Regular expression (RE) and Languages:-

aka. Language ~~generator~~ representer

Regular expression:-

- Regular expression are the algebric description of language used for representing regular languages, the languages accepted by by finite automaton.
- Many system uses regular expression as input language. Some of them are
  - Search commands such as UNIX grep.
  - Lexical analyser generator such as LEX or FLEX.

(pattern searching)

Lexical analyzer is a component of compiler that breaks the source program into logical unit called tokens.

  - Browsers and many other text formatting programs.

Operators of Regular Expression:-

⇒ Regular expressions denote languages. for a simple example, the regular expression $01^* + 10^*$ denotes the language consisting of all strings that are either a single 0 followed by any number of 1's or a single 1 followed by any no. of 0's.

There are three operators that are used to generate the languages that are regular.

① **Union** (U/|/+):-

The union of two languages L and M

Date: Page:

denoted as $L \cup M$ or $L|M$ or $L+M$, is the set of strings that are in either L or M, or both.

i.e.

If L_1 and L_2 are any two regular languages then

$$L_1 \cup L_2 = \{ s \mid s \in L_1 \text{ or } s \in L_2 \}$$

for example,

$$L_1 = \{00, 11\}, \quad L_2 = \{\epsilon, 10\}$$

$$L_1 \cup L_2 = \{\epsilon, 00, 10, 11\}$$

(2) Concatenation(.) :-

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

i.e. If L_1 and L_2 are any two regular languages then

$$L_1 \cdot L_2 = \{ l_1 \cdot l_2 \mid l_1 \in L_1 \ \& \ l_2 \in L_2 \}$$

eg:- $L_1 = \{00, 11\}$ & $L_2 = \{\epsilon, 10\}$

$L_1 \cdot L_2 = \{00, 0010, 11, 1110\}$

$L_2 \cdot L_1 = \{00, 11, 1000, 1011\}$

So, $L_1 \cdot L_2 \neq L_2 \cdot L_1$

③ Kleen closure (*) :-

The closure/star or kleen closure of a language L is denoted by L^* and represents the set of strings that can be formed by taking any number of strings from L, possibly with repetitions and concatenating all of them.

i.e. if L is any regular language then

$L^* = L^0 = L_0 \cup L_1 \cup L_2 \cup \cdots$

eg:- $L = \{0, 1\}$ then L^* is all strings of 0's and 1's

If $L = \{0, 11\}$ then L^* is,

011, 11, 0, ϵ, 110, 11110 but not 01011 or 101.

eg: $L = \{0, 11\}$ then

$L^0 = \{\epsilon\}$ → selection of zero string for zeroth power.

$L^1 = \{0, 11\}$ → selection of one string

$L^2 = \{00, 1111, 011, 110\}$ → selection of 2 string

$\boxed{L^+ = L^* - \epsilon}$ → positive closure.

$L^3 = \{000, 0011, 0110, 1100, 01111, 11011, 11110, 111111\}$

Precedence of regular expression Operator :-

1. Kleen closure — high
2. Concatenation
3. Union — low

Date: Page:

Example:-

Write a regular expression for the set of string that consists of alternating 0's and 1's over {0,1}

Soln:-

first part:- We have to generate the language {01, 0101, 010101, ----}

second part:- We have to generate the language {10, 1010, 101010, ------}.

So lets start first part.

Here, we start with the basic RE 0 and 1 that represent the language {0} and {1} respectively. Now if we concatenate these two RE, we get the RE 01 that represent the language {01}. Then to generate the language of zero or more occurance of 01, we take Kleen closure i.e. the RE $(01)^*$.

Which represent the language {01, 0101, ----}

Similarly, the RE for second part is $(10)^*$

Now, finally we take union of above two. first part and second part to get the required RE.

i.e. $(01)^* + (10)^*$ represent the given language.

Design of RE:-

$\Sigma = \{x, y, z\}$

Rule 1: (Zero or more)

x^* i.e. $\{\epsilon, x, xx, xxx, ----\}$

Suppose xy

$\Rightarrow (xy)^* = \{\epsilon, xy, xyxy, xyxyxy, ----\}$

If $xy^* = \{x, xy, xyy, xyyy, ----\}$

Rule 2: (One or more)

$xx^* \Rightarrow \{x, xx, xxx, ----\}$

$\Downarrow$

x^*x or x^+

If xy then $xy(xy)^*$ or $(xy)^*xy$ or $(xy)^+$

Rule 3: (Zero or ~~more~~ one)

$(x+\epsilon)$ or $(\epsilon+x)$

Rule 4: (~~Any~~ string at all)

$(x+y+z)^*$ $\quad \Sigma = L_1 \cup L_2 \cup L_3$

$L_0 = \{\epsilon\}$

L_1 $(x+y+z)^1 \Rightarrow x, y, z,$

L_2 $(x+y+z)^2 \Rightarrow xy, xz, xx, yy, yx, yz, zz, zx, zy, ----$

L_3 $(x+y+z)^3 \Rightarrow xyz, xxx, zzz, yyy, yxz, zyx, ---$

Rule 5:-

Any non-empty string at all.

$(x+y+z) \cdot (x+y+z)^* = (x+y+z)^* \cdot (x+y+z) = (x+y+z)^+$

Rule 6:- Any string not containing.

not contain x then $(y+z)^*$

Rule 7:- Any string containing exactly one.

$(y+z)^* \cdot x \cdot (y+z)^*$

Date: Page:

eg:① Find the regular expression for the Language $L = \{000, 001, 010, 011, 100, 101, 110, 111\}$

⇒ $\Sigma = \{0, 1\}$

$|w| = 3$

$R.E = (0+1).(0+1).(0+1)$

② Construct a RE for all strings containing exactly one x over $\Sigma = \{x, y, z\}$

⇒ $R.E. = (y+z)^* x (y+z)^*$

③ Construct a RE for the set of all strings over $\{x, y\}$ with three consecutive y's.

⇒ $(x+y)^* . yyy . (x+y)^*$

④ Construct a RE for the set of all string over $\{a, b\}$ begining with aa.

⇒ $aa(a+b)^*$

⑤ Construct a RE for the set of all strings over $\{x, y\}$ ending with xx & begining with y.

⇒ $y(x+y)^* xx$

(6) Construct a RE for the set of all string over {a,b} ending in either aba or aab

⇒ $(a+b)^*(aba+aab)$

(7) Construct a RE for the set of all strings containing no more than three a's over {a,b,c}

⇒ $(b+c)^*.(\epsilon+a).(b+c)^*(\epsilon+a).(b+c)^*(\epsilon+a).(b+c)^*$

(8) no. 2a's coming together :-

$L = \{\epsilon, b, bb, bbb, a, ab, aba, ababa, ababa, ba, bab, baba -- \}$

⇒ $(b+ab)^* + (b+ab)^*a$

$(b+ab)^*(\epsilon+a)$

OR

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

$(a+\epsilon)(b+ba)^*$

Regular Language :-

⇒ Let Σ be an alphabet, the class of regular language over Σ is defined inductively as,

- $\emptyset$ is a regular language representing empty language.
- $\{\epsilon\}$ is a regular language representing empty string.
- For each $a \in \Sigma$, $\{a\}$ is a regular language
- If $L_1, L_2, \ldots L_n$ are regular language, then so $L_1 \cup L_2 \cup \ldots L_n$.
- If $L_1, L_2, \ldots L_n$ are regular language, then so $L_1 \cdot L_2 \cdot L_3 \cdot \ldots L_n$.
- If L is regular language, then so L^*.

Application of Regular Language :-

⇒ Validation :- email, password validation

Search & selection :- browsers, text formatting programs

Tokenization :- Converting a sequence of characters into words, token (like keywords, identifiers) for later interpretion

Algebric Rules/laws of Regular expressions:-

1. Commutative:
   $r+s = s+r$ i.e. $r \cup s = s \cup r$
   but $r \cdot s \neq s \cdot r$

2. Associative:
   $t+(r+s) = (t+r)+s$
   $t.(r.s) = (t.r).s$

3. Distributive:
   $r(s+t) = rs+rt$: left distribution
   $(s+t)r = sr+tr$: right distribution

4. Identity:
   $r+\phi = \phi+r = r$ ϕ for union
   $\epsilon \cdot r = r \cdot \epsilon = r$ ϵ for Concatenation

5. Idempotent law of union
   $r+r = r$

6. Law of closure
   $(r^*)^* = r^*$
   $r^+ = r \cdot r^* = r^* r$

$r^* = \epsilon + r + rr + rrr + \cdots$

$r \cdot r^* = r\epsilon + rr + rrr + rrrr + \cdots$

$\therefore r\epsilon = r$

$\therefore r^+ = rr^*$

Finite Automata & regular expression :-

⇒ Regular expression approach of describing language is fundamentally different from the finite automata approach.

⇒ These two notation turn out to represent exactly the same languages which are called "regular languages".

- We have already studied that DFA and two kinds of NFA; with & without ε-transition. which accept the same class of languages.
- In order to show that the regular expressions define the same class, we must show that;

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

2) Every language defined by a regular expression is defined by one of these automata. For this part of proof, the easiest is to show that there is an NFA with ε-transitions accepting the same language.

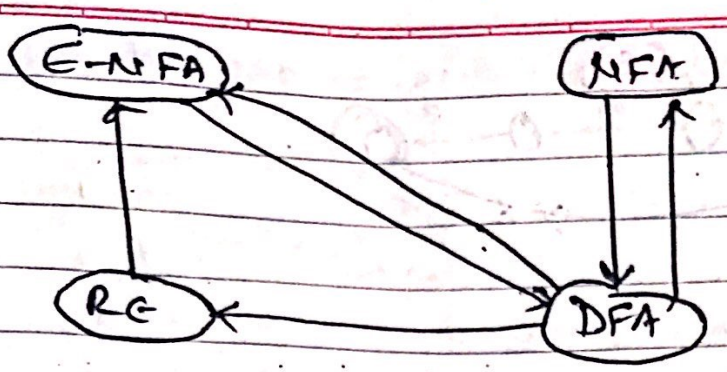
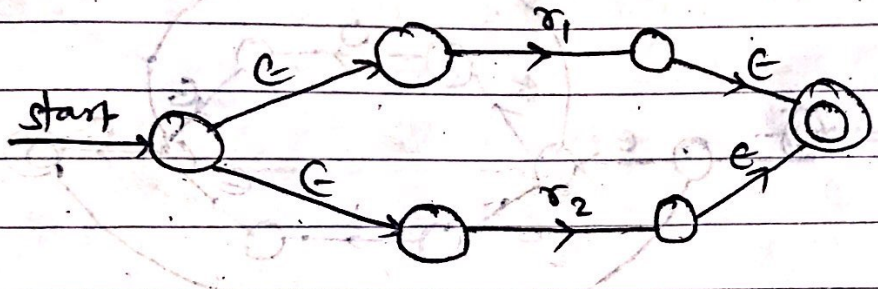


fig:- Plan for showing the equivalence of four different notations for regular language.

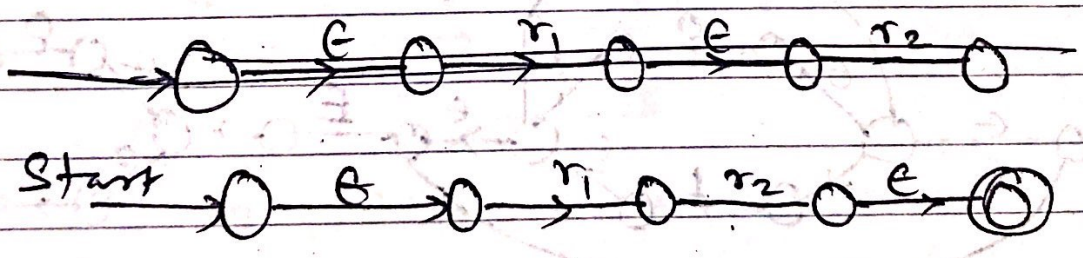
(Conversion from R.E. to ϵ-NFA)

Rules:

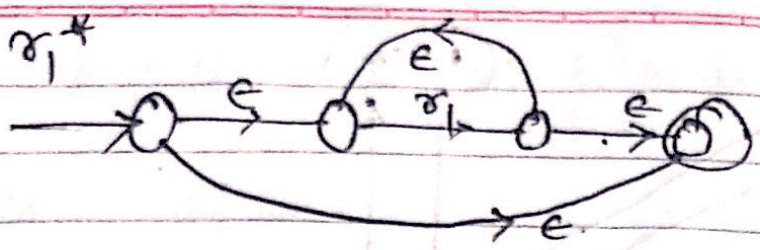
(i) $r_1 + r_2$ eg: $(1+0)$; $r_1 = 1$, $r_2 = 0$



(ii) $r_1 \cdot r_2$ eg (11) ; $r_1 = 1$, $r_2 = 1$

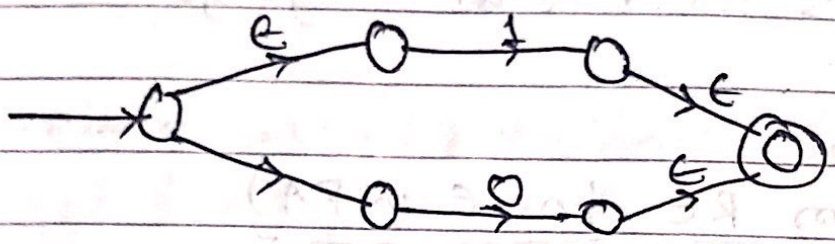


(iii) r_1^*

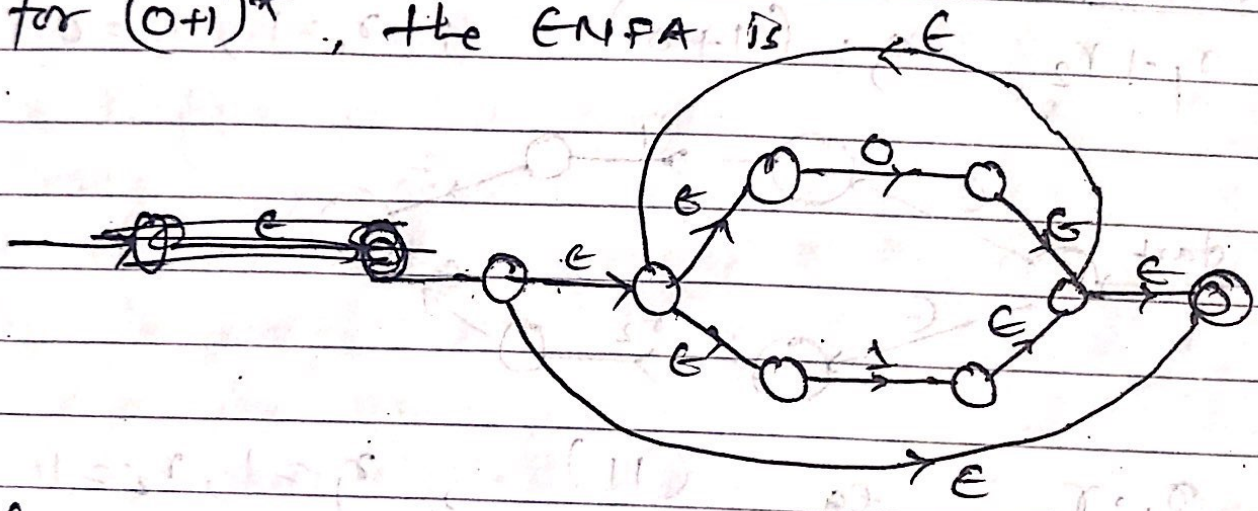


Examples :-

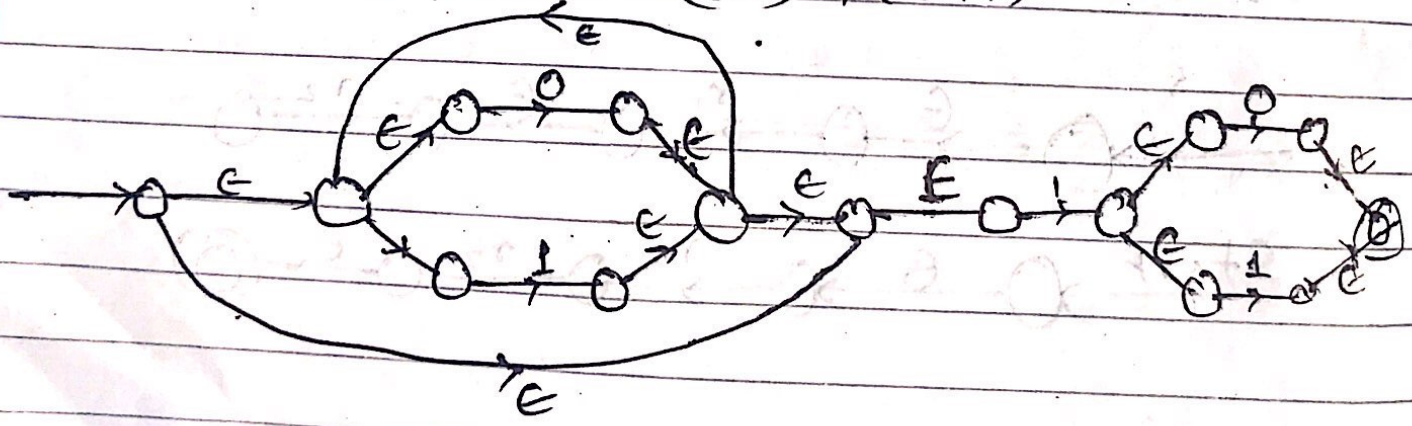
(1) For regular expression (1+0) the ε-NFA is



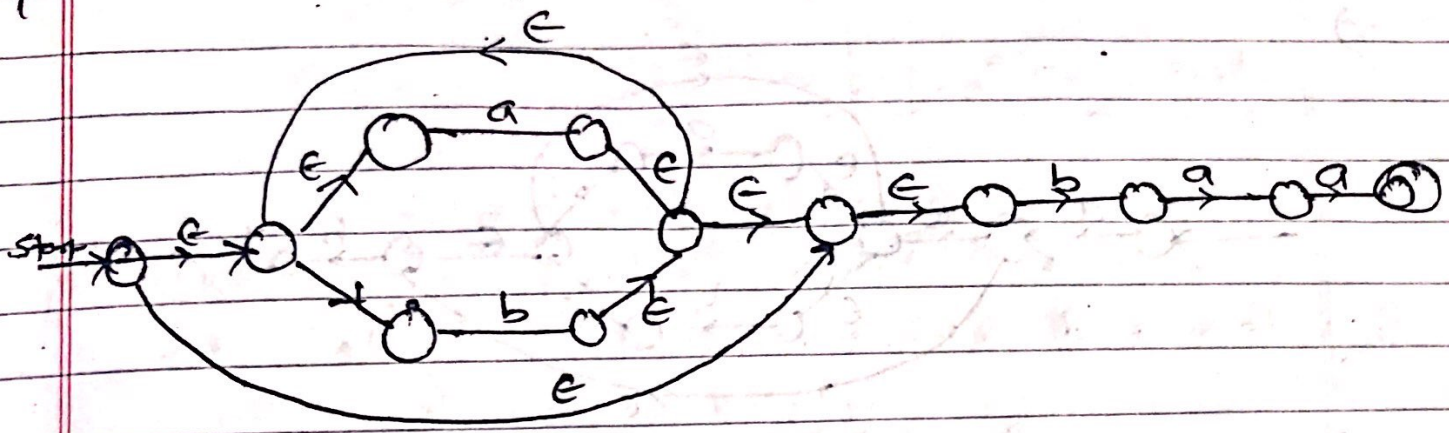
(2) for (0+1)*, the ENFA is



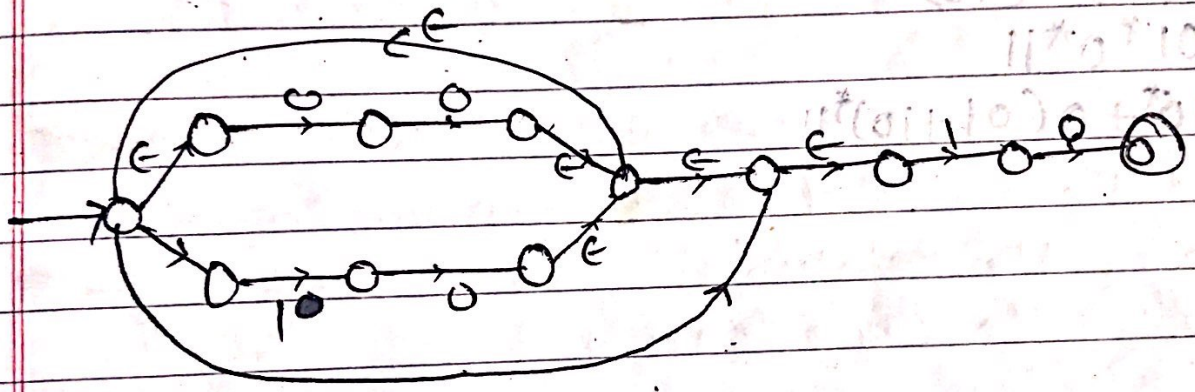
(3) for regular expression (0+1)*1(0+1)



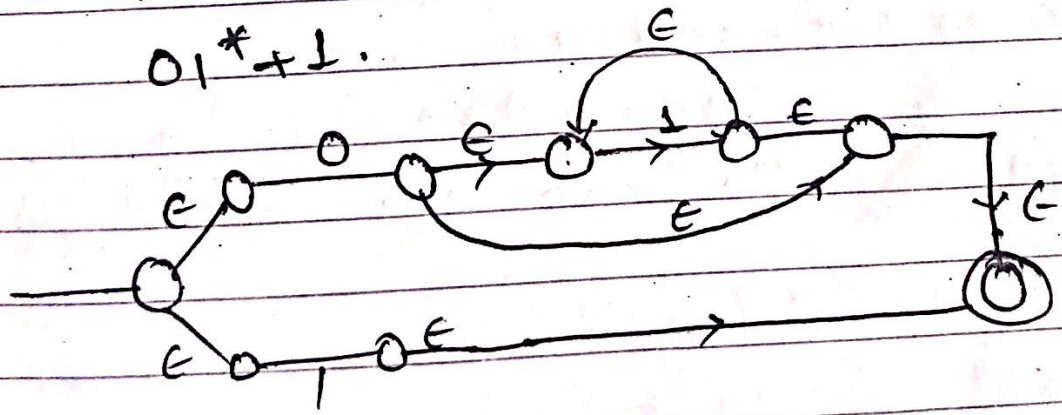
5. For regular expression $(a+b)^*baa$ construct ϵ-NFA.



5. For regular expression $(00+10)^*10$ the ϵ-NFA is

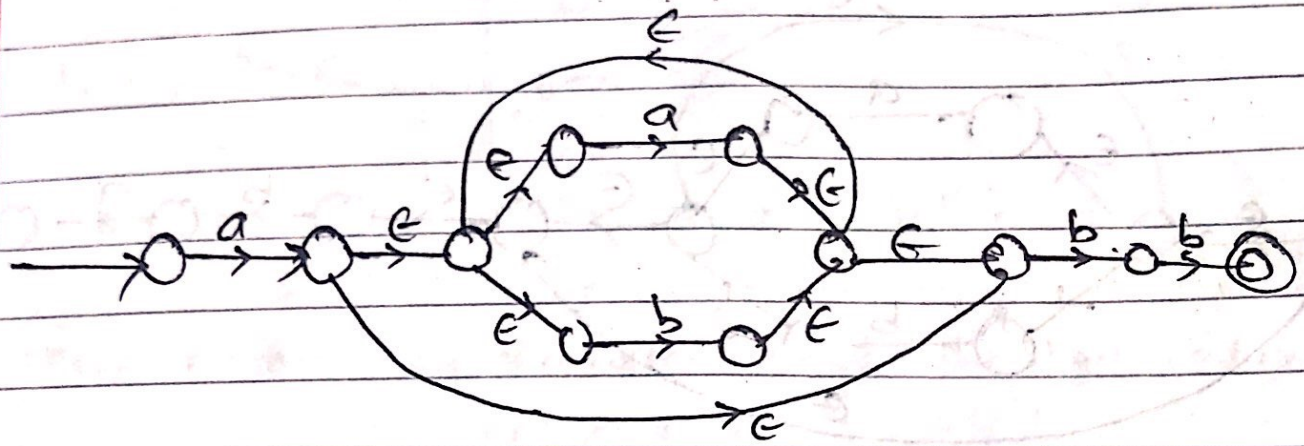


6. Construct ϵ-NFA for the regular expression 01^*+1.



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

⇒



Q1 $(0+1)^*(00+11)(0+1)^*$

Q2 $(00+1)^*(10)^*$

Q3 001^*0^*11

Q4 $010^*+0(01+10)^*11$

Date: Page:

also for NFA

Conversion from DFA to RE :-

Arden's Theorem :-

Let P and q be the two regular expressions such that P doesnot contain any empty string then a Regular Expression r of the form

$$r = q + rp$$

has a unique solution of the form

$$r = qp^*$$

proof:- Here, $r = q + rp$ ——— (I)

- put the value of $r = q + rp$ on right hand side of the relation (I), so

$$r = q + (q + rp)p$$

$$r = q + qp + rp^2 \quad \text{——— (II)}$$

- Again put the value of $r = q + rp$ in eqn (II), we get

$$r = q + qp + (q + rp)p^2$$

$$= q + qp + qp^2 + rp^3$$

Continuing in the same way, we will get as;

$$r = q + qp + qp^2 + qp^3 + \text{- - - - -}$$

$$= q(\epsilon + p + p^2 + p^3 + \text{- - - - - -})$$

$\therefore r = qp^*$ proved

Date: Page:

To convert the given DFA into a regular expression, there are some of the assumptions regarding the transition system.

- There should not have ϵ-transition
- There must be only one initial state.
- The states in the DFA are as:
  $q_1, q_2, \ldots\ldots q_n$ (Any q_i is final state)
- W_{ij} denotes the regular expression representing the set of labels of the edges from q_i to q_j.

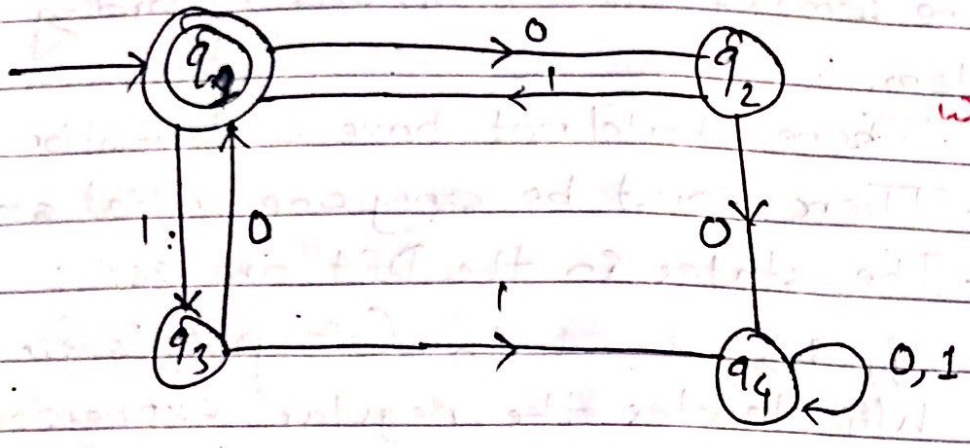
Thus we can write expression as,

$$q_1 = q_1 w_{11} + q_2 w_{21} + q_3 w_{31} + \ldots\ldots + q_n w_{n1} + \epsilon$$
$$q_2 = q_1 w_{12} + q_2 w_{22} + q_3 w_{32} + \ldots\ldots + q_n w_{n2}$$
$$q_3 = q_1 w_{13} + q_2 w_{23} + q_3 w_{33} + \ldots\ldots + q_n w_{n3}$$
$$\vdots$$
$$q_n = q_1 w_{1n} + q_2 w_{2n} + q_3 w_{3n} + \ldots\ldots + q_n w_{nn}$$

Solving these eqn for q_i in terms of w_{ij} gives the regular expression equivalent to given DFA.

Date: Page:

eg Convert the following DFA into regular expression.



w.r.t to final state

Soln:- Let the equations are :-

$$q_1 = q_2 1 + q_3 0 + \epsilon \quad \text{——— (1)}$$
$$q_2 = q_1 0 \quad \text{——— (2)}$$
$$q_3 = q_1 \cdot 1 \quad \text{——— (3)}$$
$$q_4 = q_2 \cdot 0 + q_3 1 + q_4 0 + q_4 1 \quad \text{——— (4)}$$

putting the value of q_2 and q_3 in eqn (1)

$$q_1 = q_1 01 + q_1 10 + \epsilon$$
$$\therefore q_1 = q_1 (01+10) + \epsilon$$
$$\therefore q_1 = \epsilon + q_1 (01+10)$$

by applying Arden's theorem,

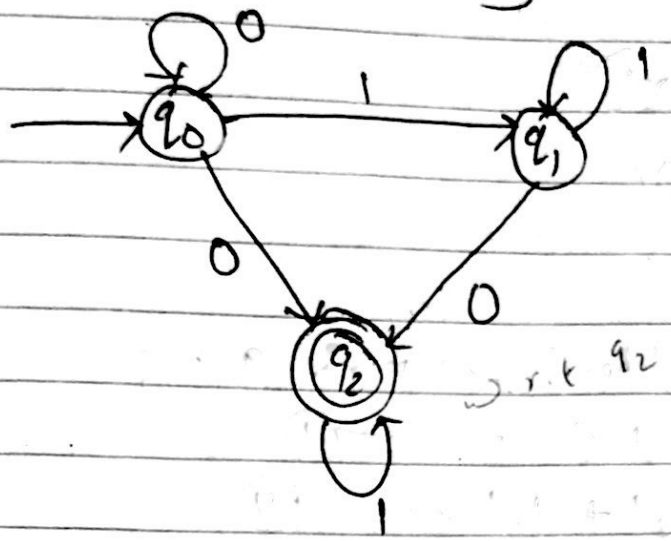
$$\therefore q_1 = \epsilon (01+10)^*$$

$$\therefore \boxed{q_1} = (01+10)^*$$

which is the required regular expression w.r.t final state

Date: Page:

① Convert the following NDFA into RE.



Solr:

equations are :-

$$q_0 = q_0 0 + \epsilon \quad \text{——} (I)$$
$$q_1 = q_0 1 + q_1 1 \quad \text{——} (II)$$
$$q_2 = q_0 0 + q_1 0 + q_2 1 \quad \text{——} (III)$$

Apply Ardens theorem in (I)

$$\underset{r}{q_0} = \underset{r\ p}{q_0 0} + \underset{q}{\epsilon}$$

$$q_0 = \epsilon(0)^* = 0^* \quad \text{——} (IV)$$

⇒ check if ardens theorem could be applied in any other eq^ns

from (II)

$$\underset{r}{q_1} = \underset{q}{q_0 1} + \underset{r\ p}{q_1 1}$$

$$q_1 = q_0 11^* = 0^* 11^* \quad \text{——} (V)$$

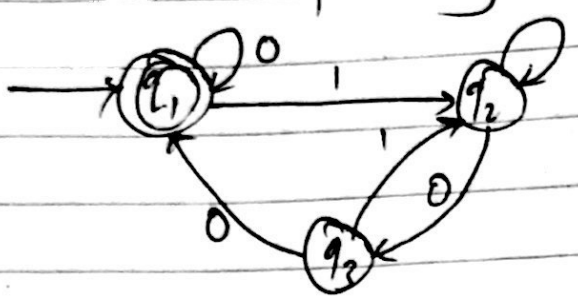
from (III)

$$\underset{r}{q_2} = \underset{q}{q_0 0 + q_1 0} + \underset{r\ p}{q_2 1}$$

$$q_2 = (q_0 0 + q_1 0) 1^* = (0^* 0 + 0^* 11^* 0) 1^* \quad \text{——} (VI)$$

req soln of R.E.

Q = Convert the following DFA into R.E.



Soln:- from the figure, the equations are;

$$q_1 = q_1 0 + q_3 0 + \epsilon \quad \text{——— (I)}$$
$$q_2 = q_1 1 + q_2 1 + q_2 1 \quad \text{——— (II)}$$
$$q_3 = q_2 0 \quad \text{——— (III)}$$

check if arden's theorem can't be applied

from eqn (II) & (III)

$$q_2 = q_1 1 + q_2 1 + q_2 01$$
$$q_2 = q_1 1 + q_2 (1 + 01)$$
$$(q = q_2,\ r = q_1 1,\ p = 1 + 01)$$

$$q_2 = q_1 1 (1+01)^* \quad \text{——— (IV)}$$

from (I) & (III)

$$q_1 = q_1 0 + q_2 00 + \epsilon \quad \text{——— (V)}$$

w.r.t q_1

from (IV) & (V)

$$q_1 = q_1 0 + q_1 1 (1+01)^* 00 + \epsilon$$
$$q_1 = q_1 (0 + 1(1+01)^* 00) + \epsilon$$
$$(q = q_1,\ r = \epsilon,\ p = 0 + 1(1+01)^* 00)$$

$$q_1 = \epsilon (0 + 1(1+01)^* 00)^*$$

$$\therefore q_1 = (0 + 1(1+01)^* 00)^*$$

is the reqd R.E.

Pumping lemma for regular language:-

⇒ Pumping lemma is used to determine the class of language of finite automata. ie. it is used to determine a language is not regular.

Statement:-

Let 'L' be a regular Language and a string $w \in L$. Let 'm' be the total no. of states of FA and 'n' be the Length of the string such that $m \leq n$.

Then, w can be decomposed into substrings x, y, z such that,

1. $y \neq \epsilon$ decompose करते null वाला नहीं आएगा।
2. $|xy| \leq n$
3. $xy^i z \in L$ for all $i \geq 0$

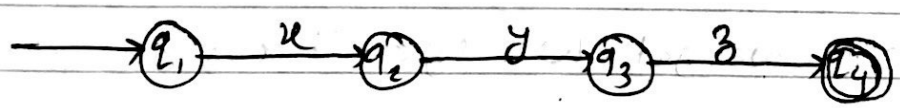
proof:-

Let L be a regular language and $w \in L$.

Let $w = a_1 a_2 a_3 \text{-----} a_n$ and the set of states are $q_1, q_2, q_3 \text{-----} q_m$.

~~let $w = xyz$ belongs to~~ let $w = xyz \in L$

Now, its finite automata is

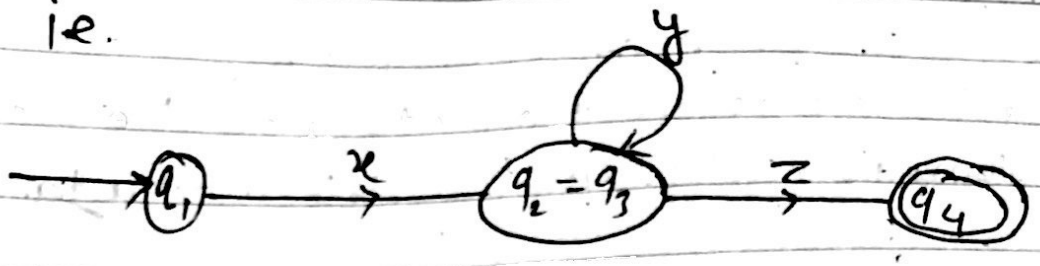


here, $|w| = 3$, but total states are 4.

By pigeon hole principle, any of the two states of FA must coincide.

According to our requirement state q_2 & q_3 coincide.

ie.



Now in general form:-

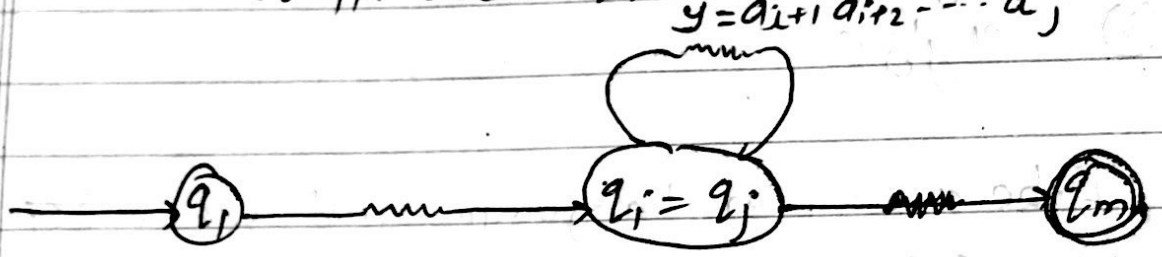
$w = a_1 a_2 a_3 - - - - - a_n$ can be decomposed as,

$x = a_1 a_2 a_3 - - - - a_i$

$y = a_{i+1} a_{i+2} a_{i+3} - - - - a_j$

$z = a_{j+1} a_{j+2} - - - - a_n$

Now our FA becomes,



Now we have to show that $xy^iz \in L$ for $i \geq 0$

When $i=0$, we get $w = xz$ which is processed by FA.

When $i=1$, we get $w = xyz$, it also accepted by FA.

Similarly, FA processes the strings for all the values of i. Hence, we say that $xy^iz \in L$ for all $i \geq 0$.

proved

Date: Page:

Example:-

Show that the language $L = \{a^n b^n : n > 0\}$ is not regular.

⇒ Let L be a regular Language and $w \in L$ (proof by contradiction)

let $w = a^p b^p$ $p \geq 1$

Now,

Using pumping lemma, w can be decomposed into x, y and z as,

$x = a^q$ (to simplicity we do only of 'a' not of 'b')

$y = a^r$, $r > 0$

$z = a^{p-(q+r)} b^p$

Now, Let pumping factor i=2

$xy^2z = a^q (a^r)^2 a^{p-(q+r)} b^p$

$= a^{q+2r+p-q-r} b^p$

$= a^{p+r} b^p$

$\because r > 0$, $p+r > p$

Therefore, $a^{p+r} b^p$ is not of the form $a^p b^p$.

i.e. $xy^2z \notin L$

Hence, the given language is not regular. by contradiction.

Proof:-

$w = a^p b^p$ $p = 5$

$= \underbrace{aa}_{x} \underbrace{aaa}_{y} \underbrace{bbbbb}_{z}$

$x = a^q = aa$ $q = 2$

$y = a^r = aaa$ $r = 3$

$z = a^{5-(2+3)} b^5$

$= a^0 b^5 = bbbbb$

$\therefore xy^iz = xy^2z = aaaaaaabbbbb$

$a^8 b^5 \neq a^5 b^5$

$y \neq \epsilon$, $y > 0$

$|xy| \leq n$

$5 \leq 10$

Additional problem

$L = \{0^n \mid n \text{ is a perfect square}\} \Rightarrow 0^{p^2}$

$\{ \text{---}\,''\,\text{---} \text{ perfect cube} \} \Rightarrow 0^{p^3}$

$L = \{0^k\}$ k is prime number $\} \Rightarrow 1, 2, 3, 5, 7, 11 \ldots$

Date: Page:

Show that the language $\{L = 1^{n^2} : n > 0\}$ is not regular.

⇒ Let L be a regular language and $w \in L$.

let $w = 1^{p^2}$ $p \geq 1$

now,

Using pumping lemma, w can be decomposed into x, y and z as,

$x = 1^{q^2}$

$y = 1^{r^2}$, $r > 0$

$z = 1^{p^2 - (q^2 + r^2)}$

now, let pumping factor is $i = 2$

$\therefore xy^2z = 1^{q^2} \cdot (1^{r^2})^2 \cdot 1^{p^2-(q^2+r^2)}$

$= 1^{q^2 + 2r^2 + p^2 - q^2 - r^2}$

$= 1^{p^2 + r^2}$.

$\because r > 0, \; r^2 > 0$

$\therefore p^2 + r^2 > p^2$

Therefore, $1^{p^2+r^2}$ is not of the form 1^{p^2}

Hence $xy^2z \notin L$. and is not regular by contradiction.

Rough:

$w = 1^9 = |||||||||$ $p = 3 \therefore p^2 = 9$

$x \; y \; z$

$x = 1^1 = 1$ $q = 1 \therefore q^2 = 1$

$y = 1^4 = 1111$ $r = 2$ so $r^2 = 4$

$z = 1^{9-(1+4)} = 1^4 = 1111$

$\therefore xy^1z = xy^2z = 1 \; 1111 \; 1111 \; 1111 = 1^3 = 1^9$

Show that the language $\{L = xx^R ; x \in \{0,1\}^*\}$ is not regular.

⇒

Let L be a regular Language and $w \in L$

let $x = 1^n 0^n$

then $x^R = 0^n 1^n$

$xx^R = 1^n 0^n 0^n 1^n$

Let $w = 1^p 0^p 0^p 1^p$

now,

Using pumping lemma, w can be decomposed into x, y & z as,

$x = 1^q$

$y = 1^r$

$z = 1^{p-(q+r)} 0^p 0^p 1^p$

now,

$$xy^2z = 1^q 1^{(r)^2} 1^{p-(q+r)} 0^p 0^p 1^p$$

$$= 1^{q+2r+p-q-r} 0^p 0^p 1^p$$

$$= 1^{p+r} 0^p 0^p 1^p$$

$\therefore r > 0$

$\therefore p + r > p$

Hence, $1^{p+r} 0^p 0^p 1^p$ is not of the form $1^p 0^p 0^p 1^p$

Therefore $xy^2z \notin L$.

Rough.

$w = 1^3 0^3 0^3 1^3 \quad p = 3$

$= \underbrace{111}_{x} \underbrace{}_{y} 000000111$

$x = 1^1 \quad q = 1$

$y = 1^2 \quad r = 2$

$z = 1^{9-(2+1)} 0^3 0^3 1^3 = 1^6 0^3 0^3 1^3$

$xy^iz \underset{i=2}{=} xy^2z = 1111 \cdot 111111 0^3 0^3 1^3$

$= 1^4 1^{y^2} 0^3 0^3 1^3$

decision properties:-

1. Is the language described empty? (reachable/unreachable)
2. Is the particular string w in the described language (for DFA w ends in accepting state)
3. Do the two descriptions of a language actually describes the same language? (equivalance)

Minimization of DFA :-

⇒ DFA minimization is the task of transforming a given DFA into an equivalent DFA that has a minimum no. of states. Here, two DFAs are called equivalent if they recognize the same language.

⇒ During the course of minimization, it involves — identifying the equivalent state & distinguishable states.

Equivalent states:-

- Two states P and q are called equivalent states, denoted by $P \equiv q$ if and only if for each — input string x, $\hat{\delta}(P,x)$ is a final state & iff $\hat{\delta}(q,x)$ is a final state.

Distinguishable states:-

- Two states P and q are said to be distinguishable state if there exists a string x, such that $\hat{\delta}(P,x)$ is a final state, $\hat{\delta}(q,x)$ is a non final state.

Table filling algorithm (Myhill nerode algorithm)

⇒ steps:-

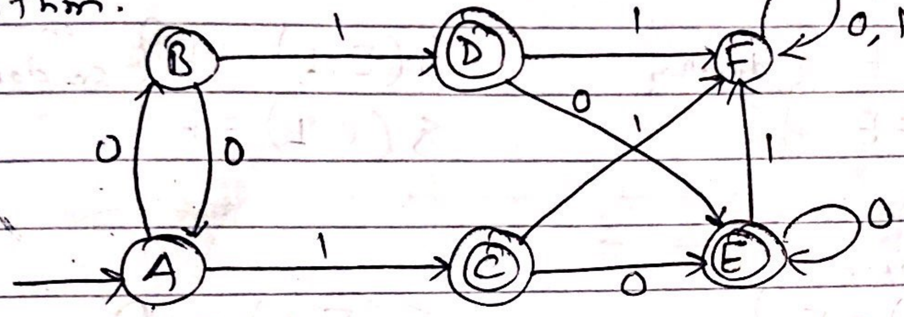
1) Draw a table for all pairs of states (P,Q)
2) Mark all pairs where $P \in F$ and $Q \notin F$.
3) If there are any unmarked pairs (P,Q)

such that $[\delta(p,x), \delta(q,x)]$ is marked, then mark $[p, q]$ where x is an input symbol.
Repeat this until no more markings can be made.

4) Combine all the unmarked pairs and make them a single state in the minimized DFA.

⇒ ~~(1, 2) steps~~
~~[illegible]~~

Q Minimize the following DFA using table filling algorithm.



Draw a table as,

Step 1 & 2

	A	B	C	D	E	F
A		AB				
B	AB					
C	✓	✓				
D	✓	✓				
E	✓	✓				
F	✓	✓	✓	✓	✓	

(C, A) ∵ C is a final & A is non final. & so on.

(F, A) both non final ∴ unmark in first iteration but might be marked in later iteration or next step.

do serially for all pairs in table

Date: Page:

Step 3: (1st iteration)

for (B, A) pair

$\delta(B,0) = A$, $\delta(A,0) = B$ } pair if ~~A is marked~~ if not marked do nothing

$\delta(A,1) = C$, $\delta(B,1) = D$ } Unmarked do nothing

for (D, C) pair

$\delta(D,0) = E$, $\delta(C,0) = E$ } do nothing

$\delta(D,1) = F$, $\delta(C,1) = F$ } do nothing

For (E, C) pair

$\delta(E,0) = E$, $\delta(C,0) = E$ } do nothing

$\delta(E,1) = F$, $\delta(C,1) = F$ } not present on table so do nothing

for (E, D) pair

$\delta(E,0) = E$, $\delta(D,0) = E$

$\delta(E,1) = F$, $\delta(D,1) = F$

for (F, A) pair

$\delta(F,0) = F$, $\delta(A,0) = B$ } not marked previously

$\delta(F,1) = F$, $\delta(A,1) = C$ } check Yes (marked)

Now mark F, A also

for (F, B) pair

$\delta(F,0) = F$, $\delta(B,0) = A$ } marked, So mark (F, B) also

$\delta(f,1) = f$, $\delta(B,1) = D$ } marked ∴

Date: Page:

- No more new markings, so end the process.
⇒ 2nd iteration (do yourself)

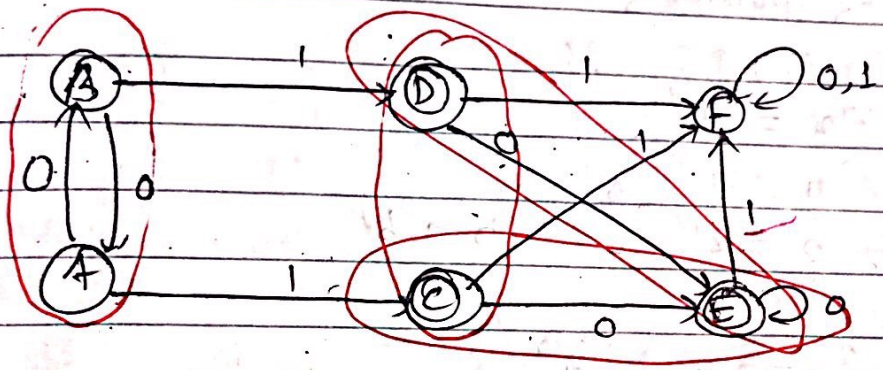
also in next iteration

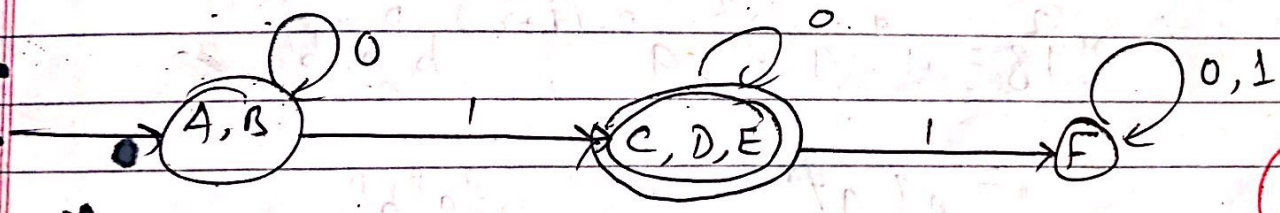
Step 4:

Combine all unmarked pair & make single state.

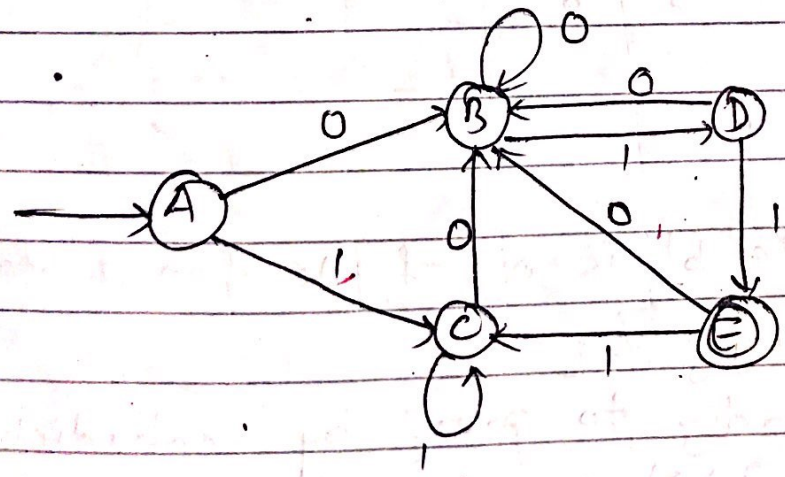
(A,B) (D,C) (E,C) (E,D) → unmarked pair

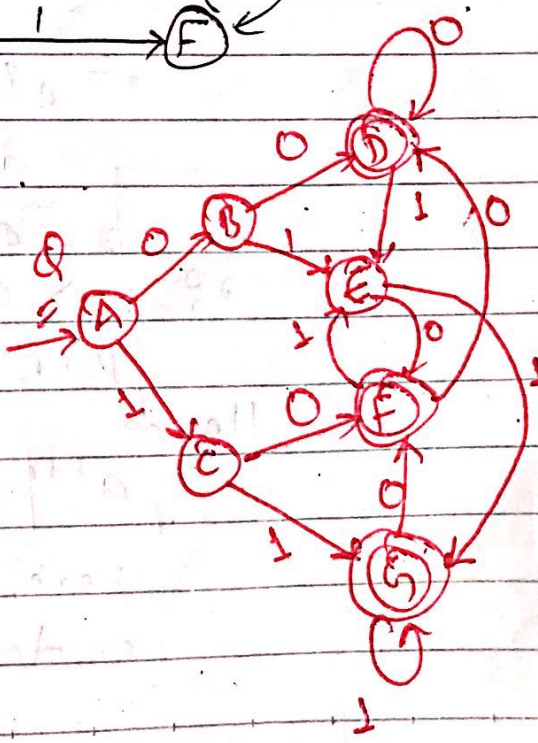
Original DFA is,





final minimized DFA.





Q) Show that the language $\{L = xx \mid x \in \{a,b\}^*\}$ is not regular.

⇒ Let L be the regular language and $w \in L$

let $x = a^n b^n$

$xx = a^n b^n a^n b^n$

let $w = a^p b^p a^p b^p$

now,

Using pumping lemma, w can be decomposed into x, y & z as,

$x = a^q$

$y = a^r$

$z = a^{p-(q+r)} b^p a^p b^p$

now,

~~$xy^2z = a \cdot a^q (a^r)^2$~~

$xy^2z = a^q (a^r)^2 \cdot a^{p-(q+r)} b^p a^p b^p$

$= a^q a^{2r} \cdot a^{p-q-r} b^p a^p b^p$

$= a^{q+2r+p-q-r} b^p a^p b^p$

$= a^{p+r} b^p a^p b^p$

∵ $r > 0$

$p + r > p$

Hence

$a^{p+r} b^p a^p b^p$ is not of the form $a^p b^p a^p b^p$

Here, $xy^2z \notin L$

i.e. According to proof by contradiction, L is not regular.