classmate

Regular Expressions

There are three distinct ways of describing regular languages finitely and these are DFA, NFA and NFA with ϵ transition.

What are regular Expressions:- A regular expression R over an alphabet Σ is defined as follows:

1. ϕ, ϵ, and a are regular expressions, where a is a symbol in Σ ($a \in \Sigma$)
2. If R1 and R2 are regular expression, then so are

$(R1)$

$R1 + R2$ ← union

$R1R2$ ← concatination.

$(R1)^*$ ← closure.

* Every regular expression over Σ denotes a language over Σ

* If R is a regular expression, then L(R) is the language denoted.

* ϕ denote the empty language ϕ [$L(\phi) = \phi \subseteq \Sigma^*$]

  ϵ denote the $\{\epsilon\}$ [$L(\epsilon) = \{\epsilon\}$]

  a denote $\{a\}$ i.e $L(a) = \{a\}$

  $(R1)$ denote $L(R1)$

  $R1+R2$ denotes $L(R1) \cup L(R2)$

  ~~R1+R~~ $R1R2$ denote $L(R1)L(R2) = \{xy \mid x, y \in \Sigma^*, x \in L(R1), y \in L(R2)\}$

$$L^* = \bigcup_{i \geq 0} L^i$$

Page No

$L^* \rightarrow L^0 = \{\epsilon\}, \Rightarrow L^1 = L \rightarrow L^{n+1} = LL^n \rightarrow L^* = \bigcup_{i \geq 0} L^i$

$(RI)^*$ denotes $(L(RI))^*$

ex if $\Sigma = \{0,1\}$ then

$\underline{0} \rightarrow \{0\}$

$\underline{1} \rightarrow \{1\}$

$\underline{0+1} \rightarrow \{0,1\}$

$(\underline{0+1})^* \rightarrow \{0,1\}^*$

└ UNION.

It is possible to have several different regular expressions denoting the same language.

ex - $((0)^*(1)^*)^* = (\underline{0+1})^*$ → any binary string.

* (closure) has the highest precedence followed by concatination followed by union (+).

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

1. $0^*10^* = \{x \in \{0,1\}^* \mid x$ has exactly one $1\}$
2. $(0+1)^* 1 (0+1)^* = \{x \in \{0,1\}^* \mid x$ has at least
3. $(0+1)^* 001 (0+1)^* = \{x \in \{0,1\}^* \mid x$ has substring $001\}$

$\overline{x}\ 001\ \overline{y}$

4. $((0+1)(0+1))^* = \{x \in \{0,1\}^* \mid x$ has even symbol$\}$

↳ $\{\{0,1\}\{0,1\}\}^* \rightarrow \{00, 01, 10, 11\}^*$

5. $0(0+1)^*0 + 1(0+1)^*1 + 0 + 1 \rightarrow \{x \in \{0,1\}^* \mid$ x has same first & last bit$\}$

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

1. $(a+b)^*(c(a+b)^*c(a+b)^*)^* = \{x \in \{a,b,c\}^* \mid x$ contains even no of C$\}$

2. $c^*(a+bc^*)^* = \{x \in \{a,b,c\}^* \mid x$ does not contain ac as a substring $\}$.

→ Regular expressions denote precisely the class of regular languages.

proof: 1. Given any regular expression R, L(R) is a regular language.

2. Given any regular language say L, there is a regular expression, R s.t L(R) = L.

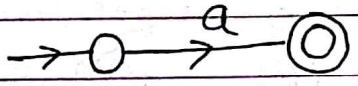
Proof for 1.

NFAs used in the proof will have exactly one final state and there will be no transition out of that state. (final state)

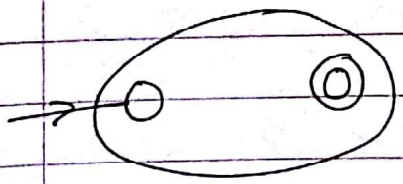
Σ is the alphabet.

ϕ  ϕ.

ϵ  $\{\epsilon\}$

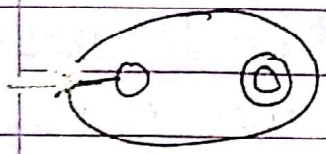
a 

If R1 and R2 are regular expressions then so is R1 + R2



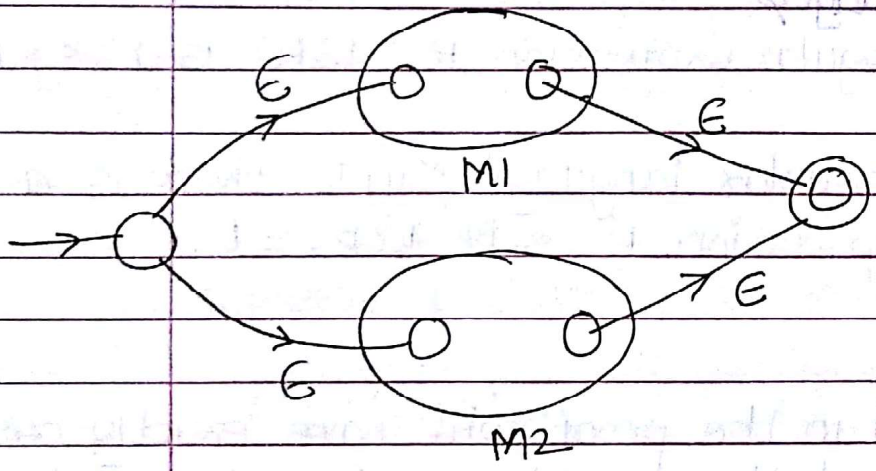
M1

∴ L(M1) = L(R1)

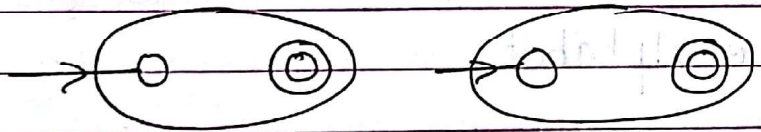


M2

L(M2) = L(R2)

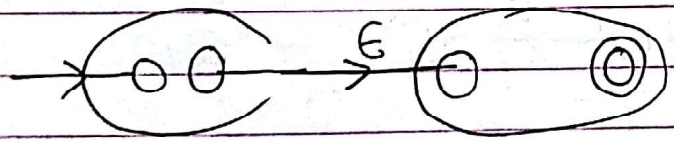


for R1R2 case

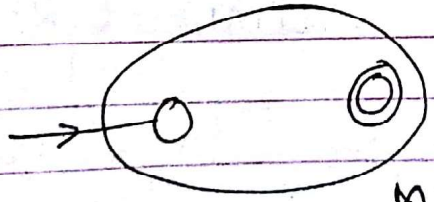


NFA for R1 NFA for R2

NFA for R1R2



suppose I have regular expression R then correspo NFA is as follow.

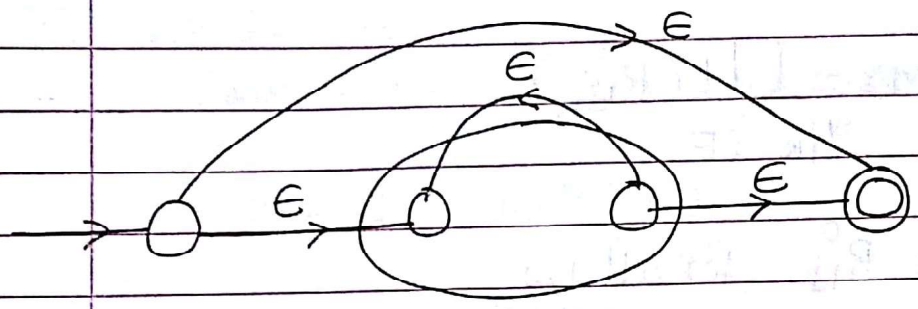


M for L(R)

Page No

then $(R)^*$

NFA for $L(R^*)$



suppose $x \in (L(R))^*$ such that

$x = x_1, x_2, \ldots x_k$

$\therefore x_i \in L(R)$

Proof for case 2 -

Given a DFA M we can construct a regular expression R s.t the Language accepted by DFA $L(M) = L(R)$ (Language represented by R)

suppose $M = (Q, \Sigma, \delta, q_1, F)$

where $Q = \{q_1, \ldots, q_n\}$

R_{ij}^{k} is a regular expression s.t

$$L(R_{ij}^{k}) = \{ x \in \Sigma^* \mid (1)\ \hat{\delta}(q_i, x) = q_j \text{ and further, The machine M does not go to any state whose number is greater than k on going from } q_i \text{ to } q_j \text{ on } x \}$$ (i,j value can be greater than k)

$$L(M) = R_{1j_1}^{n} + R_{1j_2}^{n} + \cdots + R_{j_m}^{n}$$

where $\{q_{j_1}, q_{j_2}, \ldots q_{j_m}\} = F$

Page No

$$L(R^n_{1jf}) = \{x \in \Sigma^* \mid \hat{\delta}(q_1, x) = q_{jf}\}$$
where $q_{jf} \in F$

$$\therefore L(M) = \bigcup_{q_{jk} \in F} L(R^n_{1jk})$$

Define R^0_{ij} for all i, j.

$$L(R^0_{ij}) = \{a \mid \delta(q_i, a) = q_j\} \quad \text{if } i \neq j.$$
($a \in \Sigma$)

OR

$$= \{a \in \Sigma \mid \delta(q_i, a) = q_j\} \cup \{\epsilon\} \quad \text{if } i = j$$

Algebraic Laws for Regular Expressions :-

1] $L+M = M+L$
Where L & M are any regular expressions.

2] $L+(M+N) = (L+M)+N$.

3] $L(MN) = (LM)N$

4] $\phi + L = L + \phi = L$

5] $\epsilon L = L\epsilon = L$

6] $\phi L = L\phi = \phi$

7] $L(M+N) = LM + LN$.

8] $(M+N)L = ML + NL$

9] $L+L = L$

10] $(L^*)^* = L^*$

11] $\phi^* = \epsilon$

12] $\epsilon^* = \epsilon$

closure properties of regular languages:-

The class of regular languages is closed under

(a) Union.

Proof- Let L_1, L_2 be regular

To show that $L_1 \cup L_2$ is also regular.

Let R_1 & R_2 be two regular expression s.t $L(R_1) = L_1$ and $L(R_2) = L_2$

clearly then, The regular expression $R_1 + R_2$ by definition denote $L_1 \cup L_2$

(b) Intersection

(c) Complimentation $\bar{L} = \Sigma^* - L_1$

(d) Set difference

$$L_1 - L_2 = \{x \mid x \in L_1 \text{ and } x \notin L_2\}$$

$$= L_1 \cap \bar{L}_2$$

(e) Concatination -

suppose L_1 & L_2 are regular then there are regular expression R_1 and R_2 such that $L(R_1) = L_1$ and $L(R_2) = L_2$

$\therefore L(R_1R_2) = L(R_1)L(R_2)$ by definition of regular expression for concatination.

(f) kleene closure :-

(g) Reversal

Let A be the regular expression then A^R will accept $L(A))^R$.

Base case - ϵ a ϕ

$\epsilon^R \equiv \epsilon$

$a^R = a$

$\phi^R = \phi$

Inductive step -

If A1 and A2 are regular expression then

A1+A2 , A1A2 , A1*

A_1^R , A_2^R

$\therefore (A1+A2)^R = A_1^R + A_2^R$

$(A1A2)^R = A_2^R A_1^R$

$(A^*)^R = (A_1^R)^*$

so Inductively, we assume

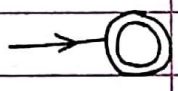
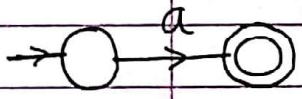
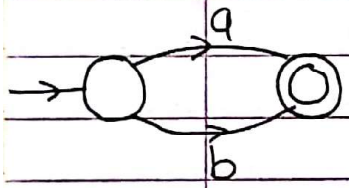
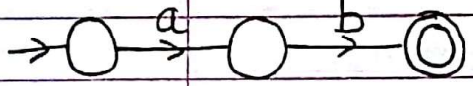
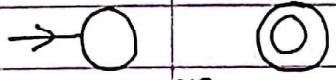
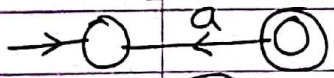
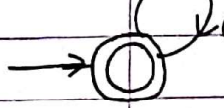
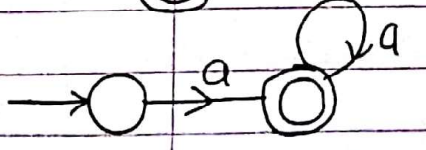
$L(A_1R) = (L(A_1))^R$

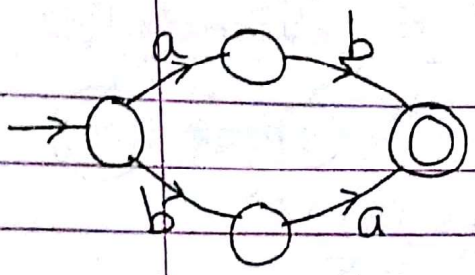
$L(A_2R) = (L(A_2))^R$.

Regular language can also be described in a compact form using a set of operators. These operators are-

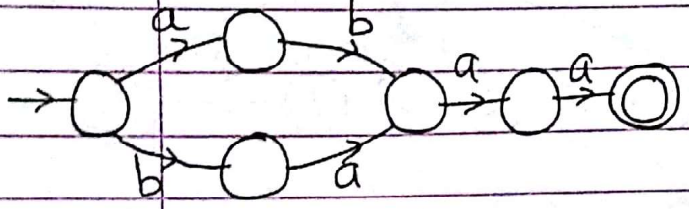
1. +, union operator.
2. ·, concatenation.
3. *, star or closure operator.

An expression written using the set of operators (+, ·, *) and describing a regular language is known as regular expression.

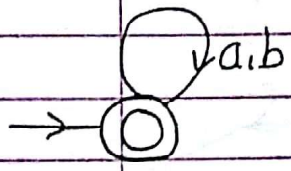
Automata	Language	RE.
	{ϵ}	RE = ϵ
	{a}	RE = a
	{a, b}	RE = a+b
	{ab}	RE = a·b or ab.
	ϕ	RE = ϕ
OR  	{ϵ, a, aa, aaa ....}	RE = a*
	{a, aa, aaa, ...}	RE = aa* OR a+



{ab, ba} RE = ab+ba



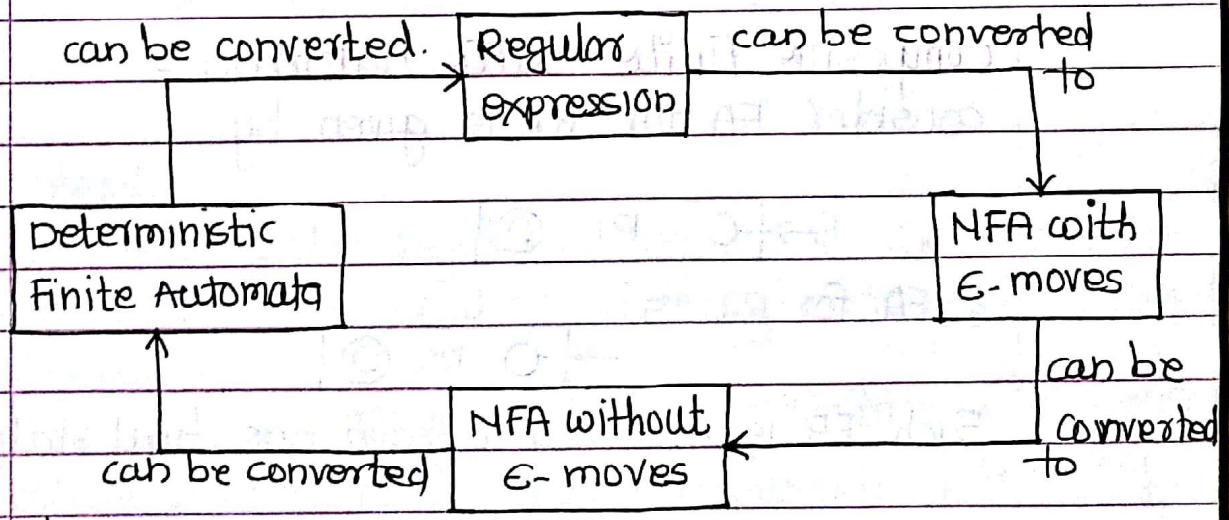
{abaa, baaa} RE = (ab+ba)aa.



{ϵ, a, b, aa, bb, ab, bba}...} RE = (a+b)*

Finite Automata and Regular Expressions:-

There is a close relationship between a finite automata and the regular expression



Technical 3-10 old

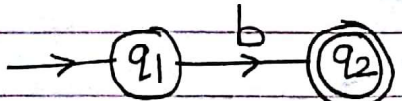
Construction of NFA From Regular Expression:-

ex

R.E = b+ba*

r = b+ba*

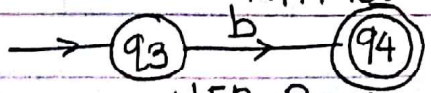
r1 = b



NFA for r1.

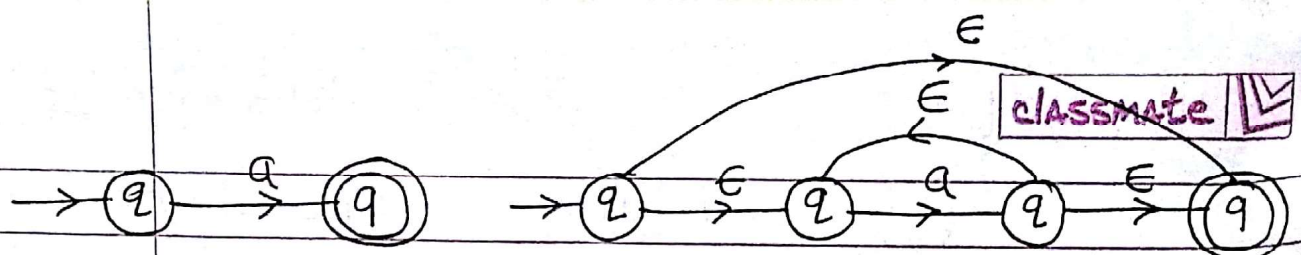
r2 = ba*

r3 = b



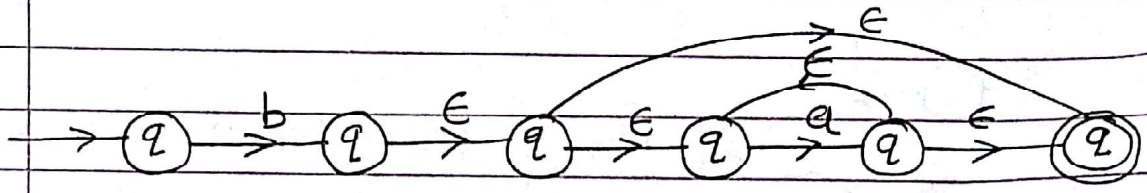
NFA for r3.

r4 = a* r5 = a

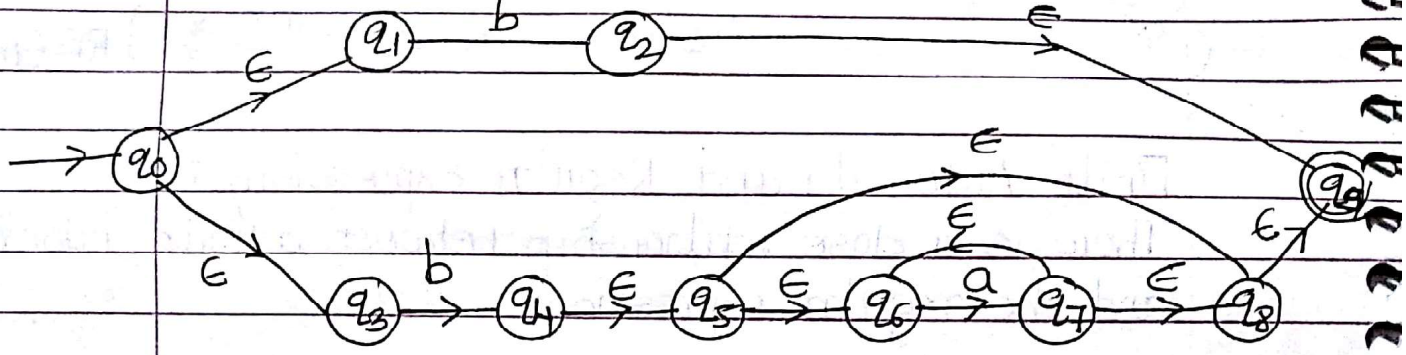


NFA for r5.

NFA for r4 = r5*



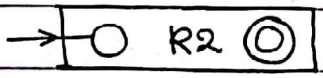
NFA for r2 = r3r4.



Composite Finite state Automata :-

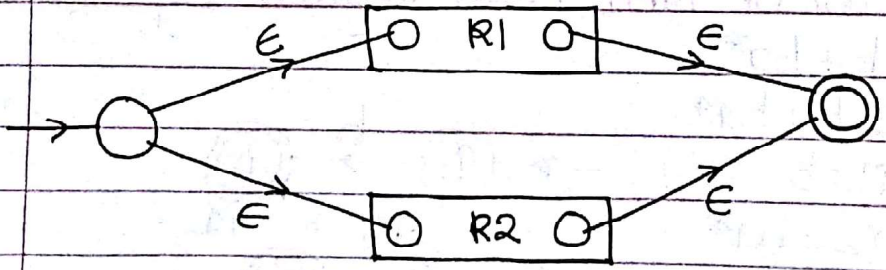
consider FA for R1 is given by.

& FA for R2 is



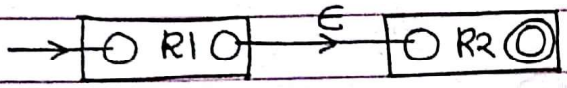
Each FA is assumed to have one final state.

1. FA for R1+R2 is given by

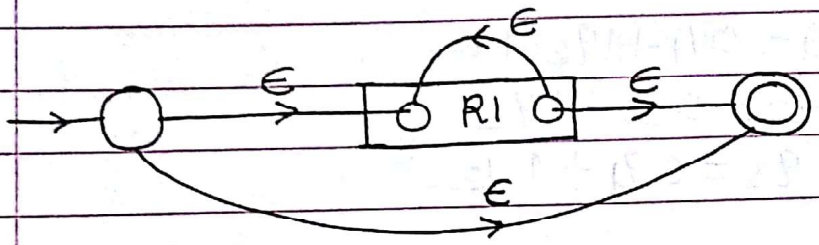


Page No

2. FA for R1 R2 is given by.



3. FA for R1* is given by.



Conversion From finite Automata to Regular Expression:

Arden's Theorem :- Let A and B be two regular expression over the alphabet Σ If A does not contain ϵ then $X = AX + B$ has a unique solution that is $X = A^*B$.

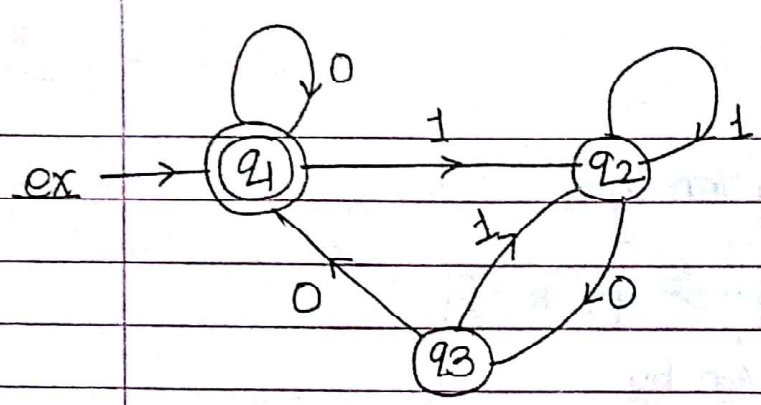
steps

1. For getting the regular expression for the automata we first create equations of the given form for all the states.

$$q_1 = w_1 q_1 + w_2 q_2 + \cdots + w_n q_n + \epsilon \text{ (only when } q_1 \text{ is initial state)}$$

where w represent the regular expression representing the set of labels of edges

2. Then we solve these equations to get the required solution.



$q_1 = 0q_1 + 1q_2 + \epsilon$

$q_2 = 0q_3 + 1q_2$

$q_3 = 0q_1 + 1q_2$

$q_2 = 0q_3 + 1q_2$

$= 1q_2 + 0q_3$

$\boxed{q_2 = 1^*0q_3}$ ..... using Arden's Theorem.

$q_3 = 0q_1 + 1q_2$

$= 0q_1 + 11^*0q_3$

$q_3 = 11^*0q_3 + 0q_1$

$\boxed{q_3 = (11^*0)^*0q_1}$ .... using Arden's theorem

$q_1 = 0q_1 + 1q_2 + \epsilon$.

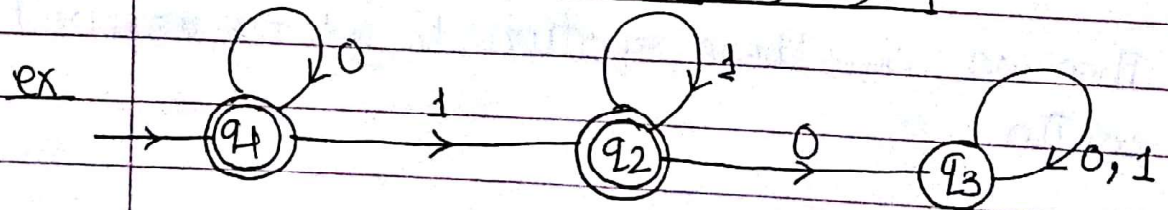
$= 0q_1 + 11^*0q_3 + \epsilon$

$= 0q_1 + 11^*0(11^*0)^*0q_1 + \epsilon$

$= (0 + 11^*0(11^*0)^*0)q_1 + \epsilon$

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

$\boxed{q_1 = (0 + 11^*0(11^*0)^*0)^*}$



$q_1 = 0q_1 + 1q_2 + \epsilon$

$q_2 = 0q_3 + 1q_2 + \epsilon$

$q_3 = 0q_3 + 1q_3$

$= (0+1)q_3 + \phi$

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

$\boxed{q_3 = \phi}$ ... using Algebric Law for RE.

$q_2 = 0q_1 + 1q_2 + \epsilon$

$q_2 = 1q_2 + \epsilon$

$= 1^*\epsilon$

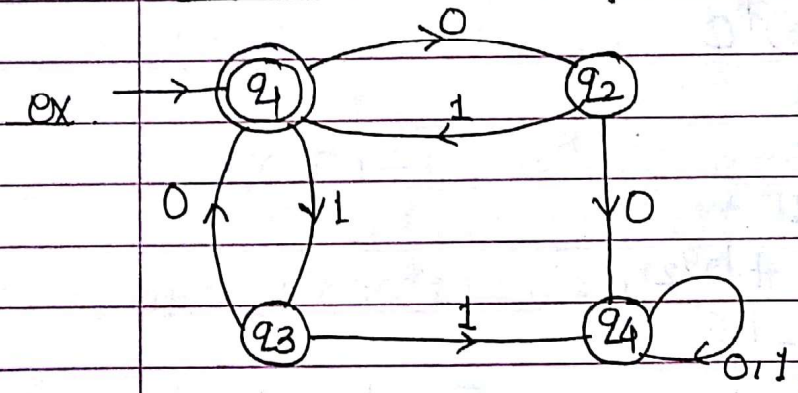
$\boxed{q_2 = 1^*}$

$q_1 = 0q_1 + 11^* + \epsilon$

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

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

$\boxed{q_1 = 0^*11^* + 0^*}$



$q_1 = 0q_2 + 1q_3 + \epsilon$

$q_3 = 0q_1 + 1q_4$

$q_2 = 1q_4$

$q_3 = 0q_1$

$q_2 = 0q_4 + 1q_1$

$q_4 = 0q_4 + 1q_4$

$= (0+1)q_4 + \phi$

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

$\boxed{q_4 = \phi}$

$q_1 = 0q_2 + 1q_3 + \epsilon$

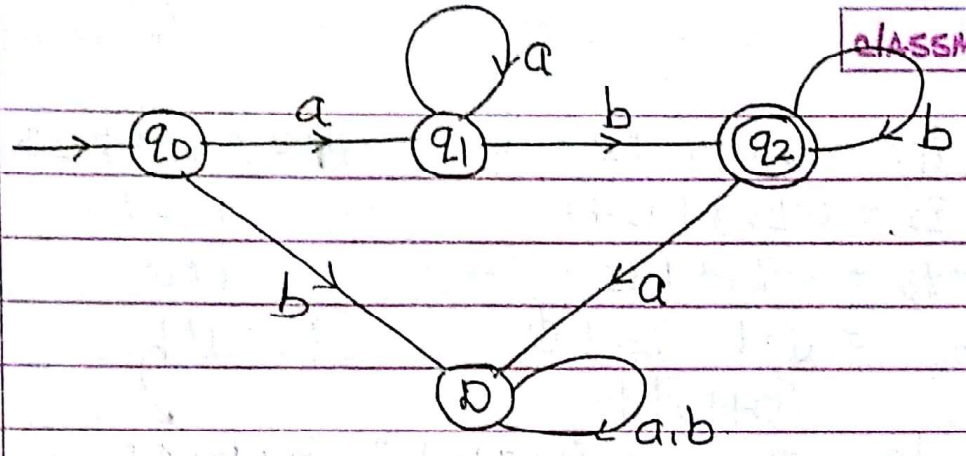
$= 01q_4 + 10q_4 + \epsilon$

$= (01+10)q_4 + \epsilon$

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

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

ex



$q_0 = aq_1 + bD$

$q_1 = aq_1 + bq_2$

$q_2 = aD + bq_2 + \epsilon$

$D = aD + bD$

$= (a+b)D + \phi$ [using Arden's theorem]

$= (a+b)^*\phi$

$\boxed{D = \phi}$

$\therefore q_0 = aq_1$

$q_1 = aq_1 + bq_2$

$q_2 = bq_2 + \epsilon$.

$= b^*\epsilon$ [Arden's law]

$\boxed{q_2 = b^*}$

$q_1 = aq_1 + bq_2$

$= aq_1 + bb^*$

$\boxed{q_1 = a^*bb^*}$ [Arden's law]

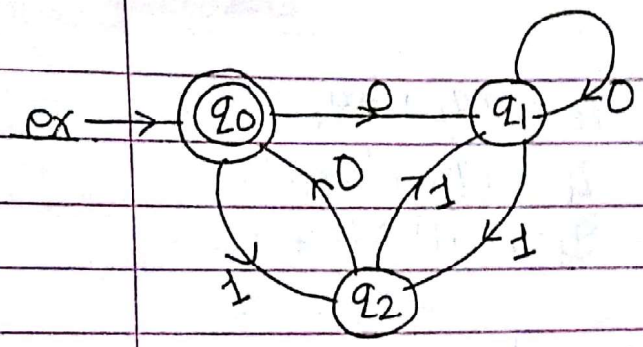
$\therefore \boxed{q_0 = aa^*bb^*}$

each q_i is a set of string that take you from q_i to final state.

$\therefore L(aa^*bb^*)$ will take machine from q_0 to q_f

classmate

ex



$q_0 = 0q_1 + 1q_2 + \epsilon$

$\rightarrow q_1 = 0q_1 + 1q_2$

$q_2 = 0q_0 + 1q_1$

$q_1 = 0q_1 + 1q_2$

$q_1 = 0^*1q_2$

put value of q_1 in q_2 eqn.

$q_2 = 0q_0 + 10^*1q_2$

$= 10^*1q_2 + 0q_0$

$q_2 = (10^*1)^*0q_0$

put q_1 value in q_0 eqn.

$q_0 = 0q_1 + 1q_2 + \epsilon$

$q_0 = 00^*1q_2 + 1(10^*1)^*0q_0 + \epsilon$

$= 00^*1q_2 + 1q_2 + \epsilon$

$= (00^*1 + 1)q_2 + \epsilon$

put q_2 value in above eqn

$q_0 = (00^*1 + 1)(10^*1)^*0q_0 + \epsilon$

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

$q_0 = [(00^*1 + 1)(10^*1)^*0]^*$

ex

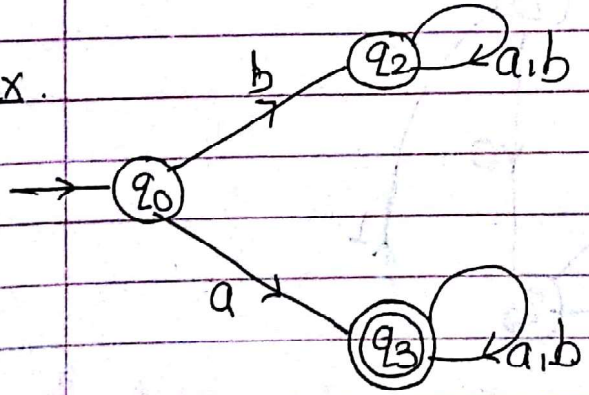
$q_0 = 0q_0 + 1q_0 + \epsilon$

$= (0+1)q_0 + \epsilon$

$= (0+1)^*\epsilon$

$q_0 = (0+1)^*$

ex.



$q_0 = aq_3 + bq_2$

$q_2 = aq_2 + bq_2$

$q_3 = aq_3 + bq_3 + \epsilon$

$q_2 = aq_2 + bq_2$

$= (a+b)q_2 + \phi$

$= (a+b)^*\phi$

$q_2 = \phi$

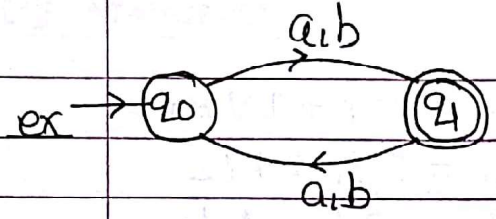
$q_3 = (a+b)q_3 + \epsilon$

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

$q_3 = (a+b)^*$

$q_0 = a(a+b)^*$

ex



$q_0 = aq_1 + bq_1$

$q_1 = aq_0 + bq_0 + \epsilon$

$q_1 = (a+b)q_0 + \epsilon$

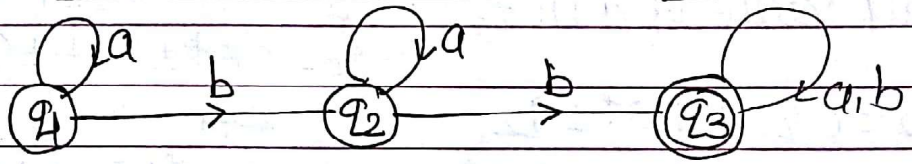
$q_0 = (a+b)q_1$

$= (a+b)(a+b)q_0 + (a+b)\epsilon$

$= [(a+b)(a+b)]^*(a+b)\epsilon$

$\boxed{q_0 = [(a+b)(a+b)]^*(a+b)}$

ex.



$q_1 = aq_1 + bq_2$ $\qquad q_2 = aq_2 + bq_3$

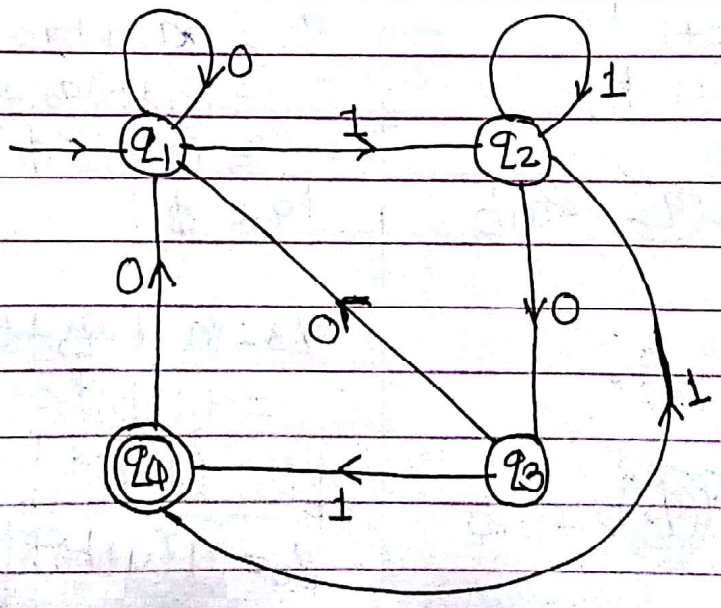
$q_3 = \epsilon$

$q_2 = aq_2 + b\epsilon = a^*b$

$q_1 = aq_1 + ba^*b$

$\boxed{q_1 = a^*ba^*b}(a+b)^*$

ex State Arden's theorem and use it to construct a regular expression corresponding to following automata. [Dec 10. Marks 10]



Page No

$q_1 = 0q_1 + 1q_2$ $\quad\quad$ $q_3 = 0q_1 + 1q_4$.

$q_2 = 1q_2 + 0q_3$ $\quad\quad$ $q_4 = 0q_1 + 1q_2 + \epsilon$

$q_1 = 0^*1q_2$ } Arden's Theorem.

$q_2 = 1^*0q_3$ }

$q_3 = 0q_1 + 1q_4$.

$= 00^*1q_2 + 1q_4$. — put $q_1 = 0^*1q_2$

$= 00^*11^*0q_3 + 1q_4$ — put $q_2 = 1^*0q_3$

$\boxed{q_3 = (00^*11^*0)^*1q_4.}$

$q_4 = 0q_1 + 1q_2 + \epsilon$

$= 00^*1q_2 + 1q_2 + \epsilon$ put $q_1 = 0^*1q_2$

$= 00^*11^*0q_3 + 11^*0q_3 + \epsilon$ put $q_2 = 1^*0q_3$.

$= (00^*11^*0 + 11^*0)q_3 + \epsilon$

$= (00^*11^*0 + 11^*0)((00^*11^*0)^*1q_4) + \epsilon$

$= [(00^*11^*0 + 11^*0)(00^*11^*0)^*1]^*\epsilon$

$\boxed{q_4 = [(00^*11^*0 + 11^*0)(00^*11^*0)^*1]^*}$

Page No

Decision Properties of Regular Languages:

Given a regular language L. What can we tell about L? We present a language by giving one of the finite representations for it that we have developed a DFA, an NFA, an ϵ-NFA or a regular expression.

1. Is a particular string w belongs to some language L? [Membership property]
2. Is the language described empty? [Emptiness Property]
3. Do the two descriptions of a language represent the same language? or whether two languages are equivalent [Equivalente property]

Examples on pumping lemma:

ex → prove $L = \{a^p \mid p \text{ is prime}\}$ is not regular.

Let us assume L is a regular & p is a prime number

prime numbers — 2, 3, 5, 7, ...

aa
aaa
aaaaa....

Now consider $x = \underbrace{aa}_{u}\underbrace{aa}_{v}\underbrace{a}_{w}$. $|x|$ is prime say p.

$i = 2$

$\underbrace{a}_{u}\underbrace{aa}_{v}\underbrace{aa}_{v}\underbrace{a}_{w}$.

$p < p+1$

but $p+1$ is not a prime number. Hence whatever

We have assumed becomes contradictory. hence L is not regular.

ex. Show that $L=\{0^n 1^{n+1} \mid n>0$ is not regular.

→ Let us assume L is regular. consider K is pumping lemma constant. consider string $x=0^K 1^{K+1}$.

$$\underbrace{00\ldots0}_{K}\ \underbrace{1\ldots1}_{K+1}$$

According to pumping lemma, there exist uvw such that $|uv|\leq K$ $|v|>0$ it means we string consist of only zero's. $|v|$ is in between 1 to K. consider $|v|=K$.

Now consider-

$uvvw$ $\quad i=2$.

$$\underbrace{000\ldots0}_{2K}\ \underbrace{11\ldots1}_{K+1}$$

∴ We found no. of zero's are greater than No. of 1's

∴ $uvvw \notin L$.

∴ so we have a contradiction, so what ever assumption we had made is wrong.

∴ L is not regular.

ex. Show that $L=\{ww \mid w\in\{0,1\}^*\}$ is not regular

→ Let us assume L is regular. consider K is pumping lemma constant. K represent length of w.

consider

$$x=\underbrace{0101}_{w}\ \underbrace{0101}_{w}$$

K K

Page No

$x = \underbrace{0101}_{uv}\ \underbrace{0101}_{w}$

According to pumping lemma there exist uvw.

$\therefore |uv| \leq k,\ |v| > 0$

$|v| = 1$.

$i = 2$.

uv^2w

$\underbrace{01101}_{k+1}\ \underbrace{0101}_{k}$

$\therefore uv^2w \notin L$.

$\therefore$ we have a contradiction. therefore our assumption is wrong.

hence L is not regular.

ex prove $L = \{a^m b^n \mid m \neq n,\ m \text{ and } n \text{ are +ve integers}\}$ is not regular.

→ Let us assume L is regular. according to defn of L.

$abb \in L$

$aaabbbb \in L$.

Now consider $x = aaabbbb$

according to pumping lemma there exist uvw such that $uvw = x$, $|uv| \leq k$, $|v| > 0$ also

$\therefore \underbrace{a}_{u}\ \underbrace{a\ a}_{v}\ \underbrace{a\ b\ b\ b}_{w}$

for all $i \geq 0$ $uv^iw \in L$

Now consider string uv^2w

$aa\underbrace{ab}_{v}\underbrace{ab}_{v}\underbrace{bbb}_{w} \notin L$

also consider $x = \underbrace{aa}_{u}\underbrace{a}_{v}\underbrace{bbbb}_{w}$

Page No

for i=2, uv^2w i.e

$aaaabbbb \notin L$.

$a^4b^4 \notin L$.

We have a contradiction, therfore our assumption was wrong.

hence L is not regular.

ex. prove that $L=\{a^nb^nc^n \mid n \geq 1\}$ is not regular.

→ Let us assume L is regular, K is pumping lemma constant, consider string $x = a^k b^k c^k$

$$x = \underbrace{\overbrace{a\ldots a}^{k}}_{u}\ \underbrace{\overbrace{b\ldots b}^{k}}_{v}\ \underbrace{\overbrace{c\ldots c}^{k}}_{w}.$$

From pumping lemma ↑

consider uv^0w.

$$\underbrace{a\ldots a}_{k}\ \underbrace{b\ldots b}_{<2k}\ \underbrace{c\ldots c}_{k}.$$

$\therefore uv^0w \notin L$

We have a contradiction, therefore our assumption was wrong.

hence L is not regular.

ex. Prove that $L=\{a^nb^{2n} \mid n>0\}$ is not regular.

→ Let us assume L is regular. K is lemma constant such that $x = a^k b^{2k}$.

$$\underbrace{a\ldots a}_{k}\ \underbrace{b\ldots b}_{2k}.$$

From pumping lemma,

$|uv| \leq k, |v| \geq 0$

$\therefore |v| = k.$

$$\underbrace{a\ldots a}_{uv}\ \underbrace{b\ldots b}_{w}$$

Page No

$i=0$. uw

$b \ldots bb \ldots b \notin L$

$i=2$ $uvvw$

$\underbrace{a \ldots a}_{2k}\,\underbrace{a \ldots a}_{2k}\,\underbrace{b \ldots b}_{2k}\,\underbrace{b \ldots b}_{2k}$

$\therefore uvvw \notin L$.

$\therefore$ We have a contradiction, $\therefore$ our assumption is wrong

hence L is not regular.

ex. Prove that $L = \{a^n b^{2m} \mid 0 < n < m\}$ is not regular.

→ Let us assume L is regular.

$n=2,\ m=3$.

$\underbrace{aa}_{u}\underbrace{bbb}_{v}\underbrace{bbb}_{w}$

$i=2$. $\therefore uvvw$.

$aa\ bbb\ bbb\ bbb$.

a^2b^9

$\therefore 9 \neq 2m$

$\therefore uvvw \notin L$

$i=0$.

uw

$aa\ bbb$

a^2b^3

$\therefore 3 \neq 2m$.

$\therefore uw \notin L$.

$\therefore$ We have a contradiction, therefore our assumption is wrong

hence L is not regular.

ex A palindrome is a string that equals its own reverse, such as 0110 or 1011101. use the pumping lemma to show that the set of palindromes is not a regular language.

→ Let us assume L is regular.

$L = \{w \in \{a,b\} \mid w = w^R\}$.

$x = \underbrace{ab}_{u}\ \underbrace{ab}_{v}\ \underbrace{ba\ ba}_{w}$. $x^R = ab\ ab\ ba\ ba$

$\therefore x \in L$.

From lemma

For $i=0$:

uw

$ab\ ba\ ba \xrightarrow{R} ab\ ab\ ba$.

$uw \notin L$.

For $i=2$.

$uvvw$.

$ab\ ab\ ab\ ba\ ba \xrightarrow{R} ab\ ab\ ba\ ba\ ab$.

$uvvw \notin L$.

We have a contradition, therefore our assumption was wrong.

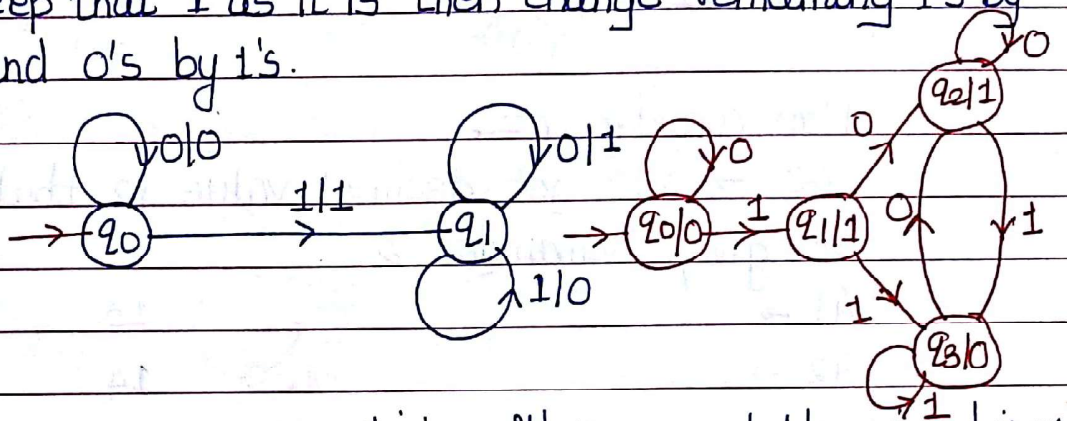
hence L is not regular.

Page No

Finite Automata continue.....

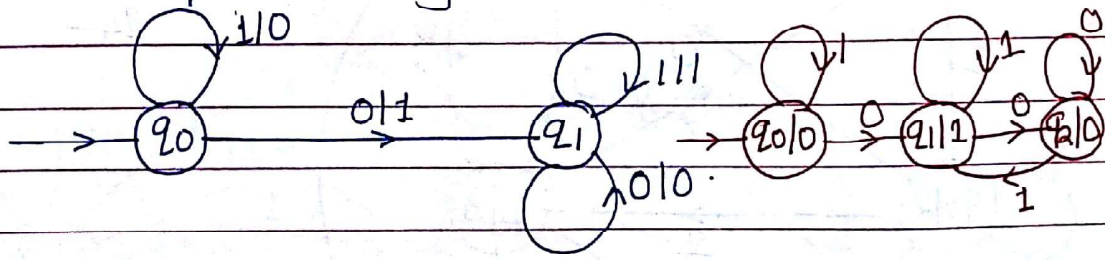
ex. Design a mealy mic to find 2's complement of a given bin number.

→ For designing 2's complement of a binary number we assume that input is read from LSB to MSB or we will give input to mealy mic from right to left. We will keep the binary number as it is until we read first 1. Keep that 1 as it is then change remaining 1's by 0's and 0's by 1's.



ex. Design a Moore mic which will increment the given binary number by 1.

→ We will read the binary number from LSB one bit at a time. We will replace each 1 by 0 until we get first 0. Once we get first 0 we will replace it by 1 & then keep remaining bits as it is.



ex. Give Moore and Mealy machine for $\Sigma = \{0,1,2\}$, print the residue modulo 5 of input treated as a ternary number.

→ The ternary number is made up of 0, 1 and 2

The interpretation of ternary number n can be

i) If we write 0 after n then number becomes 3n.

ii) If we write 1 after n then number becomes 3n+1.

iii) If we write 2 after n then number becomes 3n+2.

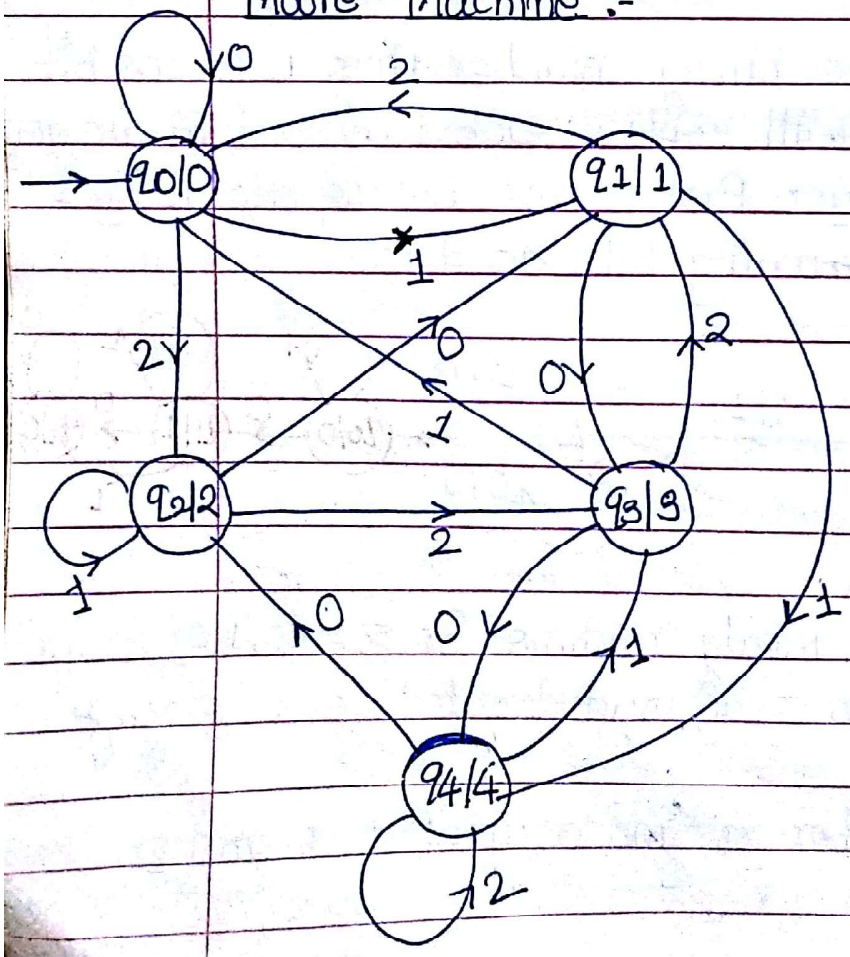
Now consider n=4.

40 → We get decimal value 12 that means 12%5=2 it gives remainder 2.

A1 → 13. =3.

A2 → 14 =4

Moore Machine :-



q_0 - remainder 0 state

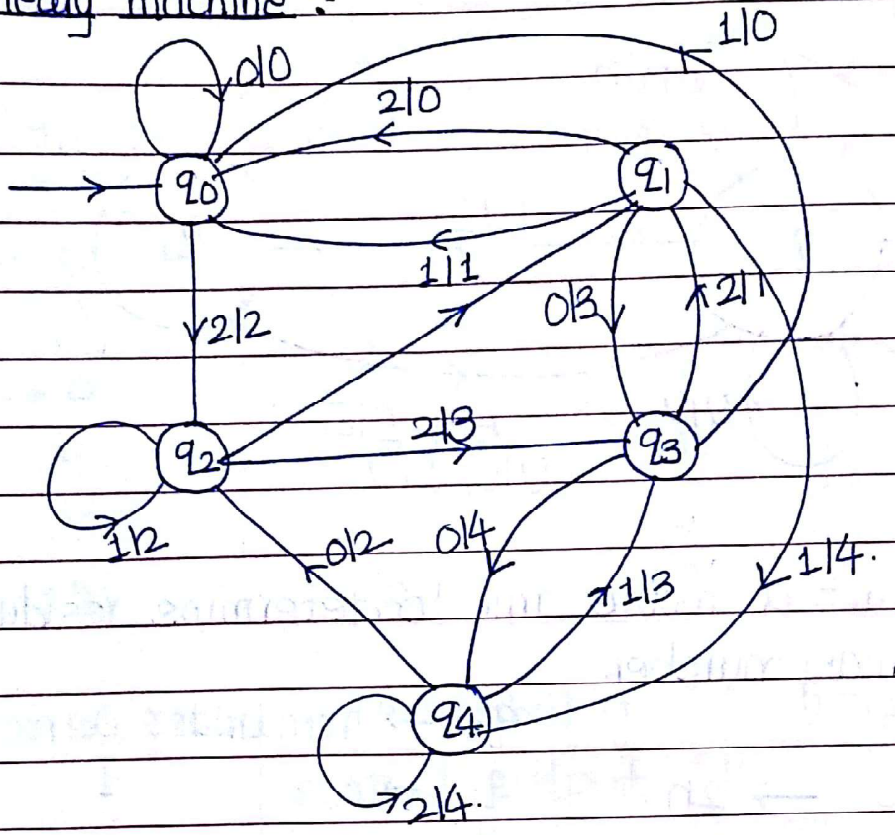
q_1 - 1

q_2 - 2

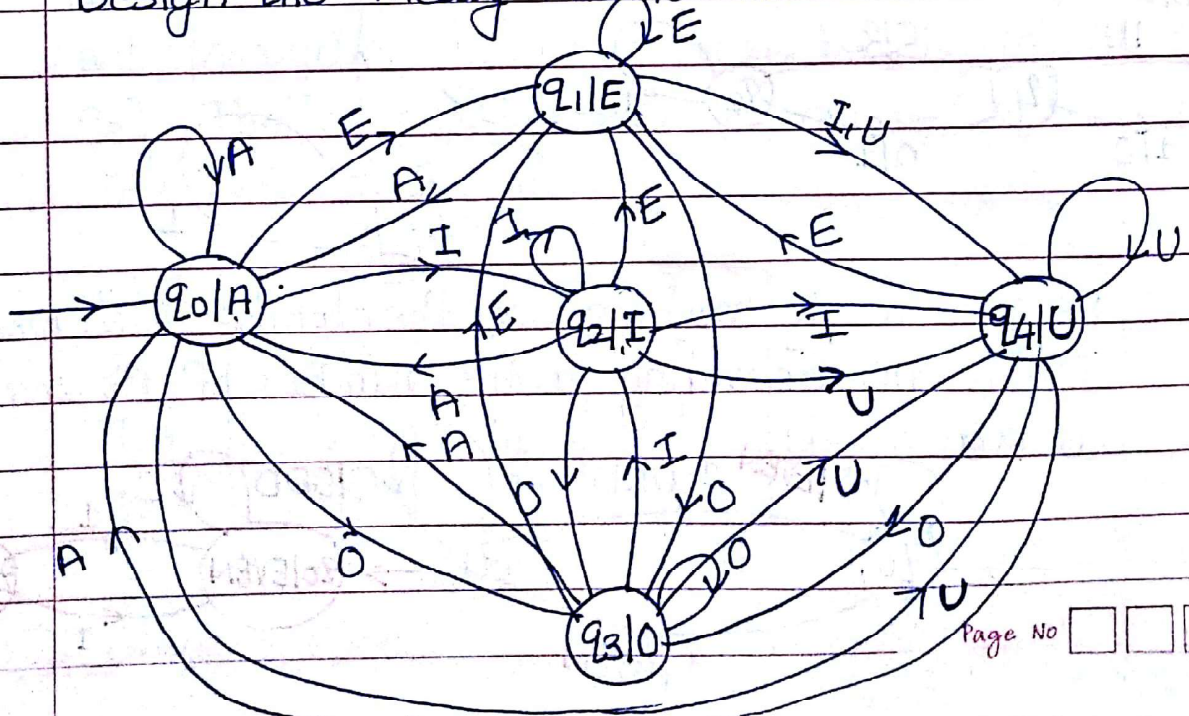
q_3 - 3

q_4 - 4.

Mealy machine :-

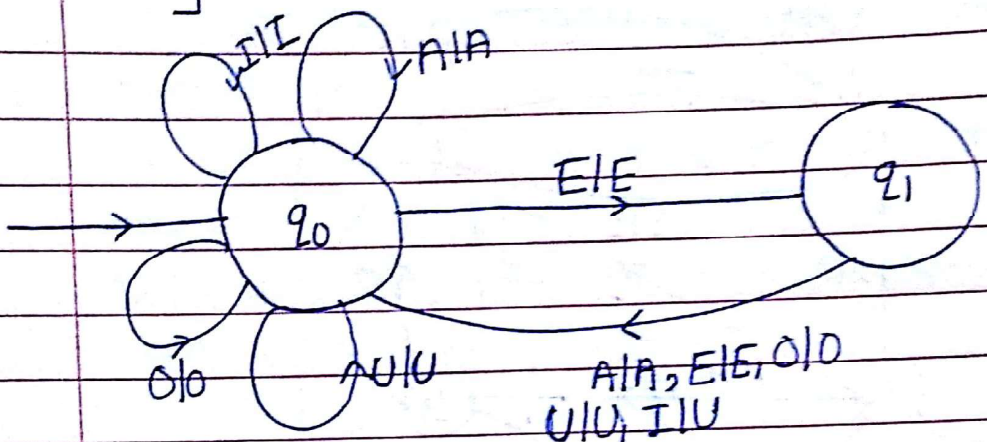


ex Design a Moore m|c that will read sequences ~~that~~ made up of letters A, E, I, O, U & will give as output the same sequences except that in case where an I directly follows an E, it will be changed to U. Design the mealy m|c for the same.



Page No

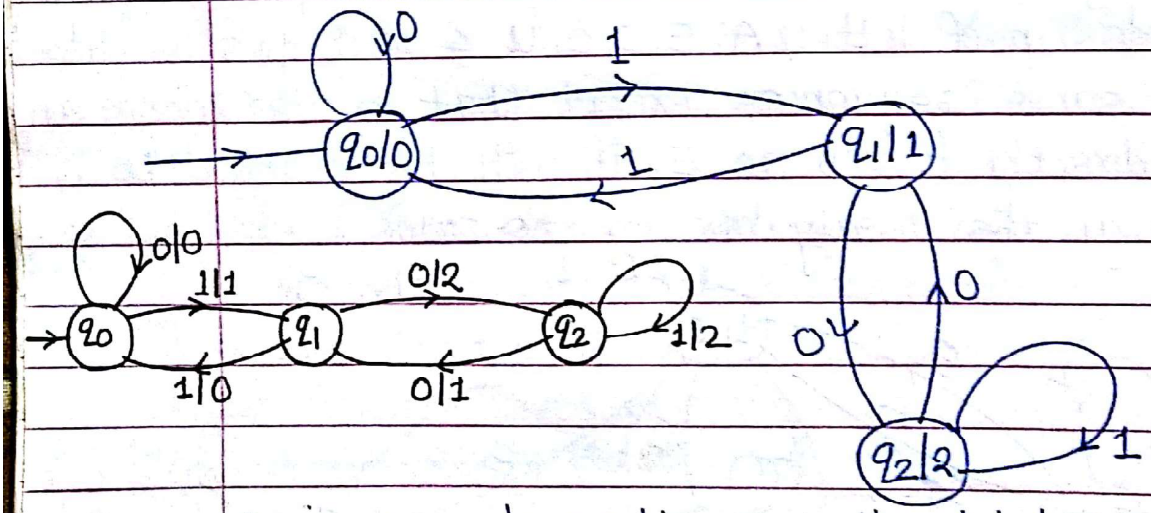
Mealy m|c



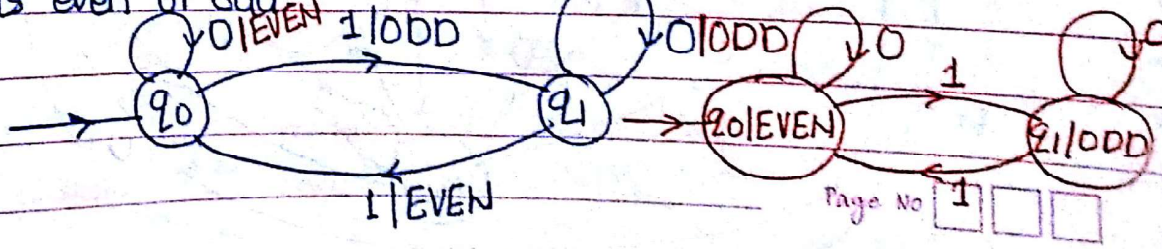
ex. construct a moore m|c to determine residue mod 3 for binary number.

q_0 → remainder 0.
q_1 → 1
q_2 → 2

$n0 \rightarrow 2n$.
$n1 \rightarrow 2n+1$.



ex. Design a mealy machine over the alphabet {0,1} which outputs EVEN, ODD according to the number of 1's encountered as even or odd



Equivalence of Moore and Mealy Machines:-

1] Conversion of a Moore m/c into a mealy m/c:-

current state	Next state		output
	0	1	
→ p	s	q	0
q	q	r	1
r	r	s	0
s	s	p	0

current state	I/P symbol 0		I/P symbol 1	
	state	output	state	output
→ p	s	0	q	1
q	q	1	r	0
r	r	0	s	0
s	s	0	p	0

Let $M = (Q, \Sigma, \delta, \lambda, q_0)$ be a Moore m/c. The equivalent Mealy m/c can be represented by $M' = (Q, \Sigma, \delta, \lambda', q_0)$. The output function λ' can be obtained as -

$$\lambda'(q, a) = \lambda(\delta(q, a))$$

ex.

Q/Σ	0	1	λ
q_0	q_0	q_1	0
q_1	q_2	q_0	1
q_2	q_1	q_2	2

	Input 0		Input 1	
Q\Σ	state	o/p	state	o/p
q_0	q_0	0	q_1	1
q_1	q_2	2	q_0	0
q_2	q_1	1	q_2	2

Tech max Pune U 2-113

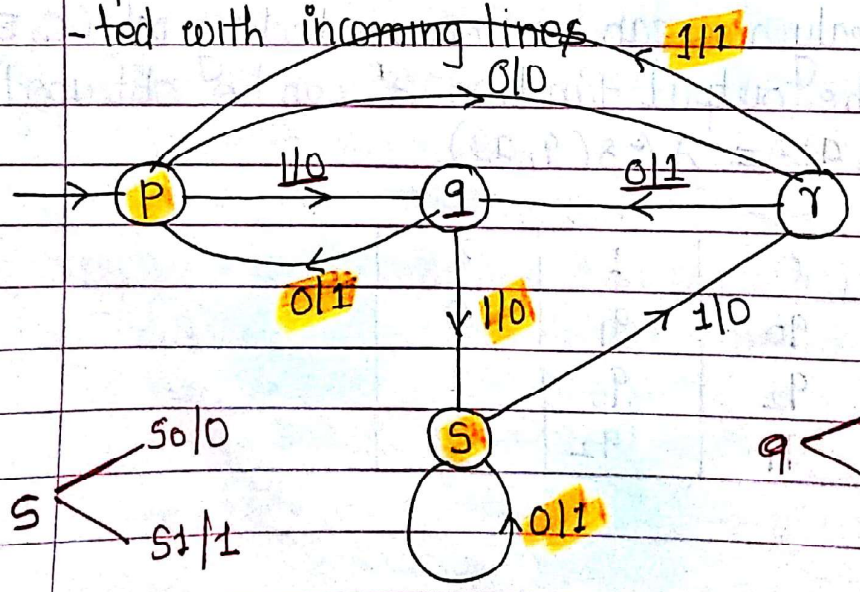
Conversion of a Mealy machine into a Moore Machi

ex

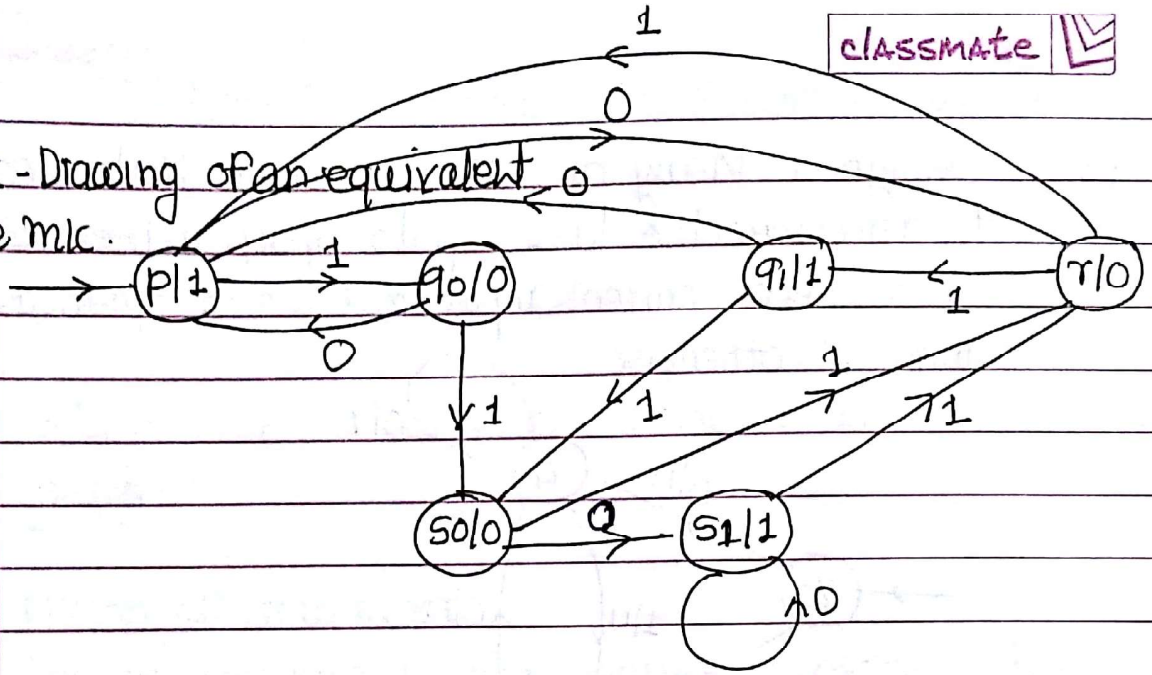
present state	Input 0		Input 1	
	state	o/p	state	o/p
→ p	r	0	q	0
q	p	1	s	0
r	q	1	p	1
s	s	1	r	0

Step I splitting of states:

It may be neccessary to split some of the states of a mealy machine to get an equivalent Moore m/c. splitting is based on incoming lines into a state and the associated o/ps. We must split a state to handle different o/p values associated with incoming lines.



step II - Drawing of an equivalent Moore m/c.

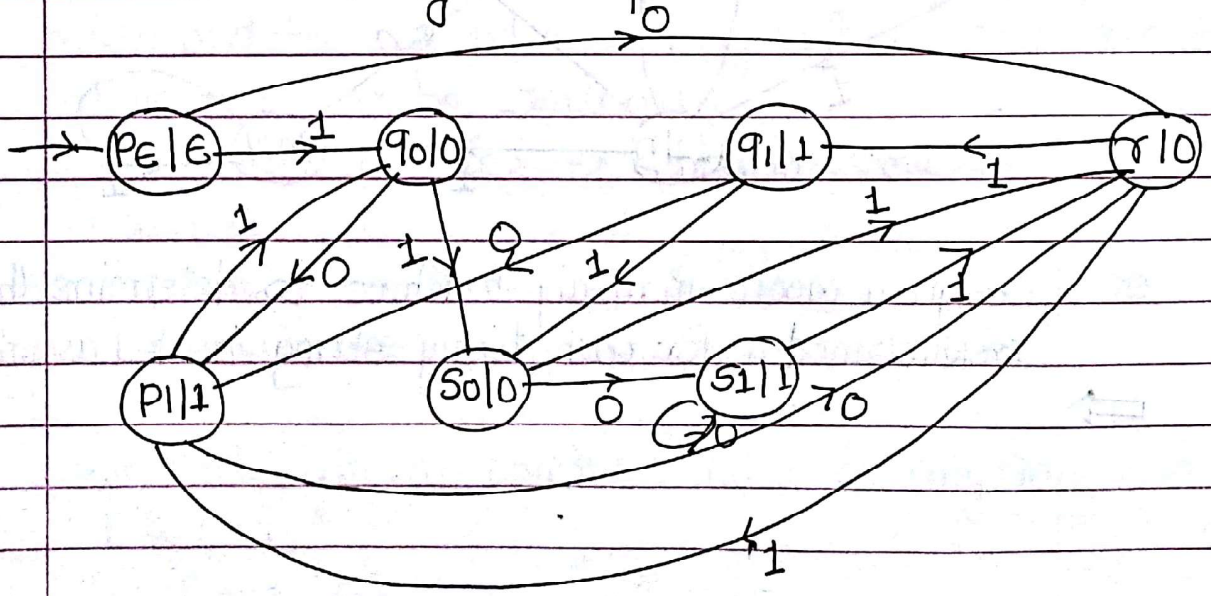


step III - Moore m/c produces an o/p 1 without any input. Thus the moore m/c gives an o/p 1 on a zero length sequence but the mealy m/c will produce no output for a zero length sequence.

To handle the above problem, we must split the start state p into two states.

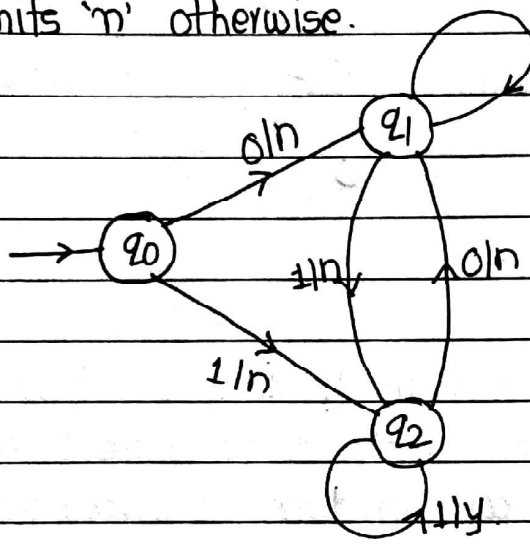
p_ϵ - Giving no output

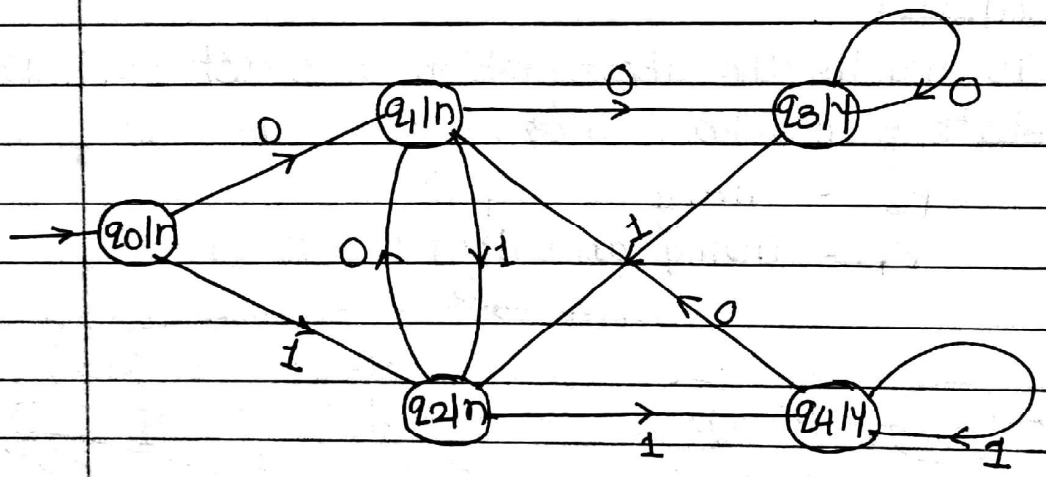
p_1 - Giving an output 1.



Page No

ex Design a Mealy and Moore machine that uses its state to remember the last symbol read and emits output 'Y' whenever current input matches to previous one and emits 'n' otherwise.





ex Design a Moore & mealy machine to determine the residue mod 4 for each binary string treated as integer

⇨