\Sigma=\{a,b\}$. let $\Sigma_1=\{0,1\}$, on substituting ‘ a ’ with ‘ 0 ’ and ‘ b ’ with ‘ 1 ’. $L=01$ is also regular.

Homomorphism: *The class of regular languages is closed under homomorphism.*

Statement: If L_1 is a regular language with some alphabet Σ , and if strings in L_1 are replaced with other strings, then resultant language is also regular.

Proof: let $\Sigma=\{a,b\}$, then aba is a regular expression denoting $L_1=\{aba\}$ and $(aa)^*$ is a regular expression denoting $L_2=\{(aa)^*\}$ therefore $L=L_1.L_2=aba(aa)^*$ is a regular language over $\Sigma=\{a,b\}$.

Formal Languages and Automata Theory – R16

on substituting ‘aba’ with ‘0’ and ‘aa’ with ‘1’.

$$h(aba)=0$$

$h(aa)=1$ then 01^* is also regular.

Inverse Homomorphism: *The class of regular languages is closed under inverse homomorphism.*

Statement: It works by applying homomorphism backwards

Proof: let $h(aba)=0$

$$h(aa)=1 \text{ then}$$

$$h(01)=aba(aa)$$

$$h^{-1}(abaaa)=01 \text{ and } h^{-1}((abaaa + (aa)^*)^*)=(01+1^*)^*$$

Quotient of Language: *The class of regular languages is closed under quotient of a language.*

Statement: It works by applying homomorphism backwards

Proof: If L_1, L_2 are two regular languages, then Quotient denoted by L_1/L_2 is also regular.

$$L_1/L_2=\{x|\exists y \text{ in } L_2 \text{ and } \exists xy \text{ is in } L_1\}$$

$$\text{e.g: Let } L_1=0^*10^*, L_2=0^*1, \text{ then } L_1/L_2=0^*10^*/0^*1=0^*$$

Finite Automata and regular grammars:

Grammar: Grammar is a set of rules used to define a valid sentence in any language.

Mathematical representation of grammar: Grammar is defined as a 4-tuple $G=\{V,T,P,S\}$ where

V - finite set of non-terminals

T - finite set of terminals

P – finite set of production rules

S – Start symbol

“Non terminals” are represented by capital letters A, B, C,... so on.

“Terminals” are represented by small letters a, b, c,... so on.

“Production rules” are of the form $\alpha \rightarrow \beta$, where α contains non-terminal symbols or may also contain both terminals and non-terminals with atleast one non-terminal.
 β contains terminal, non-terminal symbols or any combination of both terminals and non-terminals.

Types of grammar:

1. Linear grammar-
Left linear grammar and Right linear grammar
2. Non-Linear grammar
3. Simple grammar
4. Recursive grammar
5. Regular grammar

Linear grammar: A grammar with atmost one variable (non-terminal) at right side of production rule is called linear grammar.

Example: $S \rightarrow aSb \quad S \rightarrow A \quad S \rightarrow Ab$
 $S \rightarrow \epsilon \quad A \rightarrow aB|\epsilon \quad A \rightarrow aAb|\epsilon$

Formal Languages and Automata Theory – R16

$B \rightarrow Ab$

Non linear grammar: A grammar with more than one variable (non terminal) at the right side of the production is called non linear grammar.

Example: $S \rightarrow SS$ $S \rightarrow aA$
 $S \rightarrow \epsilon$ $A \rightarrow \epsilon$
 $S \rightarrow aSb$ $S \rightarrow aSAS$
 $S \rightarrow bSa$

Simple or S- Grammar: A grammar is said to be S-Grammar if all the productions are of the form $A \rightarrow ax$, where $A \in V$ (variables)

$a \in T$ (terminals) $x \in V^*$ (0
or more variables)

and any pair (A,a) occurs atmost once in production p.

Example: $S \rightarrow aS$	$S \rightarrow aAS a$	$S \rightarrow aS$
$S \rightarrow bSS$	$A \rightarrow SbA$	$S \rightarrow bSS$
$S \rightarrow c$		$S \rightarrow aSS$
		$S \rightarrow c$
(s,a), (s,b) occur only once	(s,a) occur only once	(s,a) occur twice
∴ s-grammar	∴ s-grammar	∴ not s-grammar

Recursive grammar: A grammar is said to be recursive grammar, if it contains either a recursive production or an indirectly recursive production.

Example: $S \rightarrow aS$ $S \rightarrow aA|b$ $S \rightarrow SS$
 $A \rightarrow bS|c$ $S \rightarrow aSb$
 $S \rightarrow \epsilon$

Left linear grammar: A grammar $G=(V,T,P,S)$ is said to be left linear grammar if all the productions are of the form $A \rightarrow Bx$ or $A \rightarrow x$, where $A,B \in V$ (variables) $x \in T^*$ (0 or more terminals)

i.e., RHS non terminal is at left end.

Example: $S \rightarrow Ab$ $S \rightarrow Aab$ $S \rightarrow Sb$ $A \rightarrow Aab$
 $S \rightarrow \epsilon$ $A \rightarrow B$
 $B \rightarrow a$

Right linear grammar: A grammar is said to be right linear grammar if all the productions are of the form $A \rightarrow xB$ or $A \rightarrow x$, where $A,B \in V$ (variables) $x \in T^*$ (0 or more variables)

i.e., RHS non terminal is at right end.

Example: $S \rightarrow abS$ $S \rightarrow A$
 $S \rightarrow a$ $A \rightarrow aA$
 $A \rightarrow \epsilon$

Regular Grammar: A grammar $G=(V,T,P,S)$ is said to be regular grammar, if it is either right linear grammar or left linear grammar or nothing. i.e., productions are in the form of $A \rightarrow xB$ or $A \rightarrow Bx$ or $A \rightarrow x$ or $A \rightarrow \epsilon$, where $A,B \in V$ (variables) $x \in T^*$ (0 or more terminals)

i.e., LHS contains a single non terminal and

RHS contains only terminals or single non terminal
followed by terminals or terminals followed
by single non terminal or nothing.

Formal Languages and Automata Theory – R16

Example: $S \rightarrow abS$ $S \rightarrow Aab$ $S \rightarrow a$ $A \rightarrow Aab|B$
 $B \rightarrow a$

Relation between Regular Grammar and Finite Automata:

Conversion between regular grammar and FA:

1. RLG to FA
2. FA to RLG
3. RLG to LLG
4. LLG to RLG
5. LLG to FA
6. FA to LLG

RLG to FA:

1. Let $G=(V,T,P,S)$ be the RLG and M be the FA.
2. Start symbol of G is the start symbol of M .
3. Each variable in RLG will be a state in FA.
4. If variable has an ϵ -production $A \rightarrow \epsilon$, then that state A will become final state, otherwise a new final state will be introduced.
5. If production in G is of the form $A \rightarrow a_1 a_2 a_3 \dots a_n B$, then make transition
6. If production in G is of the form $A \rightarrow a_1 a_2 a_3 \dots a_n$, then make transition introducing a new final state.

FA to RLG:

1. Let $G=(V,T,P,S)$ be the RLG and M be the FA.
2. Start symbol of M is the start symbol of G .
3. For each transition (q_i, a, q_f) , write the production $q_i \rightarrow a q_f$. Now combine all the productions to form a RLG.
5. Make substitutions if possible to minimize the grammar.

RLG to LLG:

1. Let L be RLG. Construct FA for given RLG.
2. Replace the start state with final state and vice versa and reverse the direction of all the edges.
3. Construct grammar from step2
4. Reverse the right side of each production of RLG obtained in step3.
5. Resultant grammar is LLG.

LLG to RLG:

1. Reverse the right side of each production of LLG
2. Construct FA for obtained grammar in step1.
3. Replace the start state with final state and vice versa and reverse the direction of all the edges.
4. Construct grammar from FA obtained in step3
5. Resultant grammar is RLG.

LLG to FA:

$LLG \rightarrow RLG \rightarrow FA$ **FA**

to LLG:

FA → RLG → LLG

Page

Regular Expressions and Regular Grammars:

Conversion between regular grammar and regular expression:

1.RG to RE

2.RE to RG **RG to RE:**

1. Let $G=(V,T,P,S)$ be the RLG, replace \rightarrow symbol in the production with '=' symbol.
2. If there is a equation of the form $A = aA \mid b$, then convert it into $A = a^*b$
3. By applying substitutions, compute the expression for the starting symbol.

RE to RG:

RE → FA → RLG

