AUTOMATA THEORY AND FORMAL LANGUAGES

URBAN EDGE

This subject mainly deals with the study of automatic machines using formal languages. The different automata (machines) are:

1) Finite automata
2) Push-down automata
3) Turing machines

Formal languages are

1) Grammar
2) Regular-Expressions.

MODULE 1: INTRODUCTION ABOUT FINITE AUTOMATA

TERMINOLOGY:

1. Set: Set is a collection of distinct elements, all the elements of the set should be enclosed between $\{,\}$

(ex) Set of +ve integers >0 and $<=25$ which are divisible by 5.

$S = \{5, 10, 15, 20, 25\}$

2. Empty set: Set which has no elements (also called null set). denoted by ϕ

$S = \phi$

3. Power Set: Let A be the set. The set of all subsets of A is called power set of A. It is denoted by 2^A.

(ex) $A = \{a, b, c\}$

$P(A) = \{\{\}, \{a\}, \{b\}, \{c\}, \{a,b\}, \{b,c\}, \{a,c\}, \{a,b,c\}\}$

URBAN EDGE

4. Cartesian product: The cartesian product of 2 sets A and B is given by $A \times B$. Let $A = \{a, b\}$ and $B = \{c, d\}$
$A \times B = \{(a,c), (a,d), (b,c), (b,d)\}$

5. Alphabet: It is defined as a finite non empty set of symbols. Represented by Σ - set of all alphabets of a language
$\Sigma = \{\{a, b, \ldots z\}, \{0-9\}, \{(,)\}\ \#, \$, \{, ;, 0, 1\}$

6. String: Sequence of symbols obtained from alphabet of language. (ex) $\Sigma = \{a, b\} \Rightarrow \{aa, ab, \ldots aaab \ldots\}$
Empty string is denoted by ϵ (epsilon)

7.

7. Language: Set of strings obtained from Σ^*, $L \subset \Sigma^*$
$* = 0$ followed by any no of strings.
(ex) i) Language of strings consisting of equal no. of zeros and ones.
$\Sigma = 0, 1$
$L = \{\epsilon, 01, 10, 0011, 1100 \ldots\}$

ii) Language consisting of n zeros followed by n ones
$\Sigma = \{0, 1\}$
$L = \{\epsilon, 01, 0011, 000111 \ldots\}$

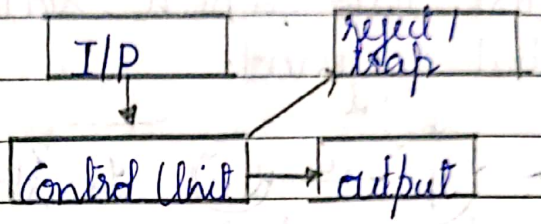
8. ~~Clean~~ Kleen closure (Kleen star): is defined as $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \ldots \Sigma^n$. which is a set of words of any length including epsilon. Take $\Sigma = \{0, 1\}$
(ex) $\Sigma^0 = \epsilon$
$\Sigma^1 = \{0, 1\}$
$\Sigma^2 = \{00, 01, 10, 11\}$
$\Sigma^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$

9. Kleen Plus: $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \ldots\ldots \cup \Sigma^n$
It is a variation of Kleen Closure denoted by Σ^+. It is set of strings except null string or ϵ.

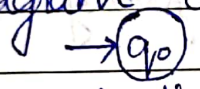
(ex) $\Sigma^+ = \{0, 1, 00, 01, 10 \ldots\ldots\}$ where $\Sigma = \{0, 1\}$.

15M FINITE AUTOMATA: 24/1/2019

1. Input unit
2. Control unit
3. Output Unit

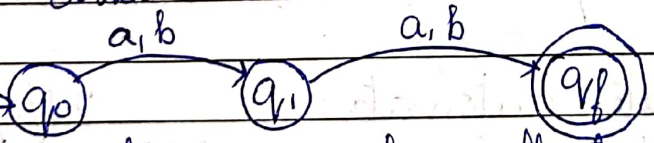


→ Finite automata machines can be represented using state transition diagrams.
Start state is identified by q_0. In state transition diagram one of the state is start state.



One of the state is final state 

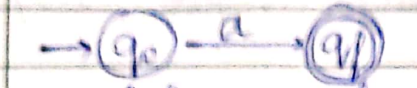
.
Transition from one state to another is represented by arrow mark →

ex 

Above diagram shows that:
The machine is in state q_0. It accepts the input symbol and changes state to q_1. In the state q_1 it accepts any symbol & changes state to q_f which is the final state.

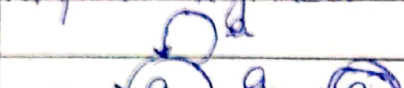
FA machines can be classified into 3 types
1. Deterministic finite automata (DFA)
2. Non-deterministic finite automata (NFA)
3. Non-deterministic finite automata with ϵ moves (NFA-ϵ moves)

1. DFA: In DFA you can determine machine enters which state after accepting the input symbol



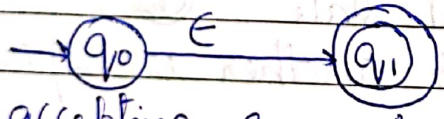
In state q_0 machine accepts the symbol a and moves to the state q_f

2. NFA: In NFA we cannot determine, machine enters into which state after accepting the input symbol.



In state q_0 machine accepts the symbol a, but we cannot determine which state it moves to (same state or next state).

3. NFA with ϵ-moves): In NFA-ϵ moves, machine enters the next state without giving any input symbol.



In state q_0, machine without accepting any input moves to next state.

Deterministic Finite automata: 27/1/2019.

DFA can be represented by machine

$M_{DFA} = (Q, \Sigma, \delta, q_0, F)$ which are five tuples. where

Q = is a set of states $\{q_0, q_1 \ldots q_n\}$

Σ = set of alphabets $\{0, 1\}$ or $\{a, b\}$ ....

δ = Transition function represented as

δ(State, input symbol) = next state

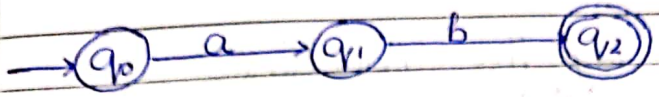
q_0 = Start state

$F = q_f$ = final state. $F = \{q_f, q_f \ldots\}$

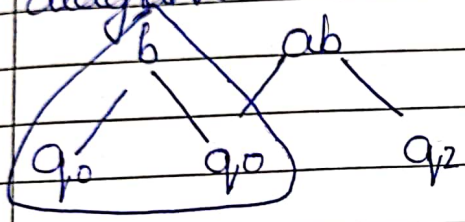
Type 1 DFA problem:

1. Obtain DFA to accept the strings a's and b's ending with ab.

ans. Step 1: Construct initial state transition diagram based on the problem statement

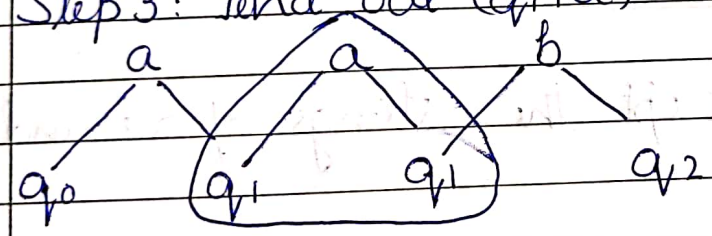


Step 2: Find out (q_0, b) i.e., q_0 on b or q_0/b for each state there exists 2 transitions a and b q_0 on a is already defined in the initial diagram.



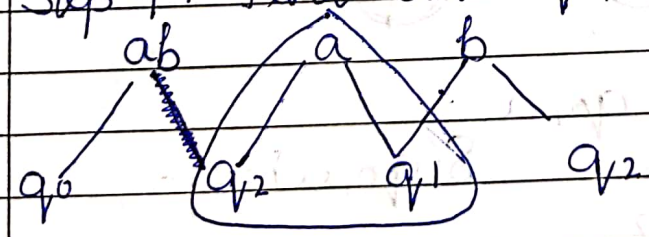
$\delta(q_0, b) = q_0$

Step 3: Find out (q_1, a)



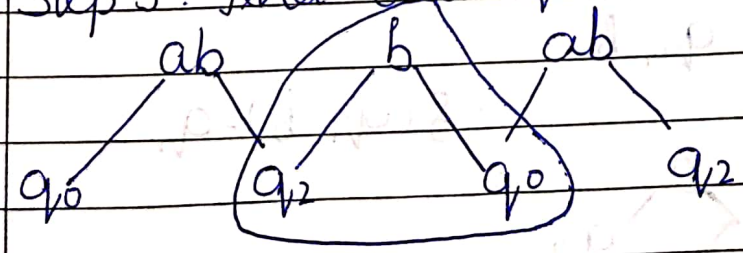
$\delta(q_1, a) = q_1$

Step 4: Find out q_2, a



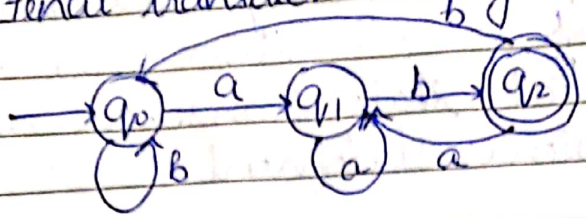
$\delta(q_2, a) = q_1$

Step 5: Find out q_2 on b



$\delta(q_2, b) = q_0$

Final transition diagram as shown below



The DFA is given by the machine

$M_{DFA} = (Q, \Sigma, \delta, q_0, F)$

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

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

$q_0 = q_0 \rightarrow$ Start state

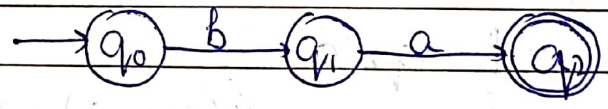
$q_f = q_2 \rightarrow$ final state

δ is as shown below.

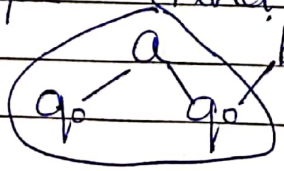
δ_{DFA}	a	b
q_0	q_1	q_0
q_1	q_1	q_2
q_2	q_1	q_0

1. Obtain DFA to accept the strings of a's and b's ending with ba

Step 1:

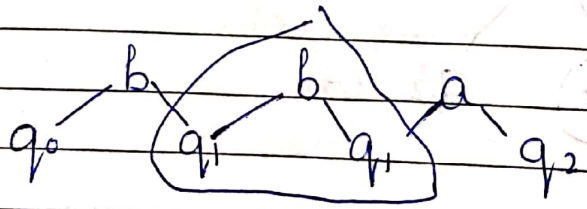


Step 2: Find out q_0, a



$\delta(q_0, a) = q_0$

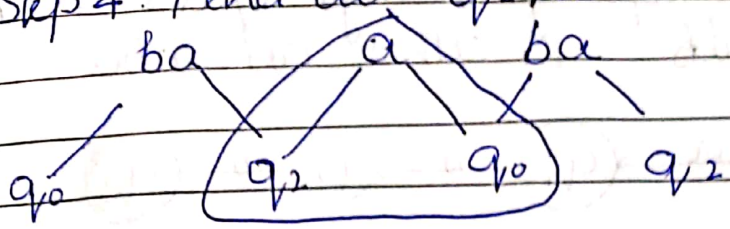
Step 3: Find out q_1, b



$\delta(q_1, b) = q_1$

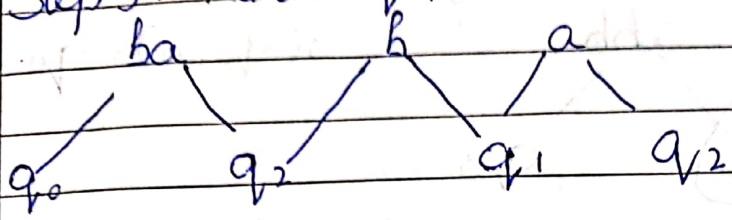
URBAN EDGE

Step 4: Find out q_2, a



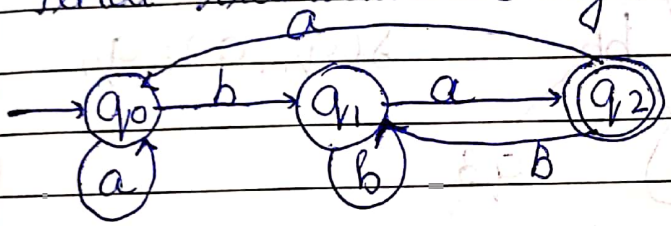
$\delta(q_2, a) = q_0$

Step 5: Find q_2, b



$\delta(q_2, b) = q_1$

Final transition diagram



The DFA is given by the machine

$M_{DFA} = (Q, \Sigma, \delta, q_0, F)$

$Q = \{q_0, q_1, q_2\}$ $\Sigma = \{a, b\}$

$q_0 \rightarrow$ Start state q_f = final state

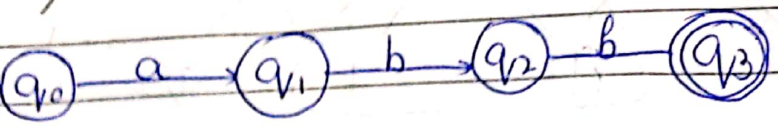
δ is as shown below:

δ_{DFA}	a	b
q_0	q_0	q_1
q_1	q_2	q_1
q_2	q_0	q_1

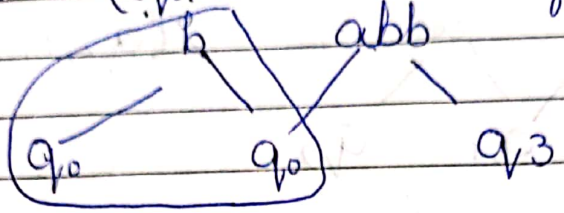
URBAN EDGE

3. Obtain DFA to accepting strings of a's and b's ending with is abb in either ab or ba

Step 1:

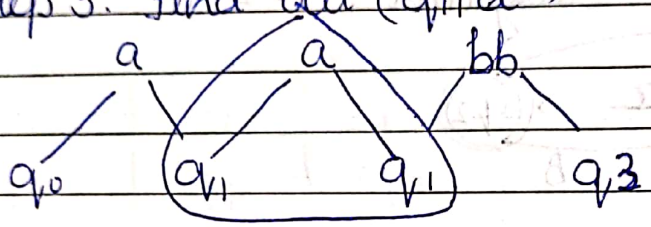


Step 2: (q_0, b) is to be found out



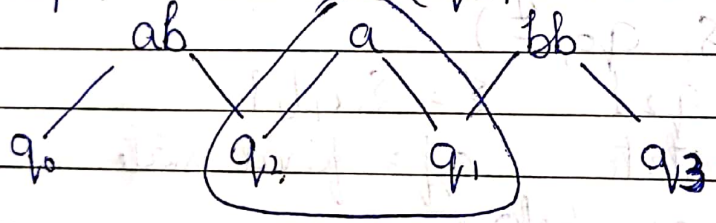
$\delta(q_0, b) = q_0$

Step 3: Find out (q_1, a)



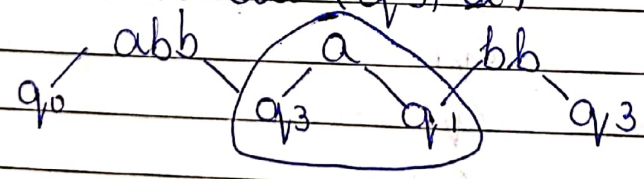
$\delta(q_1, a) = q_1$

Step 4: Find out (q_2, a)



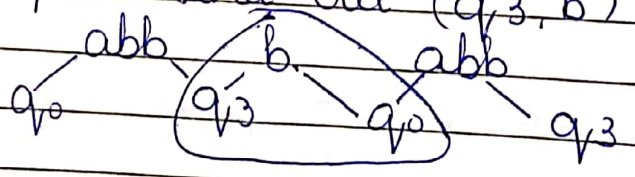
$\delta(q_2, a) = q_1$

Step 5: Find out (q_3, a)



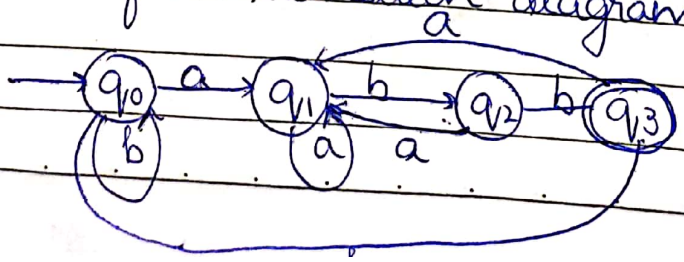
$\delta(q_3, a) = q_1$

Step 6: Find out (q_3, b)



$\delta(q_3, b) = q_0$

The final transition diagram



The DFA is given by the machine

$M_{DFA} = (Q, \Sigma, \delta, q_0, F)$

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

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

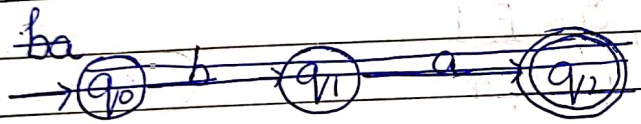
$q_0 \longrightarrow$ Start State

$q_3 \longrightarrow$ Final state

δ is as shown below

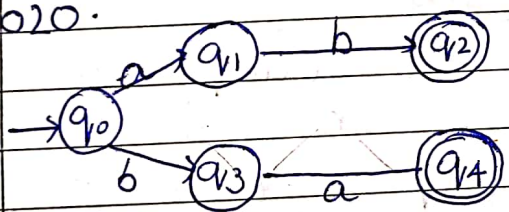
δ_{DFA}	a	b
q_0	q_1	q_0
q_1	q_1	q_2
q_2	q_1	q_3
q_3	q_1	q_0

~~# ans~~ ba



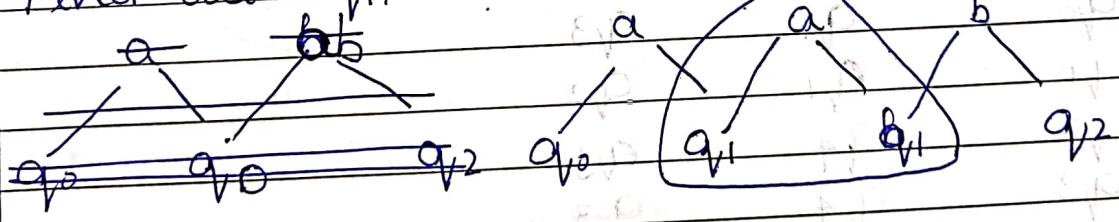
29/11/2020.

ans

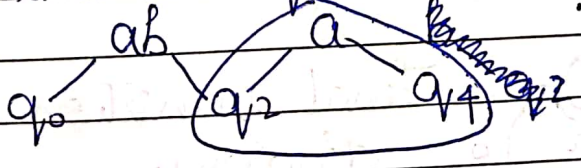


Find out q_1, a

$\delta(q_1, a) = q_1$



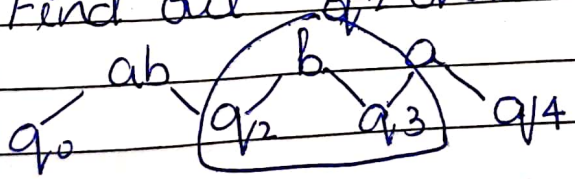
Find out q_2 on a



∵ String ba is also accepted

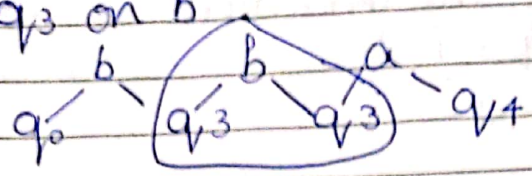
$\delta(q_2, a) = q_4$

Find out q_2 on b.



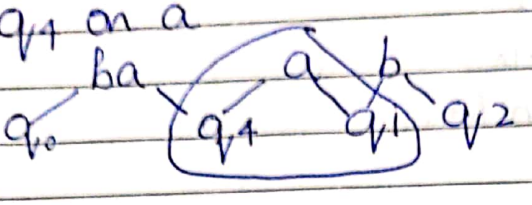
$\delta(q_2, b) = q_3$

q_3 on b



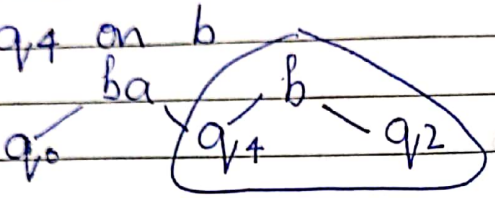
$\delta(q_3, b) = q_3$

q_4 on a



$\delta(q_4, a) = q_1$

q_4 on b



$\delta(q_4, b) = q_2$

The DFA is given by the machine

$M_{DFA} = \{Q, \Sigma, \delta, q_0, F\}$

$Q = q_0, q_1, q_2, q_3, q_4$

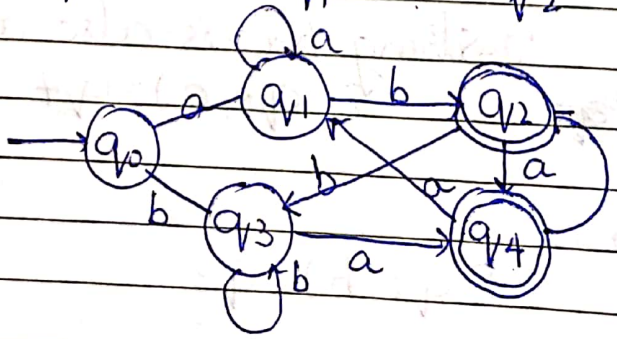
$\Sigma = \{a, b\}$

$q_0 \longrightarrow$ Start state

$\{q_2, q_4\} \longrightarrow$ final states

δ is as shown below

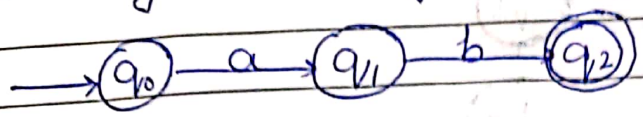
δ_{DFA}	a	b
q_0	q_1	q_3
q_1	q_1	q_2
q_2	q_4	q_3
q_3	q_4	q_3
q_4	q_1	q_2



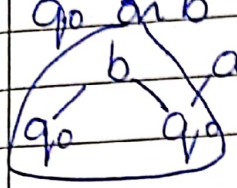
Final transition diagram.

URBAN EDGE

4. Obtain DFA to accept strings of a's and b's having substring 'ab'

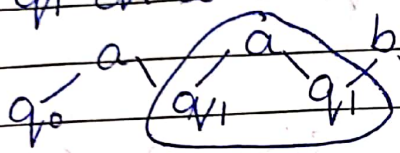


q_0 on b



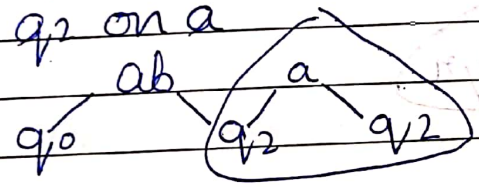
$\delta(q_0, b) = q_0$

q_1 on a



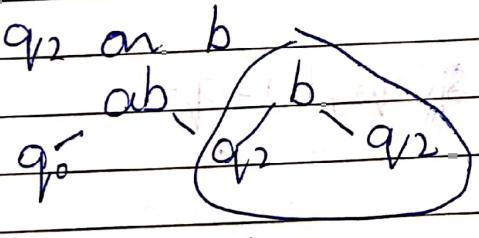
$\delta(q_1, a) = q_1$

q_2 on a



$\delta(q_2, a) = q_2$

q_2 on b



$\delta(q_2, b) = q_2$

The DFA is given by the machine

$M_{DFA} = \{Q, \Sigma, \delta, q_0, F\}$

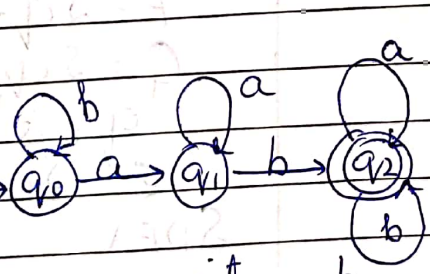
$Q = \{q_0, q_1, q_2, \cancel{q_3, q_4}\}$

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

$q_0 \longrightarrow$ Start state

$q_2 \longrightarrow$ final state

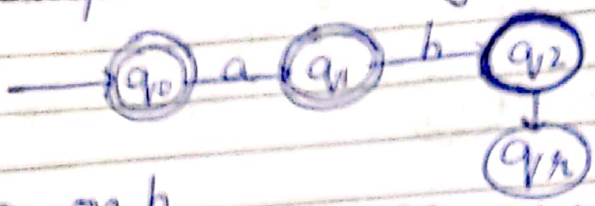
δ_{DFA}	a	b
q_0	q_1	q_0
q_1	q_1	q_2
q_2	q_2	q_2



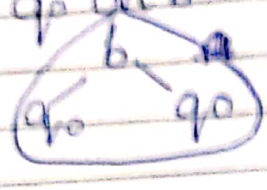
can write a, b

5. except those having the substring 'ab'

ans

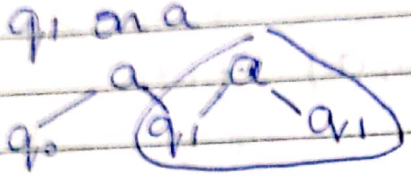


q_0 on b



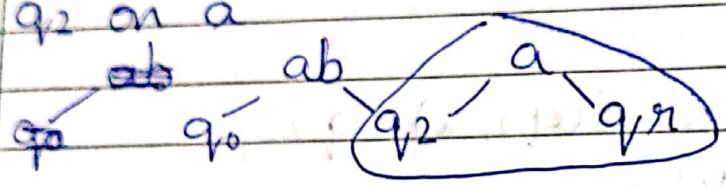
$\delta(q_0, b) = q_0$

q_1 on a



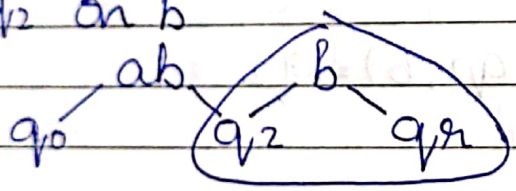
$\delta(q_1, a) = q_1$

q_2 on a



$\delta(q_2, a) = q_r$

q_2 on b



$\delta(q_2, b) = q_r$

The DFA for this machine

$M_{DFA} = \{Q, \Sigma, \delta, q_0, F\}$

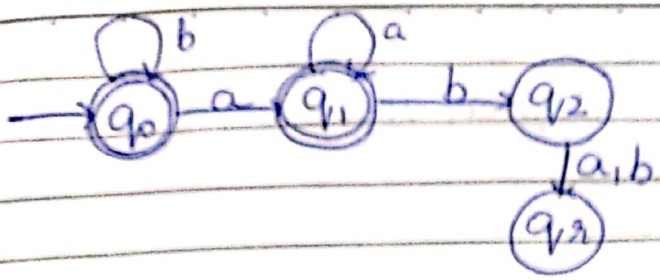
$F = \{q_0, q_1\}$

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

q_r is the rejected or trap state

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

δDFA	a	b
q_0	q_1	q_0
q_1	q_1	q_2
q_2	q_r	q_r



6. Obtain DFA to accept the strings of a's and having atleast one 'a'

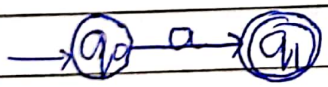
ans

7. Obtain DFA to accept the strings of a's and b's having atleast one a.

8. Obtain DFA to accept the strings of a's and b's having exactly one a.

9. DFA to accept strings of 0's and 1's having 3 concequetive zeros.

6ans Atleast one a



q_1 on a

$\delta(q_1, a) = q_1$

	a	a
q_0	q_0 q_1	q_1

SDFA

	a
q_0	q_1
q_1	q_1

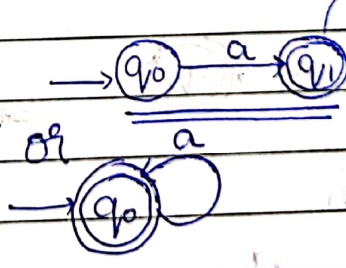
The DFA for this machine

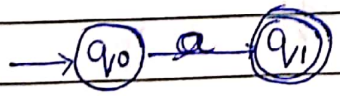
$M_{DFA} = \{Q, \Sigma, \delta, q_0, F\}$

$Q = q_0, q_1$ $\Sigma = \{a\}$

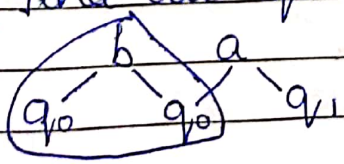
$q_0 \rightarrow$ start state

$q_1 \rightarrow$ final state



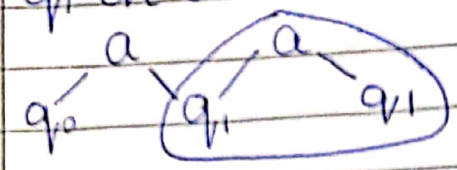
7ans 

Find del q_0 on b



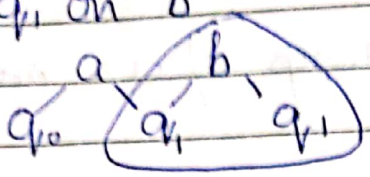
$\delta(q_0, b) = q_0$.

q_1 on a



$\delta(q_1, a) \Rightarrow q_1$

q_1 on b



$\delta(q_1, b) = q_1$

The DFA for this machine

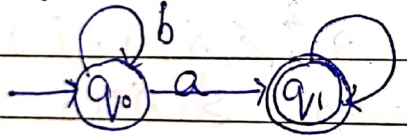
$M_{DFA} = \{Q, \Sigma, \delta, q_0, F\}$

$Q : \{q_0, q_1\}$ $q_0 \rightarrow$ Start state

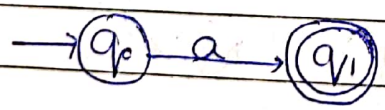
$\Sigma : \{a, b\}$ $q_1 \rightarrow$ final state

δ_{DFA}	a	b
q_0	q_1	q_0
q_1	q_1	q_1

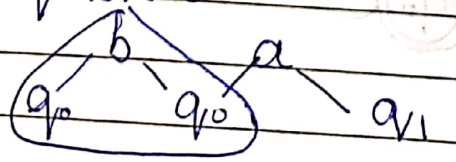
Final transition diagram



8. Exactly one a

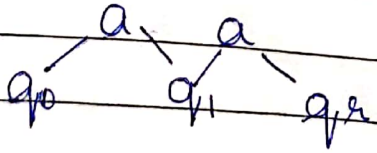


q_0 on b



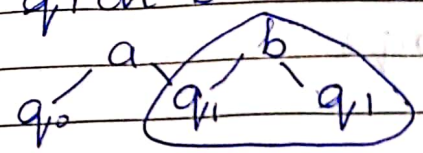
$\delta(q_0, b) \Rightarrow q_0$

q_1 on a



$\delta(q_1, a) \Rightarrow q_2$

q_1 on b



The DFA machine is given by

$M_{DFA} = \{Q, \Sigma, \delta, q_0, F\}$

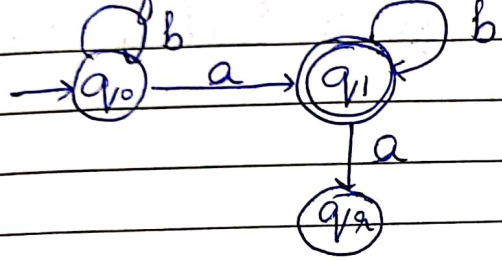
$Q = q_0, q_1$

$\Sigma = a, b$

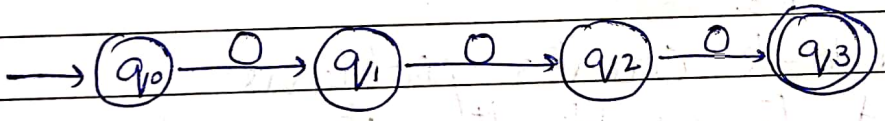
$q_0 \rightarrow$ start state $\quad F \rightarrow$ final state

δ_{DFA}	a	b
q_0	q_1	q_0
q_1	q_2	q_1

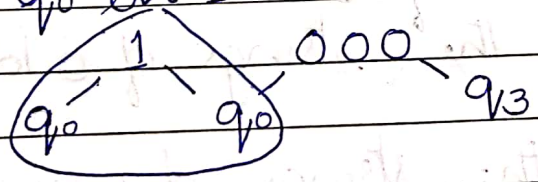
The final transition diagram



9. 0's and 1's 3 consequelive zero

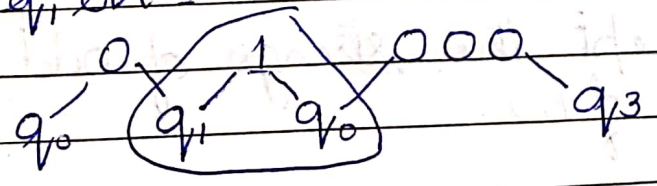


q_0 on 1



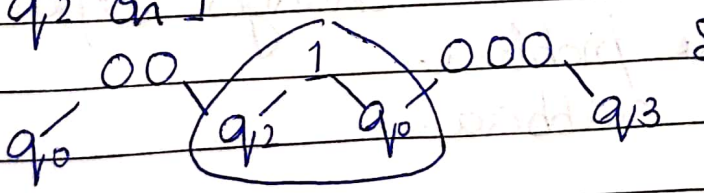
$\delta(q_0, 1) = q_0$

q_1 on 1



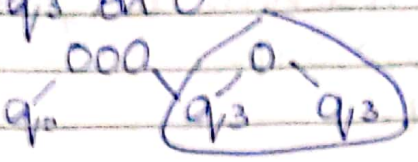
$\delta(q_1, 1) = q_0$

q_2 on 1



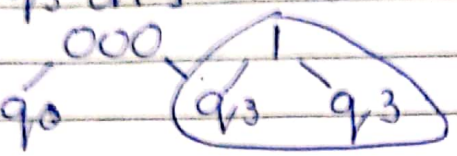
$\delta(q_2, 1) = q_0$

q_3 on 0



$\delta(q_3, 0) \Rightarrow q_3$

q_3 on 1



$\delta(q_3, 1) \Rightarrow q_3$

The DFA for this machine

$M_{DFA} = \{Q, \Sigma, \delta, q_0, F\}$ q_3 is final state

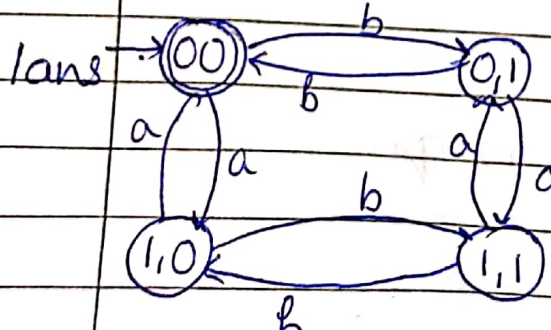
$Q = \{q_0, q_1, q_2, q_3\}$ q_0 is initial state

$\Sigma = 0, 1$

δ DFA	0	1
q_0	q_1	q_0
q_1	q_2	q_0
q_2	q_3	q_0
q_3	q_3	q_3.

Type 2: Modulo-K counter problems:

1. Obtain DFA to accept the strings of even numb of a's and even number of b's
2. Obtain DFA to accept the strings of odd numb of a's & b's
3. Obtain DFA to accept the strings of even no of a's & odd no of b's
4. Obtain DFA to accept the strings of odd no of a's & even number of b's.



Final state changes depends on the problem.

aabbaa.

MDFA = {Q, Σ, δ, q0, F}

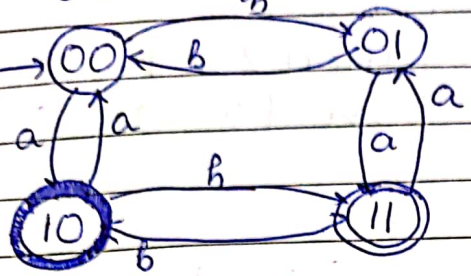
Q = {00, 01, 10, 11}

Σ = {0, 1}

00 → Start state

00 → final state

2.



MDFA = {Q, Σ, δ, q0, F}

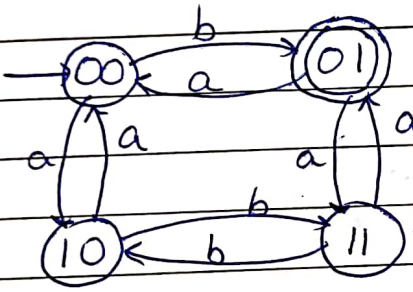
Q = {q0, q1, q2, q3}

Σ = 0, 1

00 → start state

10, 11 → final states

3.



MDFA = {Q, Σ, δ, q0, F}

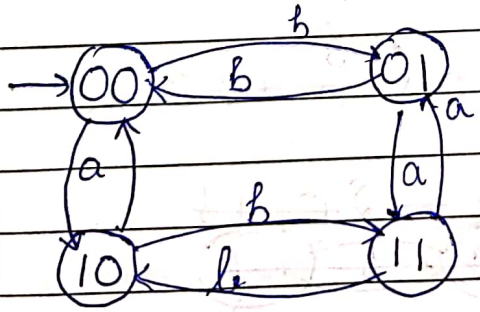
Q = {q0, q1, q2, q3}

Σ = 0, 1

00 → start state

01 → final state

4.



URBAN EDGE

5. Obtain DFA to accept the language $L = \{w | w \in (a+b)^*$ such that no. of a's in w and $n_a(w) \bmod 3 = 0$ and n_b in $w \bmod 2 = 0\}$.

ans. $n_a(w) \bmod 3 = 0$

$0 \bmod 3 = 0$

$1 \bmod 3 = 1$

$2 \bmod 3 = 2$

$3 \bmod 3 = 0$

possible remainders

$\{0, 1, 2\}$ X $\{0, 1\}$

Take cross product to get no. of states

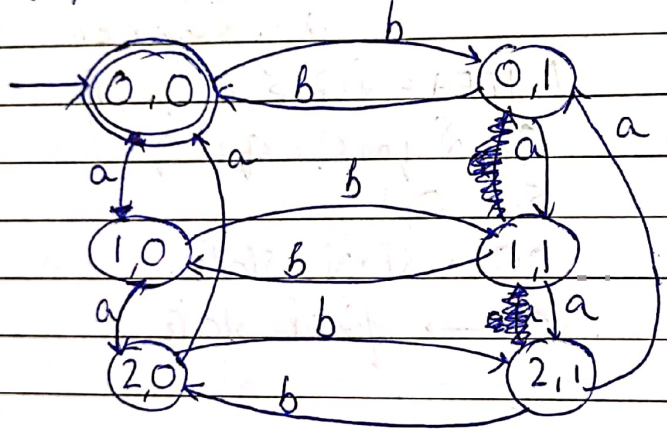
~~(0,0)~~ $\{(0,0), (0,1), (1,0), (1,1), (2,0), (2,1)\}$

$n_b(w) \bmod 2 = 0$

$0 \bmod 2 = 0$

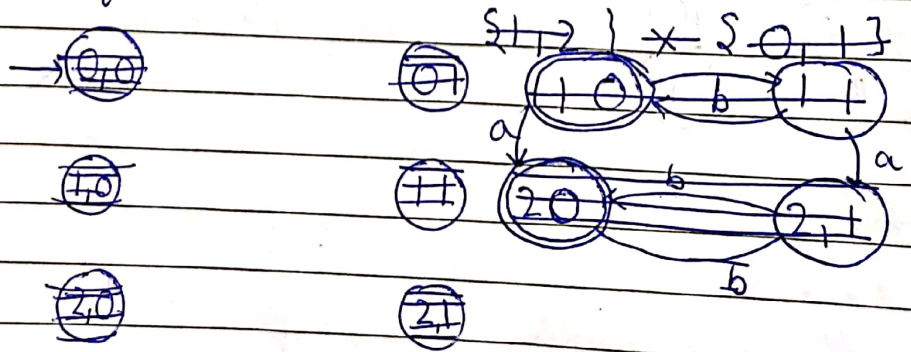
$1 \bmod 2 = 1$

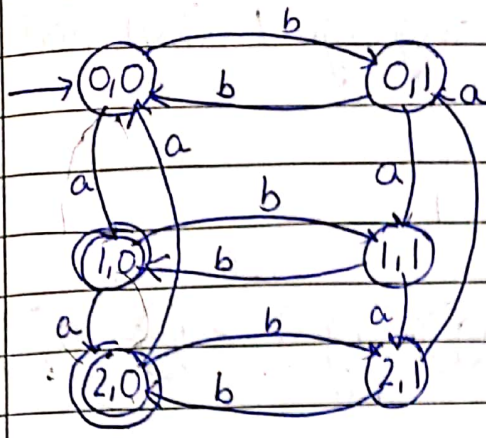
$2 \bmod 2 = 0$.



6. Obtain DFA

$L = \{w | w \in (a,b)^*$ no. of a's $\bmod 3$ ~~in w~~ is >0 and no. of b's $\bmod 2$ in $w = 0\}$.

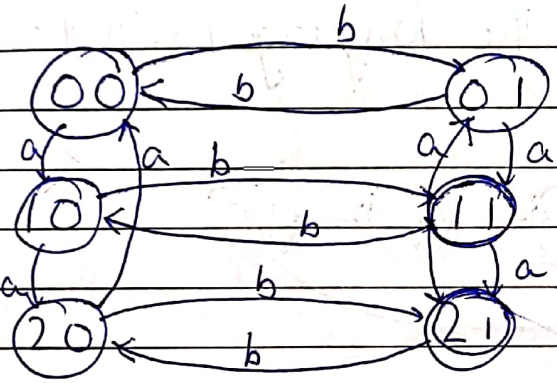




7. $n_a(w) \bmod 3 > 0$ and $n_b(w) \bmod 2 > 0$

possible remainders

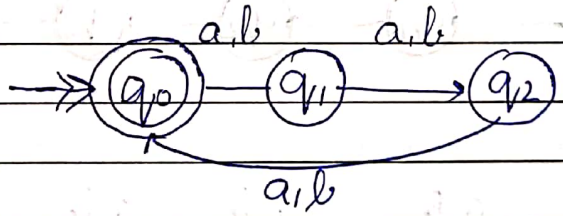
$\{0,1,2\} \times \{0,1\}$



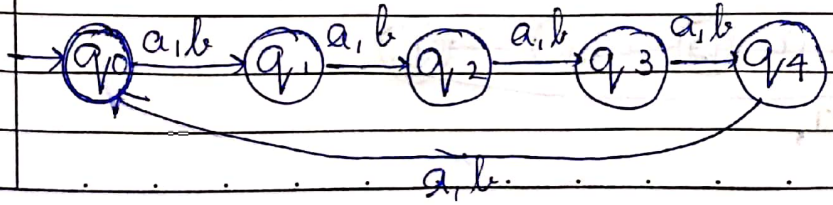
aabbaabbb

Type 3 : 1) Obtain DFA to accept the language $L = \{w : w \in |w| \bmod 3 = 0\}$ on $\Sigma = a, b$

$0 \bmod 3 = 0$
$1 \bmod 3 = 1$
$2 \bmod 3 = 2$
$3 \bmod 3 = 0$



2) Obtain DFA to accept the language $L = |w| \bmod 5 = 0$ on $\Sigma = \{a, b\}$

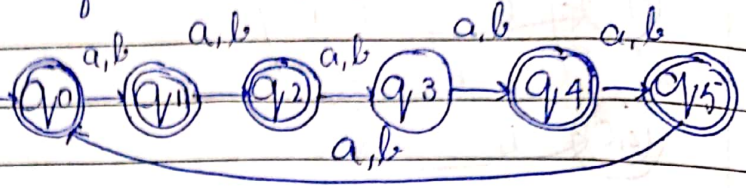


3. Obtain DFA to accept the language $L = \{|w| \bmod 3 = |w| \bmod 2\}$

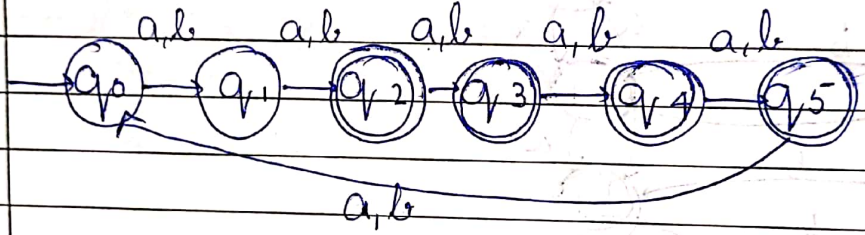
ans Possible remainders for 3 = 0, 1, 2

Possible remainders for 2 = 0, 1

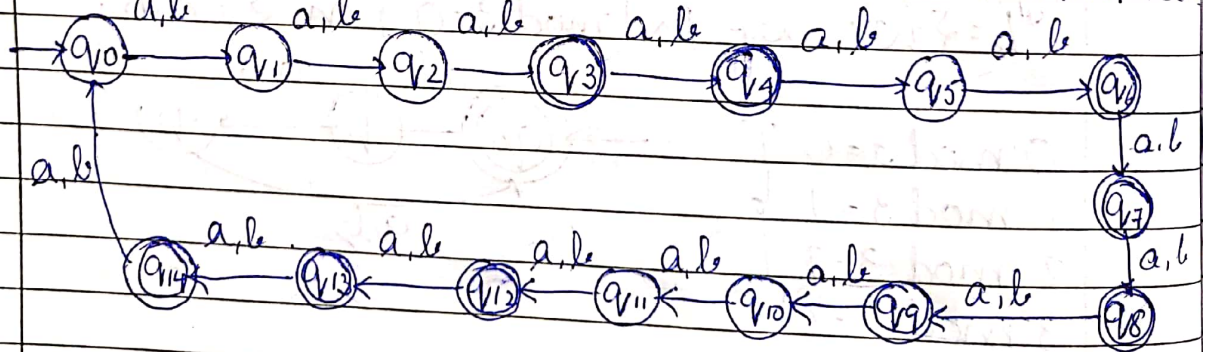
cross product	
00	01
10	11
20	21



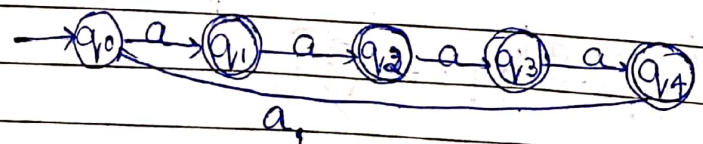
4. Obtain DFA to accept the language $L = \{|w| \bmod 3 \neq |w| \bmod 2\}$



5. ~~$L = \{|w| \bmod 3 > |w| \bmod$~~ $L = \{|w| \bmod 5 > |w| \bmod$



6. Obtain DFA to accept language $L = \{|w| \bmod 5 \neq 0\}$ on $\Sigma = \{a\}$



Type 4: Divisible of k problems:-

In this method of constructing DFA, that divide a number by k. For these type of problems the transitions can be obtained using the following relation.

$\delta(q_i, d) = q_j$ where $j = (r * i + d) \bmod k$.

where r = radix of input,

For binary $r = 2$, for decimal $r = 10$

i represents remainder obtained after dividing by k. d represents digits. For binary $d = \{0, 1\}$ for decimal $d = \{0, \ldots, 9\}$

k represents divisor.

* 1. Construct a DFA which accepts the strings of 0's and 1's where value of each string is represented as a binary number, only the strings representing 0 modulo 5 should be accepted.

ans Step 1: Identify the radix, input alphabets and k value.

$r = 2$

$d = \{0, 1\}$

$k = 5$

Step 2: Compute possible remainders after dividing by k. The possible remainders are:-

$\{0, 1, 2, 3, 4\}$

Step 3: Compute the transitions using the formula.

$\delta(q_i, d) = q_j$

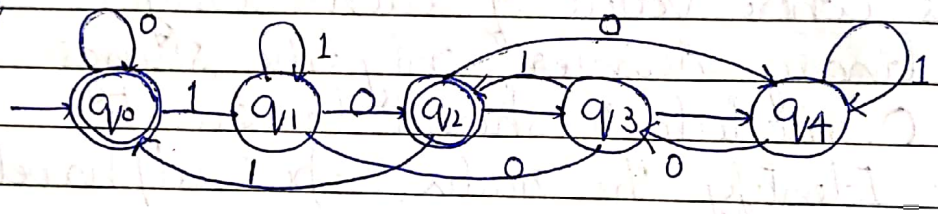
$j = (r * i + d) \bmod k$

where $i = \{0, 1, 2, 3, 4\}$

* final state.

Rem	d	$j=(R*i)+d \bmod k$	$\delta(q_i, d) = q_j$
0*	0	0	$\delta(q_0, 0) = q_0$
	1	1	$\delta(q_0, 1) = q_1$
1	0	2	$\delta(q_1, 0) = q_2$
	1	3	$\delta(q_1, 1) = q_3$
2*	0	4	$\delta(q_2, 0) = q_4$
	1	0	$\delta(q_2, 1) = q_0$
3	0	1	$\delta(q_3, 0) = q_1$
	1	2	$\delta(q_3, 1) = q_2$
4	0	3	$\delta(q_4, 0) = q_3$
	1	4	$\delta(q_4, 1) = q_4$

Transition diagram as shown below



Transition Table as shown below

δDFA	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
q_4	q_3	q_4

2. Construct a DFA which accept strings of 0's & 1's ..... 0 mod 3 must be accepted.

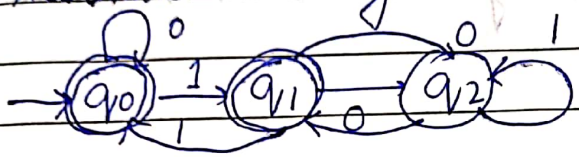
$R = 2$ $\quad i = \{0, 1, 2\}$

$d = \{0, 1\}$

$K = 3$

sm	d	j	$\delta(q_i, d) = q_j$
0*	0	0	$\delta(q_0, 0) = q_0$
	1	1	$\delta(q_0, 1) = q_1$
1*	0	2	$\delta(q_1, 0) = q_2$
	1	0	$\delta(q_1, 1) = q_0$
2	0	1	$\delta(q_2, 0) = q_1$
	1	2	$\delta(q_2, 1) = q_2$

Transition diagram as shown below



Transition table is as follows:

δ DFA	0	1
q_0	q_0	q_1
q_1	q_2	q_0
q_2	q_1	q_2

Non-deterministic finite automata:

Obtain NFA

Definition: NFA can be defined as 5 tuples

$M_{NFA} = (Q, \Sigma, \delta, q_0, F)$

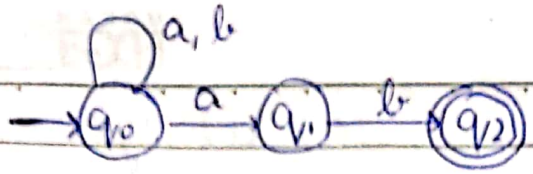
Q - set of states — q_0 = start state

Σ - set of alphabets — F = final state

δ - $Q \times \Sigma$ to 2^Q

1) Obtain NFA to accept the strings of a's and b's ending with ab

ans Step 1: Construct NFA, which is same as DFA, only difference is start state. In start state mention 2 transitions



Step 2: Construct transition table for NFA

S_{NFA}	a	b
q_0	(q_0, q_1)	q_0
q_1	—	q_2
q_2	—	—

Techniques for converting NFA to DFA:- There are 2 techniques

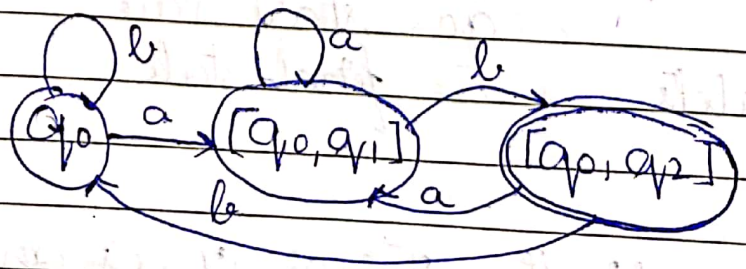
1. Lazy evaluation
2. Subset construction

Lazy Evaluation:

1. Obtain NFA to accept strings of a's & b's ending with ab & convert NFA to equivalent DFA using lazy evaluation.

Step 3:- Construct transition table for DFA

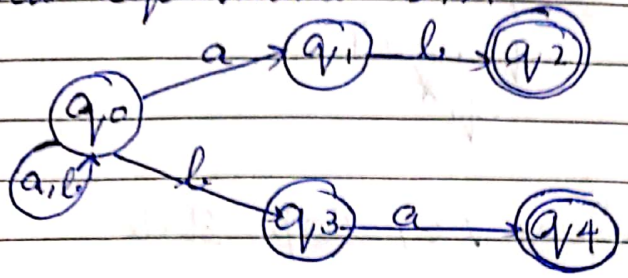
S_{DFA}	a	b
q_0	$[q_0, q_1]$	q_0
$[q_0, q_1]$	$[q_0, q_1]$	$[q_0, q_2]$
$[q_0, q_2]$	$[q_0, q_1]$	q_0



Note: In reality NFA doesn't exist, so we have to convert NFA to DFA.

2. Obtain NFA to accept the strings of a's and b's ending with 'ab' or 'ba' and convert NFA to its equivalent DFA.

Step 1:

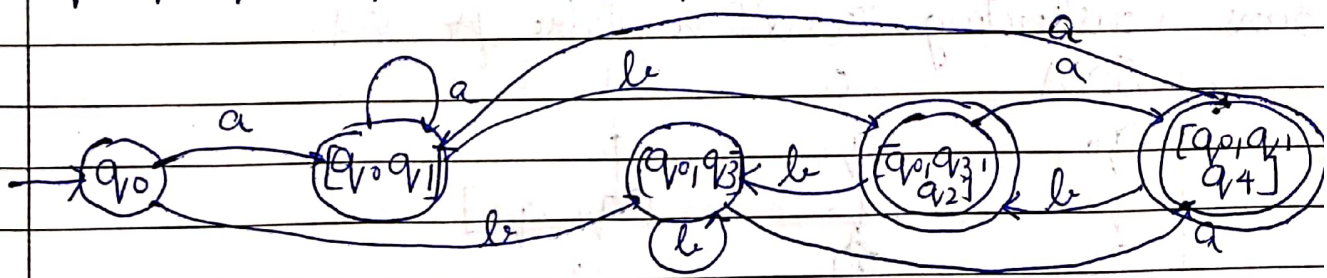


Step 2:

δNFA	a	b
q_0	$[q_0, q_1]$	$[q_0, q_3]$
q_1	—	q_2
q_2	—	—
q_3	q_4	—
q_4	—	—

Step 3: Construct transition table for DFA

δ DFA	a	b
q_0	$[q_0, q_1]$	$[q_0, q_3]$
$[q_0, q_1]$	$[q_0, q_1]$	$[q_0, q_3, q_2]$
$[q_0, q_3]$	$[q_0, q_1, q_4]$	$[q_0, q_3]$
$[q_0, q_3, q_2]$	$[q_0, q_1, q_4]$	$[q_0, q_3]$
$[q_0, q_1, q_4]$	$[q_0, q_1]$	$[q_0, q_3, q_2]$



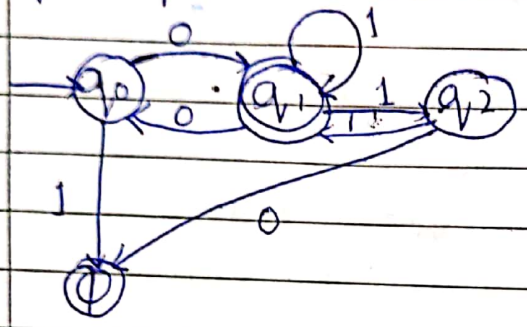
Transition diagram for DFA

Convert

3. Obtain NFA to equivalent DFA

δNFA	0	1
q_0	q_1	ϕ
q_1	q_0	$[q_1, q_2]$
q_2	ϕ	q_1*

ϕ implies ⇒ no transitions i.e., trap state.



4. Convert NFA to equivalent DFA.

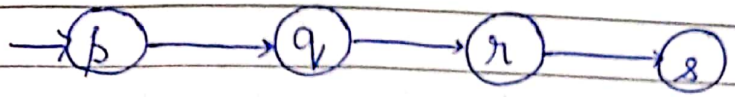
δNFA	0	1
P	$\{q, s\}$	$\{q\}$
q	r	$\{q, r\}$
r	$\{s\}$	$\{p\}$
s	ϕ	p

Subset Construction Method:-

ans Constructing transition table for DFA.

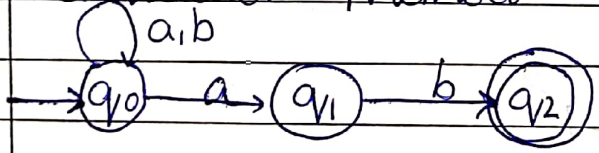
δDFA	0	1
q_0	q_1	ϕ
q_1	q_0	$[q_1, q_2]$
q_2	ϕ	q_1.
$[q_1 q_2]$	$[q_0 q_1 q_2]$	$[q_1 q_2]$

4.



Subset Construction Method :-

1. Obtain DFA for the following NFA using subset construction method



Step 1: Identify the start state of DFA.
Since q_0 is the start state of NFA, Hence q_0 is also a start state in DFA.

Step 2: Identify the alphabets of DFA.
$\Sigma = a, b$

Step 3: Identify the no. of states in NFA.
$\{q_0, q_1, q_2\}$. 8 possible subsets (2^3).
No. of states present in DFA is $Q_D = 2^3 = 8$
$\{\phi, \{q_0\}, \{q_1\}, \{q_2\}, \{q_0, q_1\}, \{q_0, q_2\}, \{q_1, q_2\},$

$\{q_0, q_1, q_2\}\}$.

Step 4: Identify the final states of DFA.

F DFA: $\{q_2\}$, $\{q_0, q_2\}$, $\{q_1, q_2\}$, $\{q_0, q_1, q_2\}$ are the the final states.

Step 5:

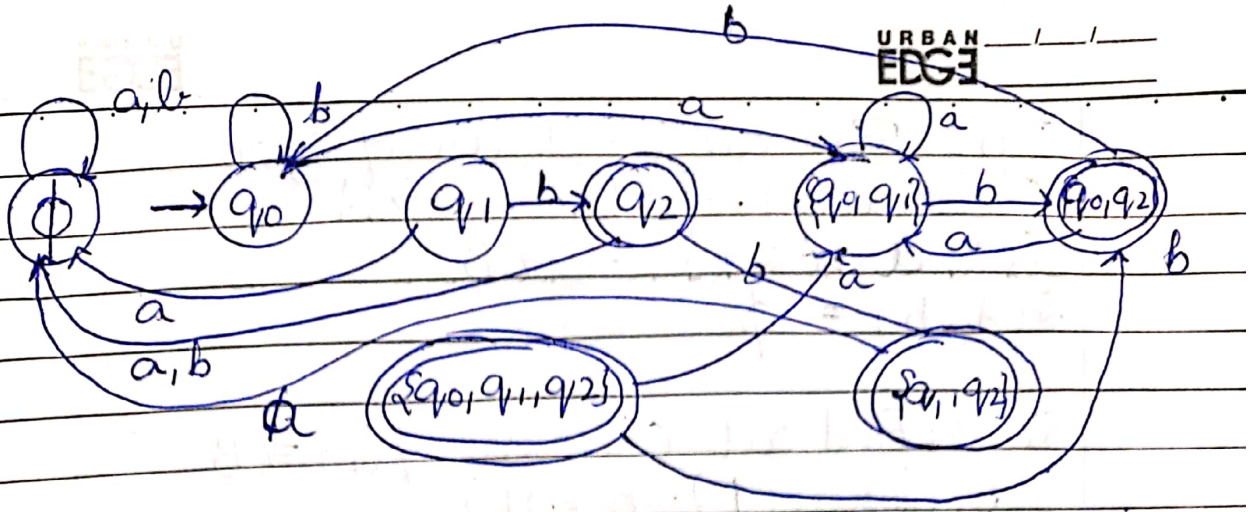
$\delta(\phi, a) = \phi$
$\delta(\phi, b) = \phi$
$\delta(q_0, a) = \{q_0, q_1\}$
$\delta(q_0, b) = \{q_0\}$
$\delta(q_1, a) = \phi$
$\delta(q_1, b) = \{q_2\}$
$\delta(q_2, a) = \phi$
$\delta(q_2, b) = \phi$

$\delta(\{q_0, q_1\}, a\} = \{q_0, q_1\}$
$\delta(\{q_0, q_1\}, b\} = \{q_0, q_2\}$
$\delta(\{q_0, q_2\}, a\} = \{q_0, q_1\}$
$\delta(\{q_0, q_2\}, b\} = \{q_0\}$
$\delta(\{q_1, q_2\}, a\} = \phi$
$\delta(\{q_1, q_2\}, b\} = \{q_2\}$
$\delta(\{q_0, q_1, q_2\}, a\} = \{q_0, q_1\}$
$\delta(\{q_0, q_1, q_2\}, b\} = \{q_0, q_2\}$

Step 6: Transition table for DFA as shown below:

SDFA	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



2. Obtain NFA to accept the strings of A's and B's ending with ab or ba and Convert NFA to DFA using subset construction method

3. Epsilon NFA:

can be defined using 5 tuples

$M_{\epsilon NFA} = \{Q, \Sigma, \delta, q_0, F\}$ where

Q is set of states

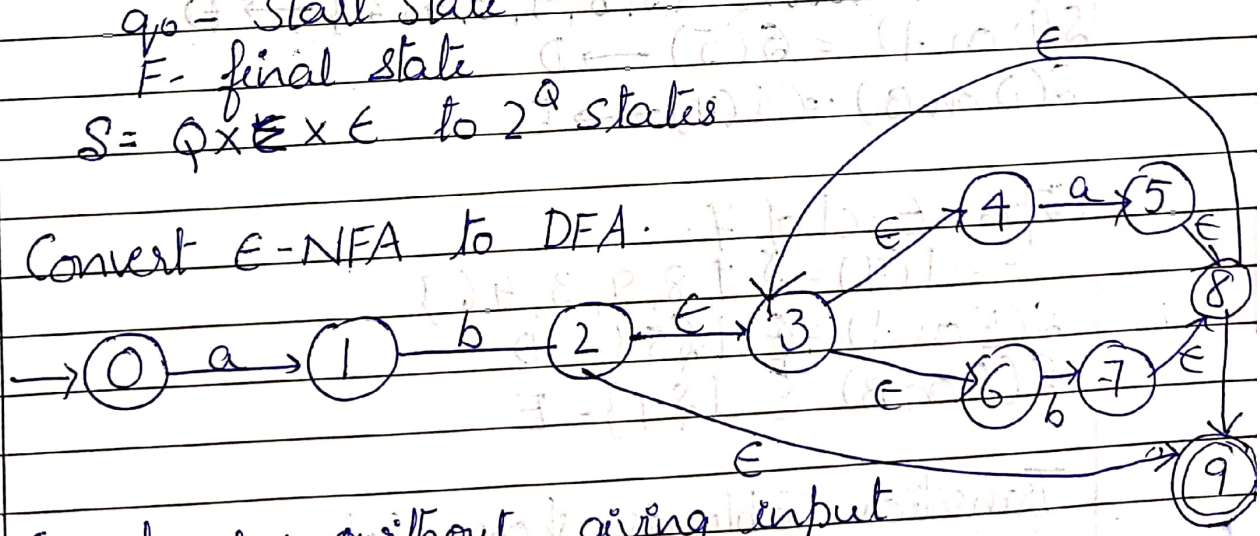
Σ set of alphabets

q_0 = start state

F - final state

$\delta = Q \times \Sigma \times \epsilon$ to 2^Q states

1. Convert ϵ-NFA to DFA.



ϵ - closure :- without giving input how many states we can reach.

ϵ-closure(3) = {3, 4, 6} ↑ including itself

Step 1 :- Here zero is the start state of NFA, then ϵ-closure of 0 is the start state of DFA.

URBAN EDGE

Step 2: Find out ϵ-closure of zero.
ϵ-closure(0) = {0} ⟶ A
δ(A, a) = ϵ-cl{1} ⟶ B
δ(A, b) = ϕ

Step 3: Find out ϵ-closure of (1) ⇒ B
ϵ-closure(1) = B = {1}
δ(B on a) = ϕ
δ(B on b) = ϵ-cl{2} ⟹ C = {2, 3, 4, 6, 9}

Step 4: Con A, Con B
ϵ-closure(2) = {2, 3, 4, 6, 9} = C
δ(C on a) = ϵ-cl(5) ⟶ D
δ(C on b) = ϵ-cl(7) ⟶ E

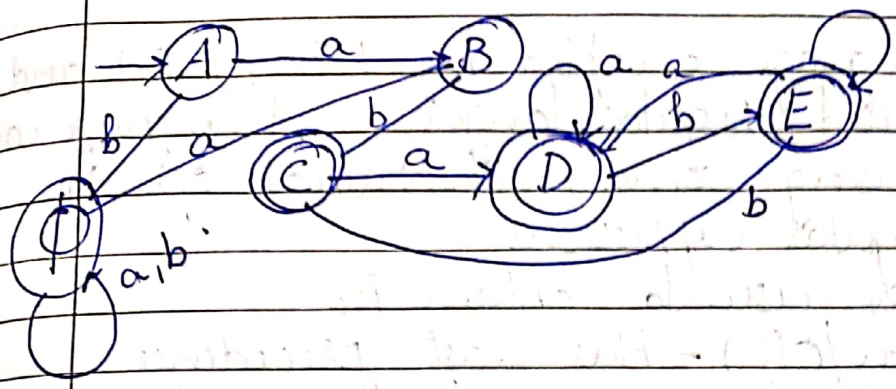
Step 5: ϵ-closure of 5
ϵ-closure(5) = {5, 8, 9, 3, 4, 6} ⟶ D
δ(D on A) = δ(5) ⟶ D
δ(D on B) = ϵ-cl(7) ⟶ E

Step 6: ϵ-cl of 7.
ϵ-cl(7) = {7, 8, 9, 3, 4, 6}
δ(E on A) = ϵ-cl{5} = D
δ(E on B) ϵ-cl{7} = E

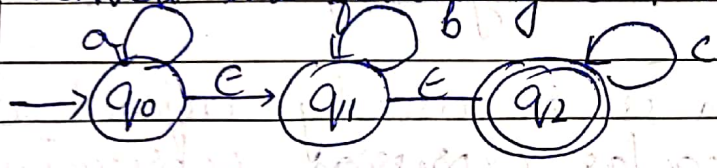
Transition Table as shown below:

SDFA	a	b
A	B	ϕ
B	ϕ	C
C	D	E
D	D	E
E	D	E

Transition diagram as shown below.



2. Convert the following ε-NFA to equivalent DFA.



URBAN EDGE

Module 2 Regular Expressions:

Definition: Language accepted by DFA, NFA and ε-NFA is called regular language. A language can be described using some mathematical operators are called regular expressions

3 operators are used to obtain RE

1) + (union/OR) - Has least precedence.
2) · (concatenation) - Next least precedence.
3) * (closure) - Has highest precedence

Regular Expression is recursively defined as

ϕ → is a regular expression representing empty language

ϵ → RE indicating empty string

a → RE indicating language containing only a.

If R and S are the regular expressions corresponding to the language L(R) and L(S) can be defined as shown below.

a) R + S : is a reg. exp corresponding to the language $L(R) \cup L(S)$.

b) R.S : corresponding to the language L(R).L(S)

c) R^* : is a reg. exp corresponding to the language L(R).

Type 1: problems

1. Obtain reg. exp consisting of any no. of a's

ns R.E = a^*

URBAN EDGE

2. Obtain RE - atleast one 'a'
ans a^+

3. Obtain RE - a's and b's of any length, including null string.
ans $(a+b)^*$ → includes ϵ

4. a's and b's ending with abb.
ans $(a+b)^* abb$

5. a's and b's starting with ab
ans $ab(a+b)^*$

6. a's and b's (any no) any no. of c (any no of)
ans $a^* b^* c^*$

7. Atleast one a, Atleast one b, atleast one c
ans $a^+ b^+ c^+$

8. 0's and 1's ending with 3 concequtive zeroes
ans $(0+1)^* 000$

9. a's & b's, whose 10th symbol from right end is a
ans ~~$(a+b)^*$~~ $(a+b)^* a (a+b)^9$

10. $L = \{w | w \bmod 3 = 0$ where $w \in (a,b)\}$
ans. $((a+b)(a+b)(a+b))^*$

11. $L = \{L = a^n b^m \mid m+n$ is even$\}$.
ans $(aa)^* (bb)^* + a(aa)^* b(bb)^*$

$R_1 \Rightarrow$ RE representing Machine M1 $L(R_1)$
$R_2 \Rightarrow$ RE representing Machine M2 $L(R_2)$

Conversion from Reg exp to finite automata

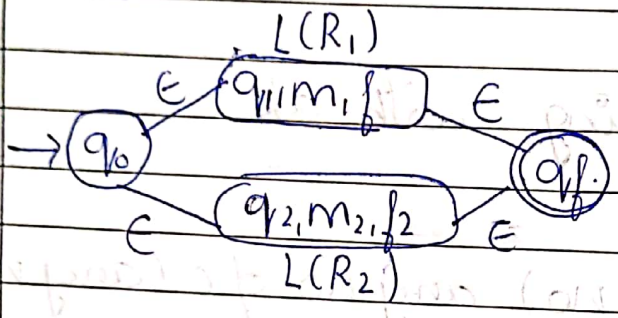
Theorm: Read it on your own.

→ Various machine corresponding to the exp R_1+R_2, $R_1.R_2$, R^*

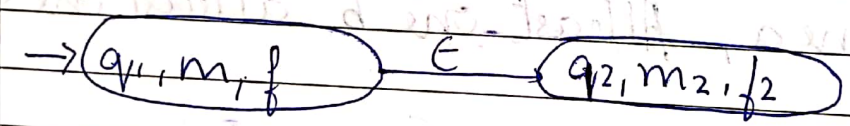
Theorm: Let R be the regular expression ∃ a finite automata $M=(Q, \Sigma, \delta, q_0, F)$

Case 1: $R = R_1 + R_2$

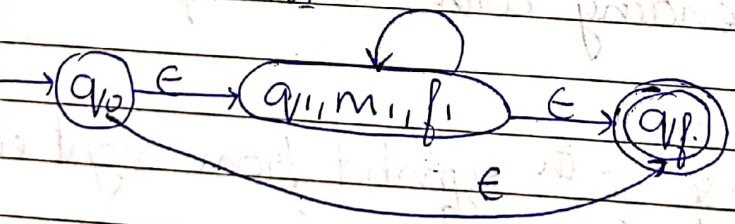
s shown below: We can construct an NFA which accepts either $L(R_1)$ or $L(R_2)$ which can be represented as shown below.



Case 2: $R = R1.R2$



Case 3: $R = R_1^*$



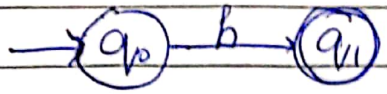
1. Obtain a RE to obtain accept strings of a's and b's starting with 'ab'. Convert the RE to finite automata.

ans. $RE = ab(a+b)^*$

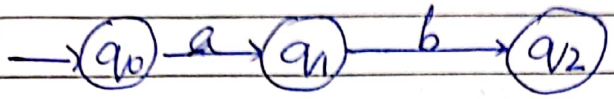
FA to accept the above RE

Step 1: Machine to accept a as shown below.

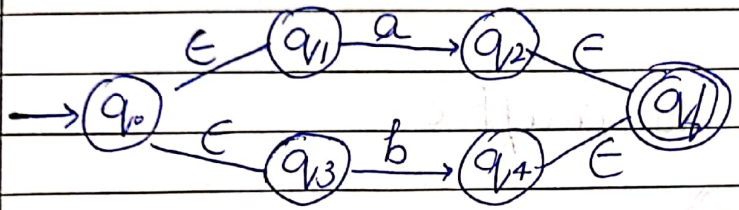
Step 2: Machine to accept b



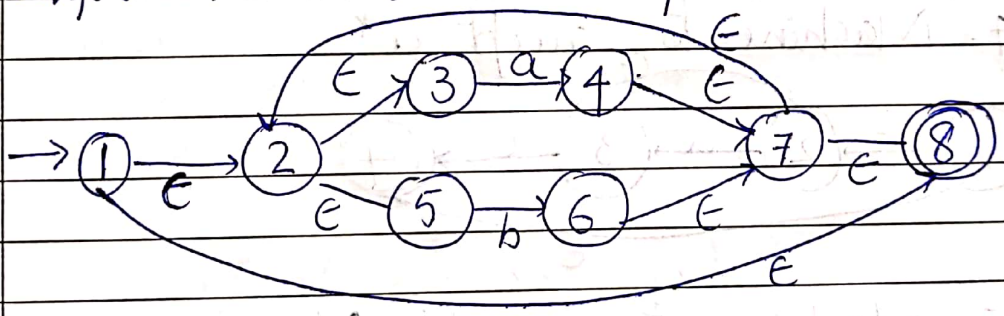
Step 3: Machine to accept ab.



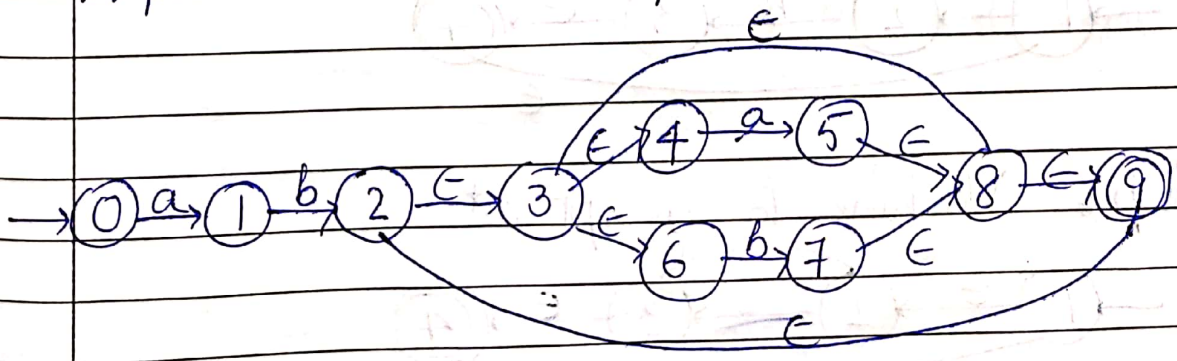
Step 4: Machine to accept a+b.



Step 5: Machine to accept (a+b)*



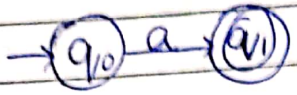
Step 6: Machine to accept ab(a+b)*



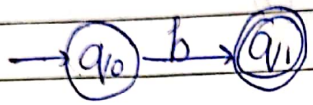
2. Convert the following RE to FA

ans RE = $a^* + b^* + c^*$

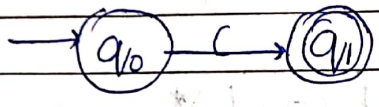
Step 1: Machine to accept a



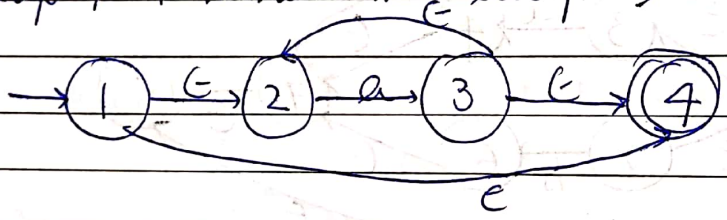
Step 2: Machine to accept b



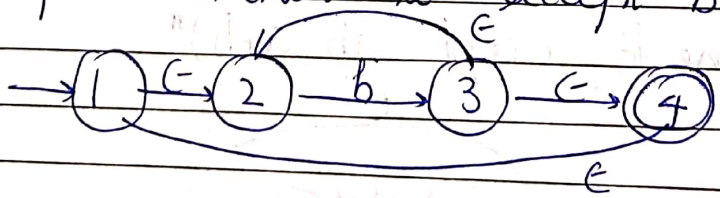
Step 3: Machine to accept c



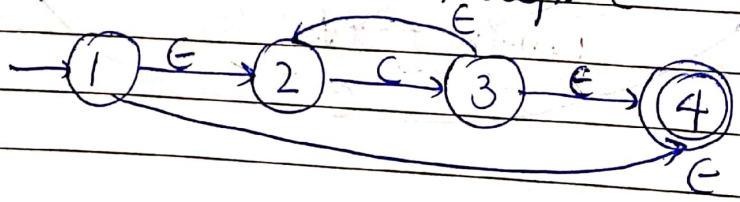
Step 4: Machine to accept a*



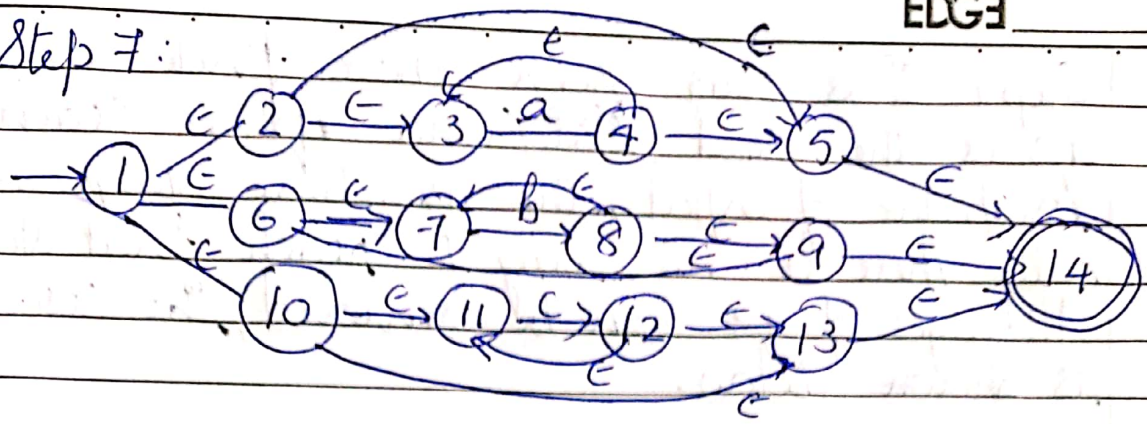
Step 5: Machine to accept b*



Step 6: Machine to accept c*



Step 7:



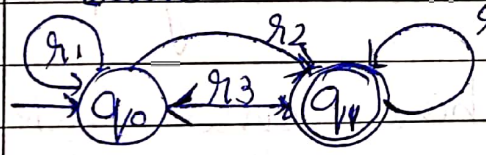
Conversion from FA to RE.

State Elimination Method — Kleen's Theorm

There Two techniques are used.

State Elimination Method: Theorm:-

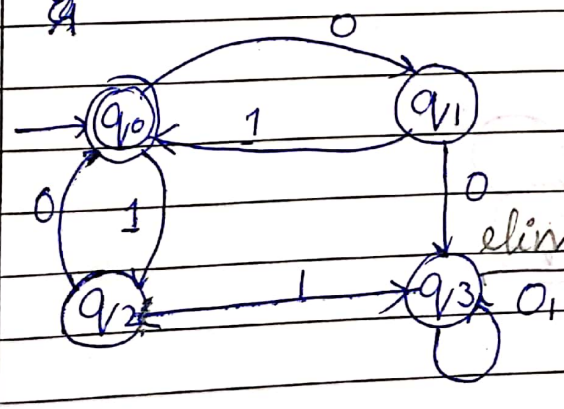
General procedure to obtain RE from FA as shown below.



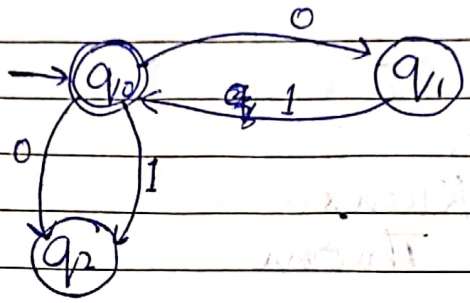
$r_1^* r_2 + r_1^* r_2 r_4^* + r_1^* r_2 r_4^* r_3 r_4^*$

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

1. Obtain a RE for the following finite automata



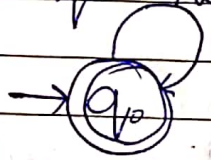
Step 1: It is clear from above finite automata q_3 is the dead state (Once q_3 is reached irrespective of input, the machine stays in q_3 only. There is no way to reach final state. All edges of q_3 can be removed, the result as shown below.



Reduce this diagram even more

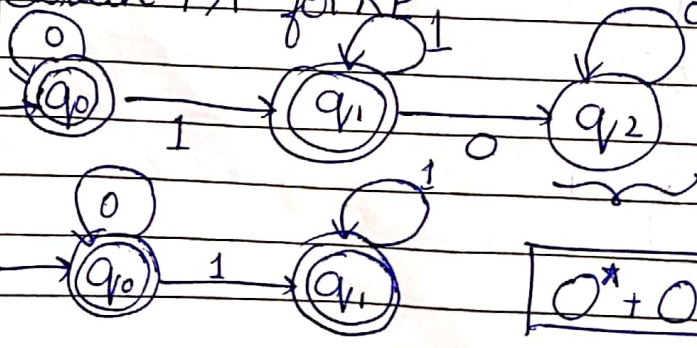
Step 2: It is clear from above diagram if input string is 0, 1 machine goes to q_1 & comes back to q_1. This process can be repeated, instead of q_1 we can loop back on the string 0, 1 as shown below.

Similarly on 1, 0 Machine comes back to state q_0. Resultant diagram as shown below



$RE = (01+10)^*$

2. Obtain FA for RE

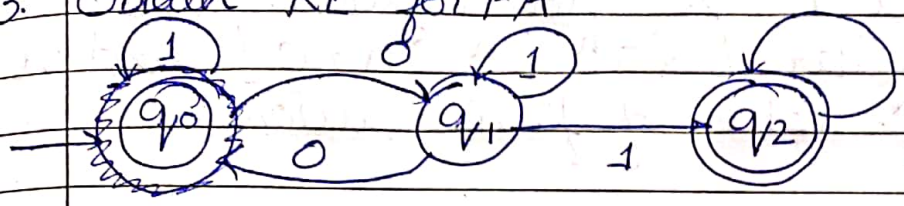


$0^* + 0^*1 + 011^*$

Pearson

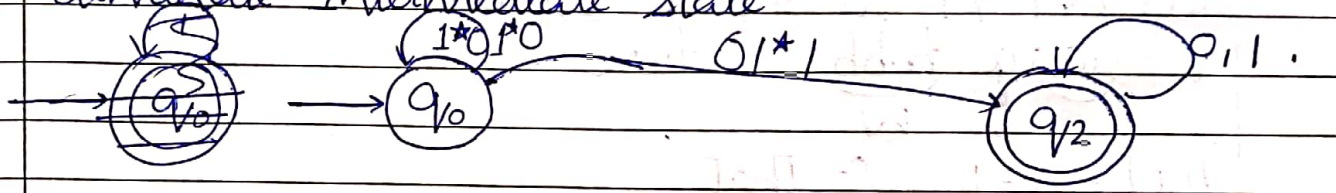
URBAN EDGE

3. Obtain RE for FA.



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

Eliminate Intermediate state



$1^*01^*001^*1(0+1)^*$ //

$R.E = (1^* + 01^*0)(01^*1)(0+1)^*$

4. Properties of R.E

Pumping Lemma:

Theorm:- Using to prove that certain languages are not regular.

Language $M = (Q, \Sigma, \delta, q_0, F)$ be a finite automata and n is no. of states. L be the regular language accepted by M.

For every string x in L, $\exists$ a constant n such that $|x| \geq n$. Now x can be broken into 3 substrings u, v and w.

Such that $x = uvw$, satisfying the following constraint

1) $v \neq \epsilon$ i.e. $|v| \geq 1$

2) $|uv| \leq n$. Then uv^iw is in L for $i \geq 0$.

URBAN EDGE

1. Show that $L = a^n b^n$ is not regular: $n \geq 0$.

Step 1: Let L be regular $L = \{a^n b^n, n \geq 0\}$ and n be the no. of states of finite automata
$x = a^n b^n$

Step 2: $|x| = 2n \geq n$.
We can split x into u, v, w such that $|uv| \leq n$ and $|v| \geq 1$ as shown below.
$x = aaaa\ bbbb$ where
$u\ v\ w$
$|u| = n-1$ so that
$|v| = 1$
$|uv| = n-1+1 = n$
$|w| = n$
According to the pumping Lemma
$uv^i w \in L$ for $i = 0, 1, 2 \ldots$

DFA
Lazy eval / Subset construction
definition
ϵ-NFA
RE

Application of FA
Apps of RE

Step 3: If you take $i = 0$
$uv^i w \in L$ when $i = 0$.
$aaa\,(a)^i\,bbbb \in L$
$aaa\,(a)^0\,bbbb \in L$
$aaabbbb \notin L$
If you substitute $i=0$ the string v doesn't appear and no. of a's will be less than the no. of b's and string x doesn't have contain $a^n b^n$. But acc to pumping lemma n no of a's must be followed b's which is contradiction to our assumption that language is regular.
imp → So $L = a^n b^n$ $n \geq 0$ is not regular

$w = ab$ $w^R = ba$
ww^R R = Reverse