MODULE-2

Regular Expression and language

The language accepted by DFA or NFA and ϵ-NFA is called Regular language. A regular language can be described using regular expression consisting of two symbols such as alphabets in Σ, the operation such as '.', '+' and '*'.

1. The union of 2 languages L and M denoted as $L \cup M$., is the set of strings that are in either L or M or both.

$L = \{001, 10\}$ $M = \{\epsilon, 01\}$

$L \cup M = \{\epsilon, 01, 10, 001\}$

2. The concatenation of languages L and M is the set of strings that can be formed by taking a string in L and concatinating it with any string in M.

$L = \{001, 10, 111\}$ $M = \{\epsilon, 001\}$

$L.M = \{001, 10, 111, 001001, 10001, 111001\}$

ϵ is the identity to r concatenation.

3. The closure of a language L is denoted L^*, it represents the set of those strings that can be formed by taking any number of strings from L.

$L = \{0, 11\}$ L.L

$L^0 = \{\epsilon\}$, $L^1 = \{0, 11\}$, $L^2 = \{00, 011, 110, 1111\}$

Building Regular Expression

A regular expression can be formally defined as follows.

Basis:-

The basis consists of 3 parts

1. The constants ϵ and $\emptyset$ are regular expression denoting the language as $\{\epsilon\}$ and $\emptyset$ respectively. i.e, $L(\epsilon) = \{\epsilon\}$ and $L(\emptyset) = \emptyset$
2. If a is any symbol, then a is regular expression. This expression denotes the language $\{a\}$. i.e, $L(a) = \{a\}$
3. A variable, usually capital such as L is a variable representing any language.

Induction:-

There are 4 paths.

1. If E and F are regular expression then E+F; regular expression denoting the union of L(E) and L(F) that is $L(E+F)=L(E) \cup L(F)$.

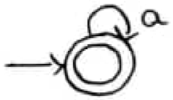
2. If E and F are regular expression then EF is a regular expression denoting the concatenation of L(E) and L(F), i.e, $L(EF)=L(E).L(F)$

3. If E is a regular expression, then E^* is a regular expression, denoting the closure of L(E) that is $L(E^*)=(L(E))^*$

4. If E is a regular expression then (E), a paranthesized E is also a regular expression denoting the same.

1. Obtain regular expression for the string consisting of any number of a's.

Regular Expression (RE) = a^*



2. Obtain regular expression for the string consisting of one a.

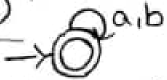
Regular expression (RE) = a^+

3. Obtain regular expression for string consisting of either one a or one b.

Regular expression (RE) = $a \cup b$ (or) $a+b$

4. Obtain regular expression for the string consisting of set of strings of a's and b's including empty strings.

Regular expression (RE) = $(a+b)^*$



5. Obtain regular expression for the string consisting of any number followed by any number of b's

Regular expression (RE) = a^*b^*

6. Obtain regular expression for string consisting of either any number of a's or b's.

Regular expression (RE) = a^*+b^*

7. Obtain regular expression for the string consisting of any number of ab's.

Regular expression (RE) = $(ab)^*$

8. Obtain a regular expression for the set of string of a's and b's ending with abb

Regular expression (RE) = $(a+b)^*abb$

9. Obtain a regular expression for the set of string of a's and b's beginning with abb

Regular expression (RE) = $abb(a+b)^*$

10. Obtain a regular expression for the set of strings of a's and b's having a sub-string abb.

Regular expression (RE) = $(a+b)^* abb (a+b)^*$

11. Obtain a regular expression for set of string consisting of any number of a's followed by any number of b's followed by any number of c's.

Regular expression (RE) = $a^* b^* c^*$

12. Obtain a regular expression for set of string consisting of atleast one a followed by atleast one b followed by atleast one c.

Regular expression (RE) = $a^+ b^+ c^+$

13. Obtain a regular expression for set of strings of a's and b's ending with either a string 'a' or 'bb'.

Regular expression = $(a+b)^* (a+bb)$

14. Obtain a regular expression for set of strings consisting of even number of a's

Regular expression = $(aa)^*$

15. Obtain a regular expression for set of strings consisting of odd number of a's

Regular expression = $a(aa)^*$

16. Obtain a regular expression for set of strings consisting of even number of a's followed by odd number of b's.

Regular expression = $(aa)^* \; b(bb)^*)$

17. Obtain a regular expression for set of strings consisting of even number of a's or even number of b's

Regular expression = $(aa)^* + (bb)^*$

18. Obtain a regular expression for set of strings of a's and b's whose length is 2.

Regular expression = $(a+b)(a+b) = (a+b)^2$

19. Obtain a regular expression for set of strings of a's and b's whose length is 100.

Regular expression = $(a+b)^{100}$

20. Obtain a regular expression for the following language.

i) $L = \{W \in \{a,b\}^* : |W| \bmod 2 = 0\}$

Regular expression = $((a+b)(a+b))^* = [(a+b)^2]^*$

ii) $L = \{W \in \{a,b\}^* : |W| \bmod 3 = 0\}$

Regular expression: $[(a+b)^3]^*$

21. To accept the strings having length less than (or) equal to 2 where $\Sigma = \{a,b\}$

Regular expression: $(\epsilon+a+b)^2$

22. To accept the strings having length less than (or) equal to 100. where $\Sigma = \{a,b\}$

Regular expression = $(\epsilon+a+b)^{100}$

23. To accept the strings of odd length where $\Sigma = \{a,b\}$

Regular expression = $(a+b)((a+b)^2)^*$

24. Obtain a regular expression to accept the strings starting with a and ending with b.

Regular expression = $a(a+b)^* b$

25. Obtain a regular expression to accept the string of a's and b's where second symbol is a.

Regular expression = $(a+b)\,a\,(a+b)^*$

26. Obtain a regular expression to accept the string of a's and b's where second symbol is a from right side

Regular expression = $(a+b)^*\,a\,(a+b)$

27. Obtain a regular expression to accept the string of a's and b's where tenth symbol is a

Regular expression = $(a+b)^9\,a\,(a+b)^*$

28. Obtain a regular expression to accept the language containing atleast one 'a' and atleast one 'b'. $\Sigma = \{a,b\}$

Regular expression = $(a+b)^* a (a+b)^* b (a+b)^* + (a+b)^* b (a+b)^* a (a+b)^*$

29. Obtain a regular expression to accept the language containing atleast one 'a' and atleast one 'b'. $\Sigma = \{a,b,c\}$

Regular expression = $(a+b+c)^* a\,(a+b+c)^* b (a+b+c)^* + (a+b+c)^* b\,(a+b+c)^* a (a+b+$

30. Obtain a regular expression for the language $L=\{W\in\{0,1\}^*: W$ does not have two consecutive zeroes$\}$.

Regular expression $=(1+01)^*(0+\epsilon)$

Step-1:- The string should end with either 11 or 01. Whenever 0 occurs it should be followed with 1 and no restriction on any number of one's. Then regular expression is $(1+01)^*$

Step-2:- Suppose, it should end with 10 means then regular expression is updated as $(1+01)^* 0$

Step-3:- If it should end with 11 or 01 or 10 means, then regular expression is $(1+01)^*(0+\epsilon)$

31. Obtain a regular expression for the language $L=\{W\in\{a,b\}^*:$ whose length is either even or multiples of $3\}$

Regular expression $=[(a+b)^2]^*+[(a+b)^3]^*$

32. Obtain a regular expression for the following language

i) $L=\{a^n b^m: m+n$ is even$\}$

Case-1:- both m and n are even

Regular expression $=(aa)^*(bb)^*$

Case-2:- both m and n are odd

Regular expression $=a(aa)^* b(bb)^*$

Final regular expression $=(aa)^*(bb)^*+a(aa)^* b(bb)^*$

ii) $L=\{a^{2n} b^{2m}: n\geq 0, m\geq 0\}$

Regular expression $=(aa)^*(bb)^*$

iii) $L=\{a^n b^m: n\geq 4, m\leq 3\}$

Regular expression $=aaaa a^*[(\epsilon+b)^3]$

iv) $L=\{a^n b^m: m\geq 1, n\geq 1, m\ \geq 3\}$

Case-1:- if $n=1$, $m=\geq 3$

Regular expression $=abbbb^*$

Case-2:- if $m=1$ then $n\geq 3$

Regular expression $=aaaa^* b$

Case-3:- if $n\geq 2$, then $m\geq 2$

Regular expression $=aaa^* bbb^*$

Final regular expression= $abbbb^* + aaaa^*b + aaa^*bbb^*$

Converting Regular Expressions to Automata (or) Kleen's Theorem:-

Theorem:-

Every language defined by a regular expression is also defined by a finite automata.

Proof:-

Suppose L=L(R) for a regular expression R, we show that L=L(E) for some ε-NFA E with

1) Exactly one accepting state

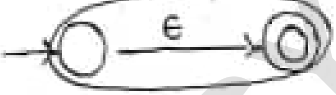
2) No arcs into initial state

3) No arcs out of the accepting state

Basis:-

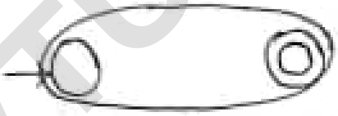
There are 3 paths to basis.

1) To handle the expression ε. The language of automata is ={ε}

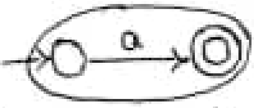


2) Construction for ∅

i.e, no path from start state to accepting state so ∅ is language of automation.

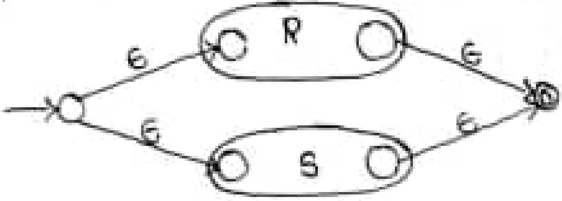


3) Give the automation for a regular expression a. The language of this automation consists of the one string a, i.e, L(a).



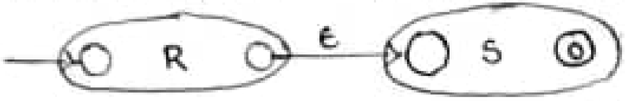
Induction:- The four parts of induction

1) The expression R+S for some smaller expressions R and S.



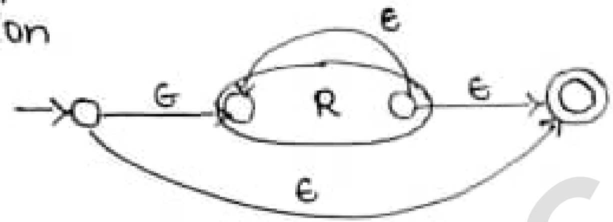
Starting at the new start state, we can go to the start state of either the automation for R or the automation for S. We then reach the accepting state of the automation for R+S. We can follow one of each ε arcs to the accepting state of new automation.

2> The expression is RS for some smaller expression R and S.
The automation for the concatenation is



The start state of the first automation because the start state of whole and the accepting state of the second automation because the accepting state of whole
L(R)L(S)

3> The expression is R^* for some smaller expression R. The automati
automation



The automation allows us to go either

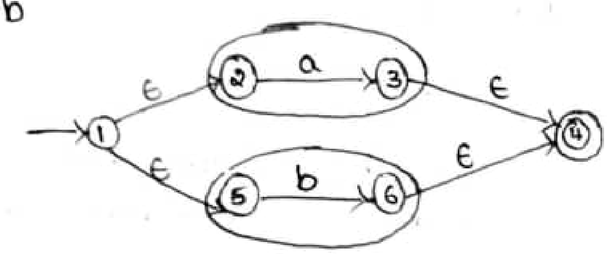
i> Directly from start state to the accepting state along a path labelled ε. The path let us accept ε which $L(R^*)$.

ii> To the start state of automata for R through that automata or more times and then to accepting state L(R) L(R) L(R) L(R)
and so on.

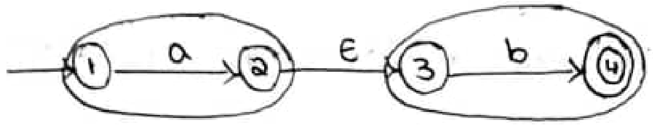
iii> The expression is R for some smaller R. The automation for R also saves as the automation for R. Since the paranthesis do not change the language defined by expression.

Convert the regular expression into finite automata.

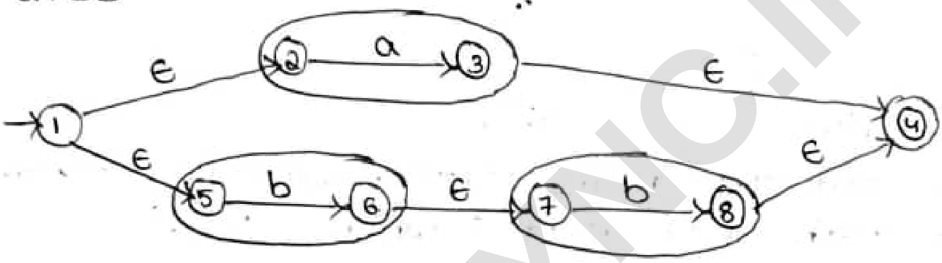
1. a+b



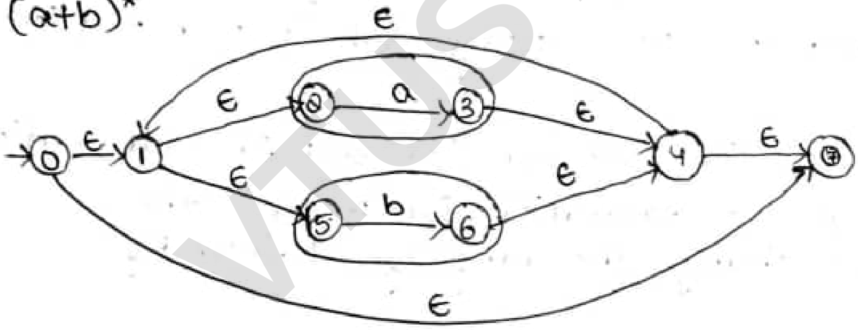
2. ab



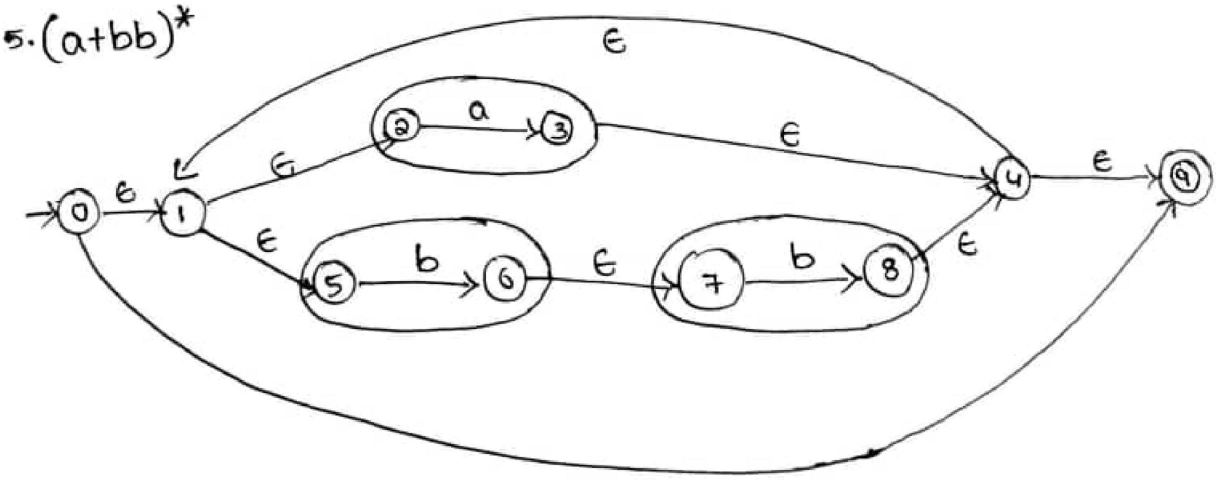
3. a+bb



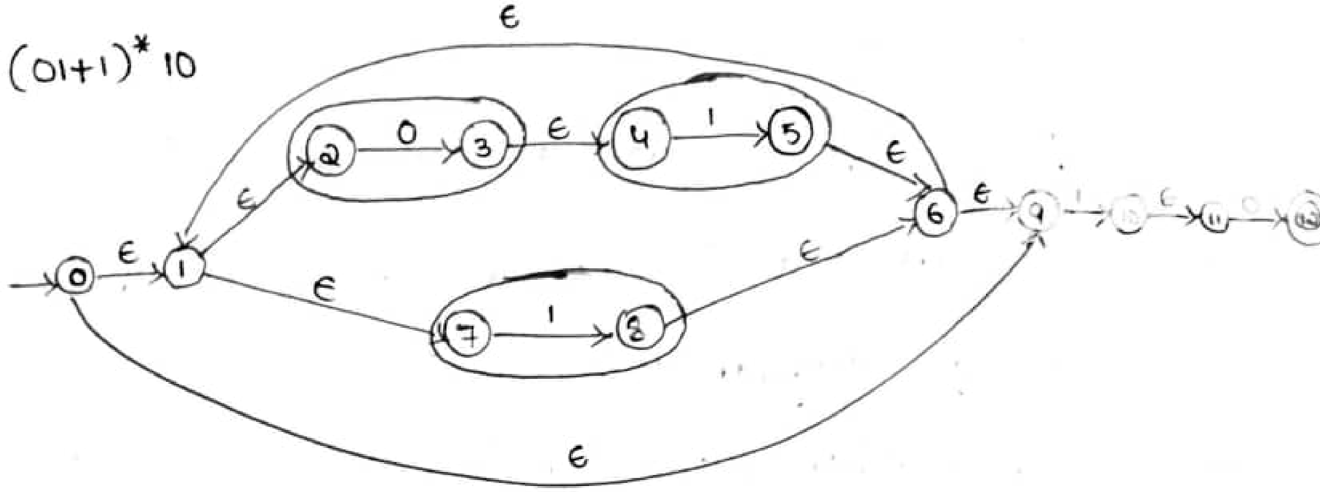
4. (a+b)*.



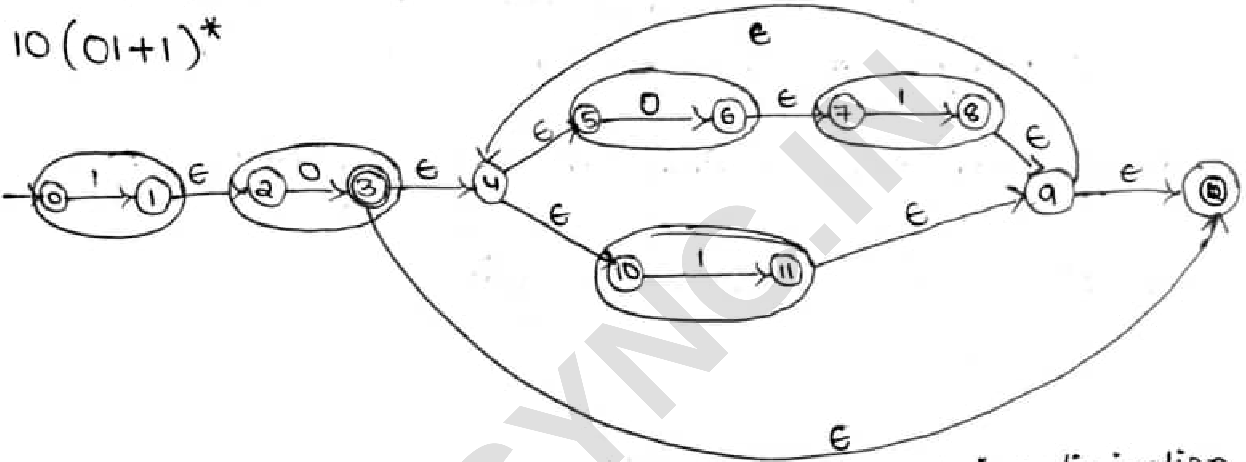
5. (a+bb)*



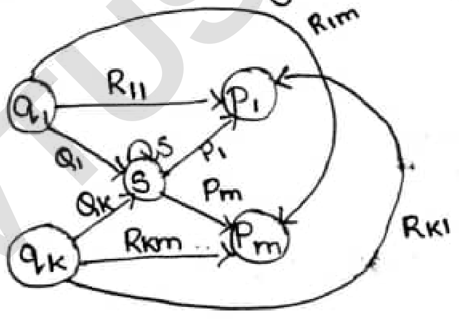
6. $(01+1)^* 10$



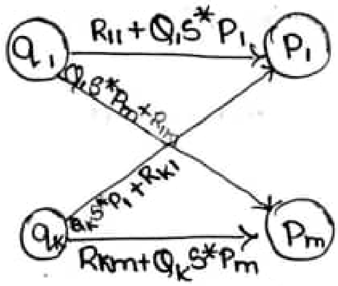
7. $10(01+1)^*$



Convertion from DFA to Regular Expression:- using elimination method:-

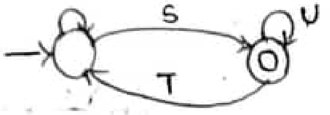


If we eliminate S



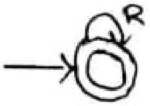
Step-1:- For each accepting state q apply the reduction process to produce an equivalent automation with Regular expression labels on the arc. Eliminate all states except q and start state q_0.

Step-2:- If $q \neq q_0$, then



Regular Expression = $(R+SU^*T)^*SU^*$

Step-3:- If start state is also an accepting state, then

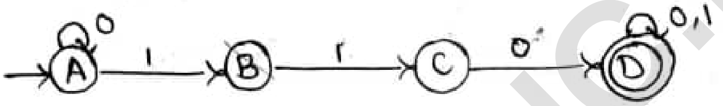


Regular expression = R^*

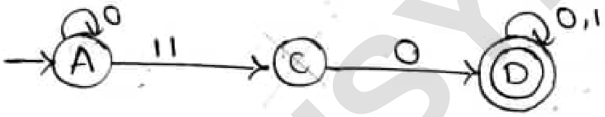
The desired regular expression is sum or union of all the expressions derived from the reduced automata from each accepting state from the above steps-2 and 3

Obtain Regular expression from the following NFA

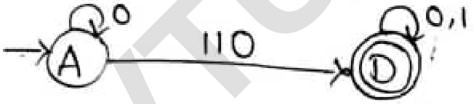
1.



Eliminate B

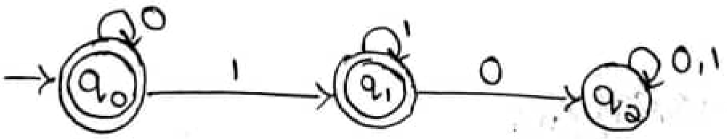


Eliminate C

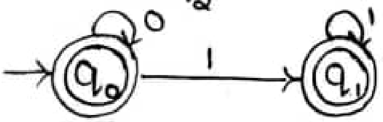


Regular expression :- $0^*110(0+1)^*$

2.



Eliminate q_2 as it is a dead state

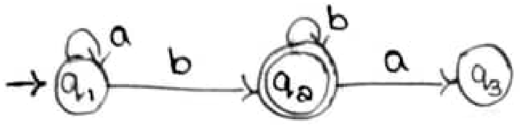


While eliminating a dead state we will delete the complete state, no arcs will be considered.

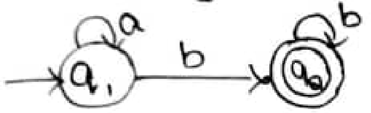
Regular expression:- $0^* + 0^*11^* = 0^*[\epsilon+11^*]$

$= 0^*[\epsilon+1^+] = 0^*1^*$

3.

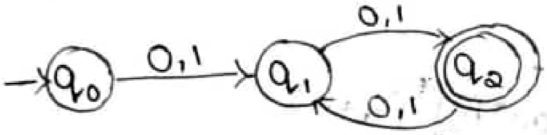


Eliminate q_3

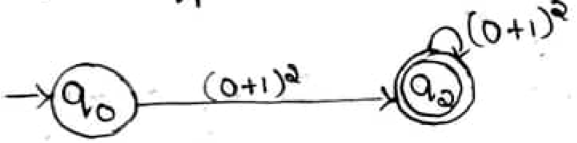


Regular expression :- $a^*bb^* = a^*b^+$

4.

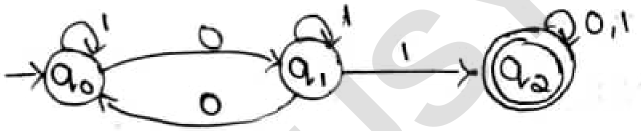


Eliminate q_1

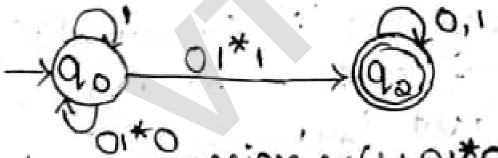


Regular expression:- $(0+1)^2 \left([0+1]^2\right)^* = \left[(0+1)^2\right]^+$

5.

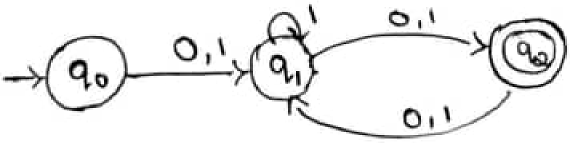


Eliminate q_1

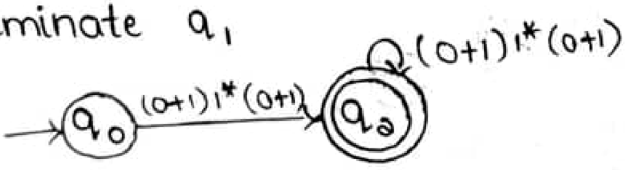


Regular expression :- $(1+01^*0)^* 01^*1 (0+1)^*$

6.

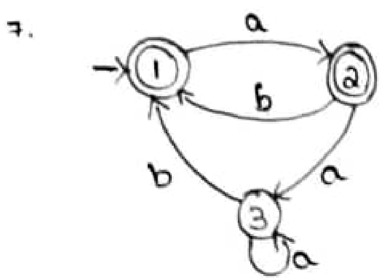


Eliminate q_1

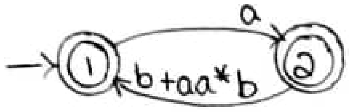


Regular expression :- $(0+1)1^*(0+1)\left[(0+1)1^*(0+1)\right]^*$

7.



Eliminate 3



Regular expression:- $[a[b+aa^*b]]^*(a+\epsilon)$

8

Proving languages not to be regular

* State and prove Pumping lemma theorem for regular languages.

Theorem statement:- Let L be a regular language then there exists a constant in such that for every string W in L such that $|W| \geq n$ we can break W into 3 strings, i.e,

$W = xyz$ such that

1) $y \neq \epsilon$ 2) $|xy| \leq n$ 3) $\forall k \geq 0$

the string xy^kz is also in L ; $xy^kz \in L$.

Proof:- Suppose L is a regular language $L = L(A)$ for some DFA A. Suppose A has n states consider W of length n $W = a_1 a_2 ---- a_n$ where $m \geq n$ and each a_i is an input symbol i.e,

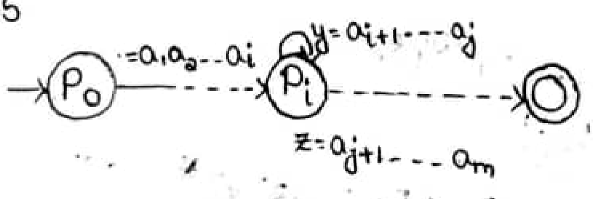
$W = xyz$ where

$x = a_1 a_2 -- a_i$

$y = a_{i+1} --- a_j$

$z = a_{j+1} --- a_m$

x takes



x takes us from P_0 to P_i once y takes us from P_i to P_i ($P_i = P_j$) and z is balance of ~~20~~W

Consider xy^kz for any $k \geq 0$, if $k=0$ automata goes from start state P_0 to P_i as input x since $P_i = P_j$ it must go from P_i to accepting state, i.e, on input z.

If $k>0$, automata (A) goes from P_0 to P_i on input x, circles from P_i to P_ik k times on input y and goes to accepting state on input z, i.e, if $k \geq 0$ xy^kz is also accepted by A.

$$xy^kz \in A$$

1. Show that the language $L = \{a^nb^n \mid n \geq 0\}$ is not regular.

Note:- For the given language, DFA cannot be constructed. So because of that we can tell the given language is not regular.

$L = \{a^nb^n \mid n \geq 0\}$

Let $n=3$

$W = \underline{\underset{x}{aa}\underset{y}{a}\underset{z}{bbb}}$

$|W| \geq n$

$6 \geq 3$

1) $|xy| \leq 3$

$3 = 3$

2) $y \neq \epsilon$

$y = a$

3) $\forall\ xy^kz \in L$

$k=0$ $xy^0z = aa\epsilon bbb = aabbb \notin L$

$k=1$ $xy^1z = aaabbb \in L$

$k \geq 2$ $xy^2z = aaaabbb \notin L$

For $k=0, k>1$ we are not getting equal number of a's and b's. So, This is contradiction to the assumption. So, the given language is not regular.

2. Show that the language $L = \{a^nb^nc^n \mid n \geq 0\}$ is not regular.

$L = \{a^nb^nc^n \mid n \geq 0\}$

Let $n=3$

$W = \underline{a\overset{x}{a}\overset{y}{a}bbbccc}$

$|W| \geq n$

$9 > 3$

1) $|xy| \leq 3$

$3 = 3$

2) $y \neq \epsilon$

$y = a$

3) $\forall\ xy^kz \in L$

$k=0 \quad xy^0z = aa\epsilon bbbccc \notin L$

$k=1 \quad xy^1z = aaabbbccc \in L$

$k\geq 2 \quad xy^2z = aaaabbbccc \notin L$

For $k=0$ and $k\geq 2$, we are not getting equal number of a's, b's and c's. This is contradiction to our assumption. So, the given language is not regular.

3. Show that the language $L = \{WW^R : W \in \{a,b\}^*\}$

$W = abb$

$W^R = bba$

$WW^R = \underset{x}{\underline{a}}\overset{y}{\underline{b}}\underline{bbba}\ z$

$n = 3$

$|W| \geq n$

$6 > 3$

1) $|xy| \leq n$

$3 = 3$

2) $y \neq \epsilon$

$y = b$

3) $\forall\ xy^kz \in L$

$k=0 \quad xy^0z = ab\epsilon bba \notin L$

$k=1 \quad xy^1z = abbbba \in L$

$k\geq 2 \quad xy^2z = abbbbba \notin L$

For $k=0$ and $k\geq 2$, we are not getting.
This is contradiction to our assumption. So, the given language is not regular.

4. Show that the given language $L = \{W \in \{a,b,c\}^* : N_a(W) = N_b(W) = N_c(W)\}$

$W = \overset{x}{a}\overset{y}{b}bacaccb$

Let $n = 3$

$|W| \geq n$

$9 > 3$

1> $|xy| \leq n$

$3 = 3$

2> $y \neq \epsilon$

$y = b$

3> $\forall\ xy^kz \in L$

$k=0 \quad xy^0z = ab\epsilon acaccb$

$k=1 \quad xy^1z = abbacaccb$

$k \geq 2 \quad xy^2z = abbbacaccb$

For $k=0$ and $k \geq 2$, we are not getting equal number of b's. This is contradiction to our assumption. So, the given language is not regular.

5. Show that language $L = \{WW \mid W \in \{a,b\}^*\}$ is not regular

$W = aba$

$WW = abaaba$

Let $n = 3$

$|W| \geq n$

$6 > 3$

1> $|xy| \leq n$

$3 = 3$

2> $y \neq \epsilon$

$y = a$

3> $\forall\ xy^kz \in L$

$k=0 \quad xy^0z = ab\epsilon aba \notin L$

$k=1 \quad xy^1z = abaaba \in L$

$k \geq 2 \quad xy^2z = abaaaba \notin L$

For $k=0$ and $k \geq 2$, we are not getting the string as per condition. This is contradiction to our assumption. So, the given language is not regular

6: Show that the language $L=\{a^n \mid n \text{ is a prime}\}$ is not regular

$L=\{a^n \mid n \text{ is a prime}\}$

Let $n=5$

$W=aaaaa$

$|W| \geq n$

$5=5$

1) $|xy| \leq n$

$3<5$

2) $y \neq \epsilon$

$y=a$

3) $\forall\ xy^kz \in L$

$k=0 \quad xy^0z = aa\epsilon aa \notin L$

$k=1 \quad xy^1z = aaaaa \in L$

$k \geq 2 \quad xy^2z = aaaaaa \notin L$

For $k=0$ and $k \geq 2$,

7. Show that the given language $L=\{a^{n!} \mid n \geq 0\}$ is not regular

$W=\underline{aaaaaa}$

x y z

$n=3$

$|W| \geq n$

$6>3$

1) $|xy| \leq n$

$3=3$

2) $y \neq \epsilon$

$y=a$

3) $\forall\ xy^kz \in L$

$k=0 \quad xy^0z = aa\epsilon aaa \notin L$

$k=1 \quad xy^1z = aaaaaa \in L$

$k \geq 2 \quad xy^2z = aaaaaa \notin L$

8. Show that given language L = { Balanced paranthesis } is not regular

$\{, \}, [,]$ $\quad n = 4$

$W = \underbrace{\{[}_{x}\underbrace{]}_{y}\underbrace{\}[\{\}]}_{z}$

$|w| \geq n$

$8 > 4$

1) $|xy| \leq n$

$3 < 4$

2) $y \notin \epsilon$

$y =]$

3) $\forall \ xy^kz \in L$

$k = 0 \quad xy^0z = \{[\}[\{\}] \notin L$

$k = 1 \quad xy^1z = \{[]\}[\{\}] \in L$

Minimization of DFA:-

Step-1:-Find the distinguishable and indistinguishable pairs using table filling algorithm

Step-2:-Obtain the states of minimized DFA

Step-3:- Compute the transition table if $[P_1, P_2, ----P_k]$ is a group and if $\delta([P_1, P_2, -----P_k], a) = [r_1, r_2, ----r_m]$,then place the edge from the group $[P_1, P_2, --P_k]$ to the group $[r_1, r_2, --r_m]$ and label the edge with symbol 'a'. Follow this procedure for each group obtained in Step-2 and for each $a \in \Sigma$

Step-4:-Identify the start state if arc of the component in the group $[P_1, P_2 --P_k]$ consists of a start of given DFA ,then $[P_1, P_2, ----P_k]$ is the start state of minimized DFA

Step-5:-Identify the final state. If the group $[P_1, P_2, ------P_k]$ contains a final state of given DFA ,then the group $[P_1, P_2, ----P_k]$ is a final state of minimized DFA.

1. Obtain the distinguishable table for automation and minimize the state for following DFA.

δ	a	b
→A	B	F
B	G	C
*C	A	C
D	C	G
E	H	F
F	C	G
G	G	E
H	G	C

Step-1:- Vertically we will write all the states from second state to last state (B,C,D,E,F,G,H). Horizontally we will write all the states to last state but one.

B							
* C							
D							
E							
F							
G							
H							
	A	B	C *	D	E	F	G

Step-2:- Since, state C is a final state, the pair (A,C),(B,C) has one final state and other one is non-final state so we will marks the pairs (A,C),(B,C) horizontally. The pairs (D,C),(E,C),(F,C),(G,C),(H,C) has one final and one non-final state.

[Internal marking with final state]

B							
*C	X	X					
D			X				
E			X				
F			X				
G			X				
H			X				
	A	B	C *	D	E	F	G

Step-3 [Marking with final and non final state and transition]:-

(B,G) both non-final so we take and check for its transition

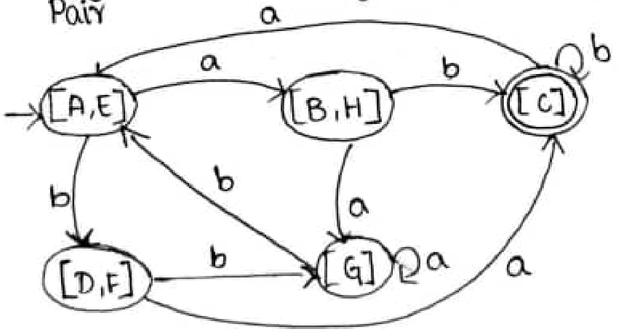
	A	B	C	D	E	F	G
B	X						
*C	X	X					
D	X	X	X				
E	①	X	X	X			
F	X	X	X	④	X		
G	②	X	X	X	⑤	X	
H	X	③	X	X	X	X	X

Step-4:- Mark with internal checkings.

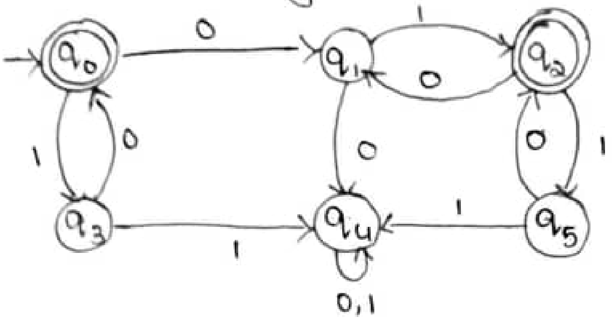
	A	B	C*	D	E	F	G
B	X						
*C	X	X					
D	X	X	X				
E	①	X	X	X			
F	X	X	X	③	X		
G	X	X	X	X	X	X	
H	X	②	X	X	X	X	X

Note:- (A,E) is not marked, check for its transition states, if it is not marked, give it a numbering.

$\underbrace{[A,E],[B,H],[D,F]}_{\text{Undistinguishable Pair}},\underbrace{[C],[G]}_{\text{Distinguishable pair}}$



2. Minimize the following DFA



	0	1
→*q_0	q_1	q_3
q_1	q_4	q_2
*q_2	q_1	q_5
q_3	q_0	q_4
q_4	q_4	q_4
q_5	q_2	q_4

Step-1 & 2 :-

q_1	X				
*q_2		X			
q_3	X		X		
q_4	X		X		
q_5	X		X		
	q_0 *	q_1	q_2 *	q_3	q_4

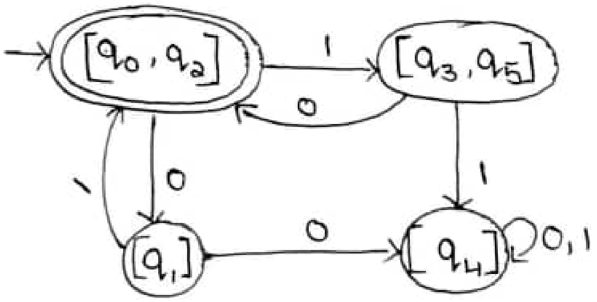
Step-3:-

	q_0 *	q_1	q_2 *	q_3	q_4
q_1	X				
* q_2	①	X			
q_3	X	X	X		
q_4	X	X	X	X	
q_5	X	X	X	②	X

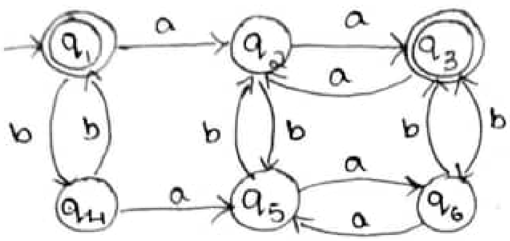
Step-4:-

	q_0 *	q_1	q_2 *	q_3	q_4
q_1	X				
* q_2	①	X			
q_3	X	X	X		
q_4	X	X	X	X	
q_5	X	X	X	②	X

$$\underbrace{[q_0, q_2], [q_3, q_5]}_{\text{Indistinguishable pairs}}, \underbrace{[q_1], [q_4]}_{\text{Distinguishable pairs}}$$



3. Minimize the following DFA



	a	b
→*q_1	q_2	q_4
q_2	q_3	q_5
*q_3	q_2	q_6
q_4	q_5	q_1
q_5	q_6	q_2
q_6	q_5	q_3

Step-1 & 2:-

q_2	X				
*q_3		X			
q_4	X		X		
q_5	X		X		
q_6	X		X		
	q_1 *	q_2	q_3 *	q_4	q_5

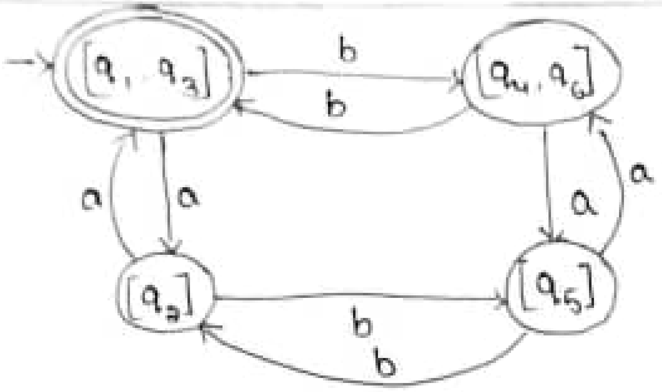
Step-3:-

q_2	X				
*q_3	①	X			
q_4	X	X	X		
q_5	X	X	X	X	
q_6	X	X	X	②	X
	q_1 *	q_2	q_3 *	q_4	q_5

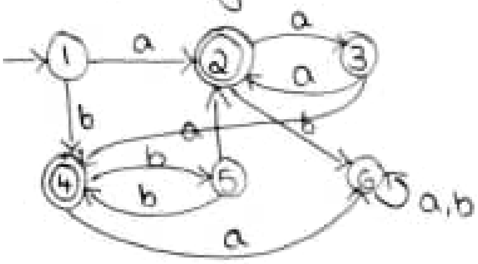
Step-4:-

q_2	X				
*q_3	①	X			
q_4	X	X	X		
q_5	X	X	X	X	
q_6	X	X	X	②	X
	q_1 *	q_2	q_3 *	q_4	q_5

$$\underbrace{[q_1, q_3], [q_4, q_6]}_{\text{Indistinguishable pairs}}, \underbrace{[q_2], [q_5]}_{\text{distinguishable pairs}}$$



4. Minimize the following DFA.



	a	b
→1	2	4
*2	3	6
3	2	4
*4	6	5
5	2	4
6	6	6

Step-1 & 2:-

*2	X				
3		X			
*4	X		X		
5		X		X	
6		X		X	
	1	2 *	3	4 *	5

Step-3:-

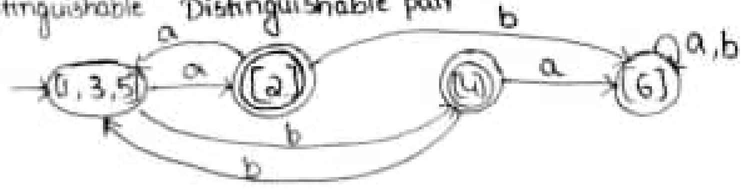
*2	X				
3	①	X			
*4	X	③	X		
5	②	X	④	X	
6	X	X	X	X	X
	1	2 *	3	4 *	5

Step-4:-

*2	X				
3	①	X			
*4	X	X	X		
5	②	X	③	X	
6	X	X	X	X	X
	1	2 *	3	4 *	5

$\underbrace{[1,3,5]}_{\text{Indistinguishable pair}}, \underbrace{[2],[4],[6]}_{\text{Distinguishable pair}}$

$\left.\begin{matrix}[1,3]\\ [1,5]\\ [3,5]\end{matrix}\right\} \rightarrow$ Indistinguishable pair



All 3 indistinguishable states [1,3],[1,5],[3,5] are behaving same i.e state {1,3,5} on a is reaching to state (2) and on b it is reaching to state (4). So we can make all three indistinguishable states as a single indistinguishable state

Closure properties of Regular Language :-

If certain languages are regular and a language L is formed from them by certain operations, then L is also regular

The operations of closure properties for a regular language are

1. Union
2. Intersection
3. Complement
4. Difference
5. Reversal
6. Closure (star)

1. Union operation :- (Closure under union)

Let L and M be the languages over alphabet Σ. Then, LUM is a language that contains all strings that are in either or both of L and M.

Proof :-

Since, L and M are regular, they have regular expression say L=L(R) M=L(S), then LUM = L(R)+L(S) by the definition of '+' operator for Regular expression.

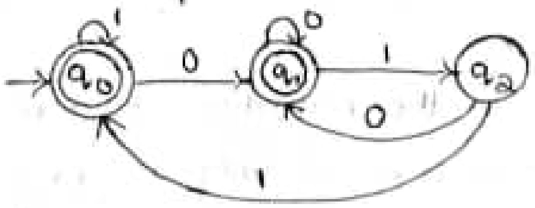
2. Closure under complementation :-

Complementation can be done by using two methods.

1. By using DFA
2. Finding regular expression for its complement

1. By using DFA :-

DFA accepting the complement of a language $(0+1)^{*}01$



2. Finding regular expression for its complement:-

If L is a regular language over alphabet Σ then $\bar{L} = \Sigma^* - L$ is also a regular language.

Proof:-

Let $L = L(A)$ for some DFA $A = (Q, \Sigma, \delta, q_0, F)$, then $\bar{L} = L(B)$: $B = (Q, \Sigma, \delta, q_0, Q - F)$ that is B is exactly like A but the accepting states of A have become non-accepting states of B and vice versa, then W is in L(B) if and only if $\hat{\delta}(q_0, W)$ is in $Q - F$ which occurs if and only if W is not in L(A).

3. Closure under intersection:-

Let us consider the intersection of two regular languages say L and M that can be obtained the intersection of languages by L and M

$L \cap M = \overline{\bar{L} \cup \bar{M}}$, this equation says it is one of DeMorgan's law, i.e. we have already proved that the operations union and complement are closed under regular languages. By using these two operation, we have proved that intersection operation is closed.

4. Closure under difference:-

If L and M are regular language, so is L-M

$$L - M = L \cap \bar{M}$$

We have already proved that the operations intersection and complement are closed under regular language, by using these two operations, we have proved that difference operation is closed.

5. Closure under reversal:-

The reversal of a string $a_1, a_2, \ldots a_n$ is the string written backwards that is, $a_n, a_{n-1} \ldots a_1$. We use W^R for the reversal of string W. Thus, 0010^R, and $e^R = \epsilon$. There are two simple proofs, one based on automata and one based on regular expression.

The reversal of a language L, written L^R, is the language consisting of the reversals of all its strings. For instance, if $L = \{001, 10, 111\}$, then $L^R = \{100, 01, 111\}$

Given a language L that is L(A) for some finite automation, perhaps with non-determinism and ϵ-transitions, we may construct an automation for L^R by:

1. Reverse all the arcs in the transition diagram for A.

2. Make the start state of A be the only accepting state for the new

automation.

3. Create a new start state p_0, with transitions on ϵ to all accepting states of A.

The result is an automation that simulates A "in reverse" and therefore accepts a string W if and only if A accepts W^R. Now, we prove the reversal theorem formally.

Theorem:- If L is a regular language, so is L^R.

Basis:- If E is ϵ, $\emptyset$ or a, for some symbol a, then E^R is the same as E. That is, we know $\{\epsilon\}^R = \{\epsilon\}$, $\emptyset^R = \emptyset$ and $\{a\}^R = \{a\}$.

Induction:- There are three cases, depending on the form of E.

1. $E = E_1 + E_2$. Then $E^R = E_1^R + E_2^R$. The justification is that the reversal of the union of two languages is obtained by computing the reversals of the two languages and taking the union of those languages.
2. $E = E_1 . E_2$. Then $E^R = E_2^R E_1^R$. Note that we reverse the order of the two languages, as well as reversing the languages themselves. For instance if $L(E_1) = \{01, 111\}$ and $L(E_2) = \{00, 10\}$, then $L(E_1 E_2) = \{0100, 0110, 11100, 11110\}$. The reversal of the latter language is.

   $\{0010, 0110, 00111, 01111\}$

   If we concatenate the reversals of $L(E_2)$ and $L(E_1)$ in the order, we get $\{00, 01\}\{10, 111\} = \{0010, 0110, 00111, 01111\}$ which is the same language as $(L(E_1 E_2))^R$. In general, if a word W in L(E) is the contatenation of W_1 from $L(E_1)$ and W_2 from $L(E_2)$, then $W^R = W_2^R W_1^R$.
3. $E = E_1^*$. Then $E^R = (E_1^R)^*$. The justification is that any string W in L(E) can be written as $W_1 W_2 \text{-----} W_n$, where each W_i is in L(E). But $W^R = W_n^R W_{n-1}^R \text{----} W_1^R$.

$(L(E_1) L(E_2))^2$ $L(E_1 E_2$