Unit: 1.

imp (6M) Central Concepts of Automata & Theory.

1) Alphabets: It is a finite non-empty set of symbols Alphabet set is represented by Σ (sigma).

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

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

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

$\Sigma = \{A, B, C, \ldots Z\}$

2) Strings: It is a finite sequence of symbols chosen from some alphabet.

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

$w = 0\ 1\ 1\ 0 \rightarrow$ This string because it is present.

$w = 0\ 1\ 1\ 2 \rightarrow$ NO, because it is absent.

3) Empty String: String with zero(0) appearence of symbols on certain alphabet. Denoted as ϵ epsilion.

Ex: $\Sigma = \{\ \}$

$w = \epsilon$

4) Length of string: It is the no. of symbols appearence in given string.

Ex: $w = 0110$

$|w| = 4.$

5) Concatenation of string: Combining of two strings.

Ex: $x = 010$, $y = 110$

$x, y = 010110$

$y, x = 110010$

$\Rightarrow x.\epsilon = \epsilon.x = x$ because it is a empty set.

6) Power of an alphabet - If ϵ is an alphabet exponential notation can be used to express the set of all string of certain length from that alphabet.

$\Sigma^k \Rightarrow$ is the set of string of length k each of whose symbol is sigma (Σ).

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

$\Sigma^0 = \{\epsilon\}$

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

$\Sigma^2 = \{00, 01, 10, 11\}$

$\Sigma^3 = \{000, 001, 010, 100, 110, 011, 101, 111\}$

* Types of Automata

(i) The Deterministic Automata (DFA):

Is defined as 5-tuple or a uni-tuple indicating five component.

$D = \{Q, \Sigma, \delta, q_0, F\}$

where,

1. Q is a finite set of states.
2. Σ is a finite set of input symbol.
3. δ is the transition fun that takes a state in Q & on i/p symbol in Σ as arguments & returns a state

$$\delta : Q \times \Sigma \rightarrow Q$$

4. q_0 is the start state that belongs to Q.
5. F is subset of Q, is the set of final (or accepting) states.

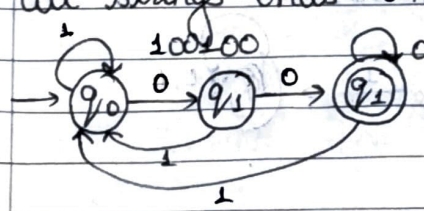
$\Rightarrow$ 2 ways to perform:

a) 

Transition Diagram

all state final state

Eg: Construct DFA on $\Sigma = \{0, 1\}$ such that all strings ends with 00.



* $w = 10010$

$q_0 \xrightarrow{0} q_0 \xrightarrow{0} q_1 \xrightarrow{0} q_2 \xrightarrow{1} q_0 \xrightarrow{0} q_1$

$\omega = 0100$

$q_0 \xrightarrow{0} q_1 \xrightarrow{1} q_0 \xrightarrow{0} q_1 \xrightarrow{0} q_2$

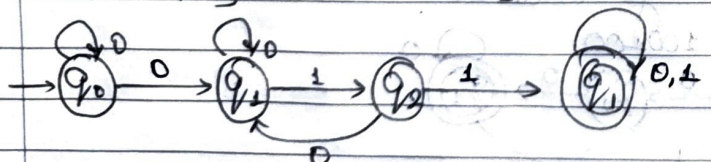
Note: If a stack 'q' is final state then in 'ω' is accepted or rejected.

b) Transition Table - Rows corresponds to the states in Q from 'TT'
* Columns corresponds the input from Σ.
* The entry for the row corresponds to the state 'q' & the column corresponds to input a is the state 'p' from δ(q, a)

Ex: TT from previous

	0	1
q_0	q_1	q_0
q_1	q_2	q_0
q_2	q_2	q_0

Q1. Construct a DFA to except all the binary string where 011 as sub string
$\Sigma = \{0, 1\}$ $\omega = 0011$



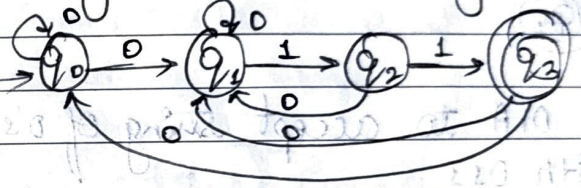
$\omega = 01011$

$q_0 \xrightarrow{0} q_1 \xrightarrow{1} q_2 \rightarrow q_1 \rightarrow q_2$

	0	1
q_0	q_1	q_0
q_1	q_1	q_2
q_2	q_1	q_3
q_3	q_3	q_3

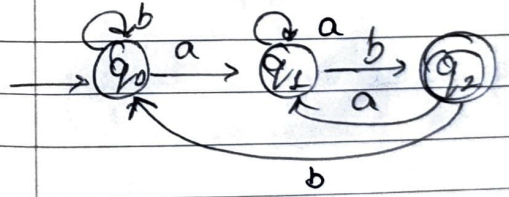
Q2. Construct a DFA to except all one binary string where all string ending with 011.

→



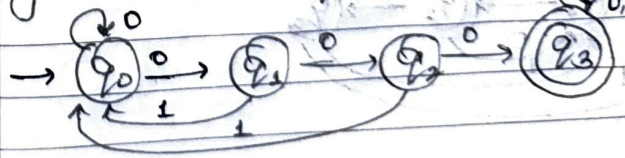
Q3. Construct a DFA an alphabet a ending with string a,b.

→ $\Sigma = \{a, b\}$



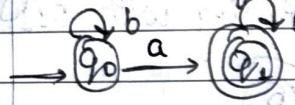
Q4. Set of all strings with three consecutive zeroes with alphabet 0 & 1 ω = 1000, 00100100

→

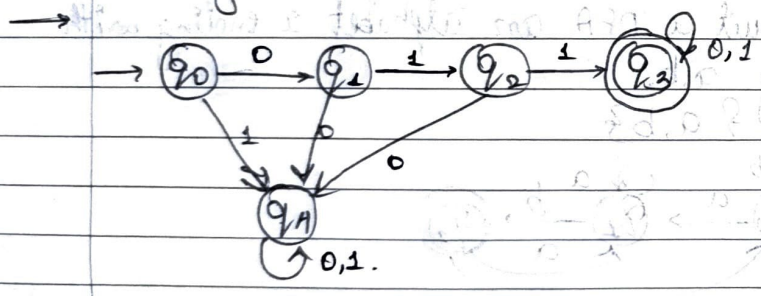


Q5. Construct to expect except strings of a's & b's having atleast one 'a'

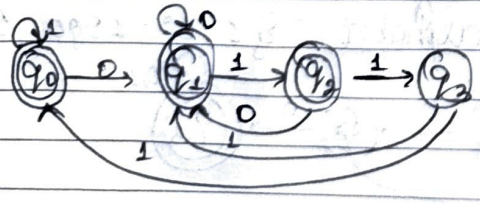
Q6. $L = \{ w = na(w) = 1 , w \in [a,b]^* \}$
occurence of a should be only 1.



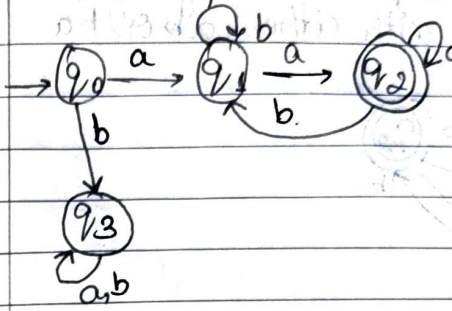
Q7. Construct a DFA to accept string of 0's & 1's starting with 011



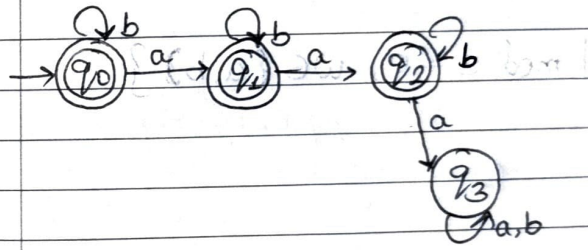
Q8. Design DFA to accept all strings of 1 that do not end with 011.



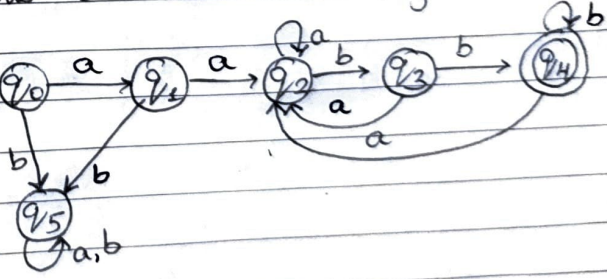
Q9. $L = \{ awa \mid w \in \{a,b\}^* \}$



Q10. Construct DFA to accept two strings with atmost two a's.

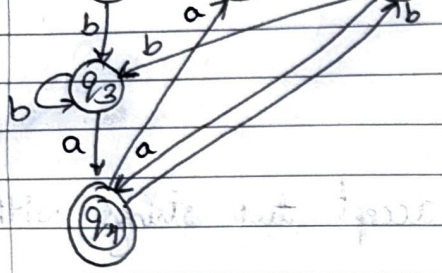


Q11. Construct a DFA to accept all binary strings that starts with atleast 2 a's & ends with atleast 2 b's



Q12. Design a DFA to accept all strings of a's & b's the string ending with either ab or ba.

→



Q13. $L = \{ w \mid |w| \bmod 3 = 0,\ w \in \{a,b\} \}$

Disadvantages of subset construction method

1. Subset construction method is exponential in time complexity all 2^n subsets are listed Thus can be verified using lazy evaluation method produces on the subset i.e, to process only accessible

Recursive definition of lazy evaluation

BASIS: The single ton set consisting of NFA start state is accessible.

INDUCTION: Suppose the subset S is accesible then for each I/p symbol compute the set of

1. Compute the following NFA to DFA

δ	a	b
→q_0	$\{q_0, q_1\}$	$\{q_2\}$
q_1	ϕ	$\{q_2\}$
*q_2	ϕ	ϕ

start state of DFA = $\{q_0\}$

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

→ Lazy Evaluation Method

(i) For state $\{q_0\}$

$\delta_0(\{q_0\}, a) = \{q_0, q_1\}$

$\delta_0(\{q_0\}, b) = \{q_2\}$

(ii) For state $\{q_0, q_1\}$

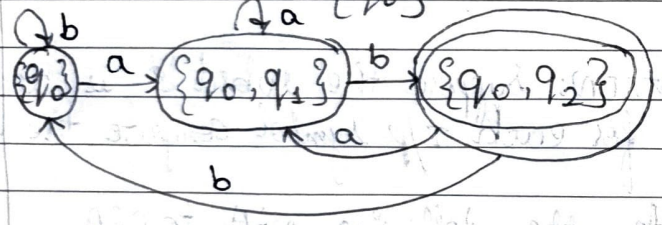
$\delta_0(\{q_0, q_1\}, a) = \delta_0(\{q_0\}, a) \cup \delta_0(\{q_1\}, a)$

$= \{q_0, q_1\} \cup \{\phi\}$

$= \{q_0, q_1\}$

$\delta_D(\{q_0,q_1\},b) = \delta_D(\{q_0\},b) \cup \delta_D(\{q_1\},b)$
$= \{q_0\} \cup \{q_2\}$
$= \{q_0,q_2\}$

(iii) For state $\{q_0,q_2\}$
$\delta_D(\{q_0,q_2\},a) = \delta_D(\{q_0\},a) \cup \delta_D(\{q_2\},a)$
$= \{q_0,q_1\} \cup \phi$
$= \{q_0,q_1\}$
$\delta_D(\{q_0,q_2\},b) = \delta_D(\{q_0\},b) \cup \delta_D(\{q_2\},b)$
$= \{q_0\} \cup \phi$
$= \{q_0\}$



2.

δ	a	b
→q_0	$\{q_0\}$	$\{q_0,q_1\}$
q_1	$\{q_2\}$	$\{q_2\}$
*q_2	ϕ	ϕ

Start state of DFA = $\{q_0\}$
$\Sigma = \{a,b\}$

→ Lazy Evaluation Method

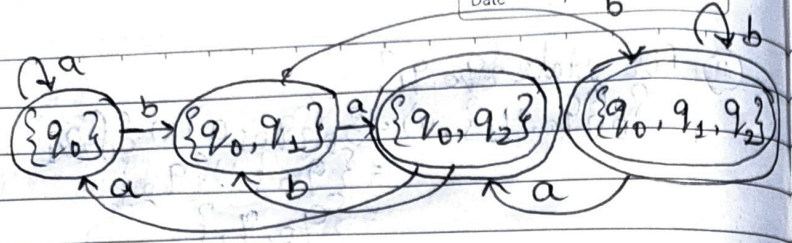
(i) For state $\{q_0\}$
$\delta_D(\{q_0\},a) = \{q_0\}$
$\delta_D(\{q_0\},b) = \{q_0,q_1\}$

(ii) For state $\{q_0,q_1\}$
$\delta_D(\{q_0,q_1\},a) = \delta_D(\{q_0\},a) \cup \delta_D(\{q_1\},a)$
$= \{q_0\} \cup \{q_2\}$
$= \{q_0,q_2\}$

$\delta_D(\{q_0,q_1\},b) = \delta_D(\{q_0\},b) \cup \delta_D(\{q_1\},b)$
$= \{q_0,q_1\} \cup \{q_2\}$
$= \{q_0,q_1,q_2\}$

(iii) For state $\{q_0,q_2\}$
$\delta_D(\{q_0,q_2\},a) = \delta_D(\{q_0\},a) \cup \delta_D(\{q_2\},a)$
$= \{q_0\} \cup \phi$
$= \{q_0\}$
$\delta_D(\{q_0,q_2\},b) = \delta_D(\{q_0\},b) \cup \delta_D(\{q_2\},b)$
$= \{q_0,q_1\} \cup \phi$
$= \{q_0,q_1\}$

(iv) For state $\{q_0,q_1,q_2\}$
$\delta_D(\{q_0,q_1,q_2\},a) = \delta_D(\{q_0\},a) \cup \delta_D(\{q_1\},a) \cup \delta_D(\{q_2\},a)$
$= \{q_0\} \cup \{q_2\} \cup \phi$
$= \{q_0,q_2\}$
$\delta(\{q_0,q_1,q_2\},b) = \delta_D(\{q_0\},b) \cup \delta_D(\{q_1\},b) \cup \delta_D(\{q_2\},b)$
$= \{q_0,q_1\} \cup \{q_2\} \cup \phi$
$= \{q_0,q_1,q_2\}$



3.

δ	a	b
→ p	{p,q}	{p}
q	∅	{r}
*r	{p,r}	{q}

Start state of DFA = {p}
Σ = {a, b}

Lazy Evaluation Method:

(i) For state {p}

$\delta_D(\{p\}, a) = \{p, q\}$

$\delta_D(\{p\}, b) = \{p\}$

(ii) For state {p, q}

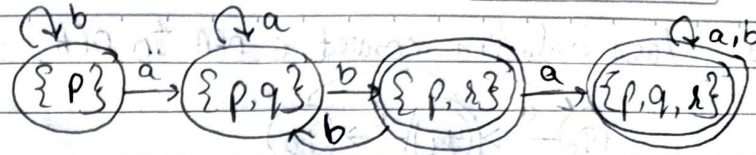
$\delta_D(\{p, q\}, a) = \delta_D(\{p\}, a) \cup \delta_D(\{q\}, a)$
$= \{p, q\} \cup \emptyset$
$= \{p, q\}$

$\delta_D(\{p, q\}, b) = \delta_D(\{p\}, b) \cup \delta_D(\{q\}, b)$
$= \{p\} \cup \{r\}$
$= \{p, r\}$

(iii) For state {p, q, r}

$\delta_D(\{p, q, r\}, a) = \delta_D(\{p\}, a) \cup \delta_D(\{q\}, a) \cup \delta_D(\{r\}, a)$
$= \{p, q\} \cup \emptyset \{p, r\}$
$= \{p, q, r\}$

$\delta_D(\{p, q, r\}, b) = \delta_D(\{p\}, b) \cup \delta_D(\{q\}, b) \cup \delta_D(\{r\}, b)$
$= \{p\} \cup \{r\} \{q\}$
$= \{p, q, r\}$



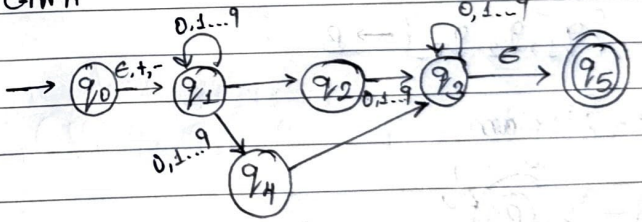
(iv) For state {p, r}

$\delta_D(\{p, r\}, a) = \delta_D(\{p\}, a) \cup \delta_D(\{r\}, a)$
$= \{p, q\} \cup \{p, r\}$
$= \{p, q, r\}$

$\delta_D(\{p, r\}, b) = \delta_D(\{p\}, b) \cup \delta_D(\{r\}, b)$
$= \{p\} \cup \{q\}$
$= \{p, q\}$

* DFA = take Σ, take I/P give only one state.
NFA = take Σ, take 1 or more I/P give 1 state.
ϵ-NFA = take Σ, take I/P's give 1 or more states.

* ϵ-NFA



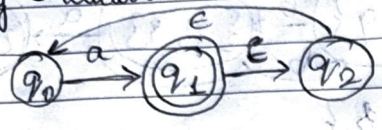
δ^* validate I/P:

1- computation of $\delta^*(q_0, -2.62)$

$\delta^*(q_0, \epsilon) = \epsilon$ - closure (q_0) → By Basic = $\{q_0, q_1\}$

$\delta^*(q_0, -) = \delta(\delta^*(q_0, \epsilon), -)$
$= \delta(\{q_0, q_1\}, -)$ ← By Induction
$= \delta(q_0, -) \cup (q_1, -)$
$= \{q_1\} \cup \{\phi\}$
$= q_1$
$= \epsilon$ - closure (q_1) ← By Induction
$= \{q_1\}$

1. Lazy Evaluation convert ϵ DFA to DFA.

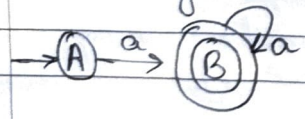


$\delta^*(q_0, a) = \delta(\delta^*(q_0, a))$
$= \delta(q_1)$
$= \epsilon$-closure $\rightarrow \{q_1\}$
$= \{q_0, q_1, q_2\}$

$\delta^*(\{q_0, q_1, q_2\}, a) = \delta(\delta^*\{q_0, q_1, q_2\}, a)$
$= \delta(\{q_0\}, a) \cup \delta(\{q_1\}, a) \cup \delta(\{q_2\}, a)$
$= q_1 \cup \phi \cup \phi$
$= \{q_1\}$
$= \epsilon$-closure $\rightarrow \{q_1\}$
$= \{q_1, q_0, q_2\}$

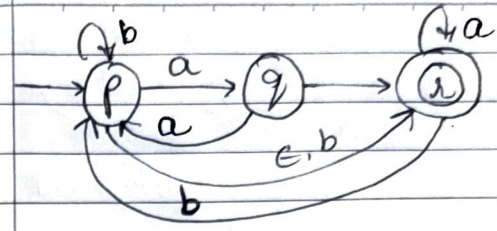
$\delta_0 = \{\{q_0\} \rightarrow A$
$\{q_1, q_2, q_0\} \rightarrow B$

DFA - Diagram



2. Convert ϵ-NFA to DFA using lazy evaluation metha

δ	ϵ	a	b
→ p	{1}	{q}	{p, r}
q	ϕ	{p}	ϕ
*r	{p, q}	{r}	{p}



$\delta^*(p, a) = \delta(\delta^*(p, a))$
$= \delta(q)$
$= \epsilon$ closure $\rightarrow \{q\}$
$=$

Unit: 2

Regular Expression & Languages.

Q1. Construct a regular expression for set of strings consisting of 0's & 1's ending with 011.

→ $(0+1)^*.011$

Q2. Construct a regular expression consisting 0's & 1's starting with 01

Q3. Any no. of a's followed by any no. of b's followed by any no. of c's.

→ $a^* b^* c^*$

Q4. String with atleast one a followed string with atleast one b followed by string with atleast one c.

→ $a\,a^*\,b\,b^*\,c\,c^* \Rightarrow a^+ b^+ c^+$

Q5 Strings ending with a or bb

→ $(a+b)^*(a+bb)$

Q6. Even no. of a's followed odd no. of b's.

→ $(aa)^*(bb)^* b.$

Q7. Strings ending with three consecutive zero's.

→ $(0+1)^*.000$

Q8. Strings of a's & b's of even length

→ $(aa+ab+ba+bb)^*$

Q9. Strings of a's & b's of odd length
→ $(aa+ab+ba+bb)^*(a+b)$

Q10. String starting with a & ending with b.
→ $a(a+b)^*b$

Q11. String starting with a or c and ending with any no. of b's.
→ $(a+c)(a+b+c)^*c^*$

Q12. Strings with exactly one a alphabet is a&b
→ b^*ab^*

Q13. All the strings of length 3 on alphabet a&b.
→ $(a+b)(a+b)(a+b) \Rightarrow (a+b)^3$

Q14. Strings of a's & b's who's second symbol from right end is a
→ $(a+b)^*a(a+b)$

Q15. Strings of a's & b's who's 10th symbol from right end is a.
→ $(a+b)^*a(a+b)^9$

Q16. Third symbol from right end is a & fourth symbol is b.
→ $(a+b)^*ba(a+b)(a+b)$

Q17. Strings of a's & b's atmost two a's.
→ $b^* + b^*ab^* + b^*ab^*ab^*$

Q18. $L = \{a^n b^m \mid m+n \text{ is even}\}$.
→ $(aa)^*(bb)^* + (aa)^*a(bb)^*b$

Q19 $L = \{a^{2n} b^{2m} \mid n \geq 0 \text{ & } m \geq 0\}$
→ $(aa)^*(bb)^*$

Q20. $L = \{a^n b^m \mid n \geq 4 \text{ & } m \leq 3\}$.
→ $aaaaa^*.(\epsilon + b + bb + bbb)$

Q21. Strings of a's & b's who's length is multiple of three.
→ $((a+b)(a+b)(a+b))^*$

Q22. Set of all the strings of 0's & 1's that don't end with 0, 1
→ $(0+1)^*.\{(00)+(01)+(11)\}(0+1)^*.(00+11+10)$

Q23.

Finite Automata & Regular Expression.

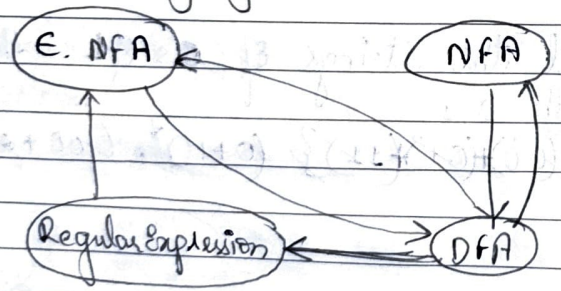
Regular Expression's & Finite Automata represent exactly same set of languages i.e, regular languages.

In order to show that regular expressions defines the same class we must show that

a) Every language is defined by one of these automata is also defined by a regular expression for this proof we assume the language is accepted by some DFA.

b) Every language denoted by regular expression is defined by one of these automata.

for this proof, the easiest is to show that there is an epsilon NFA (ϵ-NFA) accepting the same language

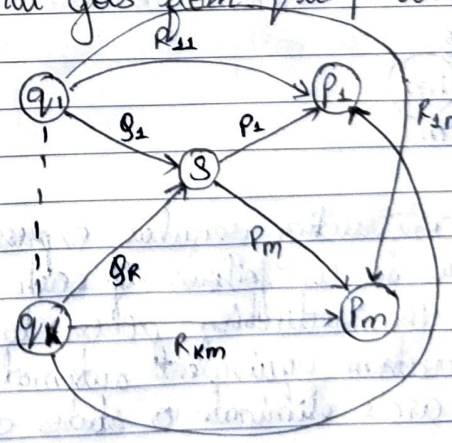


Algebraic laws of regular expression.

1. $R+S = S+R$
2. $(R+S)+P = R+(S+P)$
3. $RS(P) = R(SP)$
4. $\phi+R = R+\phi = R$
5. $\epsilon.R = R.\epsilon = R$
6. $\phi.R = R.\phi = \phi$
7. $R(S+P) = RS+RP$
8. $(R+S)P = RP+SP$
9. $R+R = R$
10. $(R^*)^* = R^*$
11. $\phi^* = \epsilon$
12. $\epsilon^* = \epsilon$
13. $R^+ = R.R^*$
14. $R^* = R^+ + \epsilon$
15. $R^*R^* = R^*$
16. $(R+S)^* = (R^*S^*)^*$
17. $(\epsilon+R)^* = R^*$

Converting DFA to Regular Expression using state elimination method:

When state s is to be eliminated all path going through s no longer exist. But the language of an automata shouldn't change we must include on the arc from q to p. the labels of the path that goes from q to p through s.



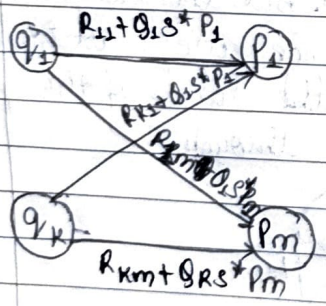
When state s is to be eliminated till $q_1, q_2 \dots q_k$ are predeccesors of state s
$p_1, p_2 \dots p_m$ are successors of state s.

$Q_i \rightarrow$ regular expression labelling arc from q_i to s

$P_j \rightarrow$ regular expression labelling arc from s to p_j

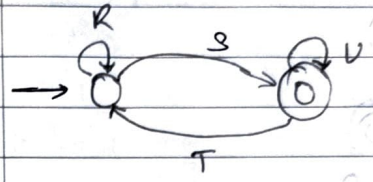
$R_{ij} \rightarrow$ regular expression labelling arc from q_i to p_j

Some of these arc's need exist the regular expression is null



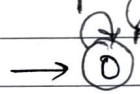
i) strategy for constructing regular expression from final automata is as follows of each accepting state q_i apply the reduction process to produce an equivalent canonical equivalent automata which RE labels on arc's eliminate or states accept q_0

ii) If $q \neq q_0$ then only two state automata is left, it looks like



$(R+SU^*T)^*SU^*$

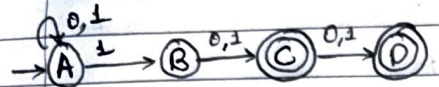
iii) If $q = q_0$ i.e start state is also an accepting state then automata,



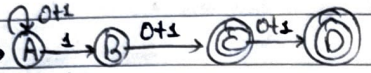
$RE \rightarrow R^*$

iv) Desired regular expression is the union of all expression derived from reduced automata for each accepting state.

Convert the following DFA using state elimination method.



Sol:



After eliminating state B.

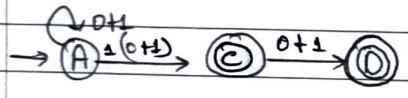
$R_{AC} + Q_1 S^* P_1$ $(\because R_{AC} = \phi)$, $S = \phi$

$Q_1 = 1$, $P_1 = 0+1$

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

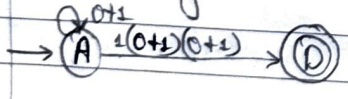
$\phi + 1(0+1)$

$1(0+1)$



Eliminating C & D separately get regular expression for two state finite automata separately and take their union.

Eliminating state C



$(R + SU^*T)^* SU^*$

$R = (0+1)$

$S = 1(0+1)(0+1)$

$U = \phi$

$T = \phi$

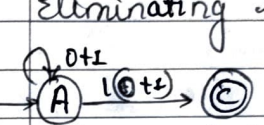
$((0+1) + 1(0+1)(0+1)\phi^*.\phi)^* 1(0+1)(0+1)(\phi)^*$

$((0+1)+\phi)^* . 1(0+1)(0+1)(\phi)^*$

$(0+1)^* . 1(0+1)(0+1)(\phi)^*$

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

Eliminating state D.



$(R + SU^*T)^* SU^*$

$R = (0+1)$

$S = 1(0+1)$

$U = \phi$

$T = \phi$

$((0+1) + 1(0+1)\phi^* . \phi)^* . 1(0+1)\phi^*$

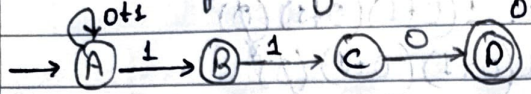
$((0+1) + \phi)^* . 1(0+1)$

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

Union of eliminating state C & D

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

Q2. Convert the following DFA to R.E



After eliminating state B.

$(R + SU^* T)^* SU^*$

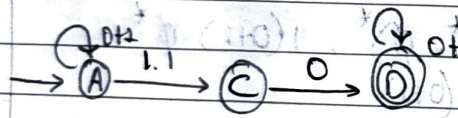
$R = 0+1$

$S = 1$

$R_{AC} + Q_1 S^* P_1$ $\quad R_{AC} = \phi,\ Q_1 = 1,\ P_1 = 1,\ S = \phi$

$\phi + 1(\phi)^* 1$

1.1



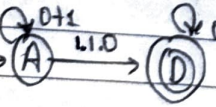
Eliminate state C.

$R_{AD} 0 + Q_1 S^* P_1$ $\quad R_{AD} = \phi,\ Q_1 = 1.1,\ S = \phi,\ P_1 = 0$

$\phi + 1.1(\phi)^* 0.$

$\phi + 1.1.0^*$

$1.1.0.$



~~Eliminate state D~~

$(R + SU^* T)^* SU^*$

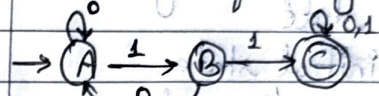
$R = 0+1,\ S = 1.1.0,\ U = 0+1,\ T = \phi$

$((0+1) + 1.1.0(0+1).\phi)^* . 1.1.0.(0+1)^*$

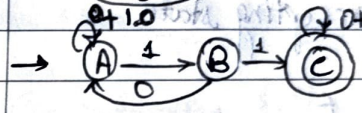
$(0+1+\phi)^* 110(0+1)^*$

$(0+1)^* 110(0+1)^*$

Q3. Convert the following DFA to R.E



Soln:



Eliminate state B.

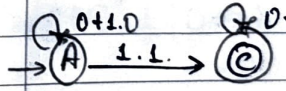
$R_{AC} + Q_1 S^* P_1.$

$R_{AC} = \phi,\ Q_1 = 1,\ S = \phi,\ P_1 = 1$

$\phi + 1(\phi)^*.1$

$\phi + 1.1$

$1.1.$



$(R + SU^* T)^* SU^*$

$R = 0+1.0,\ S = 11,\ U = 0+1,\ T = \phi$

$((0+1.0) + 1.1(0+1)^*.\phi)^* 1.1(0+1)^*$

$(0+1.0+\phi)^* 1.1(0+1)^*$

$(0+1.0)^* 1.1(0+1)^*$

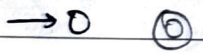
Converting Regular expression to automata.

Every language L(R) is also represented by L(E) for some ∈ NFA.(E) is also defined by finite automata with following condition:

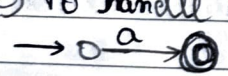
i) Exactly one accepting state
ii) No arc's into the initial state
iii) No arc's out of the accepting state

BASIS: a) To handle ∈

b) To handle ∅

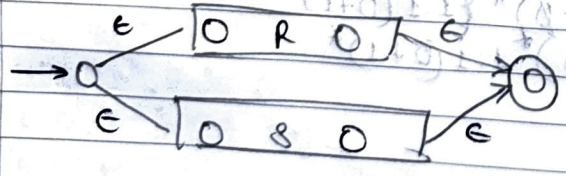


c) To handle a



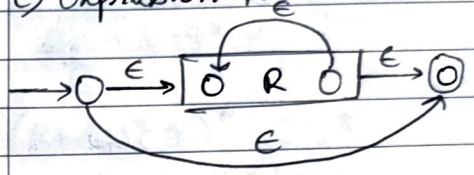
Induction: Assume statement true for inidiate sub expression

a) Expression R+S



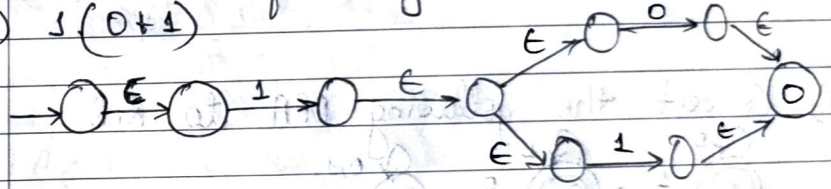
b) Expression R.S

c) Expression R^*

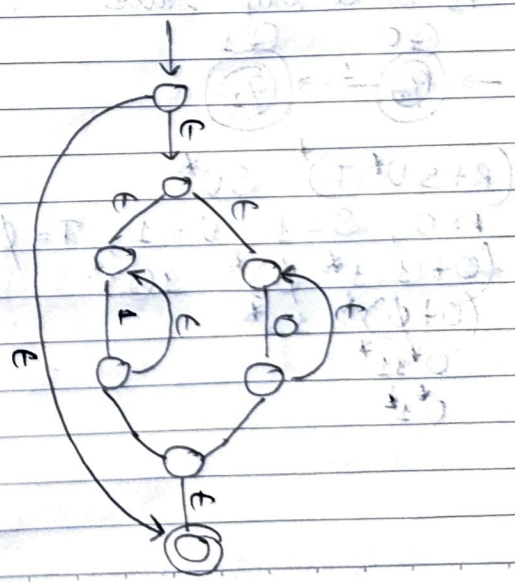


Q1. Convert the following R.E to ∈. NFA.

i) 1(0+1)

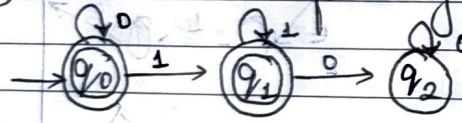


ii) $(0+1)^*1(0+1)$

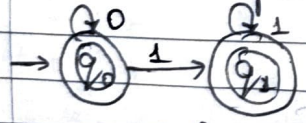


iii) $ab(a+b)^*$

iv) Convert the following DFA to R.E



Sol^n: q_2 is a trap state.



$(R+SU^*T)^* SU^*$

$R=0,\ S=1,\ U=1,\ T=\phi$

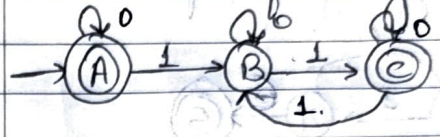
$(0+1.1^*\phi)^* 1(1)^*$

$(0+\phi)^* 11^*$

0^*11^*

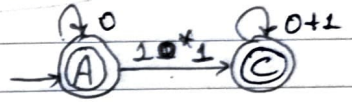
0^*1^*

Q5 Convert the following DFA to R.E.



Sol^n: After eliminating state B.

~~$R_{AC} + Q_1 S^* P_1$~~



$(R+SU^*T)^* SU^*$

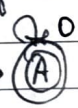
$R=0,\ S=10^*1,\ U=0+1,\ T=\phi$

$[0+10^*1(0+1)^*\phi]^* 10^*1(0+1)^*$

$(0+\phi)^* 10^*1(0+1)^*$

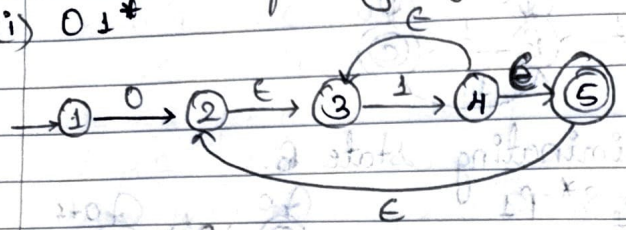
$0^*10^*1(0+1)^*$

After eliminating state C.

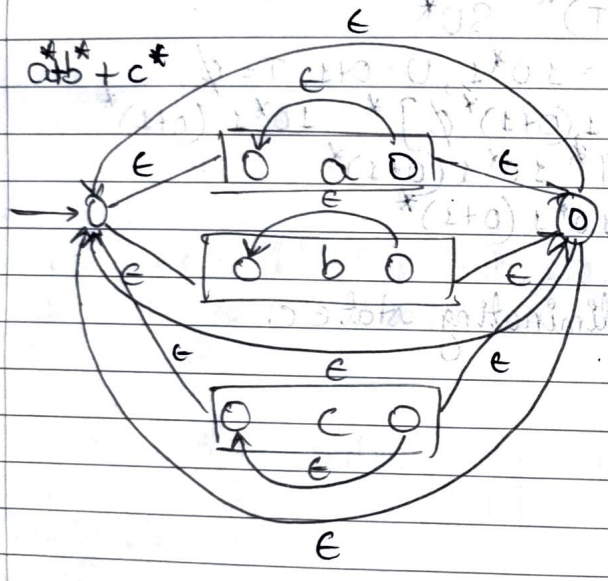


Q6. Convert the following regular expression to ε-NFA. (20M or 10M)

→(i) 01^*



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



(iii) $01+101$

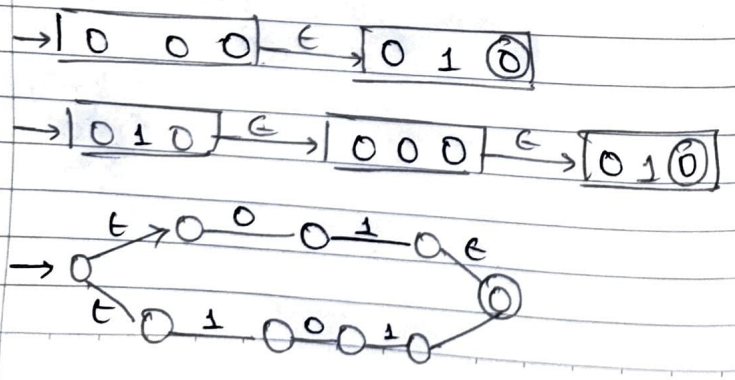
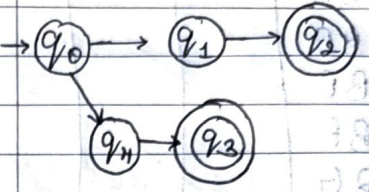


Table filling Algorithm.



Distinguishable pair (D.P)
$\delta(p,a) \in F$ } $(p,q) \rightarrow$ D.P
$\delta(q,b) \notin F$ }

Indistinguishable pair (I.P)
≡ Equivalent pairs / states
$p,q \notin F$ } (p,q) I.P
$p,q \in F$ }

$\delta(q_0,q_1) \rightarrow$ I.P $\delta(q_0,q_2) \rightarrow$ D.P
$\delta(q_0,q_3) \rightarrow$ D.P $\delta(q_2,q_3) \rightarrow$ I.P
$\delta(q_4,q_3) \rightarrow$ D.P $\delta(q_4,q_2) \rightarrow$ D.P
$\delta(q_1,q_2) \rightarrow$ D.P $\delta(q_1,q_3) \rightarrow$ D.P
$\delta(q_4,q_1) \rightarrow$ I.P $\delta(q_1,q_2) \rightarrow$ D.P
$\delta(q_1,q_3) \rightarrow$ D.P

Note: For a pair to be called distinguisable pair it must have only one final state in pair.

Q. Minimise the given 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

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

x → D.P
unmarked → I.P

I.

	δ	a	b
*	(AB)	BG	FC
X	(AD)	BC	FG
	(AE)	BH	FF
X	(AF)	BC	FG
	(AG)	BG	FE
X	(AH)	BG	FC

δ	a	b
BA		
BB		
BE		
BF		
BG		
BH		

	δ	a	b
X	BD	GC	CG
X	BE	GH	CF
X	BF	GC	CG
X	BG	GG	CF
*	BH	GG	CC

δ	a	b
~~CD~~ x DE	CH	GF
~~CE~~ x DF	CC	GG
~~CF~~ x DG	CG	GE
~~CG~~ x DH	CG	GC
CH		

	δ	a	b
X	EF	HC	FG
	EG	HG	FE
X	EH	HG	FC

δ	a	b
x FG	CG	GE
x FH	CG	GC

	δ	a	b
X	GH	GG	EC

II.

	δ	a	b
	AE	BH	FF
X	AG	BG	FE
	BH	GG	CC
	DF	CC	GG
X	EG	HG	FE

(AE), (BH), (DF) are unmarked pairs

I, P ≡ Equal states.

∴ C & G are Disquishable pairs are in the table All the pairs with C & G are marked.

III

	δ	a	b
→	(AE)	(BH)	(DF)
	(BH)	G	C
	(DF)	C	G
*	C	(AE)	C
	G	G	AE