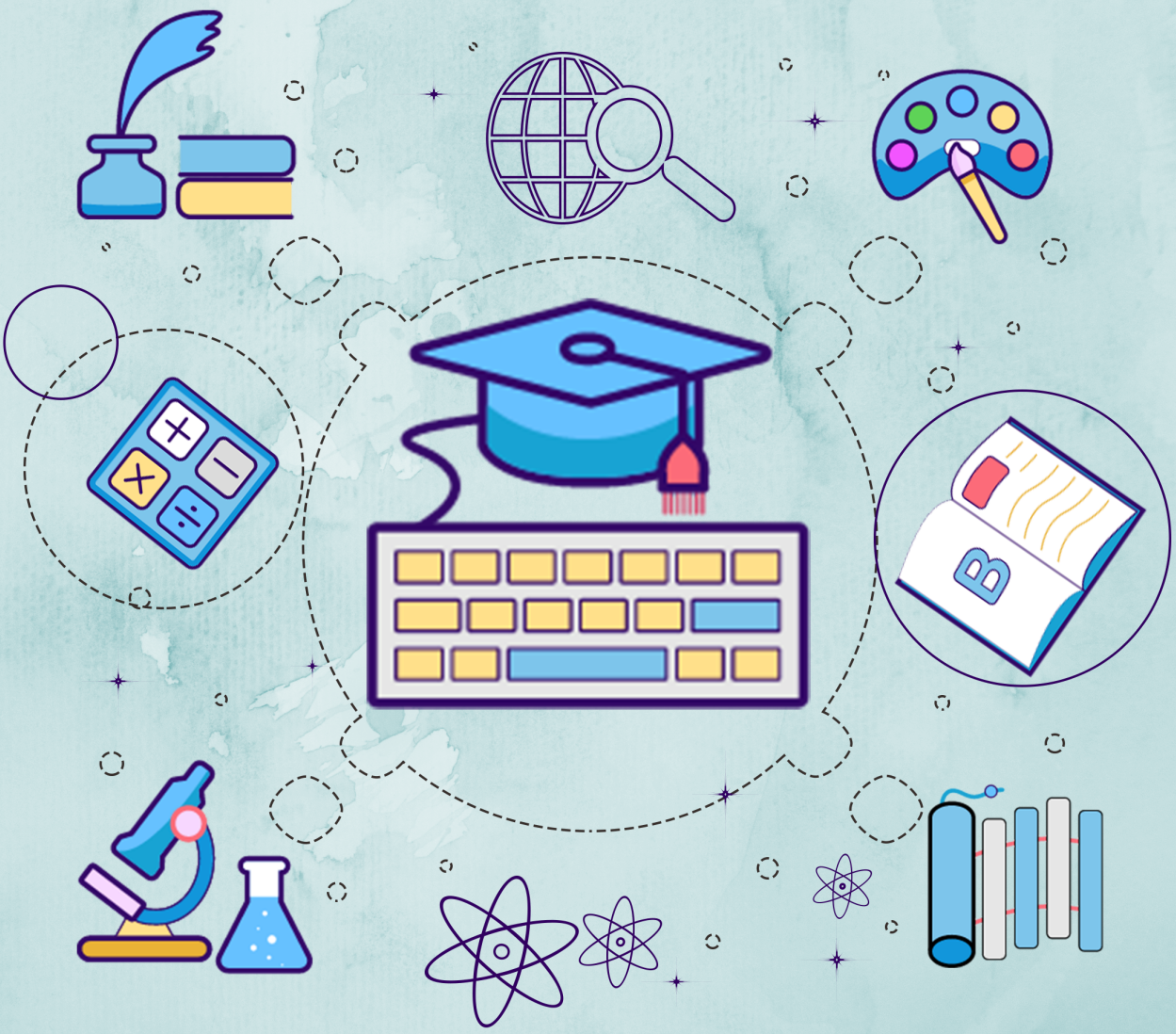
APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY

KeralaNotes



SYLLABUS | STUDY MATERIALS | TEXTBOOK

PDF | SOLVED QUESTION PAPERS

www.keralanotes.com

KeralaNotes

KTU STUDY MATERIALS

FORMAL LANGUAGES AND AUTOMATA THEORY

CST 301

Module 2

Related Link :

- KTU S5 STUDY MATERIALS
- KTU S5 NOTES
- KTU S5 SYLLABUS
- KTU S5 TEXTBOOK PDF
- KTU S5 PREVIOUS YEAR SOLVED QUESTION PAPER

FORMAL LANGUAGES AND AUTOMATA THEORY

Module 2

More on Regular Languages:

Regular Expression (RE), Equivalence of REs and DFA, Homomorphisms, Necessary conditions for regular languages, Closure Properties of Regular Languages, DFA state minimization (No proof required).

Prepared by:

Karishma PK

Assistant professor

Computer Science Department EKCTC

MODULE - II

REGULAR EXPRESSIONS

- The language accepted by finite automata can be easily described by simple expressions called Regular Expressions.
- It is the most effective way to represent any language.
- The languages accepted by some regular expression are referred to as Regular languages.
- A regular expression can also be described as a sequence of pattern that defines a string.
- Regular Expressions are used to denote regular languages.
- An expression is regular if
  - ϕ is a RE for regular language $\{\}$
  - ε is a regular expression for regular languages $\{\varepsilon\}$
  - If $a \in \Sigma$ (Σ represents the i/p alphabet), a is regular expression with language $\{a\}$
  - If a & b are regular expression, a+b is also a regular (union)
  - If a & b are regular expression, ab (concatenation of a & b) is also regular.
  - If a is regular expression, a^* (0 or more times a) is also regular (Kleene closure)

Regular languages

- A language is if it can be expressed in terms of regular expression
- In a RE, x^* means zero or more occurrence of x. It can generate $\{\epsilon, x, xx, xxx \ldots\}$
- In a RE, x^+ means one or more occurrance of x. It can generate $\{x, xx, xxx, xxxx, \ldots\}$

Q) write the regular expression for the language accepting all combinations of a's over the set $\Sigma = \{a\}$?

- All combinations of a's means 'a' may be zero, single, double and so on.
- If 'a' is appearing zero times, that means a null string. That is we expect the set of $\{\epsilon, a, aa, aaa \ldots\}$
- $\therefore$ RE $= a^*$

Q) write the RE for the language accepting all combinations of a's except the null string over the set $\Sigma = \{a\}$?

$L = \{a, aa, aaa \ldots\}$

$\therefore$ RE $= a^+$

Q) write the regular expression for the language accepting all the string containing any number of a's & b's?

$RE = (a+b)^*$

$L = \{ \epsilon, a, aa, ab, ba, bb \cdots \}$

Q) write the RE for the language accepting all the string which are starting with 1 & ending with 0 over $\Sigma = \{0,1\}$?

$\therefore RE = 1(0+1)^*0$

Regular set

Any set represented by a regular expression is called as regular set.

Eg:- Let $a, b \in \Sigma$

- $a \rightarrow \{a\}$
- $ab \rightarrow \{ab\}$
- $a+b \rightarrow \{a,b\}$
- $b^* \rightarrow \{\epsilon, b, bb, bbb \cdots\}$ or $\{b\}^*$
- $(0+1)^* \rightarrow \{0,1\}^*$

Operations of RE

→ Union ($L_1 + L_2$)

Let $L_1 = \{1, 10, 11\}$ & $L_2 = \{\epsilon, 1\}$

$L_1 + L_2 = \{\epsilon, 1, 10, 11\}$

→ concatenation ($L_1 \cdot L_2$)

Let $L_1 = \{1, 10, 11\}$ & $L_2 = \{\epsilon, 1\}$

$L_1 \cdot L_2 = \{1, 10, 11, 11, 101, 111\}$

Q) Describe the following sets by regular expressions?

(i) $\{bab\}$ => bab

(ii) $\{01, 10\}$ => 01+10

(iii) $\{\epsilon, ab\}$ => $\epsilon + ab$

(iv) $\{abb, a, b, bba\}$ => abb+a+b+bba

(v) $\{\epsilon, a, aa, aaa \cdots\}$ => a^*

(vi) $\{aa, aaa \cdots\}$ => a^+

Q) set of all strings containing exactly two a's over $\Sigma = \{a, b\}$? Find RE?

$L = \{aa, aab, aabb, baa, baba, abba, babab, \cdots\}$

$- a - a -$

$\therefore RE = b^*ab^*ab^*$

Q) Set of all strings containing atleast 2 a's over $\Sigma = \{a, b\}$? Find Regular Expression ?

– a – a –

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

Q) Set of all strings containing at most 2 a's over $\Sigma = \{a, b\}$?

$L = \{bb, bbab, aa, baa \ldots\}$.

- b^*
- $b^* a b^*$
- $b^* a b^* a b^*$

$\rightarrow b^* + b^* a b^* + b^* a b^* a b^* \Rightarrow RE$

Q) Set of all strings containing substring aa over $\Sigma = \{a, b\}$

$L = \{aa, baaa, aabaab \ldots\}$

– aa –

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

Q) Find RE for the set of all strings on $\{a, b\}$ terminated by either 'a' or bb ?

$L = \{a, bb, aa, abb, ba, bbb \ldots\}$

$RE = (a+b)^* (a+bb)$

Q) $L = \{aa, ab, ba, bb\}$

$a(a+b) + b(a+b)$

$(a+b)(a+b) \Rightarrow RE$

Q) For $\Sigma = \{0, 1\}$ give RE corresponding to Regular language!

$L(r) = \{w \in \Sigma^* : w$ has atleast one pair of consecutive zeros$\}$

$L = \{00, 100, 00111, 101000, 10100100 \cdots\}$

$RE = (0+1)^* 00 (0+1)^*$

Q) Give a regular expression for $L = \{a^n b^m : n \geq 1, m \geq 1, n+m \geq 3\}$

soln:

$L = \{abb, aab, aabb, aabbb \cdots\}$

$RE = a^+ b^+ \cdot b + a \cdot a^+ b^+$

Q) RE for $L = \{ab^n w : n \geq 3, w \in \{a,b\}^+\}$

$L = \{abbba, abbbba bb, \cdots\}$

$abbb\, b^* (a+b)^+$

Q) consider the language of given RE $(a+b)^*$.

$(a+b)^* = \{\lambda, a, b, aa, bb, \cdots\}$

Q) $(a \cdot b + b \cdot c)^* = \{\lambda, ab, bc, abbc, abab \cdots\}$

Q) Find RE from RL

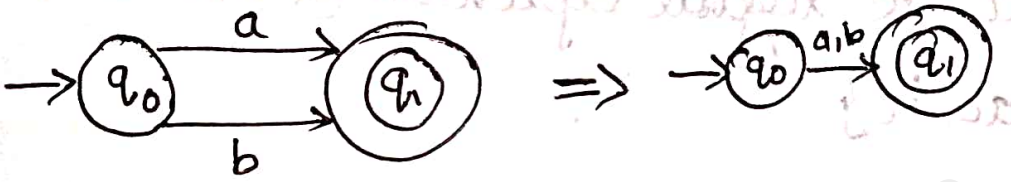
$L = \{abx \mid x \in \{a,b\}^*\}$

$ab(a+b)^*$

Conversion of RE to FA [Regular Expression to Finite Automata]

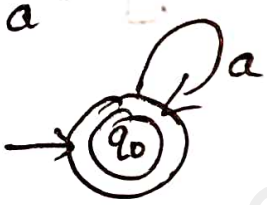
Rules

1) a+b·(a or b) or (a|b)

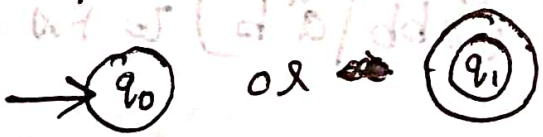


2) ab

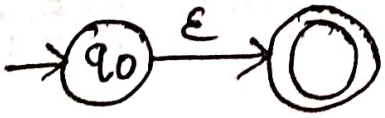
3) a*



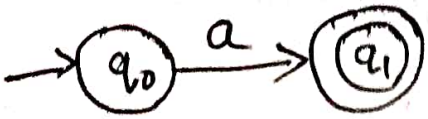
4) ϕ



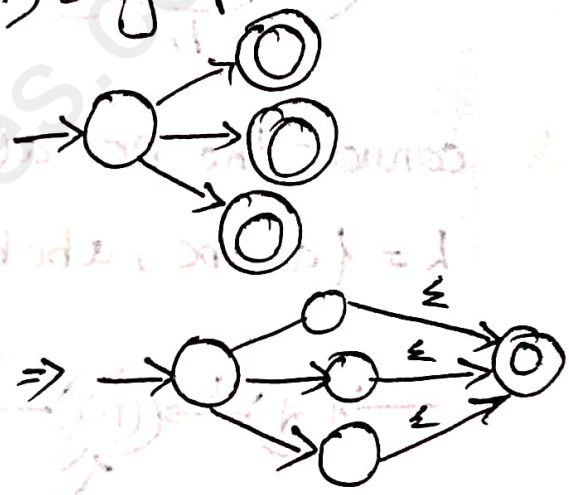
5) ε



6) {a}

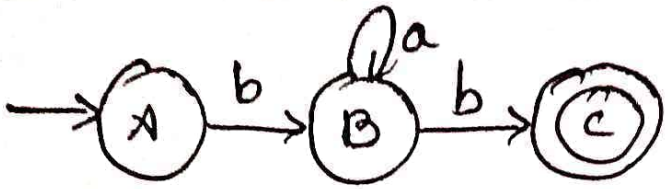


7) single final state



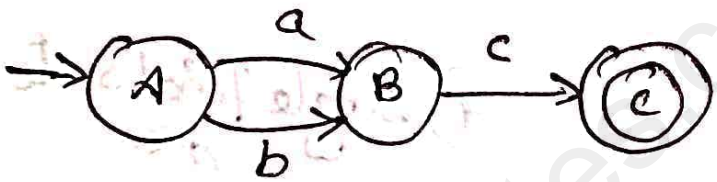
Q) Convert the regular expression ba^*b to finite Automata?

$L = \{bb, bab, baab \cdots\}$



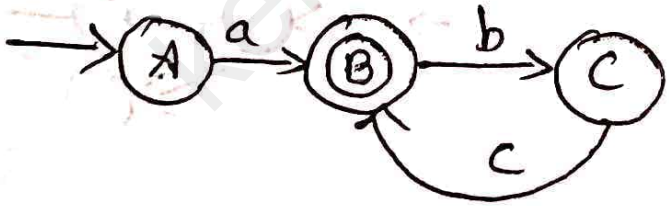
Q) Convert the regular expression $(a+b)c^*$ to FA

$L = \{ac, bc\}$

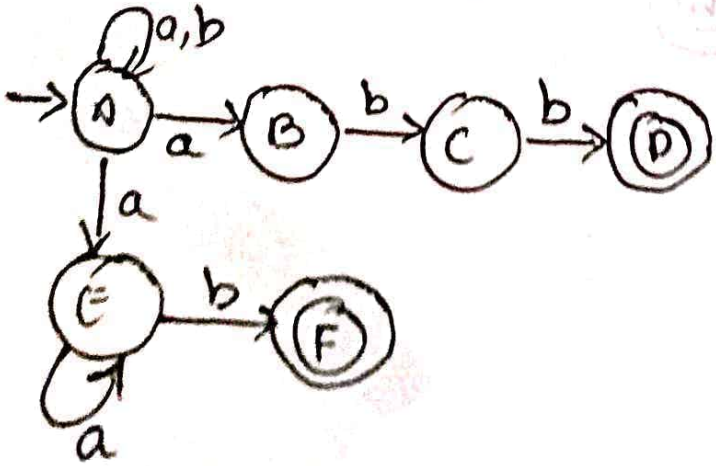


Q) Convert the RE $a(bc)^*$ to FA?

$L = \{a, abc, abcbc, abcbcbc \cdots\}$



Q) Convert the RE $(a|b)^*.(abb|a^+b)$ to FA?



a) $a+a^*b$ Find language ?

$L = \{ a, b, ab, aab, aaaab \cdots \}$

b) write RE for the language starting with a but not having consecutive 1's ?

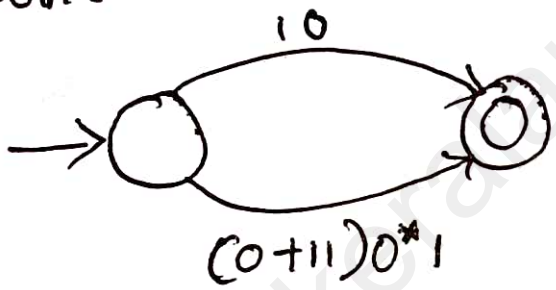
$L = \{ a, aba, aab, aaa, abab \cdots \}$

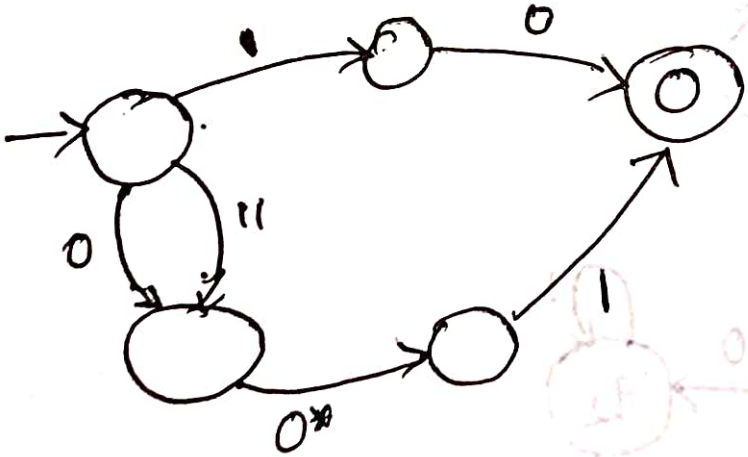
$RE = \{a+ab\}^*$

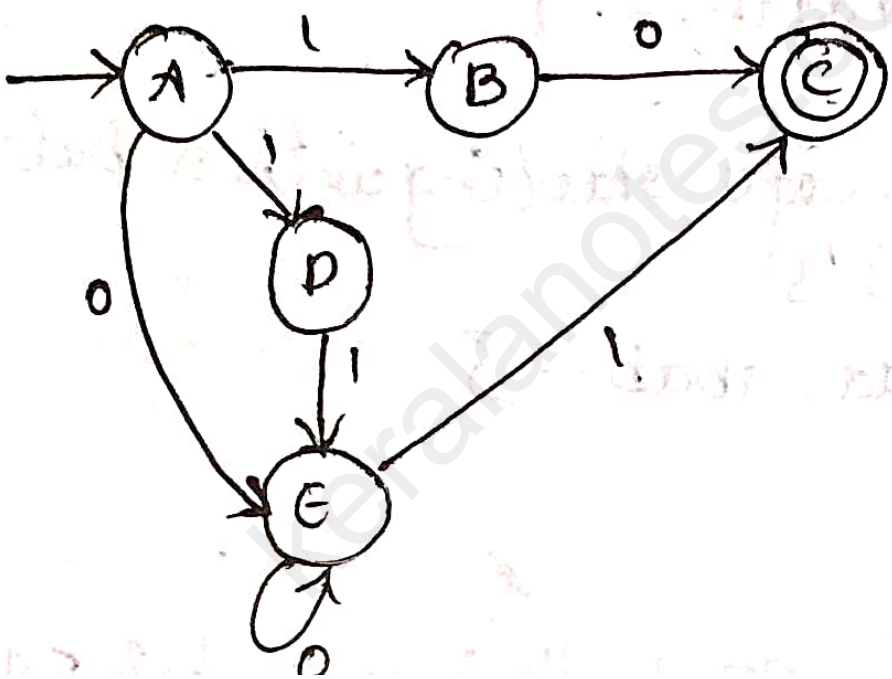
c) convert the following RE to their equivalent FA ?

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

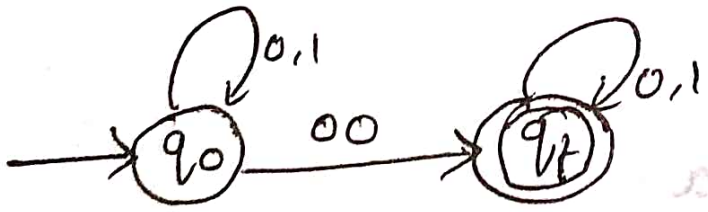
Soln :-







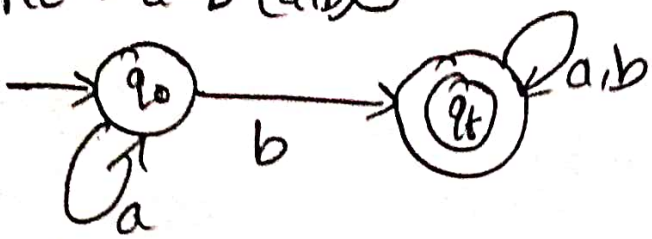
Q) construct the FA for RE $(0+1)^* 00 (0+1)^*$



Q) construct FA for RE ba^*b

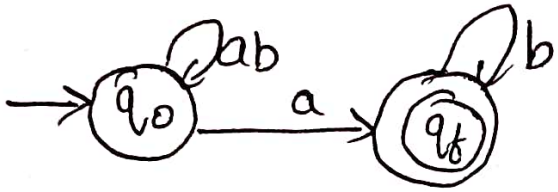
Q) Convert the given RE to FA ?

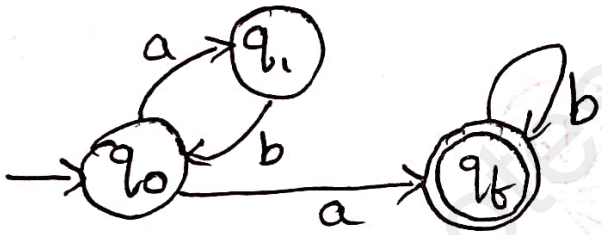
RE = $a^*b(a+b)^*$



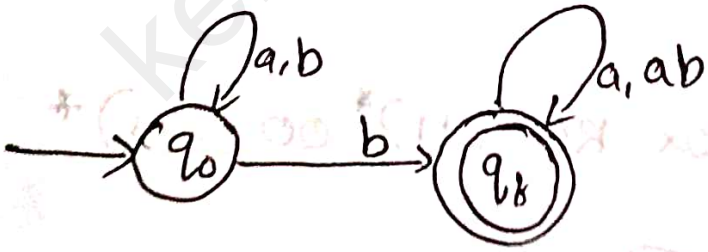
Q) convert RE to FA ?

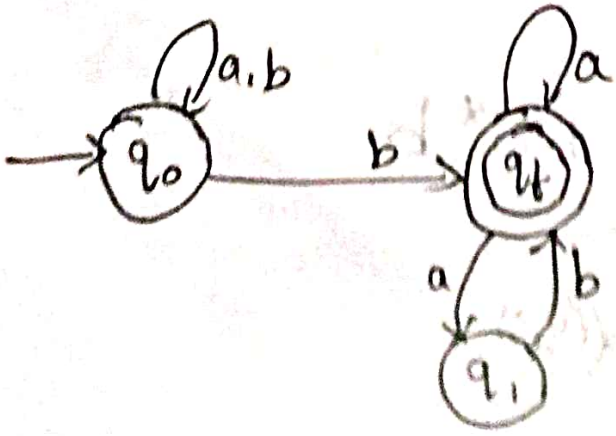
$(ab)^*a\,b^*$





Q) convert RE $((a+b)^*b(a+ab)^*)$ to FA ?





Klean's Theorem Thompson's Theorem

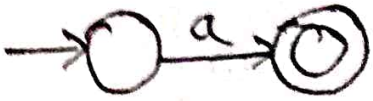
For any regular expression that represents language $L(r)$ there is a finite automaton that accepts same language.

- For any FA, 'M' that accepts language $L(M)$, there is a regular expression that represents the same language.

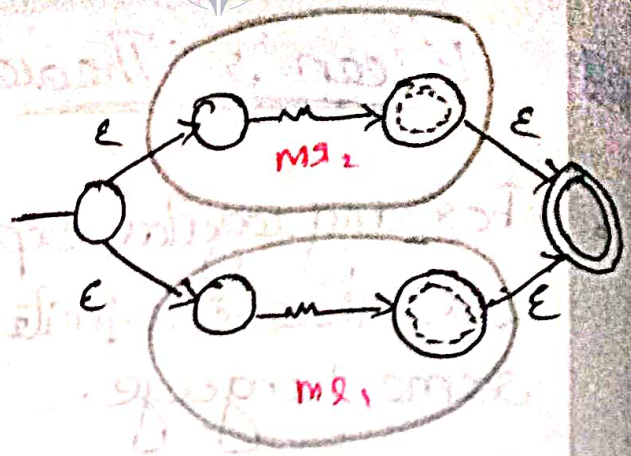
Proof (RE to FA)

If $L = L(r)$ for some RE, r then $L = L(M)$ for some FA, M.

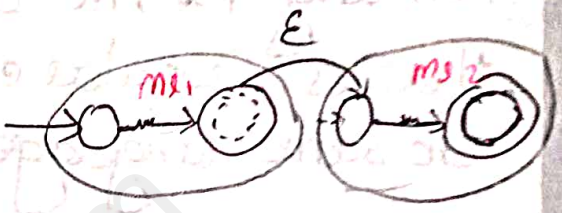
The following diagram itself is the proof by induction.

r	L(r)	M
ϕ	$\{\}$	
ε	$\{\varepsilon\}$	
a	$\{a\}$	

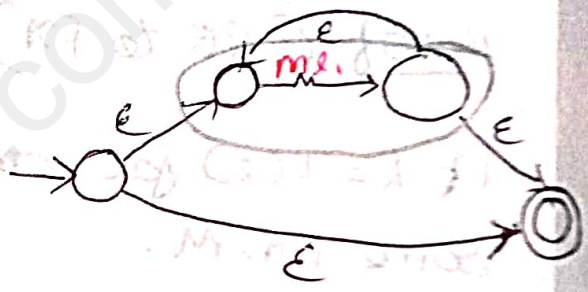
$r_1 + r_2$ $\quad L(r_1) \cup L(r_2)$



$r_1 \cdot r_2$ $\quad L(r_1) \cdot L(r_2)$



r^* $\quad (L(r))^*$

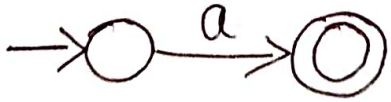


Conversion of regular expression to NFA with epsilon transition

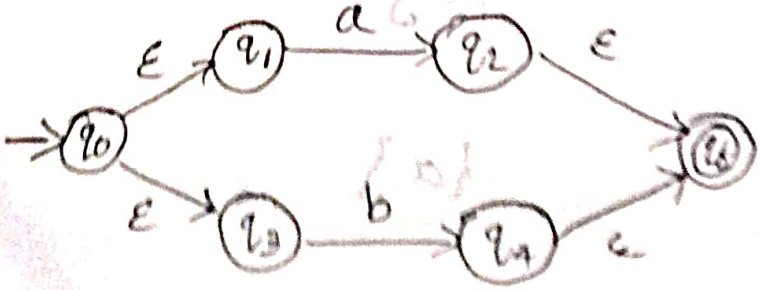
or Equivalence of RE & NFA

(Thompson construction method)

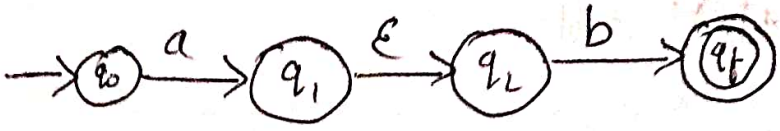
a



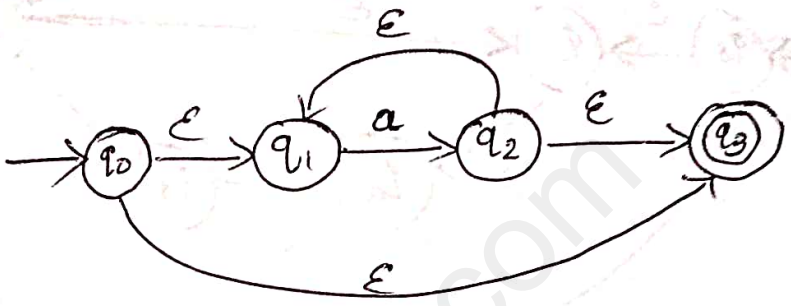
a+b



ab

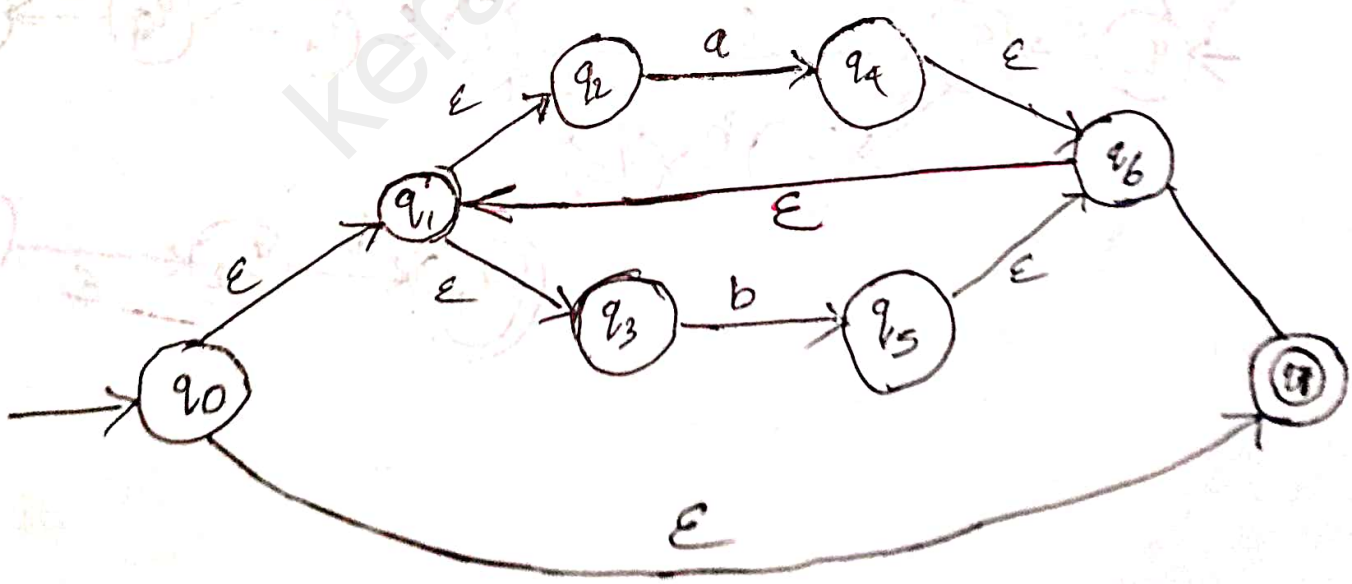


a^*

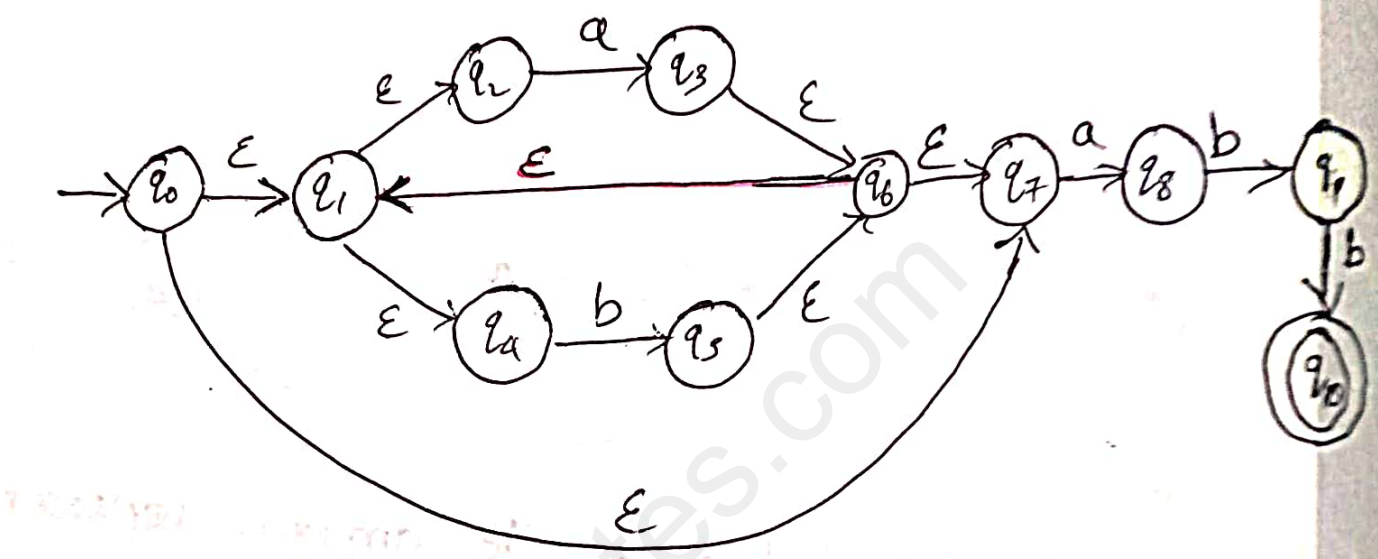


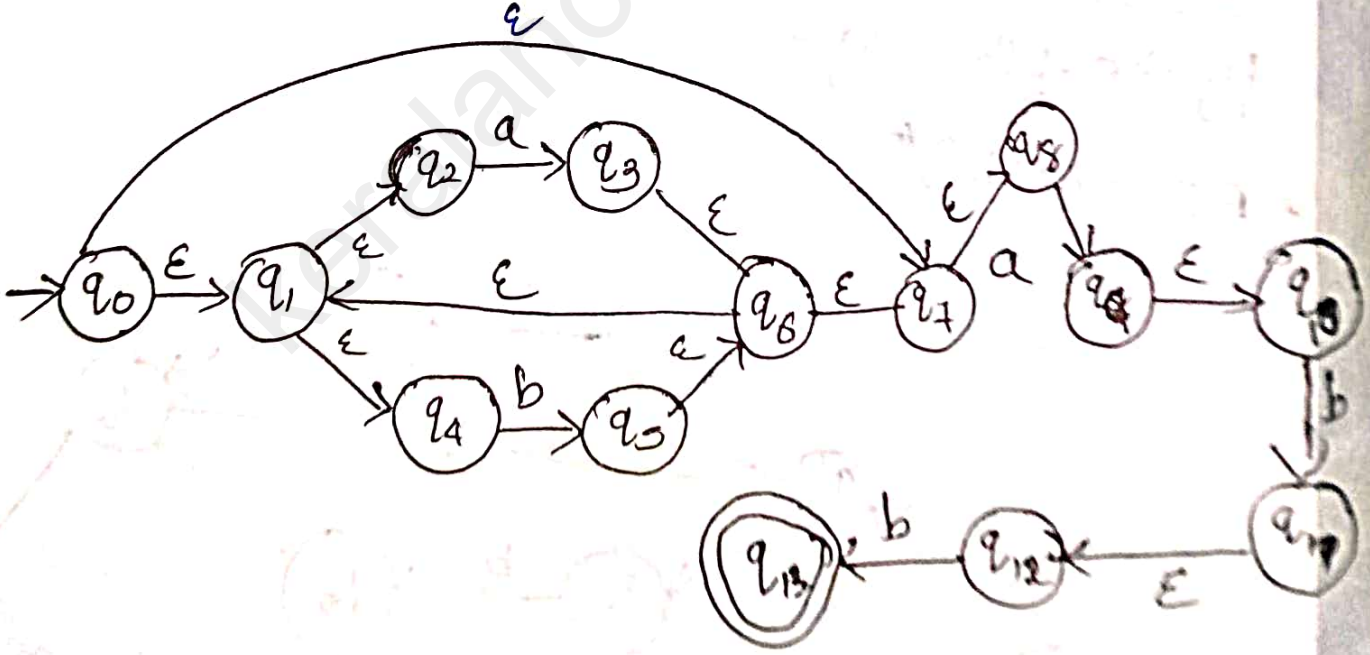
9) Construct FA that accept the language represented by the following RE ?

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

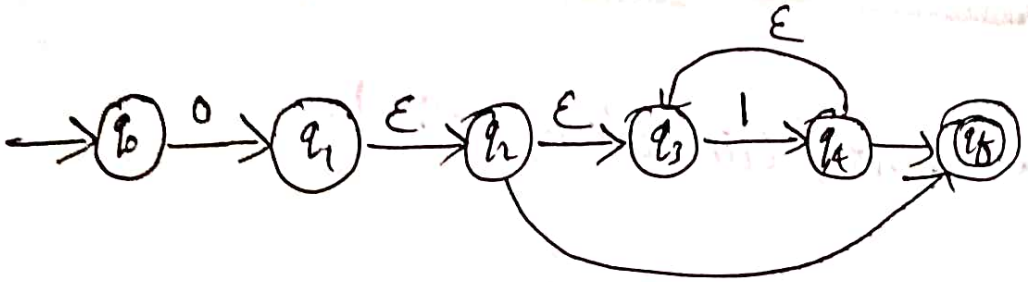


(b) $(a+b)^* abb$

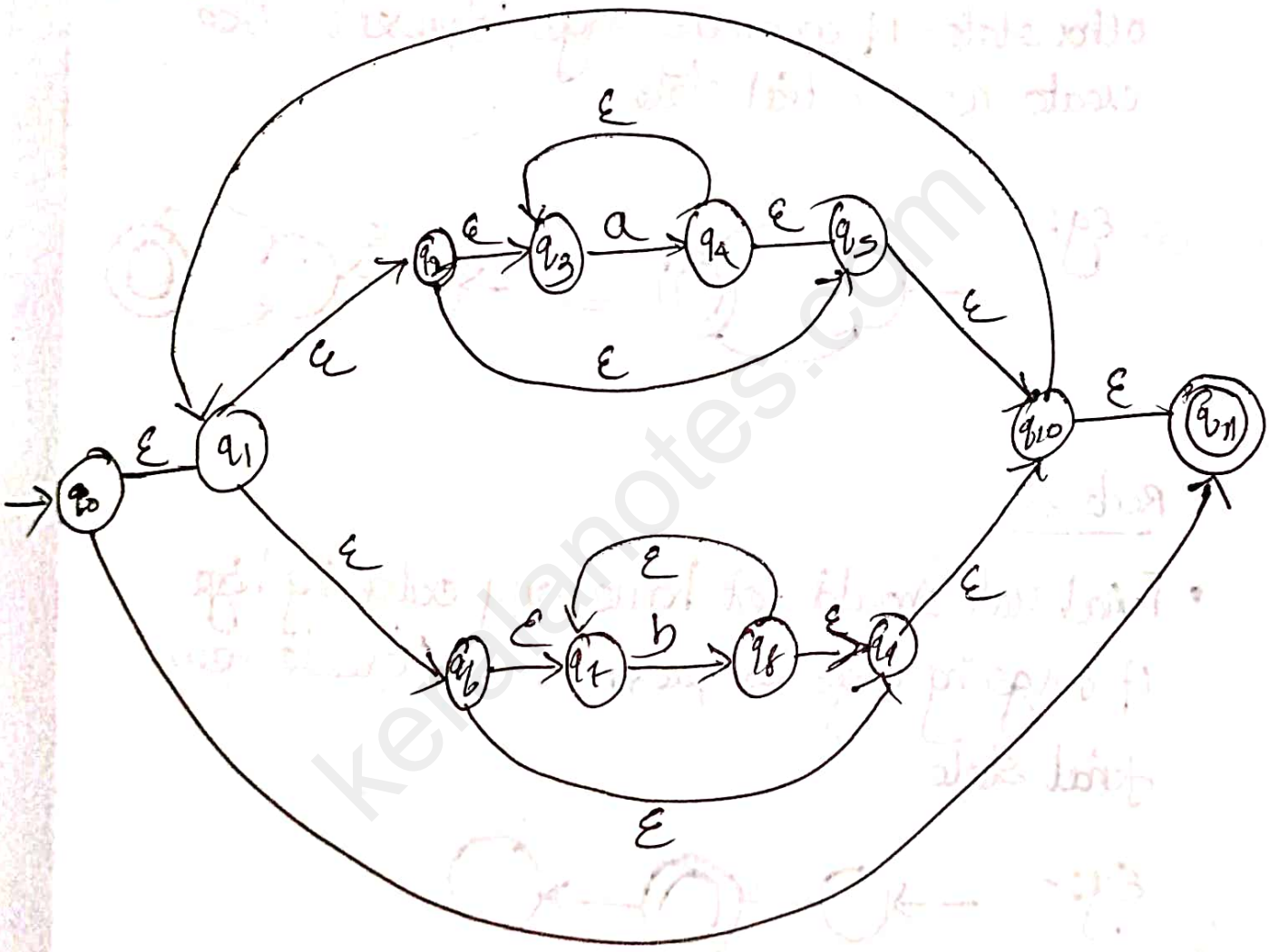




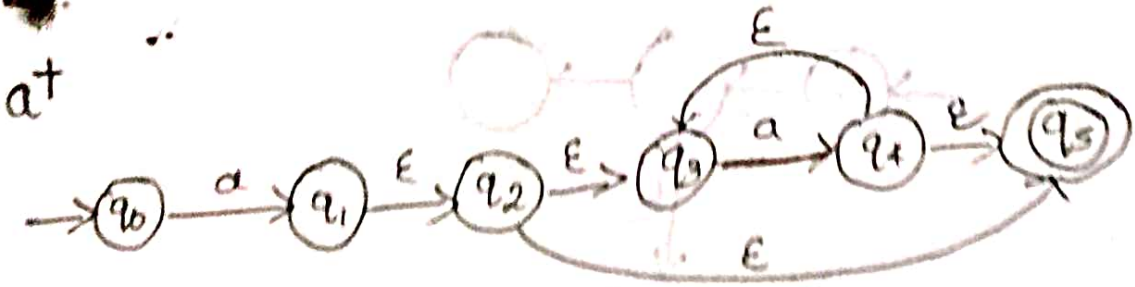
Q) 01^*



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



a^+



CONVERSION OF FA TO RE

Method : state elimination Method.

Rule 1

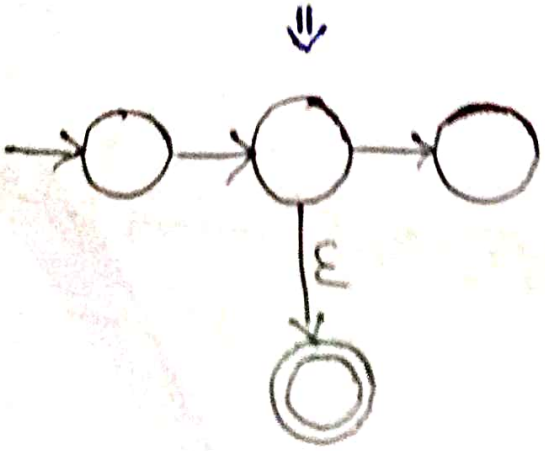
- Initial state should not have any incoming edge from other state. If incoming edge is present, then create new initial state.

Eg:-

Rule 2a

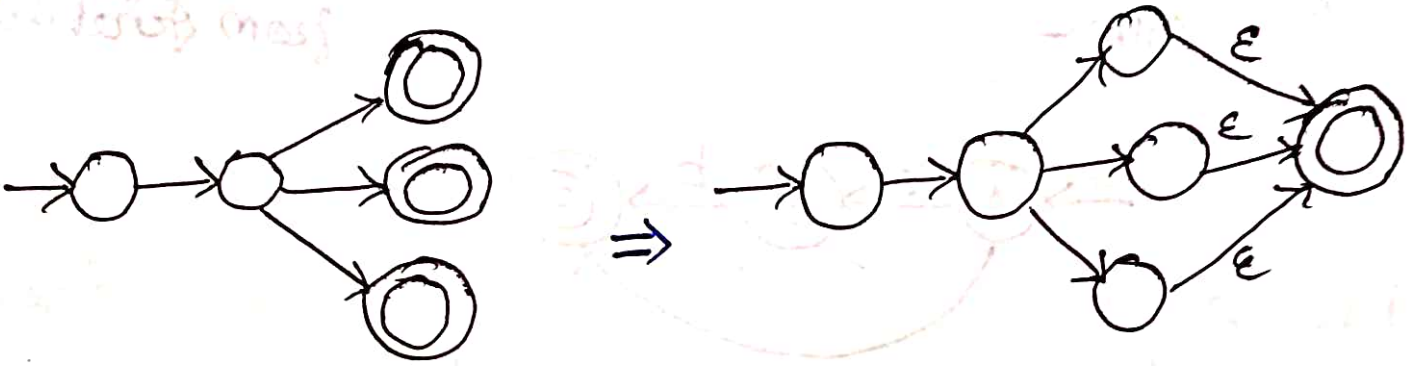
- Final state should not have any outgoing edge If outgoing edge is present, then create new final state.

Eg:-



Rule 2b

- If more than one final state is present, then convert it onto one state.

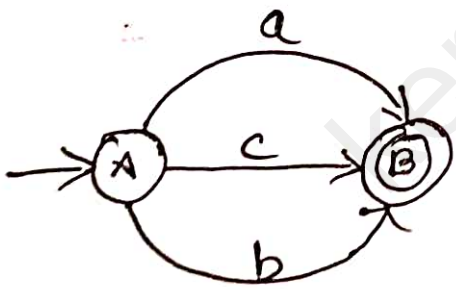


Rule 3

- Eliminate the state one by one other than the initial and final state.

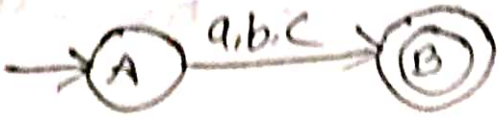
Examples

1)

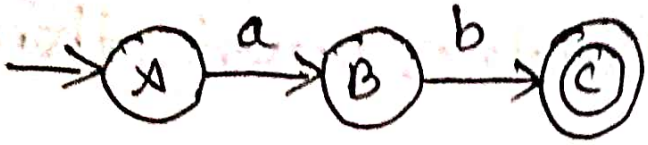


- no incoming edge from other state to initial state
- no outgoing edges from final state.

RE = a+b+c

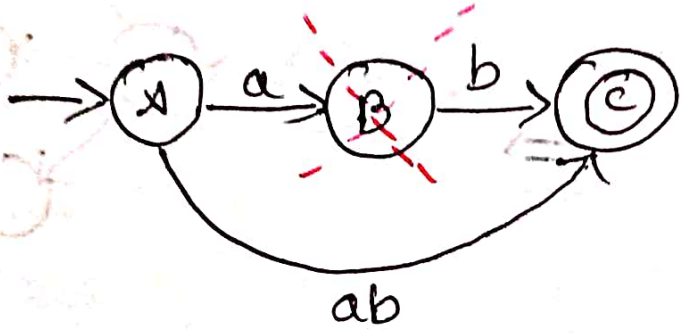


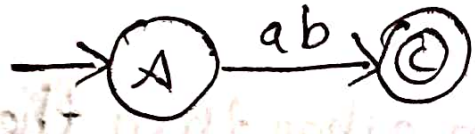
2)



- no incoming edge to initial state
- no outgoing edges from final states.

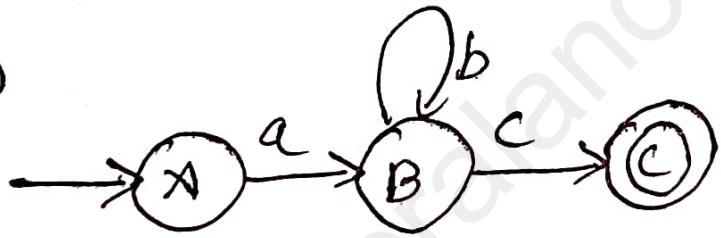
- soln:-



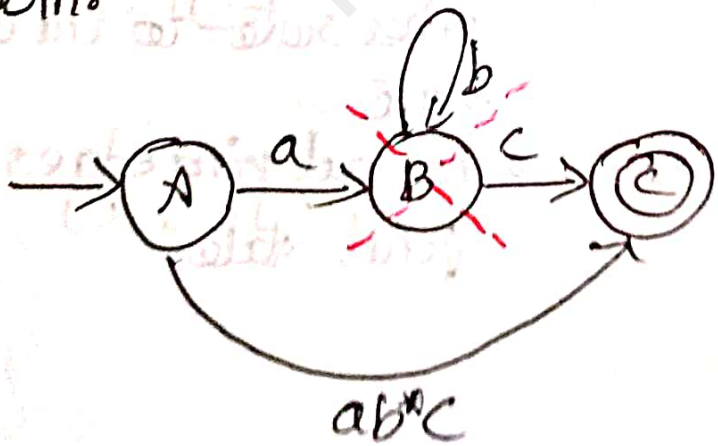


∴ RE = ab

3)

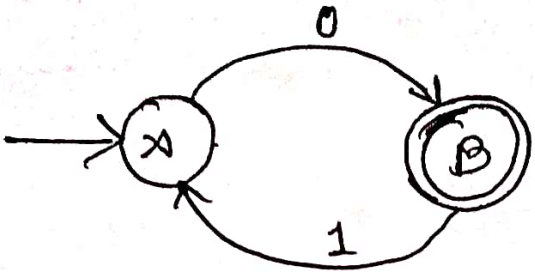


soln:-

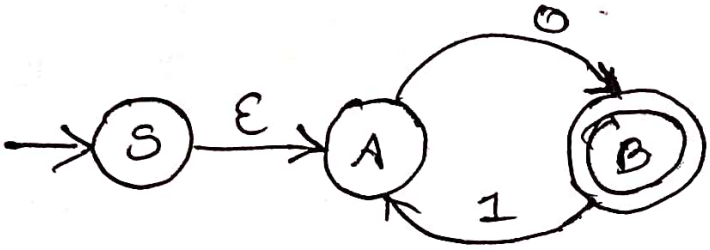


∴ RE = ab^*c

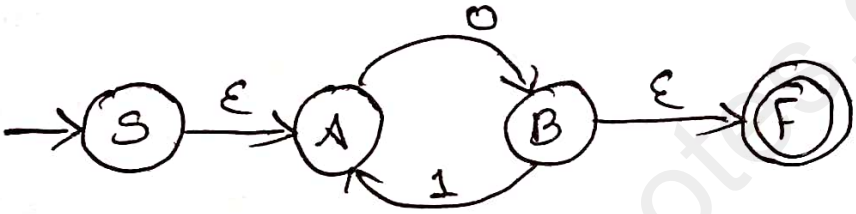
4)

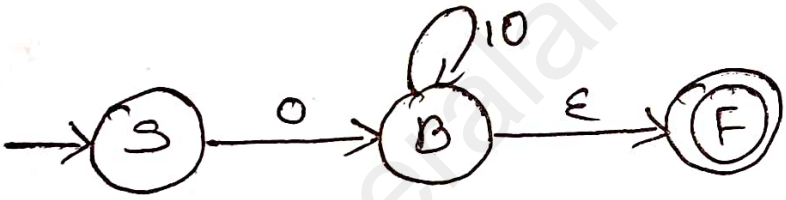


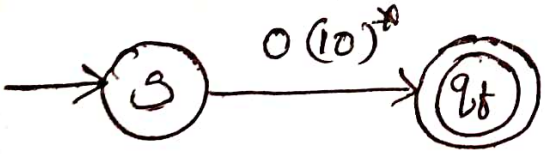
incoming edge is present from other state to initial state. So, create a new initial state.



outgoing edge is present from final state. so create new final state







$\therefore$ RE = $0(10)^*$

Identity Rules of Regular Expression

$\phi + R = R + \phi = R$

$\phi \cdot R = \phi$

$\varepsilon \cdot R = R \cdot \varepsilon = R$

$\varepsilon + R = R + \varepsilon$

$R + R = R$

$R^* \cdot R = R \cdot R^* = R^+$

$\varepsilon + RR^* = R^*$

$\varepsilon^* = \varepsilon$

$\phi^* = \varepsilon$

$(R^*)^* = R^*$

$R^* \cdot R^* = R^*$

$(P+Q)^* = (P^* Q^*)^* = (P^* + Q^*)^*$

$(PQ)^* P = P(QP)^*$

$R(P+Q) = RP + RQ$

$(P+Q)R = PR + QR$

$P + Q = Q + P$

eg:-

(8) Prove that $(1+00^*1)+(1+00^*1)(0+10^*1)^*(0+10^*1)$ is equal to $0^*1(0+10^*1)^*$?

$$LHS = (1+00^*1)+(1+00^*1)(0+10^*1)^*(0+10^*1)$$

$$= (1+00^*1)[\varepsilon + (0+10^*1)^*(0+10^*1)]$$

$$= (1+00^*1)(0+10^*1)^*$$

$$= (\varepsilon\cdot(1+00^*1))(0+10^*1)^*$$

$$= (\varepsilon\cdot 1+00^*1)(0+10^*1)^*$$

$$= (\varepsilon+00^*)1(0+10^*1)^*$$

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

(9) Prove that $(1+100^*)+(1+100^*)(0+10^*)(0+10^*)^*$ is equal to $10^*(0+10^*)^*$?

$$LHS = (1+100^*)+(1+100^*)(0+10^*)(0+10^*)^*$$

$$= (1+100^*)[\varepsilon + (0+10^*)(0+10^*)^*]$$

$$= (1+100^*)(0+10^*)^*$$

$$= \varepsilon\cdot(1+100^*)(0+10^*)^*$$

$$= \varepsilon\cdot 1+100^*(0+10^*)^*$$

$$= 1(\varepsilon+00^*)(0+10^*)^*$$

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

HOMOMORPHISM / SUBSTITUTION FUNCTION

Suppose Σ and Γ are alphabets, then a function

$$h: \Sigma \to \Gamma^*$$

is called homomorphism.

Homomorphism is a substitution in which a single letter is replaced with a string. The domain of the function is extended to strings in an obvious fashion

$$\text{if } w = a_1 . a_2 . a_3 \ldots \ldots a_n$$

$$h(w) = h(a_1), h(a_2) \ldots \ldots h(a_n)$$

If L is a language on Σ, then its homomorphic language is defined as $h(L) = \{h(w) : w \in L\}$

Example:

$\Sigma = \{a, b, c\}$ and $\Gamma = \{a, b, c\}$ defined by

$h(a) = ab$

$h(b) = bbc$

$h(aba) = abbbcab$

Homomorphic image of $L = \{aa, aba\}$ is the language

$h(L) = \{abab, abbbcab\}$.

If r is the RE for language L, then a RE for $h(L)$ can be obtained by simply applying homomorphism to each symbol $\in$ of r.

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

$h(a) = dbcc$

$h(b) = bdc$

L is a regular language denoted by

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

then $r_1 = (dbcc + (bdc)^*)(dbccdbcc)^*$

Theorem

Let h be a homomorphism. If L is a regular language, then its homomorphic image $h(L)$ is also regular. The family of regular language is closed under homomorphism.

non Regular languages

Languages that cannot be represented using Finite Automata.

9) using homomorphism on regular languages, prove that the language.

$L = \{a^n b^n c^{2n} \mid n \geq 0\}$ is not regular.

Given that the language $\{a^n b^n : n \geq 1\}$ is not regular?

Let $h(a) = h(b) = a$ and $h(c) = b$.

when apply homomorphism the language L becomes $L' = \{a^n a^n b^{2n}\}$

which is equal to

$\{a^{2n} b^{2n}\}$.

Given that $\{a^n b^n : n \geq 1\}$ is not regular.

By homomorphism

$L = L'$ and given that.

$L' = \{a^{2n} b^{2n}\}$ is not regular.

$\therefore$ L is not regular.

Pigeonhole principle

If we put n objects into m boxes (pigeonholes) and n > m, then atleast one box have more than one item in it.

Pumping lemma for regular Languages

* not every language is a regular language.

- Pumping lemma is a technique that can be used for showing a certain language is not regular, but it cannot be used to prove a language is Regular.
- Let L be a regular language. Then there exists a constant n (which depends on L) such that for every string x in L such that $|x| \geq n$, we can break x into three strings $x = uvw$, such that:

  i) $v \neq \epsilon$

  ii) $|uv| \leq n$

  iii) for all $i \geq 0$, the string uv^iw is also in L

ie, we can always find a nonempty string v not too far from the beginning of x that can be pumped; (repeating any number of times); Keeps the resulting string in the language L.

Proof

Suppose L is a regular langc. Then L = L(D) for some DFA, D.

Let n be no: of states in D.

Consider any string x of length n or more such that

$x = a_1 a_2 \cdots a_m$

where $m \geq n$ and each a_i is an input symbol.

For $i = 1, 2, 3, \ldots m$. define state P_i such that

$P_i = \delta^*(q_0, a_1 a_2 a_3 \cdots a_i)$.

where δ^* is the extended transition fn of D.

q_0 - start state of D

P_i - the state reached after reading the first i symbols of x.

note that $P_0 = q_0$.

Then, the set of states traversed by the string x in the transitional path from start state to final state are

$\{P_0, P_1, P_2 \cdots P_m\}$.

But no: of states in D is n

As $m \geq n$, by pigeon hole principle there exists atleast two states P_i and P_j such that $0 \leq i < j \leq n$ and $P_i = P_j$

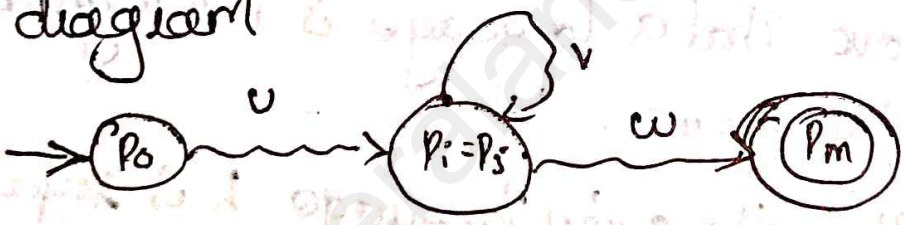
Now we can break x into three substrings, say

$u = a_1 a_2 \cdots a_i$

$v = a_{i+2} a_{i+2} \cdots a_j$

$w = a_{j+1} a_{j+2} \cdots a_m$

This can be represented by following transition diagram



Then we have

$|uv| = |a_1 a_2 \cdots a_i a_{i+1} \cdots a_j| = j \leq n$

ie, $|uv| \leq n$

$|v| = |a_{i+1} a_{i+2} \cdots a_j|$

$= j - i > 0$

ie, $|v| > 0$

$\therefore v \neq \varepsilon$

From the transition diagram, it is clear that automaton starts from P_0, on applying u it reaches state P_i. On applying string v, it comes back to P_i as $P_i = P_j$. So on applying string v any time (say $t \geq 0$) on P_i, it will be in same state P_i. And on applying w on P_i, the automaton will reach to the final state.

ie, for all $t \geq 0$, the string $uv^t w$ will reach the final state.

ie, $uv^t w \in L$ for all $t \geq 0$

Hence proved.

⇒ Steps to prove that a language is not Regular using pumping lemma.

1) • Assume that the given language L is regular
   • Therefore L satisfies the pumping lemma. Let the pumping length is n.

2) • select a string x such that $|x| \geq n$ satisfying $x = uvw$
   • select u such that $|uv| \leq n$
   • select v such that $|v| \geq 0$
   • Assign the remaining string to w

8) • select t such that uv^tw is not in L
• Hence L is not regular.

DT the $L = \{0^i 1^i \mid i \geq 1\}$ is not regular.

i) Assume that L is regular. Therefore L satisfies the pumping lemma. There exists an integer 'n' such that $\forall x \in L$. with $|x| \geq n$ $\exists u, v, w$. satisfying
$x = uvw$.
$|uv| \leq n$
$|v| > 0$ and for all non negative integer i.

(ii) $x = 0^n 1^n$

$|x| = 2n \geq n$

$x = \underbrace{000\ldots0}_{n}\underbrace{11\ldots1}_{n}$

now we may take $u = 0^p$, $v = 0^q$, $w = 0^r 1^n$

$x = uvw$
$= 0^p 0^q 0^r 1^n$
$= 0^{p+q+r} 1^n$.

$p+q+r = n$ — (1)
$|uv| = |0^p 0^q|$
$p+q \leq n$ — (2)
$|v| = |0^q| = q > 0$ — (3)

Let $i = 2$

$uv^2w = 0^p 0^q 0^q 0^l 1^n$

$= 0^{p+q+l+q} 1^n = 0^{n+q} 1^n \notin L$

$\therefore$ by pumping lemma it is not regular.

Q) Using pumping lemma, prove that the language $L = \{a^n b^n \mid n \geq 0\}$ is not regular.

soln:-

Assume L is regular and pumping length is p

Take string

$w = a^p b^p$ and divide 'w' in to three (x, y, z)

Assume $p = 3$

So, $w = a^3 b^3$

$= aaabbb$

$|xy| \leq n$

$|y| > 0$

$w \geq p$

case 1: $\underbrace{a}_{x}\underbrace{aa}_{y}\underbrace{bbb}_{z}$

Let $i = 2$

$\therefore xy^iz = aa(a)^2 bbb$

$= aaaa\,bbb$

$= a^4 b^3 \notin L$

case 2: $\underbrace{aa}_{x}\underbrace{aa}_{y}\underbrace{bbb}_{z}$

Let $i = 2$

$xy^iz = a(aa)^2 bbb$

$= aaaaabbb$

$= a^5 b^3 \notin L$

Case 3: $\underbrace{aaa}_{y}\ \underbrace{bbb}_{z}$

Here $|x| = 0$

Let $i = 2$

$$xy^iz = (aaa)^2 bbb$$

$$= aaaaaa\,bbb$$

$$= a^6 b^3 \notin L$$

∴ These strings are not regular. So the language L is not regular language (ie, Language 'L' violate pumping lemma).

Hence proved.

9) using pumping lemma for regular languages, prove that the language $L = \{0^n \mid n \text{ is a perfect square}\}$ is not regular?

soln:- $n = \{4, 9, 16, \ldots\}$

Assume 'L' is regular and pumping length is p.

Take string $w = 0^p$ and divide the string 'w' into three (x, y, z)

Assume $p = 4$

So, $w = 0^4 = 0000$

case 1: $\underset{x}{0}\underset{y}{00}\underset{z}{0}$

Let $i = 3$

$\therefore xy^iz = xy^3z$

$= 00(0)^30$

$= 0000000 = 0^6 \notin L$

case 2: $\underset{x}{00}\underset{y}{00}\underset{z}{0}$

Let $i = 3$

$\therefore xy^iz = xy^3z$

$= 0(00)^30$

$= 000000000$

$= 0^8 \notin L$

case 3: $\underset{x}{0}\underset{y}{000}$

Here $|z| = 0$

Let $i = 3$

$xy^iz = xy^3z$

$= 0(000)^3 = 0000000000$

$= 0^{10} \notin L$

∴ These strings are not regular. so the language L is not regular language.

Hence proved.

Closure Properties of Regular Languages

A language is closed under an operation if, whenever the operation is applied to members of the language, the result is also a member of the same language.

And this property is called closure property of a language.

→ Regular languages have the following closure properties.

(i) Closure under union

The union of 2 regular languages is regular. ie, If L & M are regular languages then L∪M is also regular.

Eg:- Let us take 2 regular expressions.

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

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

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

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

$\therefore RE(L_1 \cup L_2) = a^*$

.

(ii) closure under Intersection

The intersection of 2 regular languages is regular. If L & M are regular languages, then $L \cap M$ is also regular.

Eg:-

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

$L_1 = \{a, aa, aaa \cdots\}$

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

$L_1 \cap L_2 = \{aa, aaaa, aaaaaa \cdots\}$

$RE(L_1 \cap L_2) = aa(aa)^*$

(iii) closure under complementation

complement of a regular language is regular. If L is a regular language over alphabet Σ, then $L' = \Sigma^* - L$ is also a regular language.

Eg:- $RE = (aa)^*$

$L = \{\epsilon, aa, aaaa, aaaaaa \cdots\}$ even with ϵ.

complement of L is all the strings that is not in L.

So, $L' = \{a, aaa, aaaaa \cdots\}$ odd excluding null.

$\therefore RE(L') = a(aa)^*$ which is a RE itself

(iv) closure under difference

The difference of 2 RL is regular. If L and M are regular languages, then L-M is also regular.

Eg:-

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

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

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

$RE(L_1 - L_2) = a(aa)^*$

$\Rightarrow$ which is a RE

(v) closure under Reverse

The reverse of a RL is regular. If L is a regular language, then L^R is also regular.

Let $L = \{01, 10, 11, 10\}$

$RE(L) = 01 + 10 + 11 + 10$

$L^R = \{10, 01, 11, 10\}$

$RE(L^R) = 01 + 10 + 11 + 10$

$\Rightarrow$ which is regular.

(vi) closure under closure

The closure (* operation) of a regular language is regular.

If L is regular, then L^* is also regular.

(vii) Closure under concatenation

Concatenation of a RL is regular ie, if L & M are RL, then LM is also regular.

$RE_1 = (0+1)^*0$ & $RE_2 = 01(0+1)^*$

$L_1 = \{0, 00, 10, 000 \cdots\}$

$L_2 = \{01, 010, 011 \cdots\}$

$L_1L_2 = \{001, 0010, 0011, 0001, 00010, \cdots\}$

$\therefore RE = (0+1)^*001(0+1)^*$.

DFA State Minimization : - equivalence method

→ Suppose we have 2 states (P, Q)

→ we can say that P & Q are equivalent, when

$\delta(P, w) \in F \Rightarrow \delta(Q, w) \in F$

$\delta(P, w) \notin F \Rightarrow \delta(Qw) \notin F$.

→ If the above condition is satisfied, we can combine the states P and Q into one state.

→ The above condition can be used for combine the 2 states into a single state.

if $|w| = 0$, then states p & q are called 0 equivalent

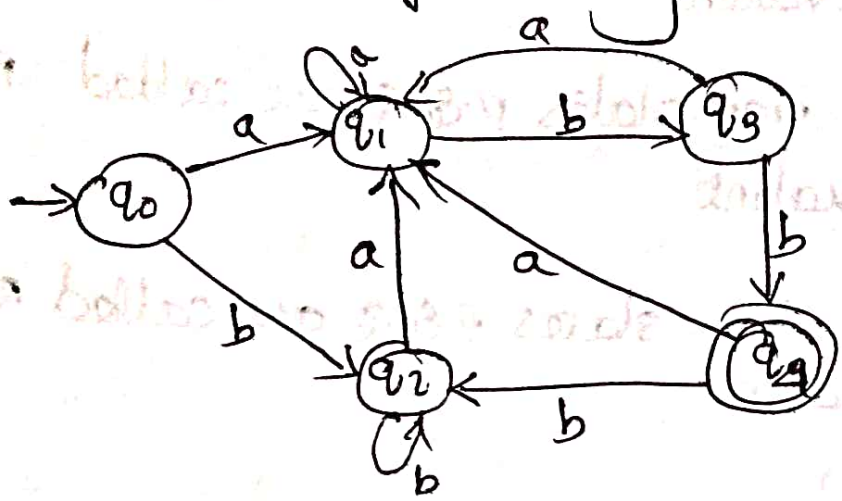
if $|w| = 1$, then states p & q are called 1 equivalent

if $|w| = 2$, then states p & q are called 2 equivalent

∴ In general.

if $|w| = n$, then states p & q are called n equivalent.

9) Minimize the following DFA?



Here q_4 is final state, & q_0 is initial state

Step 1 :- Identify the unreachable states (The states which are not reachable from initial state). If such state exist, delete it.

Step 2 :- Draw state transition table.

	a	b
q_0	q_1	q_2
q_1	q_1	q_3
q_2	q_1	q_2
q_3	q_1	$*q_4$
$*q_4$	q_1	q_2

Step 3:- Find 0 equivalent state (separate non final & final states)

$[q_0, q_1, q_2, q_3]$ $[q_4]$

- Find 1 equivalent states
  - check 1 equivalent of (q_0, q_1)
  - check 1 equivalent of (q_0, q_2)
  - check 1 equivalent of (q_2, q_3)

1 equivalent states

$[q_0, q_1, q_2]$ $[q_3]$ $[q_4]$

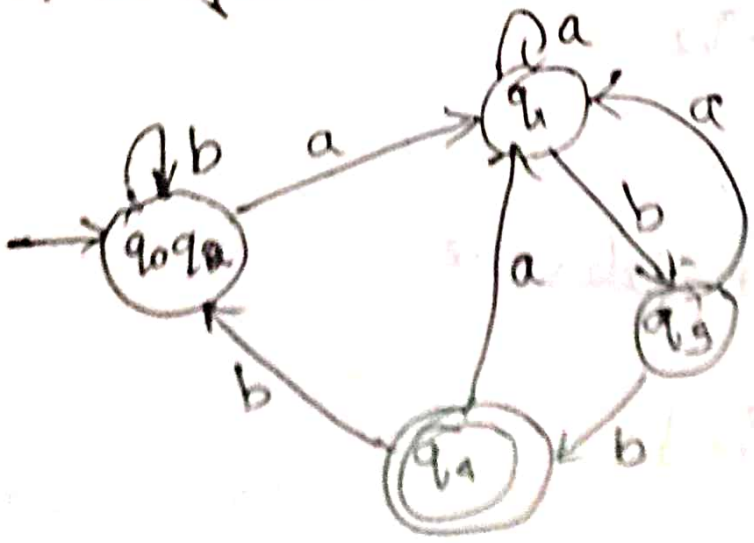
2nd equivalent states

$[q_0, q_2]$ $[q_1]$ $[q_3]$ $[q_4]$

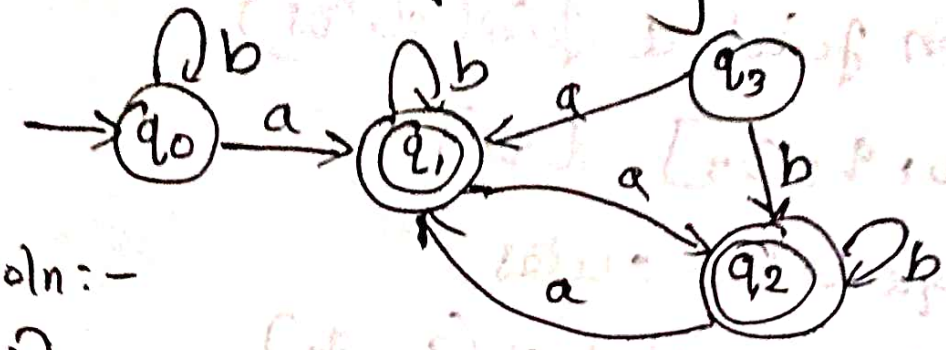
3 equivalent states

$[q_0, q_2]$ $[q_1]$ $[q_3]$ $[q_4]$

i.e, no further division is possible



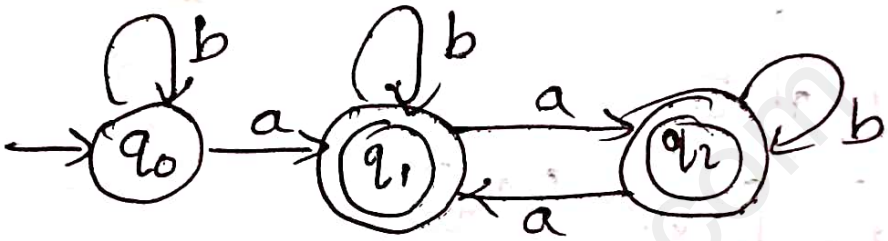
9) Minimize the following DFA



Soln:-

1)

∴ q_3 is unreachable state, remove it



2) transition table.

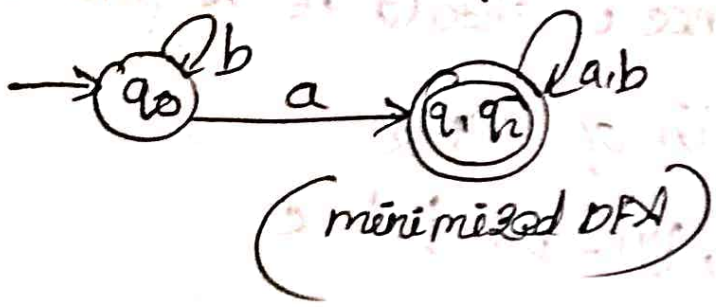
	a	b
q_0	$*q_1$	q_0
$*q_1$	$*q_2$	$*q_1$
$*q_2$	q_1	$*q_2$

3) 0 equivalent states are

$[q_0]$ $[q_1, q_2]$

1st equivalent states

$[q_1, q_2]$ $[q_0]$, no further division is possible



(minimized DFA)

Ultimate Periodicity

A subset X of N_0 is ultimately periodic if there exist $n_0 \geq 0$, $p \geq 1$ in N_0 such that for all $m \geq n_0$

$m \in X$ iff $m+p \in X$

or

$m \in X \Rightarrow m+p \in X$

Definition :- A subset $S \subseteq N_0$ is called ultimately periodic if there exists constants $n_0 \in N_0$ and $p \in N$ such that for all $n \geq n_0, n \in S$ iff $n+p \in S$., p is called a period of S.

Eg:- $\{0, 1, 4, 8, 10, 15, 20, 25, 30, 35, \ldots\}$ is up

ultimate periodicity is for length of strings accepted by the automation N. It can be directly demonstrated with pumping lemma (PL)

PL states that a strings can be broken down to 3 components. x, y, z. If then process y is in a loop hence it should be regular.

Consider the length of x, y, z

Assume $|x| = a$, $|y| = b$, $|z| = c$

As per PL rule, xz, xyz, xyyz, xyyyz ... are in L(N)

Consider their lengths, $|xz| = a+c$

$|xyz| = a+b+c$

$|xyyz| = a+b+b+c$

Take the set of lengths $L = \{a+c, a+b+c, a+2b+c, a+3b+c, a+4b+c \cdots\}$

Each element is b more than the previous element. ie, starting from $n_0 = a+c$

It is stated as there is a value $n_0 = a+c$ & period $p = b$, such that if some n_k is in L, then $n_k + b$ is in L.

Eg:- $L = a^{n^2} \mid n \geq 0$

$a^0, a^{1^2}, a^{2^2}, a^{3^2}, a^{4^2}$

$a^0, a^1, a^4, a^9, a^{16}$

length set = $\{0, 1, 4, 9, 16 \cdots\}$

There is no periodicity in the length set. according to ultimate periodicity. If a language is regular then its length set is ultimately periodic.

Here the length set is not ultimately periodic.

Hence the language is not regular.