USN □□□□□□□□□□

CELEBRATING 25 YEARS
CMRIT
CMR INSTITUTE OF TECHNOLOGY, BENGALURU.
ACCREDITED WITH A+ GRADE BY NAAC

Internal Assessment Test 2 – November 2021

Sub:	Automata Theory and Computability			Sub Code:	18CS54	Branch:	ISE
Date:	20/12/2021	Duration:	90 min's	Max Marks:	50	Sem/Sec:	V A, B & C

OBE

Answer any FIVE FULL Questions — MARKS | CO | RBT

1) a) Write down the formal definition of NFA - epsilon (Non deterministic finite automata with epsilon move), explain extended transition function and language acceptance of NFA – epsilon. [4] CO1 L2

b) Prove the equivalence between NFA - epsilon and regular expression. [6] CO1 L2

2) Construct regular expression for the following languages. [2.5+2.5+2.5+2.5=10] CO3 L3

a) L = $\{n^a m^b \mid a,b >= 1, a*b >=3\}$, ∑ = {n,m}.

b) L = $\{a^{2n} b^{2m} \mid n,m >=0\}$, ∑ = {a,b}.

c) Set of all strings where no pair of consecutive ‘00’s is present. ∑ = {0,1}.

d) Set of all strings where number of 1’s is odd. ∑ = {0,1}.

3) a) Convert the following NFA - epsilon to NFA. Here q_s is the starting state, q_2 is the final state, and {Φ} denotes the null set. [5+5=10] CO3 L3

	0	1	Epsilon (null move)
q_s	$\{q_s\}$	{Φ}	$\{q_1\}$
q_1	{ Φ }	$\{q_3\}$	$\{q_2\}$
q_2	$\{q_2\}$	$\{q_2\}$	{ Φ }
q_3	$\{ q_1 \}$	{ Φ }	{ Φ }

(b) Design NFA – epsilon for the following regular expression. CO3 L3

(00+11)*1

4) (a) Construct regular expression for the following FSA. (Transition table is given below). Here q_3 is the final state and q_s is the start state. [5+5=10] CO3 L3

	a	b	Epsilon (null move)
q_s	{ Φ }	{ Φ }	$\{q_{1.}q_6\}$
q_1	$\{q_4\}$	$\{q_2\}$	{ Φ }
q_2	{ Φ }	$\{ q_5\}$	$\{ q_3\}$
q_3	$\{q_3\}$	$\{ q_3\}$	{ Φ }
q_4	$\{ q_1\}$	{ Φ }	{ Φ }
q_5	{ Φ }	$\{q_2\}$	{ Φ }
q_6	$\{q_8\}$	{ Φ }	$\{ q_7\}$
q_7	$\{ q_9\}$	{ Φ }	$\{q_3\}$
q_8	$\{ q_6\}$	{ Φ }	{ Φ }
q_9	{ Φ }	$\{q_7\}$	{ Φ }

(b) Design DFA for the given regular expression CO3 L3

$(01 + (1 + 00))^{*}0$

5	Prove the following closure properties of regular languages. (a) Reversal (b) Complement (c) Intersection (d) Set Difference	[2.5+2.5+2.5+2.5=10]	CO2	L2
6	(a) State Pumping Lemma (b) prove the given language is not regular using pumping lemma. L ={WW | W € $(a+b)^*$} ∑ = {a,b}	5+5=10	CO2 CO3	L2 L3

Faculty Signature CCI Signature HOD Signature

Formal def^n of NFA-ε move :

$M = (Q, \Sigma, q_0, \delta, F)$ where

- Q : finite set of states
- Σ : input alphabet
- q_0 : start state where $q_0 \in Q$
- δ : transition function, $\delta : Q \times \Sigma \cup \{\varepsilon\} \rightarrow 2^Q$
- F : finite set of final states, $F \subseteq Q$.

Now, we extend the transition function δ to strings, and here we introduce $\hat{\delta}$.

Basis : $\hat{\delta}(q, \varepsilon) = \varepsilon$-close$(q)$ [that is, if the label of the move is ε, then we can follow only ε-moves extending from state q, that is exactly what ε-close does.].

Induction : w is an input string of the followi form, $w = xa$ where a is the las symbol of w and x is the rest part. We comput $\delta(q, w)$ as follows:

1. let $\hat{\delta}(q, \overset{x}{\cancel{w}}) = \{P_1, P_2, .., P_K\}$. Now this path may end with one or mo transitions with ε move, and may ha Other ε-moves, as welle.

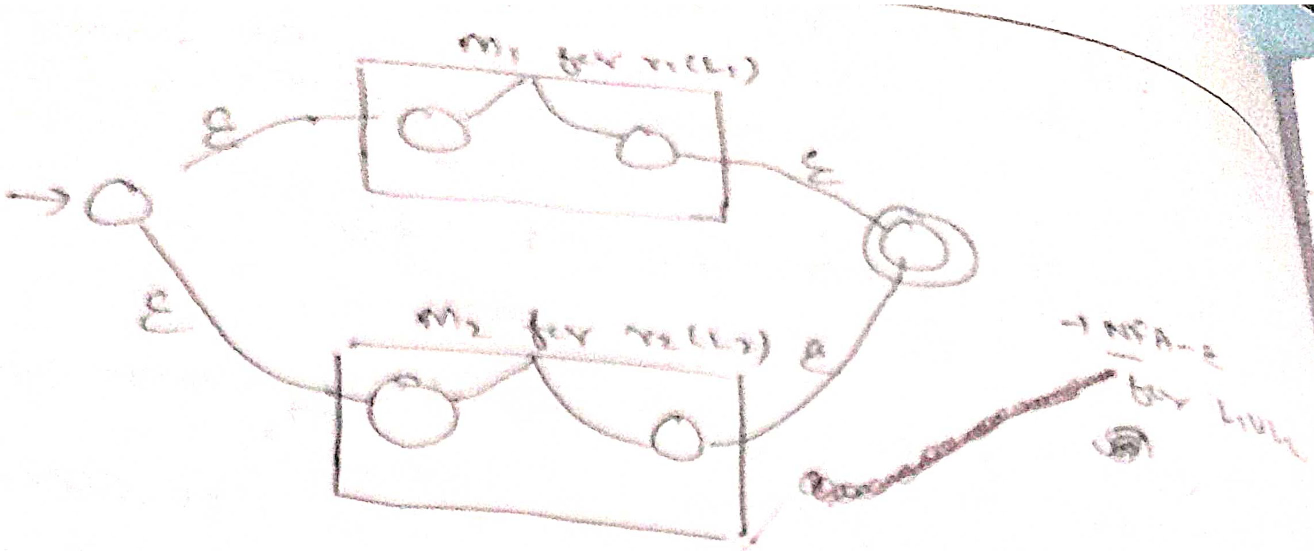
(2.) let $\bigcup_{i=1}^{k} \delta(p_i, a) = \{r_1, r_2, \ldots, r_m\}$. That is, follow all transitions with a from states we can reach from q along moves x. The r_j's are some of the states we can reach from q along moves w. The additional states we can reach with ε-moves.

(3) $\hat{\delta}(q, w) = \bigcup_{j=1}^{m}$ ε-close(r_i).

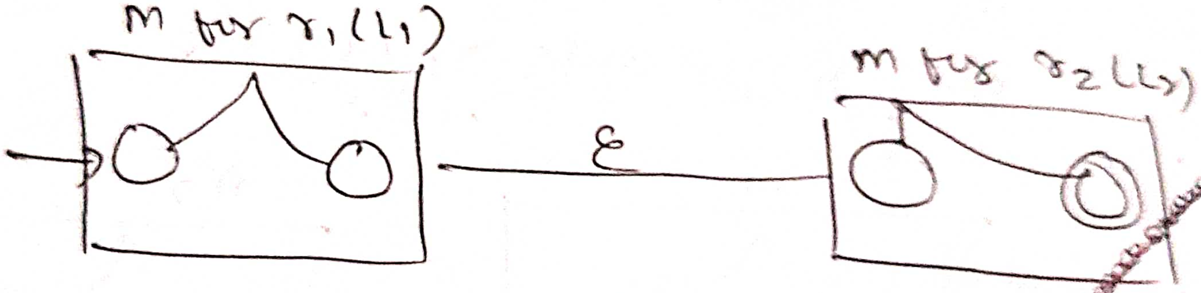
This additional closure step includes all the moves from q on w, by considering the possibility that there are additional ε-moves that we can follow after making a transition on the final symbol a

Now we define,

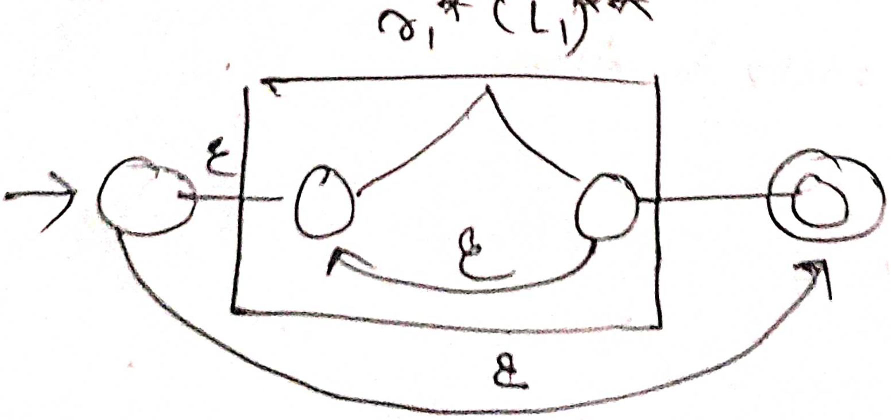
$L(M) = \{ w \mid \hat{\delta}(q_0, w) \cap F \neq \phi \}$



② **Concatenation** $r_1 r_2$ reperesent $L_1 L_2$, where $L_1 L_2$ is regular langua



③ **Kleen's star** r_1^* represents L_1^*



(a) $n^+ mmm^+ + nmn^+m^+ + nn^+ mm^+$

(b) $(aa)^* (bb)^*$

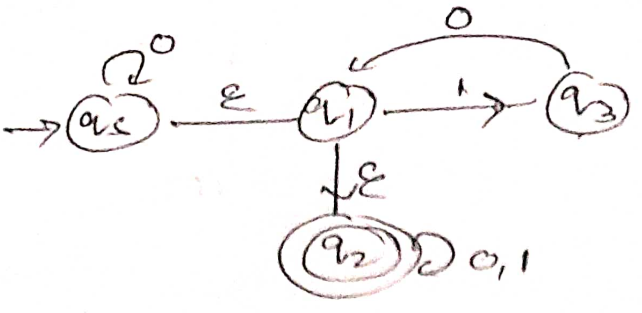
(c) $(0+\varepsilon)(1+10)^*$

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

[only zeroes can be accepted, if 1 is present it should be in odd number]

3

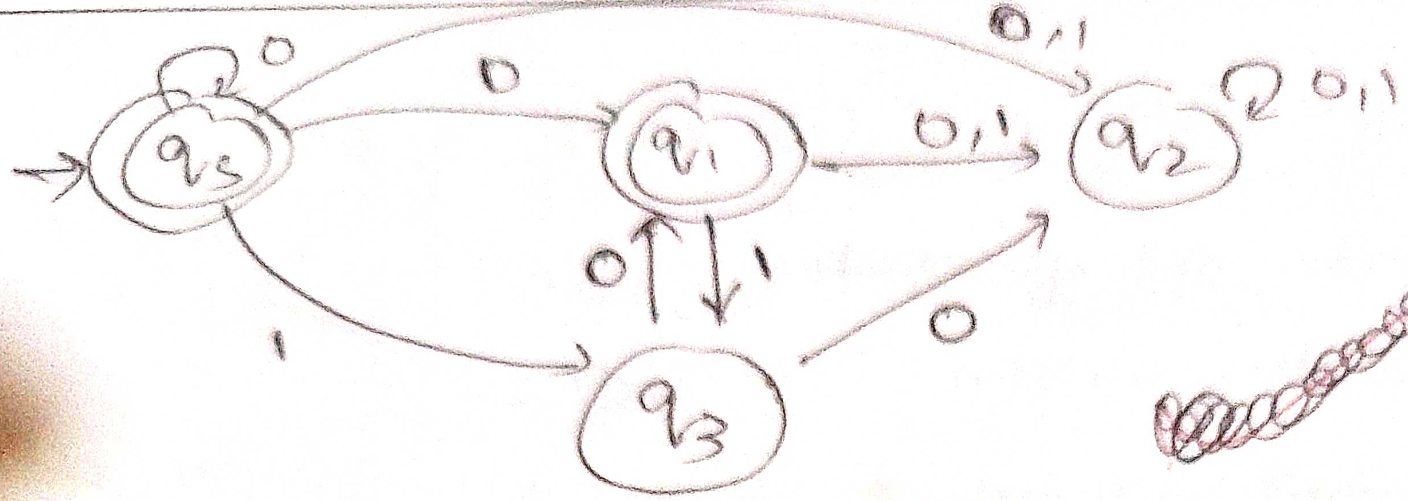
(a)



← NFA – ε

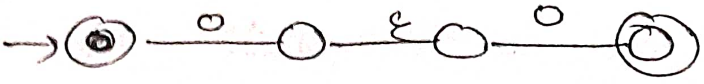
Transition table

	ε*	0	ε*	ε*	1	ε*
q_s	$\{q_s, q_1, q_2\}$	$\{q_s, q_2\}$	$\{q_s, q_1, q_2\}$.	$\{q_s, q_1, q_2\}$	$\{q_3, q_2\}$	$\{q_3, q_2\}$
q_1	$\{q_1, q_2\}$	$\{q_2\}$	$\{q_2\}$.	$\{q_1, q_2\}$	$\{q_3, q_2\}$	$\{q_3,$
q_2	$\{q_2\}$	$\{q_2\}$	$\{q_2\}$	$\{q_2\}$	$\{q_2\}$	$\{q$
q_3	$\{q_3\}$	$\{q_1\}$	$\{q_1, q_2\}$	$\{q_3\}$	ϕ	ϕ

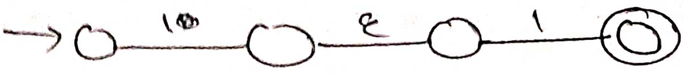

0
0
0,1
0,1
0,1
q_s
q_1
q_2
0
1
1
0
q_3

⑥ (00+11)* 1

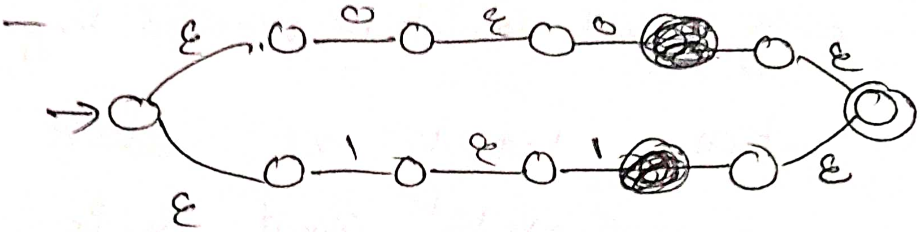
00 -



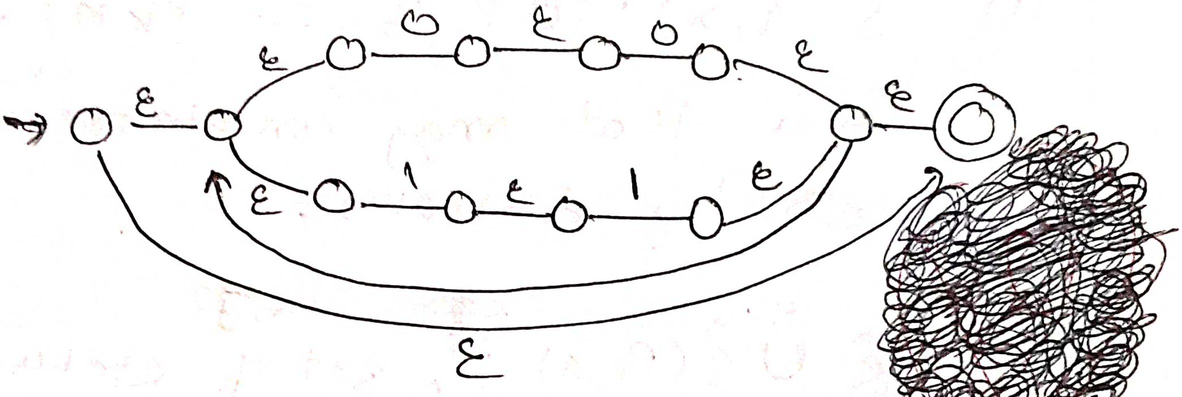
11 -



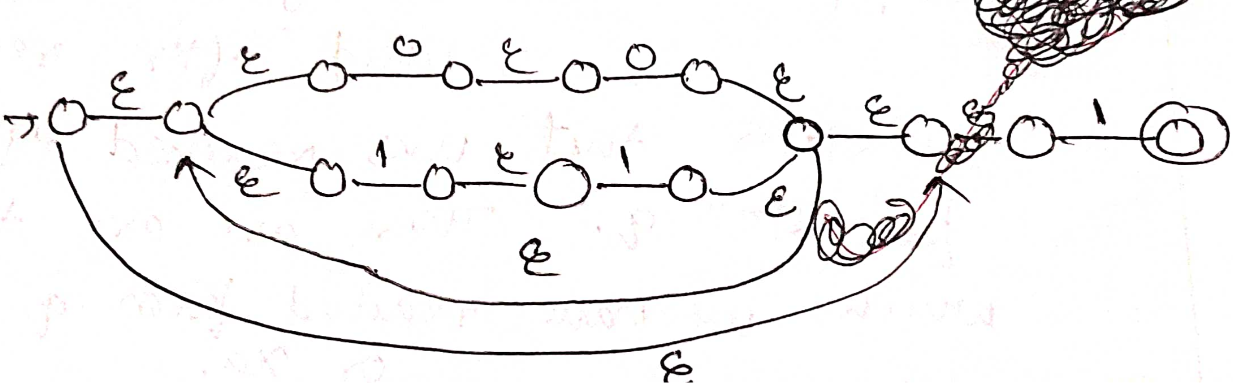
(00+11) -



(00+11)* -

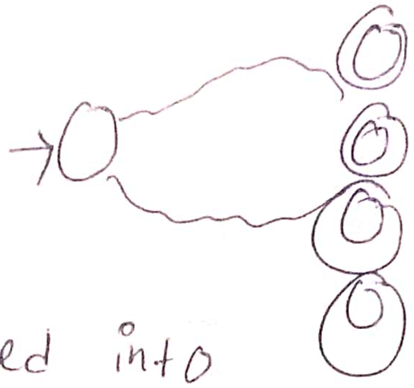


(00+11)* 1

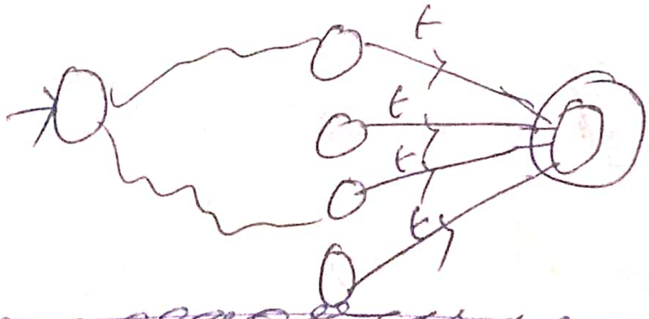


Properties of regular languages / Give any regular expression we can construct NFA-ε

Note: W.l.o.g we can consider that any NFSA-ε can have only one final state.



→ if can be changed into a single final state diagram by introducing ony new final state, and adding ε-moves from previous final states to new final state.



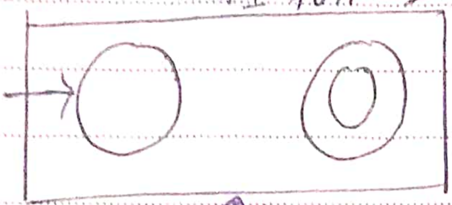
r_1 is a regular expression.

r_2 " " " " .

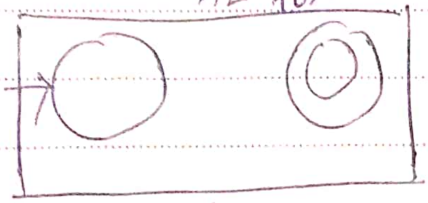
M_1 represents r_1

M_2 " r_2

M1 for r_1



M2 for r_2

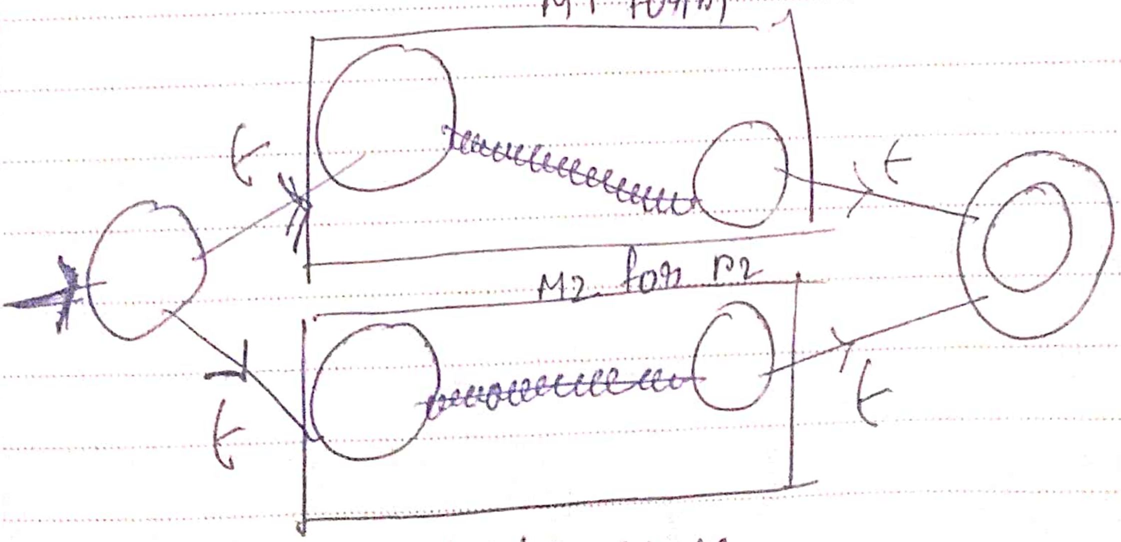


r_2

Union

How can we represent $r_1 + r_2$ by transition diagram or NFA - ε. r_1 represents L_1.

r_2 " L_2.



1- Introduce new intial state.

- Make previous intial states, not initial.
- Add ~~[illegible]~~ ε move from newly introduced initial state two previous initial states.

- Introduce new final state.
- Make previous final states, non final.
- Add ϵ moves from previous final states to newly introduced final states.

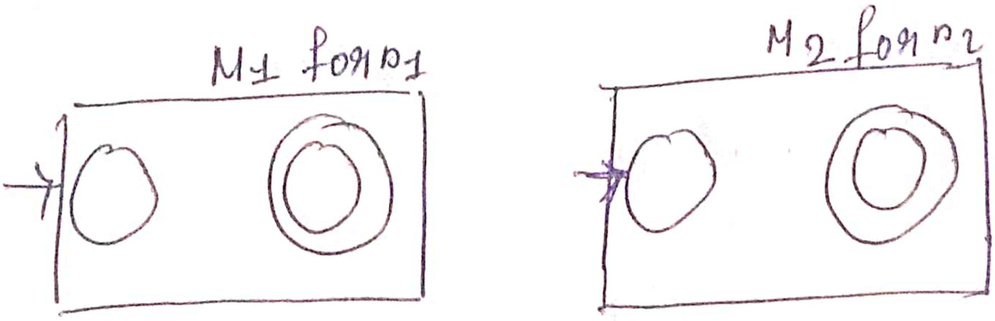
r_1 represents L_1 (regular language)

r_2 " L_2 (regular language)

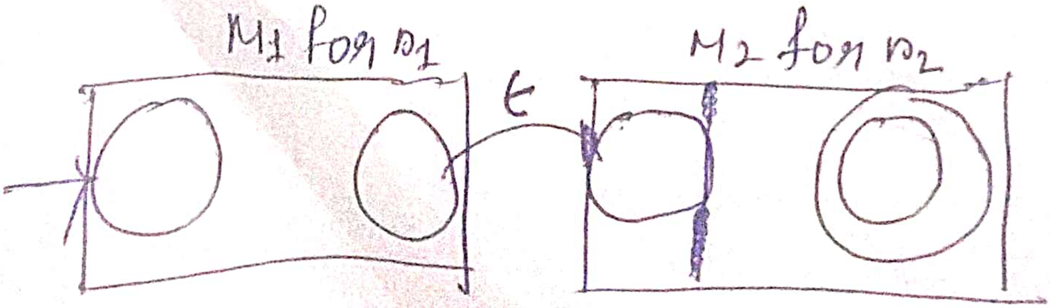
$L_1 \cup L_2 = \{x \mid x \in L_1 \vee x \in L_2\}$

We have represented $r_1 + r_2$ ($L_1 \cup L_2$) by NFA-ϵ, therefore $L_1 \cup L_2$ is also reguear language.

Concatenation : r_1, r_2 represent L_1, L_2 respectiv



How to represent $r_1 r_2$ ($L_1 L_2$).



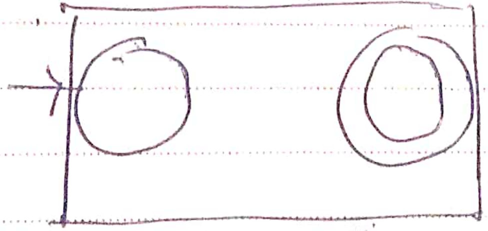
- Final state of M_1 becomes non final.
- Initial " " M_1 remains initial state.
- Final " " M_2 remains final state.
- ~~Previous final state of M_1 and~~
- Initial state of M_2, will no more initial state.
- Previous final state of M_1 and previous initial state of M_2 - add ϵ move between them.

$$L_1 L_2 = \{xy \mid x \in L_1 \wedge y \in L_2\}$$

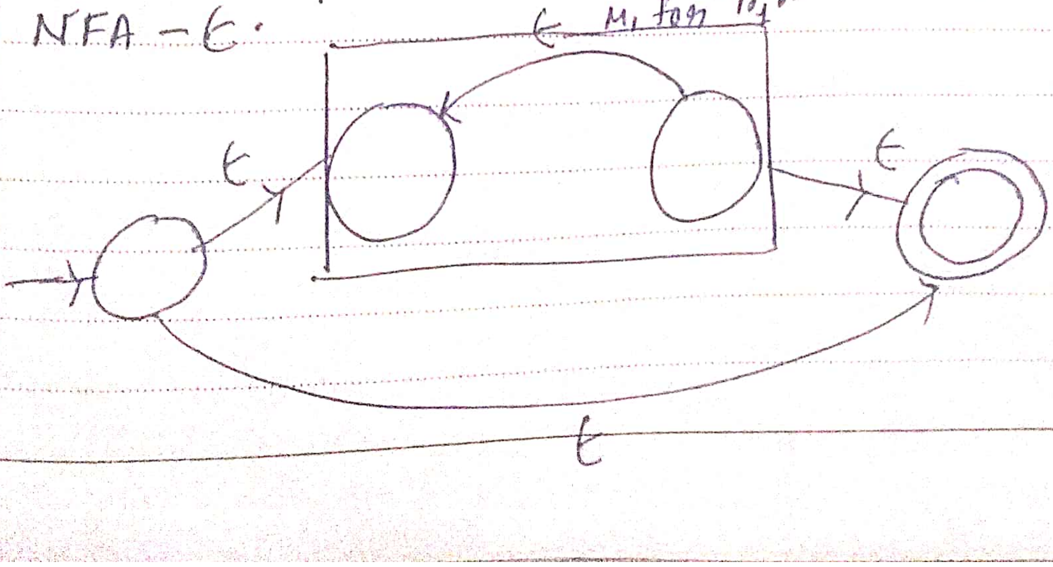
We have presented $r_1 r_2$ ($L_1 L_2$) by NFA-ϵ, therefore $L_1 L_2$ is also regular language.

$(L_1)^*$

M_1 for r_1 ~~ϵ~~ r_1 represents L_1



How to represent r_1^* (L_1^*) by an NFA-ϵ.

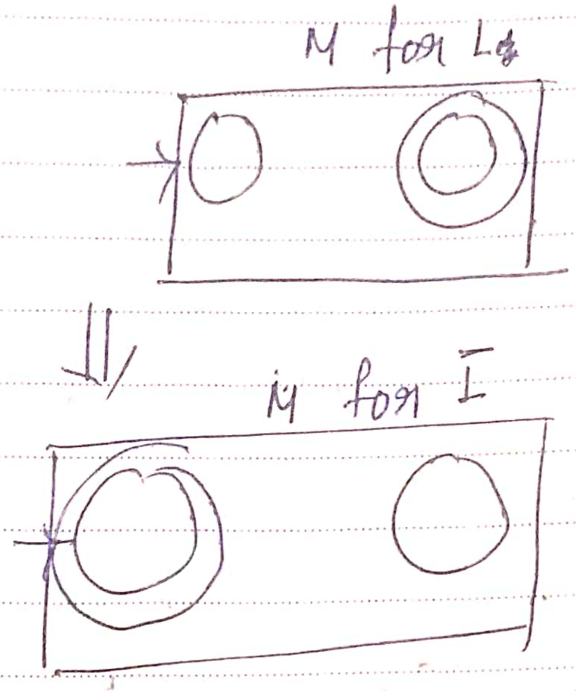


L_1^* is also regular

$\bar{L}$: L is a regular language.
Easily we can just make a swap between final and non final state, we get the complement.

for L

for $\bar{L}$



for $\bar{L}$, we can design M easily, so $\bar{L}$ is also a regular language.

$L_1 \cap L_2$: L_1 is regular.
L_2 " " .

- we know $\bar{L_1}$, $\bar{L_2}$ are also regular, as we already showd the complement of regular language is also regular.

- $\bar{L_1} \cup \bar{L_2}$: Regular language is closed under union, so $\bar{L_1} \cup \bar{L_2}$ is regular.

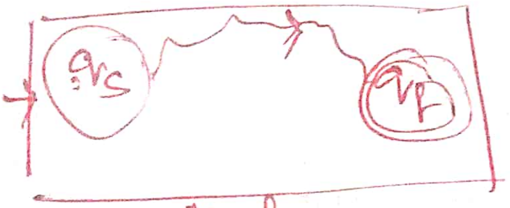
- We know regular languages are closed under complement.

$$\therefore \overline{\overline{L_1} \cup \overline{L_2}} \quad \text{(using D'morgans law)}$$

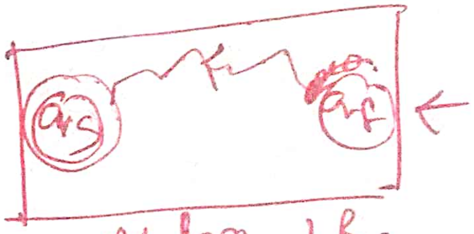
$$= \overline{\overline{L_1}} \cap \overline{\overline{L_2}}$$

$$= L_1 \cap L_2 \quad \text{(regular)}$$

Reversal : L is regular, then L^R is also regular.



M for L

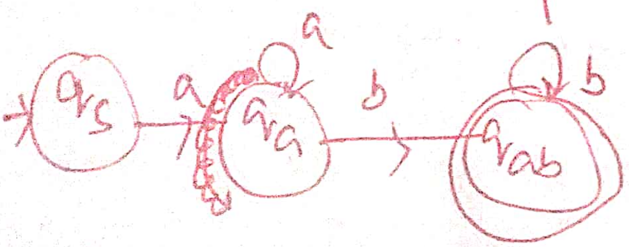


M for LR.

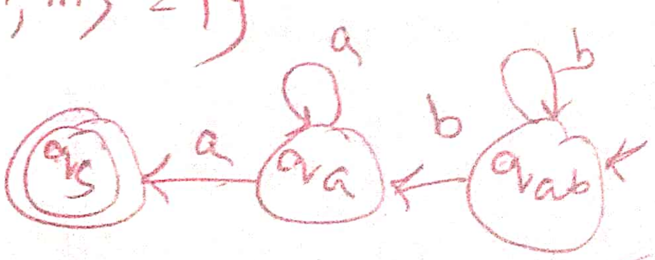
- Swap between final & start state.
- Change the direction of all the transitions.

Ex: $L = \{a^n b^m \mid n, m \geq 1\}$

$L^R = \{b^m a^n \mid n, m \geq 1\}$



$a^n b^m$



$b^m a^n$

$(ab)^*$

Derivative : Let $x \in \Sigma^*$, L is a language

L_x = the derivative of L with respect to x.

$= \{ y \mid xy \in L \}$

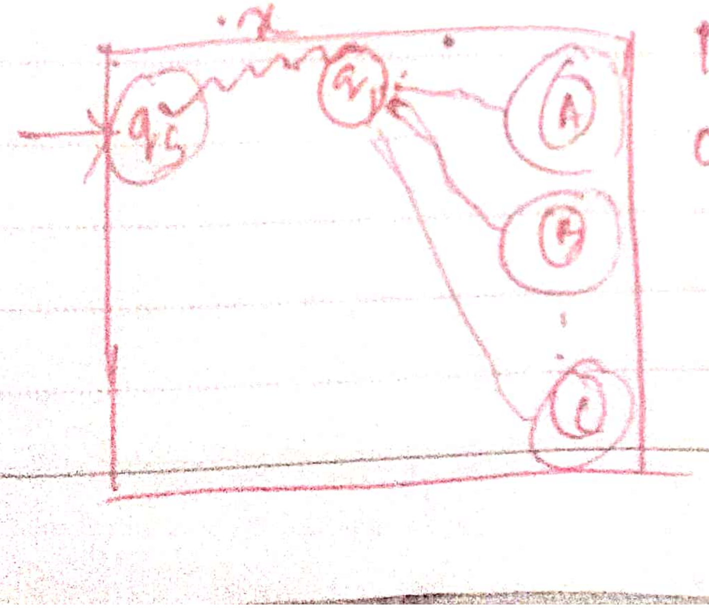
Set of strings beginning with a and ending with b.

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

$L_a = (a+b)^*b$ [set of all strings ends with 'b'.]

$L_b = \phi$

If L is regular then derivative of L is also regular. L is accepted by DFA.



Now remove q_s, and q_1 will be the new start state.

Set difference :

L_1 is a regular language.

L_2 " " " "

$L_1 - L_2$ is also regular language.

$$L_1 - L_2 = L_1 \cap \overline{L_2}$$

L_2 is regular

$\overline{L_2}$ " " according to previous proof as regular languages are closed under complement.

Also regular languges are closed under intersection, therefore $L_1 \cap \overline{L_2}$ is regular where $\boxed{L_1 \cap \overline{L_2} = L_1 - L_2}$

Pumping lemma :

Pumping lemma is used to prove that a language is not regular.

Lemma : Let L be a reguear language. then there exists an integer $P >= 1$ depending only on L such that every string w in L of length atleast P (P is called th "pumping length") can be written as $w = xyz$ (w can be divided into three substriings), satisfying the following conditions :

- $|y| >= 1$
- $|xy| <= P$
- $(\forall n >= 0)(xy^n z \in L)$.