Course outcomes

CO1 : Construct DFA, NFA and ϵ-NFA

CO2 : proeedure regular expression and regular grammer

CO3 : Construct context free grammer, Context free language.

CO4 : Construct push down Automata and its equivalence with CFG

CO5 : Construct turing Machine

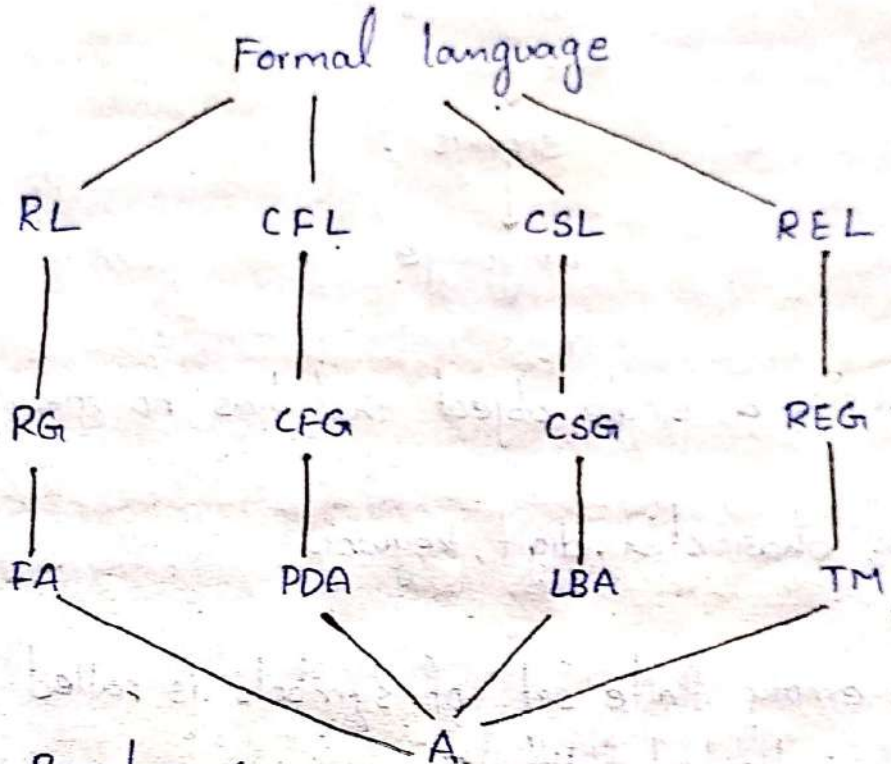
CO6 : Discuss dicidability theorey

UNIT-1

Alphabet, strings, language, finite Automata defination, Transition systems, acceptance of string by finite automata, DFA, Design of DFA, NFA, Design of NFA, equivalence between NFA and DFA, NFA with ϵ transition, Equivalence between NFA and ϵ-NFA, minimization of finite automata, moore and mealy machines and their equivalences, Application of finite automata:

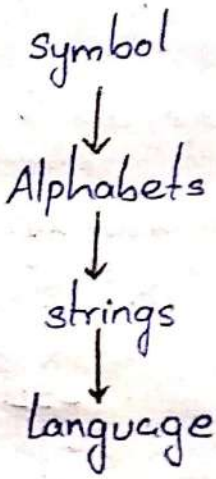
TIO: Overview of FLAT, identifying why do we have to study FLAT

Formal languages are divided into 4 types



RG - Regular Grammer
CFG - Context free Grammer
CSG - Context sensitive Grammer
REG - Recursively enumerated Grammer
FA - Finite Automata
PDA - Pushdown Automata
LBA - linear Bounded Automata
TM - Turing Machine
A - Automata
DFA - Deterministic finite Automata
NFA - Non deterministic finite Automata
RE - Regular expression
ϵ-NFA - ϵ - non deterministic finite Automata
TOC - Theorey of computations
ATFL - Automata Theorey and Formal language

ILO: know about symbols, Alphabets, strings and language



Symbol:

symbol is a single object that has no meaning by itself.

Example: character, digit, keyword.

Alphabet:

A non-empty finite set of symbols is called Alphabet.

It is indicated with "Σ"

Example

① $\Sigma = \{a, b, c\}$

② $\Sigma = \{a, b, c, \ldots, z\}$

③ $\Sigma = \{0, 1\}$

④ $\Sigma = \{-, \cup, \cap\}$

strings:

The sequence of symbols from the given alphabet 'Σ' called as string.

It is denoted with 'w' (or) 's'

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

$w_1 = a$

$w_2 = ab$

$w_3 = b$

$w_4 = abba$

$w_5 = abca$ (invalid)

* length of the string is nothing but how many numbers of symbols are present in that string. It is denoted as $|W|$ (or) $|S|$.

Example Consider the previous example

$|W_1| = 1$

$|W_2| = 2$

$|W_3| = 1$

$|W_4| = 4$

~~thus~~

* We can write the empty string as $W = \epsilon$

$|W| = 0$

Substrings of a string:

Consider the string W = PAVAN

$|W| = 5$

Substrings

0 length substring – ϵ
1 length substring – P, A, V, N
2 length substring – PA, AV, VA, AN
3 length substring – PAV, AVA, VAN
4 length substring – PAVA, AVAN
5 length substring – PAVAN

For this example trivial substrings are ϵ & pavan
Non-trivial substrings are remaining all substrings.

* Identify the prefixes for the following string

W = Vasavi

prefixes are ϵ, v, va, vas, vasa, vasav, Vasavi

* Identify the suffixes for the following string

W = Vasavi

suffixes are vasavi, ~~v~~asavi, savi, avi, vi, i, ϵ

* Identify the prefixes and suffixes for the string "gate"

prefixes are - ϵ, g, ga, gat, gate

suffixes are - gate, ate, te, e, ϵ

Language:

The collection of the string is called as language. It is denoted by "L"

Example: 1. $\Sigma = \{a, b\}$

$L = \{\epsilon, a, b, ab, ba, aab ----\}$

* The language may be either finite or infinite depends upon the condition.

Example 2. Derive the language L from the given alphabet $\Sigma = \{a, b\}$ where the strings length should be exactly 2

$L = \{aa, bb, ab, ba\}$

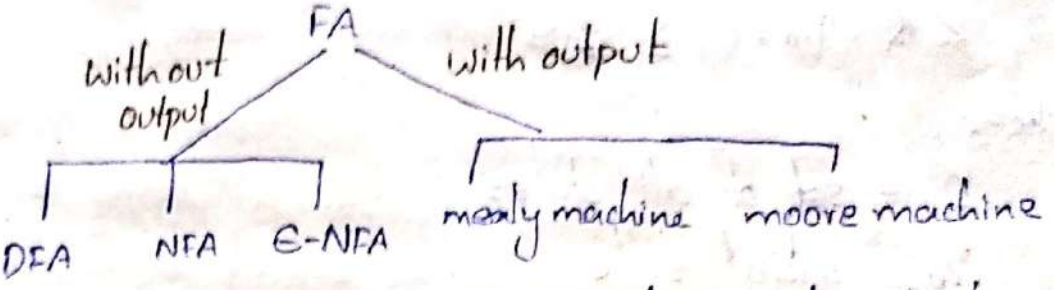
Example 3. string length should be atmost 2

$L = \{a, b, aa, bb, ab, ba\}$

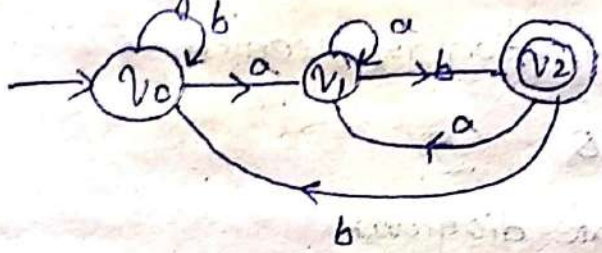
Example 4 string length should be atleast 2

$L = \{ab, ba, aa, bb, aba, -----\}$

I10: Discuss about finite automata defination, transition system and acceptance of a string by finite automata (FA)



* Consider the following FA which accepts the language where strings are ends with ab



* In finite automata we can have several states as shown below.

state is indicated with - ○

initial state is indicated with - ⟶○

final state is indicated with - ◎

From the above diagram the initial state is v_0 and final state is v_2

* In finite automata without output state diagram, there is only one initial state and there is atleast one final state

→ Defination of finite automata

We can define the finite automata with 5-tuple representation as shown below.

$$<Q, V_0, F, \Sigma, \delta>$$

where,

Q - set of all states

V_0 - initial state

F - set of final states, $F \subseteq Q$

Σ - input alphabet

δ - Transition function represented as

$$\delta : Q \times \Sigma \rightarrow Q$$

Conside the above state diagram

$$Q = \{V_0, V_1, V_2\}$$

initial state = V_0

$$F = \{V_2\}, \quad F \subseteq Q$$

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

$$\delta = Q \times \Sigma \rightarrow Q$$

↑ present state ↑ Next state

* we can represent the finite automata in

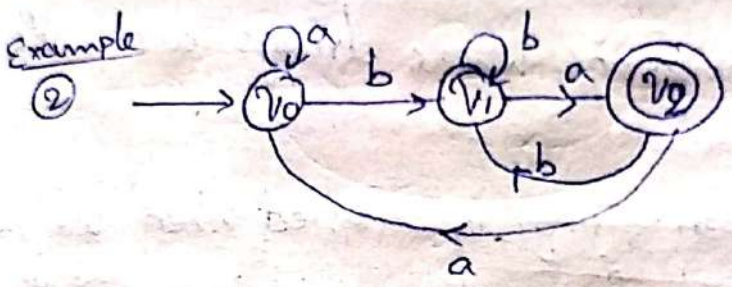
1. state diagram
2. state transition table is called as state transition system.

* We can represent the above state diagram in state transition table as shown below

Q ↓ / Σ →	a	b
→ q_0	q_1	q_0
q_1	q_1	q_2
(q_2)	q_1	q_0

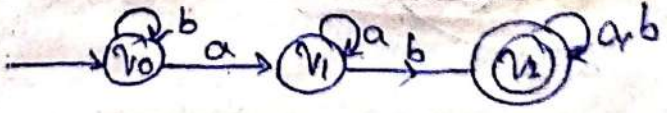
→ entries

Example ②



Q ↓ / Σ →	a	b
→ q_0	q_0	q_1
q_1	q_2	q_1
(q_2)	q_0	q_1

Example ③



Q ↓ / Σ →	a	b
q_0	q_1	q_0
q_1	q_1	q_2
q_2	q_2	q_2

→ Acceptance of a string by a finite automata

We know that string is nothing but a sequence of symbols from the given alphabet

Consider the input alphabet $\Sigma = \{a, b\}$

Consider the string where strings ends with ba

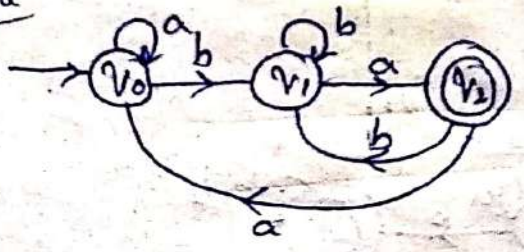
Generate the language where every string is ends with ba and verified by the corresponding finite automata

Consider the following finite automata state diagram which accepts the language where strings are ends with ba

We can say the given string is accepted when we are reading the last symbol of the string and in the f go to the final state, the string is accepted.

We can say the. We are reading the last symbol of the given string and go to the state which is not final state, now the string is rejected.

Example



① $W = aaba$

$\rightarrow q_0 \xrightarrow{a} q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_1 \xrightarrow{a} q_2$

② $W = ababaa$

$\rightarrow q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_1 \xrightarrow{a} q_2 \xrightarrow{b} q_1 \xrightarrow{a} q_2 \xrightarrow{a} q_0$

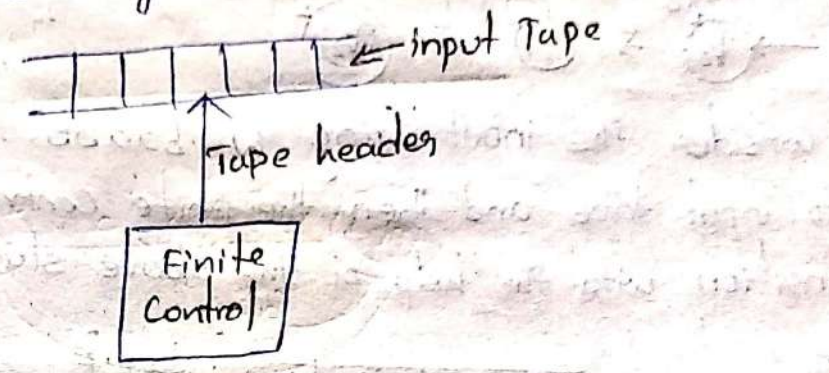
(3) $W = aaaabbbbab$

$\rightarrow q_0 \xrightarrow{a} q_0 \xrightarrow{a} q_0 \xrightarrow{a} q_0 \xrightarrow{a} q_0 \xrightarrow{b} q_1 \xrightarrow{b} q_1 \xrightarrow{b} q_1 \xrightarrow{a} q_2 \xrightarrow{b} q_1$

* The finite automata consists of three components. They are

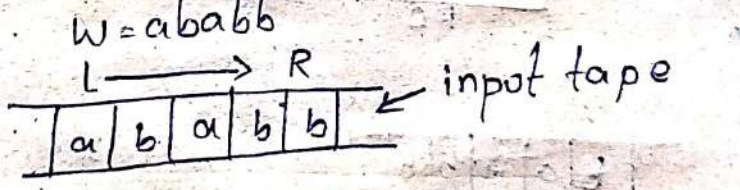
1. Input Tape
2. Tape header
3. Finite control

The components of the finite automata is represented in diagramatic way as shown below



Input tape:

- The input tape is divided into cells.
- Each cell is capable of storing only one input symbol.
- For example the string $W = ababb$ is the input string, it is stored into input tape from left to right as shown below



Tape header:

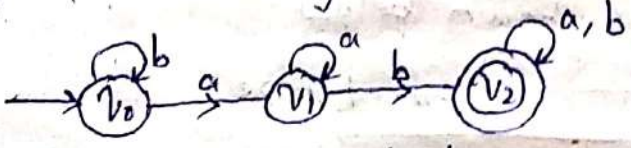
Here the header is indicated with arrow and it is pointing to one of the cell of the input tape at a time. i.e the tape header can pointing to only one cell at any moment.

Finite control:

The finite control controls the overall operation of the finite automata.

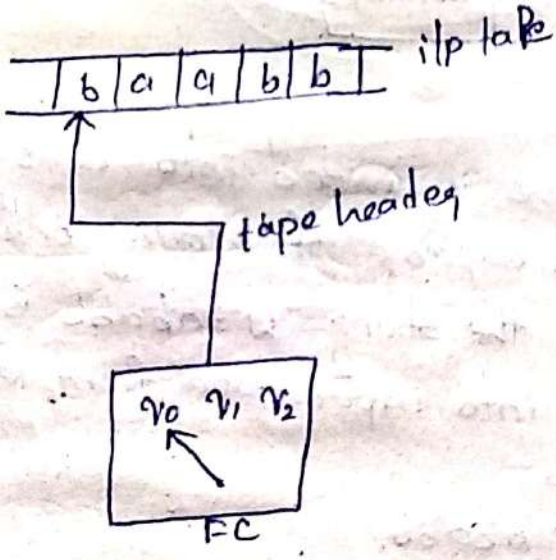
In finite control, the control can move from one state to another state or self state by reading the input symbol of the tape which is pointing by the tape header.

We can explain the finite automata in mathematical way by considering the following state diagram.

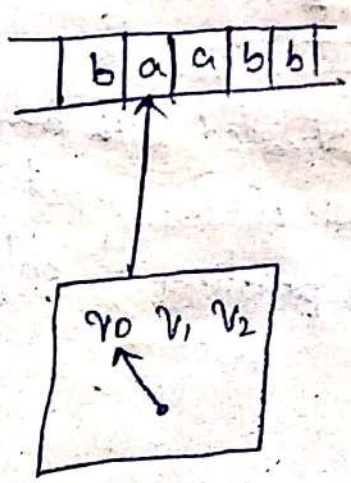


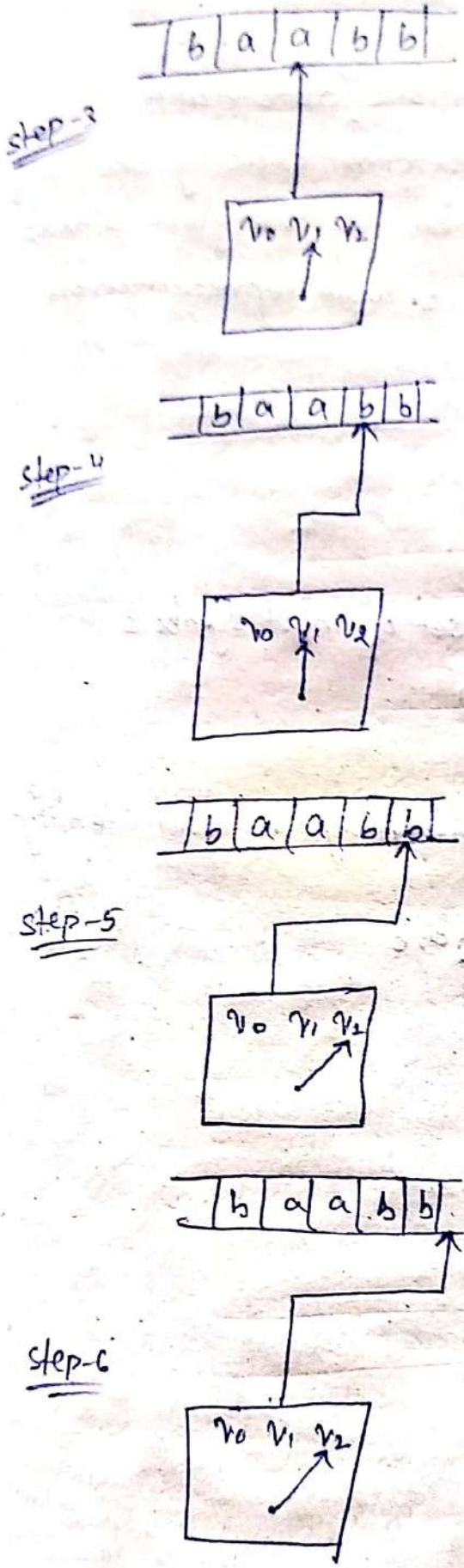
Consider the input string $w = baabb$, it can be stored into input tape and then the finite control perform the transition with the help of the above state diagram.

step 1



step-2




b a a b b
step-3
v_0 v_1 v_2
b a a b b
step-4
v_0 v_1 v_2
b a a b b
step-5
v_0 v_1 v_2
b a a b b
step-6
v_0 v_1 v_2

DFA:

DFA is expanded as deterministic finite automata.

In DFA every state should perform exactly one transition for every input symbol of the given alphabet.

We can define the DFA using 5 tuple representation as shown below.

$$< Q, q_0, \Sigma, \delta, F >$$

where Q = set of all states

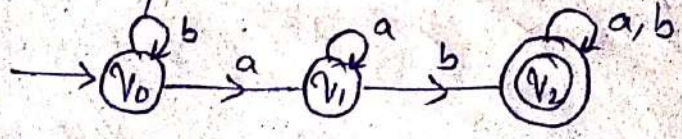
q_0 = initial state

Σ = input Alphabet (or) set of input symbols

δ = Transition function

F = set of final states

Consider the following DFA state diagram and identify the each tuple.



From the above diagram

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

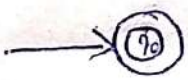
initial state $= q_0$

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

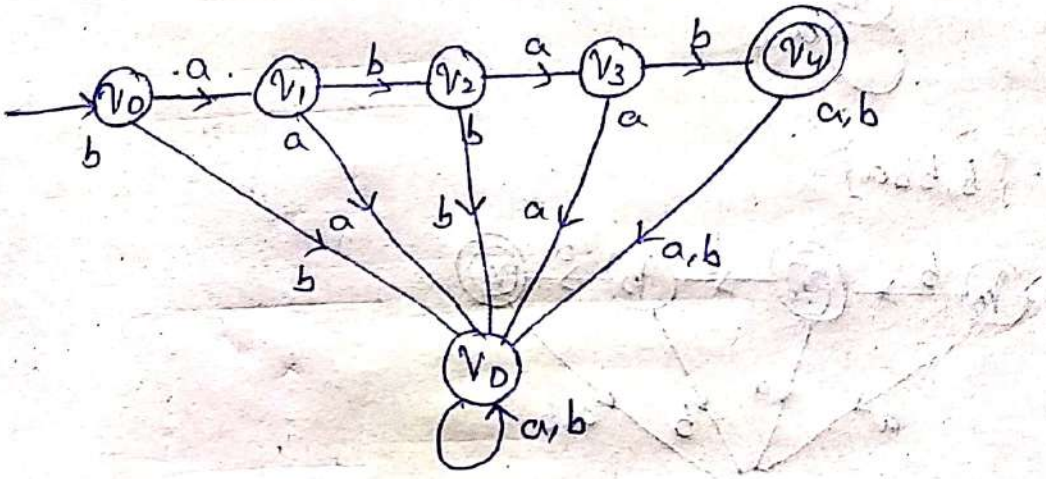
$\delta : Q \times \Sigma \longrightarrow Q$

$F = \{ q_2 \}$

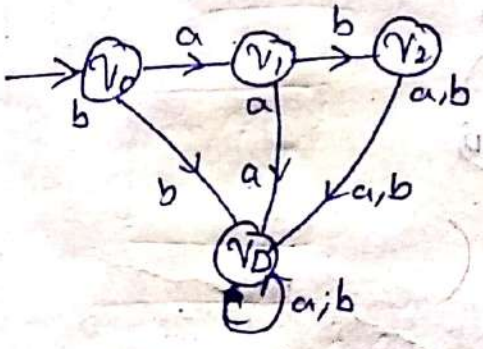
Design DFA to accept the language, ~~$L = \phi$ $\in$ (or) $L = \{\}$~~ ① $L = \in$



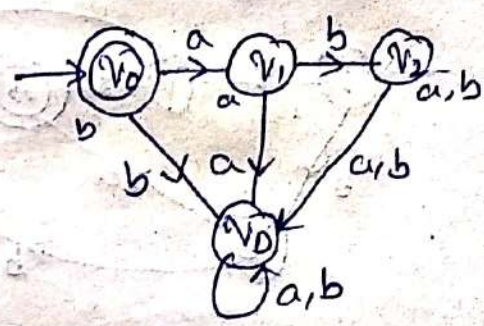
Design DFA to accept the language ② $L = \{ab, ab\}$



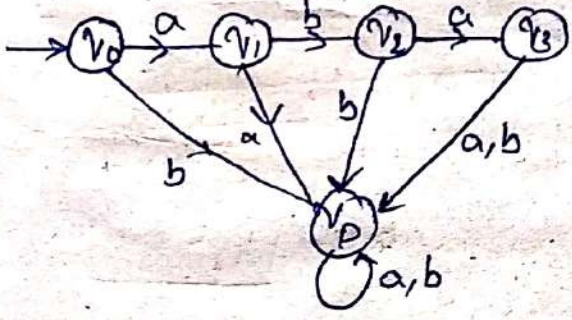
③ $L = \{ab\}$



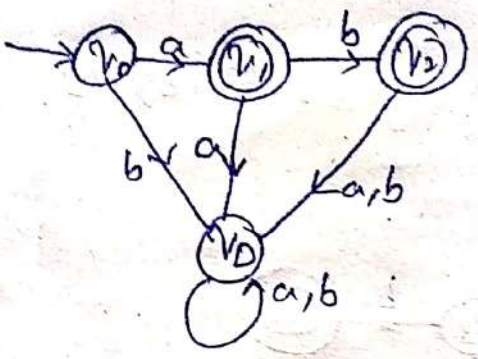
⑤ $L = \{\epsilon, ab\}$



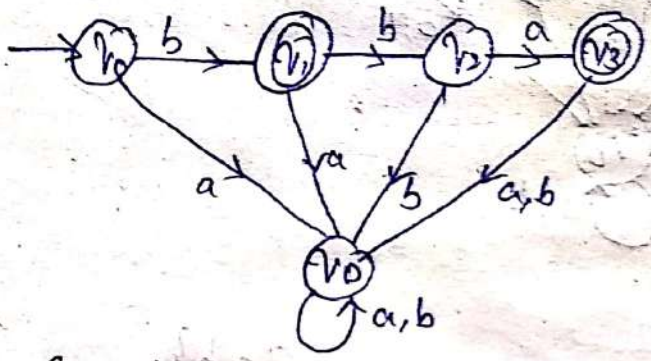
④ $L = \{aba\}$



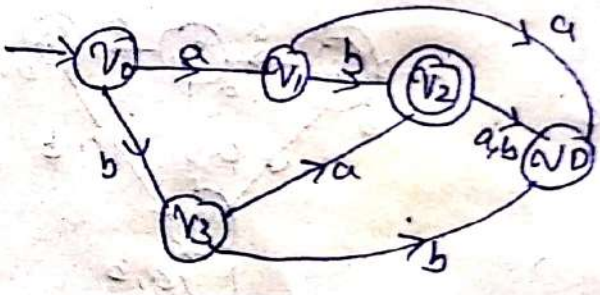
⑥ $L = \{a, ab\}$



⑦ $L = \{b, bba\}$



⑧ $L = \{ab, ba\}$

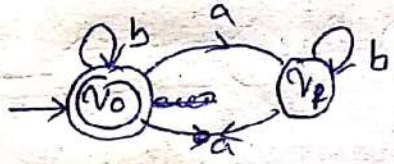


Design DFA to accept the language where every string contains even number of a's over $\Sigma = \{a, b\}$

The even numbers are 0, 2, 4, 6, 8 ....

ϵ, aa, aaaaa, ----

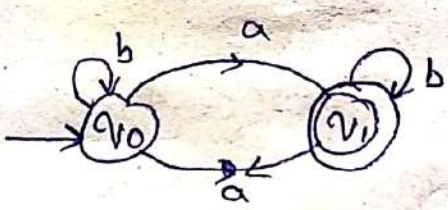
By adding '2' we can get the next state so we have to take two states.



Odd number of a's

Odd numbers 1, 3, 5, 7 ....

a, aaa, aaaaa ----



For ~~for~~ infinite language

1. Loops.
2. previous states
3. Dead state

Design DFA to accept the language over the alphabet $\Sigma = \{a, b\}$

① where every string starts with ab

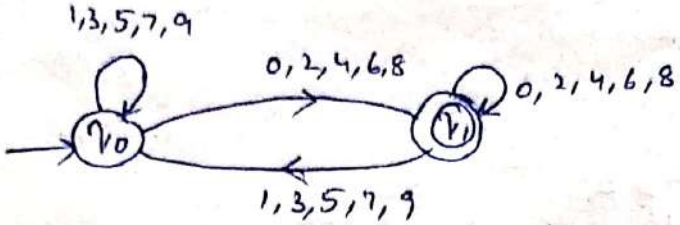
ab ------ , $L = \{ab, abab, ababb \cdots\}$

② where every string starts with ab

---- ab $L = \{ab, bab, bbab, \cdots\}$

③ ---ab--- $L = \{ab, aab, abaa, \cdots\}$

① Design DFA that accepts all integers which are multiple of 2 from $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

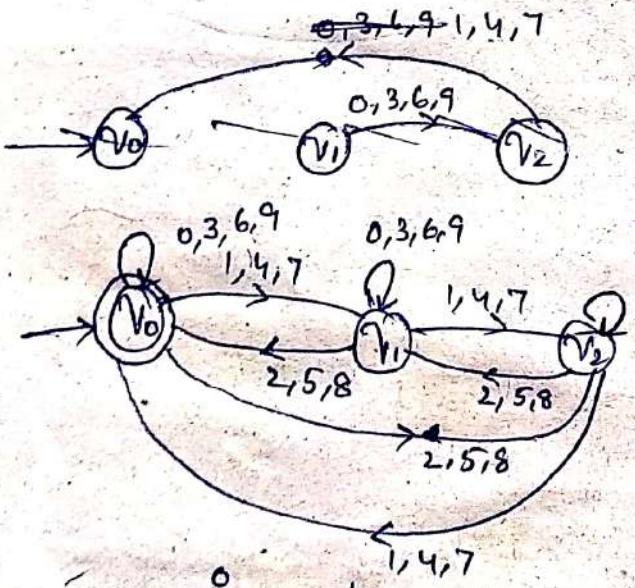


Graphical representation

Transition table representation

	0	1	2	3	4	5	6	7	8	9
→ q_0	q_1	q_0	q_1	q_0	q_1	q_0	q_1	q_0	q_1	q_0
(q_1)	q_1	q_0	q_1	q_0	q_1	q_0	q_1	q_0	q_1	q_0

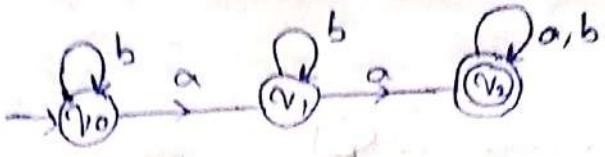
② Multiple of 3



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

③ Design DFA from $\Sigma = \{a, b\}$ that accepts atleast 2 a's

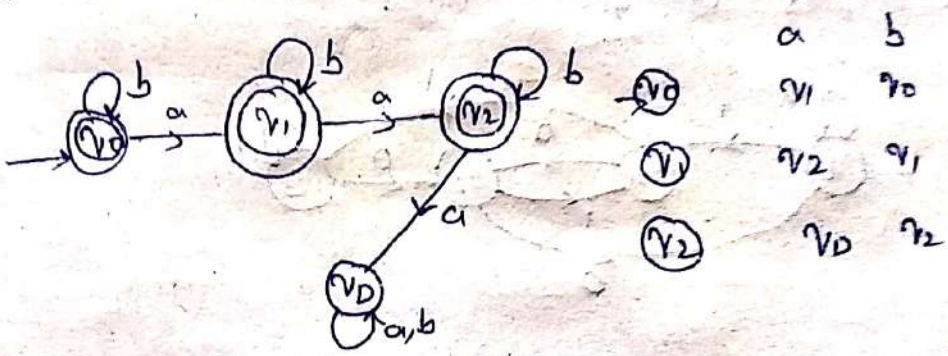
$L = \{aa, aba, baa, aab, \ldots\}$



Transition table.

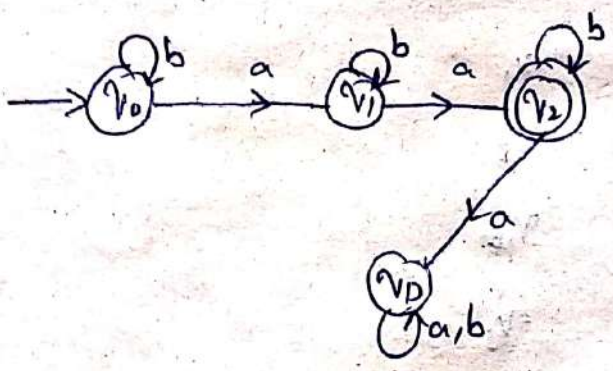
	a	b
$\rightarrow v_0$	v_1	v_0
v_1	v_2	v_1
(v_2)	v_2	v_2

④ Atmost 2 a's



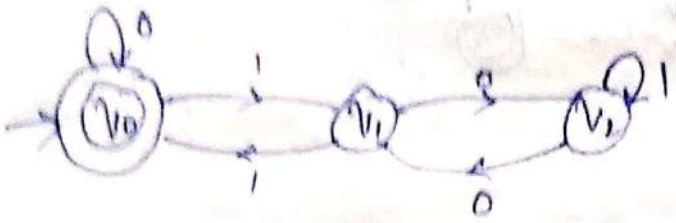
	a	b
v_0	v_1	v_0
v_1	v_2	v_1
v_2	v_D	v_2

⑤ Exact 2 a's



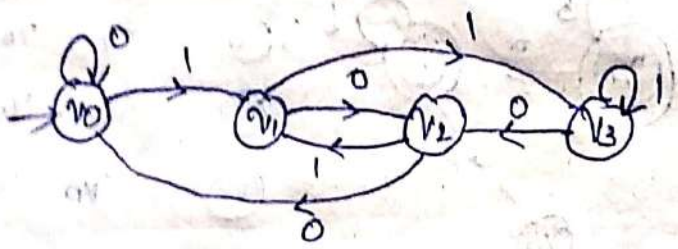
	a	b
$\rightarrow v_0$	v_1	v_0
v_1	v_2	v_1
(v_2)	v_D	v_2

(3) Design a ~~define~~ dfa which accepts binary strings which are multiples of 3



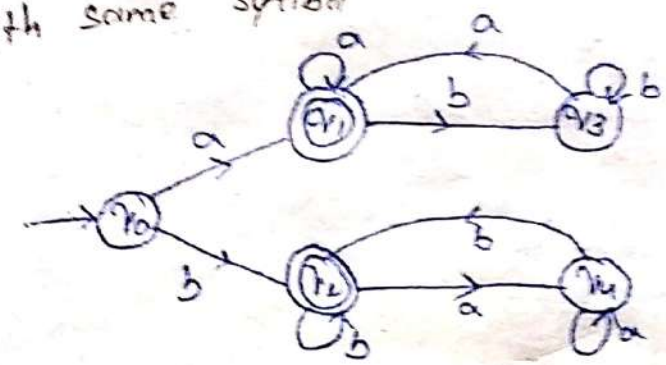
	0	1
v_0	v_0	v_1
v_1	v_2	v_0
v_2	v_1	v_3

(4) Multiples of 4

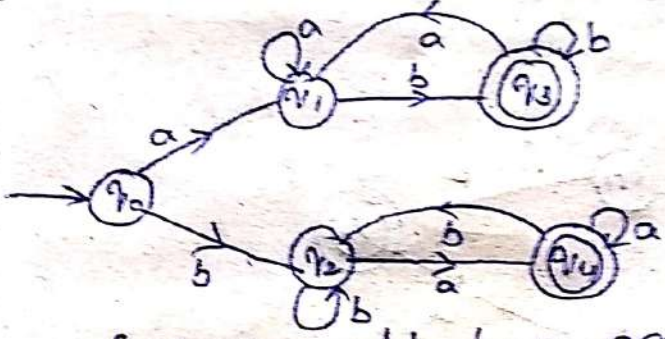


	0	1
v_0	v_0	v_1
v_1	v_2	v_3
v_2	v_0	v_1
v_3	v_2	v_3

5. Define a DFA over $\Sigma = \{a, b\}$ that starts and ends with same symbol



6. starts and end with different symbol.



$L = \{ab, ba, abb, baa, aab, bba - - -\}$

NFA: (Non deteaministic finite automata.

$\{Q, v_0, F, \delta, \Sigma\}$

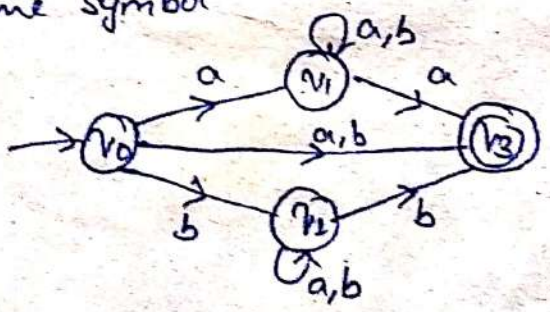
Q - set of all states
v_0 - initial state
F = set of all final states

$\delta \rightarrow Q \times \Sigma = 2^Q$

$\Sigma \rightarrow$ input alphabet

Example:

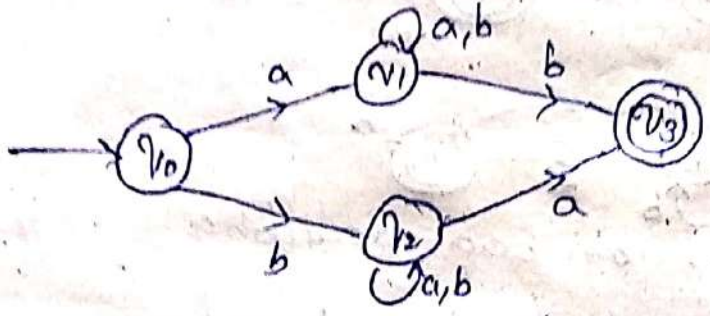
Design a NFA over $\Sigma = \{a, b\}$ that starts and ends with same symbol



Transition table :

	a	b
v_0	$\{v_1, v_2\}$	$\{v_2, v_3\}$
v_1	$\{v_1, v_3\}$	v_1
v_2	$\{v_2\}$	$\{v_2, v_3\}$
v_3	—	—

Design a NFA over $\Sigma = \{a, b\}$ that starts and ends with different symbol



	a	b
v_0	$\{v_1\}$	$\{v_2\}$
v_1	$\{v_1\}$	$\{v_1, v_3\}$
v_2	$\{v_2, v_3\}$	$\{v_2\}$
v_3	—	—

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

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

$\Sigma^1 = \{a, b\}$

$\Sigma^2 = \{aa, ab, ba, bb\}$

Σ^k = set of all strings of length k

$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \cdots \cup \Sigma^k$

$\Sigma^* = \sum_{k=0}^{\infty} \Sigma^k$

$\Sigma^+ = \sum_{k=1}^{\infty} \Sigma^k$

Construct DFA for the language over $\Sigma = \{a\}$, $L = \{a^n / n \geq 0\}$

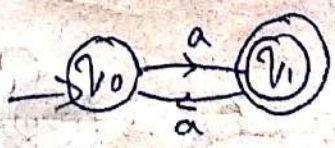
$L = \{a^n / n \geq 0\}$

$L = \{\epsilon, a, aa, aaa, \cdots\}$

Construct DFA for the language over $\Sigma = \{a\}$, $L = \{a^{2n+1} / n \geq 0\}$

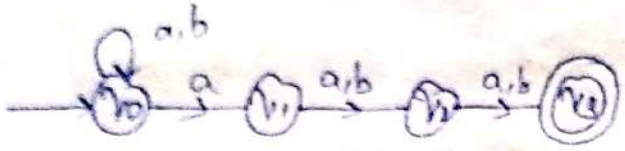
$L = \{a^{2n+1} / n \geq 0\}$

$L = \{a, aaa, aaaaa, \cdots\}$



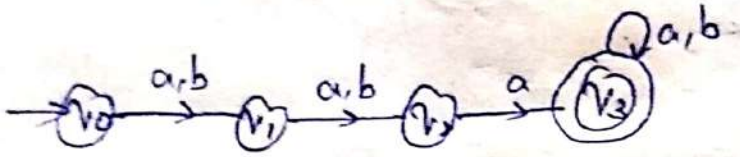
Construct NFA over $\Sigma = \{a, b\}$ such that 3rd symbol from the right end is a

$L = \{aaaa, aaab, baaa, abab, \dots\}$

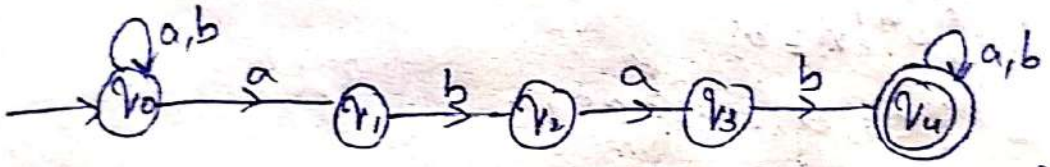


Left end

$L = \{aaa, aba, baa, bba, aaab, aaaa, abab, \dots\}$

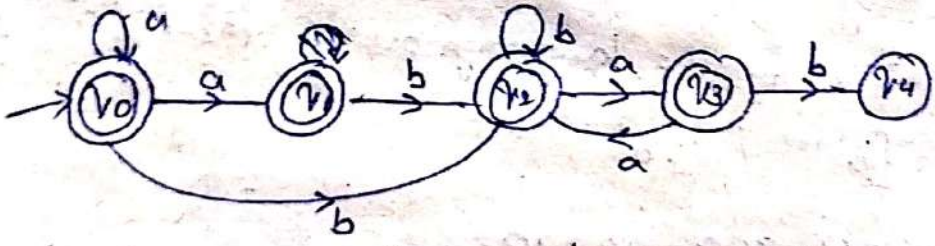


Construct NFA over $\Sigma = \{a, b\}$ such that having abab as substring

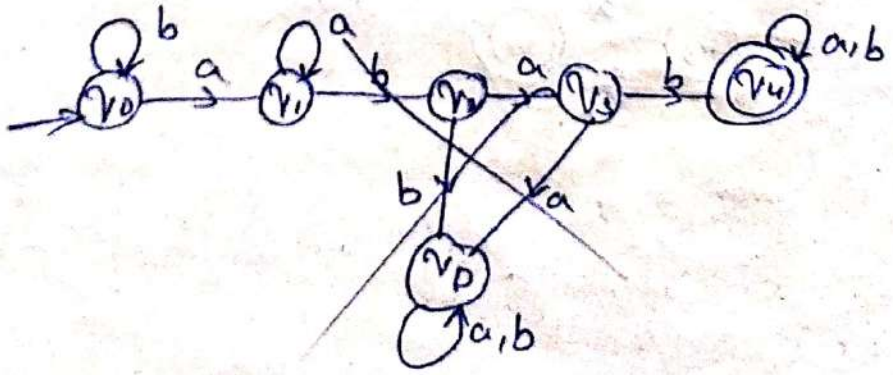


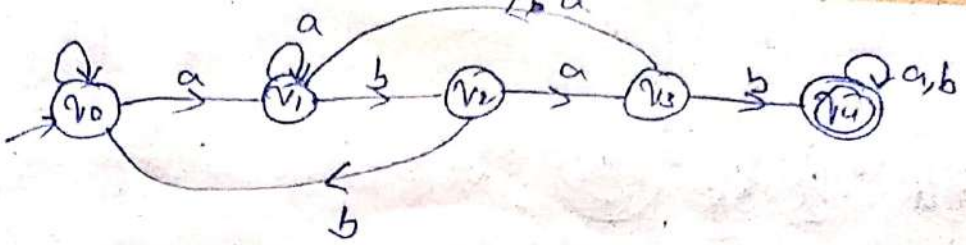
$L = \{abab, aabab, ababa, ababb \dots\}$

does not have abab as substring



construct DFA that accepts the substring abab





Conversion of NFA to DFA

Let $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ be the NFA

$D = (Q_D, \Sigma, \delta_D, q_D, F_D)$

no change in the initial state and Alphabet (q_0, Σ)

S = power set of F_N, $S \cap F_N \neq \phi$

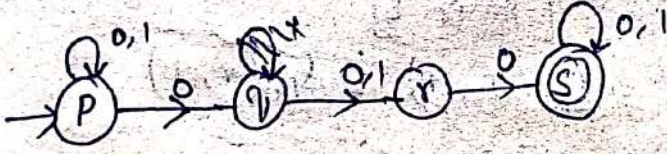
for each sub. set $S \subseteq Q_N$ and for each i/p symbol Σ

$$\delta_D(S, a) = \bigcup_{p \text{ in } S} \delta_N(p, a)$$

Problem

	0	1
$\rightarrow P$	{P, q}	{P}
q	{r}	{r}
r	{s}	ϕ
*S	{s}	{s}

Sol



	0	1
$\rightarrow P$	[P q]	[P]
[P q]	[P q r]	[P r]
[P q r]	[P q r s]	[P r]
[P r]	[P q s]	[P]
* [P q r s]	[P q r s]	[P r s]
* [P q s]	[P q r s]	[P r s]

	0	1
*[P r s]	[P q s]	[P s]
* [P s]	[P q s]	[P s]

$\delta_D([P, \nu], \sigma) = \delta_N(P, 0) \cup \delta_N(\nu, 0)$

$= [P \cdot \nu] \cup [r]$

$= [P \nu r]$

$\delta_D([P \nu], 1) = \delta_N(P, \phi) \cup \delta_N(\nu, 1)$

$= [P] \cup [r]$

$= [P\ r]$

$\delta_D([P\ \nu\ r], 0) = \delta_N(P, 0) \cup \delta_N(\nu, 0) \cup \delta_N(r, 0)$

$= [P\ \nu] \cup [r] \cup [s]$

$= [P\ \nu\ r\ s]$

$\delta_D([P\ \nu\ r], 1) = \delta_N(P, 1) \cup \delta_N(\nu, 1) \cup \delta_N(r, 1)$

$= [P] \cup [r]$

$= [P \phi]$

$\delta_D([P \phi], 0) = \delta_N(P, 0) \cup \delta_N(r, 0)$

$= [P, \nu] \cup [s]$

$= [P\ \nu\ s]$

$\delta_D([P\ r], 1) = \delta_N(P, 1) \cup \delta_N(r, 1)$

$= [P] \cup \phi$

$= [P]$

$\delta_D([P\ \nu\ r\ s], 0) = \delta_N(P, 0) \cup \delta_N(\nu, 0) \cup \delta_N(r, 0) \cup \delta_N(s, 0)$

$= [P\ \nu] \cup [r] \cup [s] \cup [s]$

$= [P\ \nu\ r\ s]$

$\delta_D([P\ \nu\ r\ s], 1) = \delta_N(P, 1) \cup \delta_N(Q, 1) \cup \delta_N(R, 1) \cup \delta_N(s)$

$= [P] \cup [r] \cup [\phi] \cup [s] = [P\ r\ s]$

$\delta_D([P \vee S], 0) = \delta_N(P, 0) \cup \delta_N(\vee, 0) \cup \delta_N(S, 0)$

$= [P \vee] \cup [\vee] \cup [S]$

$= [P \vee \vee S]$

$\delta_D([P \vee S], 1) = \delta_N(P, 1) \cup \delta_N(\vee, 1) \cup \delta_N(S, 1)$

$= (P) \cup [\vee] \cup [S]$

$= [P \vee \vee S]$

$\delta_D([P \vee S], 0) = \delta_N(P, 0) \cup \delta_N(\vee, 0) \cup \delta_N(S, 0)$

$= [P \vee] \cup [S] \cup [S]$

$= [P \vee S]$

$\delta_D([P \vee S], 1) = \delta_N(P, 1) \cup \delta_N(\vee, 1) \cup \delta_N(S, 1)$

$= [P] \cup [\phi] \cup [S]$

$= [P \vee S]$

$\delta_D([P S], 0) = \delta_N(P, 0) \cup \delta_N(S, 0)$

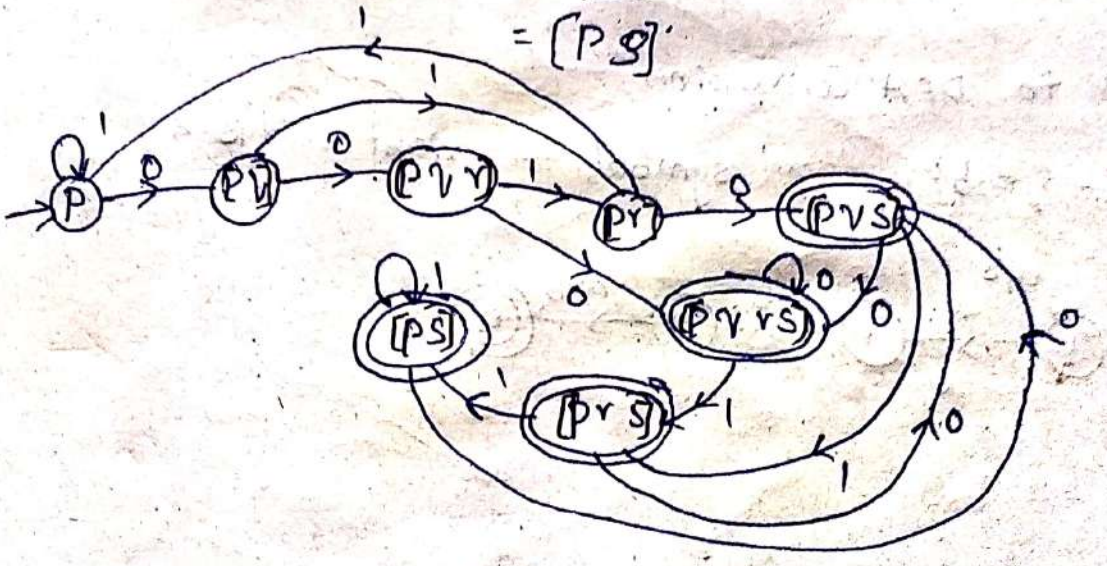
$= [P \vee] \cup [S]$

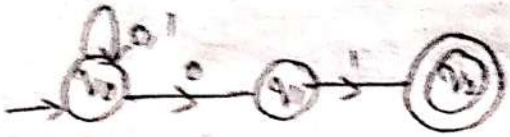
$= [P \vee S]$

$\delta_D([P S], 1) = \delta_N(P, 1) \cup \delta_N(S, 1)$

$= [P] \cup [S]$

$= [P S]$

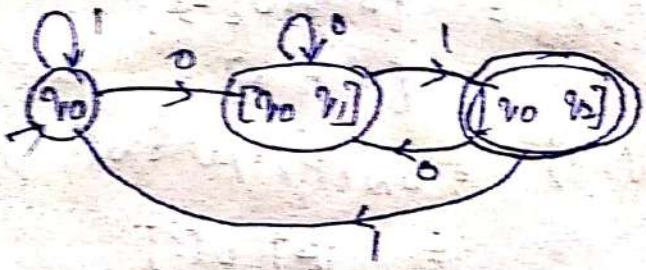


2. 

$\delta'[q_0, 1] = \delta_N(q_0, 1) \cup \delta_N(q_1, 1)$
$= [q_0\ q_1]$

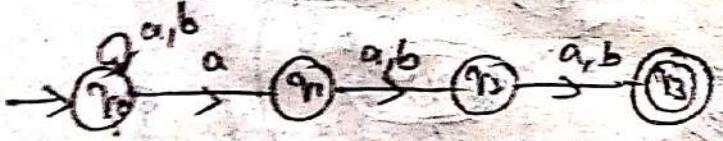
	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	q_0
q_1	ϕ	q_2
$*q_2$	ϕ	ϕ

	0	1
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



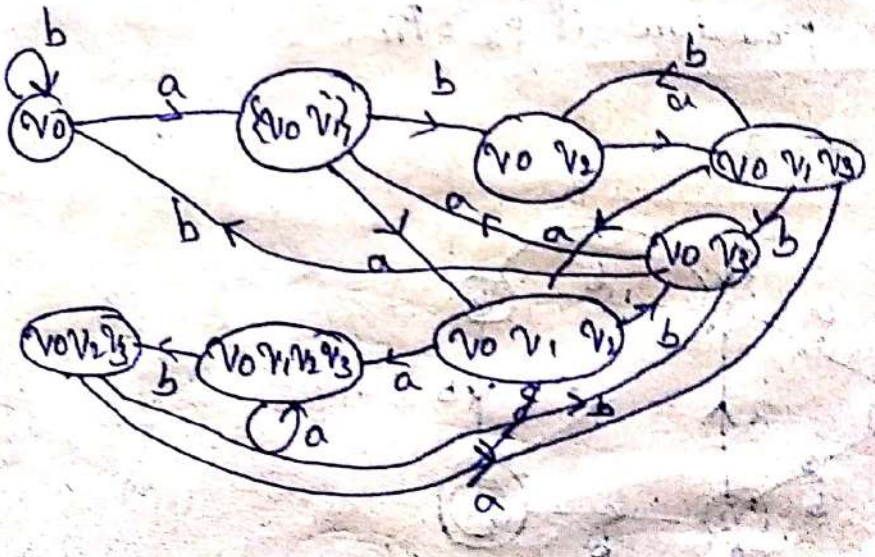
NFA to DFA Conversion

$\Sigma = \{a, b\}$ — 3rd symbol for right end is a



	a	b
→v_0	$\{v_0, v_1\}$	v_0
v_1	v_2	v_2
v_2	v_3	v_3
*v_3	ϕ	ϕ

	a	b
→v_0	$[v_0\ v_1]$	v_0
$[v_0\ v_1]$	$[v_0\ v_1\ v_2]$	$[v_0\ v_2]$
$[v_0\ v_1\ v_2]$	$[v_0\ v_1\ v_2\ v_3]$	$[v_0\ v_2\ v_3]$
$[v_0\ v_2]$	$[v_0\ v_1\ v_3]$	$[v_0\ v_3]$
*$[v_0\ v_1\ v_2\ v_3]$	$[v_0\ v_1\ v_2\ v_3]$	$[v_0\ v_2\ v_3]$
*$[v_0\ v_2\ v_3]$	$[v_0\ v_1\ v_3]$	$[v_0\ v_3]$
*$[v_0\ v_1\ v_3]$	$[v_0\ v_1\ v_2]$	$[v_0\ v_3]$
*$[v_0\ v_3]$	$[v_0\ v_1]$	v_0



MINIMIZATION of DFA:

State equivalence Method:

step-1

1. identify the dead states and unreachable states
2. Draw the transition table without dead states and unreachable states.
3. partition the states initially into two sets

$P_0 = \{$ all non final states $\}, \{$ all the final states $\}$

step-2

Construct P_k from the set P_{k-1} repeatedly based on distinguishable states in P_{k-1}

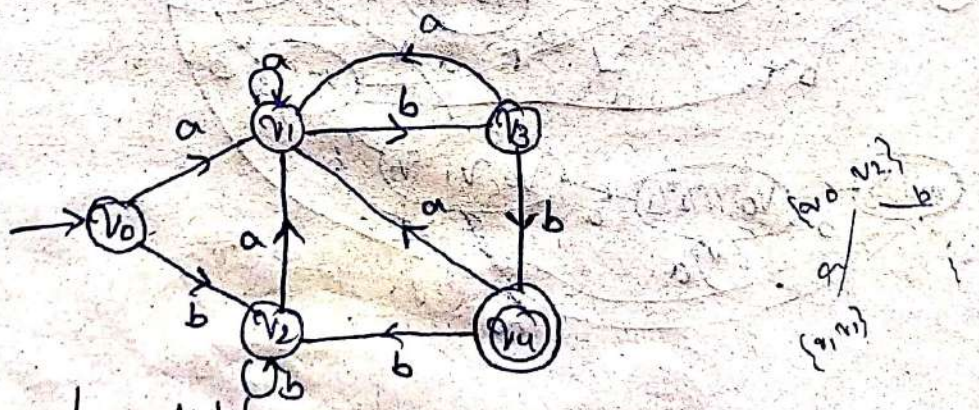
(a) let v_i, v_j be the 2 states from the same partition set v_i, v_j are distinguishable.

$\delta(v_i, a)$ and $\delta(v_j, a)$ leads to different partition sets in P_{k-1}

step-3 Stop the procedure if $P_k = P_{k-1}$

Problem

1.



Transition table:

	a	b
→ v_0	v_1	v_2
v_1	v_1	v_3
v_2	v_1	v_2
v_3	v_1	v_4
* v_4	v_1	v_2

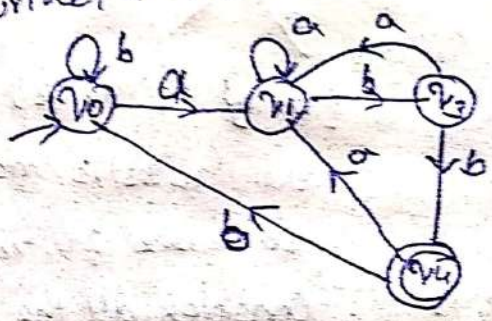
$P_0 = \{v_0, v_1, v_2, v_3\}, \{v_4\}$

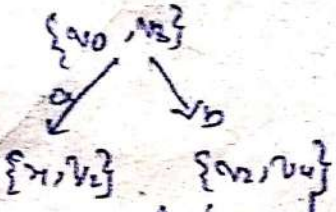
$P_1 = \{v_0, v_1, v_2\}, \{v_3\}, \{v_4\}$

$P_2 = \{v_0, v_2\}, \{v_1\}, \{v_3\}, \{v_4\}$

$P_3 = \{v_0, v_2\}, \{v_1\}, \{v_3\}, \{v_4\}$

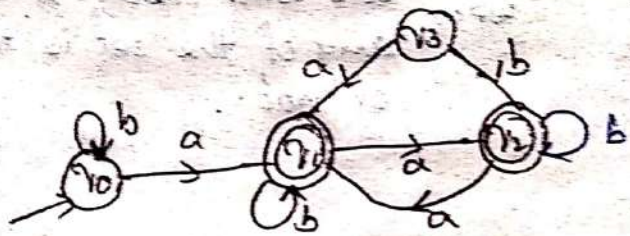
Here P_2 and P_3 are same so there is no possibility for further partition. so merge v_0 and v_2





= The set doesnot contain v_4 so we have to apply partition.

2.



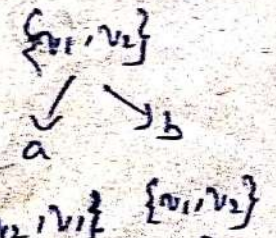
In this v_3 is unreachable state. So we to remove v_3

transition table:

	a	b
$\rightarrow v_0$	v_1	v_0
v_1	v_2	v_1
$* v_2$	v_1	v_2

$P_0 = \{v_0\}, \{v_1, v_2\}$

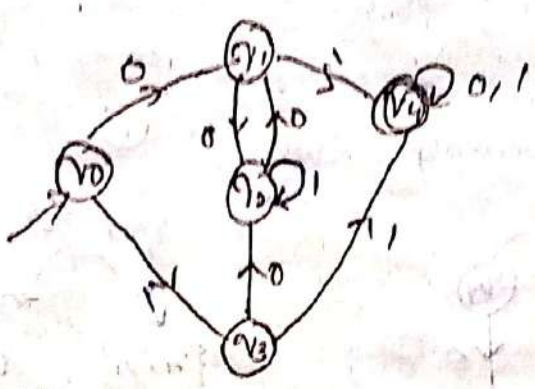
$P_1 = \{v_0\}, \{v_1, v_2\}$



In this v_1 and v_2 are same there is possible for further transition.

So merge v_1 and v_2

2.



Myhill-Nerode Method

	v_0	v_1	v_2	v_3	v_4
v_0					
v_1	x				
v_2	x	x			
v_3	x	x	x		
v_4	x	x	x	x	

{v_1, v_0}

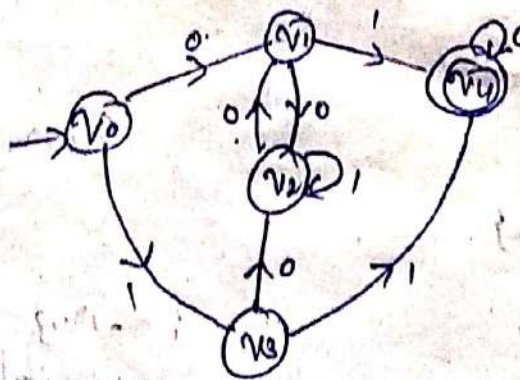
0 / \ 1

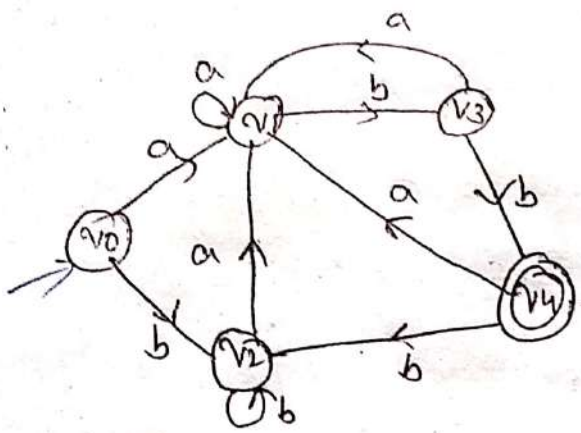
{v_2, v_1} {v_4, v_3}

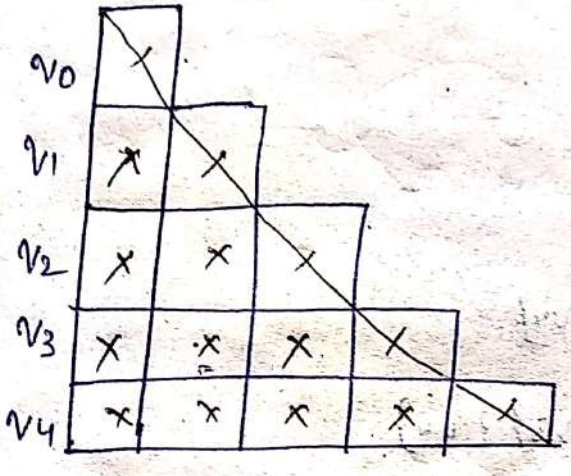
Here (v_4 v_3) is marked.

So mark {v_1, v_0}. Like Consider all the unmarked sets and checks whether any of the set is marked or not.

In this all the states are marked. so we dont have to merge any states.







① $\{v_1, v_0\}$

a / \ b

$\{v_1, v_1\}$ $\{v_3, v_2\}$

② $\{v_2, v_0\}$

a / \ b

$\{v_1, v_1\}$ $\{v_2, v_2\}$

③ $\{v_2, v_1\}$

a / \ b

$\{v_1, v_1\}$ $\{v_2, v_3\}$

④ $\{v_3, v_0\}$

a / \ b

$\{v_1, v_1\}$ $\{v_4, v_2\}$

⑤ $\{v_3, v_1\}$

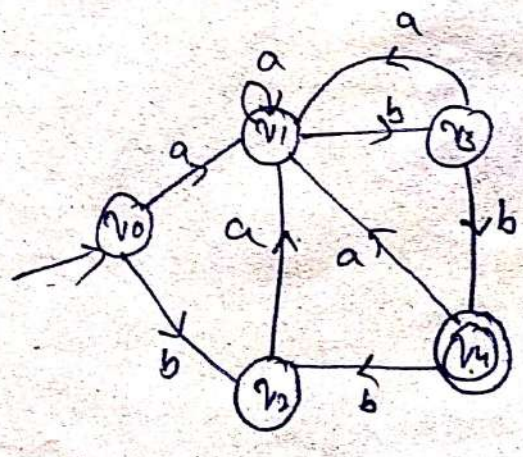
a / \ b

$\{v_1, v_1\}$ $\{v_4, v_3\}$

⑥ (v_3, v_2)

a / \ b

$\{v_1, v_1\}$ $\{v_4, v_2\}$



Method:-

partition method

Transmition table.

	a	b
→ v_0	v_1	v_2
v_1	v_1	v_3
v_2	v_1	v_2
v_3	v_1	v_4
* v_4	v_2	v_2

$P_0 = \{v_0, v_1, v_2, v_3\}\ \{v_4\}$

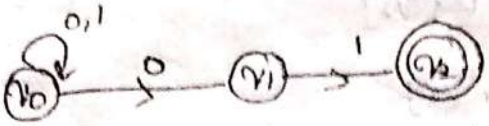
$P_1 = \{v_0\ v_1\ v_2\}\ \{v_3\}\ \{v_4\}$

$P_2 = \{v_0, v_2\}\ \{v_1\}\ \{v_3\}\ \{v_4\}$

$P_3 = \{v_0\}\ \{v_2\}\ \{v_1\}\ \{v_3\}\ \{v_4\}$

Extended transistions on NFA:

Problem

1. 

string is 01001

it accepts set of binary strings that ends with 01

$(q_0, \epsilon) = q_0$

$(q_0, 0) = \{q_0, q_1\}$

$(q_0, 01) = \delta(q_0, 1) \cup \delta(q_1, 1)$

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

$= \{q_0, q_2\}$

$(q_0, 010) = \delta(q_0, 0) \cup \delta(q_2, 0)$

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

$= \{q_0, q_1\}$

$(q_0, 0100) = \delta(q_0, 0) \cup \delta(q_1, 0)$

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

$= \{q_0, q_1\}$

$(q_0, 01001) = \delta(q_0, 1) \cup \delta(q_1, 1)$

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

$= \{q_0, q_2\}$

in the final set it contains a final state so the string is accepted.

ii. 1010

$(q_0, \epsilon) = q_0$

$(q_0, 1) = q_0$

$(q_0, 10) = \delta(q_0, 0)$

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

$= \{q_0, q_1\}$

$(q_0, 101) = \delta(q_0, 1) \cup \delta(q_1, 1)$

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

$= \{q_0, q_2\}$

$(q_0, 1010) = \delta(q_0, 0) \cup \delta(q_2, 0)$

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

$= \{q_0, q_1\}$

The final set does not contain final state. So the NFA rejects the string 1010

ϵ-NFA :

* NFA is allowed to make transitions spontaneously without receiving any input symbol
* The numbers of languages accepted by DFA, NFA and ϵ-NFA are same
* $(Q, \Sigma, \delta, q_0, F)$

Q - set of all states

Σ = input alphabet

$\delta : Q \times \Sigma \cup \{\epsilon\} \longrightarrow 2^Q$ (ϵ-NFA)

q_0 : initial state

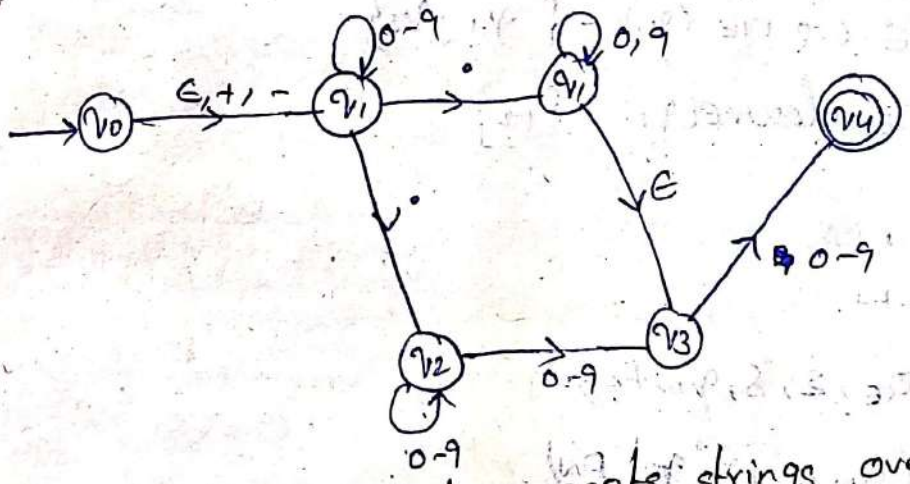
F : set of all final states

Example

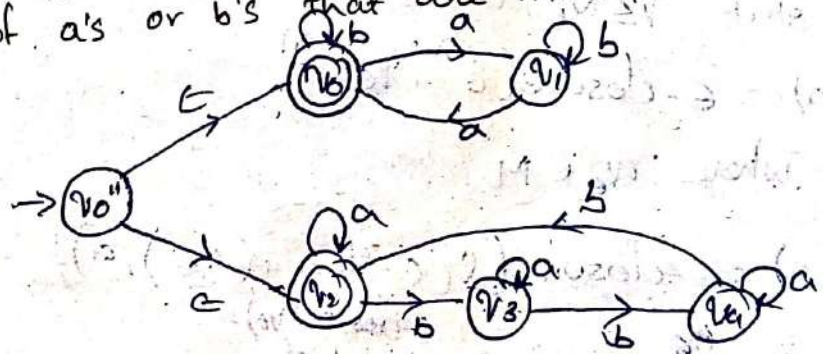
Problem: ∈-NFA that accepts decimal numbers consisting of

1. an optional + or - sign
2. string of digits.
3. another string of digits (either the string of digits or the string can be empty but atleast two stings of the digits must be non-empty.

Solution



Construct a ∈-NFA that accepts strings over $\Sigma = \{a, b\}$ such that the string should contain even no. of a's or b's that are multiplies of 3.



$L = \{aa, abab b, aaaaa, bbb, \ldots\}$

ε-NFA to NFA

$\delta(q_0, \epsilon)$ (or) ϵ-closure $(q_0) = \{q_0, q_1, q_2\}$

ϵ-closure of state q is the set of states in which the next state is come without using any input symbol

$\delta(q_1, \epsilon)$ (or) ϵ-closure $(q_1) = \{q_1, q_2\}$

$\delta(q_2, \epsilon)$ (or) ϵ-closure$(q_2) = \{q_2\}$

* M – ϵ-NFA

M' – NFA

$M = (Q_\epsilon, \Sigma, \delta, q_0, F_\epsilon)$

$M' = (Q_N, \epsilon, \delta', q_0', F_N)$

$q_0' = \epsilon$-closure(q_0)

for a state $q \subseteq Q_N$, $a \in \Sigma$

$\delta'(q, a) = \epsilon$-closure$(\delta^\wedge(q_\epsilon, a))$

where $q_\epsilon \in M$

$\delta'(q, a) = \epsilon$-closure$(\delta(\epsilon$-closure$(q_\epsilon), a))$

$$F_N = \begin{cases} F_\epsilon \cup q_0 & \text{if } F_\epsilon \cap \epsilon\text{-closure}(q_0) \neq \phi \\ F_\epsilon & \text{if } F_\epsilon \cap \epsilon\text{-closure}(q_0) = \phi \end{cases}$$

δ	0	1	2	ϵ
v_0	v_0	ϕ	ϕ	v_1
v_1	ϕ	v_1	ϕ	v_2
v_2	ϕ	ϕ	v_2	ϕ

$\delta'(v_0,0) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(v_0),0))$

$= \epsilon\text{-closure}(\delta(\{v_0,v_1,v_2\},0))$

$= \epsilon\text{-closure}(\delta(v_0,0) \cup \delta(v_1,0) \cup \delta(v_2,0))$

$= \epsilon\text{-closure}(\{v_0\} \cup \phi \cup \phi)$

$= \epsilon\text{-dovre}(v_0)$

$= \{v_0, v_1, v_2\}$

$\delta'(v_0,b) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(v_0),1))$

$= \epsilon\text{-closure}(\delta(\{v_0,v_1,v_2\},1))$

$= \epsilon\text{-closure}(\delta(v_0,1) \cup \delta(v_0,1) \cup \delta(v_2,1))$

$= \epsilon\text{-closure}(\{\phi\},\{v_1\},\phi)$

$= \epsilon\text{-closure}(v_1)$

$= \{v_1, v_2\}$

$\delta'(v_0,2) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(v_0),2))$

$= \epsilon\text{-closure}(\delta(\{v_0,v_1,v_2\},2))$

$= \epsilon\text{-closure}(\delta(v_0,2) \cup \delta(v_1,2) \cup \delta(v_2,2))$

$= \epsilon\text{-closure}(\{\phi\},\phi,\{v_2\})$

$= \epsilon\text{-closure}(v_2)$

$= \{v_2\}$

$\delta'(v_1,0) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(v_1),0))$

$= \epsilon\text{-closure}(\delta(\{v_1,v_2\},0))$

$= \epsilon\text{-closure}(\delta(v_1,0) \cup \delta(v_2,0))$

$= \epsilon\text{-closure}(\phi \cup \phi)$

$= \epsilon\text{-closure}(\phi)$

$\delta'(v_1, 0) = \{v_1, v_2\}$ ϕ

$\delta'(v_1, 1) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(v_1)), 1)$

$= \epsilon\text{-closure}(\delta(\epsilon\text{-closure}\{v_1, v_2\}, 1))$

$= \epsilon\text{-closure}(\delta(v_1, 1) \cup \delta(v_2, 1))$

$= \epsilon\text{-closure}(\{v_3\} \cup \phi)$

$= \epsilon\text{-closure}(v_3)$

$= \{v_3, v_2\}$

$\delta'(v_1, 2) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(v_1)), 2)$

$= \epsilon\text{-closure}(\delta(\{v_1, v_2\}, 2))$

$= \epsilon\text{-closure}(\delta(v_1, 2) \cup \delta(v_2, 2))$

$= \epsilon\text{-closure}(\phi \cup \{v_2\})$

$= \epsilon\text{-closure}(v_2)$

$= \{v_2\}$

$\delta'(v_2, 0) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(v_2), 0))$

$= \epsilon\text{-closure}(\delta\{v_2\}, 0)$

$= \epsilon\text{-closure}(\delta(v_2, 0))$

$= \epsilon\text{-closure}(\phi)$

$= \phi$

$\delta'(v_2, 1) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(v_2), 1))$

$= \epsilon\text{-closure}(\delta(v_2, \phi))$

$= \epsilon\text{-closure}(\phi)$

$= \phi$

$\delta'(v_2, 2) = \epsilon\text{-closure}(\delta(\epsilon\text{-closure}(v_2), 2))$

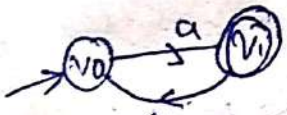
$= \epsilon\text{-closure}(\delta(v_2, 2))$

$= \epsilon\text{-closure}(v_2)$

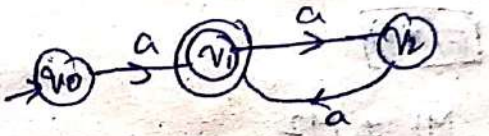
$= \{v_2\}$

δ'	0	1	2
v_0	$\{v_0, v_1, v_2\}$	$\{v_1, v_2\}$	$\{v_2\}$
v_1	ϕ	$\{v_1, v_2\}$	$\{v_2\}$
v_2	ϕ	ϕ	$\{v_2\}$

Equivalence of two Finite Automata:



M1



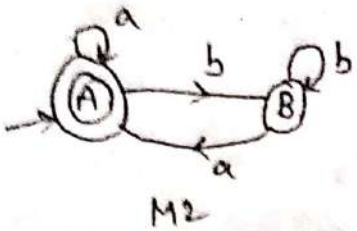
M2

These two finite automata's are accepting the same languages then they are Equivalence.

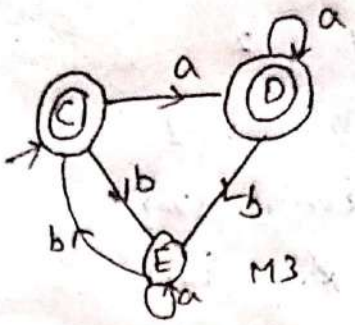
$L(M1) = L(M2)$ = odd no. of a's

Procedure to check for the equivalence of 2 Finite Automata's:

1. Construct the transition table that has a pair of states (P, Q) where $P \in M1$ and $Q \in M2$
2. Start the construction of the transistion table starting from the pair of initial states of both M1 and M2
3. If the transition table contains any pairs of the form (Final, Nonfinal), (NF, F) then stop the construction and say M1 is not equal to M2 ($M1 \neq M2$)
4. If the transition table contains states of the form (F, F) or (NF, NF) then say $M1 = M2$



M2



M3

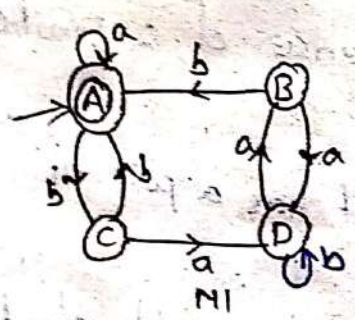
δ	a	b
(A, C)	(A, D)	(B, E)
(A, D)	(A, D)	(B, E)
(B, E)	(A, E)	(

(F, NF)

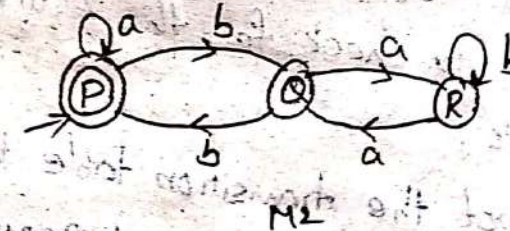
$M1 \neq M2$

Here A is final and E is non final states. So we can stop the construction.

M1 is not equal to M2



M1



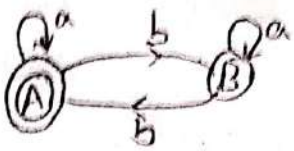
M2

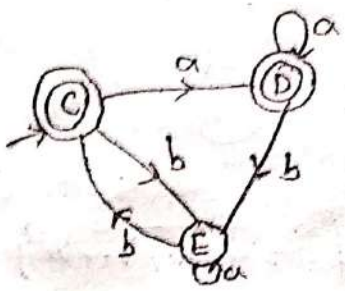
δ	a	b
(A, P)	(A, P)	(C, Q)
(C, Q)	(D, R)	(A, P)
(D, R)	(B, Q)	(D, R)
(B, Q)	(D, R)	(A, P)

There is no further construction. so stop construction

$M1 = M2$

M1 and M2 accepts the same language.





δ	a	b
(A, C)	(A, D)	(B, E)
(A, D)	(A, D)	(B, E)
(B, E)	(B, E)	(A, C)

$M1 = M2$

ϵ-NFA to DFA

Let $E = (Q_E, \Sigma \cup \{\epsilon\}, \delta_E, v_0, F_e)$ be the ϵ-NFA and $D = (Q_D, \Sigma, \delta_D, v_0', F_D)$ be the required DFA

1. $v_0' = \epsilon\text{-closure}(v_0)$
2. For any $S \subseteq Q_E$, $a \in \Sigma$

$$\delta_D(S, a) = \left(\bigcup_{i=1}^{m} \epsilon\text{-closure}(r_i)\right)$$

Let $S = \{P_1, P_2, P_3, \ldots P_k\}$

$$= \bigcup_{i=1}^{k} \delta(P_i, a) = \{r_1, r_2, r_3, \ldots r_m\}$$

$$- \bigcup_{i=1}^{m} \epsilon\text{-closure}(r_i)$$

3. $F_D = F_e \cup \{v_0\}$ if $F_e \cap \epsilon\text{-closure}(v_0) \neq \emptyset$

$F_D = F_e$ if $F_e \cap \epsilon\text{-closure}(v_0) = \emptyset$

$F_D = \{S \in Q_D \mid S \cap F_e \neq \emptyset\}$

ϵ-closure(q_0) = $\{q_0, q_1, q_2\}$

ϵ-closure(q_1) = $\{q_1, q_2\}$

ϵ-closure(q_2) = $\{q_2\}$

$q_0' = \epsilon$-closure(q_0)

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

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

$P_1 \quad P_2 \quad P_3$

$\delta(q_0', 0) = \epsilon$-closure($\delta(q_0, 0) \cup \delta(q_1, 0) \cup \delta(q_2, 0)$)

$= \epsilon$-closure($q_0 \cup \phi \cup \phi$)

$= \epsilon$-closure(q_0)

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

$\delta(q_0', 1) = \epsilon$-closure($\delta(q_0, 1) \cup \delta(q_1, 1) \cup \delta(q_2, 1)$)

$= \epsilon$-closure($\phi \cup q_1 \cup \phi$)

$= \epsilon$-closure(q_1)

$= \{q_1, q_2\}$

$\delta(q_0', 2) = \epsilon$-closure($\delta(q_0, 2) \cup \delta(q_1, 2) \cup \delta(q_2, 2)$)

$= \epsilon$-closure($\phi \cup \phi \cup q_2$)

$= \epsilon$-closure($\{q_2\}$)

$= \{q_2\}$

$S = [v_1, v_2]$

$\delta_D([v_1 v_2], 0) = \epsilon\text{-closure}(\delta_e(v_1, 0) \cup \delta_e(v_2, 0))$

$= \epsilon\text{-closure}(\phi \cup \phi)$

$= \epsilon\text{-closure}(\phi)$

$= \phi$

$\delta_D([v_1 v_2], 1) = \epsilon\text{-closure}(\delta_e(v_1, 1) \cup \delta_e(v_2, 1))$

$= \epsilon\text{-closure}(v_1 \cup \phi)$

$= \epsilon\text{-closure}(v_1)$

$= \{v_1, v_2\}$

$\delta_D([v_1 v_2], 2) = \epsilon\text{-closure}(\delta_e(v_1, 2) \cup \delta_e(v_2, 2))$

$= \epsilon\text{-closure}(\phi \cup v_2)$

$= \epsilon\text{-closure}(v_2)$

$= \{v_2\}$

$S = [v_2]$

$\delta_D(v_2, 0) = \epsilon\text{-closure}(\delta_e(v_2, 0))$

$= \epsilon\text{-closure}(\phi)$

$= \phi$

$\delta_D(v_2, 1) = \epsilon\text{-closure}(\delta_e(v_2, 1))$

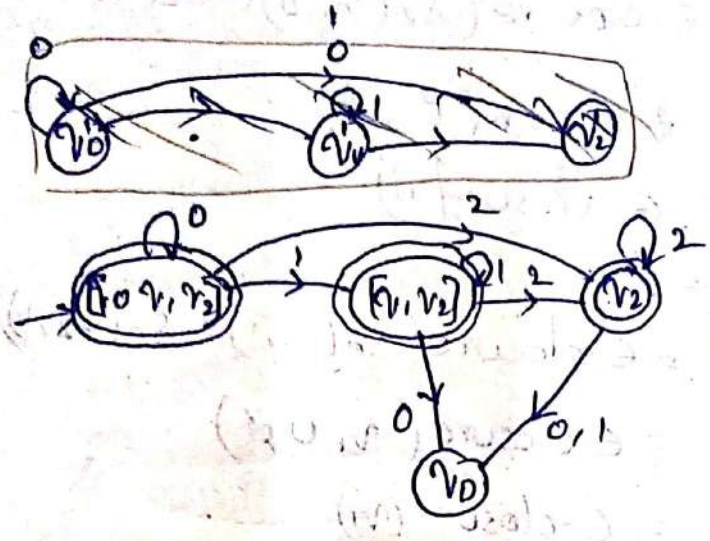
$= \phi$

$\delta_D(v_2, 2) = \epsilon\text{-closure}(\delta_e(v_2, 2))$

$= \epsilon\text{-closure}(v_2)$

$= \{v_2\}$

	0	1	2
$[v_0 v_1 v_2] = v_0'$	$[v_0 v_1 v_2]$	$[v_1 v_2]$	$[v_2]$
$[v_1 v_2] = v_1'$	ϕ	$[v_1 v_2]$	$[v_2]$
$[v_2] = v_2'$	ϕ	ϕ	v_2



$$F_D = \{ S \in Q_D / S \cap F_E \neq \phi \}$$

$$Q_D = \{ S \subseteq Q \in / S = \epsilon\text{-closure}(S) \}$$

$$F_E = \{ q_2 \}$$

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

$$F_D = \{ (q_0 q_1 q_2), (q_1 q_2), (q_2) \}$$

Moore & Mealy Machines:

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

Q = set of states — q_0 = initial state

Σ = i/p alphabet — λ = output function

Δ = o/p alphabet — $\lambda = Q \rightarrow \Delta$

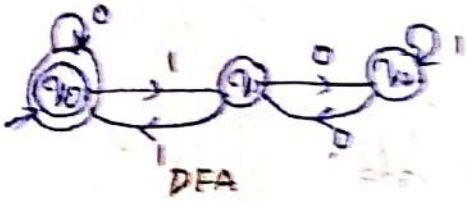
$\delta = Q \times \Sigma \rightarrow Q$

In moor machine o/p is associated with state.

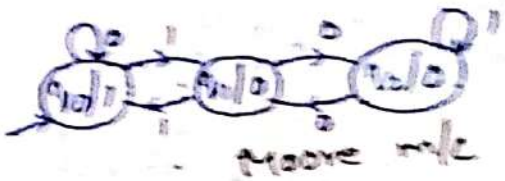
* DFA can be termed as special case of moore machine with $\Delta = \{0, 1\}$

↓ NF ↓ Final

Construct a moore m/c that accepts strings in binary which are multiples of 3



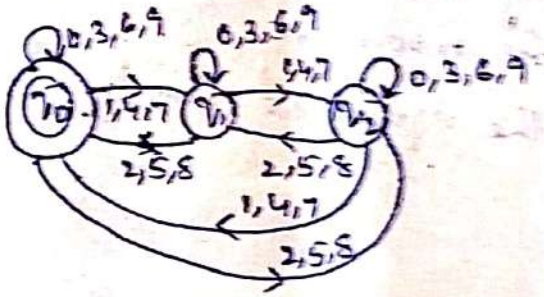
DFA



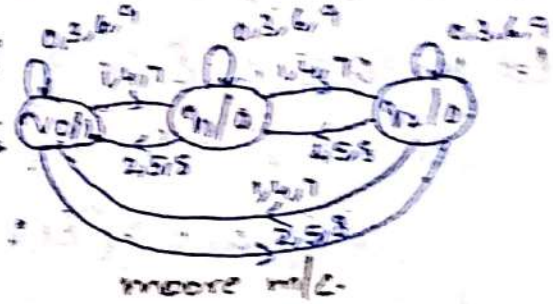
moore m/c

δ	a=0	a=1	Δ
q_0	q_0	q_1	1
q_1	q_2	q_0	0
q_2	q_1	q_2	0

$\Sigma = \{0, 1, 2, \ldots 9\}$ multiples of 3



DFA



moore m/c.

δ	0	1	2	3	4	5	6	7	8	9	Δ
q_0	q_0	q_1	q_2	q_0	q_1	q_2	q_0	q_1	q_2	q_0	1
q_1	q_1	q_2	q_0	q_1	q_2	q_0	q_1	q_2	q_0	q_1	0
q_2	q_2	q_0	q_1	q_2	q_0	q_1	q_2	q_0	q_1	q_2	0

Mealy m/c:

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

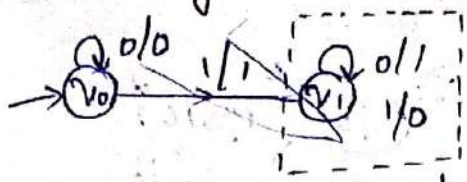
Q = set of all states q_0 = initial state

Σ = i/p alphabet $\lambda : Q \times \Sigma \rightarrow \Delta$

Δ = o/p alphabet output associated with state and current i/p symbol

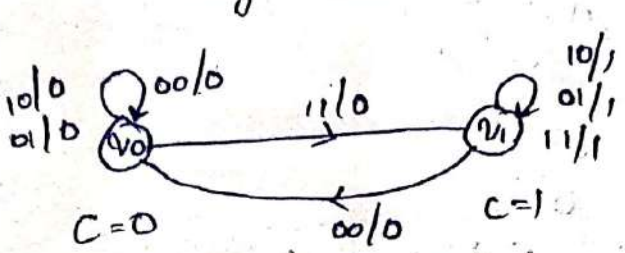
$\delta : Q \times \Sigma \rightarrow Q$

Construct mealy machine that accepts 2's complement of a string in binary.



Compliment

Construct mealy machine for serial adder



a	b	s	c
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Conversion of moore m/c to mealy m/c

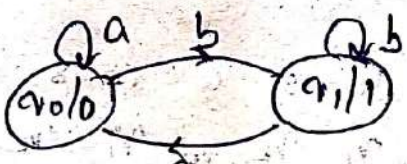
Let $M = (Q, \Sigma, \Delta, \delta, q_0, \lambda)$ be the moore m/c

$M' = (Q, \Sigma, \delta, q_0, \lambda')$ be the equivalent mealy m/c

$$\lambda'(q, a) = \lambda(\delta(q, a))$$

Example

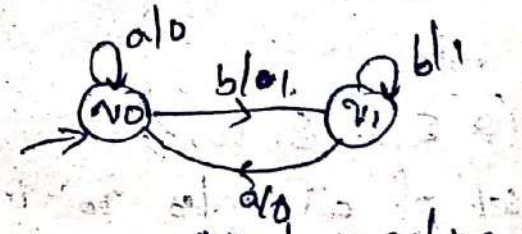
(1)



Moore machine

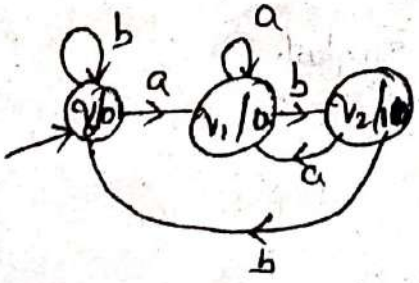
δ	a	b	Δ
$\rightarrow q_0$	q_0	q_1	0
q_1	q_0	q_1	1

δ	a	b
q_0	q_0 0	q_1 1
q_1	q_0 0	q_1 1



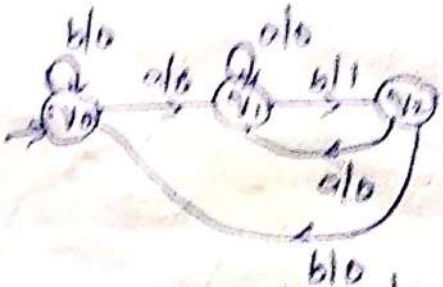
Mealy machine

2. Convert moore m/c to mealy m/c



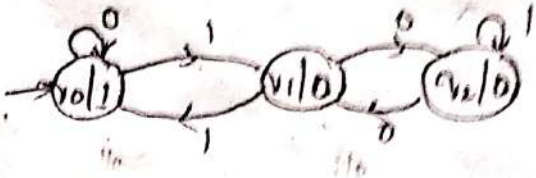
	a	b	Δ
$\rightarrow q_0$	q_1	q_0	0
q_1	q_1	q_2	0
q_2	q_1	q_0	1

	a	b
→ v_0	v_1 0	v_0 0
v_1	v_1 0	v_2 1
v_2	v_1 0	v_0 0



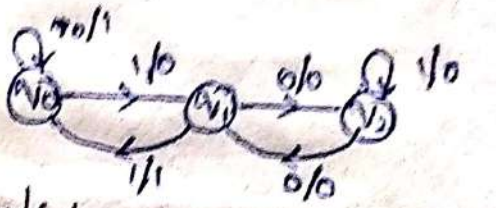
mealy m/c

③



	0	1
v_0	v_0 1	v_1 0
v_1	v_2 0	v_0 1
v_2	v_1 0	v_2 0

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



conversion of mealy to more m/c:

Let $M = (Q, \Sigma, \Delta, \delta, v_0, \lambda)$ be the mealy m/c

$M' = (Q \times \Delta, \Sigma, \Delta, \delta', [v_0, b], \lambda')$

$\delta'([v, b], a) = (\delta(v,a), \lambda(v,a))$

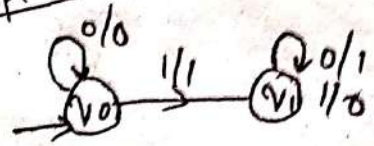
↑ o/p symbol

↳ i/p symbol

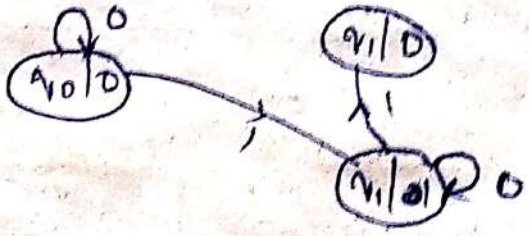
$\lambda'([q, b]) = b$

Example

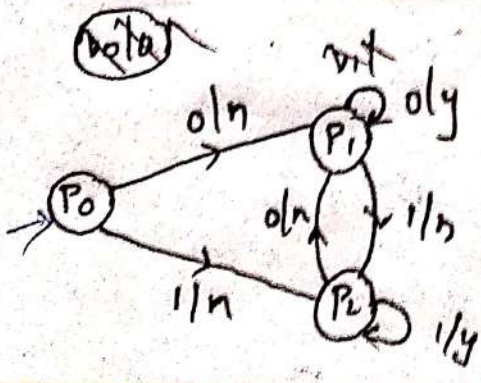
①



	0	1
v_0	v_0 0	v_1 1
v_1	v_1 1	v_1 0



②



	0	1
P_0	P_1 n	P_2 n
P_1	P_1 y	P_2 n
P_2	P_1 n	P_2 y

UNIT-2

Regular Expression

Regular expression:

The expressions that are constructed over i/p alphabet Σ such as $+, \cdot, *$

Priority order
$* > \cdot > +$

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

$a+b,\ a \cdot b,\ a^*(a+ba)^*$

Concatenation of L_1 and L_2

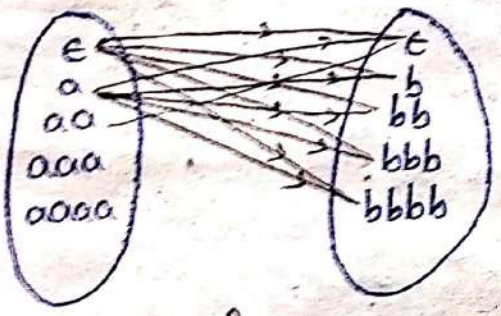
Concatenation of L_1 and $L_2 = L_1L_1 = \{xy / x \in L_1 \text{ \& } y \in L_2\}$

Suppose,

$L_1 = \{\epsilon, a, aa, aaa, \ldots\}$

$L_2 = \{\epsilon, b, bb, bbb, \ldots\}$

$L_1L_2 = \{\epsilon, b, bb, bbb, a, abb, ab, abbb, \ldots\}$



Kleen closure of L:-

$$L^* = \bigcup_{i=0}^{\infty} L^i$$

$L = \{a\}$

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

$L^1 = \{a\}$

$L^2 = \{aa\}$

$L^3 = \{aaa\}$

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

$= \{\epsilon\} \cup \{a\} \cup \{aa\} \cup \ldots$

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

Positive closure of $L : (L^+)$

$$L^+ = \bigcup_{i=1}^{\infty} L^i$$

let $L = \{a\}$

$L^1 = \{a\}$

$L^2 = \{aa\}$

$L^3 = \{aaa\}$

$\vdots$

$L^+ = L^1 \cup L^2 \cup L^3 \cup \ldots$

$= \{a\} \cup \{aa\} \cup \{aaa\} \cup \ldots$

$= \{a, aa, aaa, \ldots\}$

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

$L^1 = \{a, b\}$

$L^2 = \{aa, ab, ba, bb\}$

$L^3 = \{aaa, aba, baa, \ldots\}$

$L^+ = L^1 \cup L^2 \cup L^3 \cup \ldots$

$= \{a, b, aa, ab, ba, \ldots\}$

Regular expression over Σ and sets

(1) ϕ is a regular expression that denotes empty set

(2) ϵ is a regular expression that denotes $\{\epsilon\}$

$r = \epsilon$

(3) $r = a$ $\{a\}$

Precedence of operators

1. $*$
2. $\cdot$
3. $+$

* r is a regular expression
* L(r) → set / language denoted by r
* $L(\emptyset) = \{\} = \emptyset$
* $L(\epsilon) = \{\epsilon\}$

r = ε

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

r = a

Rules:

* L(r+s) represents $L(r) \cup L(s)$
* L(rs) represents $L(r) \cdot L(s)$
* $L(r^*) = [L(r)]^*$

Construct regular expression having over $\Sigma = \{a, b\}$ having aab as substring

( ) aab ( )

any no. of a's & b's

L = any no. of a's and b's

$L = \{\epsilon, a, aa, aaa, b, bb, bbb, \ldots, ab, abb, aab, bab, \ldots\}$

L(r) for any no. of a's and b's

$r = (a+b)^*$

$L(r) = \{\epsilon, (a+b), (a+b)^2, (a+b)^3, \ldots\}$

(1+a)

r = a+b

L(r) = {a, b}

$s = (a+b)^2$

L(r) = {aa, ab, ba, bb}

$r = (a+b)^3$

L(r) = {aaa, aba, baa, bba, baa ...}

$r = (a+b)^*$

L(r) = {ε, a, b, aa, ba, ab, bb, aaa, aba, baa, bba, abb, bba, bab, ...}

Regular expression having aab as substring is

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

set of all string of 0's and 1's with atleast two

( ) 0 ( ) 0 ( )

any no. of 0's and 1's

Regular expression is

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

All binary strings begin with 1 and not having two consecutive zeros.

1 ( )

write Regular expressions for the languages on $\{0,1\}$

a. all strings ending with 01

( ) 01

any number of 0's and 1's

$(0+1)^* 01$

$$(a+b)^* = (b+a)^* = (a^* \cdot b^*)^* = (b^* \cdot a^*)^* = (a^* + b)^* = (a+b^*)^*$$

b. all strings not ending with 01

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

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

c) all strings contains even number of 0's

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

d) all strings having atleast two occurances of substring

$((\,)00(\,)00(\,))^*$

any no. of 0's & 1's

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

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

length of the each string must be the multiple of 3

a/b a/b a/b

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

$L = \{\epsilon, aaa, aba, bab, \ldots\}$

* $L = \{w = na(w) \bmod 3 = 0\}$

$(\,)a(\,)a(\,)a(\,)$

any number of b's

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

$[L(01+1)^*]$

$= L[(0+1)]^*$

$= [L(01) \cup L(1)]^*$

$= [\{01\} \cup \{1\}]^*$

$= \{01, 1\}^*$

$= \{\epsilon, 01, 1, 011, 110 \ldots\}$

$L[(0+1)^* 1 (0+1)]$

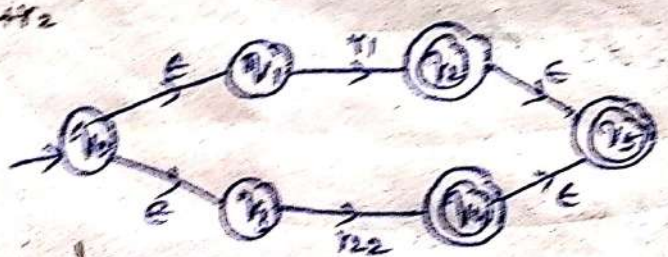
$L(0+1) \overset{F}{L}(1) \cdot L(0+1)$

$L = \{1, 010, 011, 0011, 0010 \ldots\}$

Atleast one 1's is accepted.

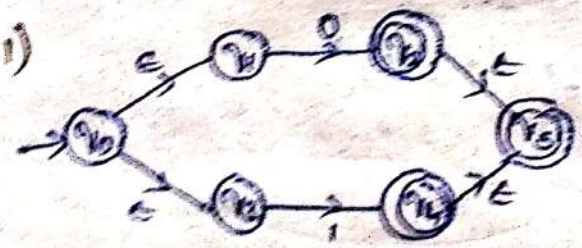
Regular expressions to ϵ-NFA:

$r_1 + r_2$



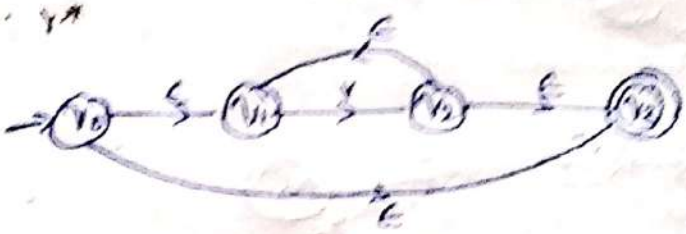
Example

$(0+1)$



2. $r_1 \cdot r_2$

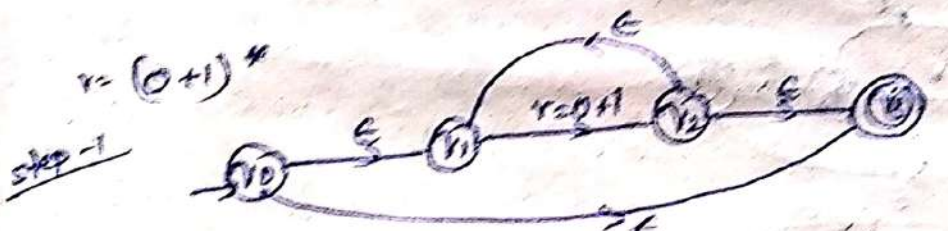
3. r^*



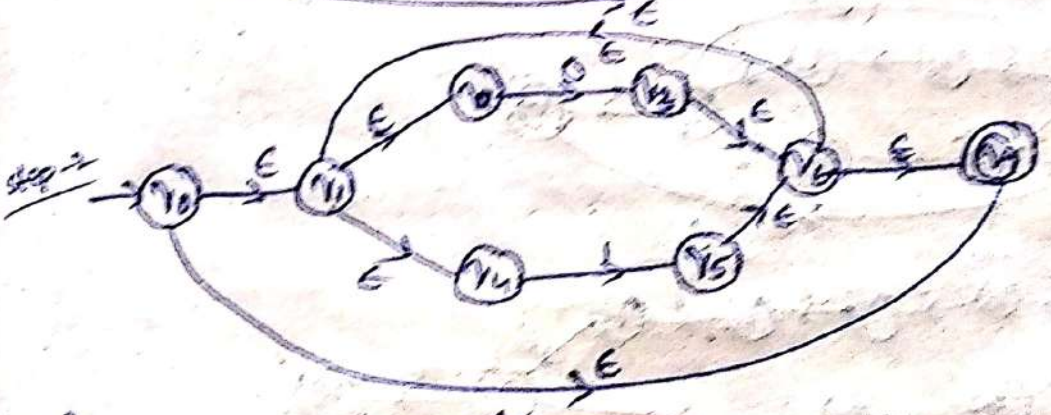
$L((0+1)^* 1 (0+1))$

$L(0+1)^* \; L(1) \; L(0+1)$

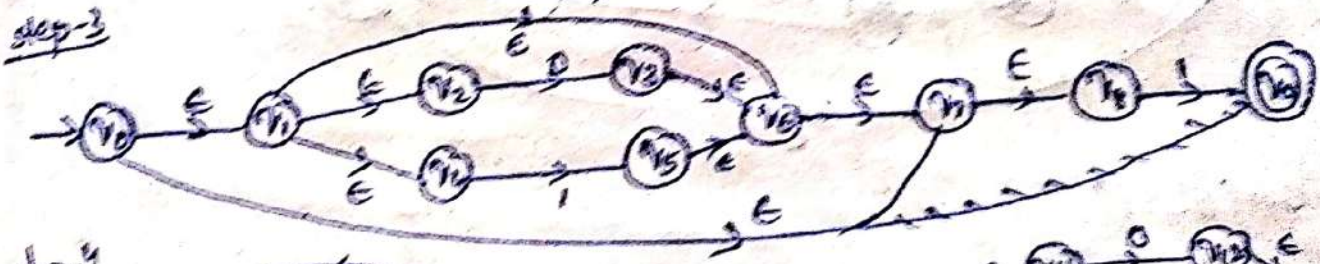
$r = (0+1)^*$



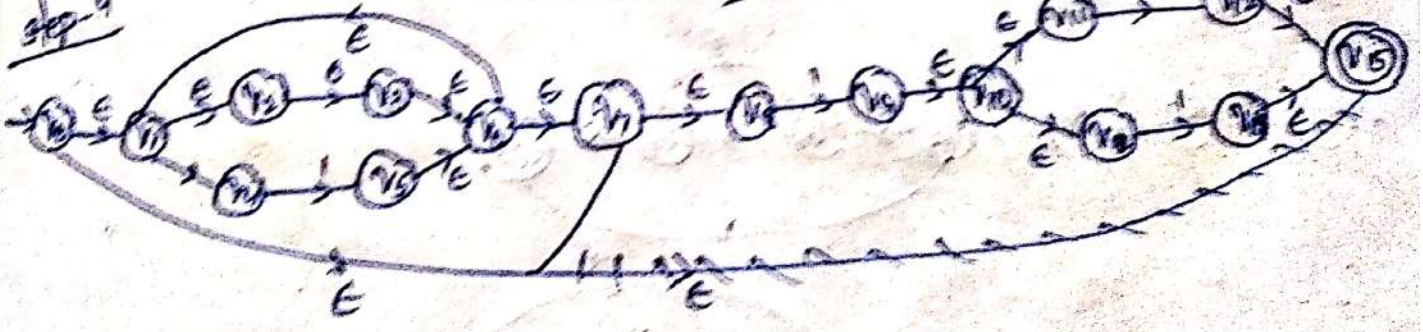
step-1



step-2

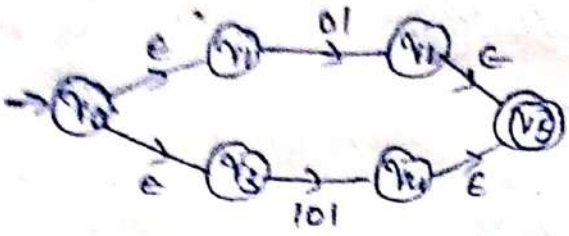


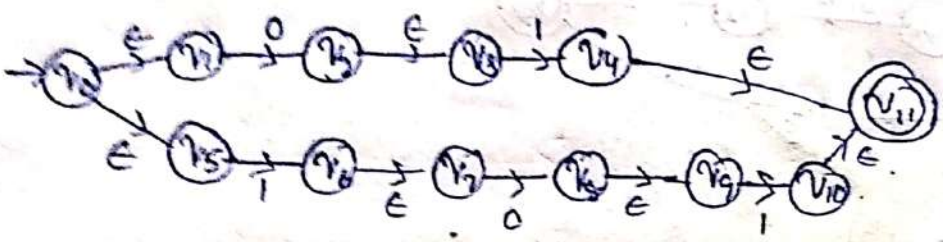
step-3



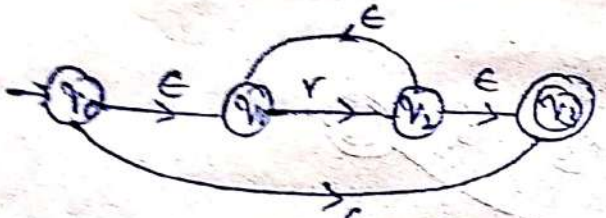
step-4

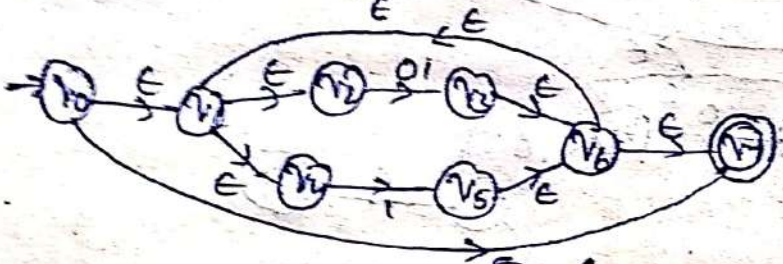
01+101

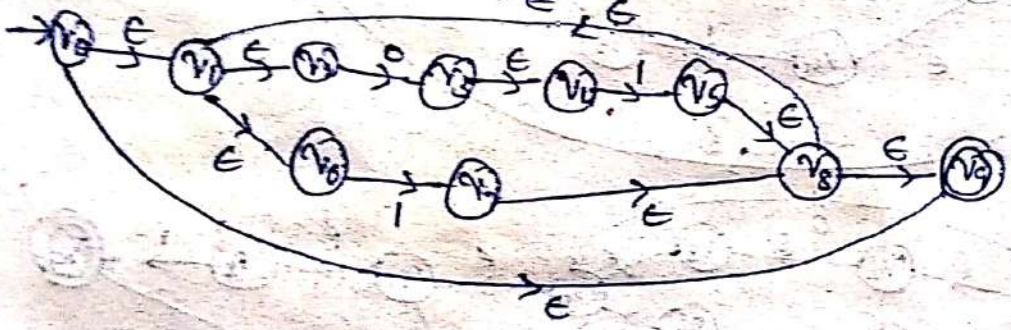




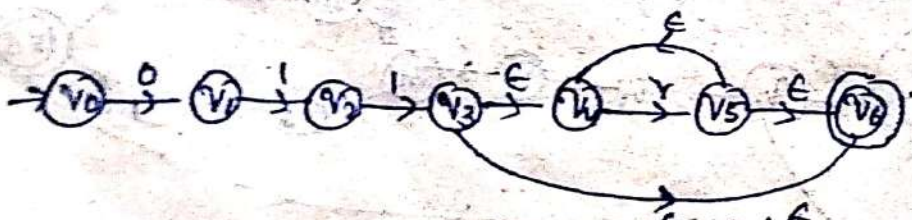
(01+1)*

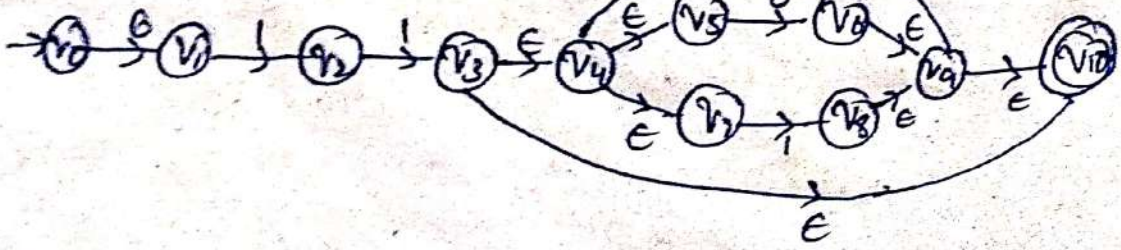






011(0+1)*





closure properties of Regular Languages:

Union: if L_1, L_2 are two Regular languages then $L_1 \cup L_2$ is regular.

$L_1 = \{a^n, n \geq 0\}$

$L_2 = \{b^n, n \geq 0\}$

$L_1 \cup L_2 = \{\epsilon, a, aa, \ldots, b, bb, bbb \ldots\}$

$L_1 \cup L_2 = a^* + b^*$

The language which can be written as regular expression are regular.

intersection:

If L_1, L_2 are two regular languages the $L_1 \cap L_2$ is regular

- $L_1 - M_1 = (Q_1, \Sigma_1, \delta_1, v_0, F_1)$

  $L_2 - M_2 = (Q_2, \Sigma_1, \delta_2, v_0', F_2)$

  $L_1 \cap L_2 - M = (Q_1 \times Q_2, \Sigma, \delta'', v_0 v_0', F_1 \times F_2)$

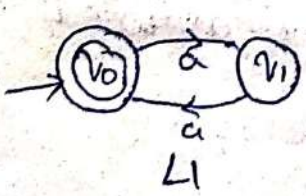
  $\delta''((p, v), a) = \delta_1(p, a) \cup \delta_2(v, a)$

  where $p \in Q_1$ and $v \in Q_2$

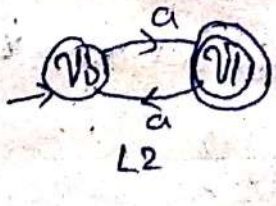
  $a \in \Sigma$

$L_1 = \{a^{2n} / n \geq 0\}$ - M_1

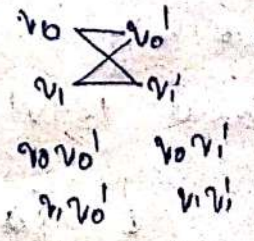
$L_2 = \{a^{2n+1} / n \geq 0\}$ - M_2

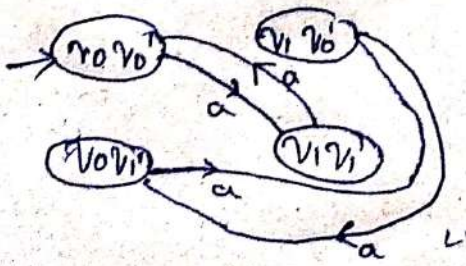


L_1



L_2





$L_1 \cap L_2$

	a
$v_0 v_0'$	$v_1 v_1'$
$v_0 v_1'$	$v_1 v_0'$
$v_1 v_0'$	$v_0 v_1'$
$v_1 v_1'$	$v_0 v_0'$