Unit - I

Finite Automata & Formal Languages

Automata theory is the study of abstract computing devices or machines

Central concepts of Automata Theory

Alphabets (Σ)

An alphabet is a finite, nonempty set of symbols.

① $\Sigma = \{0, 1\}$, the binary alphabet

② $\Sigma = \{a, b, c, \ldots\}$, set of all lower case letters.

③ The set of all ASCII characters.

Strings :-

Finite sequence of symbols chosen from some Σ

Ex:- 01110, 111 etc are the strings from $\Sigma = \{0, 1\}$

The empty string (ϵ)

string with zero occurrences of symbols.

Length of a string

is the numbers of positions for symbols in string

$|w| = 3$ where $w = 001$

$|w| = 0$ where $w = \epsilon$

powers of an alphabet

set of all strings of certain length from that alphabet

Σ^k is the set of all strings of length k.

Ex:- $\Sigma = \{0, 1\}$ then

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

$\Sigma^2 = \{00, 01, 10, 11\}$

The set of all strings over an alphabet Σ is de

Σ^* is $\Sigma^* = \{\epsilon, 0, 1, 00, 01, 10, 11, 000, 001 \ldots\}$

$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots\ldots$

$\boxed{\Sigma^* = \Sigma^n \text{ where } n >= 0}$

If we want to exclude ϵ from the Σ^n we us

is $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \ldots\ldots$

$\Rightarrow \Sigma^* = \Sigma^0 \cup \Sigma^+$ or $\Sigma^+ \cup \{\epsilon\}$.

Concatenation of strings

Let $x = abc$, $y = ad$ be two strings then x is the concatinated string is $xy = abcad$

$|xy| = 5$

$\boxed{\epsilon w = w\epsilon = w}$ where w is any string.

ϵ is identity for concatination.

3) Languages

Set of strings all of which are chosen from Σ^* where Σ is a particular alphabet.

If Σ is an alphabet, $L \subseteq \Sigma^*$, then L is a lan of Σ.

L need not includes strings with all the sym of Σ. so once L is established as a language of language over any alphabet that

Ex for languages

1. The lang of all strings consisting of n 0's follow by n 1's, for som n, >= 0:
{ ∈, 01, 0011, 000111, ... }.

2. The set of strings of 0's & 1's with equal no of 0

{ ∈, 01, 10, 0011, 0101, 11100100 .... }

3. The set of binary nos whose value is a prime.

{ 10, 11, 101, 111, 1011, ... }

4. Σ^* is a lang for any alphabet Σ.

5. ϕ, the empty lang, is a lang over any alphabet.

. { ∈ }, the lang consisting of only the empty string, is a lang over any alphabet.

Here $\phi \neq \{\in\}$;

ϕ have non strings & { ∈ } have atleast one str

oblems :-

is a question of deciding whether a given string is meber of some particular language.

Given a string w in Σ^*, decide whether or not w in in L.

Deterministic Finite Automata

The term 'deterministic' refers to the fact that on each input there is one and only one state to which the automaton can transition from its current state.

Definition of a DFA

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

Q: is a set of all states.

Σ is any alphabet

q_0 is initial state

F is set of accepting states. $F \subseteq Q$

δ is a transition function.

$Q \times \Sigma = Q$ $\Rightarrow \delta(q_1, a) = q_2$

How is DFA Processes Strings:

The lang of DFA is the set of all strings that DFA accepts.

Let $w = a_1 a_2 a_3 \ldots a_n$ be a input strings

$\delta(q_0, a_1) = q_1$

$\delta(q_1, a_2) = q_2$

$\delta(q_{n-1}, q_n) = q_n$.

If q_n is a member of F then the w is accepted else it is rejected.

:- { w | w is of form xoiy for some strings x & y consisting of 0's & 1's only }.

other way description using x & y

{ xoiy | x & y are any strings of 0's & 1's }.

Automata for the given lang will be the

$A = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_2\})$

Q S δ q0 F

where δ is given as.

$\delta(q_0, 1) = q_0$

$\delta(q_0, 0) = q_1$

$\delta(q_1, 1) = q_2$

$\delta(q_1, 0) = q_1$

$\delta(q_2, 0) = q_2$

$\delta(q_2, 1) = q_2$

plus Notation for DFA's

There are two notation for describing automata.

1. A transition diagram - its a graph.
2. A transition table - tabular listing of δ function.

Transition diagram:-

DFA for accepting all strings with substring 0i

q0 → Initial state q2 → Accepting state.

② Transition Tables :-

	0	1
→q_0	q_1	q_0
q_1	q_1	q_2
*q_2	q_2	q_2

Transition table for accepting all strings with 01 as sub

Extending the Transition Function to Strings

DFA defines a lang : the set of all strings result in a sequence of state transitions from start state to an accepting state

In terms of the transition diagram, the lang DFA in the set of labels along all the paths t lead from the start state to any accepting st

Extended transition function that describes wha happens when we start in any state & follow sequence of inputs.

The ETF $\hat{\delta}$ is a fun that takes a state q & w & returns a state p → the state that the autom reaches when starting in state q & processing t sequence of inputs w.

Basis :- $\hat{\delta}(q, \epsilon) = q$. i.e if we are in state q and read no i/p, then we are still in state q

Induction :-

Assume string w of form xa where a is last symb of w. & remaining as x.

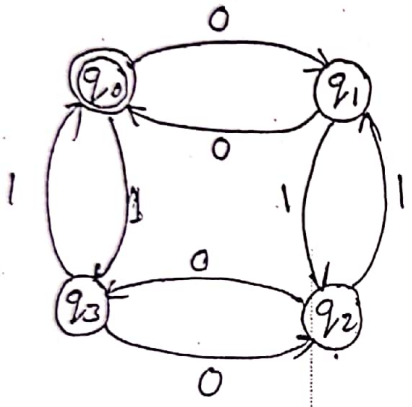
ii) if w = 110 and then x = 11 and a = 0.

$$\hat{\delta}(q, w) = \delta(\hat{\delta}(q, x), a);$$

Examples

Design DFA to accept the languages.

L = {w/w has both an even no of 1's & 0's}



	0	1
*→ q_0	q_1	q_3
q_1	q_0	q_2
q_2	q_3	q_1
q_3	q_2	q_0

$Q = \{q_0, q_1, q_2, q_3\}$

δ = as shown in transition table.

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

q_0 = Initial State

$F = \{q_0\}$.

Construct DFA

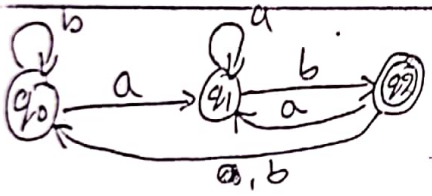
1) 

L = { w | w with atleast 2 1's }.

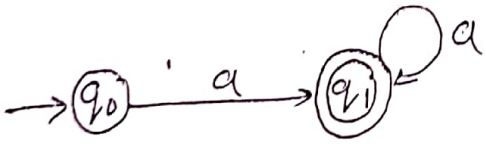
2) L = { w | w ends with ab }

abaab - Accept

abb - Not.

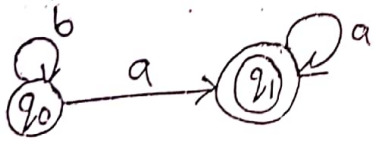


3) L = { w | a^n where n >= 1 }

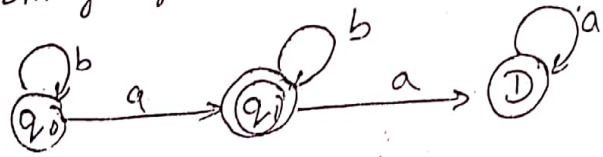


4) L = { w | a^n where n >= 0 }

5) L = { w | string of a's & b's having atleast o



6) L = { w | strings of a's & b's having exactly one a }



DFA construction steps

Step 1 :- Identifying the min string

Step 2 :- Identifying the alphabet

Step 3 :- Construct the skeleton DFA.
DFA for the string identified in step 1

Step 4 :- Identify the transitions not defined in step

Step 5 :- construction of DFA.

L = {w / strings of a's & b's starting with ab}

Step 1 :- The minimum string ab.

Step 2 :- The alphabet $\Sigma = \{a, b\}$

Step 3 :- DFA for ab (skeleton DFA)

Step 4 :- Identifying the transitions not defined in step 3.

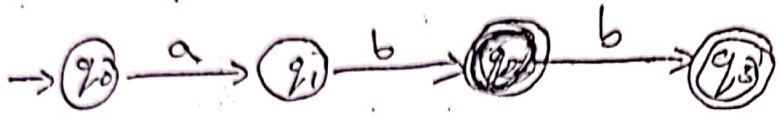
① $\delta(q_0, b) = ?$
② $\delta(q_1, a) = ?$
③ $\delta(q_2, a) = ?$
④ $\delta(q_2, b) = ?$

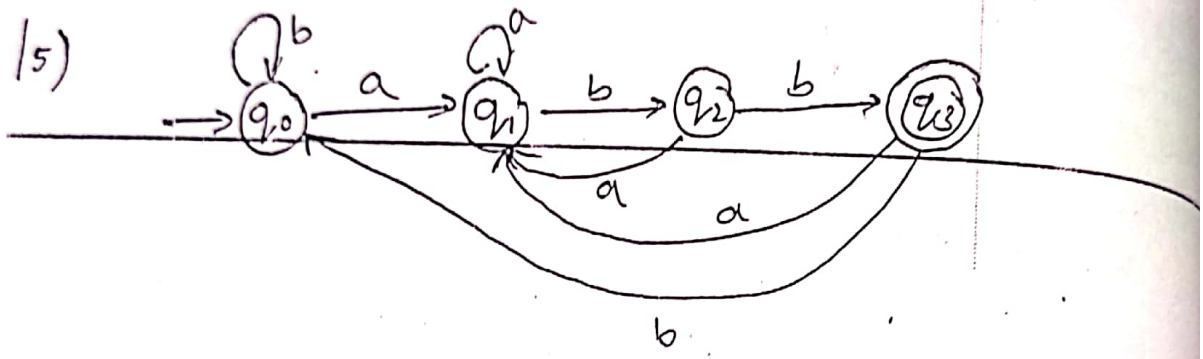
Step 5 :- construct DFA

9) $L = \{ w / (a+b)^n abb , n >= 0 \}$

① abb

② $\Sigma = \{a, b\}$

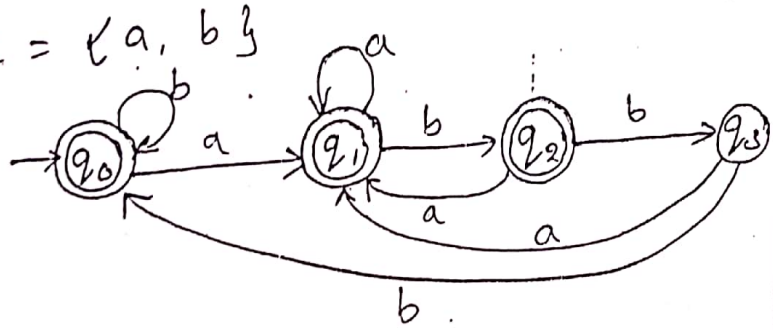
③ 

15) 

3) $L = \{ w \mid$ strings of a's & b's that do not end with abb $\}$

① a or b

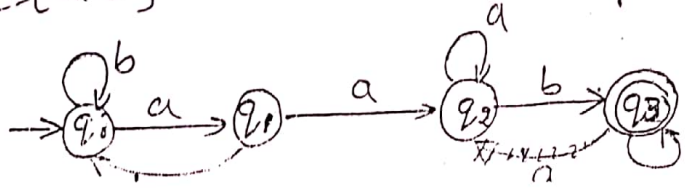
② $\Sigma = \{a, b\}$

③ 

10) $L = \{ w \mid$ strings of a's & b's having aab as substring $\}$

① aab

② $\Sigma = \{a, b\}$

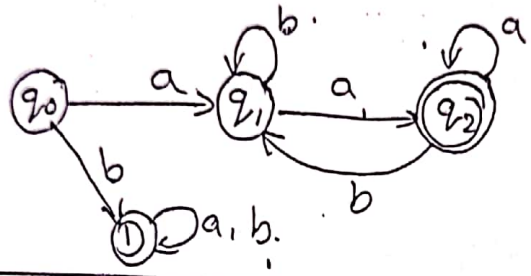
③ 

Draw a DFA to accept strings of a's & b's such that $L = \{awa \mid w \in (a+b)^n ; n >= 0\}$ ⑥

① min string aa

② $\Sigma = \{a, b\}$

③



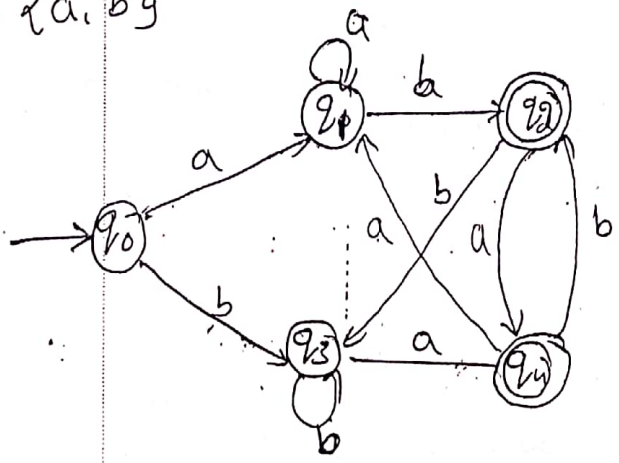
---

accept strings of a's & b's ending with ab/ba

① min string ab or ba

② $\Sigma = \{a, b\}$

③

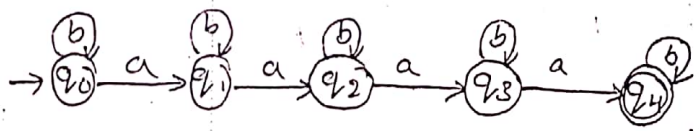


DFA to accept strings of a's & b's having 4 a's

① aaaa.

② $\Sigma = \{a, b\}$.

③



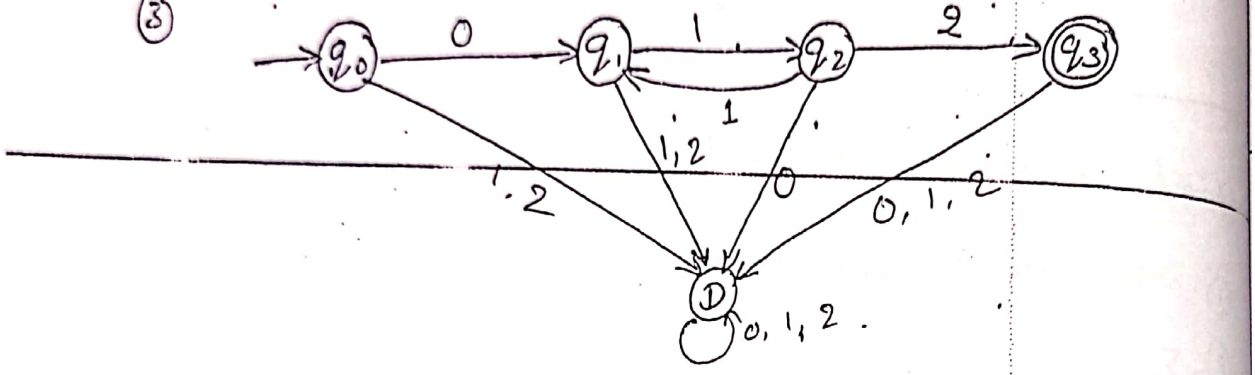
$L = \{w : n_a(w) \leq 3, \ w \in \{a, b\}^*\}$

(4) Strings of 0's, 1's & 2's beginning with a '0' followed by odd no of 1's and ending with

① min string 012

② $\Sigma = \{0, 1, 2\}$

③

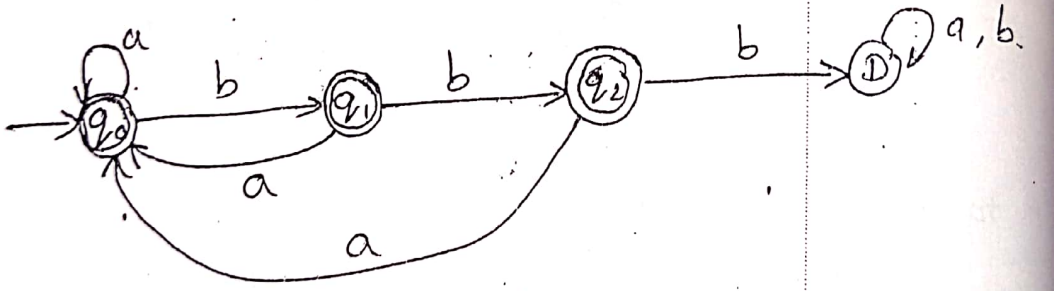


(5) Strings of a's & b's with at most 2 consicutiv

① bb

② $\Sigma = \{a, b\}$

③

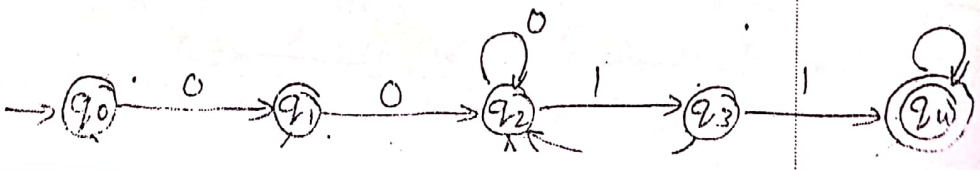


(6) Strings of 0's & 1's starting with atleast 2 0's a ending with atleast 2 1's.

① 0011

② $\Sigma = \{0, 1\}$

③

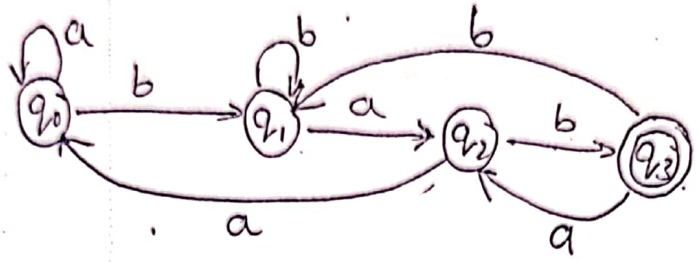


) $L = \{ wbab \mid w \in \{a, b\}^* \}$ (7)

① bab

② $\Sigma = \{a, b\}$

③



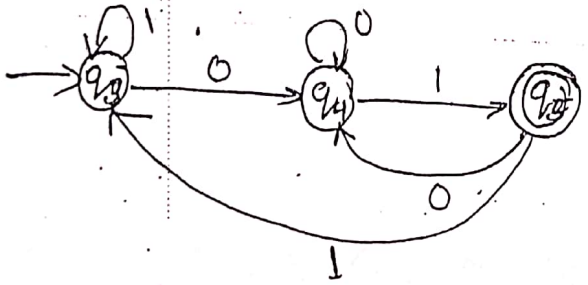
Strings of a's & b's having not more than 3 a's

Strings that either begins or ends or both with 01

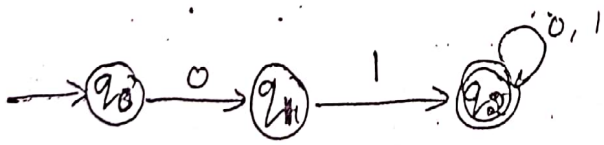
① 01

② $\Sigma = \{0, 1\}$

③

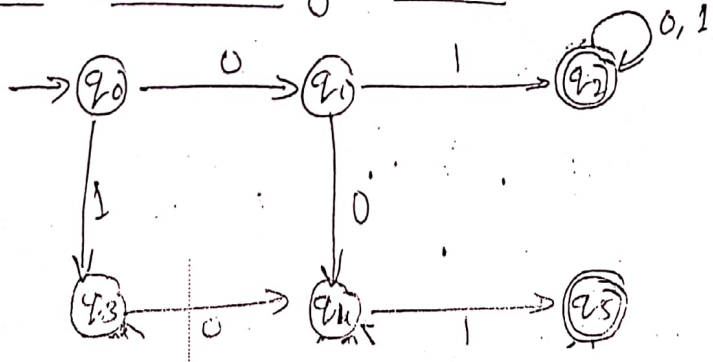


DFA for ending with 01



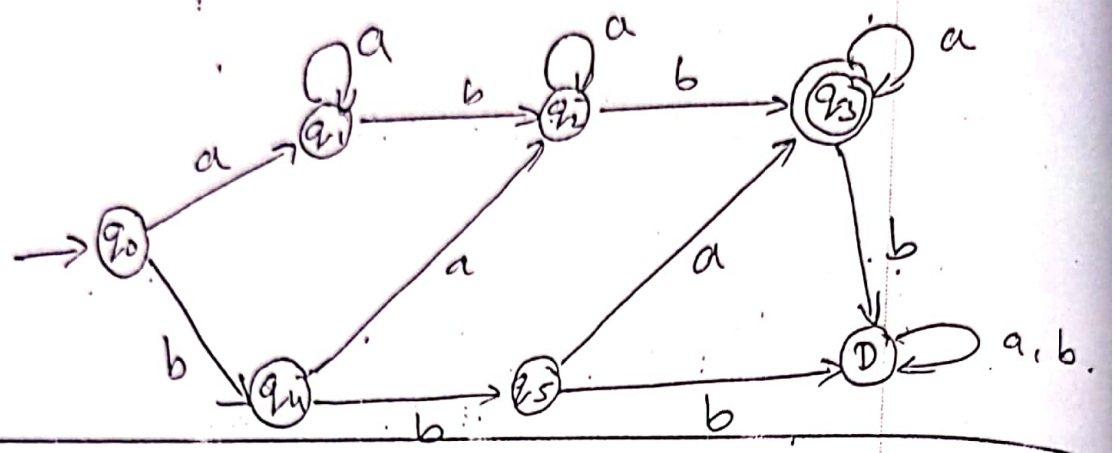
DFA for starting with 01

Now combining 'both'.



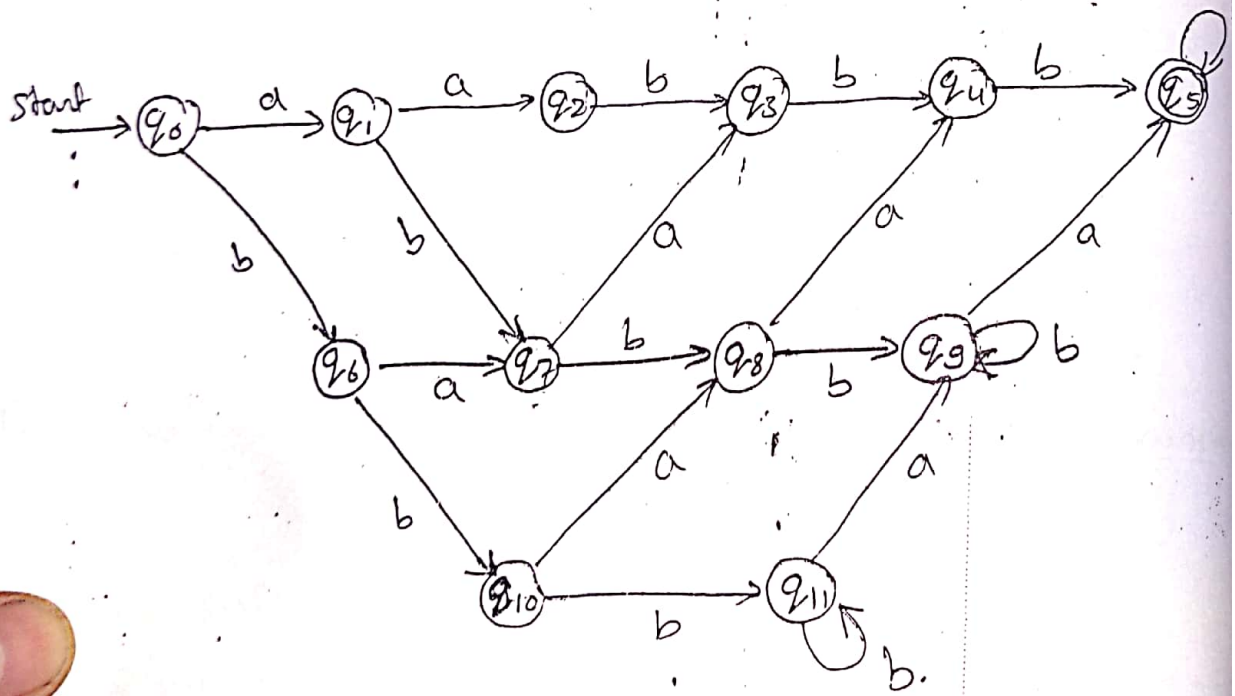
(19) $L = \{ w : n_a(w) \geq 1,\ n_b(w) = 2 \}$

① abb , bab , bba



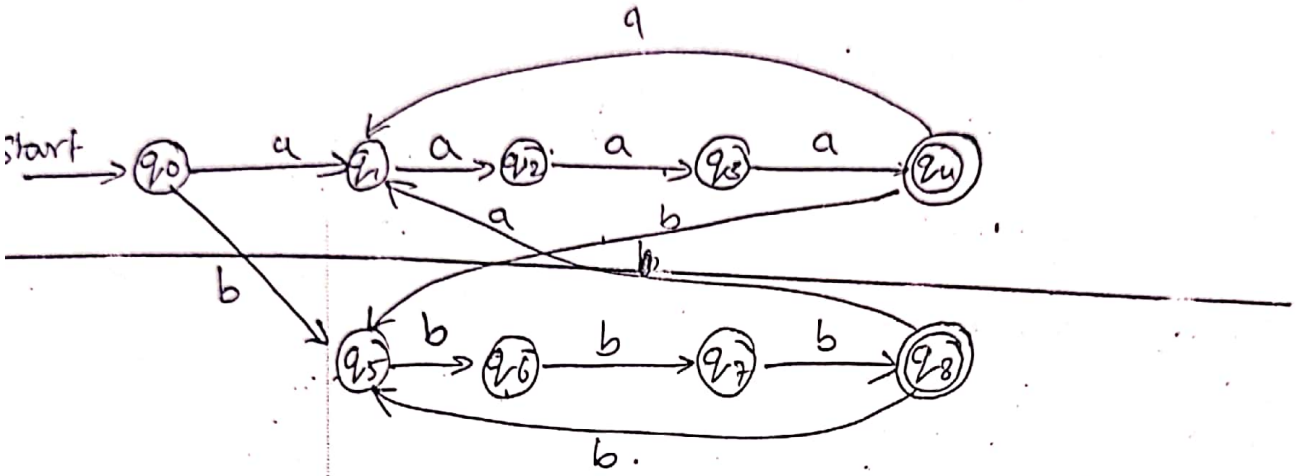
(19) $L = \{ w : n_a(w) = 2,\ n_b(w) >= 3 \}$

① aabbb baabb bbaab bbbaa.
ababb babab bbaba
abbab babba
abbba

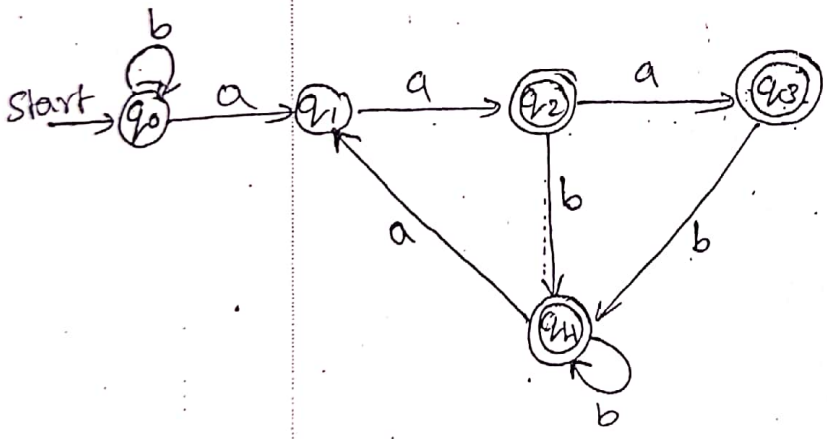


) $L = \{w : w$ contains no runs of length $< 4\}$ (8)

(a) 4 consecutive a's
(b) 4 consecutive bs
(c) combination of above: (a) & (b)



$L = \{w :$ every run of a's has length either 2 or 3$\}$



$L = \{w :$ w mod 5 = 0 , $\Sigma = \{0, 1\}\}$.

remainder : 0 1 2 3 4

	0	1
q_0	q_0	q_1
q_1	q_2	q_3
q_2	q_4	q_0
q_3	q_1	q_2

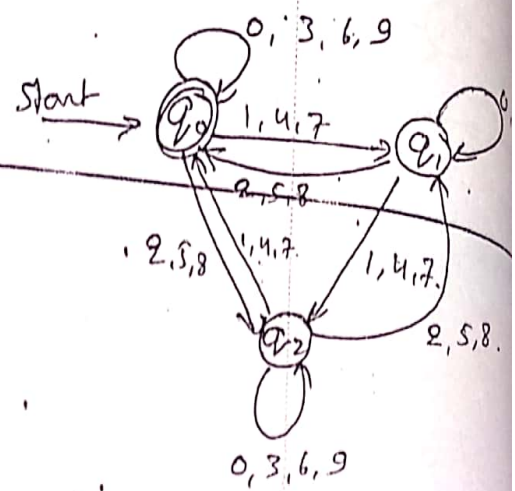
4) $L = \{w$: decimal strings divisible by 3$\}$.

	0	1	2	3	4	5	6	7	8	9
$\rightarrow q_0$	q_0	q_1	q_2	q_0	q_1	q_2	q_0	q_1	q_2	q_0
q_1	q_1	q_2	q_0	q_1	q_2	q_0	q_1	q_2	q_0	q_1
q_2	q_2	q_0	q_1	q_2	q_0	q_1	q_2	q_0	q_1	q_2

$\delta(q_0, (0, 3, 6, 9)) = q_0$

$\delta(q_0, (1, 4, 7)) = q_1$

$\delta(q_0, (2, 5, 8)) = q_2$



$\delta(q_1, (0, 3, 6, 9)) = q_1$

$\delta(q_1, (1, 4, 7)) = q_2$

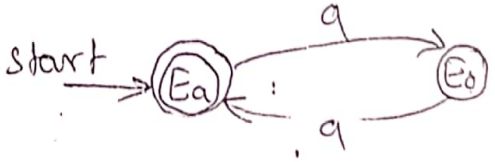
$\delta(q_1, (2, 5, 8)) = q_0$

$\delta(q_2, (0, 3, 6, 9)) = q_2$

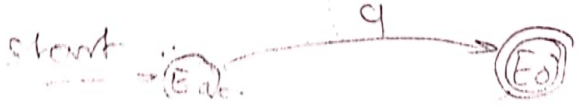
$\delta(q_2, (1, 4, 7)) = q_0$

$\delta(q_2, (2, 5, 8)) = q_1$

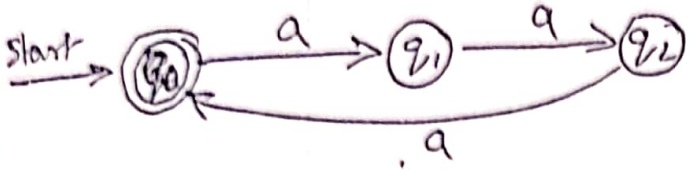
5) $L = \{w$ / even no of a's$\}$



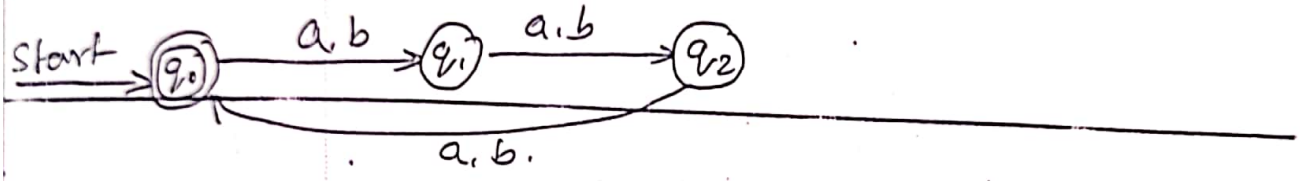
6) $L = \{w$ / odd no of a's$\}$



$L = \{ w : |w| \bmod 3 = 0 \}$ $\Sigma = \{a\}$



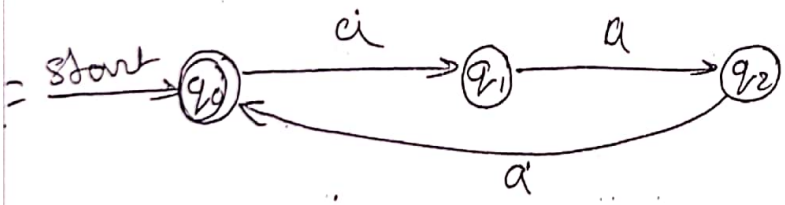
$L = \{ w : |w| \bmod 3 = 0 \}$ $\Sigma = \{a, b\}$



$L = \{w$ such that

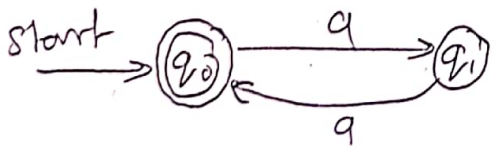
(a) $|w| \bmod 3 \geq |w| \bmod 2$, $\Sigma = \{a\}$

(b) $|w| \bmod 3 \neq |w| \bmod 2$, $\Sigma = \{a\}$



$|w| \bmod 3 = 0$

$Q_1 = \{q_0, q_1, q_2\}$
$\{0, 1, 2\}$



$|w| \bmod 2 = 0$

$Q_2 = \{q_0, q_1\}$
$\{0, 1\}$

$Q_1 \times Q_2 = \{(0\,0), (0\,1), (1\,0), (1\,1), (2\,0), (2\,1)\}$

$(00, a) = (\underset{M_1}{\delta(0,a)}, \underset{M_2}{\delta(0,a)})$

$\delta((1,1), a) \Rightarrow \delta(1,a), \delta(1,a)$

$\Rightarrow (2,0)$

$\delta((2,0), a) \Rightarrow (\delta(2,a), \delta(0,a))$

$\Rightarrow (0,1)$

$\delta((0,1), a) \Rightarrow (\delta(0,a), \delta(1,a))$

$\Rightarrow (1,0)$

$\delta((1,0), a) \Rightarrow (\delta(1,a), \delta(0,a))$

$(2,1)$

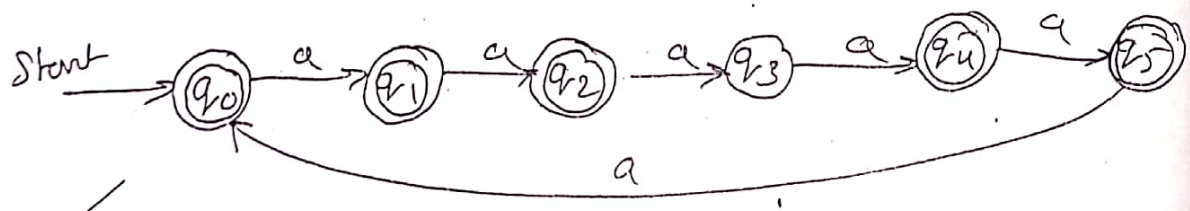
$\delta((2,1), a) \Rightarrow (\delta(2,a), \delta(1,a))$

$(0,0)$

a: $|w| \bmod 3 >= |w| \bmod 2$

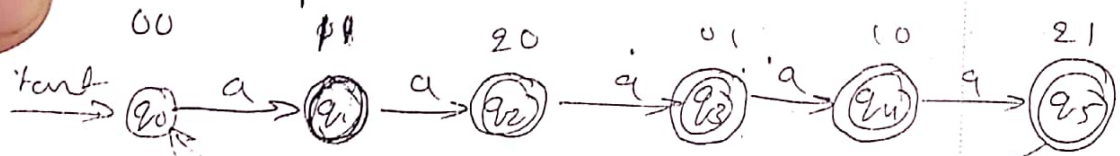
$x >= y$ will be final state

x	y
0	0
1	1
2	0
0	1
1	0
2	1



b: $|w| \bmod 3 \neq |w| \bmod 2$.

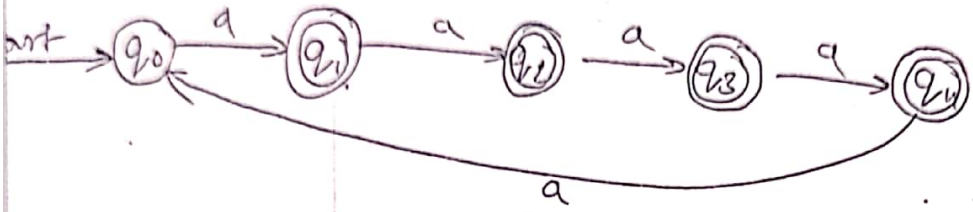
Final state will be $x \neq y$



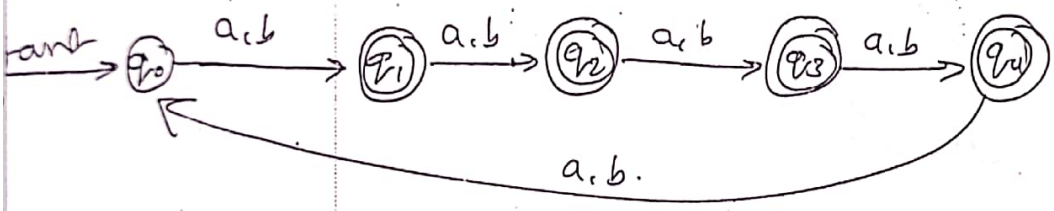
$L = \{ w : |w| \bmod 5 \neq 0 \}$ on $\Sigma = \{a\}$

no states = remainder for $k = 5$

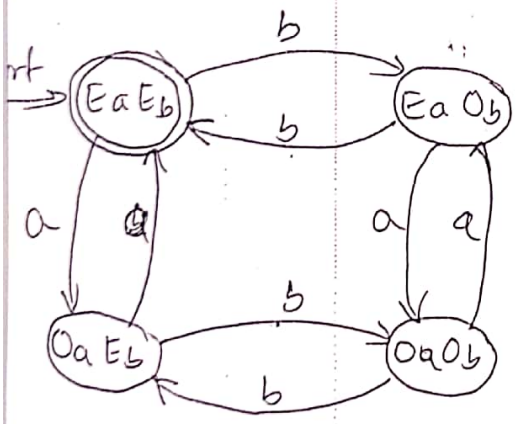
	0	1	2	3	4



$L = \{ w : |w| \bmod 5 \neq 0 \}$ on $\Sigma = \{a, b\}$

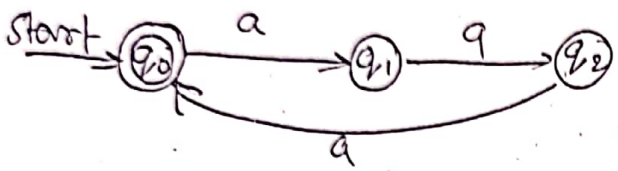


$L = \{ w \mid n_a(w)$ even and $n_b(w)$ odd $\}$



(32). $L = \{ w / w \in (a+b)^* \quad n_a(w) \bmod 3 = 0 \ \& \ n_b(w) \bmod 2$

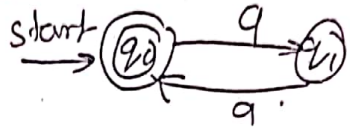
$n_a(w) \bmod 3 = 0$



$Q_A = \{A_0, A_1, A_2\}$

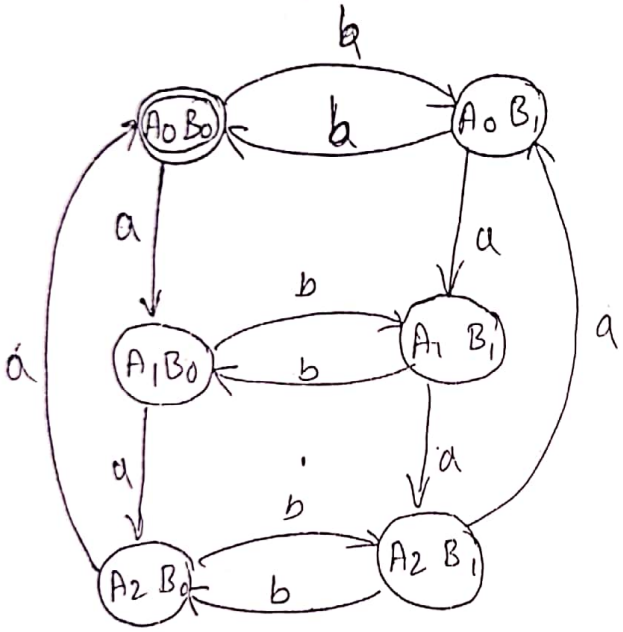
$n_b(w) \bmod 2$

B.

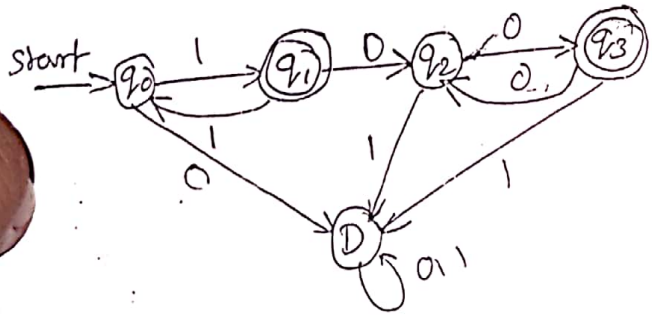


$Q_B = \{B_0, B_1\}$

$Q_A \times Q_B = \{(A_0 B_0), (A_0 B_1), (A_1 B_0), (A_1 B_1), (A_2 B_0), (A_2 B_1)\}$



3) $L = \{ w : w$ has odd no. 1's followed by even no. of 0's

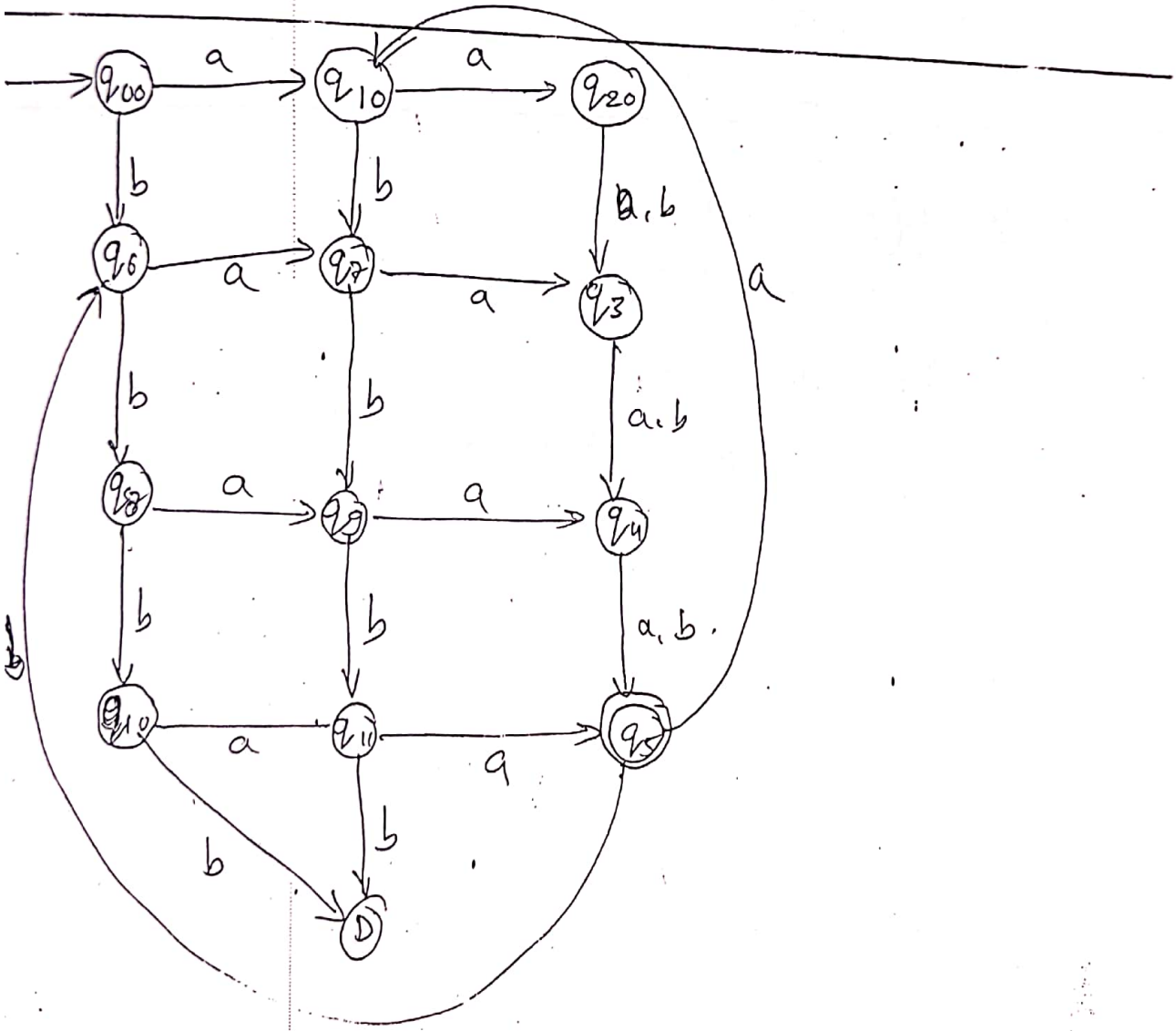


) $L = \{ w : n_a(w) \bmod 5 = 0 \ \& \ n_b(w) \bmod 3 = 0 \}$

$L = \{ w : n_a(w) \bmod 5 \neq n_b(w) \bmod 3 \}$.

7) obtain a DFA to accept strings of a's & b's such that each block of 5 consecutive symbols have at least two a's. (11)

aabbb baabb bbaab bbbaa
ababb babab bbaba
abbab bbbaa
abbba



Extended Transition function . proof.

$\hat{\delta}(q_0, w) = \delta(\hat{\delta}(q_0, x), a)$ To be proved

$\Rightarrow$ Let $w = xa$.

$\hat{\delta}(q_0, w) = P$.

$\hat{\delta}(q_0, xa) = P$.

$\hat{\delta}(q_0, x) = r$

Hence $\delta(r, a) = P$.

replace r now.

$\delta(\hat{\delta}(q_0, x), a) = P$.

Hence proved that $\hat{\delta}(q_0, w) = \delta(\hat{\delta}(q_0, x), a)$

Nondeterministic Finite Automata

NFA has the power to be in several states at once. His ability is often expressed as an ability to "guess" something about its inputs.

NFA accepts a language iff also accepted by DFA.

NFA can be converted to DFA & the DFA will have exponentially more states than NFA.

Informal view of Nondeterministic Finite Automata.

It is same as DFA but there is difference in δ function. The return value will be different.

Definition of NFA

$$A = (Q, \Sigma, \delta, q_0, F)$$

where Q - set of all states.

Σ - Alphabet (set of symbols)

q_0 - Start state $\in Q$.

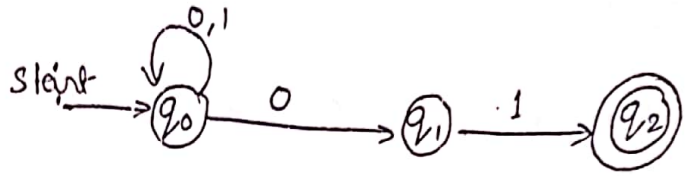
F - set of accepting states $\in Q$.

δ - $Q \times \Sigma \rightarrow \{$subset of $Q\}$.

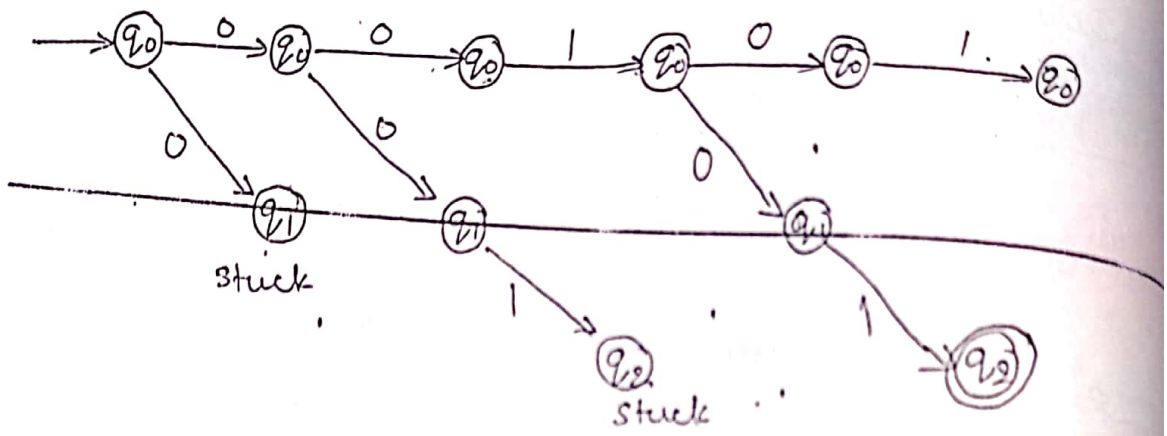
i.e. $Q \times \Sigma = 2^Q$

Accepts Q & Σ as parameter and returns zero, one, or more states.

Ex:- NFA accepting all strings that ends with



Let $w = 00101$



$A = \{Q, \Sigma, \delta, q_0, F\}$

$Q = \{q_0, q_1, q_2\}$

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

$q_0 \rightarrow q_0$

$F = \{q_2\}$

δ can be given as.

	0	1
→ q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	ϕ	$\{q_2\}$
* q_2	ϕ	ϕ

The Extended transition Function

$\hat{\delta}$ that takes a state q and a string of input symbols w, and returns the set of states that the NFA is in if it starts in state q and processes the string w.

$$\hat{\delta}(q, w) = \{p_i\}$$

Basis :- $\hat{\delta}(q_0, \epsilon) = \{q_0\}$

Induction :- Let $w = xa$

$$\hat{\delta}(q_0, x) = \{p_1, p_2 \ldots p_k\}$$

$$\bigcup_{i=1}^{k} \delta(p_i, a) = \{r_1, r_2 \ldots r_m\}$$

nguage accepted by a NFA

The language accepted by NFA in formally be efined as ⇒ Let $M = (Q, \Sigma, \delta, q_0, F)$ be an NFA, A string in accepted by the machine M, if it takes the initial tate q_0 to final state $\equiv$ $\hat{\delta}(q_0, w)$ is in F.

$$L(M) = \{w \mid w \in \Sigma^* \text{ and } \hat{\delta}(q_0, w) \in F\}$$

where Σ is any input alphabet.

Conversion of NFA to DFA

The NFA converted into DFA using two methods

1. Subset construction method
2. Lazy Evaluation method.

Subset construction method :-

Step 1 :- Identify the start state of DFA in $\{q_0\}$

Step 2 :- Identify the alphabet of DFA in Σ of N_L

Step 3 :- Identify the states of DFA in Q_D will have 2^n where n is the number of states in Q_N.

Let $Q_N = \{q_0, q_1, q_2\} = |Q_N| = 3$

$Q_D = \{\{\}, \{q_0\}, \{q_1\}, \{q_2\}, \{q_0, q_1\}, \{q_1, q_2\}, \{q$
$\{q_0, q_1, q_2\}\} = 8$ possible s

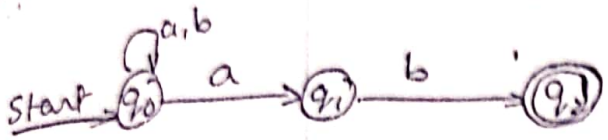
$$|Q_D| = 2^3 = 8$$

Step 4 :- Identify the final states of DFA: If $\{q_i, q_j \dots q_k\}$ a state in Q_D, then $\{q_i, q_j \dots q_k\}$ will be final s of DFA provided one of $q_i, q_j, \dots q_k \in F_N$ where in the set of final states of NFA.

Step 5 :- Identify the transition in δ_D of DFA : For each $\{q_i, q_j \dots q_k\}$ in Q_D and for each input symbol a in the transition can be obtained as

$$\delta_D(\{q_i, q_j, \dots q_k\}, a) = \delta_N(q_i, a) \cup \delta_N(q_j, a) \cup \dots \delta_N(q_k,$$

1) Obtain the DFA for the following NFA.



Step 1:- Start state of DFA = $\{q_0\}$

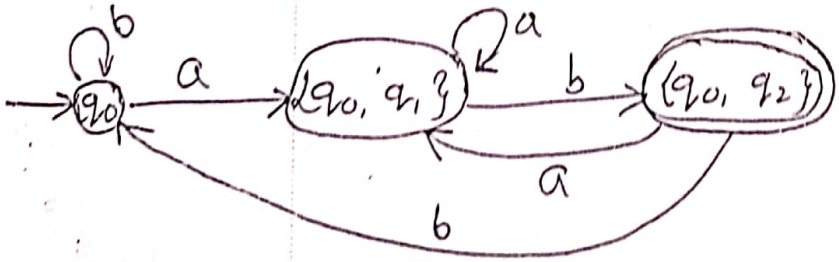
Step 2:- input alphabet $\Sigma = \{a, b\}$

Step 3:- $\delta_D = \{\phi, \{q_0\}, \{q_1\}, \{q_2\}, \{q_0, q_1\}, \{q_0, q_2\}, \{q_1, q_2\}, \{q_0, q_1, q_2\}\}$.

Step 4:- Transition Table

	a	b
ϕ	ϕ	ϕ
→ $\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_1\}$	ϕ	$\{q_2\}$
*$\{q_2\}$	ϕ	ϕ
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
*$\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$
*$\{q_1, q_2\}$	ϕ	$\{q_2\}$
*$\{q_0, q_1, q_2\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$

Transition Diagram



Ans

$= \{q_0, q_2\}$.

Lazy Evaluation method.

Let $N = \{Q_N, \Sigma, \delta_N, q_0, F_N\}$ which accepts language $L(N)$. If $D = \{Q_D, \Sigma, \delta_D, q_0, F_D\}$ is a obtained from the NFA N. then the language accepted by the DFA is $L(D) = L(N)$.

Algorithm :-

1. Identify starting state ~~of~~ DFA ~~is~~ $\{q_1\}$.
2. Identify the input alphabet of DFA is Σ
3. Compute δ_D as follows.

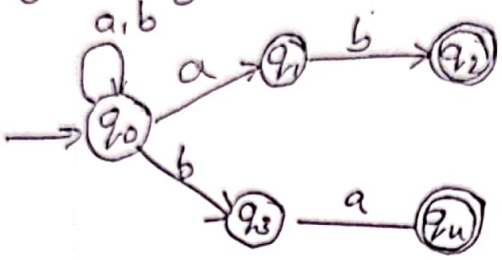
$$\delta_D(\{q_i, q_j \cdots q_k\}, a) = \delta_N(q_i, a) \cup \delta_N(q_j, a) \cdots \delta_N($$
$$= \{r_1, r_2 \cdots r_n\}$$

Add the state $\{r_1, r_2 \cdots r_n\}$ into the set Q_D if is already not there in the Q_D.
If it is already a member of Q_D, then not h

4. Identifying final state as follows.
the $\{q_i, q_j \cdots q_k\}$ is said to be final state iff q_i or q_j or $\cdots q_k \in F_N$.

Q:- Draw a NFA to accept strings of a's and b's ending with ab or ba. and convert it into DFA using Lazy evaluation method.

Sol:- NFA

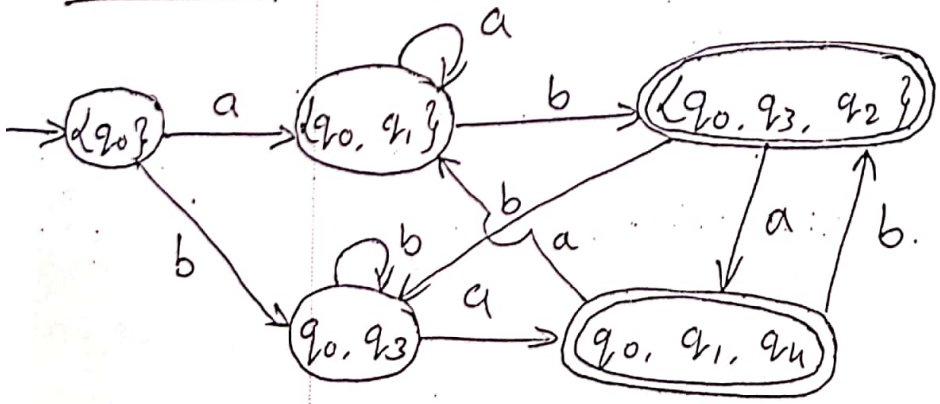


Conversion to DFA.

1) Start state for DFA = $\{q_0\}$
2) Input alphabets $\Sigma = \{a, b\}$

	a	b
→ $\{q_0\}$	$\{q_0, q_1\}$	$\{q_0, q_3\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_3, q_2\}$
$\{q_0, q_3\}$	$\{q_0, q_1, q_4\}$	$\{q_0, q_3\}$
$q_2, q_3\}$	$\{q_0, q_1 q_4\}$	$\{q_0, q_3\}$
$q_1, q_4\}$	$\{q_0, q_1\}$	$\{q_0, q_3, q_2\}$

Transition Diagram.



Theorem 2.11:

If $D = \{Q_D, \Sigma, \delta_D, \{q_0\}, F_D\}$ is the DFA constr
from NFA $N = \{Q_N, \Sigma, \delta_N, q_0, F_N\}$ by the subset cons
then the $L(D) = L(N)$.

Proof:- we have to first prove that.

$$\hat{\delta}_D(\{q_0\}, w) = \hat{\delta}_N(q_0, w).$$

Basis:- Let $|w| = 0$; i.e. $w = \epsilon$. By the basis
of $\hat{\delta}$ for DFA's and NFA's, both $\hat{\delta}_D(\{q_0\}$
and $\hat{\delta}_N(q_0, \epsilon) = \{q_0\}$.

Induction:-

By the Inductive hypothesis,

$\hat{\delta}_D(\{q_0\}, x) = \hat{\delta}_N(q_0, x)$ where $w = xa$.

Let both these sets of N's states be $\{p_1, p_2 \ldots p$

The inductive part of the definition of $\hat{\delta}$ for N
tells us that

$$\hat{\delta}_N(q_0, w) = \bigcup_{i=1}^{k} \delta_N(p_i, a) \quad \text{—(1)}$$

The subset construction tells us that.

$$\delta_D(\{p_1, p_2 \ldots p_k\}, a) = \bigcup_{i=1}^{k} \delta_N(p_i, a) \quad \text{—(2)}$$

According Inductive part of $\hat{\delta}$ of DFA, we h

$$\hat{\delta}_D(\{q_0\}, w) = \delta_D(\hat{\delta}_D(\{q_0\}, x), a) = \delta_D(\{p_1, p_2, \ldots p_k\}, a)$$

$$= \bigcup_{i=1}^{k} \delta_N(p_i, a) \quad (3)$$

Finite Automata with Epsilon-Transition

This is the another extension of the finite automaton The new feature added is, transition on ϵ is allowed ϵ-NFA's are closely related to RE. and useful in proving the equivalence between the classes of languages accepted by FA & by RE.

Definition :- The ϵ-NFA is 5 tuple or quintuple indicative five components ie $M = (Q, \Sigma, \delta, q_0, F)$

where Q is non-empty, finite set of states

Σ is non-empty, finite set of input symbols.

$\delta : Q \times (\Sigma \cup \epsilon)$ to 2^Q is Based on the current state there can be a transition to other states with or without any input symbols

$q_0 \in Q \rightarrow$ is the start state

$F \subseteq Q$ - is set of accepting or final states.

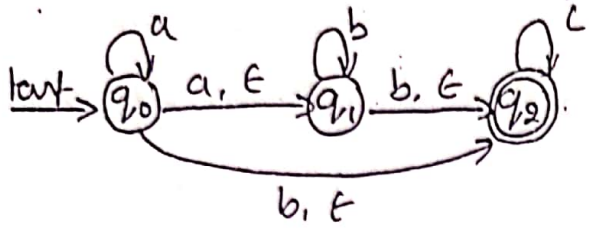
ϵ-CLOSURE :-

The ϵ-CLOSURE of q denoted by ECLOSE(q) is a set of all states which are reachable from q on transition only. It can be recursively defined as

state q is in ECLOSE(q) if ECOSE(q)^i = q.

If ECLOSE(q) contains p and if there is a transition from state p to state r labeled ϵ, then state r is also in ECLOSE(q).

The ϵ-CLOSURE(q) for each $q \in Q$ in shown below.



⇒ The states q_0, q_1, and q_2 are reachable from q_0 giving any input, so. ECLOSE(q_0) = { q_0, q_1, q_2 }

⇒ The states q_1 and q_2 are reachable from q_1 without any input. so, ECLOSE(q_1) = { q_1, q_2 }

⇒ The state q_2 is reachable from q_2 without giving a

so, ECLOSE(q_2) = { q_2 }

Extended Transition Function of ϵ-NFA to strings

The extended transition function $\hat{\delta}$ describes wh appens to a state of machine when the input i string (sequence of symbols). Let M = $(Q, \Sigma, \delta, q_0, F)$ in ϵ-NFA. $\hat{\delta}$: $Q \times (\Sigma \cup \epsilon)^*$ to 2^Q in defined recursi is shown below:

Basis :- $\hat{\delta}(q, \epsilon)$ = ECLOSE(q).

Induction :- Let $w = xa$

$\hat{\delta}(q, x) = \{ p_1, p_2 \dots p_m \}$

Let the transition from $\{ p_1, p_2 \dots p_n \}$ on a

$\delta(\{ p_1, p_2 \dots p_m \}, a) = \delta(p_1, a) \cup \delta(p_2, a) \dots \delta(p_m$

$= r_1, r_2 \dots r_m$.

Now $\hat{\delta}(q, w)$ = ECLOSE($r_1, r_2, \dots r_m$)

= ECLOSE(r_1) ECLOSE(r_2) ... ECLOSE(

Thus, various properties of ETF for an ϵ-NFA can be

$\Rightarrow \hat{\delta}(q, \epsilon) = \text{ECLOSE}(q)$.

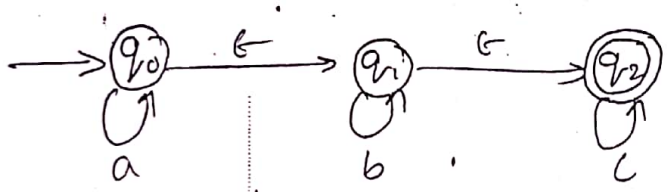
$\Rightarrow \hat{\delta}(q, w) = \hat{\delta}(q, xa) = \text{ECLOSE}\left(\delta\left(\hat{\delta}(q, x), a\right)\right)$

where $w = xa$.

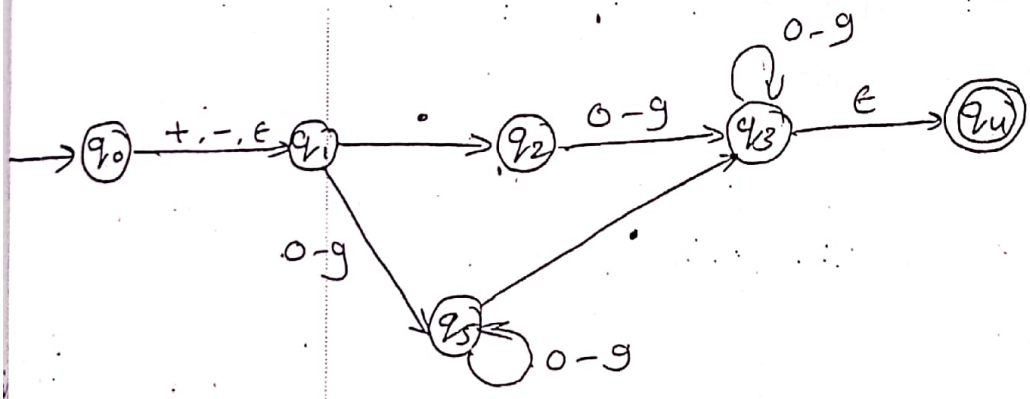
$\Rightarrow \hat{\delta}(q, w) = \hat{\delta}(q, ax) = \text{ECLOSE}\left(\delta\left(\hat{\delta}(q, a), x\right)\right)$

where $w = ax$.

Obtain an ϵ-NFA which accepts strings consisting of zero or more a's followed by zero or more b's followed by zero or more c's.



Obtain an NFA with ϵ-transition to accept decimal numbers.



Conversion from ε-NFA to DFA

Let $M_E = (Q_E, \Sigma, \delta_E, q_0, F_E)$ be an ϵ-NFA

then the equivalent DFA $M_D = (Q_D, \Sigma, \delta_D, q_{0D}, F_D)$

obtained as follows.

step 1:- If q_0 is the start state of NFA, then ECLOSE(q_0) is the start state of DFA.

ie q_{0D} = ECLOSE(q_0)

---

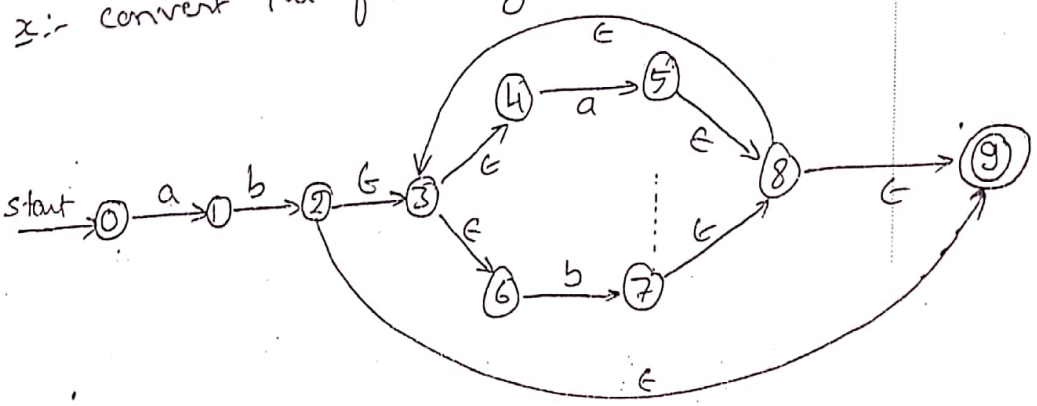
step 2:- Compute the transition for DFA. Let $\{q_i, q_j \cdots q_k\}$ is a state in DFA. then $\delta_D(\{q_i, q_j \cdots q_k\}, a)$ is as follows.

* Let $\delta_E(\{q_i, q_j \cdots q_k\}, a) = \{p_1, p_2 \cdots p_m\}$
* Then take ECLOSE($\{p_1, p_2 \cdots p_m\}$)

Then, $\delta_D(\{q_i, q_j \cdots q_k\}, a)$ = ECLOSE($\delta_E(\{q_i, q_j \cdots q_k\}$,

step 3:- If $\{q_i, q_j \cdots q_k\}$ is a state in DFA and this set set contains at least one final state of ϵ-NFA then $\{q_i, q_j \cdots q_k\}$ is the final state

Ex:- convert the following ϵ-NFA into DFA.



step 1:- $q_{0D} = ECLOSE(q_0) = ECLOSE(0)$

$= \{0\}$.

step 2:- First identify the ECLOSE of all states.

ECLOSE(0) = {0}

ECLOSE(1) = {1}

ECLOSE(2) = {2, 3, 4, 6, 9}

~~ECLOSE(3) = {3, 4, 6}~~

ECLOSE(4) = {4}

ECLOSE(5) = {5, 8, 9, 3, 4, 6}

ECLOSE(6) = {6}

ECLOSE(7) = {7, 8, 9, 3, 4, 6}.

ECLOSE(8) = {8, 3, 4, 6 9}.

ECLOSE(9) = {9}.

Equivalence and minimization of Finite Automata

If the two automata accept the same language, we can minimize the automata that have an few states as possible. minimization of FA is very important in the design of switching circuits. This is because, as the no of states of the automaton implimented by the circuit decreases, the cost of the circuit tends to decrease.

Equivalence of Two states.

Two states p and q of a DFA are equivalent (indistinguishable) if and only if $\delta(p, w)$ and $\delta(q, w)$ are final states or both are non final states. for all $w \in \Sigma^*$

i.e $\delta(p, w) \in F$ and $\delta(q, w) \in F$

$\delta(p, w) \notin F$ and $\delta(q, w) \notin F$

If there is atleast one string w such that one of $\delta(p, w)$ and $\delta(q, w)$ is final state and the other is non-final state, the states p and q are not equivalent and are called distinguishable states.

ii $\delta(p, w) \in F$ and $\delta(q, w) \notin F$

$\delta(p, w) \notin F$ and $\delta(q, w) \in F$

The distinguishable and indistinguishable states can be obtained using table-filling algorithm (also known as mark procedure).

Table filling Algorithm. (mark procedure)

~~step 1:- Identify the initial marking:~~

For each pair (P, q) where $p \in Q$ and $q \in Q$ and $q \notin F$ or vice versa then, the pair (P, q) is distinguishable and mark the pair (P, q) say with 'x'.

step2 :- Identify the subsequent marking :-

~~For each pair (P, q)~~ and for each $a \in \Sigma$, if $\delta(P, a) = r$ and $\delta(q, a) = s$. If the pair (r, s) is already marked as distinguishable then the pair (P, q) is also distinguishable and mark it as say 'x'. Repeat until no previously unmarked pairs are marked.

Minimization of DFA (procedure or Algorithm)

step 1 :- Find the distinguishable and indistinguishable using table filling algorithm.

step 2 :- Obtain the states of minimized DFA :

These groups consist of indistinguishable pairs obtained in step 1 and individual distinguishable states.

step 3 :- Compute the transition table :

If $[P_1, P_2 \ldots P_k]$ is a group and if $\delta([P_1, P_2 \ldots P_k], a) = [r_1, r_2 \ldots r_m]$ then place an edge from $[P_1, P_2 \ldots$ to $[r_1, r_2 \ldots r_m]$, labeled a. Follow this step for each group obtained in step 2 and for each symbol $a \in \Sigma$

Step 5:- Identify the final state:

If the group $[P_1, P_2 \dots P_k]$ contains a final state of given DFA then the group $[P_1, P_2 \dots P_k]$ is final state of minimized DFA.

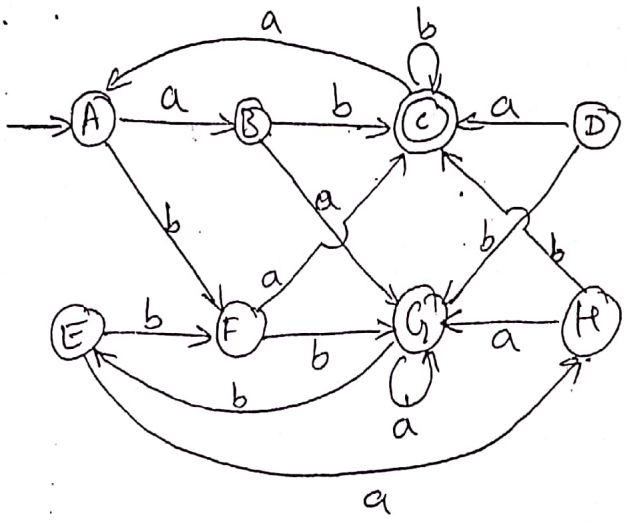
Step 4:- Identify the start state:-

If one of the component in the group $[P_1, P_2 \dots P_k]$ consists of a start state of given DFA then:

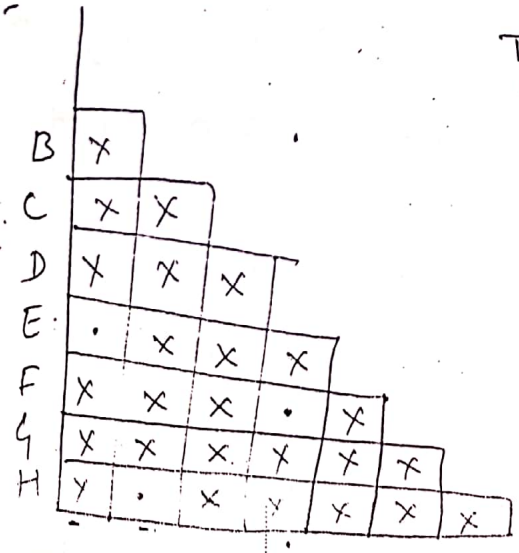
$[P_1, P_2 \dots P_k]$ is the start state of DFA.

1:- Minimize the following DFA

	a	b
→ A	B	F
B	G	C
* C	A	C
D	C	G
E	H	F
F	C	G
G	G	E
H	G	C



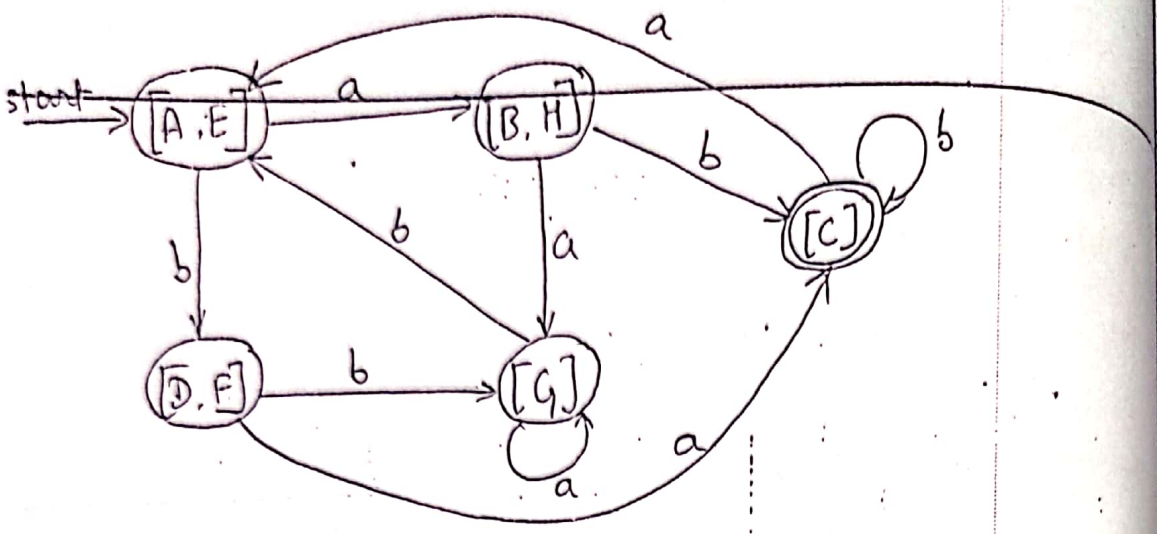
:-



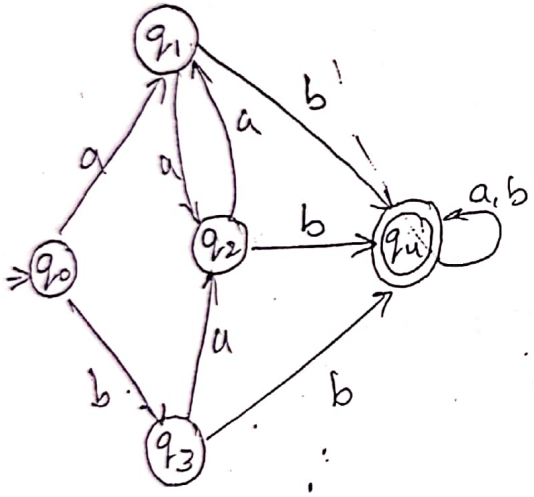
The equivalent states are [A, E], [B, H] and [D, F].

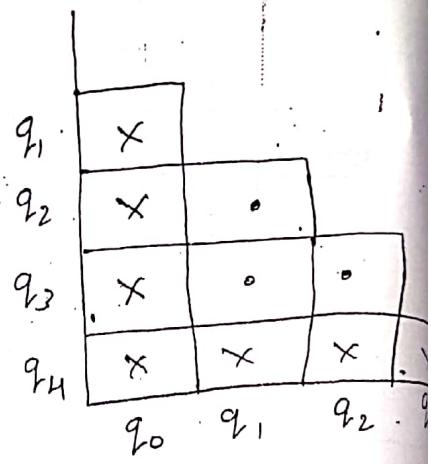
Hence the the states for minimized DFA will be (A, E), (B, H) and (D, F) and (C) (G)

Final minimized DFA is



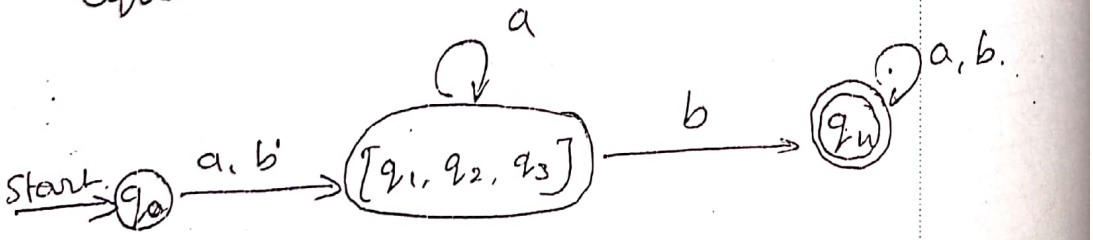
Ex 2:- minimize the following DFA





$[q_1, q_2]$, $[q_1, q_3]$ and $[q_2, q_3]$ are equiva

According to transitive rule $[q_1, q_2, q_3]$ equivalent. Hence final minimized DF



3:- Find the minimized DFA for the following.

	a	b
→A	B	A
B	A	C
C	D	B
*D	D	A
E	D	F
F	G	E
G	F	G
H	G	D

[A B C D E F G H]

↓ ε

[A B C E F G H] [D]

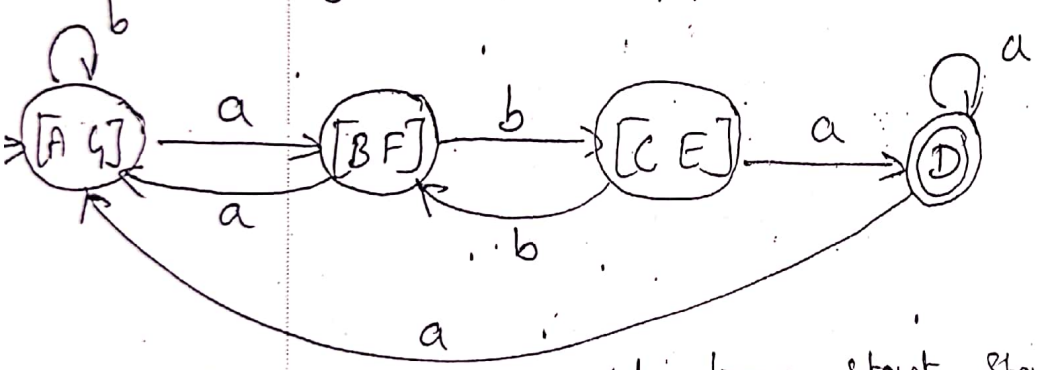
↓ a

[A B F G H] [C E] [D]

↓ b

[A G] [C E] [B F] [H] [D]

Final minimized DFA is.



Since [H] is not reachable from start state, it can be removed.

esign DFA for the following

~~to accept strings of a's having at least one 'a'~~

to accept strings of a's and b's having at least one 'a'

to accept strings of a's and b's having exactly one 'a'

to accept strings of a's and b's starting with substring ab.

to accept strings of a's and b's ending with substring abb.

~~to accept strings of a's and b's not ending with abb.~~

to accept strings of a's and b's having substring aab.

to accept strings of a's and b's except those having aab.

o accept strings of 0's and 1's having three consecutive 0's

$L = \{ awa \mid w \in (a+b)^n \text{ where } n \geq 0 \}$

to accept strings of a's and b's ending with ab or ba.

to accept strings of a's and b's having 4 a's

to accept strings of 0's, 1's and 2's biginning with a '0' followed by odd number of 1's and ending with a 2.

to accept strings of a's and b's with at most two consecutive b's

to accept strings of 0's & 1's starting with at least two 0's nd ending with at least two 1's

$L = \{ wbab \mid w \in \{a,b\}^* \}$

trings of a's and b's having not more than three a's

trings that either begins or ends or both with substring 01

$L = \{ w : n_a(w) \geq 1, \; n_b(w) = 2 \}$.

$L = \{ w : n_a(w) = 2, \; n_b(w) >= 3 \}$.

UNIT - II

Regular Expressions

Algebraic instructions of regular languages are called as RE.

Regular expressions are defined by induction as follows.

Basis step :-

① The constants : ϵ & ϕ represents RE denoting the languages $\{\epsilon\}$ and ϕ.

i) $\epsilon = \{\epsilon\}$ $\quad L(\epsilon) = \{\epsilon\}$

$\phi = \phi$ $\quad L(\phi) = \phi$.

② If 'a' is any symbol then a is RE

ii) $a = \{a\}$ $\quad L(a) = \{a\}$

Induction step :-

① If E & F are RE then E+F is RE denoting

$L(E+F) = L(E) \cup L(F)$

② If E and F are RE then E.F is RE denoting

$L(E.F) = L(E).L(F)$.

③ If E is RE then E^* is RE which represents the language which is closure of $(L(E))^*$.

④ If E is RE then (E) is RE denoting same lang

$(E) \rightarrow L(E) \rightarrow L((E)) = L(E)$

Ex:- $a^* \rightarrow \{\epsilon, a, aa \ldots\}$

$a^+ \rightarrow \{a, aa, aaa \ldots\}$

operators of RE and precedence of RE

( ) : grouping

* : star closure

. : concatenation

\+ : union/or

Exercises

write the Regular expression for the following

a) The set of strings over $\Sigma = \{a, b, c\}$ containing at one a & atleast one b.

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

b) The set of strings of 0's & 1's whose tenth symbol from the right end is 1

$$(0+1)^* 1 (0+1)^9$$

c) The set of strings of 0's & 1's with at most one pair of consecutive 1's.

$$(0+10)^* + 1(0+10)^* + (\epsilon+1)(0+01)^*$$

1) The set of all strings of 0's and 1's such that every pair of adjacent 0's appears before any pair of adjacent 1's

$(0+10)^*(1+01)^*(0+\epsilon)$ $(0011+01+1)^*(0+\epsilon)$

2) The set of strings of 0's & 1's whose no. of 0's is divisible by 5

$(1^*01^*01^*01^*01^*01^*)^*$

3) strings over $\Sigma = \{0,1\}$, not containing 101 as substring.

$0^*(1^*000^*)^*1^*0^*$

4) strings with an equal no 0's & 1's such that no prefix has two more 0's than 1's, nor two more 1's than 0's.

$(01)^* + (10)^*$

strings of 0's & 1's without any consecutive 1's

$0^*(100^*)^*(1+\epsilon)$ or $(1+\epsilon)(00^*1)^*0^*$

strings of 0's and 1's ending with any no of 1's.

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

strings of a's and b's whose length is 2.

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

* odd strings represented in binary.

$(0+1)^*1$

* Even strings represented in binary

$(0+1)^*0$

* $L = \{ w \mid |w| \bmod 3 = |w| \bmod 2 , w \in \}$

$\Rightarrow ((a+b)^6)^* + (a+b)((a+b)^6)^*$.

* strings ending with ab or abb.

$\Rightarrow (a+b)^*(ab+abb)$

* strings having substring abaa.

$\Rightarrow (a+b)^* abaa (a+b)^*$.

* strings beginning with aab or bbb.

$\Rightarrow (aab+bbb)(a+b)^*$.

*

Write the RE for the following Languages.

$L = \{ w \mid w \in \{a,b\}^2 \}$

$aa + ab + bb + ba$ or $(a+b)(a+b)$

$L = \{ w \mid |w| \leq 2, \ w \in \{a,b\} \}$

$\epsilon + a + b + aa + ab + bb + ba$ or $(\epsilon + a + b)^2$

$L = \{ w \mid |w| \leq 10, \ w \in \Sigma, \ \Sigma = \{a,b\} \}$

$(\epsilon + a + b)^{10}$

$L = \{ w \mid |w| \bmod 2 = 0, \ w \in \Sigma, \ \Sigma = \{a,b\} \}$

$(aa + ab + bb + ba)^*$ or $((a+b)(a+b))^*$

$L = \{ w \mid |w| \bmod 2 = 1, \ w \in \Sigma, \ \Sigma = \{a,b\} \}$

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

$L = \{ w \mid w \in \{0,1\}^* \}$ with at least 3 consecutive zeros

$(0+1)^*000(0+1)^*$

$L = \{ w \mid w \in \{0,1\}^* \}$ having no two consecutive zeros

$(01+1)^*(0+\epsilon)$ or $(0+\epsilon)(10+1)^*$

8) $L = \{ w \mid aw \in \{a,b\}^*$, starting with a, end

~~$L = \{ w \mid awb$ where $w \in \{a,b\}^* \}$~~

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

9) second symbol from right is a.

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

10) Third symbol from ~~right~~ is a.

$$(a+b)^* a (a+b)(a+b) \text{ or } (a+b)^* a (a+b)^2$$

11) ~~Tend~~ Symbol from right is a

$$(a+b)^* a (a+b)^9$$

12) Third symbol is a and 4th symbol is b from

$$(a+b)^* ba (a+b)^2$$

13) Begins and ends with same letters.

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

14) ~~Length~~ either even or multiples of three.

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

or

$$((a+b)^2)^* + ((a+b)^3)^*.$$

Every block of four consecutive symbols contains at ~~least two a's~~

aabb baab bbaa
abab baba
abba b

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

$L = \{a^n b^m \mid m+n \text{ is even}\}$

case 1: if n and m both are even then n+m is even.

$(aa)^*(bb)^*$.

case 2: if n and m both are odd then also n+m is even.

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

So Final RE will be

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

$L = \{a^n b^m \mid m \geq 1, n \geq 1, nm \geq 3\}$

case 1:- $m = 1$, $n \geq 3$, $nm \geq 3$

$aaaa^*b$.

case 2 - $m \geq 3$, $n = 1$ $nm \geq 3$

$abbbb^*$.

Case 3 :- if $n \geq 2$, $m \geq 2$, $nm \geq 3$.

~~aaa* bbb*~~

Final RE will be.

$$aaaa^*b + abbbb^* + aaa^*bbb^*$$

(8) $L = \{a^{2n} b^{2m} \mid n \geq 0, m \geq 0\}$.

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

(9) strings containing not more than three a's.

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

(10) $L = \{a^n b^m : n \geq 4, m \leq 3\}$:

$$aaaaa^*(\epsilon + b)(\epsilon + b)(\epsilon + b)$$

or

$$a^4 a^* (\epsilon + b)^3$$

) $|w|$ multiple of three, $w \in \{a,b\}^*$.

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

) $L = \{w : n_a \bmod 3 = 0, w \in \{a,b\}^*\}$

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

$b^*(aaa)^* b^* (aaa)^* b^* (aaa)^* b^*$

) strings that do not end with 01. over $\{0,1\}^*$.

$$(0+1)^*(00 + 10 + 11)$$

4) $L = \{vuv : u, v \in \{a,b\}^* \text{ and } |v| = 2\}$.

Possible substrings of length 2 will be

aa ab bb ba.

$$aa(a+b)^*aa + ab(a+b)^*ab + bb(a+b)^*bb + ba(a+b)^*ba$$

5) $L = \{w : \text{strings ends with ab or ba}, w \in \{a,b\}^*\}$.

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

strings having at most one pair of consecutive 0's

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

6) at least one a, at least one b, $\Sigma = \{a, b, c\}$.

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

3) strings of 0's and 1's with at most one pair of consecutive 0's

The string containing at most 2 0's will have the following 3 cases:

case 1 :- No 0's, so, any no of 1's i.e. 1^*.

case 2 :- A zero preceded by any combination of 1's & 01's and followed by any combination of 1's & 10's

i.e. $(1+01)^*0(1+10)^*$.

case 3 : Two consecutive 0's preceded by any c
of 1's and 0's and followed by any c
of 1's and 0's

∴ $(1+01)^*00(1+10)^*$

Final RE on combining all three above RE

$R = 1^* + (1+01)^*0(1+10)^* + (1+01)^*00($

~~Strings of a's & b's with alternate a's and b's~~

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

Strings beginning with zero's and followed
odd no of 1's.

$00^*(11)^*1$.

Strings having at most two consecutive b's

$R = a^* + (a+ba)^*b(a+ab)^* + (a+ba)^*bb(a+$

Strings having $n_a = 2$ and $n_b \geq 3$

aabbb baabb. bbaab bbbaa
ababb. babab bbaba
abbab. babba
abbba

$RE = aabbb^* + abb^*abbb^* + abbb^*abb^* + abbbb^*$
$+ bb^*abbb^* + bb^*abb^*abb^* + bb^*abbb^*a + bbb^*aabb$
$+ bbb^*abb^*a + bbbb^*aa$

To obtain ε-NFA from RE

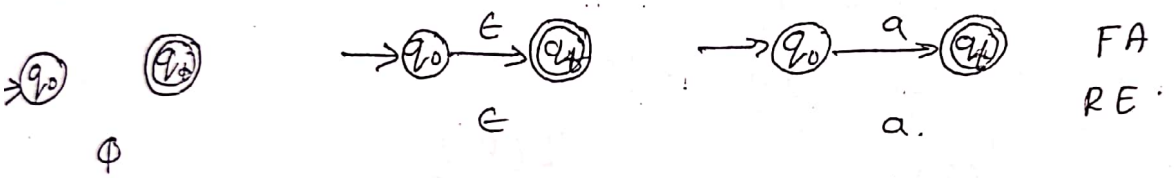
Theorem :-

Let R be a RE. Then there exists a finite automaton $M = (Q, \Sigma, \delta, q_0, F)$ which accepts $L(R)$.

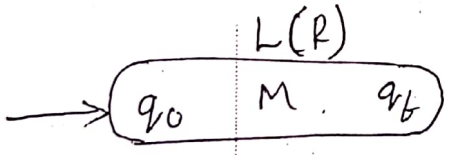
or

prove that there exists a FA to accept the language $L(R)$ corresponding to the RE R.

Proof :- By definition, ϕ, ϵ and a are RE. So, the corresponding machines to recognize the lang for the respective expressions are shown below.



The schematic representation of a RE R to accept the language $L(R)$ is given as below.

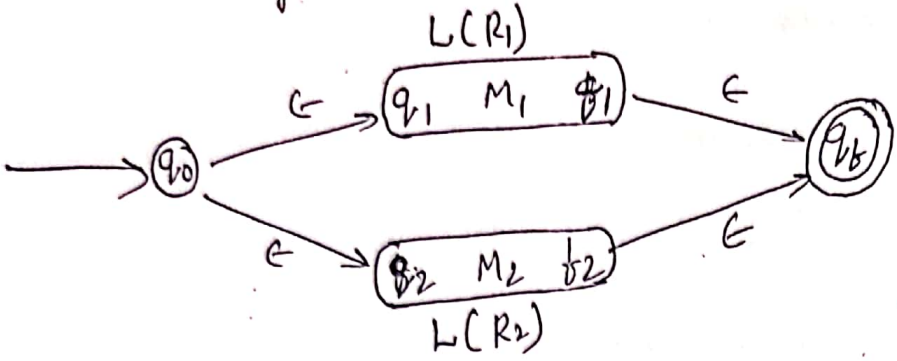


Let $M_1 = (Q_1, \Sigma_1, \delta_1, q_1, f_1)$ be a machine which accep
e lang $L(R_1)$ corresponding to the RE R_1.

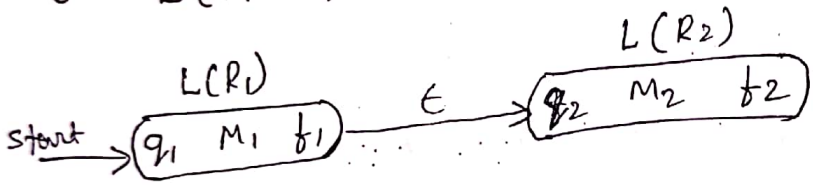
Let $M_2 = (Q_2, \Sigma_2, \delta_2, q_2, f_2)$ be a machine which accept
lang $L(R_2)$ corresponding to the RE R_2.

Then the various machines corresponding to the RE R_1+R_2, R_1R_2 and R^* are show below in three cases.

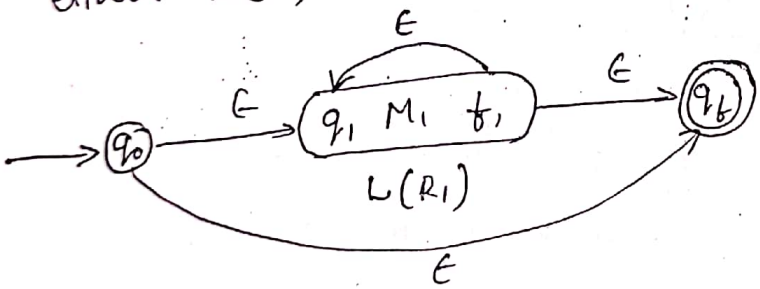
Case 1 : $R = R_1 + R_2$. we can construct an ϵ-NFA accepts either $L(R_1)$ or $L(R_2)$ which can represented as $L(R_1 + R_2)$ as below.



Case 2 : $R = R_1 . R_2$. we can construct an ϵ-NFA which accepts $L(R_1)$ followed by $L(R_2)$ which can be represented as $L(R_1 . R_2)$ as below.

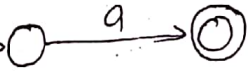


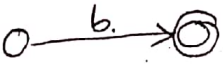
Case 3 :- $R = (R_1)^*$ we can construct an ϵ-NFA which accepts either $L(R_1^*)$ or $L(R_1)^*$.



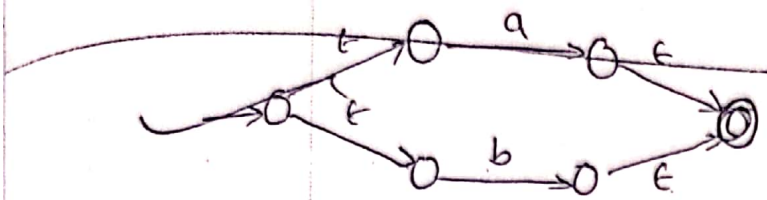
; obtain an ϵ-NFA which accepts strings of a's and starting with ab.

Sol : The RE is $ab(a+b)^*$

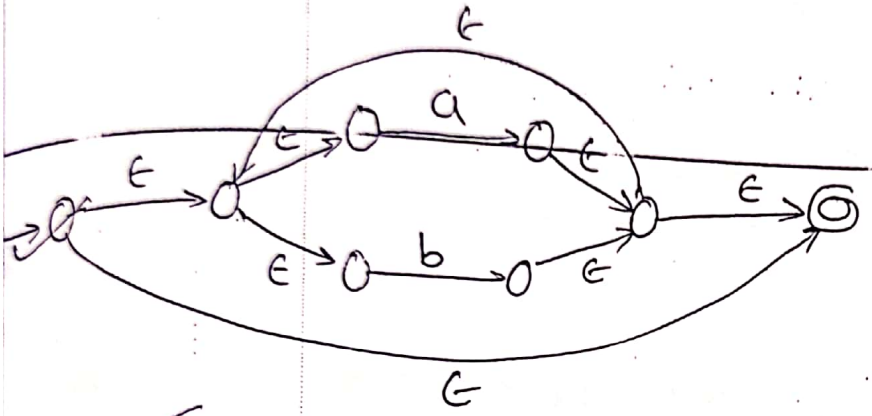
step 1 :- m/c to accept a 

step 2 :- m/c to accept b 

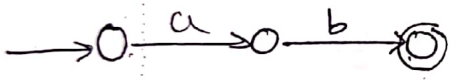
step3 :- m/c to accept a+b

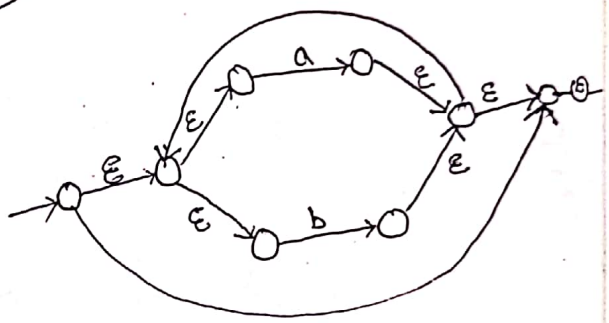


step 4 :- m/c to accept (a+b)*

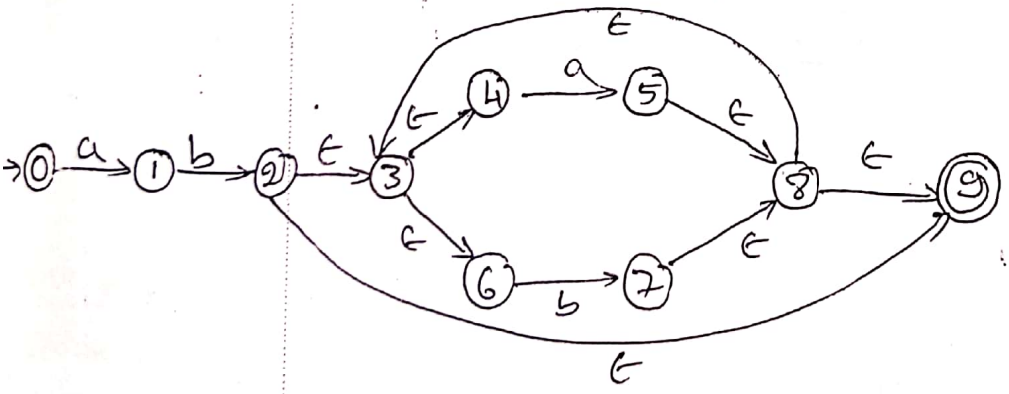


step5 :- m/c to accept ab



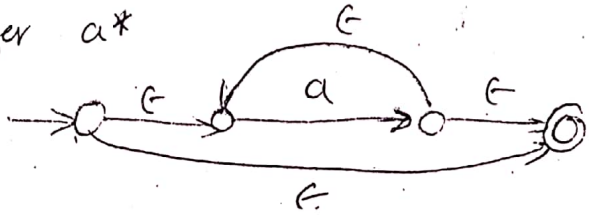


step6 :- Now combine all the above machines to accept ab(a+b)* as shown below.

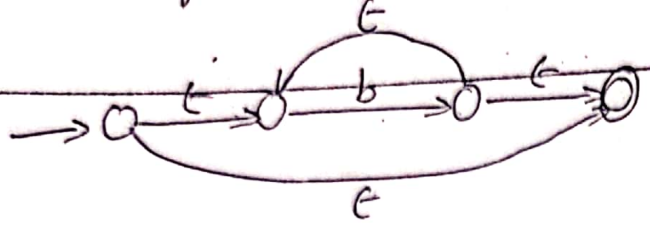


obtain an ε-NFA for the RE a* + b* + c*

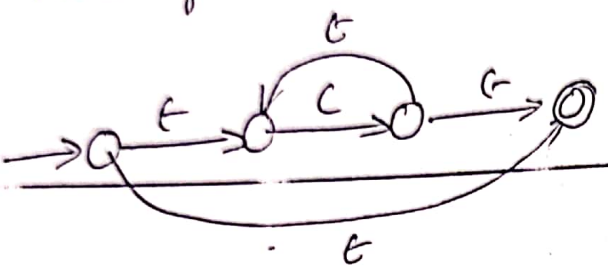
step1 :- m/c for a*



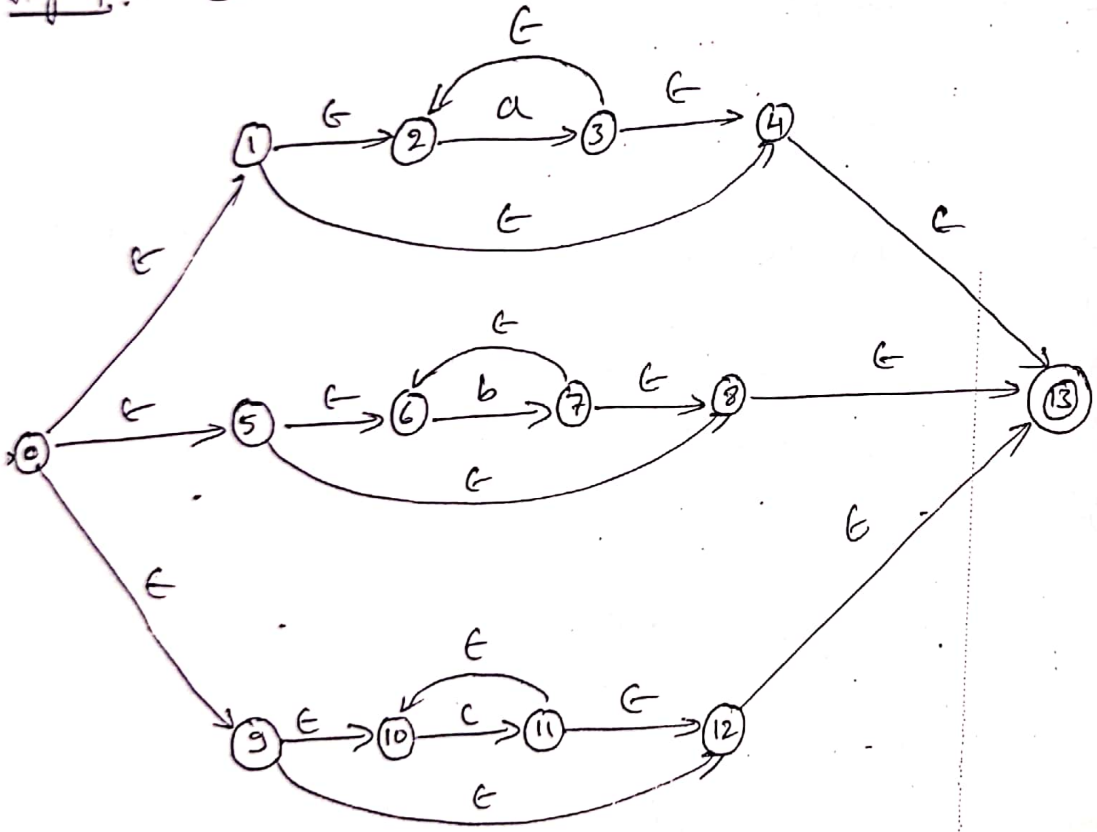
step2:- m/c for b*



step3:- m/c for c*



step4:- combine all the above m/cs

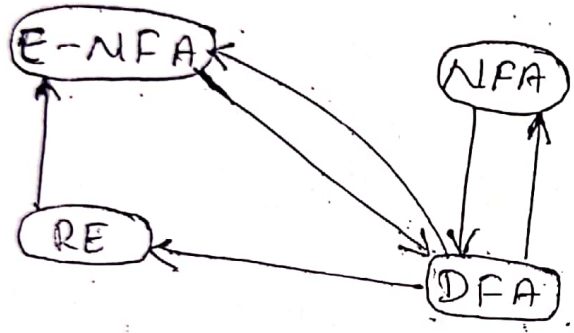


Finite Automata and Regular Expressions

FA and RE are the two notations used to represent exactly the same set of languages. In order to show that the REs define the same class of lang, we must show that.

① Every lang defined by one of the automata is also defined by RE

② Every lang defined by RE is defined by one of the automata.



Plan for showing the equivalence of four diff notations for Regular languages.

DFA's to RE :-

Theorem:- If $L = L(A)$ for some DFA A, then there is a RE R such that $L = L(R)$.

Proof:- Let the states of A are $\{1, 2, \ldots, n\}$ for some integer n.

Let $R_{ij}^{(k)}$ be the name of a RE whose lang is the set of strings w such that w is the label of a path from state i to state j in A, and that path has no intermediate node whose no is greater than k.

To construct the expressions $R_{ij}^{(k)}$, we use the following inductive definition, starting at $k=0$ and finally reaching $k=n$.

Basis :- $k = 0$ [No intermediate node] There kinds of paths that meet such a condition

① An arc from node (state) i to node j.

② A path of length 0 that consists of only no

If $i \neq j$ then only case ① is possible

Find the input symbols a such that there transition from state i to state j on symbol

(a) If no such symbol a then $R_{ij}^{(0)} = \phi$

(b) If there is exactly one symbol a then, $R_{ij}^{(0)}$

(c) If there are symbols $a_1, a_2 \ldots a_m$ that label from state i to state j the $R_{ij}^{(0)} = a_1 + a_2 \ldots$

If $i = j$, then the legal paths are the path of l and all loops from i to itself.

(a) $R_{ii}^{(0)} = \epsilon$ No symbols, a

(b) $R_{ii}^{(0)} = a + \epsilon$ only one symbol a

(c) $R_{ii}^{(0)} = a_1 + a_2 \ldots + a_m + \epsilon$, m no of sy

Induction :-

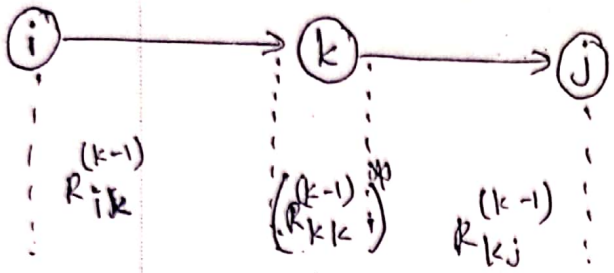
Suppose there is a path from state i t through no state higher than k. Then thre two possible cases.

① path does not go through state k at all. case, the label of the path is in the langua $R_{ij}^{(k-1)}$

(2) The path goes through state k at least once.

Then the path can be cut into several pieces.

i to k, k to itself, k to j path.



The set of labels for all path ~~of this type~~ is represented by the RE $R_{ik}^{(k-1)} \cdot (R_{kk}^{(k-1)})^* \cdot R_{kj}^{(k-1)}$

when above two cases are combined. we get RE

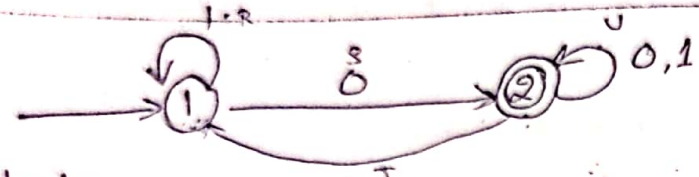
$$R_{ij}^{(k)} = R_{ij}^{(k-1)} + R_{ik}^{(k-1)} \cdot (R_{kk}^{(k-1)})^* \cdot R_{kj}^{(k-1)}$$

for the labels of all paths from state i to j that through no state higher than k.

By constructing the RE in increasing order of subscripts. each $R_{ij}^{(k)}$ depends only on expressions with a smaller superscript and all the regular expression are available whenever there is a need. Finally, we will have $R_{ij}^{(n)}$ for all i & j.

Assume that state i is the initial state and the RE the language is the sum of all RE $R_{ij}^{(n)}$ where state is an accepting state.

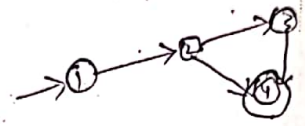
Ex:- Convert the following DFA to RE



Soln:- Basis step

$$R_{11}^{(0)} = \epsilon + 1 \qquad R_{21}^{(0)} = \phi$$

$$R_{12}^{(0)} = 0 \qquad R_{22}^{(0)} = \epsilon + 0 + 1$$



Induction step:-

$$R_{ij}^{(k)} = R_{ij}^{(k-1)} + R_{ik}^{(k-1)} \cdot \left(R_{kk}^{(k-1)}\right)^{*} \cdot R_{kj}^{(k-1)}$$

where $k = 1$.

$$R_{ij}^{(1)} = R_{ij}^{(0)} + R_{ik}^{(0)} \left(R_{kk}^{(0)}\right)^{*} \cdot R_{kj}^{(0)}$$

i) $$R_{11}^{(1)} = (\epsilon+1) + (\epsilon+1)\cdot(\epsilon+1)^{*}\cdot(\epsilon+1)$$

ii) $$R_{12}^{(1)} = 0 + (\epsilon+1)\cdot(\epsilon+1)^{*}(0)$$

iii) $$R_{21}^{(1)} = \phi + \phi\cdot(1+\epsilon)^{*}(1+\epsilon)$$

iv) $$R_{22}^{(1)} = (\epsilon+0+1) + \phi\cdot(1+\epsilon)^{*}0$$

$R_{11}^{2} = 1^{*} + 1^{*} = 1^{*}$

$R_{12}^{2} = 1^{*}0(0+1)^{*}$

$R_{21}^{2} = \phi$

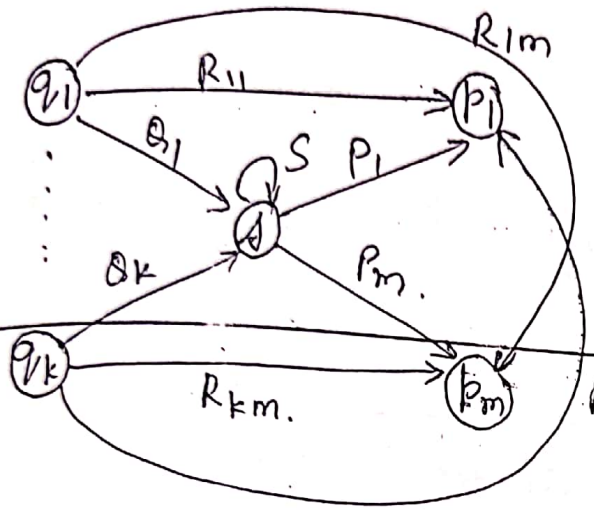
$R_{22}^{2} = (0+1)^{*}$

The Final RE will be R_{ij}^{n} where initial state, j is final state in the max no state.

iii) $\boxed{R_{12}^{2} = 1^{*}0(0+1)^{*}}$

State Elimination method for FA to RE conversion.

General form of DFA can be given as



⇒ $q_1 \ldots q_k$ are predecessor states.

⇒ s is the intermediate state.

⇒ $p_1 \ldots p_m$ are successor nodes

⇒ The labels on arc are RE along the path.

on eliminating node 's', the arcs to be updated are identified as below

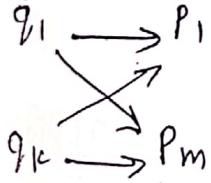
↓ ↑

$2 \times 2 = 4$

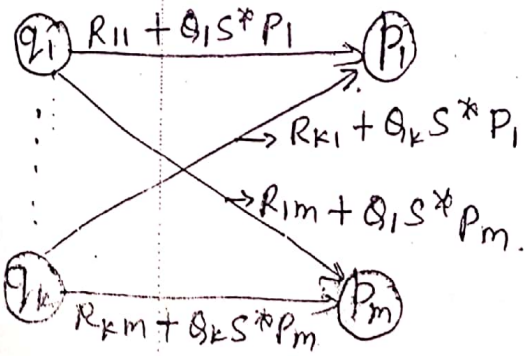
↓ → incident lines on s.

↑ → outgoing lines from s

4 → gives the no of arcs to be updated.

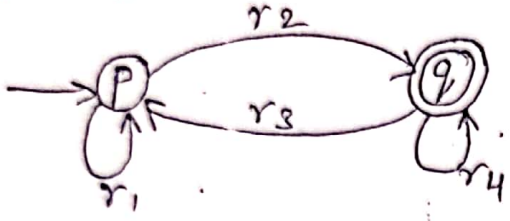


The arcs to be updated are as shown here. Finally we will have the reduced Automaton as follows.



By applying the above reduction process recursively on given FA. we will get results possible cases as follows.

Case 1 :- If start state is not final state



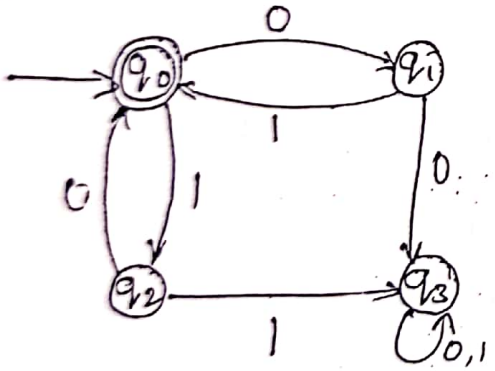
$RE = r_1^* r_2 (r_4 + r_3$

Case 2 :- If start state is final state

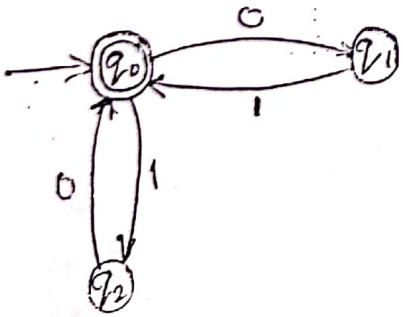
$RE = (r)^*$

The final RE is the sum of all the RE obtain the reduced automata for each accepting stat

Ex: ① Obtain RE for the below FA.



q_3 is the dead st and it is to be remov given DFA.



⇒

elimination of q_1

elimination of q_2

$RE = (01 + 10)^*$

Algebraic Laws for RE

Associativity and commutativity

(a) commutative law for union.

$$L+M = M+L$$

(b) Associative law for union.

$$(L+M)+N = L+(M+N)$$

(c) Associative law for concatenation.

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

Identities and Annihilators

(a) $\phi+L = L+\phi = L$. [ϕ is identity for union]

(b) $\epsilon L = L\epsilon = L$ [ϵ is identity for join]

(c) $\phi L = L\phi = \phi$ [ϕ is annihilator for join]

Distributive Laws.

(a) $L(M+N) = LM+LN$. [Left distributive law of join]

(b) $(M+N)L = ML+NL$ [Right distributive law of join]

Idempotent Law

(a) $L+L = L$.

Laws involving closures

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

(b) $\phi^* = \epsilon$

(c) $\epsilon^* = \epsilon$

(d) $L^+ = LL^*$

(e) $L^* = \epsilon + L^+$

Properties of Regular Languages

RL exhibit two types of properties.

① Closure properties

② Decision properties.

① Closure properties

some new lang can be constructed from existing lang using certain operations such as intersection, concatenation etc we can build re... for these new languages using som of the pr... of RL. These properties are called closure pro...

There are variou closure properties. as listed

① RL are closed under union, concatenation and ...

Theorem: If L_1 and L_2 are Regular, then $L_1 \cup$ $L_1.L_2$ and L_1^* also denote RL.

Proof:- It in given that L_1 and L_2 are RL there exists RE R_1 and R_2 such that

$L_1 = L(R_1)$

$L_2 = L(R_2)$

By the definition of RE, we have

(a) $R_1 + R_2$ is a RE denoting the lang $L_1 \cup$

(b) $R_1 . R_2$ is a RE denoting the lang $L_1.L$

(c) R_1^* is a RE denoting the lang L_1^*

so, the RL are closed under union, concatena... and star...

Closure under complementation

Theorem :- If L in RL, then the complement of L denoted by $\bar{L}$ is also regular.

proof :-

Let $M_1 = (Q, \Sigma, \delta, q_0, F)$ be a DFA which accepts the Lang L which is regular.

Let us define the machine $M_2 = (Q, \Sigma, \delta, q_0, Q-F)$ which accepts $\bar{L}$.

Note that there is no diff but M_1 and M_2 except the final states.

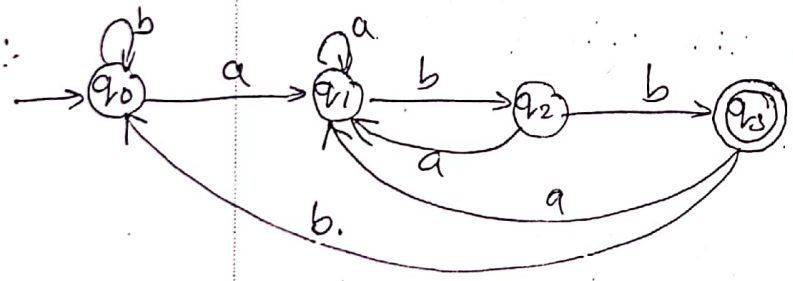
The final state of M_1 will be the non-final states for M_2 and vice versa.

So the lang rejected by M_1 is accepted by M_2.

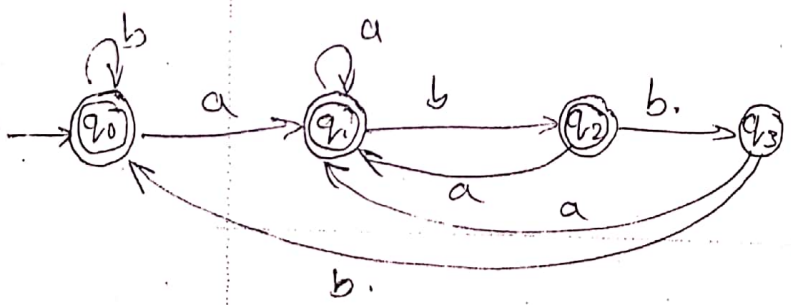
Thus, we have a machine M_2 which accepts all those strings denoted by $\bar{L}$ that are rejected by M_1.

So RL are closed under complimentation.

Ex :-



L = strings ending with abb.



$\bar{L}$ = strings not ending with abb.

③ Closure under intersection

Theorem :- If L_1 and L_2 are regular, then $L_1 \cap L_2$ is also regular.

Proof :- Let $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ accepts L_1

$M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ accepts L_2

~~Then $L_1 \cap L_2$ is accepted by the machine~~

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

$Q = Q_1 \times Q_2$

Σ is common to all.

$q_0 = (q_1, q_2)$

$F = \{ (p, q) \mid p \in F_1 \text{ and } q \in F_2 \}$

$\delta : Q \times \Sigma$ to Q is defined by

$$\delta((p, q), a) = (\delta(p, a), \delta(q, a))$$

the string is accepted iff

$\hat{\delta}((q_1, q_2), w) \in F$

iff $(\hat{\delta}_1(q_1, w), \hat{\delta}_2(q_2, w)) \in F$ This will happen

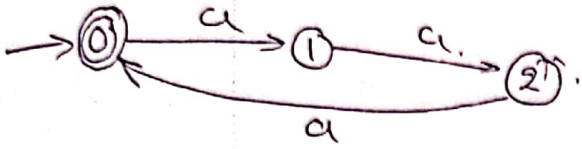
$\hat{\delta}_1(q_1, w) \in F_1$ and $\hat{\delta}_2(q_2, w) \in F_2$.

iff $w \in L_1 \cap L_2$.

x:- obtain a DFA for the language L below.

$L = \{w : |w| \bmod 3 = |w| \bmod 2,\ w \in \{a\}^*\}$

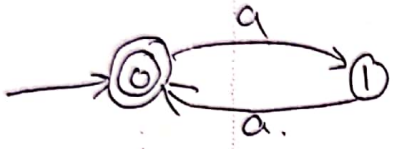
1:- $|w| \bmod 3 = 0$. M_1.



$Q_1 = \{0, 1, 2\}$.

$F_1 = \{0\}$.

$|w| \bmod 2 = 0$ M_2.



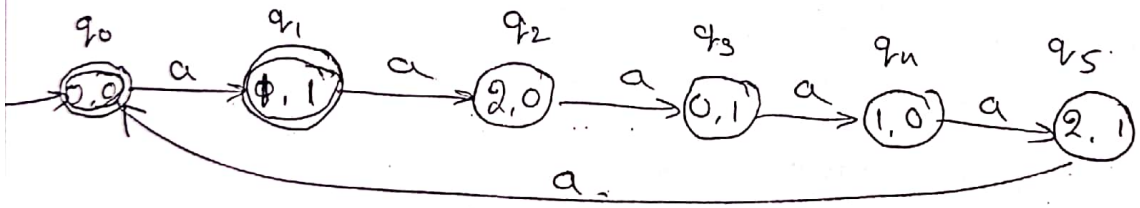
$Q_2 = \{0, 1\}$

$F_2 = \{0\}$

$\times Q_2 = Q = \{(0,0), (0,1), (1,0), (1,1), (2,0), (2,1)\}$.

$F = \{(0,0), (1,1)\}$

The DFA for $M_1 \cap M_2$ can be given as.



losure under Difference

Theorem:- If L_1 and L_2 are RL, then ~~L_1~~ $L_1 - L_2$ is regular

Proof:- Let $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ accepts L_1

$M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ accepts L_2

the we can define $L_1 - L_2$ as follows

$L_1 - L_2 = M = (Q, \Sigma, \delta, q, F)$

$Q \to Q_1 \times Q_2$ where (p, q) is in Q.

$\Sigma \to$ same as in M_1 and M_2

$\delta \to Q \times \Sigma$ to Q is defined by $\delta((p, q), a) = (\delta_1(p, a), \delta_2(q, a))$

$q \to (q_1, q_2)$ is the start state

$F \to \{(p, q) \mid p \in F_1 \text{ and } q \in F_2\}$

The string w is accepted iff.

$\hat{\delta}((q_1, q_2), w)$ is in F

i.e. $(\hat{\delta}_1(q_1, w), \hat{\delta}_2(q_2, w))$ is in F

This will happen iff

$\hat{\delta}_1(q_1, w) \in F_1$ and $\hat{\delta}_2(q_2, w) \notin F_2$

3) Closure under Reversal.

Let $w = a_1 a_2 a_3 \ldots a_n$. Then the reverse

string w denoted by w^R is defined as a str

which is written backwards i.e. $w^R = a_n a_{n-1} \ldots a_3 a$

Ex:- $w = 0111$ then $w^R = 1110$.

$w = \epsilon$ then $w^R = \epsilon^R = \epsilon$.

The reversal of Lang L is denoted by L^R is

as the set of all strings of L that are reversed.

Ex:- $L = \{\epsilon, 0, 10, 110, 11100\}$.

$L^R = \{\epsilon, 0, 01, 011, 00111\}$

Note.

Given a finite automata, the reversal of finite

can be obtained by reversing all the arcs in the

tion diagram