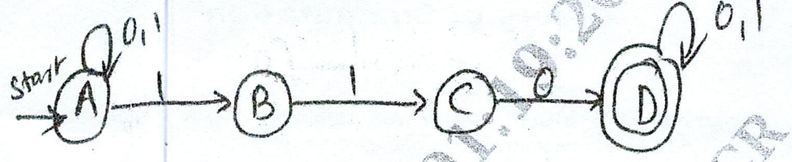
		OR			
Q.4	**a.**	State and prove pumping theorem for regular languages. Show that the language $L = \{a^n b^n \mid n \geq o\}$ is not regular.	**06**	**L3**	**CO2**
	b.	Convert the following FA to RE using state elimination method. 	**06**	**L3**	**CO2**
	c.	Prove that the Regular Languages are closed under : i) Union ii) Complementation iii) Intersection iv) Difference	**08**	**L3**	**CO2**
		Module – 3			
Q.5	**a.**	Design CFG for the following Languages. i) $L = \{ ww^R$ where $W \in \{a, b\}^* \}$ ii) $L = \{ 0^m 1^m 2^n \mid m \geq 1$ and $n \geq 0 \}$ iii) $L = \{ a^n b^{n+3} \mid n \geq 0 \}$	**10**	**L3**	**CO3**
	b.	Consider the grammar G with productions $S \rightarrow aSbS \mid bSaS \mid \in$ Obtain LMD, RMD and parse tree for the string aababb. Is the grammar ambiguous.	**10**	**L3**	**CO3**
		OR			
Q.6	**a.**	Obtain PDA to accept the language $L = \{ wCw^R \mid W \in (a + b)^* \}$ and show the moves made by the PDA for the string aabCbaa.	**10**	**L3**	**CO3**
	b.	Convert the following CFG to PDA $S \rightarrow aABC$ $A \rightarrow aB \mid a$ $B \rightarrow bA \mid b$ $C \rightarrow a$	**10**	**L3**	**CO3**
		Module – 4			
Q.7	**a.**	Define CNF. Convert the following CFG to CNF $E \rightarrow E + T/T$ $T \rightarrow T * F/F$ $F \rightarrow (E) / I$ $I \rightarrow Ia / Ib \mid a \mid b$	**10**	**L3**	**CO4**
	b.	State and prove pumping lemma for context Free Grammars. Show that $L = \{ 0^n 1^n 2^n \mid n \geq 1 \}$ is not content free.	**10**	**L3**	**CO4**
		OR			
Q.8	**a.**	Define CNF convert the following CFG to CNF $S \rightarrow AB$ $A \rightarrow aA \mid bB \mid b$ $B \rightarrow b$	**10**	**L3**	**CO4**
	b.	Prove that the content free languages are closed under i) Union ii) Concatenation iii) Homomorphism	**10**	**L3**	**CO4**
		Module – 5			
Q.9	**a.**	Define a Turing Machine. Explain the working of a basic Turing machine with neat diagram.	**08**	**L2**	**CO5**
	b.	Design a Turing Machine to accept the language $L = \{ a^n b^n c^n \mid n \geq 1 \}$. Draw the transition diagram and show the moves made by TM for the string : aabbcc.	**12**	**L3**	**CO5**
		OR			
Q.10	**a.**	What are the programming Techniques for Turing Machine. Explain.	**10**	**L2**	**CO5**
	b.	Write short notes on : i) Multi Tape Turing Machine ii) Non Deterministic Turing Machine	**10**	**L2**	**CO5**

CBCS SCHEME

USN | | | | | | | | | | |

BCS503

Fifth Semester B.E./B.Tech. Degree Examination, Dec.2025/Jan.2026
Theory of Computation

Time: 3 hrs. Max. Marks: 100

Note: 1. Answer any FIVE full questions, choosing ONE full question from each module.
2. M : Marks , L: Bloom's level , C: Course outcomes.

			M	L	C
		Module – 1			
Q.1	**a.**	Define the following with example. i) Alphabet ii) String iii) Power of an Alphabet iv) Language v) Problem	**05**	**L2**	**CO1**
	b.	Design DFA for the following languages. i) DFA to accept string of 0's, 1's and 2's beginning with a '0' followed by odd number of 1's and ending with a '2'. ii) $L = \{ W \in \{0,1\}^* : w$ does not have 001 as a substring$\}$	**08**	**L3**	**CO1**
	c.	Convert the following NFA to DFA. 	**07**	**L3**	**CO1**
		OR			
Q.2	**a.**	Design an NFA to recognize the following set of strings 0101, 101 and 011.	**05**	**L2**	**CO1**
	b.	Design an DFA to accept binary numbers divisible by 5.	**08**	**L3**	**CO1**
	c.	Obtain DFA for the following $\in$ – NFA δ | $\in$ | a | b | c $\rightarrow p$ | ϕ | {p} | {q} | {r} q | {p} | {q} | {r} | ϕ *r | {q} | {r} | ϕ | {p}	**07**	**L3**	**CO1**
		Module – 2			
Q.3	**a.**	Write the Regular Expression for the following languages. i) $L = \{a^n b^m \mid n >= 4, m <= 3\}$ ii) RE to accept words with two or more letters but beginning and ending with the same letter where $\sum = \{a, b\}$	**06**	**L3**	**CO2**
	b.	Obtain $\in$ – NFA for given Regular Expression $(a + b)^* aa (a + b)^*$	**06**	**L3**	**CO2**
	c.	Find the minimized DFA of the following δ | a | b $\rightarrow A$ | B | F B | G | C *C | A | C D | C | G E | H | F F | C | G G | G | E H | G | C	**08**	**L3**	**CO2**

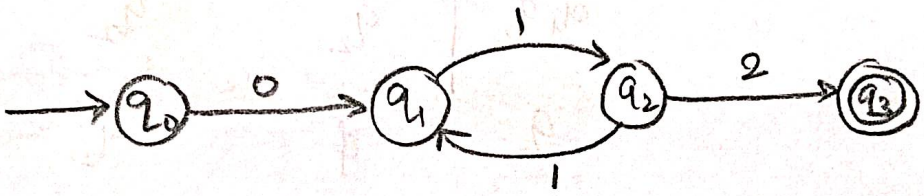
VTU EXAM – DEC 2025/JAN

Q-1. [a] i) Alphabet, ii) string (iii) Power of Alphabet iv) Language v) Problem. 5M

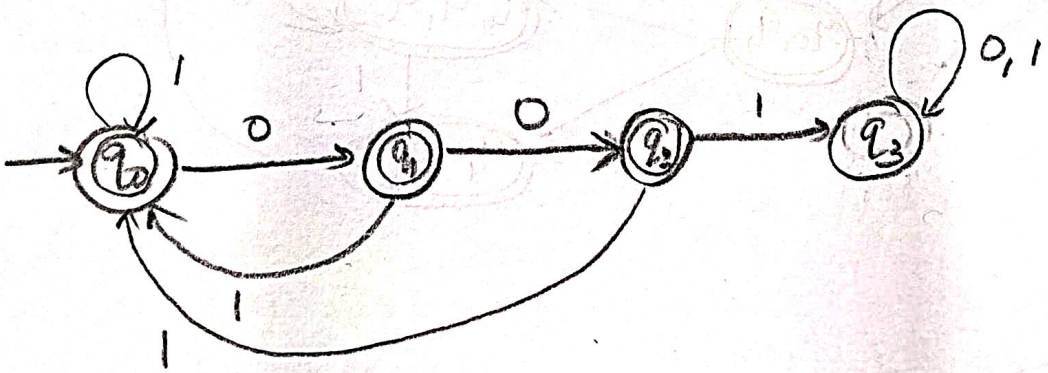
- Definition with example.
- 1 Mark each

Q-1. [b] DFA

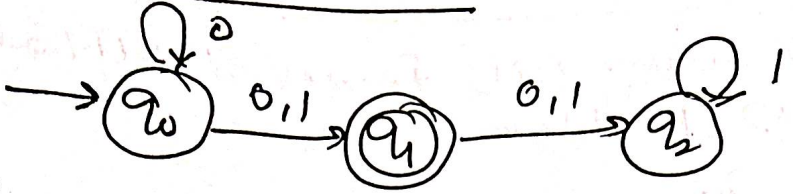
(i) DFA to accept strings of 0's, 1's & 2's beginning with a '0' followed by odd number of 1's and ending with a '2'



(ii) $L = \{w \in \{0,1\}^* ; w$ does not have 001 as a substring



Q-1. [C] NFA to DFA

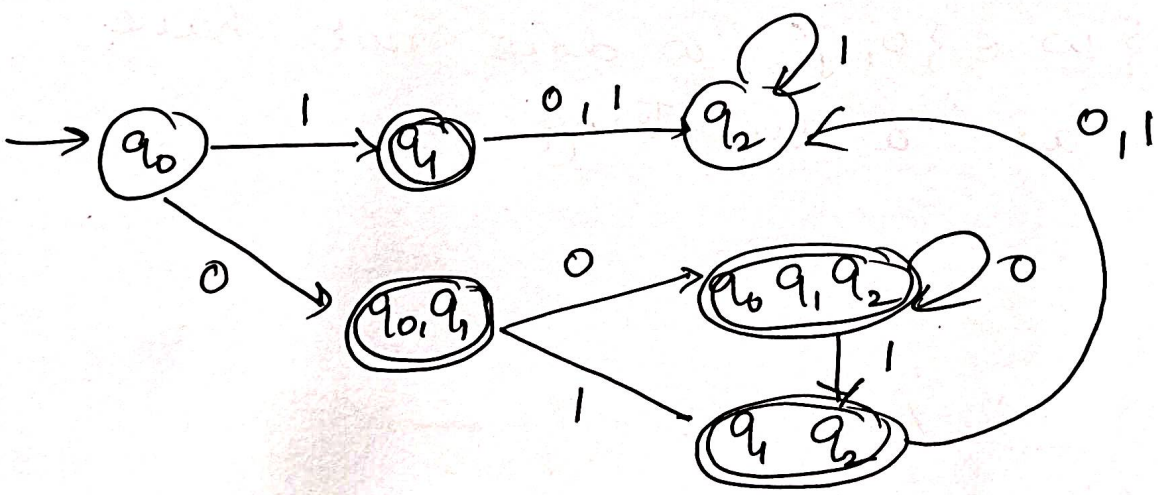


NFD

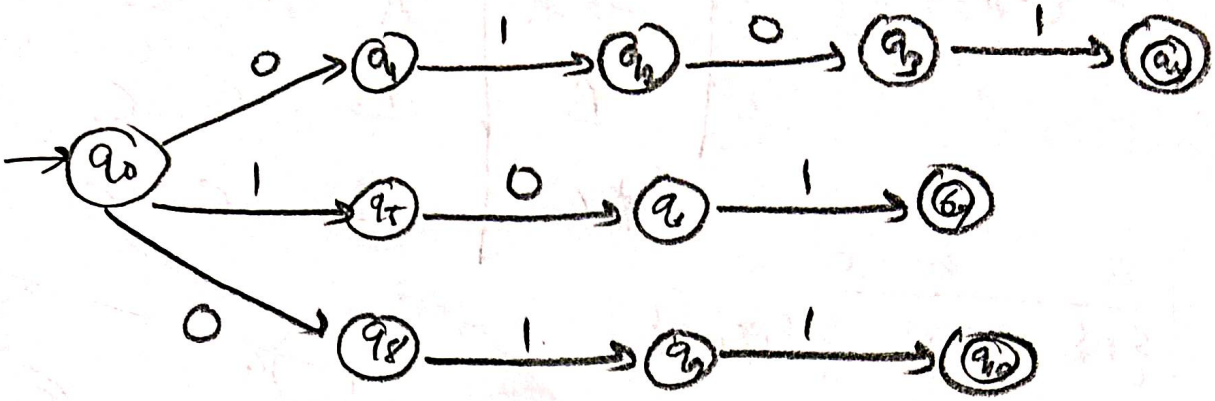
	0	1
q_0	q_0, q_1	q_1
q_1	q_2	q_2
q_2	ϕ	q_2

DFA

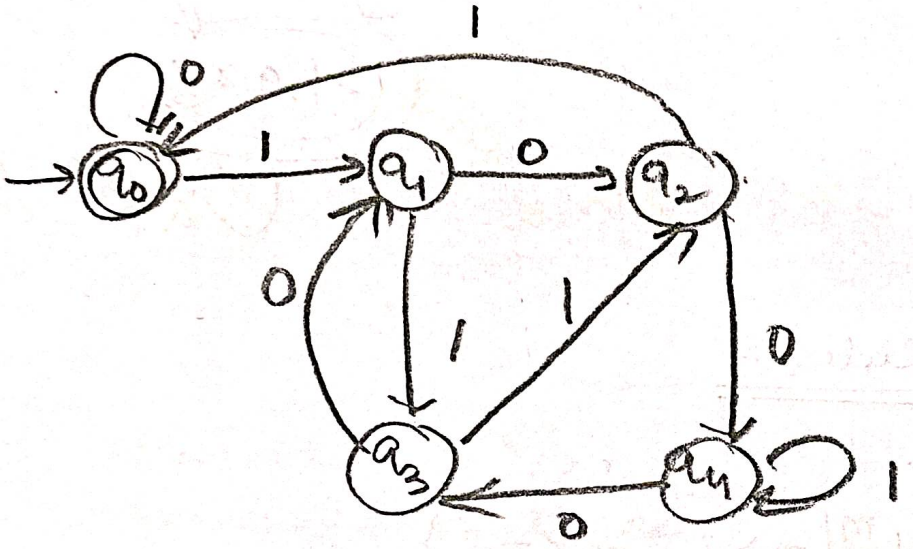
	0	1
q_0	q_0, q_1	q_1
q_0, q_1	$q_0, q_1 q_2$	$q_1 q_2$
q_1	q_2	q_2
$q_0 q_1 q_2$	$q_0, q_1 q_2$	$q_1 q_2$
$q_1 q_2$	q_2	q_2
q_2	ϕ	q_2



[a] NFA to recognize the following set of string 0101, 101 and 011



[b] DFA to accept binary nos. divisible by 5.



Remainder.

0

1

2

3

4

1 % 5 = 1

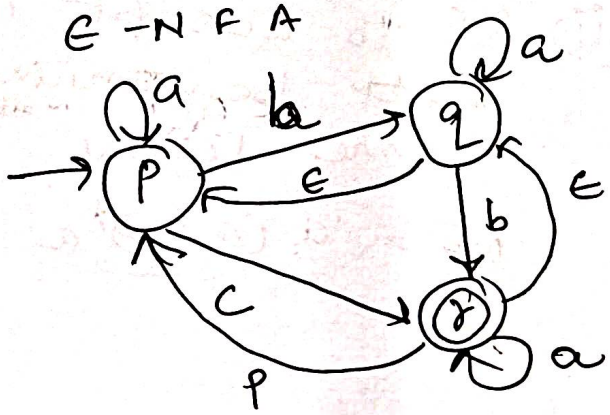
q_1

10 - 2%5 = 2

11 - 3%5 = 3

[c] DFA for following ∈-NFA.

δ	∈	a	b	c
→ p	ϕ	{p}	{q}	{r}
q	{p}	{q}	{r}	ϕ
* r	{q}	{r}	ϕ	{p}



$\epsilon\{P\} = \{P\}$

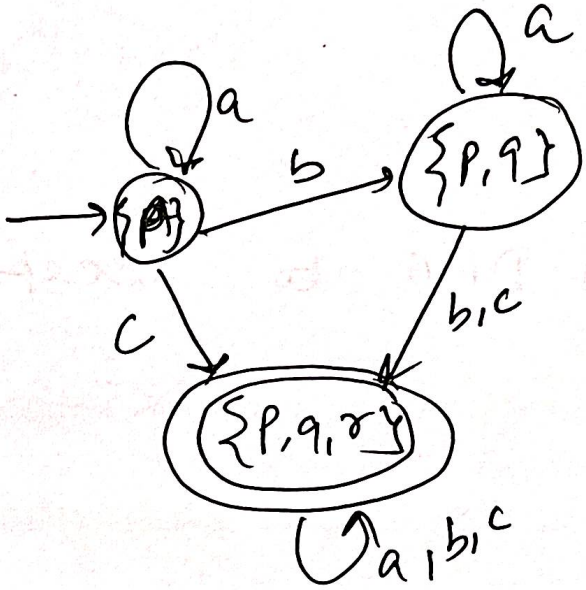
$\epsilon\{q\} = \{q, P\}$

$\epsilon\{r\} = \{r, q, P\}$

DFA

δ	a	b	c
→ A	A	B	C
B	B	C	C
*C	C	C	C

δ	a	b	c
→ {P}	{P}	{P,q}	{P,q,r}
{P,q}	{P,q}	{P,q,r}	{P,q,r}
*{P,q,r}	{P,q,r}	{P,q,r}	{P,q,r}



Module - 2

Q.3[a] (i) RE.

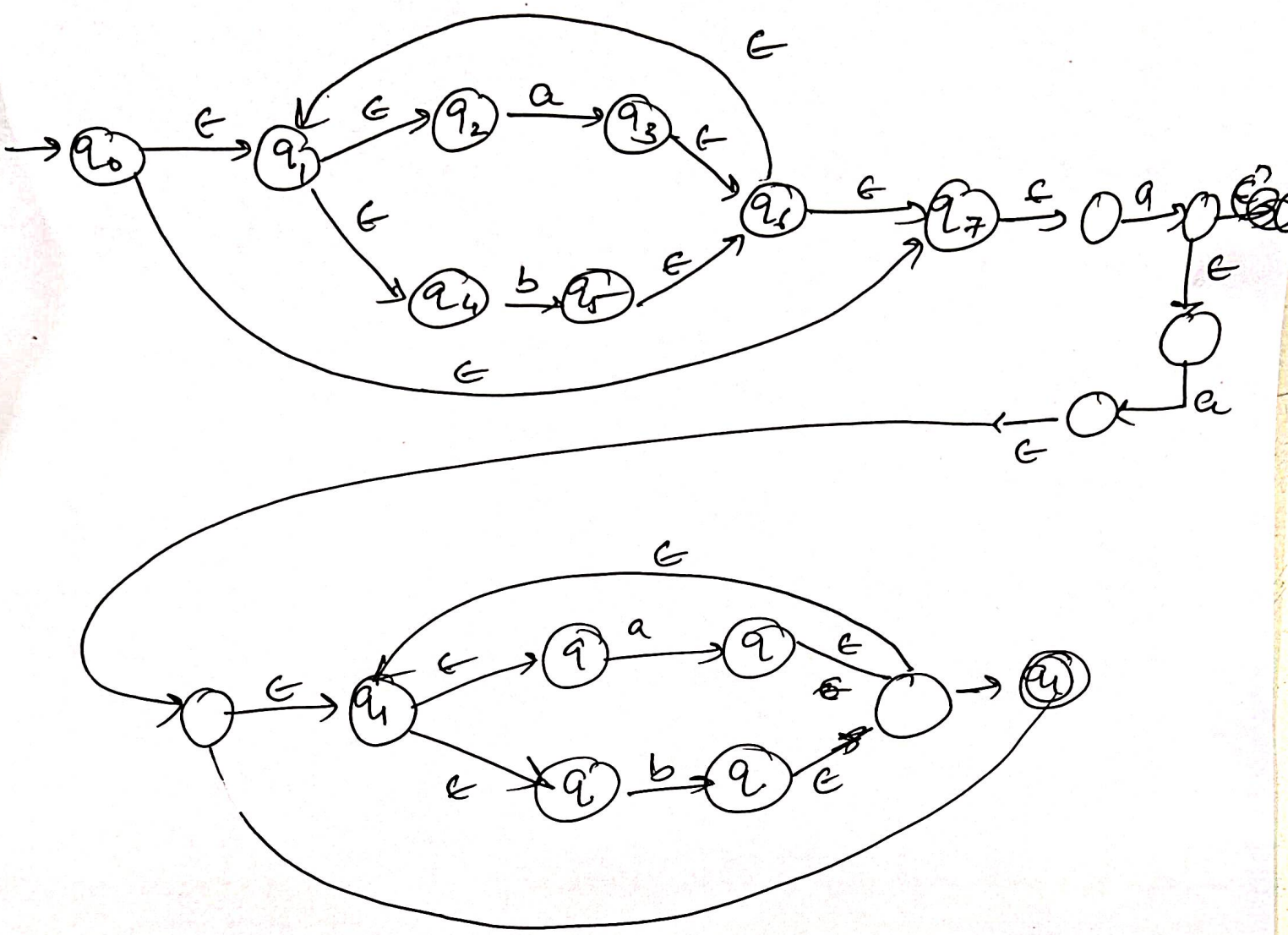
$L = \{a^n b^m \mid n \geq 4, m <= 3\}$

$R.E = aaaa\, a^* (\epsilon + b + bb + bbb)$

(ii) RE to accept words with two or more letters but beginning and ending with same letter where $\Sigma = \{a, b\}$.

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

3.[b] ∈-NFA for R.E. $(a+b)^* aa (a+b)^*$

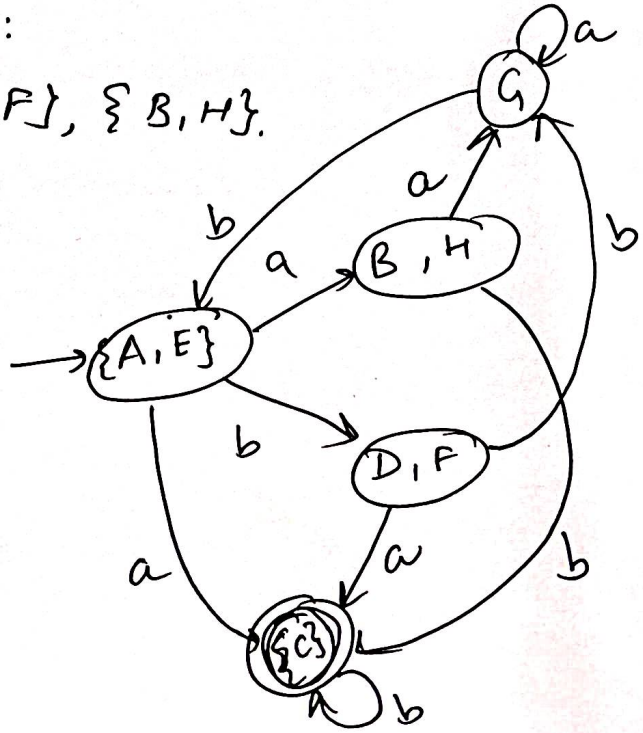


[C] Minimize DFA.

equivalent state are:

{C}, {A, E}, {G}, {D, F}, {B, H}.

	a	b
→ {A, E}	{B, H}	{D, F}
☆ {C}	{A, E}	{C}
{G}	{G}	{A, E}
{D, F}	{C}	{G}
{B, H}	{G}	{C}



State and prove pumping theorem for regular languages. Show that the language $L = \{a^n b^n \mid n \geq o\}$ is not regular.

Pumping Theorem for Regular Languages

Statement

If L is a regular language, then there exists a constant p ≥ 1 (called the pumping length) such that every string w ∈ L with |w| ≥ p can be written as w = xyz satisfying:

1. |y| > 0
2. |xy| ≤ p
3. For all i ≥ 0, xy^i z ∈ L

Proof (Idea)

Since L is regular, there exists a DFA M that accepts L. Let p be the number of states in M. Any string of length at least p must repeat a state while processing the input. The loop formed can be repeated any number of times, so the string can be pumped. Hence, the theorem holds.

Showing that L = {a^n b^n | n ≥ 0} is not regular

Proof using Pumping Theorem

Assume L = {a^n b^n | n ≥ 0} is regular. Then the pumping theorem applies.

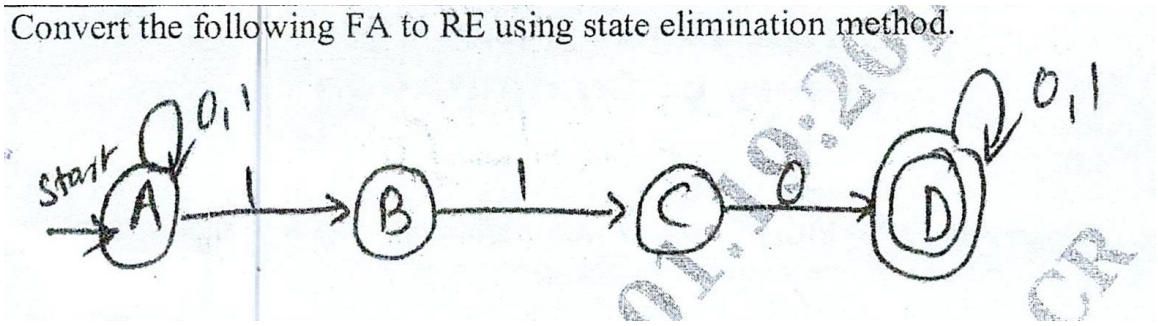
Let p be the pumping length and choose the string w = a^p b^p.

According to the pumping theorem, w = xyz where |xy| ≤ p and |y| > 0. Since the first p symbols are all a's, y consists only of a's.

Pumping with i = 0 gives the string xz = a^(p−k) b^p, where k > 0.

The number of a's and b's are no longer equal, so xz ∉ L. This contradicts the pumping theorem.

Hence, L is not a regular language.



Conversion of FA to Regular Expression

Convert the following Finite Automaton (FA) to a Regular Expression using the state elimination method.

Given Finite Automaton

States: A (Start), B, C, D (Final)
Transitions:
A → A on 0,1
A → B on 1
B → C on 1
C → D on 0
D is the final state

Step 1: Add New Start and Final States

Introduce a new start state S and a new final state F.
Add ε-transitions:
S → A (ε)
D → F (ε)

Step 2: Eliminate State A

State A has a self-loop on (0+1) and a transition to B on 1.
S → B = ε(0+1)*1 = (0+1)*1

Step 3: Eliminate State B

S → C = (0+1)*1 · 1 = (0+1)*11

Step 4: Eliminate State C

S → D = (0+1)*11 · 0 = (0+1)*110

Step 5: Eliminate State D

S → F = (0+1)*110

Final Regular Expression

The regular expression corresponding to the given FA is:

(0+1)*110

Prove that the Regular Languages are closed under :
i) Union ii) Complementation
iii) Intersection iv) Difference

Closure Properties of Regular Languages

Introduction

A class of languages is said to be **closed** under an operation if applying that operation to languages in the class results in a language that also belongs to the same class.

Let L_1 and L_2 be regular languages over an alphabet Σ.

(i) Closure under Union

Claim: $L_1 \cup L_2$ is regular.

Proof:

Since L_1 and L_2 are regular, there exist DFAs M_1 and M_2 that accept them.

Using the **product construction**, a DFA can be constructed whose states are pairs of states from M_1 and M_2.

A state is made final if **either** of its component states is final.

Hence, the constructed DFA accepts $L_1 \cup L_2$.

Therefore, $L_1 \cup L_2$ is regular. □

(ii) Closure under Complementation

Claim: The complement of a regular language is regular.

Proof:

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA accepting L_1.

Construct a new DFA by replacing the set of final states F with $Q - F$.

The resulting DFA accepts $\Sigma^* - L_1$, which is the complement of L_1.

Hence, regular languages are closed under complementation. □

(iii) Closure under Intersection

Claim: $L_1 \cap L_2$ is regular.

Proof:

Using the **product DFA construction**, a DFA is built where a state is final **only if both component states are final**.

This DFA accepts exactly the strings accepted by both L_1 and L_2.

Therefore, $L_1 \cap L_2$ is regular. □

(iv) Closure under Difference

Claim: $L_1 - L_2$ is regular.

Proof:

$$L_1 - L_2 = L_1 \cap (\Sigma^* - L_2)$$

Since regular languages are closed under **complementation** and **intersection**,
$L_1 - L_2$ is also regular. □

Design CFG for the following Languages.

i) $L = \{ ww^R \text{ where } W \in \{a, b\}^* \}$

ii) $L = \{ 0^m 1^m 2^n \mid m \geq 1 \text{ and } n \geq 0 \}$

iii) $L = \{ a^n b^{n+3} \mid n \geq 0 \}$

(i) $L = \{ww^R \mid w \in \{a,b\}^*\}$

Idea

The language contains **even-length palindromes** over `{a, b}`.

CFG

Let the grammar be $G = (\{S\}, \{a,b\}, P, S)$, where the productions are:

$$\boxed{S \to aSa \mid bSb \mid \varepsilon}$$

(ii) $L = \{0^m 1^m 2^n \mid m \geq 1,\ n \geq 0\}$

Idea

- Equal number of `0`s and `1`s (at least one)
- Any number of `2`s at the end

CFG

Let $G = (\{S, A\}, \{0,1,2\}, P, S)$, where:

$$\boxed{\begin{aligned} S &\to 0S1 \mid 01A \\ A &\to 2A \mid \varepsilon \end{aligned}}$$

(iii) $L = \{a^n b^{n+3} \mid n \geq 0\}$

Idea

- Number of `b`s is always **three more** than number of `a`s

CFG

Let $G = (\{S\}, \{a, b\}, P, S)$, where:

$$\boxed{S \to aSb \mid bbb}$$

Consider the grammar G with productions S → aSbS | bSaS | ∈
Obtain LMD, RMD and parse tree for the string aababb. Is the grammar ambiguous.

1 Leftmost Derivation (LMD)

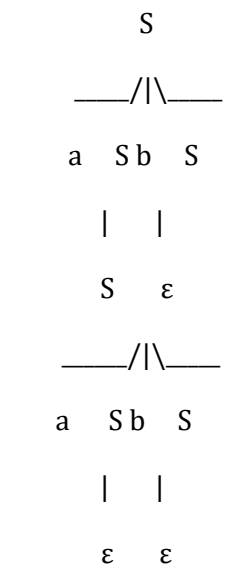
In **LMD**, always expand the **leftmost non-terminal**.

$$\begin{aligned} S &\Rightarrow aSbS \\ &\Rightarrow a\,aSbS\,bS \\ &\Rightarrow a\,a\,\varepsilon\,bS\,bS \\ &\Rightarrow aab\,bSaS \\ &\Rightarrow aabb\,aSbS \\ &\Rightarrow aabba\,\varepsilon\,b\,\varepsilon \\ &\Rightarrow \boxed{aababb} \end{aligned}$$

2 Rightmost Derivation (RMD)

In **RMD**, always expand the **rightmost non-terminal**.

$$\begin{aligned} S &\Rightarrow aSbS \\ &\Rightarrow aSb\,\varepsilon \\ &\Rightarrow a\,aSbS\,b \\ &\Rightarrow a\,aSb\,\varepsilon\,b \\ &\Rightarrow aabSaSb \\ &\Rightarrow aab\,\varepsilon\,aSb \\ &\Rightarrow \boxed{aababb} \end{aligned}$$



Definition

A grammar is **ambiguous** if **a string has more than one distinct parse tree** (or LMD/RMD).

Observation

- The string `aababb` can be derived using **different structural groupings**
- Multiple valid parse trees exist for the same string

The grammar is ambiguous

Obtain PDA to accept the language
$L = \{ wCw^R \mid W \in (a + b)^* \}$ and show the moves made by the PDA for the string aabCbaa.

PDA Definition

Let the PDA be

$$M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$$

Components

- $Q = \{q_0, q_1, q_f\}$
- $\Sigma = \{a, b, C\}$
- $\Gamma = \{a, b, Z_0\}$
- Start state: q_0
- Start stack symbol: Z_0
- Final state: q_f

1 Push phase (before `C`)

$$\delta(q_0, a, X) = (q_0, aX)$$

$$\delta(q_0, b, X) = (q_0, bX)$$

2 Read middle symbol `C`

$$\delta(q_0, C, X) = (q_1, X)$$

3 Pop & match phase

$$\delta(q_1, a, a) = (q_1, \varepsilon)$$

$$\delta(q_1, b, b) = (q_1, \varepsilon)$$

4 Accept

$$\delta(q_1, \varepsilon, Z_0) = (q_f, Z_0)$$

Moves of the PDA for `aabCbaa`

Step	Input	Stack	State
1	aabCbaa	Z_0	q_0
2	abCbaa	aZ_0	q_0
3	bCbaa	baZ_0	q_0
4	Cbaa	$bbaZ_0$	q_0
5	baa	$bbaZ_0$	q_1
6	aa	baZ_0	q_1
7	a	aZ_0	q_1
8	ε	Z_0	q_1
9	ε	Z_0	qf

Convert the following CFG to PDA

S → aABC

A→ aB|a

B → bA|b

C → a

PDA Definition

$$M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$$

Components

- $Q = \{q_0, q_1, q_f\}$
- $\Sigma = \{a, b\}$
- $\Gamma = \{a, b, S, A, B, C, Z_0\}$
- Start state: q_0
- Start stack symbol: Z_0
- Final state: q_f

1 Start

$$\delta(q_0, \varepsilon, Z_0) = (q_1, SZ_0)$$

---

2 Variable Expansions (Productions)

$$\delta(q_1, \varepsilon, S) = (q_1, aABC)$$

$$\delta(q_1, \varepsilon, A) = (q_1, aB) \quad | \quad (q_1, a)$$

$$\delta(q_1, \varepsilon, B) = (q_1, bA) \quad | \quad (q_1, b)$$

$$\delta(q_1, \varepsilon, C) = (q_1, a)$$

---

3 Terminal Matching

$$\delta(q_1, a, a) = (q_1, \varepsilon)$$

$$\delta(q_1, b, b) = (q_1, \varepsilon)$$

---

4 Accept

$$\delta(q_1, \varepsilon, Z_0) = (q_f, Z_0)$$

TOC Solutions to these questions

Q.7	a.	Define CNF. Convert the following CFG to CNF $E \rightarrow E + T/T$, $T \rightarrow T * F/F$ $F \rightarrow (E) / I$ $I \rightarrow Ia / Ib / a / b$	10	L3	CO4
	b.	State and prove pumping lemma for context Free Grammars. Show that $L = \{ 0^n 1^n 2^n \mid n \geq 1 \}$ is not content free.	10	L3	CO4
		OR			
Q.8	a.	Define CNF convert the following CFG to CNF $S \rightarrow AB$ $A \rightarrow aA \mid bB \mid b$ $B \rightarrow b$	10	L3	CO4
	b.	Prove that the content free languages are closed under i) Union ii) Concatenation iii) Homomorphism	10	L3	CO4
		Module – 5			
Q.9	a.	Define a Turing Machine. Explain the working of a basic Turing machine with neat diagram.	08	L2	CO5
	b.	Design a Turing Machine to accept the language $L = \{ a^n b^n c^n \mid n \geq 1 \}$. Draw the transition diagram and show the moves made by TM for the string : aabbcc.	12	L3	CO5
		OR			
Q.10	a.	What are the programming Techniques for Turing Machine. Explain.	10	L2	CO5
	b.	Write short notes on : i) Multi Tape Turing Machine ii) Non Deterministic Turing Machine	10	L2	CO5

7.a Convert CFG to CNF

The Grammar

E→E+T
E→T
T→T*F
T→F
F→(E)
F→x

Step 1 Assign variables to terminals

A→ +
B→ *
C→(
D→)
F→x

Step 2

Remove epsilon which in this grammar is not available

Step 3

Remove useless symbols which there are none

Step 4 Remove unit rule

S→E
E→T
T→F

So we have

E→T*F|E+T
T→(E)|x
F→(E)|x

Step 5 Add a start symbol

S→T*F|E+T
E→T*F|E+T
T→(E)|x

F→(E)|x
A→ +
B→ *
C→(
D→)
F→x

Step 6 Convert in the form A→BC and A→a Until now we have

S→T*F|E+T
E→T*F|E+T
T→(E)|x
F→(E)|x
A→ +
B→ *
C→(
D→)
F→x

Using terminal variables we get

S=EAT|TBF
T→CED|x
E→EAT|TBF
F→CED|x
A→ +
B→ *
C→(
D→)
Let
A_1=AT
B_1=BF
E_1=ED

So final CNF is

S=EA_1 |TB_1
T→CE_1 |x
E→EA_1 |TB_1
F→CE_1 | x
A→ +
B→ *
C→(
D→)
F→x
A_1=AT
B_1=BF
E_1=ED

7. b. pumping lemma

Prove that:

L = { 0n1n2n | n ≥ 1}is not context-free.

Proof: Assume L012 = {0n1n2n | n ? 0} is context-free.
By the CFL Pumping Lemma, there exists p such that any string s ∈ L with | s| ≥ p can be written as s=uvwxy satisfying:
| vwx| ≤p ,

| vx| ≥1

and uviwxiy ∈ L for all i ≥ 0.

Take s=0p1p2p .
Since | vwx| ≤p, vwx cannot include both 0's and 2's. Thus it lies within one block or at most spans two adjacent blocks.

Pumping i=0 or i=2 changes the number of symbols in only one or two blocks, while the third block remains unchanged. Hence the numbers of 0's, 1's, and 2's are no longer equal, so the pumped string is not in L.

This contradicts the Pumping Lemma.
Therefore, L012 = {0n1n2n | n >=1} is **not** context-free.

8. a . cfg to CNF

- $S \rightarrow AB$
- $A \rightarrow aA|bB|b$
- $B \rightarrow b$

2. Introduce New Variables for Terminals:
We need rules like $A \rightarrow a$ and $A \rightarrow b$.

- Let $X \rightarrow a$
- Let $Y \rightarrow b$

Now the grammar is:

- $S \rightarrow AB$
- $A \rightarrow XA|YB|Y$
- $B \rightarrow Y$
- $X \rightarrow a$
- $Y \rightarrow b$

3. Convert Mixed Productions to Binary (A → BC) or Terminal (A → a):

- **A → XA**: This is already in CNF form (A → BC, where X, A are non-terminals).
- **A → YB**: This is already in CNF form (A → BC).
- **A → Y**: This is a terminal production (A → a).
- **B → Y**: This is a terminal production (A → a).
- **S → AB**: This is already in CNF form (A → BC).

4. Final CNF Grammar:
All rules are now in the form $A \rightarrow BC$ or $A \rightarrow a$.

4. Final CNF Grammar:
All rules are now in the form $A \rightarrow BC$ or $A \rightarrow a$.

- $S \rightarrow AB$
- $A \rightarrow XA|YB|Y$
- $B \rightarrow Y$
- $X \rightarrow a$
- $Y \rightarrow b$

This grammar is now in Chomsky Normal Form (CNF).

8.b. union, concatenation homomorphism

Property 1 Each of the classes $\mathscr{L}_0$, $\mathscr{L}_{cs1}$, $\mathscr{L}_{cfl}$, $\mathscr{L}_{rl}$ is closed under union, concatenation, closure and transpose operations (Theorems 4.5–4.7).

Property 2 $\mathscr{L}_{rl}$ is closed under intersection and complementation (Theorems 5.7 and 5.8).

Property 3 $\mathscr{L}_{cfl}$ is not closed under intersection and complementation.

We establish property 3 by a counter-example. We have already seen that $L_1 = \{a^n b^n c^i \mid n \geq 1,\ i \geq 0\}$ and $L_2 = \{a^i b^n c^n \mid n \geq 1,\ j \geq 0\}$ are context-free languages (Examples 4.8 and 4.9). $L_1 \cap L_2 = \{a^n b^n c^n \mid n \geq 1\}$. In Example 6.18, we have shown that $\{a^n b^n c^n \mid n \geq 1\}$ is not context-free. Thus, $\mathscr{L}_{cfl}$ is not closed under intersection.

Using DeMorgan's law, we can write $L_1 \cap L_2 = (L_1^c \cup L_2^c)^c$. We have proved in Chapter 4 that $\mathscr{L}_{cfl}$ is closed under union. If $\mathscr{L}_{cfl}$ were closed under complementation, then $L_1 \cap L_2$ turns out to be context-free which is not true. Hence, $\mathscr{L}_{cfl}$ is not closed under complementation.

9.

9.a **Turing Machine Model**

Each cell can store only one symbol. The input to and the output from the finite state automaton are effected by the R/W head, which can examine one cell at a time. In one move, the machine examines:

the present symbol under the R/W head, and

the present state of the automatonto determine:

A new symbol to be written on the tape in the cell under the R/W head.

A motion of the R/W head along the tape: either one cell left (L) or one cell right (R).

The next state of the automaton.

Whether to halt or continue.

Turing machine M is a 7-tuple**:**

(Q,Σ,Γ,δ,q0,b,F)(Q, \Sigma, \Gamma, \delta, q_0, b, F)(Q,Σ,Γ,δ,q0,b,F)

where:

Q is a finite, non-empty set of states.

Γ is a finite, non-empty set of tape symbols.

b ∈ Γ is the blank symbol.

Σ is a non-empty set of input symbols, where Σ⊆Γ\Sigma \subseteq \GammaΣ⊆Γ and b∉Σb \notin \Sigmab∈/Σ.

δ is the transition function mapping

(q,x)↦(q′,y,D)(q, x) \mapsto (q', y, D)(q,x)↦(q′,y,D)

where **D** denotes direction of R/W head movement (L or R).

q_0 ∈ Q is the initial state.

F ⊆ Q is the set of final (accepting) states.

A string is accepted if there is a transition path from the initial state to a final state. Hence, final states are also called accepting states.

The transition function δ may not be defined for all elements of Q×ΓQ \times \GammaQ×Γ.

Representation of Turing Machines

Turing machines can be described using:

Instantaneous descriptions (IDs)

Transition tables

Transition diagrams (graphs)

Representation by Instantaneous Descriptions

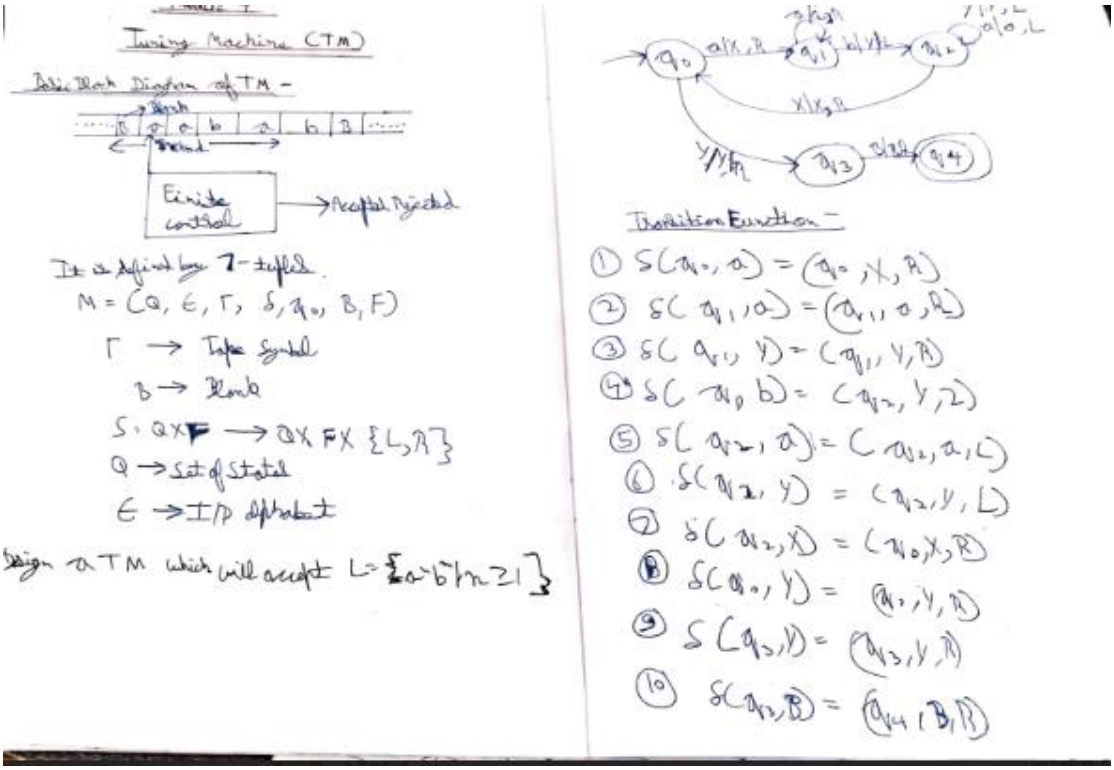
A *snapshot* of a Turing machine in action is called an instantaneous description (ID).
It records:

the current state

the entire tape contents

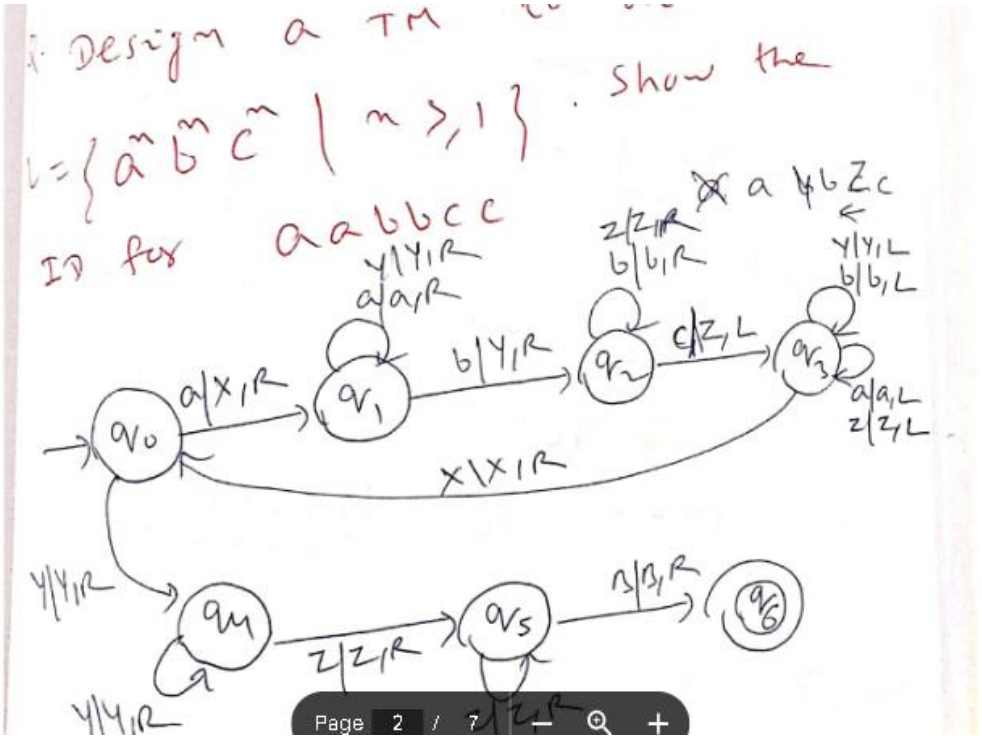
the position of the R/W head

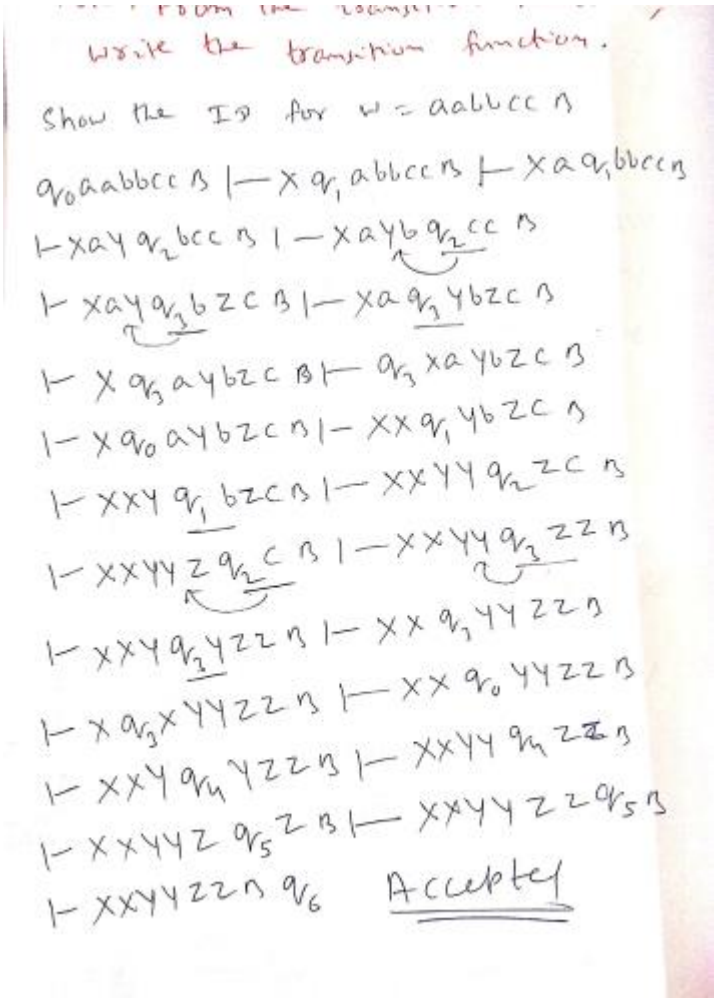
Unlike PDA (pushdown automata), where only unread input and stack matter, in a Turing machine the head may move left, so the entire tape must be considered.



9. b turing machine for language

10.





10.a techniques of TM

10. **B I)Multi-tape Turing Machine**

The multitape Turing machine is a type of Turing machine in which there are more than one input tapes. Each tape is divided into cells and each cell can hold any symbol of finite tape alphabet. The multitape Turing machine is as shown in the Fig. 7.1.1.

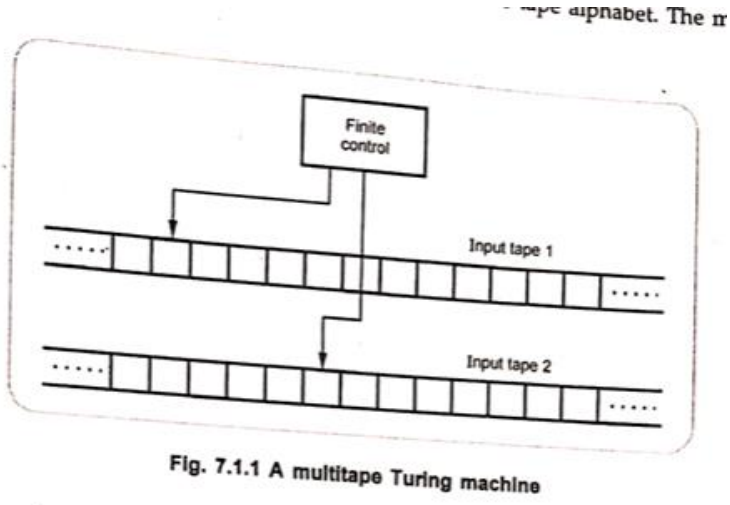


Fig. 7.1.1 A multitape Turing machine

Fig. 7.1.1 A multitape Turing machine

This TM is more powerful than the basic Turing machine, because finite control reads more than one input tape and more symbols can be scanned at a time.

ii) **Non Deterministic Turing Machine**

The non deterministic Turing machine is a kind of Turing machine in which the set of rules denote more than one specific action on reading a particular input in the current specific state.

This kind of TM is just similar to NFA.

For example Note that any language accepted by a non deterministic Turing machine can also be accepted by a deterministic Turing machine.

The power of a non deterministic Turing machine is just similar to the power of a deterministic Turing machine.