DFA: $\delta : Q \times \Sigma \to Q$.

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

Present State	Present Input	Next State
q0	0	q1.

THEORY OF COMPUTATION

Introduction to finite Automata.

Finite automata are abstract machines (→ imaginary) used to recognize patterns in input sequences. They consist of states, transitions, input symbols, processing each symbol step-by-step.

* If the machine ends in an accepting state after processing the input, it is accepted; otherwise rejected.

* These techniques are widely used in text processing, compilers, & network protocols.

* A Finite automa

$M = \{ Q, \Sigma, q, F, \delta$

Q : Finite set of

Σ : Set of input

q_0 : Initial state

F : Set of final st

δ : Transition func

Basic Terms

1) Symbols :- A symbol (is character) which can be any alphabet, or letter, or digit
Eg :- a, b, c, 0, 1 ... @, $, ... etc.

2) Alphabet : (

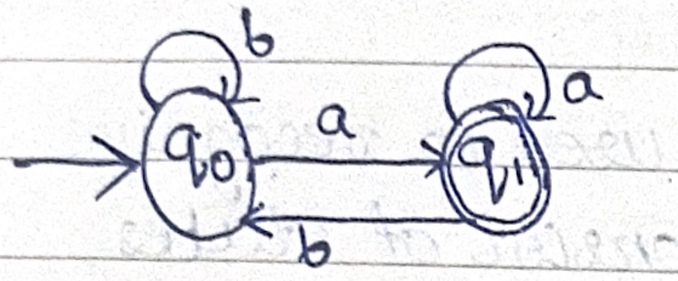
1. Deterministic Finite Automata (DFA).

In a DFA for each input symbol, the machine transitions to one and only one state. DFA does not allow any null transitions, meaning every state must have a transition defined for every input symbol.

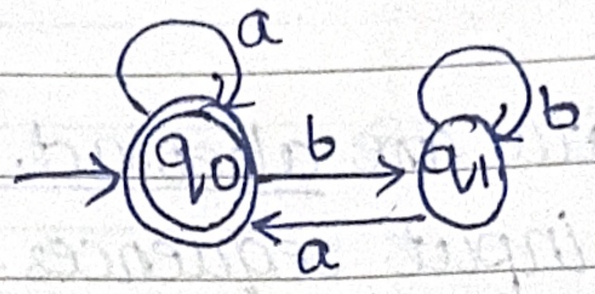
Q. Construct a DFA that accepts all strings ending with 'a'.

Given: $\Sigma = \{a, b\}$.

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



or



(aba)

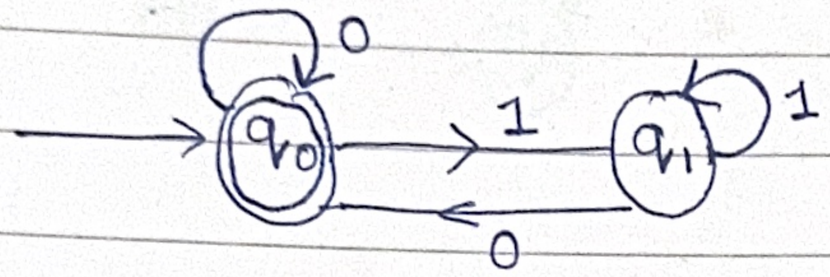
(bba)

State / symbol	a	b
q_0	q_1	q_0
q_1	q_1	q_0

Q. Design finite automata which accepts the string over $\Sigma\{0,1\}$ which when interpreted as binary number which is divisible by 2.

Hint: Divide by 2, two possibilities (0, 1).

Diagram:

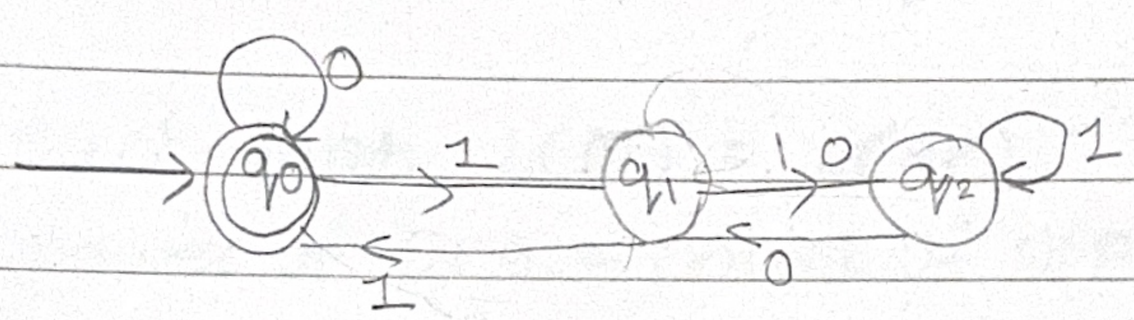


$\tilde{0}$ $\bar{1}$

1001

1010

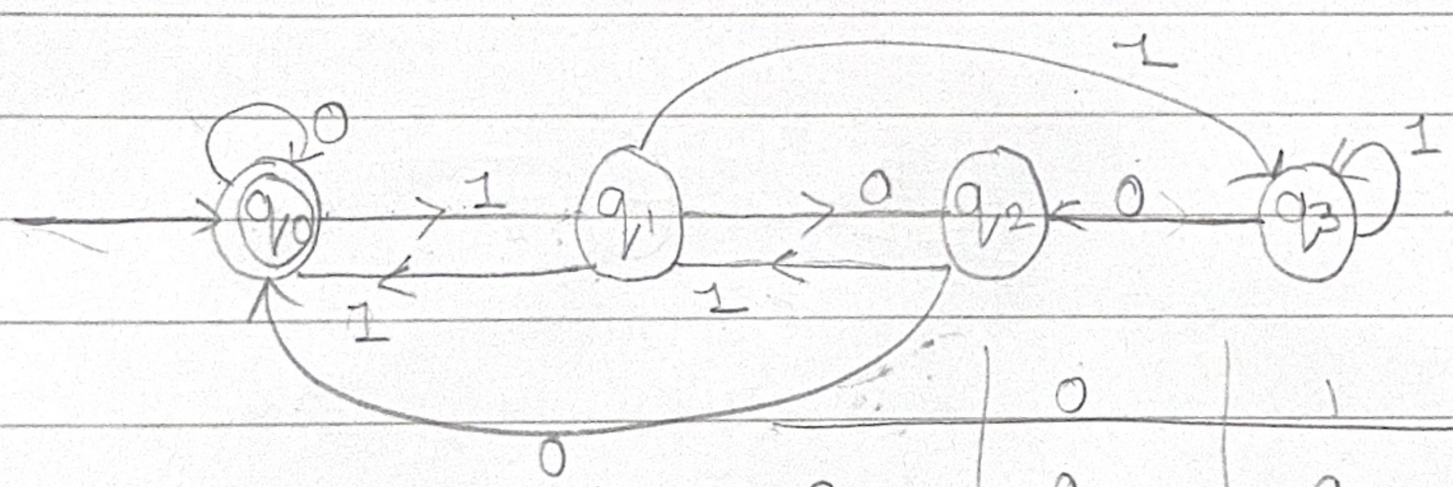
1011



↳ Divisible by 3

0 mod 3 ⇒ {0, 1, 2}

$\tilde{0}$ $\bar{1}$ $\hat{2}$



↳ Divisible by 4

0 mod 4 ⇒ {0, 1, 2, 3}

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

$\tilde{0}$ $\bar{1}$

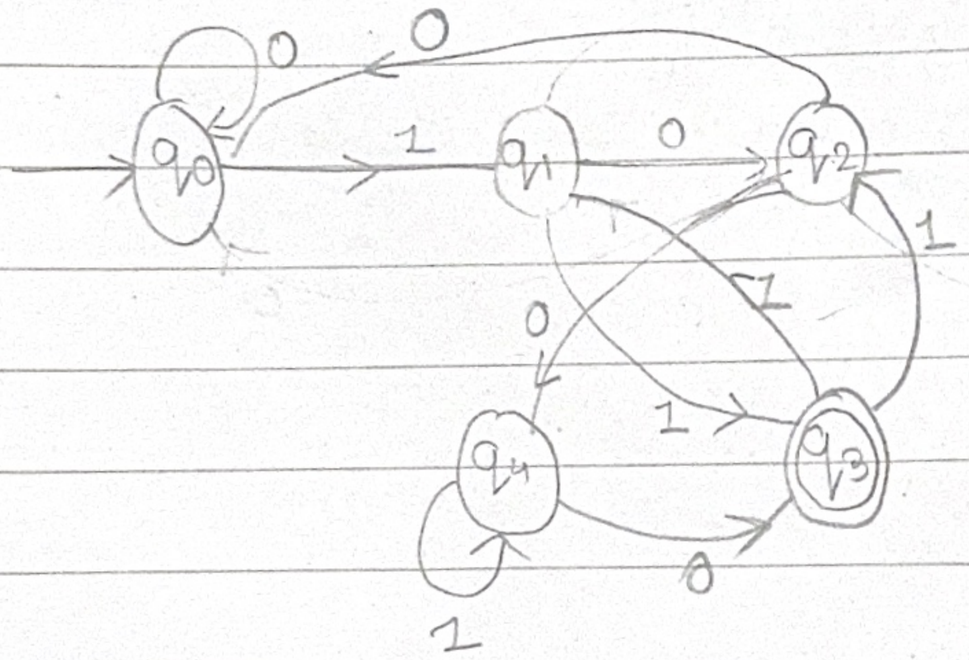
10000 - 0

10001 - 1

10010 - 18 → 2

3 mod 5

state \ Σ	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

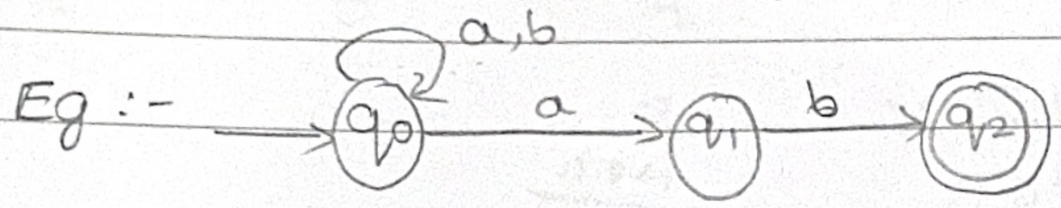


2. NOn-Deterministic Finite Automata

$$\delta : Q \times \Sigma \rightarrow 2^{Q}$$

(↑ length of Σ)

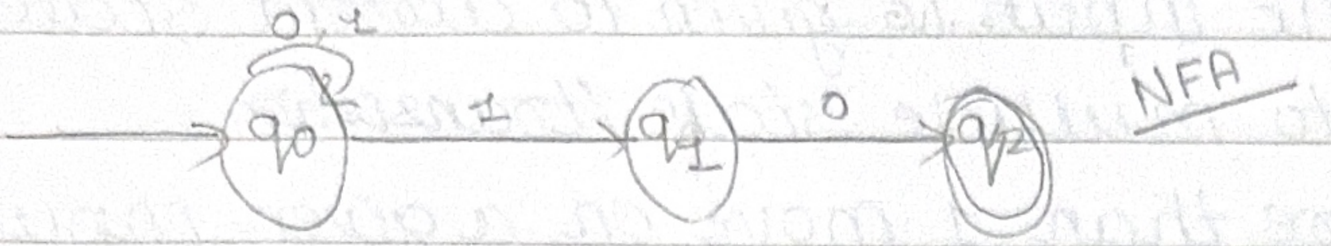
In NFA when a specific input is given to current state, the machine can go to multiple state/transition. It can have 0 or more than 1 move on a given input symbol.



* Let $m = \{Q, \Sigma, q, F, \delta\}$ Is an NFA, which accepts the language $L(m)$,
Then there should be equivalent DFA denoted $m' = \{Q', \Sigma', q', F', \delta'\}$,
such that $L(m) = L(m')$.
power of DFA = power of NFA.

*

9. Design a finite automata ending with '10' over $\Sigma\{0, 1\}$. (Non - Deterministic).



NFA

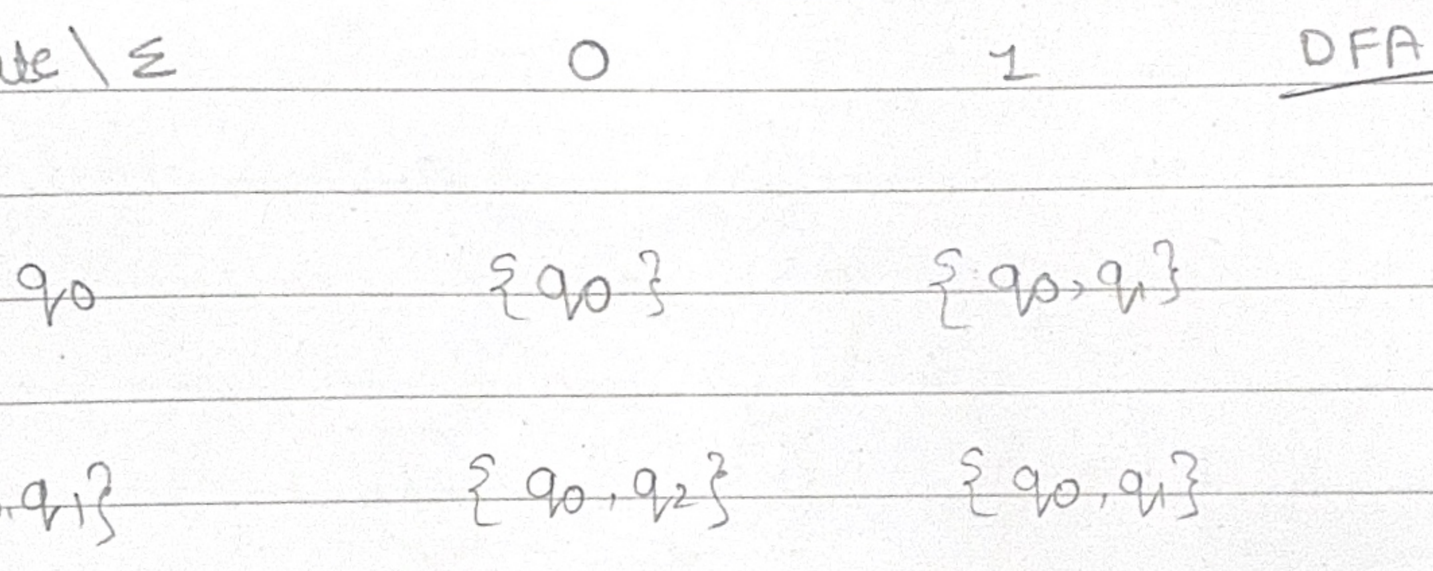
↓ convert

State \ Σ	0	1
→ q_0	q_0	q_0, q_1
q_1	q_2	-
q_2	-	-

NFA

State \ Σ	0	1
q_0	$\{q_0\}$	$\{q_0, q_1\}$
$\{q_0, q_1\}$	$\{q_0, q_2\}$	$\{q_0, q_1\}$
$\{q_0, q_2\}$	$\{q_0\}$	$\{q_0, q_1\}$

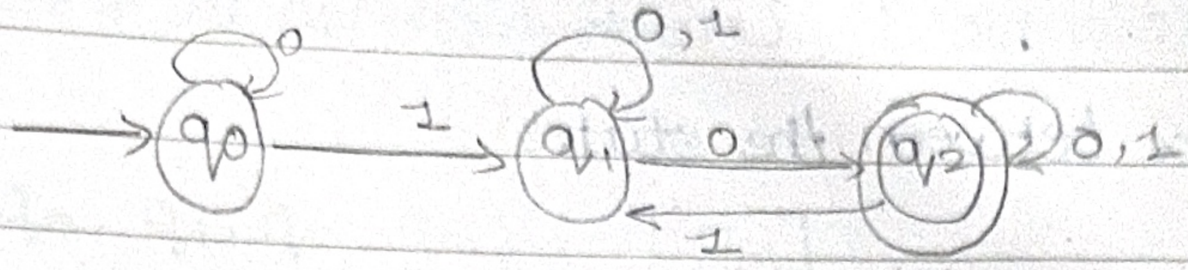
DFA



DFA

* check for unreachable state.

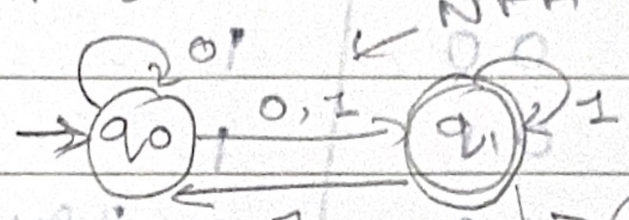
Eg - (2) NFA



state \ Σ	0	1
→ q_0	q_0	q_1
q_1	$\{q_1, q_2\}$	q_1
q_2	$\{q_2\}$	$\{q_1, q_2\}$

homework -

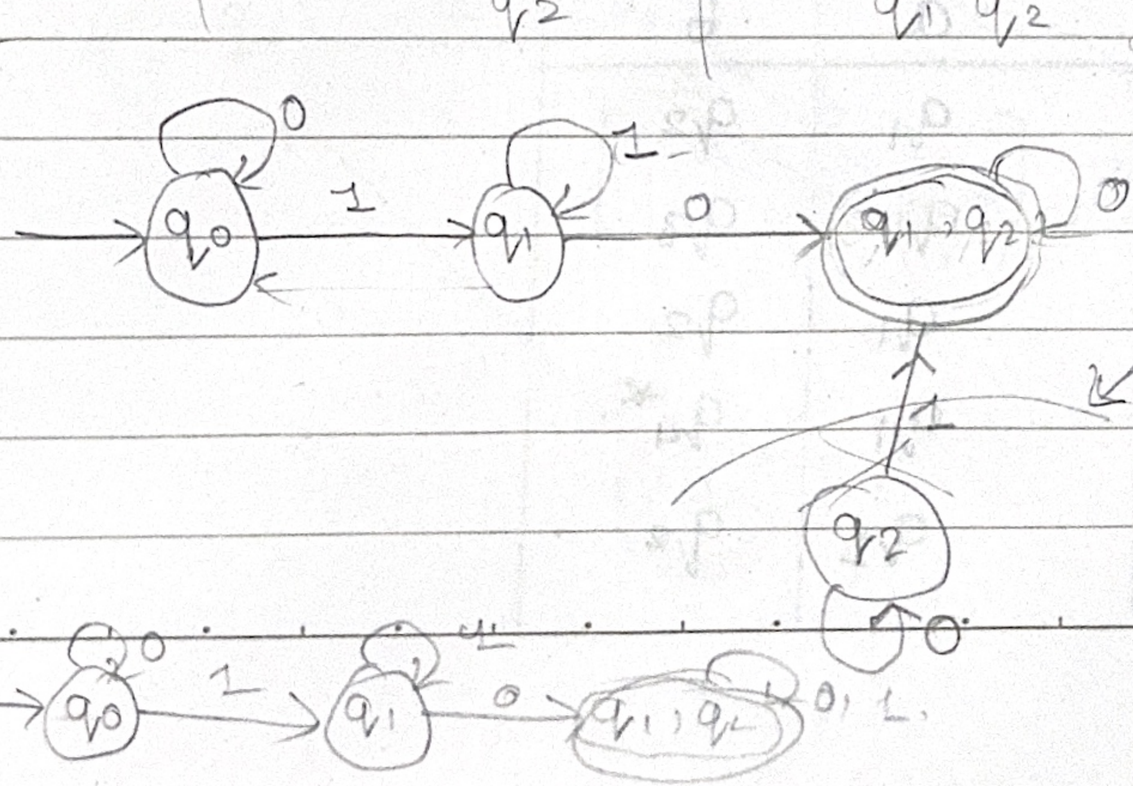
* Questions could be asked for gate.



→ (three states)

DFA -

state \ Σ	0	1
→ q_0	$\{q_0\}$	$\{q_1\}$
q_1	$\{q_1, q_2\}$	$\{q_1\}$
q_1, q_2	$\{q_1, q_2\}$	$\{q_1, q_2\}$
q_2	q_2	q_1, q_2



→ or eliminated

excluded as unreachable state.

- Minimization of DFA : (One question compulsary).

(I) State and equivalance between the states.
(P, q) equivalent.

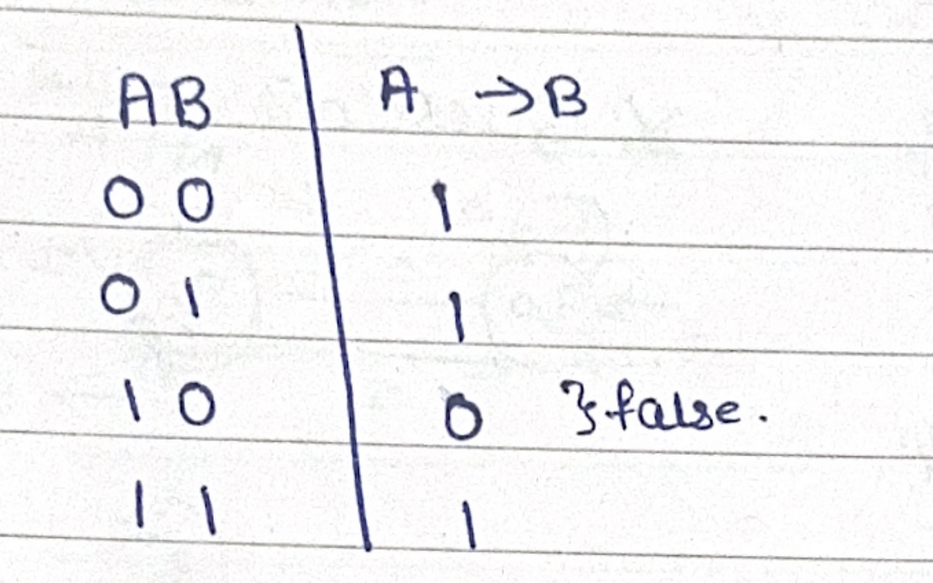
$\delta(P,w) \in F \Rightarrow \delta(q,w) \in F$

Example :-

AB	A → B
0 0	1
0 1	1
1 0	0 ∃ false.
1 1	1

and $\delta(P,w) \notin F \Rightarrow \delta(q,w) \notin F$.

Example 1 :



State transition table :-

State \ Σ	a	b
→ q_0	q_1	q_2
q_1	q_1	q_3
q_2	q_1	q_2
q_3	q_1	q_4*
* q_4	q_1	q_2

check for unreachable state

(I) 0 equivalence states.

[set of all non-final state/s] [set of all final state/s]

⇒ $[q_0, q_1, q_2, q_3]$ $[q_4]$.

(II) One (1) equivalence states.

step 1: compare q_0 with q_1, check set [ ]

q_0 with q_2, check set [ ]

⋮

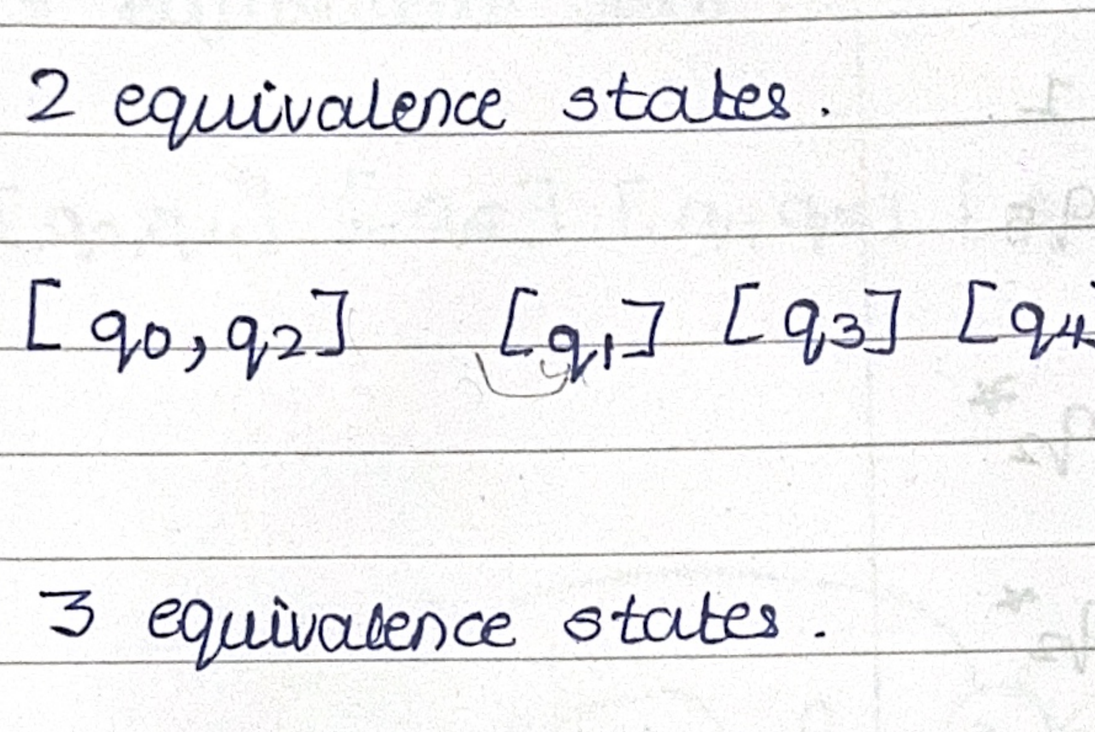
⇒ $[q_0, q_1, q_2]$ $[q_3]$ $[q_4]$.

(III) 2 equivalence states.

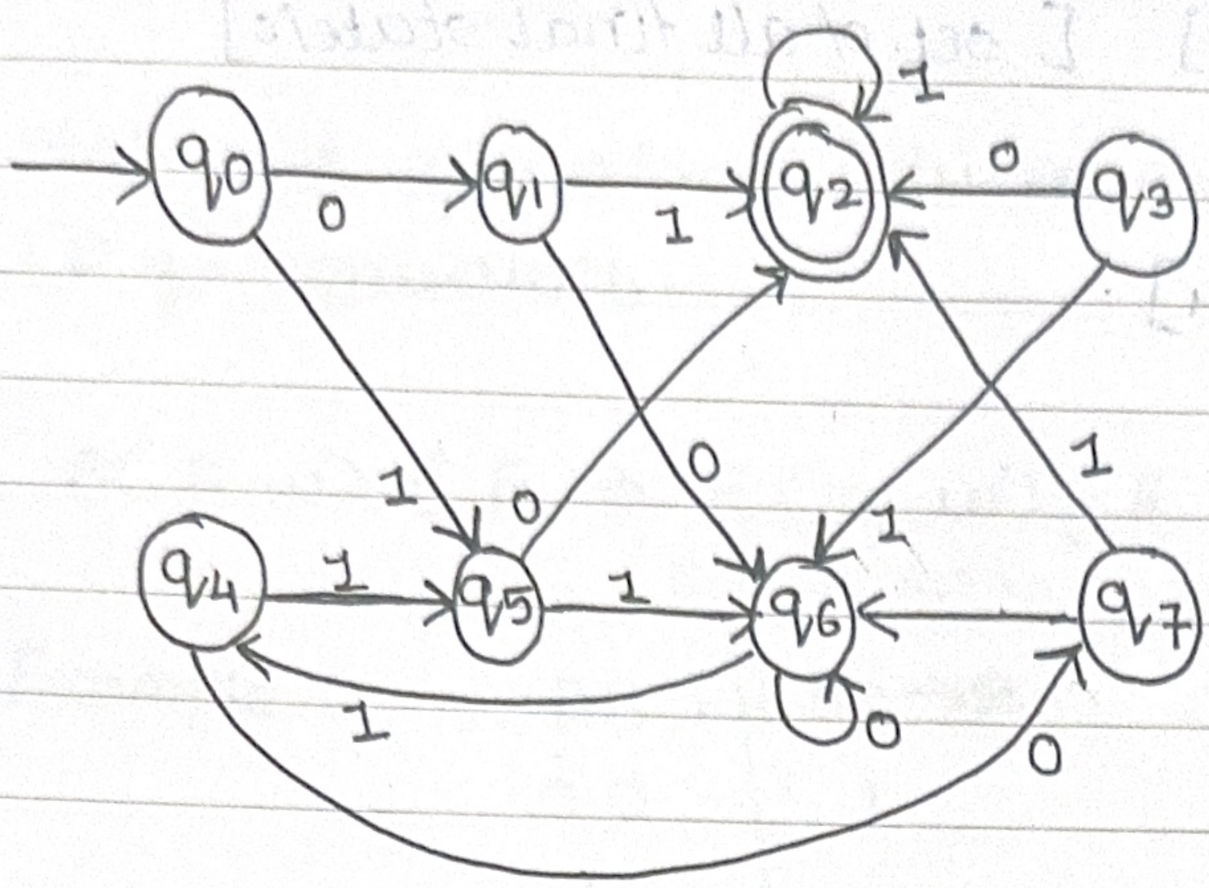
⇒ $[q_0, q_2]$ $[q_1]$ $[q_3]$ $[q_4]$.

(IV) 3 equivalence states.

⇒ $[q_0, q_2]$ $[q_1]$ $[q_3]$ $[q_4]$.



Example 2:



step1 : identify unreachable state (q_3).

step2 :

State \ Σ	0	1
→ q_0	q_1	q_5
q_1	q_6	q_2^*
* q_2	q_0	q_2^*
q_4	q_7	q_5
q_5	q_2^*	q_6
q_6	q_6	q_4
q_7	q_6	q_2^*

(I) 0 equivalence state.

⇒ [q_0, q_1, q_4, q_5, q_6, q_7] [q_2]

(II) 1 equivalence state.

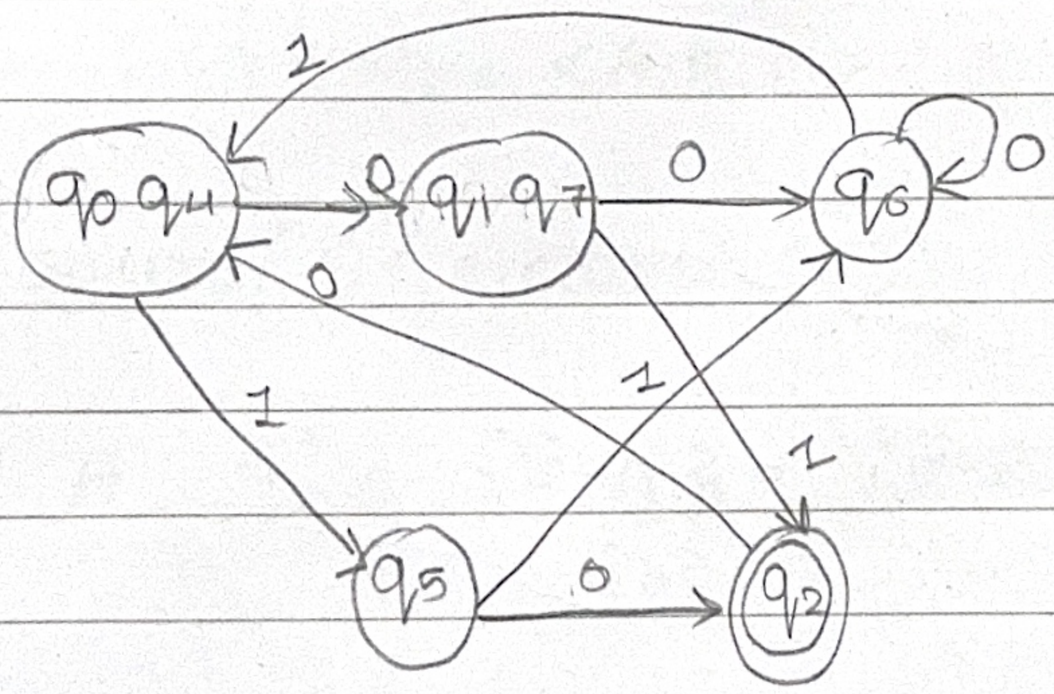
⇒ [q_0, q_4, q_6] [q_1, q_7] [q_5] [q_2]

(III) 2 equivalence state.

⇒ [q_0 q_4] [q_6] [q_1 q_7] [q_5] [q_2].

(III) 3 equivalence state.

→ [q_0, q_4] [q_6] [q_1 q_7] [q_5] [q_2].



chomskey hierarchy {toc syllabus}.

Combination to check if pda, finite automata, etc will fail or work?

* TOC
  - Languages
  - Automata ⇒ type
  - Grammar
    - set of rules.

Type 0 :- Turing Machine

Type 1 :- Linear Bounded Automata

Type 2 :- Pushdown Automata

Type 3 :- Finite Automata.

Definition of these

1) Symbol
2) Alphabet
3) String. ⟹ String of zero length
4) Language.

⇒ PDA
- DPDA
- NPDA

concept of Prefix & suffix

different length of Prefix or suffix = {n+1}.

Null Prefix or suffix = {n}.

Substring Problems.

w = abcde

- 0 len → ∈ (1)
- 1 len → a, b, c, d, e (5).
- 2 len → ab, bc, cd, de. (4).
- 3 len → abc, bcd, cde (3).
- 4 len → dbcd, bcde (2).
- 5 len → abcde. (1).

Formula: $\frac{n(n+1)}{2} + 1$ ← ∈

↑ Arithmetic Summation

DFA :- $\delta: Q \times \Sigma \to Q$. ϵ = zero length string.

NFA :- $\delta: Q \times \Sigma \to 2^Q$.

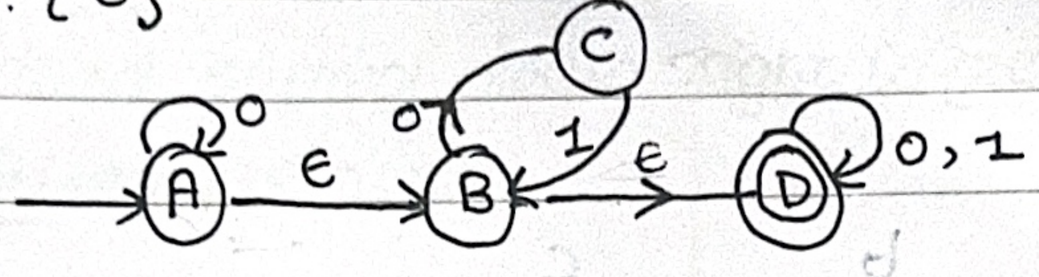
ϵNFA :- $\delta: Q \times \Sigma \cup \{\epsilon\} \to 2^Q$.

- ϵ-NFA : $\{\epsilon = NULL\}$ .... assume.

$M = \langle Q, \Sigma, \delta, q_0, S, F \rangle$.

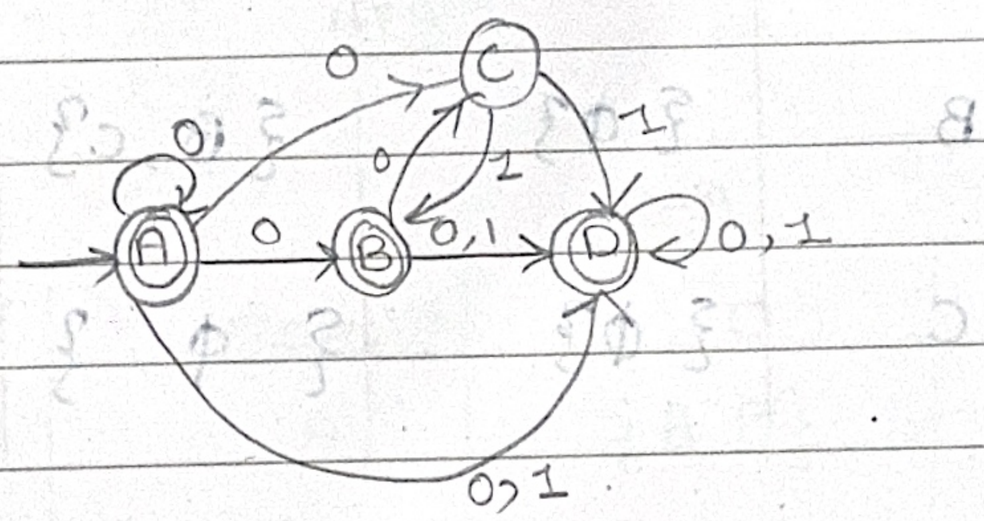
$\delta = Q \times \Sigma \cup \{\epsilon\} \to 2^Q$.

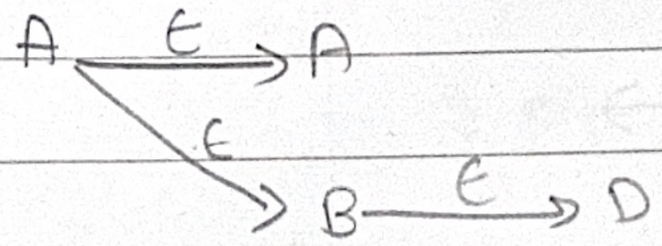
Example :-



States \ Σ	0	1
→ (A)	{A, B, C, D}	{D}
(B)	{C, D}	{D}
C	{ϕ}	{B, D}
(D)	{D}	{D}

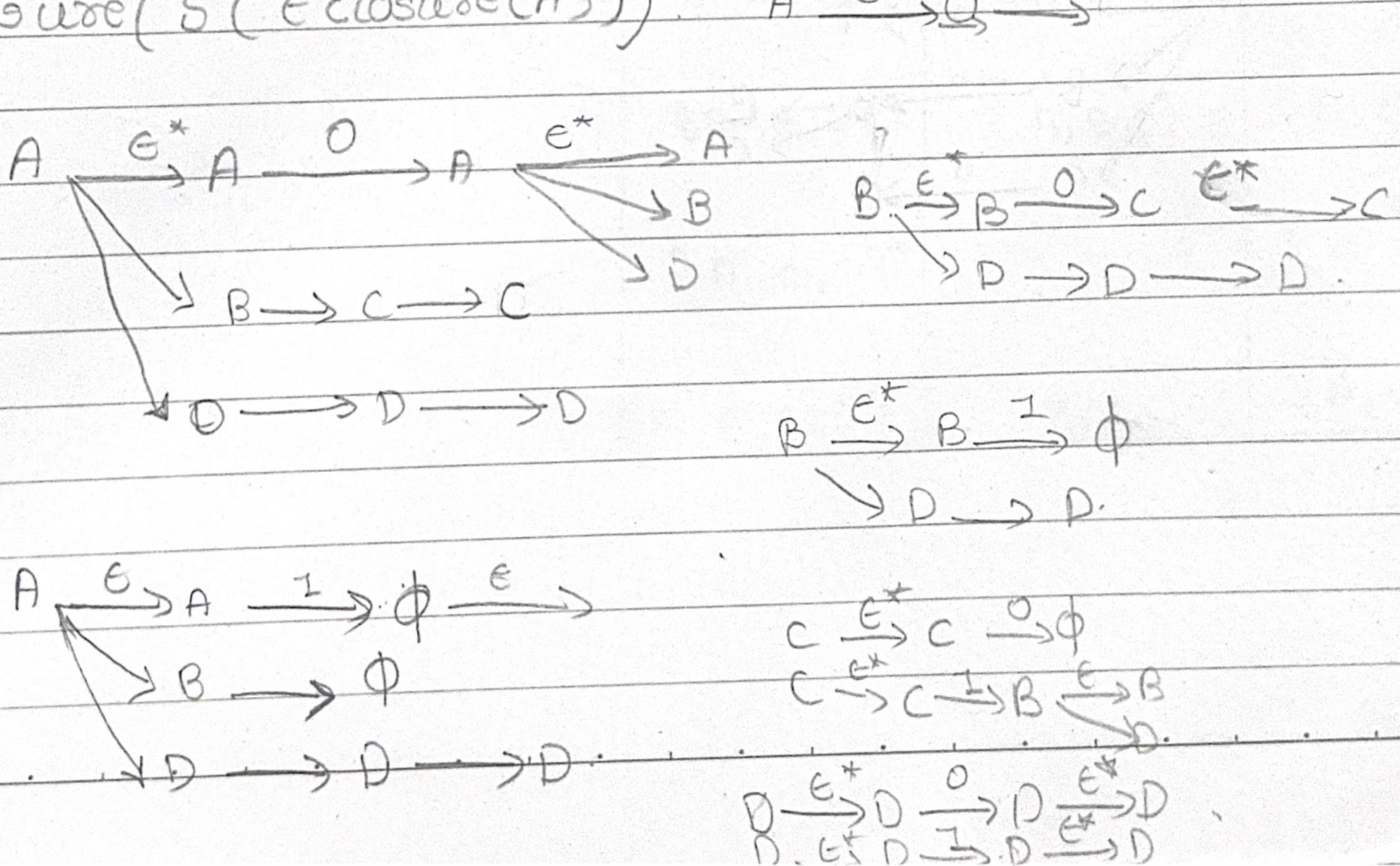
NFA





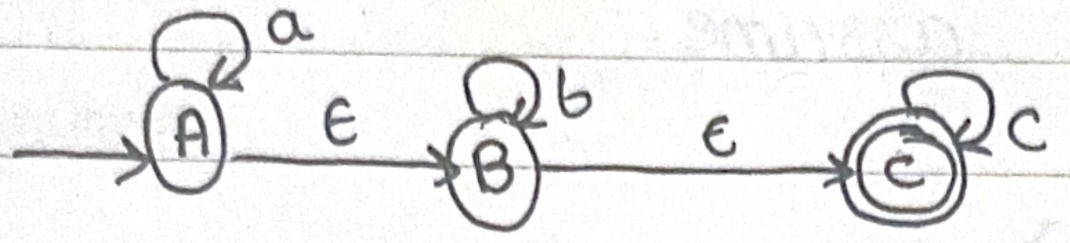
ϵ - closure (A) = $\langle A, B, D \rangle$.

ϵ - closure (δ (ϵ closure (A))). $A \xrightarrow{\epsilon^*} Q \xrightarrow{\epsilon^*}$



Divya → kulkarni
Library book
LB Mishra.

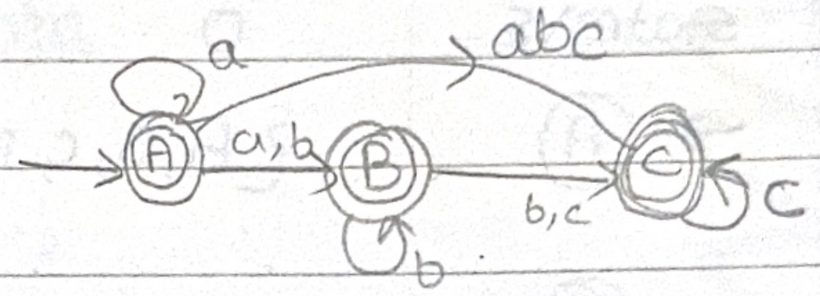
Example 2 :

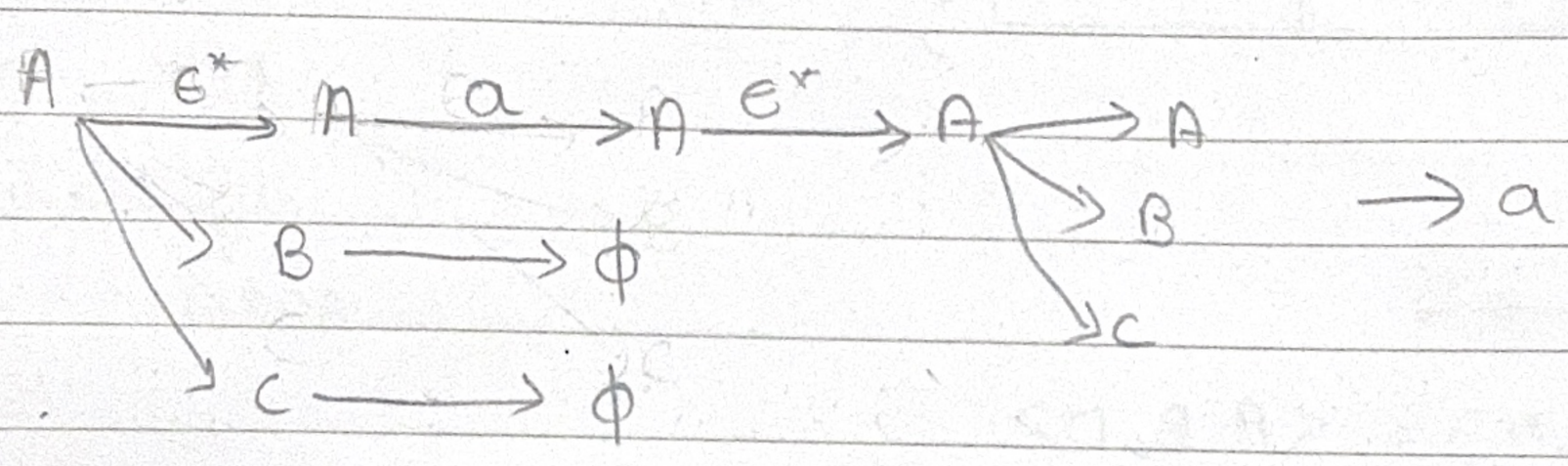


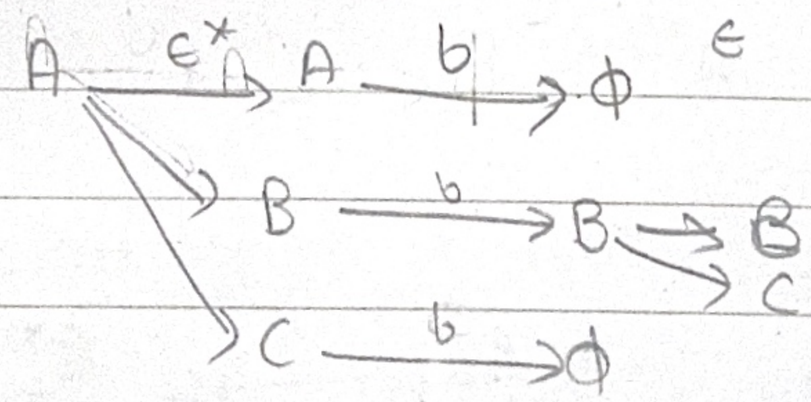
Given : = $\Sigma \langle a, b, c \rangle$.

States \ Σ	a	b	c
→ A	{A, B, C}	{B, C}	{C}.
B	{Φ}	{B, C}	{C}
C	{Φ}	{Φ}	{C}.

NFA





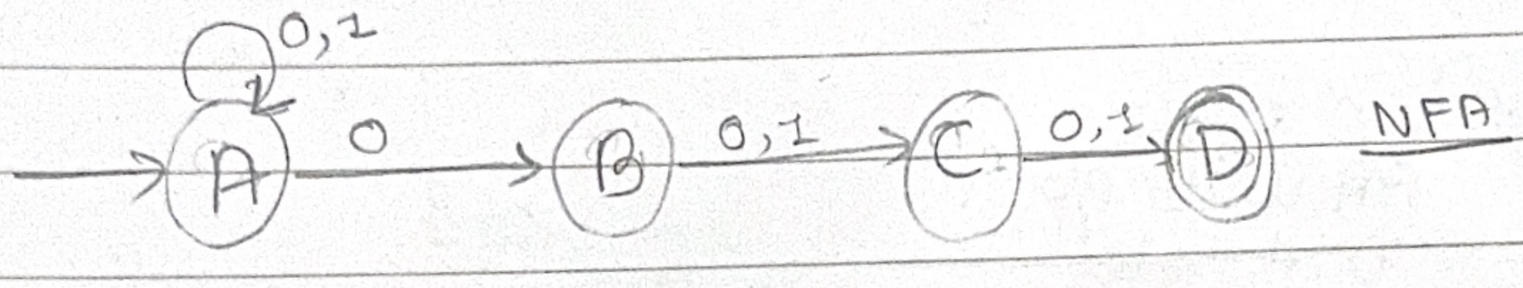


Hence → NFA solving N states ⇒ DFA solving $n \leq 2^n$

* Number of states in ε-NFA and NFA are always same.

Example 3 : Design a finite automata for the string which is having third symbol from right is {zero} '0'.

Given : $\Sigma = <0, 1>$



NFA

DFA

States \ Σ	0	1
→ A	{A, B}	{A}
B	{C}	{C}
C	{D}	{D}
D	{ϕ}	{ϕ}

States \ Σ	0	1
A	{A, B}	{A}
{A, B}	{A, B, C}	{A, C}
{A, C}	{A, B, D}*	{A, D}*
{A, B, C}	{A, B, C, D}*	{A, C, D}*
{A, D}*	{A, B}	{A}
{A, B, D}*	{A, B, C}	{A, C}
{A, C, D}*	{A, B, D}*	{A, D}*
{A, B, C, D}*	{A, B, C, D}*	{A, C, D}*

FA

without output

DFA

NFA

ϵ-NFA

with output

melay machine

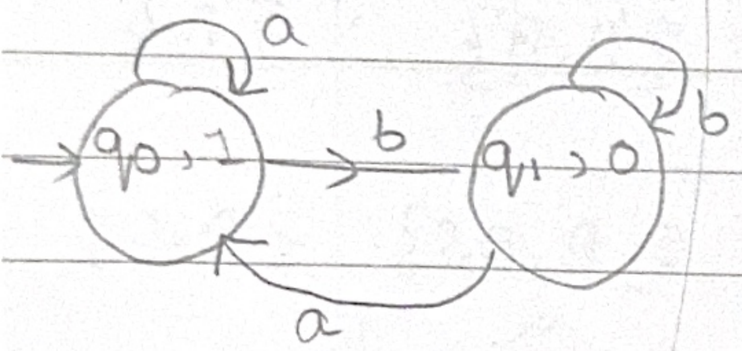
moore machine

FA with o/p.

one output

moore machine

melay machine.



$\lambda : Q \to \Delta$

$(Q, \Sigma, \delta, q_0, \Delta, \lambda)$

Q: finite set of states.

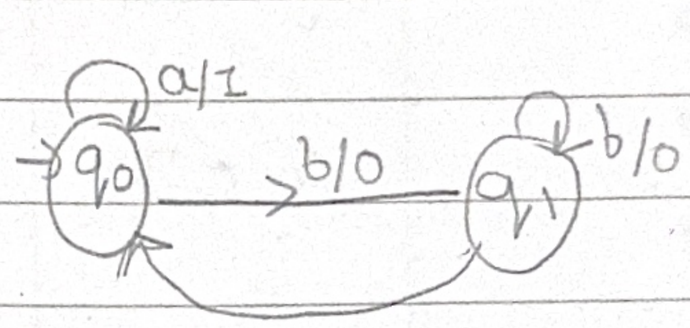
Σ: finite input alphabet

δ: Transition function.

$Q \times \Sigma \to Q$.

Δ: o/p alphabet.

λ: o/p function.



$\lambda : Q \times \Sigma \to \Delta$

output.

$(q_0, a) \to 1$

$(q_0, b) \to 0$

$(q_1, a) \to 1$

$(q_1, b) \to 0$.

i/p :

$q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_1$

1 1 0

o/p 1

{ input : n
output : n+1

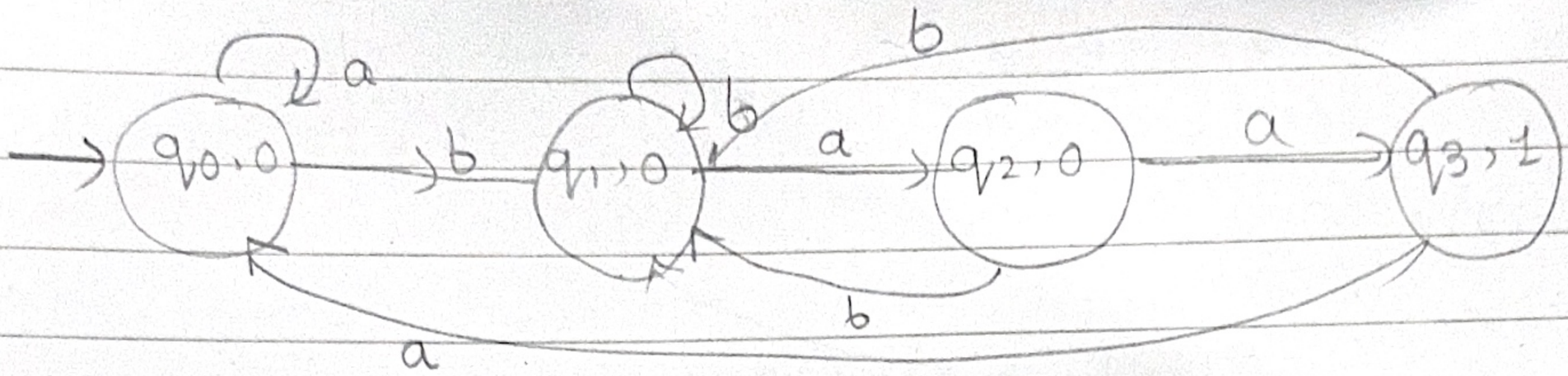
i/p :

$q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_1$

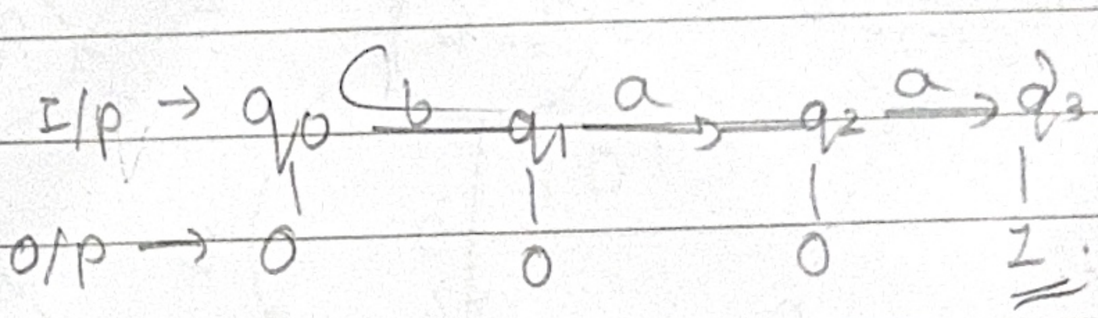
o/p. 1 0

{ input : n
output : n

* Example: input $\{a, b\}$ $\Sigma = \{a, b\}$, $\Delta = \{0, 1\}$.
count occurence of 'baa'



bababaa



* Example: Construct moore machine that takes binary number as input & produces residue modulo output.

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

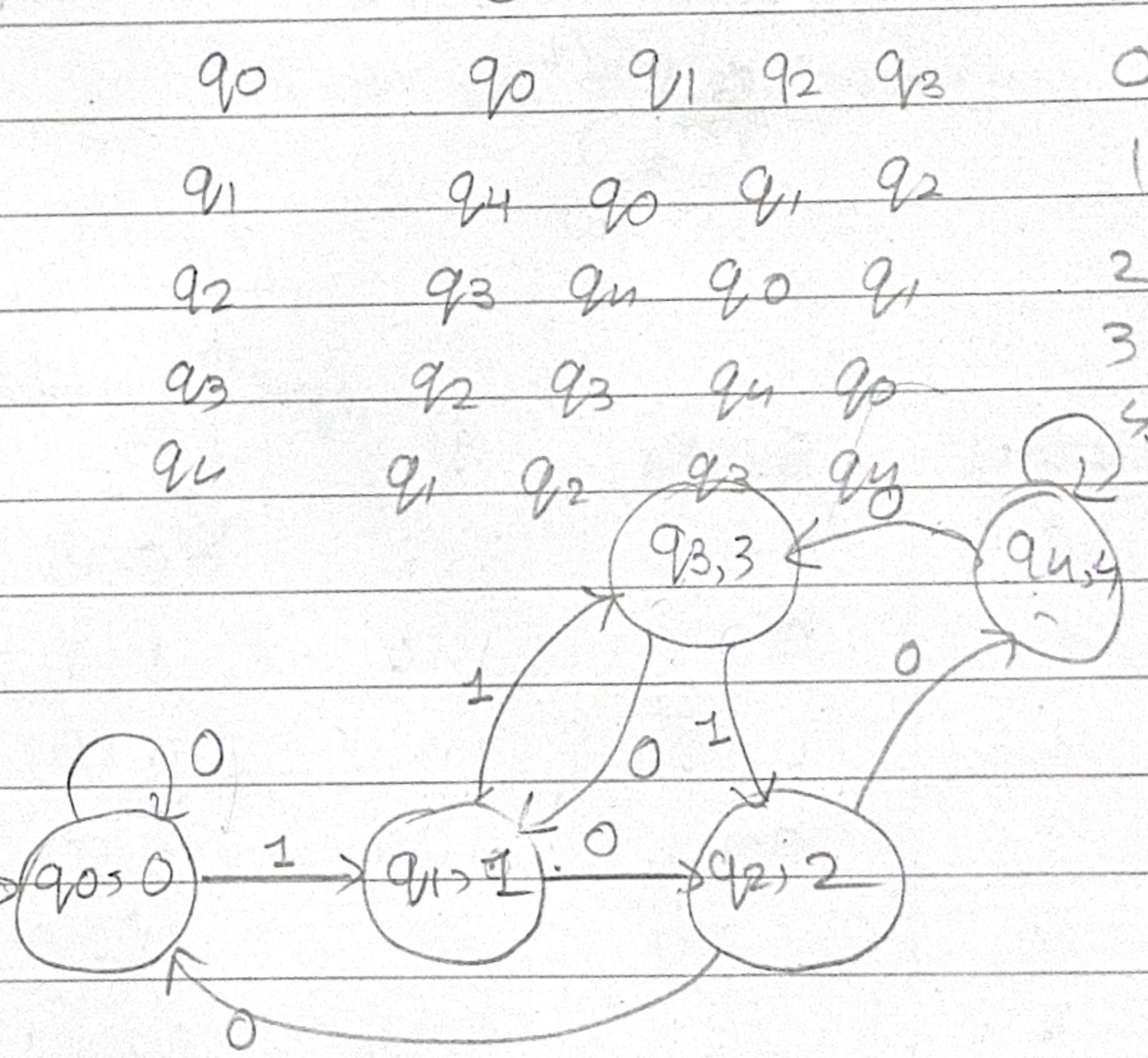
residue modulo of 3, $\Delta = \{0, 1, 2\}$.

↳ shortcut :- zig zag pattern.

	0	1	Δ
q_0	q_0	q_1	0
q_1	q_2	q_0	1
q_2	q_1	q_2	2

	0	1	2	3	Δ
q_0	q_0	q_1	q_2	q_3	0
q_1	q_4	q_0	q_1	q_2	1
q_2	q_3	q_4	q_0	q_1	2
q_3	q_2	q_3	q_4	q_0	3
q_4	q_1	q_2	q_3	q_4	4

	0	1	Δ
→ q_0	q_0	q_1	0
q_1	q_2	q_3	1
q_2	q_4	q_0	2
q_3	q_1	q_2	3
q_4	q_3	q_1	4

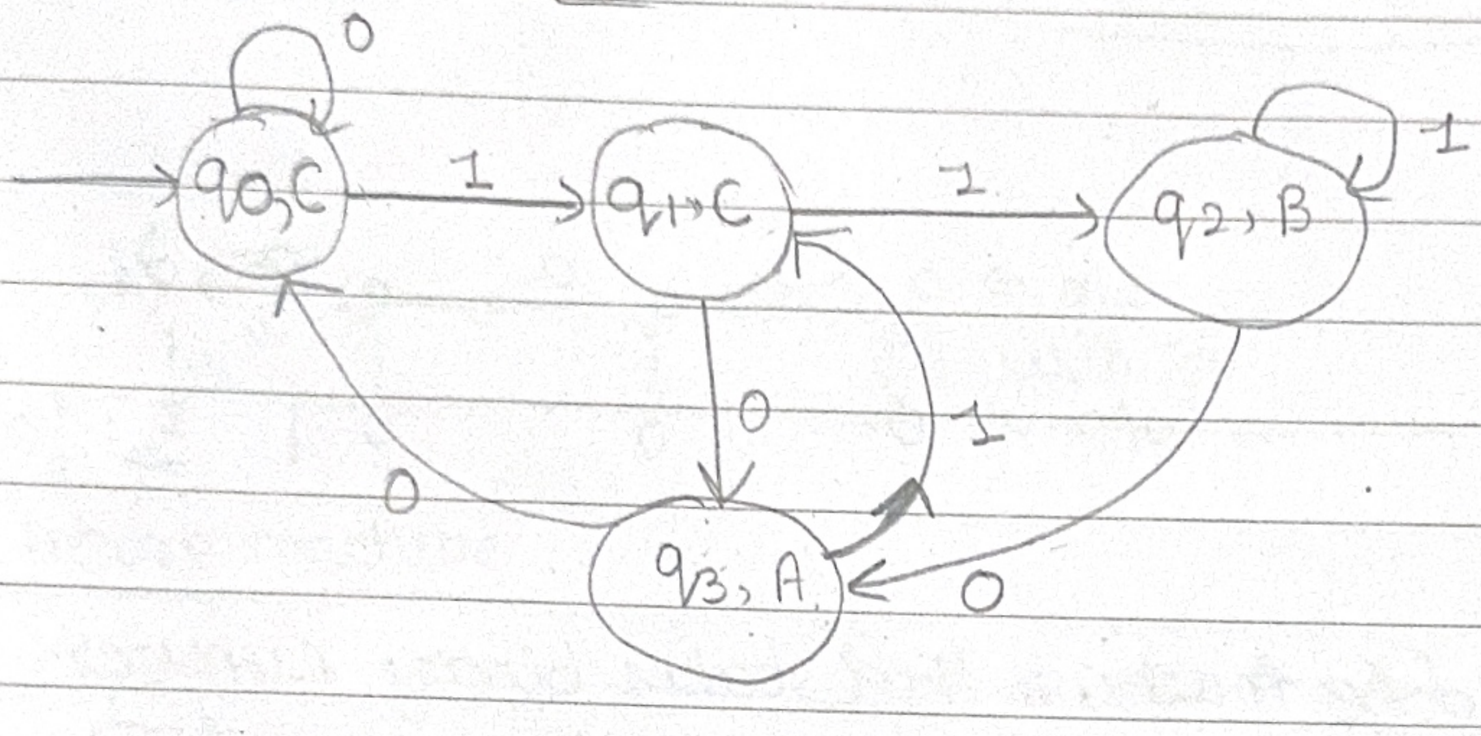


$\Sigma = \{0, 1\}$

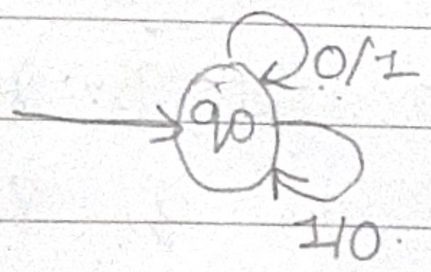
$A = \{A, B, C\}$

	O/P	I/P
	A →	10
	B →	11
	C →	0.w

~~1010XXX101~~



Example : Design a melay machine to 1's compliment by taking input binary number.



Eg:- 1010
↓
0101

2's complement : 10111
↓↓↓↓↓
01000
+1
01001

Decimal	Binary	1's	2's
0	000	111	1000 - 0
1	001	110	111 - -1
2	010	101	110 - -2
3	011	100	101 - -3
4	100	011	100 - -4
5	101	010	011 - 3
6	110	001	010 - 2
7	111	001	001 - 1

MSB

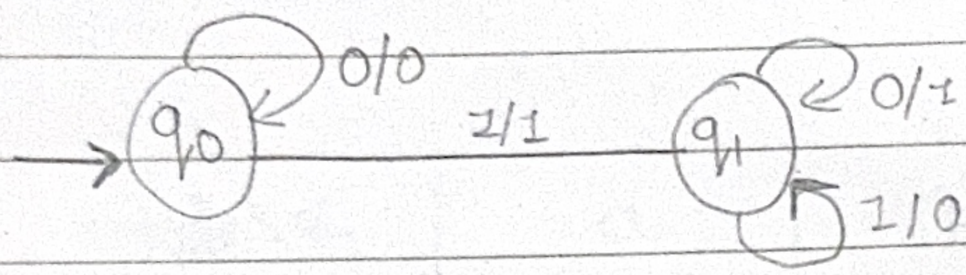
010100
↓↓↓↓↓↓
101011 →
+1
101100

1 0 1 0 0 0 0 0
↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓
0 1 0 1 1 1 1 1
+ 1
0 1 1 0 0 0 0 0

1 1 1 1 0 0 1 ← LSB
↓ ↓ ↓ ↓ ↓ ↓ ↓
0 0 0 0 1 1 0
+ 1
0 0 0 0 1 1 1

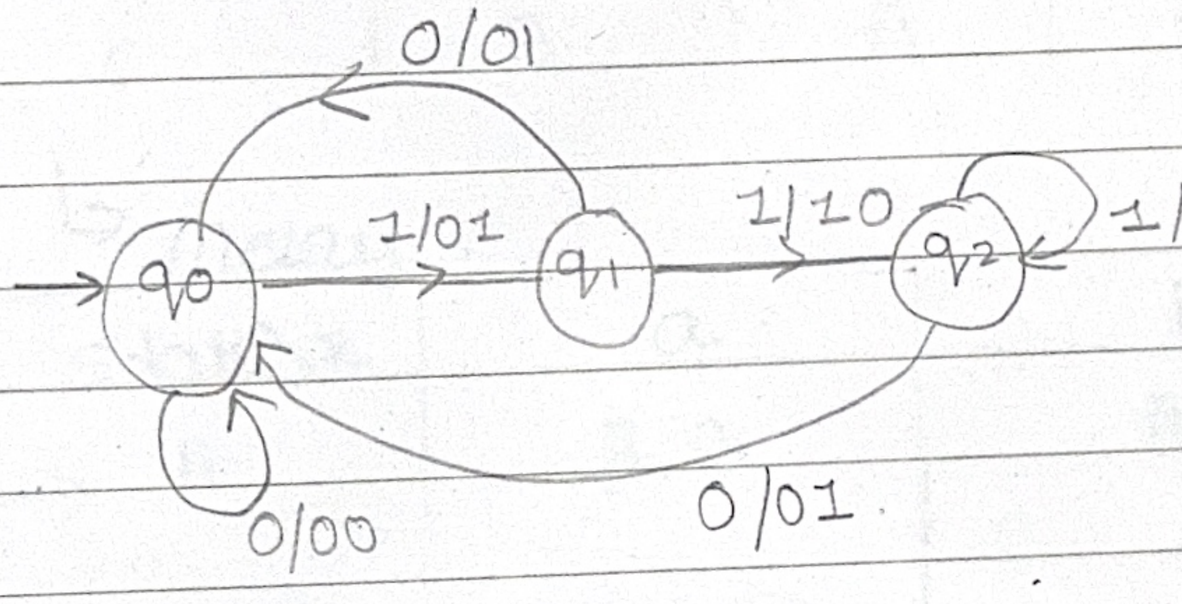
→ encounter 1st (1) & copy as it is, left side complement right side as it is.

→ Delay for 2's complement.



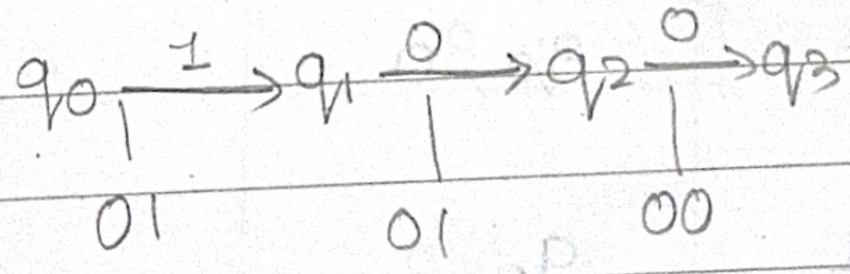
0 1 0 1 0 0 ← LSB.
↓ ↓ ↓ ↓ ↓ ↓
1 0 1 0 1 1
+ 1
1 0 1 1 0 0

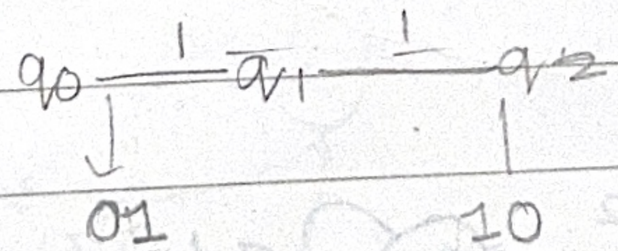
* GATE QUESTION



1 1 0 1 0 1 0 1 1 1 1 1

a) 11 → 01
b) 10 → 00
~~c) Sum of prev & present bit~~
~~d)~~ 11 → 10 (None)

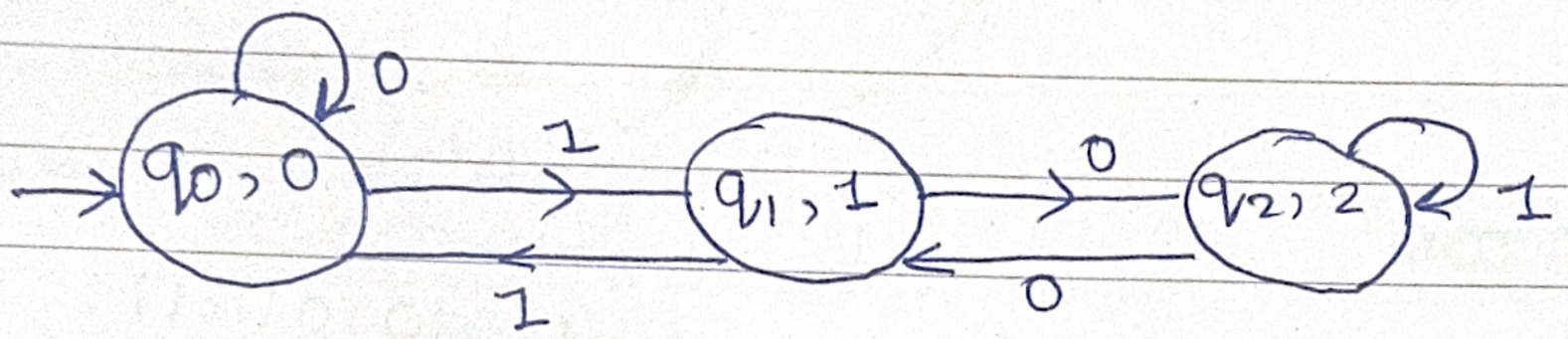




- Conversion of Moore Machine to Melay Machine (Same No. of States).

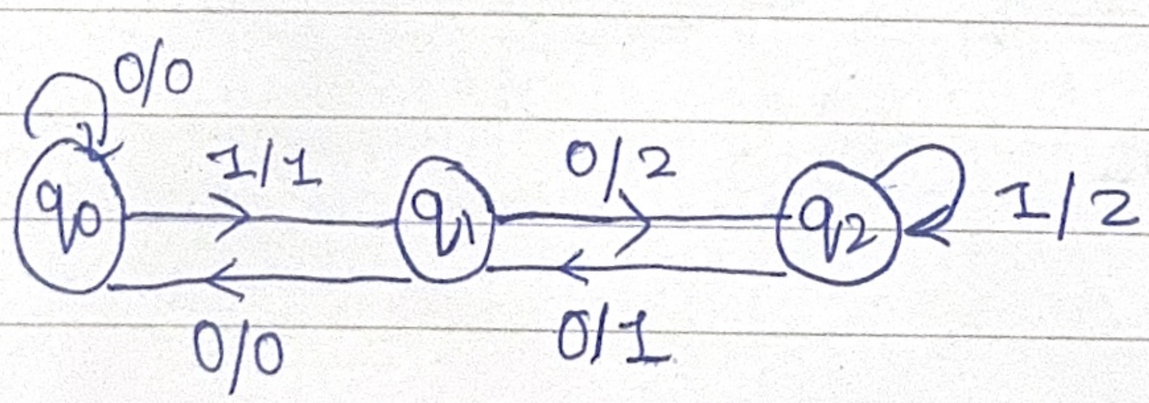
Eg1:

State \ Σ	0	1	Δ
→ q_0	q_0	q_1	0
q_1	q_2	q_0	1
q_2	q_1	q_2	2

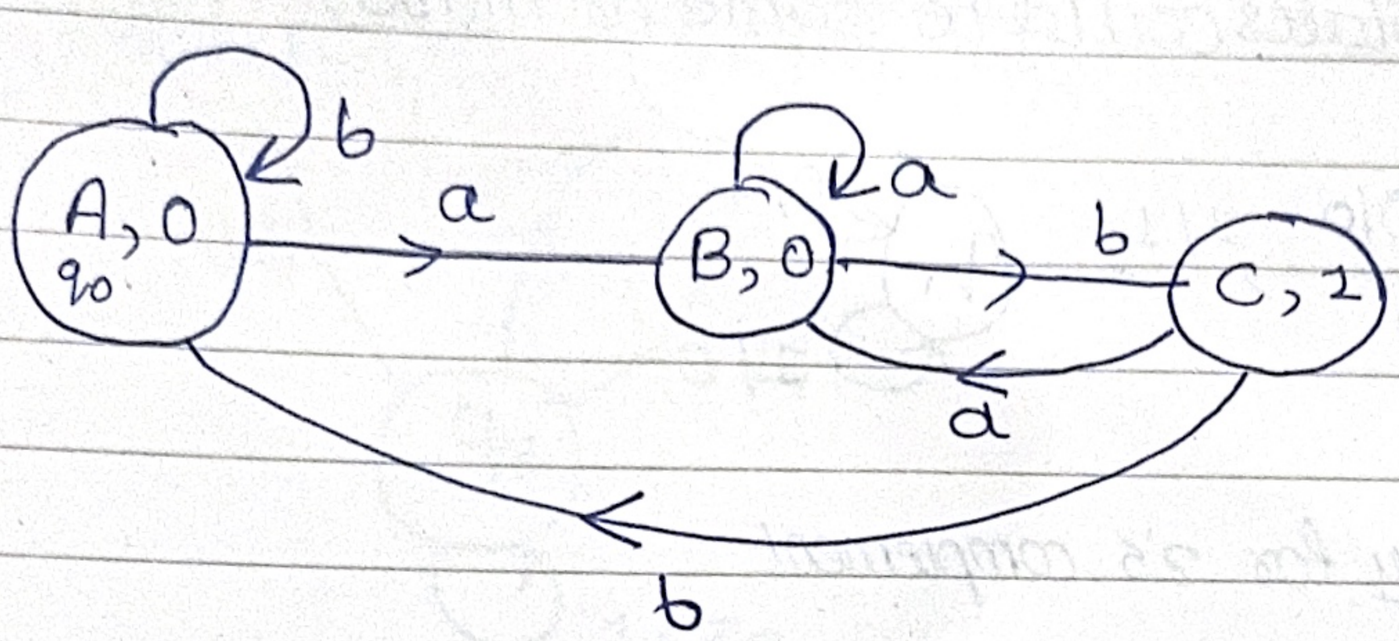


⇒ Melay.

State \ Σ	0	1
→ q_0	q_0, 0	q_1, 1
q_1	q_2, 2	q_0, 0
q_2	q_1, 1	q_2, 2



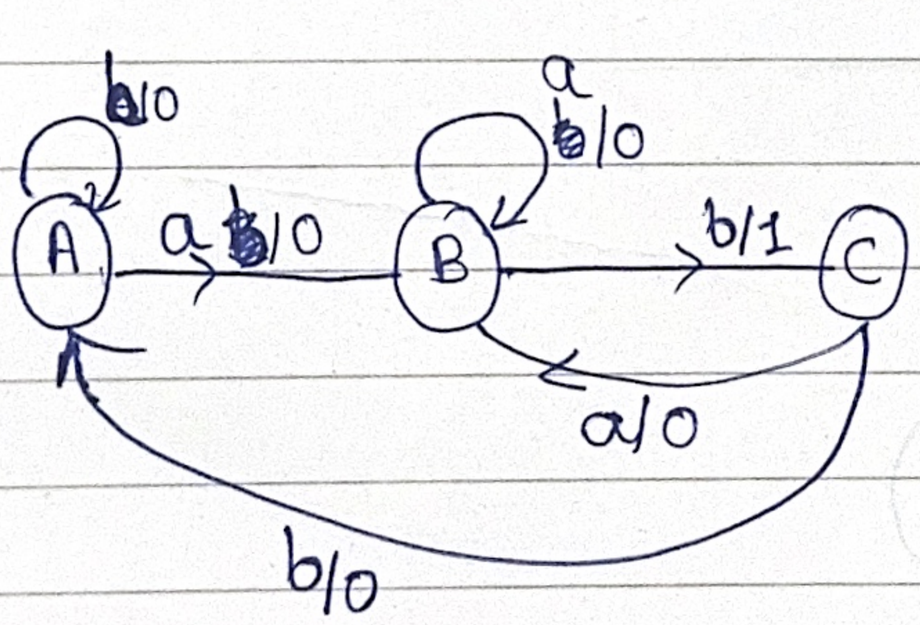
Eg2: Moore Machine :-



state\Σ	a	b	Δ
→ A,	B	A	0
B	B	C	0
C	B	A	1

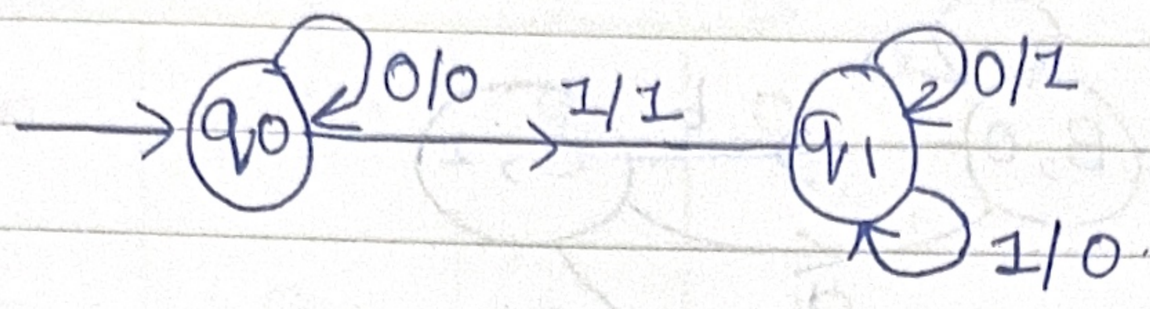
↳ Melay.

state\Σ	a	b
→ A	B, 0	A, 0
B	B, 0	C, 1
C	B, 0	A, 0.



- Conversion of Melay Machine to Moore Machine
  (Number of States could be same or more).

Eg :- 1

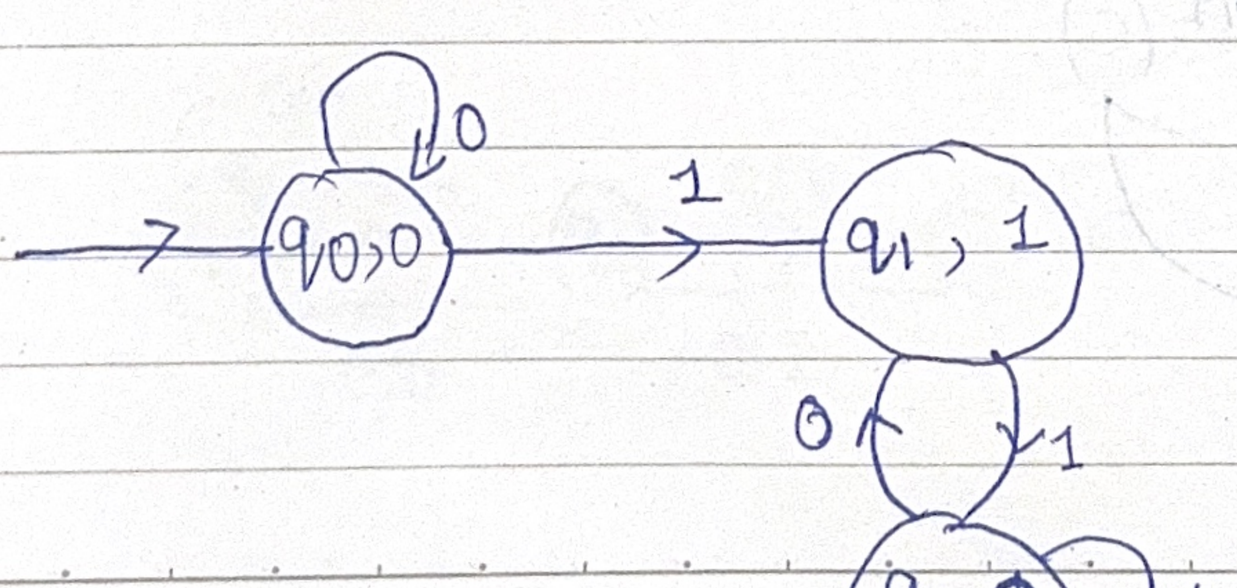


(Melay for 2's complement.

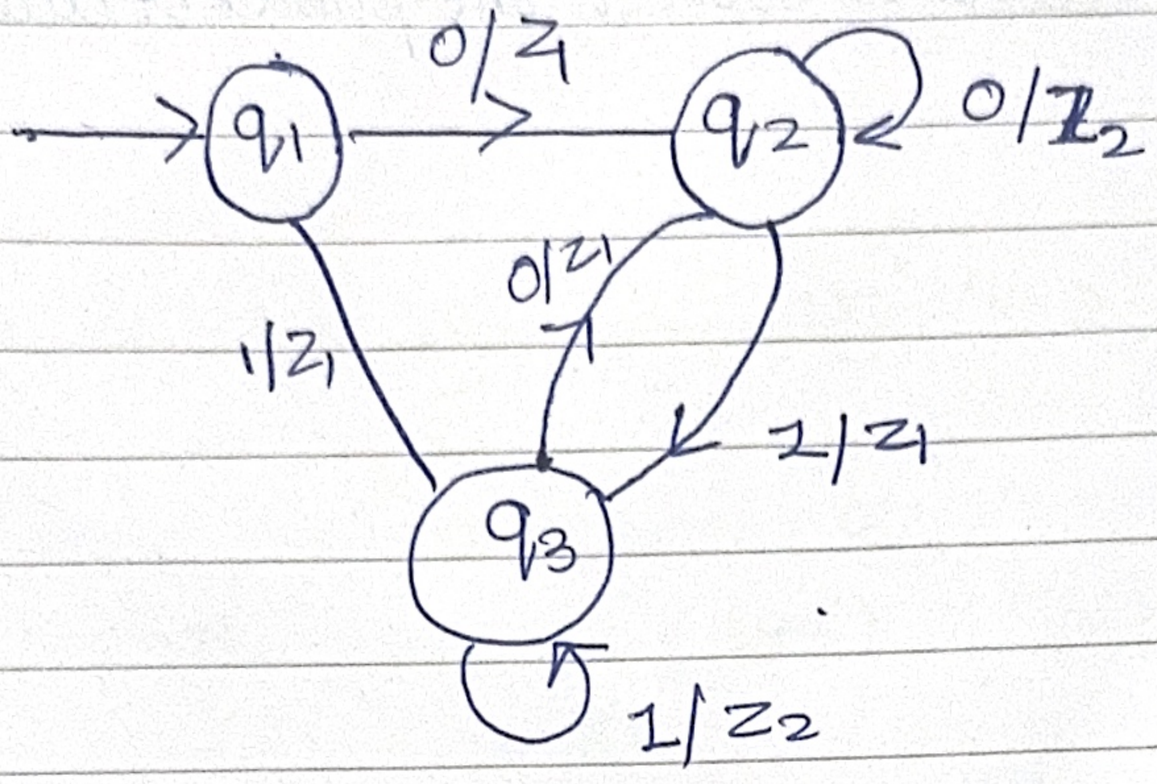
state\Σ	0	1
→q0	q0, 0	q1, 1
q1	q1, 1 (1)	q1, 0 (2)

↳ Moore Machine

State/Σ	0	1	Δ	
→q0	q0	q1	0	← output associate with state i.e, q0 = 0
(1) part of q1	q1	q1	q2	1
(2) part of q1	q2	q1	q2	0



Eg:- 2.



State/Σ	z_1	z_2
→ q_0 1		

* ϵ = any string length '0'.

Unit-2 Regular Expression

Eg:- $L_1 = \langle aa, ab, ba, bb \rangle$; $\Sigma = \langle a, b \rangle$

Soln: $\Rightarrow$ $aa + ab + ba + bb$

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

$= \underline{(a+b)(a+b)}$

↳ (string length exactly '2')

Eg2:- $L = \langle aa, ab, ba, bb, aaa, aba, abb, bbb \ldots \rangle$

Soln: $\Rightarrow$ $\underline{(a+b)(a+b)(a+b)^*}$

↳ (string length atleast '2' or more)

$\phi = \{\}$,

$\epsilon = \{\epsilon\}$.

$a = \{a\}$.

$a^* = \{\epsilon, aa, aaa, a \ldots\}$

$a^+ = \{a,$

$(a+b)^* = \{\epsilon, a, b, aa, ab, bb \ldots\}$

Eg 3 :-> For string length atmost 2.

$L \rightarrow \langle \epsilon, a, b, aa, ab, ba, bb \rangle$

Soln = $\underline{(a+b+\epsilon)(a+b+\epsilon)}$.