Theory of Computation - BCS503

Module 1

Ch-1 : Automata - The methods & the madness.

(1.1.1)
(*) Auto Introduction

- CS is more than writing code, compiling code, fixing bugs & compiling again. Infact CS is more than how your computer's processors & chips work.
- At its core, CS is about one thing, problem solving. ToC will go beyond hardware & try to understand computing & answering one question:
  What are fundamental capabilities & limitations of computer?
- Lets say we write an automated test to check if a program will eventually finish running and stop or if it will continue to run forever - Halting Problem
  - The automated test never actually runs the program it reads lines of code.
  - We cannot test the code without executing it.
- In ToC we mathematically prove that such problems can & cannot be solved using something called models of Computation.
- Model of computation is abstract representation of machine. It is not some physical machine but like any of those machines that takes an i/p & computes an o/p.
  Eg: Turing Machine.

Eg: ① Software for designing & checking the behavior of digital circuits.
② Lexical analyzer of a typical compiler.
③ Software for scanning large bodies of texts.
④ Software for verifying systems of all types that have a finite number of distinct states.

- A finite automaton is a mathematical model which is used to study the abstract machine.

(1.1.2) Structural Representations.

- There are two important notations that are not automation like, but play an important role in the study of automata & their applications.

(1) Grammars: They are useful models when designing software the processes data with a recursive structure

$E \Rightarrow E + E$ (concatination of strings)

(2) Regular Expression: The pattern of string they describe are exactly the same as what can be described by finite automata.

[A-Z][a-z]* [ ] [A-Z] [A-Z] Bangalore BA
[[A-Z][a-z][ ]]* [A-Z][A-Z] Bangalore Rural BR

classmate
Date
Page

1.1.3 Automata of Complexity.

- Automata are essential for the study of the limits of computation.
- There are two important issues.

- What can a computer do at all?
  This study is called DECIDABILITY, and the problems that can be solved by computer are called DECIDABLE.

- What can a computer do efficiently?
  This study is called INTRACTABILITY & the problems that can be solved by computer using no more time are called TRACTABLE.

1.5 The Central Concepts of Automata Theory

The key concepts that helps us understand what a computer can do & cannot do are:

(1.5.1) Alphabets

- a finite non-empty set of symbols.
- represented by Σ

$\Sigma = \{0, 1\}$ binary alphabets
$\Sigma = \{a, b, \ldots, z\}$ english alphabets
$\Sigma = \{A, T, G, C\}$ DNA alphabets

(1.5.2) String.

- A finite set of symbols obtained from the alphabets of a language is called a string.

(*) Empty string.

- A string with zero occurance of symbols represented by ϵ

(*) Concatination of strings

a string obtained by writing the letters of u followed by letters of v.

$u = a_1 a_2 a_3 \ldots a_i$

$v = b_1 b_2 b_3 \ldots b_j$

$uv = a_1 a_2 a_3 \ldots a_i b_1 b_2 b_3 \ldots b_j$

(*) Length of string

The length of string u is the no of symbols in u & is denoted by $|u|$

$|u| = i$

(*) Power of alphabet

denoted by Σ^i is set of words of length i.

$\Sigma^0 = \{\epsilon\}$

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

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

$\Sigma^3 = \{000, 001, 010, 011, 100, 101, 110, 111\}$

$$\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \ldots$$
$$= \{0, 1, 00, 01, 10, 11, 000, 001, \ldots\}$$

$$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ldots$$
$$= \{\epsilon, 0, 1, 00, 01, 10, 11, 000, 001\}$$

$$\boxed{\Sigma^* = \Sigma^+ + \epsilon}$$

(.5.3) Language

A Language can be defined as a set of strings obtained from Σ^* denoted as:

$$L \subseteq \Sigma^*$$

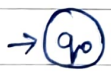
Ch 2 : Finite automata.

- An abstract machine is a conceptual or theoretical model of a computer hardware or software system which really does not exist.
- The various types of abstract machines
  - Finite automata
  - Lineal bounded automata
  - Push down automata
  - Turing ~~maching~~ machine

- A finite automata is a mathematical model which is used to study the abstract machines or abstract computing devices, with the inputs chosen from Σ.

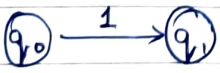
Symbols used in FA

a state



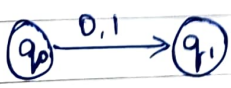
a start state

a final state



a transition from q_0 to q_1 on input 1

a transition from q_0 to q_0 on input 0



a transition from q_0 to q_1 on both input 0 & 1

Types of finite automata
- Deterministic finite automata (DFA)
- Non-Deterministic finite automata (NFA)
- Non-Deterministic finite automata with ε-moves (ε-NFA)

2.2 Deterministic Finite Automata (DFA)

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

(2.2.1) Definition of a DFA

The deterministic finite automata in short DFA is 5-tuple or quintuple :

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

where, A is name of machine
Q non empty, finite set of states
Σ non empty, finite set of input alphabets.
δ transition function which is a mapping from $Q \Sigma$ to Q
$q_0 \in Q$ is a start state
$F \subseteq Q$ is a set of accepting / finite sets.

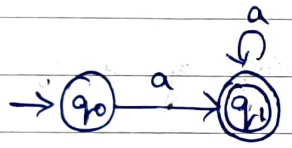
There are three types of problems for which we can construct a DFA :

- → Pattern recognition problems
- → Divisible by k problems
- → Modulo - k counter problems.

(*) Pattern Recognition Problems.

1. Draw a DFA to accept strings of a's having atleast one a.

strings) $L = \{ a, aa, aaa, aaaa, \ldots \}$



δ	a
→ q_0	q_1
* q_1	q_1

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

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

$\Sigma = \{ a \}$.

$\delta = \{ \delta(q_0, a) = q_1 \quad \delta(q_1, a) = q_1 \}$

$S = q_0$

$F = \{ q_1 \}$

2. Draw a DFA to accept string of a's & b's having at least one a.

$L = \{ a, ab, ba, aaa, aab, aba, abb, baa, bab, bba, bbb, \ldots \}$

	a	b
→ q_0	q_1	q_0
* q_1	q_1	q_1

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

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

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

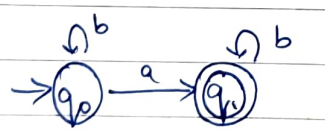
δ = transition function

$S = q_0$

$F = \{q_1\}$

3. Draw a DFA to accept strings of a's & b's having exactly one a.

$L = \{ a, ab, ba, abb, bab, bba, \ldots \}$



	a	b
→ q_0	q_1	q_0
* q_1	-	q_1

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

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

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

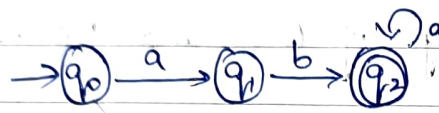
δ = transition function

$S = q_0$

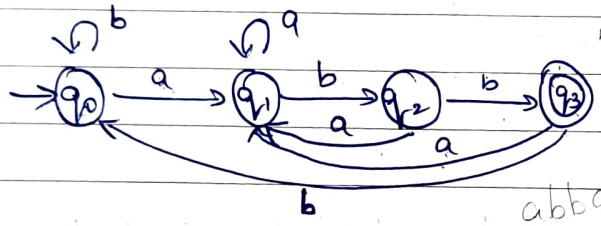
$F = \{q_1\}$

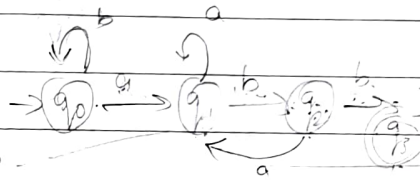
4. Draw a DFA to accept strings of a's & b's
(a) starting with the string ab
(b) ending with the string abb
(c) doesn't end with string abb.

(a) $L = \{ab, aba, abb, abaa, abab, abba, abbb, \ldots\}$



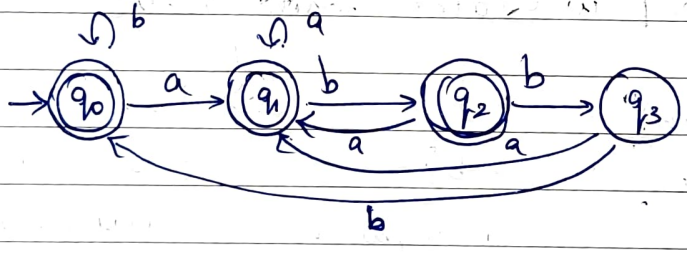
(b) $L = \{abb, aabb, babb, aaabb, ababb, baabb, bbabb, \ldots\}$



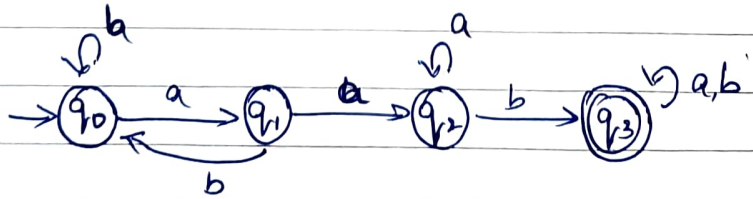


abba abb
abb b abb
abb

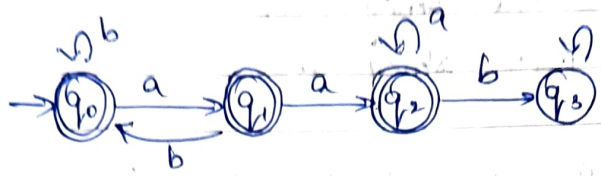
(c)



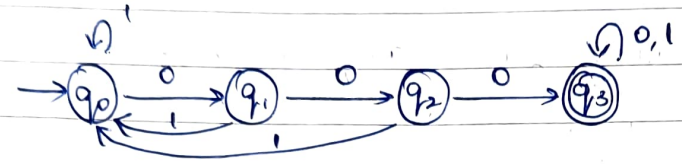
5. Draw a DFA to accept strings of a's & b's having a substring aab.
$L = \{aab, aaab, aaba, baab, aabb, \ldots\}$



6. Draw a DFA to accept strings of a's & b's except those having the substring ~~abbb~~ aab

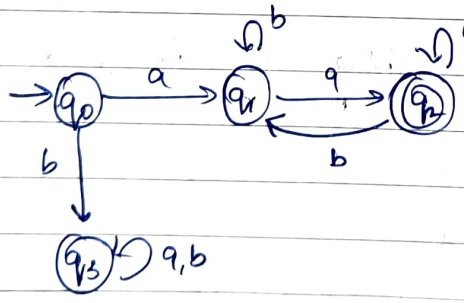


7. Draw a DFA to accept strings of 0's and 1's having three consecutive 0's.



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

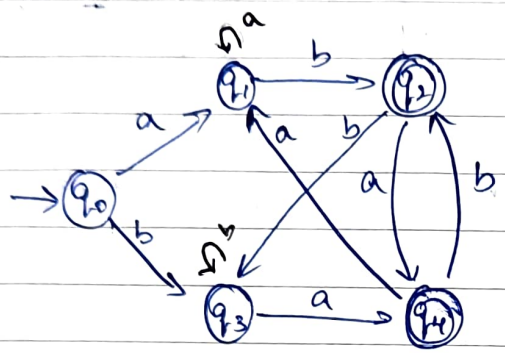
		L
$n = 0$, $(a+b)^0 = \epsilon$		$\Rightarrow \{aa\}$
$n = 1$, $(a+b)^1 = \{a, b\}$		$\Rightarrow \{aaa, aba\}$
$n = 2$, $(a+b)^2 = \{aa, ab, ba, bb\}$		$\Rightarrow \{aaaa, aaba, abaa, abba\}$



9. Draw a DFA to accept string of a's and b's ending with ab or ba.

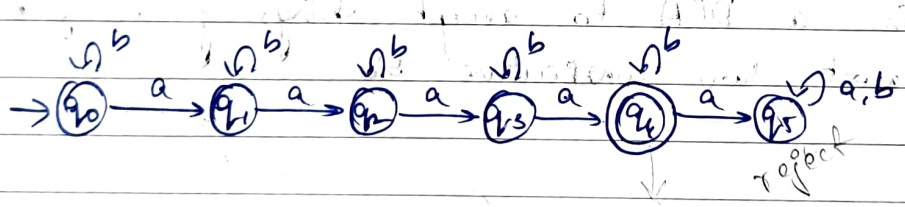
$L = \{ w(ab+ba) \mid w \in \{a,b\}^* \}$

$L = \{ ab, ba, aab, aba, bab, bba, \ldots \}$



10. Obtain a DFA to accept strings of a's & b's having four a's where $\Sigma = \{a,b\}$.

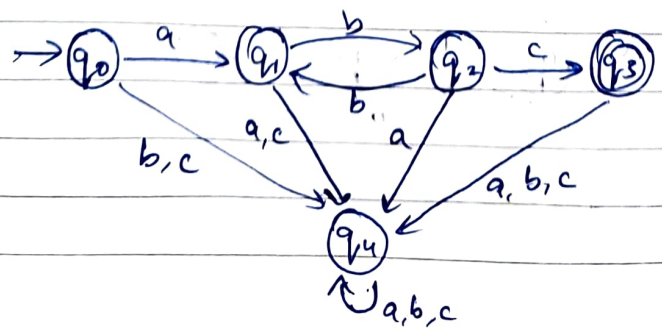
$L = \{ aaaa, aaaab, aaaba, aabaa, abaaa, baaaa \}$.



11. Obtain a DFA to accept strings of a's, b's & c's beginning with a 'a', followed by odd number of b's & ending with a 'c'.

a _____ c

$L = \{ abc, abbbc, abbbbbc, \ldots \}$



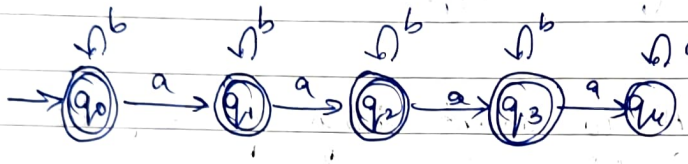
12. Obtain a DFA to accept strings of a's & b's with
(i) exactly one a
(ii) atleast one a
(iii) not more than 3 a's.

(i)

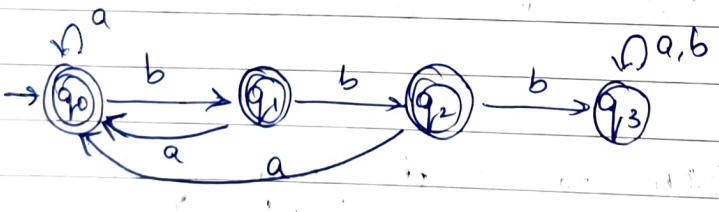

$L = \{a, aa, ab, ba, abb, bab, bba, \ldots\}$

(ii)

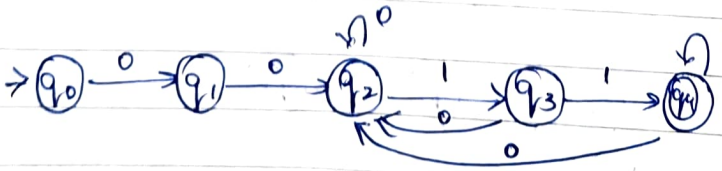

$L = \{a, ab, ba, aa, bbb, \ldots\}$

(iii)


13. Obtain a DFA to accept strings of a's & b's with at most two consecutive b's

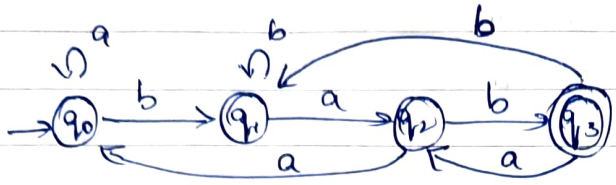


14. Obtain a DFA to accept strings of 0's & 1's starting with at least two 0's & ending with at least two 1's

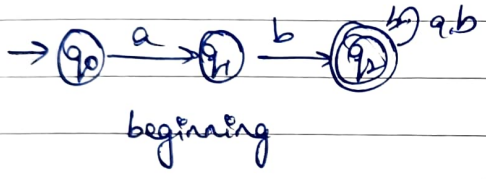


15. Obtain a DFA to accept the language $L = \{ wbab \mid w \in \{a,b\}^* \}$

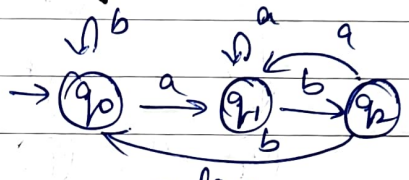
$L = \{ bab, abab, bbab, aabab, abbab, babab, bbbab, \ldots \}$



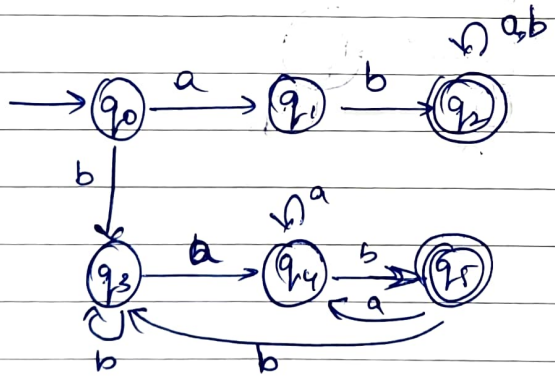
16. Draw a DFA to accept set of all strings on the alphabet $\Sigma = \{a,b\}$ that either begins or ends or both with the substring ab.



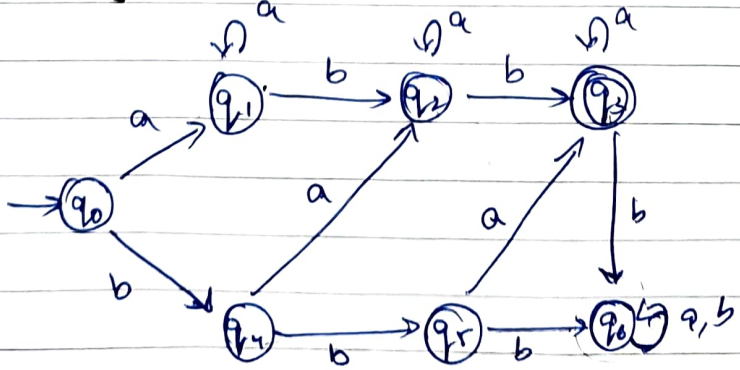
beginning



ending

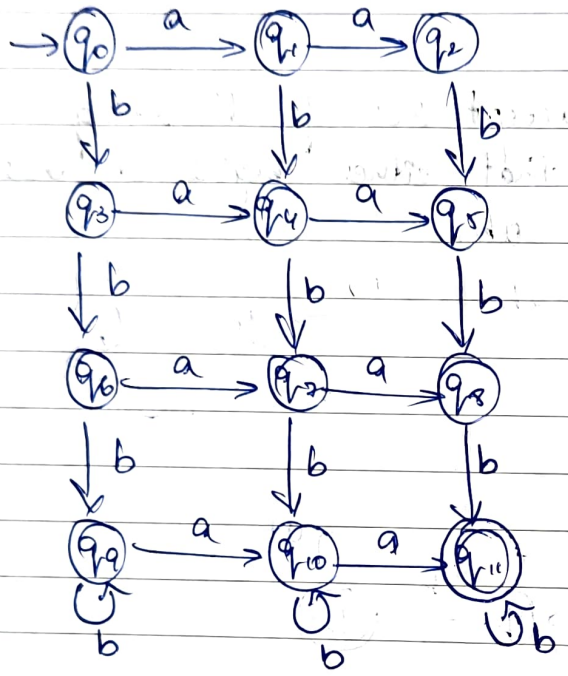


17. Draw a DFA to accept the language $L = \{ w : n_a(w) \geq 1, n_b(w) = 2 \}$



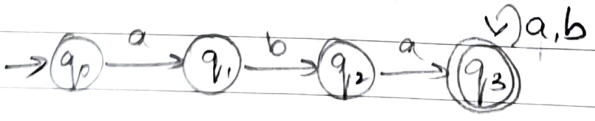
18. Draw a DFA to accept the language L
$L = \{ w : n_a(w) = 2^n, n_b(w) \geq 0, n \geq 3 \}$

L = { aabbb, ababb, abbab, abbba, baabb, babab, babba
bbaab, bbaba, bbbaa.

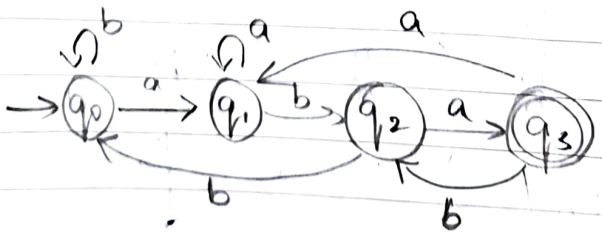


(i) starting with aba (ii) ending with aba

(i)



(ii)



(*) Divisible by K problems.

transitions can be obtained by following relations.

$$\delta(q_i, a) = q_j \quad \text{where} \quad j = (r * i + d) \bmod k.$$

r → radix of input (for binary, $r = 2$)
i → remainder obtained after dividing by k.
d → digits (for binary $d = \{0, 1\}$)
k → divisor.

Step 1 : Identify the radix, input alphabet & divisor k

Step 2 : Compute the possible remainders : These remainders represents states of DFA.

Step 3 : Find transition using above function.

Step 4 : Construct DFA using these transitions.

19. Construct a DFA which accepts strings of 0's & 1's where the value of each string is represented as binary number. Only the strings representing zero modulo five should be accepted
For eg: 0000, 0101, 1010, 1111,

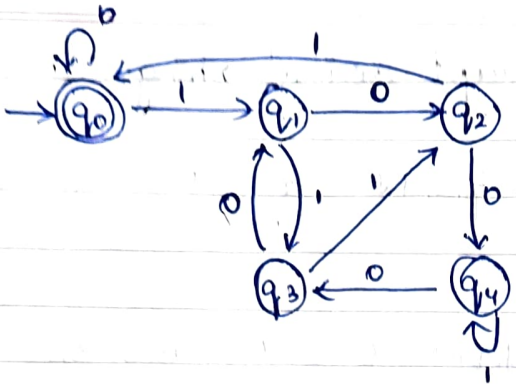
Zero modulo five => 0 is remainder after dividing by 5

$Q = Q$
$\Sigma = \{0,1\}$
$k = 5$
$i = 0,1,2,3,4.$

0 → 0000 | 20 → 10100
5 → 0101 | 25 → 11001
10 → 1010 | 30 → 11110
15 → 1111 | 35 → 100011
40 → 101000

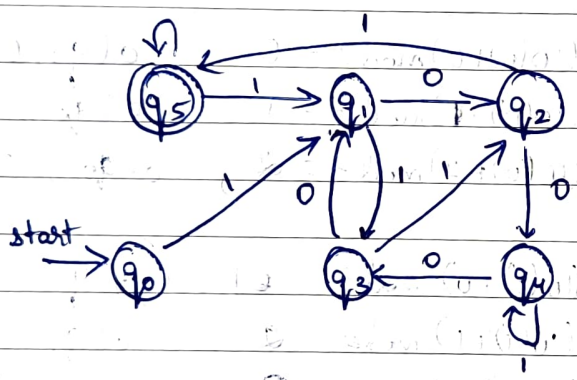
$\delta(q_i, a) = q_j$ where $j = (r * i + d) \bmod k$
$j = (2i + d) \bmod 5$

remainder	d	$(2i+d) \bmod 5 = j$	$\delta(q_i, d) = q_j$
$i = 0$	0	$(2(0)+0) \bmod 5 = 0$	$\delta(q_0, 0) = q_0$
	1	$(2(0)+1) \bmod 5 = 1$	$\delta(q_0, 1) = q_1$
$i = 1$	0	$(2(1)+0) \bmod 5 = 2$	$\delta(q_1, 0) = q_2$
	1	$(2(1)+1) \bmod 5 = 3$	$\delta(q_1, 1) = q_3$
$i = 2$	0	$(2(2)+0) \bmod 5 = 4$	$\delta(q_2, 0) = q_4$
	1	$(2(2)+1) \bmod 5 = 0$	$\delta(q_2, 1) = q_0$
$i = 3$	0	$(2(3)+0) \bmod 5 = 1$	$\delta(q_3, 0) = q_1$
	1	$(2(3)+1) \bmod 5 = 2$	$\delta(q_3, 1) = q_2$
$i = 4$	0	$(2(4)+0) \bmod 5 = 3$	$\delta(q_4, 0) = q_3$
	1	$(2(4)+1) \bmod 5 = 4$	$\delta(q_4, 1) = q_4$



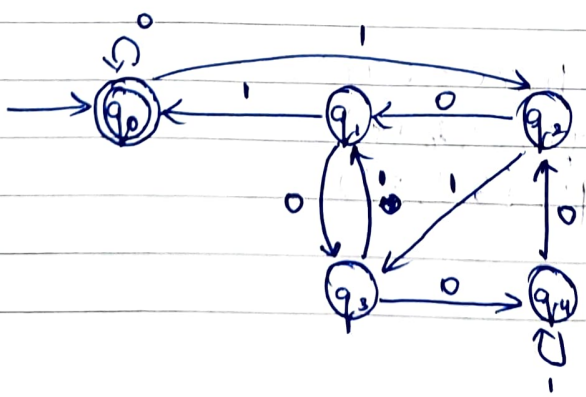
20. Obtain a DFA which accepts the set of all strings beginning with a 1 that when interpreted as a binary integer, is a multiple of 5.

Eg : 101, 1010, 1111, ...



21. Obtain a DFA that accepts set of all strings that, when interpreted in reverse as a binary integer, is divisible by 5.

reverse the direction of arcs. check question 19



22. Obtain a DFA to accept decimal strings divisible by 3

$d = \{0,1,2,3,4,5,6,7,8,9\}$, $r = 10$ $k = 3$

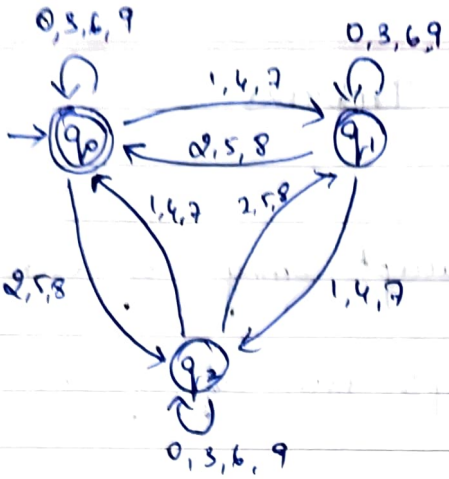
$i = 0,1,2.$

Grouping the digits from 0 to 9 based on remainder

$\{0,3,6,9\}$ with 0 as remainder
$\{1,4,7\}$ with 1 as remainder
$\{2,5,8\}$ with 2 as remainder

remainder	d	$(10i+d) \bmod 3 = j$	$\delta(q_i, d) = q_j$
$i=0$	0	$(10(0)+0) \bmod 3 = 0$	$\delta(q_0, 0) = q_0$
	1	$(10(0)+1) \bmod 3 = 1$	$\delta(q_0, 1) = q_1$
	2	$(10(0)+2) \bmod 3 = 2$	$\delta(q_0, 2) = q_2$
$i=1$	0	$(10(1)+0) \bmod 3 = 1$	$\delta(q_1, 0) = q_1$
	1	$(10(1)+1) \bmod 3 = 2$	$\delta(q_1, 1) = q_2$
	2	$(10(1)+2) \bmod 3 = 0$	$\delta(q_1, 2) = q_0$
$i=2$	0	$(10(2)+0) \bmod 3 = 2$	$\delta(q_2, 0) = q_2$
	1	$(10(2)+1) \bmod 3 = 0$	$\delta(q_2, 1) = q_0$
	2	$(10(2)+2) \bmod 3 = 1$	$\delta(q_2, 2) = q_1$

$\delta(q_i, 0) \Rightarrow \delta(q_i, \{0,3,6,9\})$
$\delta(q_i, 1) \Rightarrow \delta(q_i, \{1,4,7\})$
$\delta(q_i, 2) \Rightarrow \delta(q_i, \{2,5,8\})$


0,3,6,9
0,3,6,9
1,4,7
2,5,8
1,4,7
2,5,8
2,5,8
1,4,7
0,3,6,9

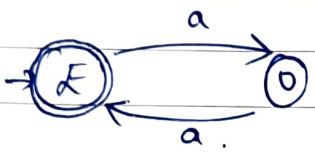
(*) Modulo K Counter Problems

Q3. Obtain a DFA to accept strings of even number of a's

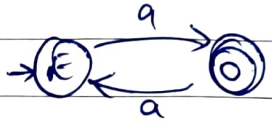
Case 0 : String accepts even no. of a's (E)

Case 1 : Strings accepts odd no of a's (O)

$\delta(E, a) = O$ & $\delta(O, a) = E$



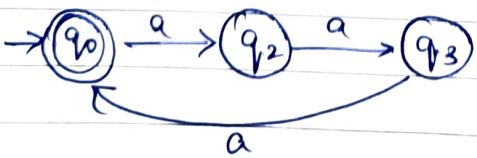
DFA accept even no of a's



DFA accept odd no of a's

Q4. Obtain a DFA to accept language $L = \{w : |w| \bmod 3 = 0\}$ where $\Sigma = \{a\}^*$

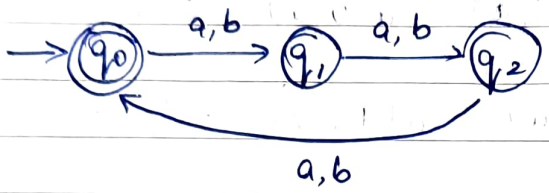
Case 0 : remainder is 0 (q_0)
Case 1 : remainder is 1 (q_1)
Case 2 : remainder is 2 (q_2)



Q5. Obtain a DFA to accept language $L = \{w : |w| \bmod 3 = 0\}$ on $\Sigma = \{a, b\}$.

L = contains strings of a's & b's whose length is a multiple of 3

L = { ∈, aaa, aab, aba, abb, baa, bab, bba, bbb, aabaaa, ... }



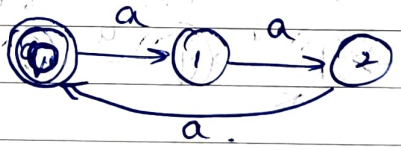
Q6. Obtain a DFA to accept the following language L

L = { w such that

(a) $|w| \bmod 3 \geq |w| \bmod 2$ where $w \in \Sigma^*$ and $\Sigma = \{a\}$

(b) $|w| \bmod 3 \neq |w| \bmod 2$ where $w \in \Sigma^*$ and $\Sigma = \{a\}$

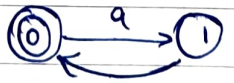
$|w| \bmod 3 = 0$



M1

Q_1 (remainder) = {0, 1, 2}

$|w| \bmod 2 = 0$



Q_2 (remainder) = {0, 1}

The transitions of DFA which has strings of w with $|w| \bmod 3$ & $|w| \bmod 2$ can obtained by taking the cross product of Q1 & Q2

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

M1 M2

$q_1 = \delta((0,0), a) = (\delta(0,a), \delta(0,a)) = (1,1)$

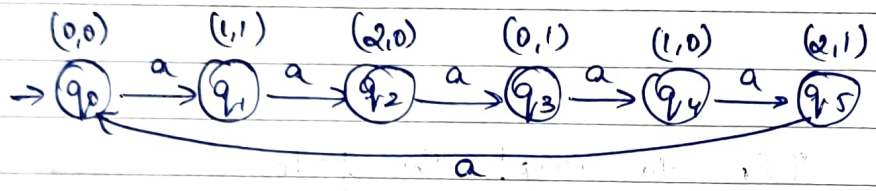
$q_2 = \delta((1,1), a) = (\delta(1,a), \delta(1,a)) = (2,0)$

$q_3 = \delta((2,0), a) = (\delta(2,a), \delta(0,a)) = (0,1)$

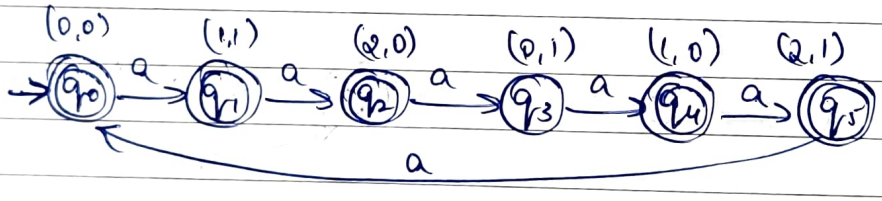
$q_4 = \delta((0,1), a) = (\delta(0,a), \delta(1,a)) = (1,0)$

$q_5 = \delta((1,0), a) = (\delta(1,a), \delta(0,a)) = (2,1)$

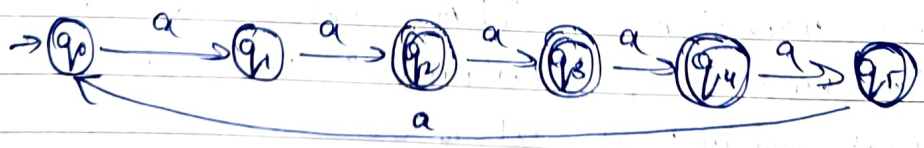
$q_6 = \delta((2,1), a) = (\delta(2,a), \delta(1,a)) = (0,0)$



(a) $(x, y) \Rightarrow x \geq y$ are final states.



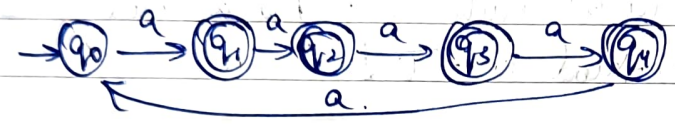
(b)



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

Remainders = {0, 1, 2, 3, 4}

Case 0 : Remainder is 0 ⇒ q_0
Case 1 : Remainder is 1 ⇒ q_1
Case 2 : Remainder is 2 ⇒ q_2
Case 3 : Remainder is 3 ⇒ q_3
Case 4 : Remainder is 4 ⇒ q_4

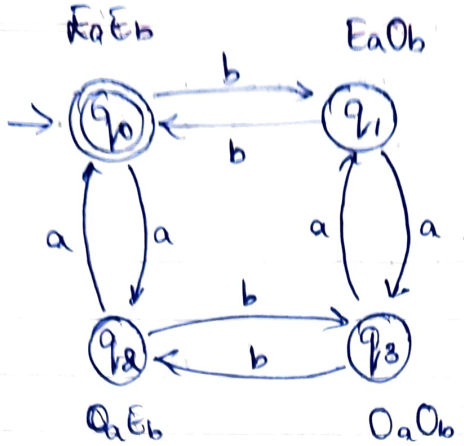


28. Obtain a DFA to accept strings of a's & b's having even number of a's and even number of b's.

Even → E Odd → O

Case 0 : strings having Even a's & Even b's ⇒ q_0
Case 1 : Ea & Ob ⇒ q_1
Case 2 : Oa & Eb ⇒ q_2
Case 3 : Oa & Ob ⇒ q_3

$\delta(E_a, a) = O_a$ $\delta(E_b, b) = O_b$
$\delta(O_a, a) = E_a$ $\delta(O_b, b) = E_b$

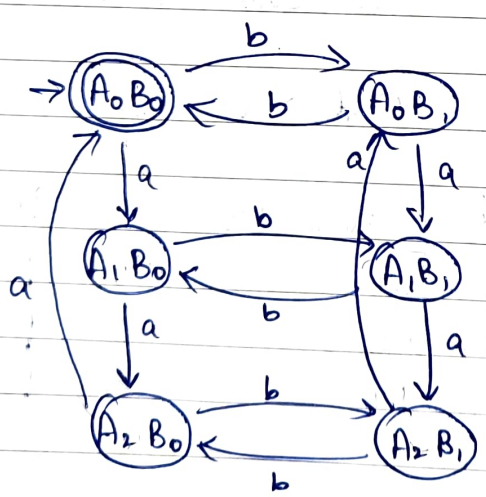


Q9. Obtain a DFA to accept strings of a's & b's such that

$L = \{ w \mid w \in (a+b)^* \text{ such that } N_a(w) \bmod 3 = 0 \text{ \& } N_b(w) \bmod 2 = 0 \}$

$N_a(w) \bmod 3 = 0 \Rightarrow Q_1 = \{A_0, A_1, A_2\}$

$N_b(w) \bmod 2 = 0 \Rightarrow Q_2 = \{B_0, B_1\}$

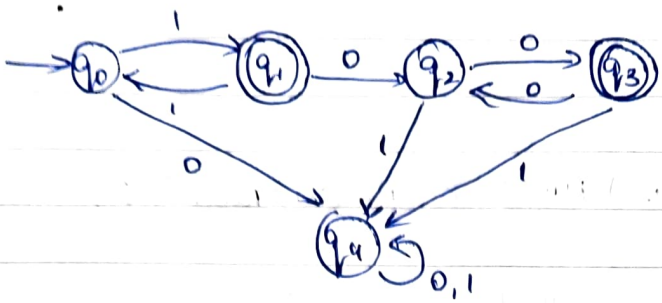


$L = \{ N_a(w) \bmod 3 = 1, N_b(w) \bmod 2 = 0 \} \Rightarrow A_1B_0$

$L = \{ N_a(w) \bmod 3 = 2, N_b(w) \bmod 2 = 0 \} \Rightarrow A_2B_0$

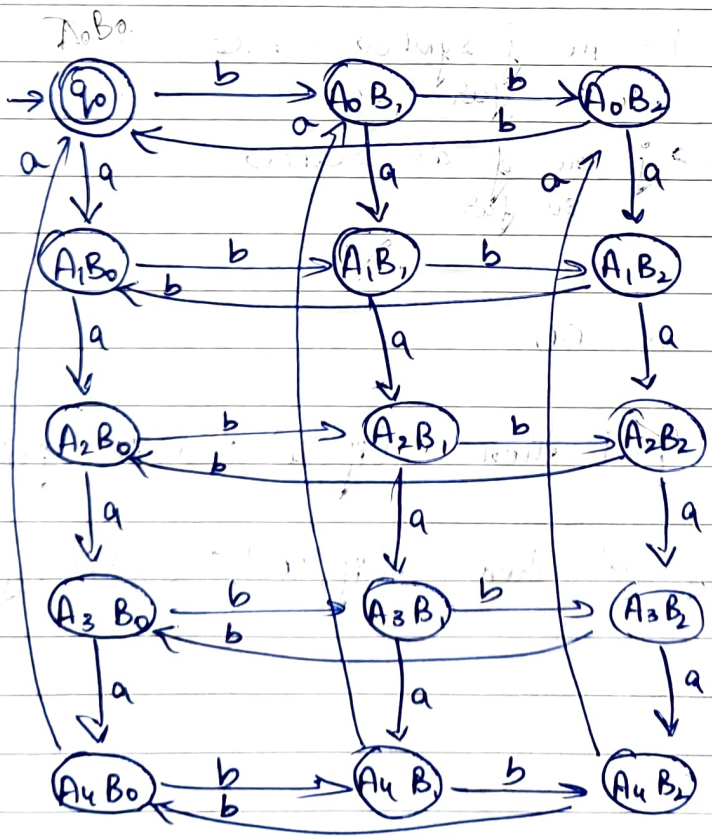
$L = \{ N_a(w) \bmod 3 = 2, N_b(w) \bmod 2 = 1 \} \Rightarrow A_2B_1$

30. Draw a DFA to accept language $L = \{w : w$ has odd number of 1's followed by even number of 0's$\}$.



31 Obtain a DFA to accept strings of a's & b's such that the number of a's is divisible by 5 & no of b's is divisible by 3.

$Q_1 = \{ A_0, A_1, A_2, A_3, A_4 \}$
$Q_2 = \{ B_0, B_1, B_2 \}$.



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

$F \neq \{A_0B_0, A_1B_1, A_2B_2\}$ rest all final state.

(*) $L_w = \{ w : n_a(w) \bmod 5 > n_b(w) \bmod 3 \}$.

$F = \{ A_xB_y$ such that $x > y \}$.

$= \{ A_1B_0, A_2B_0, A_2B_1, A_3B_0, A_3B_1, A_3B_2, A_4B_0, A_4B_1, A_4B_2 \}$.

32. Obtain a DFA to accept string of a's & b's such that each block of 5 consicutive ~~numbers~~ symbols have atleast 2 a's.

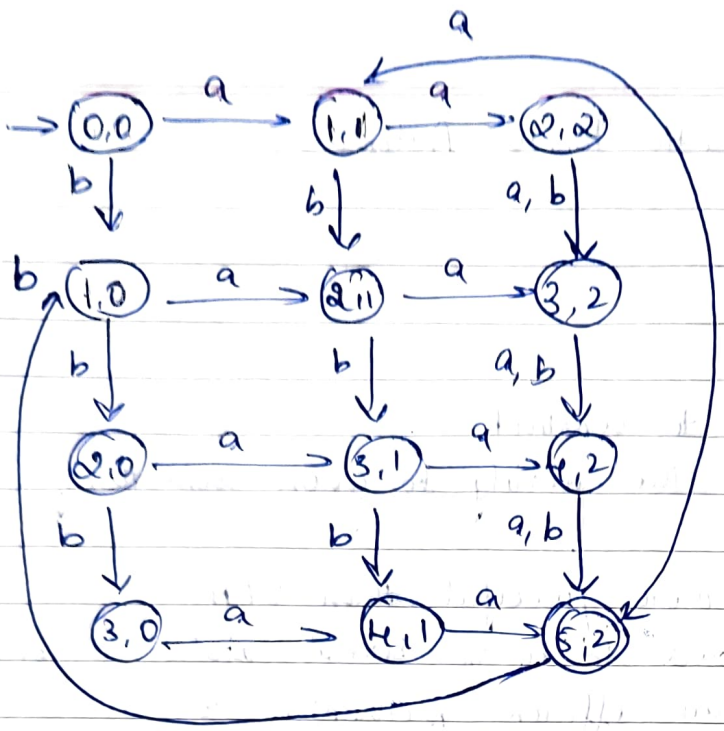
(ij) → i = no of symbols scanned so far.

→ j = no of a's scanned so far.

start state is 00.

if $i \leq 4$ & $j \leq 1$ then $\delta(ij, a) = \delta(i+1, j+1)$

if $i \leq 4$ & $j = 2$ then $\delta(ij, x) = \delta(i+1, j)$.



Disadvantages of DFA.

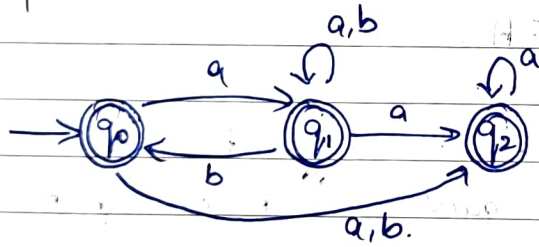
- Constructing a DFA is difficult
- The DFA cannot guess about its input
- The DFA is not very powerful.
- At any point of time, the DFA is in only one state.

Non-Deterministic Finite Automata.

Advantages :

- Very easy to construct
- NFA has the ability to guess something about its input
- NFA is more poweeful than DFA
- It has power to be in several states at once.
- NFA is an efficient mechanism to describe some complicated languages concisely

Example :



- States : $Q = \{q_0, q_1, q_2\}$.

power set of Q (2^Q) is a set of all subsets of Q.

$2^Q = \{\{\}, \{q_0\}, \{q_1\}, \{q_2\}, \{q_0 q_1\}, \{q_1 q_2\}, \{q_0 q_2\}, \{q_0, q_1, q_2\}\}$

- Input alphabet : $\Sigma = \{a, b\}$

- Transition table

$\delta(q_i, a) = q_j$ or $\delta(q_i, a) = \{q_i, q_j\}$.

	a	b
q_0	$\{q_1, q_2\}$	$\{q_2$
q_1	$\{q_1, q_2\}$	$\{q_0, q_1\}$
q_2	q_2	-

$\delta: Q \times \Sigma$ to 2^Q

or

$\delta(q, a) = \{p_1, p_2, \ldots, p_n\}$.

$\delta \rightarrow$ Transition function
$q \rightarrow$ current state of machine
$a \rightarrow$ input
$p_1, p_2, \ldots, p_n \rightarrow$ possible states of machine.

Definition:

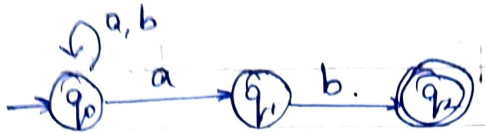
The non deterministic finite automaton in short NFA is 5-tuple or quintuple indicating five components.

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

where

M is name of machine
Q is non-empty, finite set of states
Σ is non-empty, finite set of input alphabets
$\delta: Q \times \Sigma$ to 2^Q
$q_0 \in Q$ is the start state
$F \subseteq Q$ is set of accepting of final states.

Eg ②

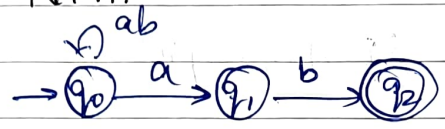


	a	b
q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\{\phi\}$	$\{q_2\}$
q_2	$\{\phi\}$	$\{\phi\}$

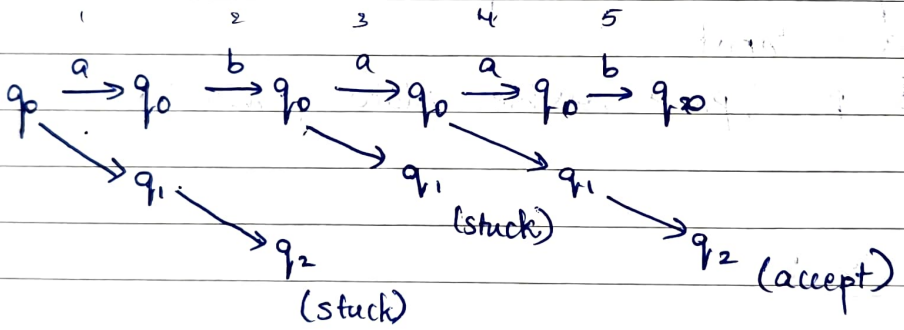
$\phi \rightarrow$ empty set

Moves made by NFA.

for TD,



① abaab



Extended Transition function of NFA. to string.

Prefix	
ε	$\delta^*(q_0, \varepsilon) = \{q_0\}$
a	$\delta^*(q_0, a) = \delta(\delta^*(q_0, \varepsilon), a) = \delta(q_0, a) = \{q_0, q_1\}$
ab	$\delta^*(q_0, ab) = \delta(\delta^*(q_0, a), b) = \delta(q_0, b) \cup \delta(q_1, b) = \{q_0\} \cup \{q_2\}$
aba	$\delta^*(q_0, aba) = \delta(\delta^*(q_0, ab), a) = \delta(q_0, a) \cup \delta(q_2, a) = \{q_0, q_1\} \cup \{\phi\}$

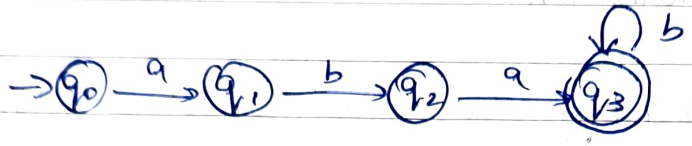
Problems :

① Obtain a NFA to accept the following language

$L = \{w \mid w \in abab^n \text{ or } aba^n \text{ where } n \geq 0\}$

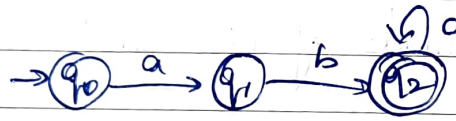
for $abab^n$

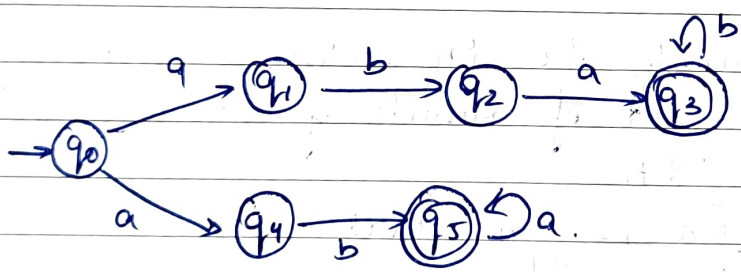
$L_1 = \{aba, abab, ababb, \ldots\}$.



for aba^n

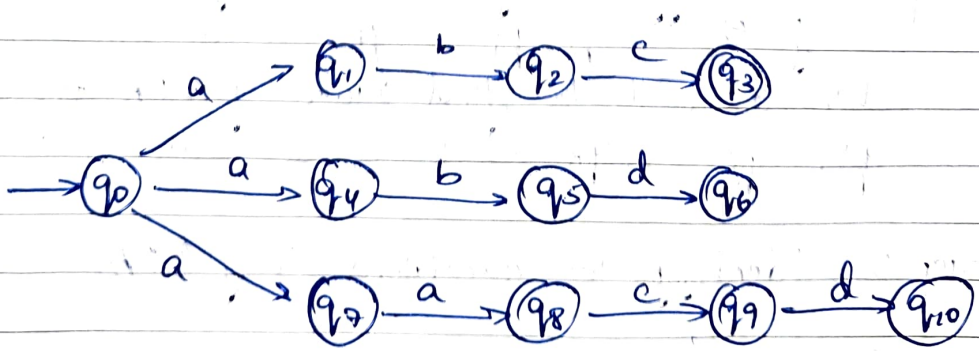
$L_2 = \{ab, aba, abaa, \ldots\}$.





2. Obtain a NFA to recognize the following strings abc, abd, aacd.

$= \{q_0, q_4\}$



Conversion of NFA to DFA.

NFA can be converted to DFA using two methods.

(1) Subset Construction
(2) Lazy Evaluation

(*) Subset Construction.

Step 1 : Identify the start state of DFA. Since q_0 is the start state of NFA, $\{q_0\}$ is start state of DFA.

Step 2 : Identify the input alphabets of DFA
The input alphabets of DFA are the input alphabets of NFA.

Step 3 : Identify the Q_D which are states of DFA.
The set of subsets of Q_N will be the states of DFA
if Q_N has n states,
$\boxed{Q_D = 2^n}$

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

Step 5 : Identify the transitions δ_D of DFA.

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

Example :

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

TF =

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

Step 1 : $\{q_0\}$ is start state of DFA

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

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

Step 4 : $F_D = \{\{q_2\}, \{q_0, q_2\}, \{q_1, q_2\}, \{q_0, q_1, q_2\}\}$.

Step 5 : δ_D

for state ϕ :

input symbol = a

$\delta_D(\phi, a) = \phi$

input symbol = b.

$\delta_D(\phi, b) = \phi$.

for state $\{q_0\}$

IS = a

$\delta_D(\{q_0\}, a) = \delta(q_0, a)$

$= \{q_0, q_1\}$

IS = b.

$\delta(\{q_0\}, b) = \delta(q_0, b)$

$= \{q_0\}$

for state $\{q_1\}$

$\delta_D(\{q_1\}, a) = \delta(q_1, a)$

$= \{\phi\}$

$\delta_D(\{q_1\}, b) = \delta(q_1, b)$

$= \{q_2\}$

for state $\{q_2\}$

$\delta_D(\{q_2\}, a) = \delta(q_2, a)$

$= \{\phi\}$

$\delta_D(\{q_2\}, b) = \delta(q_2, b)$

$= \{\phi\}$

for state $\{q_0, q_1\}$

$\delta_D(\{q_0, q_1\}, a) = \delta(q_0, a) \cup \delta(q_1, a)$

$= \{q_0, q_1\} \cup \phi$

$= \{q_0, q_1\}$

$\delta_D(\{q_0, q_1\}, b) = \delta(q_0, b) \cup \delta(q_1, b)$

$= \{q_0\} \cup \{q_2\}$

$= \{q_0, q_2\}$

for state $\{q_0, q_2\}$

$\delta_D(\{q_0, q_2\}, a) = \delta(q_0, a) \cup \delta(q_1, a)$

$= \{q_0, q_1\} \cup \{\phi\}$

$= \{q_0, q_1\}$

$\delta_D(\{q_0, q_2\}, b) = \delta(q_0, b) \cup \delta(q_2, b)$

$= \{q_0\} \cup \{\phi\}$

$= \{q_0\}$

for state $\{q_1, q_2\}$

$\delta_D(\{q_1, q_2\}, a) = \delta(q_1, a) \cup \delta(q_2, a)$

$= \{\phi\} \cup \{\phi\}$

$= \{\phi\}$

$\delta_D(\{q_1, q_2\}, b) = \delta(q_1, b) \cup \delta(q_2, b)$

$= \{q_2\} \cup \{\phi\}$

$= \{q_2\}$

for state $\{q_0, q_1, q_2\}$

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

$$\delta_D(\{q_0,q_1,q_2\}, b) = \delta(q_0,b) \cup \delta(q_1,b) \cup \delta(q_2,b)$$
$$= \{q_0\} \cup \{q_2\} \cup \{\phi\}$$
$$= \{q_0,q_2\}.$$

Transition table :

	δ	a	b
	ϕ	ϕ	ϕ
→	q_0	$\{q_0,q_1\}$	$\{q_0\}$
	q_1	ϕ	$\{q_2\}$
*	q_2	ϕ	ϕ
	~~q_0q_1~~	~~$\{q_0,q_1\}$~~	~~$\{q_0,q_2\}$~~
	q_0q_1	$\{q_0,q_1\}$	$\{q_0,q_2\}$
*	q_0q_2	$\{q_0,q_1\}$	$\{q_0\}$
*	q_1q_2	ϕ	$\{q_2\}$
*	$q_0q_1q_2$	$\{q_0,q_1\}$	$\{q_0,q_2\}$

⇒

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

Even though we have 8 states, observe from table that B is the start state.
The states reachable from B are B, E, F. Rest of the states are non-reachable & hence can be eliminated.

(*) Conversion of NFA to DFA using Lazy Evaluation method.

Step 1 : Identify the start state of DFA
Since q_0 is start state of NFA, $\{q_0\}$ is the start state of DFA.

Step 2 : Identify the alphabets of DFA.
Input alphabets of DFA are the input alphabets of NFA. $\Sigma = \{a, b\}$.

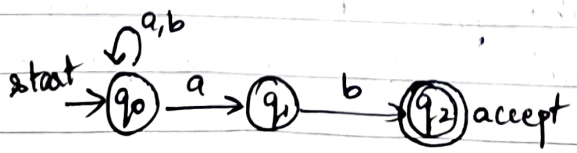
Step 3 : Identify the transitions δ_D
for state $\{q_i, q_j \ldots q_k\}$ in Q_D & for each input in Σ,

$$\delta_D(\{q_i, q_j, \ldots q_k\}, a) = \delta(q_i, a) \cup \delta(q_j, a) \cup \ldots \cup \delta(q_k, a)$$
$$= \{q_e, q_m \ldots q_n\} \text{ say}$$

add state $\{q_e, q_m, \ldots q_n\}$ to Q_D if not present

Step 4 : Identify final states of DFA.
If $\{q_i, q_j, \ldots, q_k\}$ is a state in Q_D and if one of $q_i, q_j, \ldots, q_k$ is the final state of NFA then $\{q_i, q_j, \ldots q_k\}$ will be the final states of DFA.

Example : Obtain the DFA for the following NFA using lazy evaluation method.



Transition function / table.

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

Step 1 : $\{q_0\}$ is start state of DFA

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

Step 3 :

for state $\{q_0\}$

$\delta_D(\{q_0\}, a) = \{q_0, q_1\}$

$\delta_D(\{q_0\}, b) = \{q_0\}$

for state $\{q_0, q_1\}$

$\delta_D(\{q_0, q_1\}, a) = \delta_N(q_0, a) \cup \delta_N(q_1, a)$
$= \{q_0, q_1\} \cup \{\phi\}$
$= \{q_0, q_1\}$

$\delta_D(\{q_0, q_1\}, b) = \delta_N(q_0, b) \cup \delta_N(q_1, b)$
$= \{q_0\} \cup \{q_2\}$
$= \{q_0, q_2\}$

for state $\{q_0, q_2\}$

$\delta_D(\{q_0, q_2\}, a) = \delta_N(q_0, a) \cup \delta_N(q_2, a)$
$= \{q_0, q_1\} \cup \{\phi\}$
$= \{q_0, q_1\}$

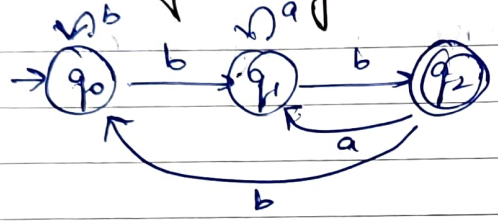
$\delta_D(\{q_0, q_2\}, b) = \delta_N(q_0, b) \cup \delta_N(q_1, b)$
$= \{q_0\} \cup \{\phi\}$
$= \{q_0\}$.

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

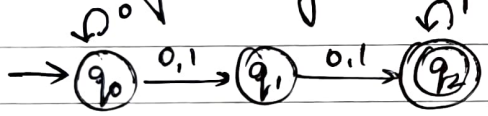
⇒

	a	b
→ A	B	A
B	B	C
* C	B	A

Q. Convert the following NFA to its equi



Q. Convert the following NFA to its equivalent DFA



transition table :

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

Step 1 : $\{q_0\}$ is start state of DFA

Step 2 : $\Sigma = \{0, 1\}$

Step 3 : $Q_0 = \{\{q_0\}\}$

for state q_0

$\delta_D(\{q_0\}, 0) = \delta_N(q_0, 0)$
$= \{q_0, q_1\}$ - (1)

$\delta_D(\{q_0\}, 1) = \delta_N(q_0, 1)$
$= \{q_1\}$ - (2)

→ $Q_D = \{\{q_0\}, \{q_0, q_1\}, \{q_1\}\}$

for state $\{q_0, q_1\}$

$\delta_D(\{q_0, q_1\}, 0) = \delta_N(q_0, 0) \cup \delta_N(q_1, 0)$
$= \{q_0, q_1\} \cup \{q_2\}$
$= \{q_0, q_1, q_2\}$ - (3)

$\delta_D(\{q_0, q_1\}, 1) = \delta_N(q_0, 1) \cup \delta_N(q_1, 1)$
$= \{q_1\} \cup \{q_2\}$
$= \{q_1, q_2\}$ - (4)

→ $Q_D = \{\{q_0\}, \{q_0, q_1\}, \{q_1\}, \{q_0, q_1, q_2\}, \{q_1, q_2\}\}$

for state $\{q_1\}$

$\delta_D(\{q_1\}, 0) = \delta_N(q_1, 0)$
$= \{q_2\}$ - (5)

$\delta_D(\{q_1\}, 1) = \delta_N(q_1, 1)$
$= \{q_2\}$

→ $Q_D = \{\{q_0\}, \{q_0, q_1\}, \{q_0\}, \{q_0, q_1, q_2\}, \{q_1, q_2\}, \{q_2\}\}$

for state $\{q_0, q_1, q_2\}$

$\delta_D(\{q_0, q_1, q_2\}, 0) = \delta_N(q_0, 0) \cup \delta_N(q_1, 0) \cup \delta_N(q_2, 0)$
$= \{q_0, q_1\} \cup \{q_2\} \cup \{\phi\}$
$= \{q_0, q_1, q_2\}$

$\delta_D(\{q_0, q_1, q_2\}, 1) = \delta_N(q_0, 1) \cup \delta_N(q_0, 1) \cup \delta_N(q_2, 1)$
$= \{q_1\} \cup \{q_2\} \cup \{q_2\}$
$= \{q_1, q_2\}$.

→ Q_D remains same.

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

for state $\{q_1, q_2\}$

$\delta_D(\{q_1,q_2\},0) = \delta_N(q_1,0) \cup \delta_N(q_2,0)$
$= \{q_2\} \cup \{\phi\}$
$= \{q_2\}$

$\delta_D(\{q_1,q_2\},1) = \delta_N(q_1,1) \cup \delta_N(q_2,1)$
$= \{q_2\} \cup \{q_2\}$
$= \{q_2\}$

→ $Q_D = \{ \{q_0\}, \{q_0,q_1\}, \{q_1\}, \{q_0,q_1,q_2\}, \{q_1,q_2\}, \{q_2\} \}$

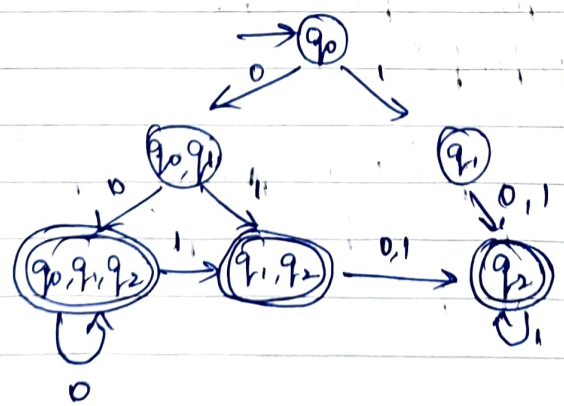
for state $\{q_2\}$

$\delta_D(\{q_2\},0) = \delta_N(q_2,0)$
$= \{\phi\}$

$\delta_D(\{q_2\},1) = \delta_N(q_2,1)$
$= \{q_2\}$.

→ $Q_D = \{ \{q_0\}, \{q_0,q_1\}, \{q_1\}, \{q_0,q_1,q_2\}, \{q_1,q_2\}, \{q_2\} \}$

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



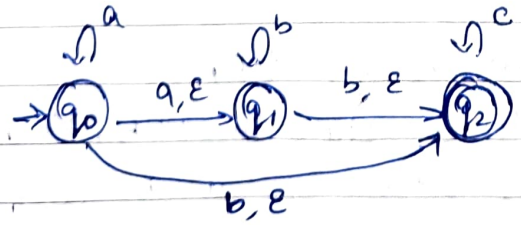
Hold Obtain an NFA to accept strings of a's & b's ending with ab or ba. From this obtain an equivalent DFA.

ε - NFA (Finite Automata with Epsilon Transition)

- A transition with an empty input string is called ε - transition.
  (If there is a transition from one state to another state without any input.

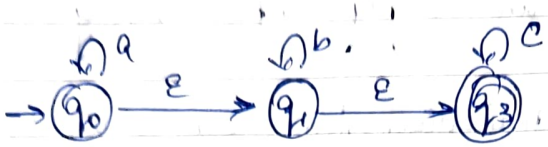
- The ε - NFA is a 5-tuple indicating ~~the~~ five components:
  $M = (Q, \Sigma, \delta, q_0, F)$
  where Q is non-empty, finite set of states
  Σ is non-empty finite set of input symbol.
  $\delta: Q \times (\Sigma \cup \varepsilon)$ to 2^Q
  $q_0 \in Q$ is the start state
  $F \subseteq Q$ is the set of final states.

- The ε - Closure of q denoted by EClosure (q) is the set of all states which are reachable from q on ε - transitions only. It is recursively defined as
  - state q is in ECLOSURE(q) i.e ECLOSURE(q) = q.
  - If ECLOSURE(q) contains p & if there is a transition from state p to state r labeled ε, then state r is also in ECLOSURE(q).



: ECLOSURE(q_0) = $\{q_0, q_1, q_2\}$
ECLOSURE(q_1) = $\{q_1, q_2\}$
ECLOSURE(q_2) = $\{q_2\}$.

① 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.



	a	b	c	ε
q_0	q_0	–	–	q_1
q_1	–	q_1	–	q_2
q_2	–	–	q_2	–

$M = (Q, \Sigma, \delta, S, F)$

where

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

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

δ refer transition table

$S = q_0$

$F = \{q_2\}$

Conversion of ε-NFA to DFA.

Step 1 : If q_0 is the start state of NFA, then ECLOSURE(q_0) is the start state of DFA.

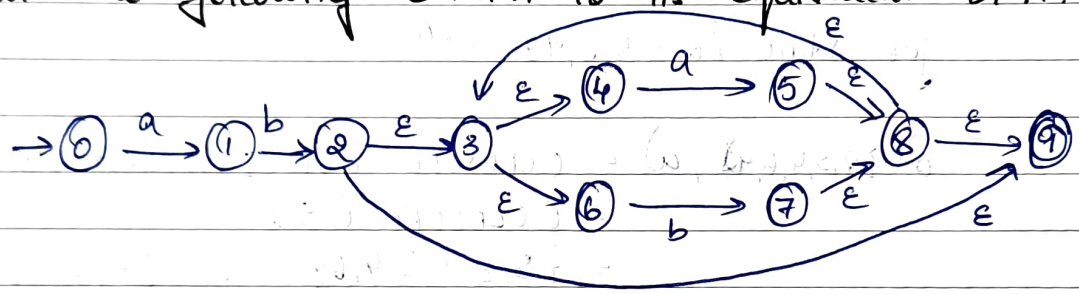
Step 2 : Compute the transitions for DFA.
Let $\{q_i, q_j, \ldots, q_k\}$ is a state in DFA. Then, the transitions from this state on a is computed as.

$\delta_E(\{q_i, q_j, \ldots, q_k\}, a) = \{p_1, p_2, \ldots, p_n\}$
ECLOSURE $(\{p_1, p_2, p_3, \ldots, p_n\})$

$\delta_D(\{q_i, q_j, \ldots q_k\}, a) = \text{ECLOSURE}(\{p_1, p_2, \ldots, p_m\})$

Step 3: If $\{q_i, q_j, \ldots q_k\}$ is a state in DFA and if this set contains atleast one final state of ε-NFA, then $\{q_i, q_j, \ldots, q_k\}$ is a final state of DFA.

① Convert the following ε-NFA to its equivalent DFA.



Step 1: Since 0 is start state of ε-NFA, ECLOSURE of 0 is start state of DFA.

$\text{ECLOSURE}(0) = \{0\}$

for state $\{0\}$, — (A)

$\delta(A, a) =$

$\delta(0, a) = \text{ECLOSURE}(\delta_E(0, a))$

$= \text{ECLOSURE}(\phi)$

$= \{\}$.

$\delta(A, b)$

$\delta(0, b) = \text{ECLOSURE}(\delta_E(0, b))$

$= \text{ECLOSURE}(\phi)$

$= \{\phi\}$

for state 1, → (B)

$\delta(B, a)$

$\delta(1, a)$ = ECLOSURE ($\delta_E(1, a)$)
= ECLOSURE (ϕ)
= $\{\phi\}$

$\delta(B, b)$

$\delta(1, b)$ = ECLOSURE ($\delta_E(1, b)$)
= ECLOSURE ($\{2\}$)
= $\{2, 3, 4, 6, 9\}$

for state $\{2,3,4,6,9\}$ → (C)

$\delta(C, a)$

$\delta(\{2,3,4,6,9\}, a)$ = ECLOSURE ($\delta_E(\{2,3,4,6,9\}, a)$
= ECLOSURE (5)
= $\{5, 8, 9, 3, 4, 6\}$ or $\{3, 4, 5, 6, 8, 9\}$

$\delta(C, b)$

$\delta(\{2,3,4,6,9\}, b)$ = ECLOSURE ($\delta_E(\{2,3,4,6,9\}, b)$
= ECLOSURE (7)
= $\{7, 8, 9, 3, 4, 6\}$ or $\{3, 4, 6, 7, 8, 9\}$

for state $\{3,4,5,6,8,9\}$ → (D)

$\delta(D, a)$

$\delta(\{3,4,5,6,8,9\}, a)$ = ECLOSURE ($\delta_E(\{3,4,5,6,8,9\}, a)$
= ECLOSURE (5)
= $\{3, 4, 5, 6, 8, 9\}$

$\delta(D, b)$

$\delta(\{3,4,5,6,8,9\}, b)$ = ECLOSURE ($\delta_E(\{3,4,5,6,8,9\}, b)$
= ECLOSURE (7)
= $\{3, 4, 6, 7, 8, 9\}$

for state $\{3,4,6,7,8,9\} \rightarrow (E)$.

$\delta(\{3,4,6,7,8,9\}, a)$

$\delta(\{3,4,6,7,8,9\}, a) = \text{ECLOSURE}(\delta_E(\{3,4,6,7,8,9\}, a))$

$= \text{ECLOSURE}(\{5\})$

$= \{3,4,5,6,8,9\}$.

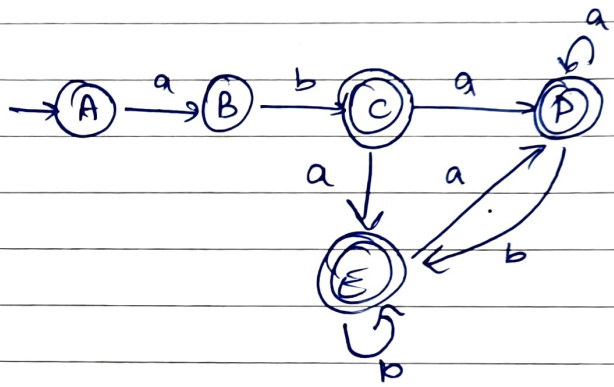
$\delta(\{3,4,6,7,8,9\}, b)$

$\delta(\{3,4,6,7,8,9\}, b) = \text{ECLOSURE}(\delta_E(\{3,4,6,7,8,9\}, b))$

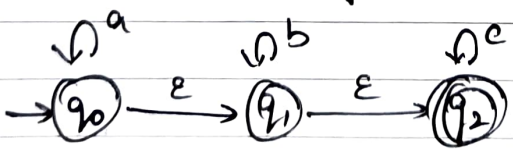
$= \text{ECLOSURE}(\{7\})$

$= \{3,4,6,7,8,9\}$.

	a	b
→ A	B	ϕ
B	ϕ	C
* C	D	E
* D	D	E
* E	D	E



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



Applications of FA

- Design of digital circuits: The FA is used during designing and checking the behavior of the digital circuits using software. The FA is very useful in hardware design such as circuit verification, in design of automatic traffic signals etc.
- Compiler construction: Used in the design of *lexical analyzer* (the first phase of compiler design) which breaks the input text into various units such as identifiers, keywords, punctuation etc.
- String matching: In designing a software for identifying the words, phrases and other patterns in large bodies of text (such as collection of web pages).
- String processing: To write software for processing the natural language (Ex: Speech processing). Large natural vocabularies can be described which includes the applications such as spelling checkers and advisers, multi-language dictionaries, indenting the documents etc.
- Software design: In building the software to verify the systems having finite number of states (for example, communication protocols in computer networks).
- Other applications: The FA are used in variety of applications in Artificial intelligence and knowledge engineering, in game theory and games, computer graphics, linguistics, etc.

1.8. Disadvantages of DFA

Now, let us see "What are the various disadvantages of DFA?" The various disadvantages of DFA are listed below:

- Constructing a DFA is difficult
- The DFA cannot guess about its input
- The DFA is not very powerful
- At any point of time, the DFA is in only state. So, a DFA does not have the power to be in several states at once

1.9. Why NFA?

Computers are completely deterministic machines (DFA). The state of the computer can be predicted from the input and initial state. We cannot find a computer which is non-deterministic. In such case, the question is "Why non-deterministic Finite Automata?" All the disadvantages of DFA mentioned earlier can be overcome using Non-Deterministic Finite Automata (in short NFA or NDFA). The various advantages of NFA are:

- Very easy to construct

- A "non-deterministic" finite automaton has the ability to guess something about its input
- A "non-deterministic" finite automaton is more powerful than DFA
- It has the power to be in several states at once
- An NFA is an efficient mechanism to describe some complicated languages concisely

❖ **Definition:** The Deterministic Finite Automaton in short DFA is 5-tuple or quintuple (indicating five components):

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

where

- M is the name of the machine. It can also be called by any name.
- Q is non-empty, finite set of states.
- Σ is non-empty, finite set of input alphabets.
- $\delta : Q \times \Sigma$ to Q i.e., δ is transition function which is a mapping from $Q \times \Sigma$ to Q. Based on the current state and input symbol, the machine enters into another state.
- $q_0 \in Q$ is the start state.
- $F \subseteq Q$ is set of accepting or final states.

1.11.1. Conversion from NFA to DFA (Subset Construct Method)

Now, let us "Describe the subset construction procedure to convert an NFA into a DFA"

Procedure NFA_DFA: Given an NFA $M_N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ which accepts the language $L(M_N)$, we can find an equivalent DFA $M_D = (Q_D, \Sigma, \delta_D, q_0, F_D)$ such that $L(M_D) = L(M_N)$.

Step 1: Identify the start state of DFA: Since q_0 is the start state of NFA, $\{q_0\}$ is the start state of DFA.

Step 2: Identify the alphabets of DFA: The input alphabets of DFA are the input alphabets of NFA. So, $\Sigma = \{a, b\}$.

Step 3: Identify Q_D which are the states of DFA: The set of subsets of Q_N will be the states of DFA Q_D. So, if Q_N has n states then Q_D will have 2^n states. For example,

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

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

1 2 3 4 5 6 7 8

So, $|Q_D| = 8$

Note: In this example, the number of states of NFA = 3 and hence number of states of DFA = 8. If n is number of states of NFA the number of states of DFA will be 2^n.

In general, $\{q_i, q_j, \ldots, q_k\}$ is considered as a state in Q_D.

Step 4: Identify the final states of DFA: If $\{q_i, q_j, \ldots, q_k\}$ is a state in Q_D, then $\{q_i, q_j, \ldots, q_k\}$ will be final state of DFA provided one of $q_i, q_j, \ldots, q_k$ is the final state of NFA.

Step 5: Identify the transitions (i.e., δ_D) of DFA: For each state $\{q_i, q_j, \ldots, q_k\}$ in Q_D and for each input symbol a in Σ, the transition can be obtained as shown below:

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

Thus, DFA can be obtained using subset construction method.

1.11.3. Conversion from NFA to DFA (Lazy Evaluation Method)

Now, let us "Describe the lazy evaluation method to convert NFA into a DFA". The lazy evaluation method of obtaining a DFA from NFA is shown below:

Procedure NFA_DFA: Given an NFA $M_N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ which accepts the language L(MN), we can find an equivalent DFA $M_D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$ such that $L(M_D) = L(M_N)$.

Step 1: Identify the start state of DFA: Since q_0 is the start state of NFA, $\{q_0\}$ is the start state of DFA.

Step 2: Identify the alphabets of DFA: The input alphabets of DFA are the input alphabets of NFA. So, $\Sigma = \{a, b\}$.

Step 3: Identify the transitions (i.e., δ_D) of DFA: For each state $\{q_i, q_j, \ldots, q_k\}$ in Q_D and for each input symbol a in Σ, the transition can be obtained as shown below:

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

$$= [q_l, q_m, \ldots, q_n] \text{ say}$$

- Add the state $[q_l, q_m, \ldots, q_n]$ to Q_D, if it is not already in Q_D.
- Add the transitions from $[q_i, q_j, \ldots, q_k]$ to $[q_l, q_m, \ldots, q_n]$ on the input symbol *a*

Note: The step 3 has to be repeated for each state that is added to Q_D.

Step 4: Identify the final states of DFA: If $\{ q_i, q_j, \ldots, q_k\}$ is a state in Q_D and if one of $q_i, q_j, \ldots, q_k$ is the final state of NFA, then $\{q_i, q_j, \ldots, q_k\}$ will be the final state of DFA.

Thus, DFA can be obtained using lazy evaluation method.

❖ **Definition:** The ε-NFA is 5-tuple or quintuple indicating five components:

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

where

- **Q** is non-empty, finite set of states.
- Σ is non-empty, finite set of input alphabets.
- $\delta : Q \times (\Sigma \cup \varepsilon)$ to 2^Q i.e., δ is transition function which is a mapping from $Q \times (\Sigma \cup \varepsilon)$ to 2^Q. Based on the current state there can be a transition to other states with or without any input symbols.
- $q_0 \in Q$ - is the start state.
- $F \subseteq Q$ - is set of accepting or final states.

DFA	NFA	ε-NFA
The DFA is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$ where • Q is set of finite states • Σ is set of input alphabets • $\delta : Q \times \Sigma$ to Q • q_0 is the start state • $F \subseteq Q$ is set of final states	An NFA is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$ where • Q is set of finite states • Σ is set of input alphabets • $\delta : Q \times \Sigma$ to 2^Q • q_0 is the start state • $F \subseteq Q$ is set of final states	An ε-NFA is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$ where • Q is set of finite states • Σ is set of input alphabets • $\delta : Q \times (\Sigma \cup \varepsilon)$ to 2^Q • q_0 is the start state • $F \subseteq Q$ is set of final states
• There can be zero or one transition from a state on an input symbol	• There can be zero, one or more transitions from a state on an input symbol	• There can be zero, one or more transitions from a state with or without giving any input
• More number of transitions	• Less number of transitions	• Relatively more transitions when compared with NFA
• Difficult to construct	• Easy to construct	• Easy to construct using regular expressions
• Less powerful since at any point of time it will be in only one state	• More powerful than DFA since at any point of time it will be in more than one state	• More powerful than NFA since at any point of time it will be in more than one state with or without giving any input