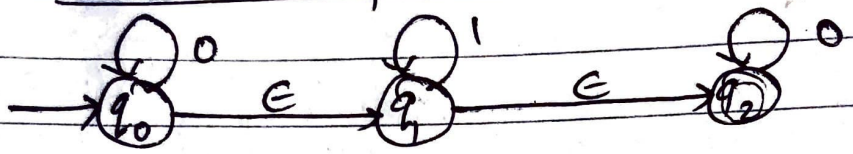
Date: Page:

Conversion of ϵ-NFA to DFA :-

eg



⇒ Here we first have to find the ϵ-closure for q_0, q_1, q_2.

ϵ-closure $\{q_0\} = \{q_0, q_1, q_2\}$: x

ϵ-closure $(q_1) = \{q_1, q_2\}$: y

ϵ-closure $(q_2) = \{q_2\}$: z

Now we have to find all the transitions for the new state x, y & z.

$\delta'(\{q_0, q_1, q_2\}, 0)$

$= \epsilon\text{-closure}(\delta(q_0, q_1, q_2), 0)$

$= \epsilon\text{-closure}(\delta(q_0, 0)) \cup \epsilon\text{-closure}(\delta(q_1, 0)) \cup \epsilon\text{-closure}(\delta(q_2, 0))$

$= \epsilon\text{-closure}(q_0) \cup \epsilon\text{-closure}\{\emptyset\} \cup \epsilon\text{-closure}(q_2)$

$= \{q_0, q_1, q_2\} \cup \emptyset \cup \{q_2\}$

$= \{q_0, q_1, q_2\}$: x

$\delta'(\{q_0, q_1, q_2\}, 1)$

$= \epsilon\text{-closure}(\hat{\delta}(q_0, q_1, q_2), 1)$

$= \epsilon\text{-closure}(\delta(q_0, 1)) \cup \epsilon\text{-closure}(\delta(q_1, 1)) \cup \epsilon\text{-closure}(\delta(q_2, 1))$

$= \epsilon\text{-closure}\{\emptyset\} \cup \epsilon\text{-closure}\{q_1\} \cup \epsilon\text{-closure}\{\emptyset\}$

$= \{q_1, q_2\}$: y

$\delta'(\{q_1, q_2\}, 0)$ Date: Page:

$= \epsilon\text{-closure}(\hat{\delta}(q_1, q_2), 0)$

$= \epsilon\text{-closure}(\hat{\delta}(q_1, 0)) \cup \epsilon\text{closure}(\hat{\delta}(q_2, 0))$

$= \epsilon\text{-closure}(\emptyset) \cup q_2 \epsilon\text{-closure}(q_2)$

$= \emptyset \cup \{q_2\}$

$= \{q_2\}$: Z

$\delta'(\{q_1, q_2\}, 1)$

$\epsilon\text{-closure}(\hat{\delta}(q_1, q_2), 1)$

$= \epsilon\text{-closure}(\delta(q_1, 1)) \cup \epsilon\text{-closure}(\delta(q_2, 1))$

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

$= \{q_1, q_2\}$: Y

$\delta'(\{q_2\}, 0)$

$= \epsilon\text{-closure}(\hat{\delta}(q_2), 0) = \epsilon\text{closure}(\delta(q_2, 0))$

$= \epsilon\text{closure}\{q_2\}$

$= \{q_2\}$: Z

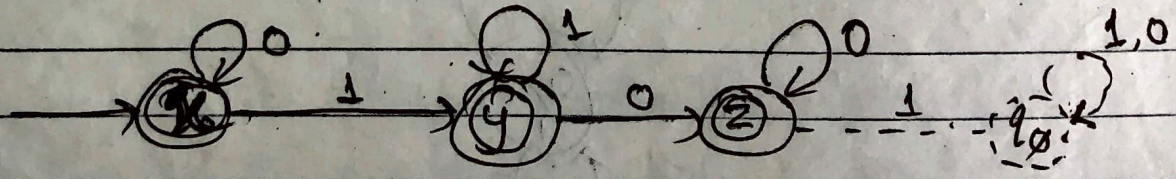
$\delta'(\{q_2\}, 1)$

$= \epsilon\text{-closure}(\hat{\delta}(q_2), 1) = \epsilon\text{closure}(\delta(q_2, 1))$

$= \epsilon\text{closure}\{\emptyset\}$

$= \{\emptyset\}$

∴ The equivalent DFA is:



Same in NFA & no need to make transition of (E) & dead state also

Date: Page:

Regular expression (R.E.) and Languages:-

aka. Language 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 analyzer generator such as LEX or FLEX.
    Lexical analyzer is a component of compiler that breaks the source program into logical unit called tokens.
  - Browsers and many other text formatting program

(pattern searching)

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 (∪/|/+):-

The union of two languages L and M

Date: Page:

denoted as L∪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

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

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

Date: Page:

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

$L^* = L^i = L^0 \cup L_1 \cup L_2 \cup \dots$

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

$(01+10)^*$ is different

01, 10, 0110, 0101, 1010, 1001

$(01+10)^1$ $(01+10)^2$

Date: Page:

Design of RE:-

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

Rule 1: (Zero or more)

$x^* = \{\epsilon, x, xx, xxx, \dots\}$

Suppose xy

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

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

Rule 2: (One or more)

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

$\Downarrow$

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)

$(x+y+z)^*$

$L_0 = \{\epsilon\}$

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

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

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

Date: Page:

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.

⇒ $y(x+y)^* xx$

Date: Page:

(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. 2 a's coming together :-

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

⇒ $(b+ab)^* + (b+ab)^*a$

$(b+ab)^*(\epsilon+a)$ ✓

OR

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

$(a+\epsilon)\ (b+ba)^*$ ✓

Date: Page:

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 $L_1 \cdot L_2 \cdot L_3 \ldots L_n$
- If ~~$L_1, L_2, \ldots L_n$ are~~ 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

Properties of regular sets.

Date: Page:

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:
   $t+(r+s) = (t+r)+s$
   $t \cdot (r \cdot s) = (t \cdot r) \cdot s$
3. Distributive:
   $r(s+t) = rs+rt$: left distribution
   $(s+t)r = sr+tr$: right distribution
4. Identity:
   $r \cdot \phi = \phi$
   $r+\phi = \phi+r = r$ 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
   $(r^*)^* = r^*$
   $r^+ = r \cdot r^* = r^* r$

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

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

$r\epsilon = r$

$\therefore r^+ = rr^*$

Date: Page:

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.

Date: Page:

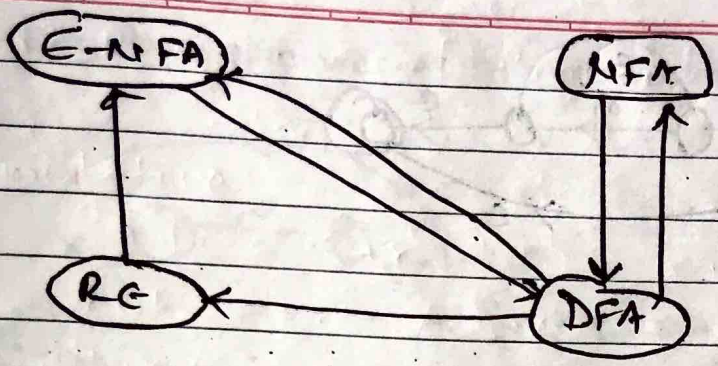
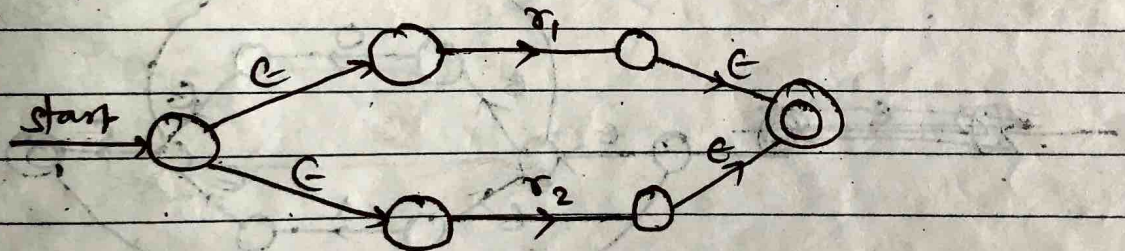


fig:- Plan for showing the equivalance of four different notations for regular language.

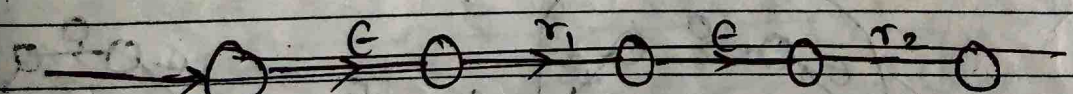
(Conversion from R.E. to ϵ-NFA)

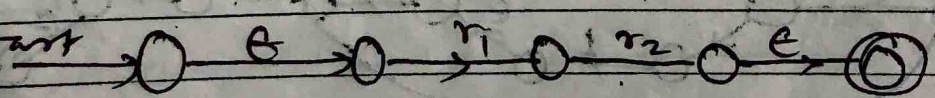
Rules:

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



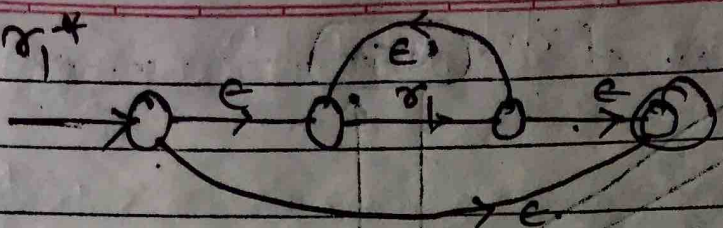
(ii) $r_1 \cdot r_2$ eg (11) ; $r_1 = 1$, $r_2 = 1$





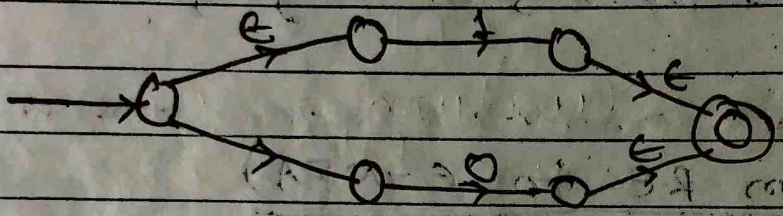
Date: Page:

(iii) r_1^*

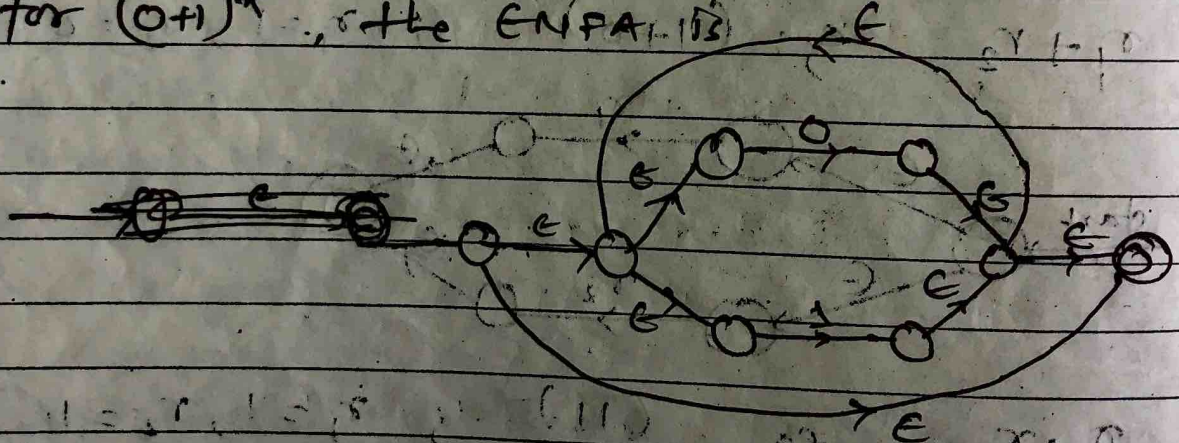


Examples :-

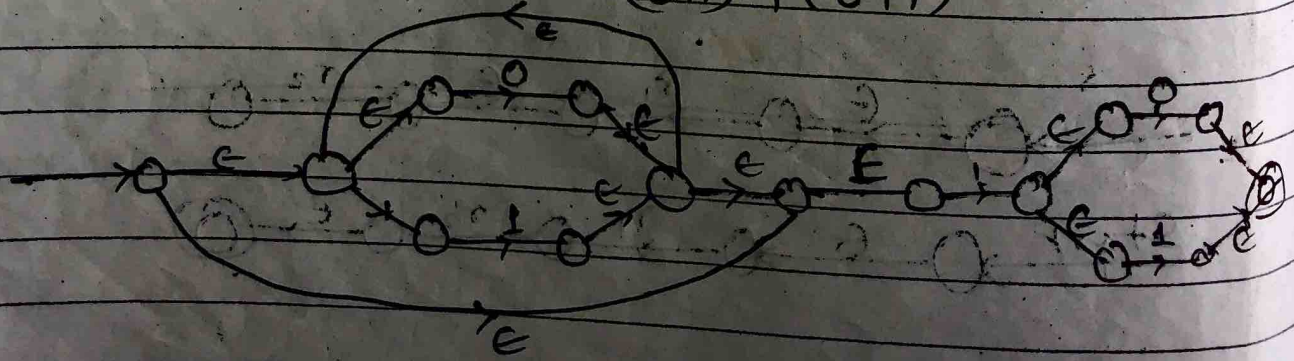
(1) For regular expression (1+0) the ε- NFA is



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

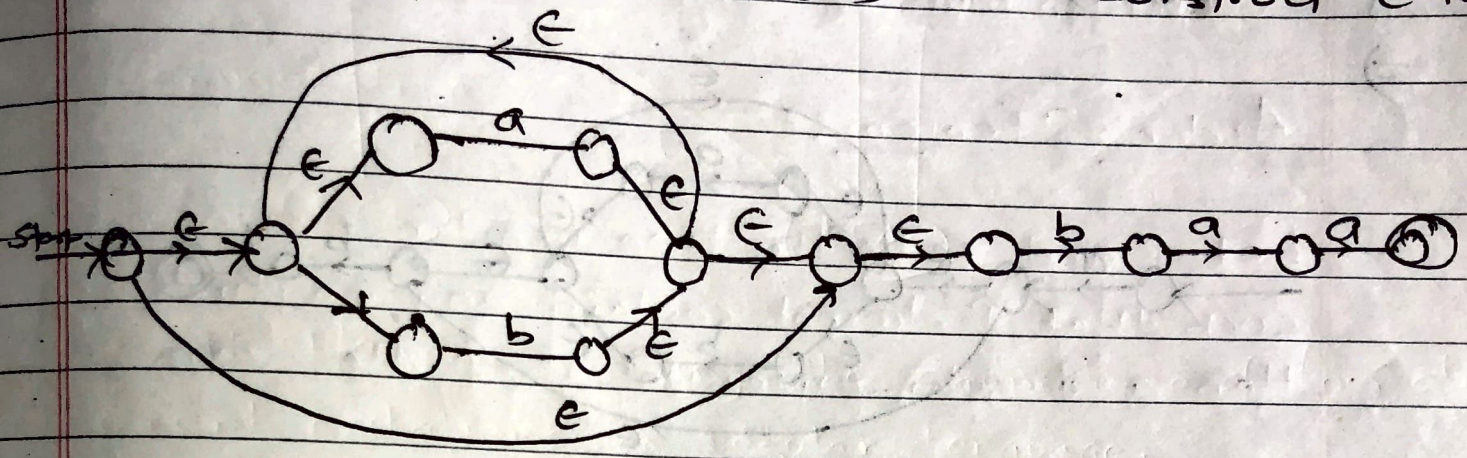


(3) for regular expression (0+1)*1(0+1)

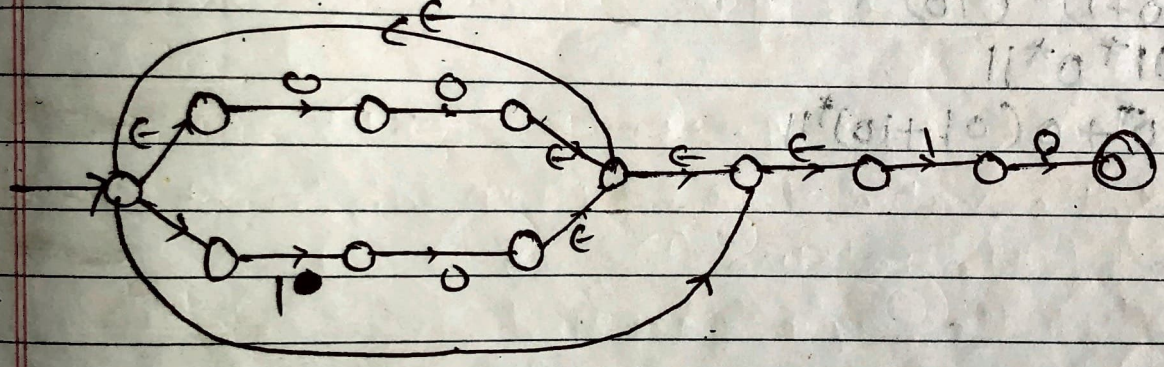


Date: Page:

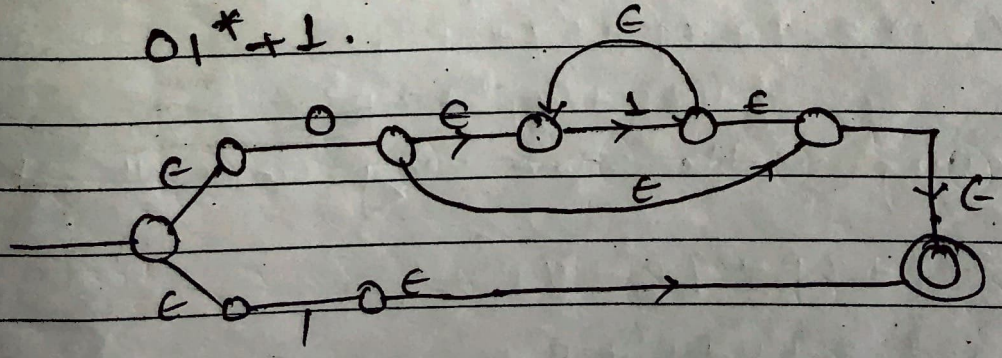
5. For regular expression $(a+b)^*baa$ construct ϵ-NFA.



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



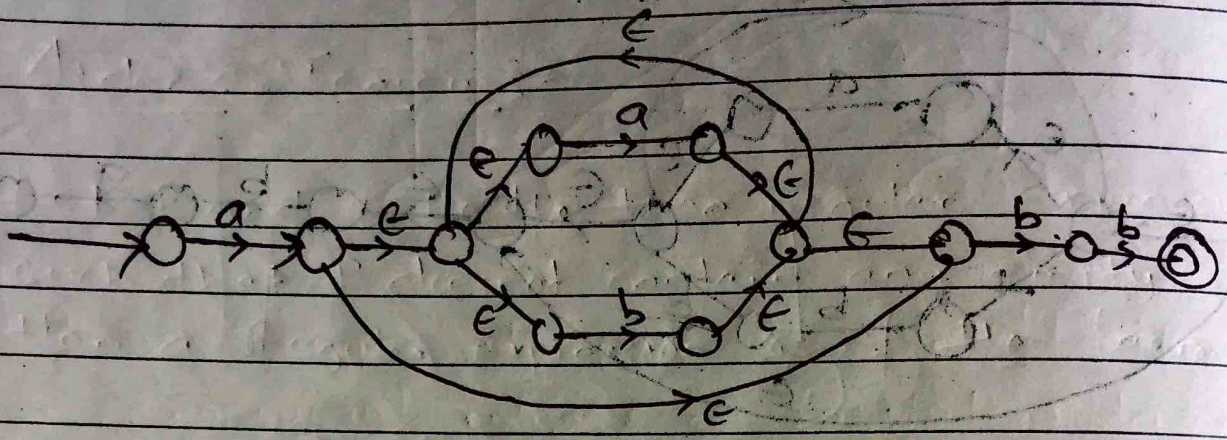
7. Construct ϵ-NFA for the regular expression 01^*+1.



Date: Page:

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

⇒



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

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

Q3 001^*0^*11

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

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.

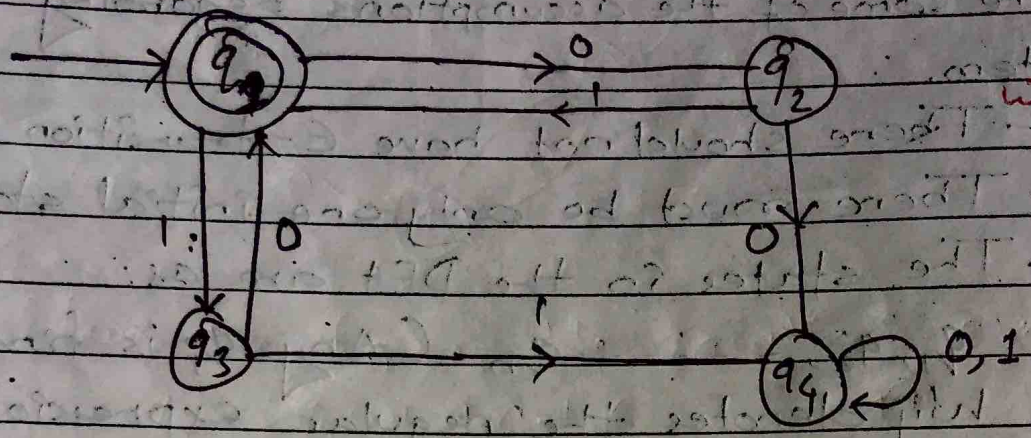
Thus we can write expression as,

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

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 a regular expression.



with respect to final state

Soln:- Let the equations are :-

$$q_1 = q_2 1 + q_3 0 + \epsilon \quad \text{---- (1)}$$

$$q_2 = q_1 0 \quad \text{---- (2)}$$

$$q_3 = q_1 1 \quad \text{---- (3)}$$

$$q_4 = q_2 0 + q_3 1 + q_4 0 + q_4 1 \quad \text{---- (4)}$$

Putting the value of q_2 and q_3 in eq^n (1)

$$q_1 = q_1 01 + q_1 10 + \epsilon$$

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

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

by applying Arden's theorem,

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

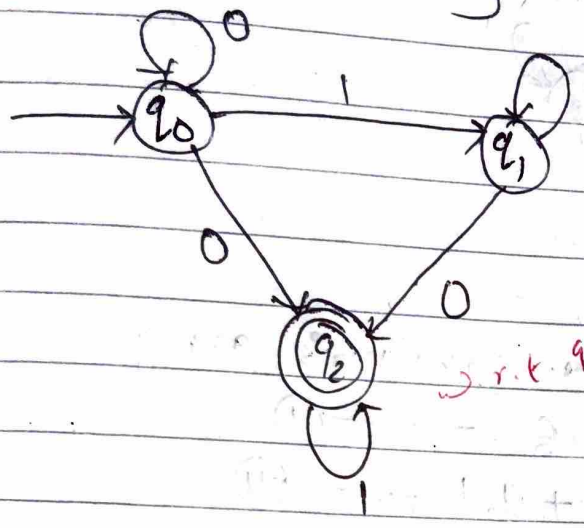
$$\therefore q_1 = (01 + 10)^*$$

which is the required regular expression

w.r.t q_1 final state

Date: Page:

Q) Convert the following NDFA into RE.



→ r.t. q_2

Soln:-

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 Arden's theorem in (I)

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

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

⇒ check if arden's theorem could be applied in any other eqns

from (II)

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

$$q_1 = \cancel{\quad}^* \; q_0 1 1^* = 0^* 1 1^* \quad \text{——— (V)}$$

from (III)

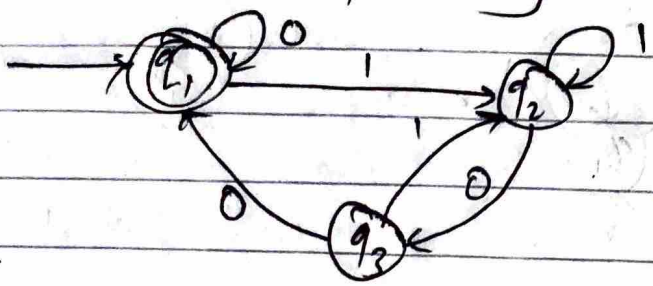
$$\underset{r}{q_2} = \underset{q}{q_0 0 + q_1 0} + \underset{r}{q_2} \underset{p}{1}$$

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

req soln of r.e.

Date: Page:

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 ardem's theorem can't be applied

from eqn (II) & (III)

$$q_2 = q_1 1 + q_2 1 + q_2 01$$
$$q_2 = q_1 1 + q_2 (1+01)$$
$$R = Q + RP \Rightarrow R = QP^*$$

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

from (I) & (III)

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

w.r.t q_1

from (IV) & (V)

$$q_1 = q_1 0 + q_1 1 (1+01)^* 00 + \epsilon$$
$$q_1 = q_1 (0 + 1(1+01)^* 00) + \epsilon$$
$$R = RP + Q \Rightarrow R = QP^*$$

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

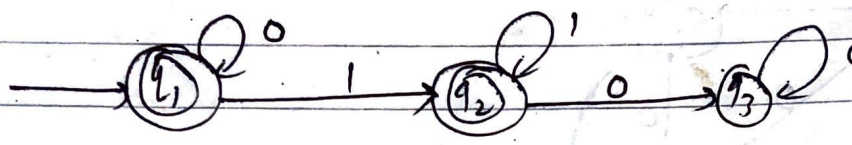
$$\therefore q_1 = (0 + 1(1+01)^* 00)^*$$

is the reqd R.E.

Case multiple final states.
find RE for all final states as before &
do union of those final states.

Date: Page:

Q) find the R.E for the transition diagram,



Soln:

$q_1 = q_1 \cdot 0 + \epsilon$ ——— (I)

$q_2 = q_1 \cdot 1 + q_2 \cdot 1$ ——— (II)

$q_3 = q_2 \cdot 0 + q_3 \cdot 0 + q_3 \cdot 1$ ——— (III)

Using Arden's theorem in eqn (I)

$q_1 = \epsilon(0)^*$ ——— (IV)

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

Again using (II) & (IV)

$q_2 = 0^* \cdot 1 + q_2 \cdot 1$

$q_2 = 0^* \cdot 1(1)^*$ ——— (V)

$\therefore q_2 = 0^* \cdot 11^*$ $\quad \because r^* = r^*$

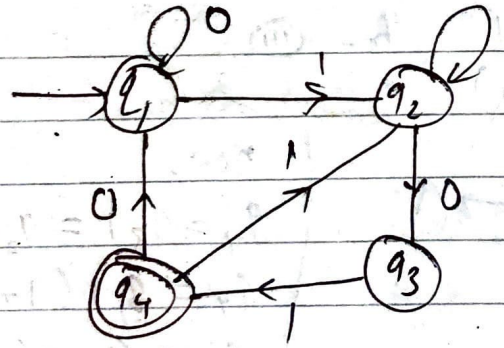
OR $q_2 = 0^* \cdot 1^+$

R = Union of both final states $= 0^* + 0^* 1 1^*$

$= 0^* (\epsilon + 11^*)$

q_1 q_2

Q)



convert to R.E.

$(0 + 1(1 + 011)^* 010)^* (1 + 011)^* 01$