3|3|13

UNIT-5

TURING MACHINE

Turing machine contains 7 tuples $(Q, \Sigma, \Gamma, \delta, q_0, F, B)$

where Q = Set of States

Σ = Set of input symbols

Γ = Set of input tape symbols

δ = Transition function

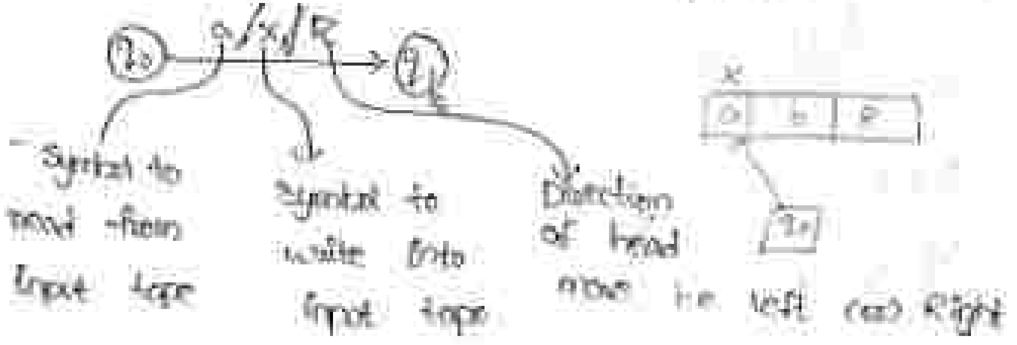
q_0 = Initial state

F = final State

B = Blank symbol.

where δ can be defined as $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times (L/R)$

The transition diagram:- The transition function can be represented in the form of graphical notation.

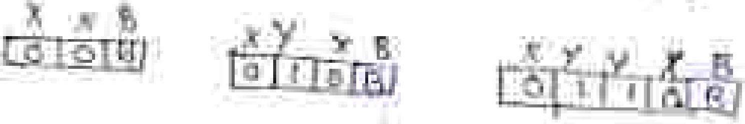


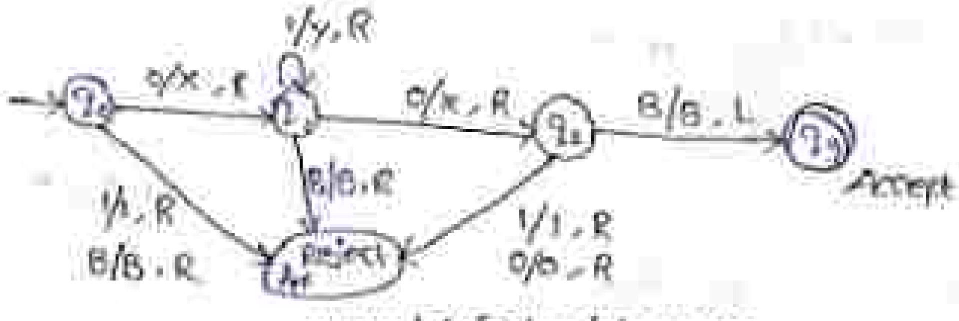
q_0
a/x/R
q_1
Symbol to read from input tape
Symbol to write into input tape
Direction of head move i.e. left (or) Right
x
a
b
c
q_0

1) Design a turing machine for $L = 01^*0$

$L = 01^*0$

$L = \{00, 010, 0110, 01110, \cdots\}$

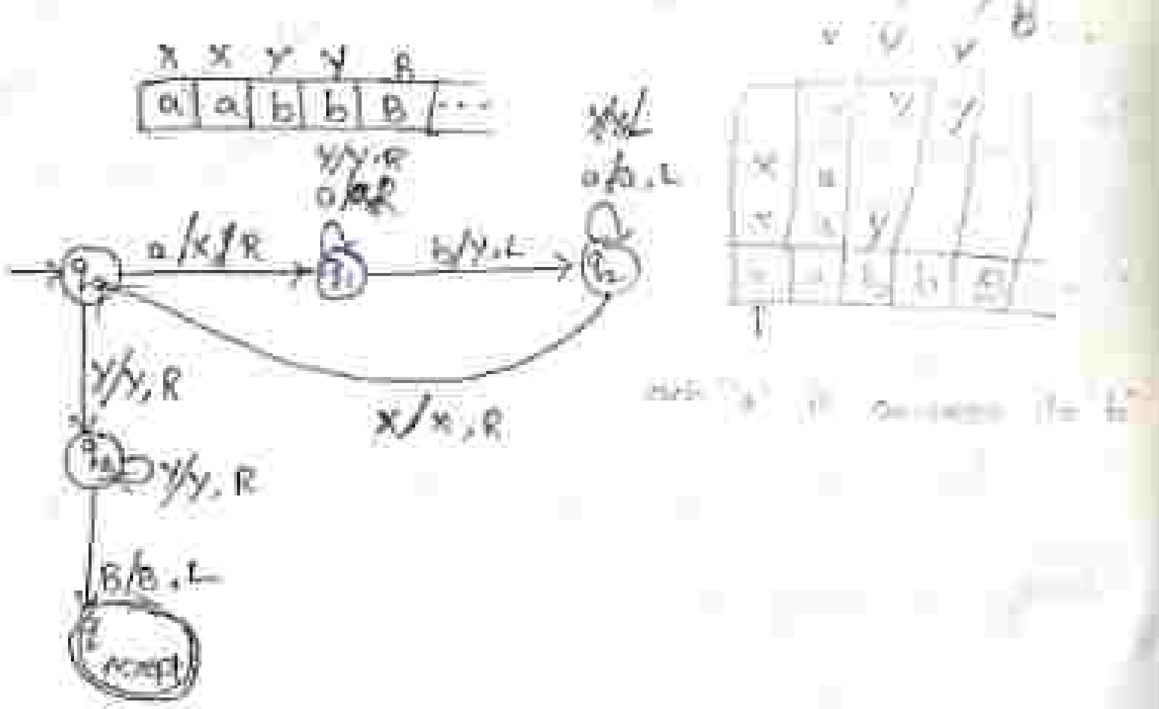




Tape Symbol / State	0	1	x	y	B
q_0	$<q_1, x, R>$	$<q_4, 1, R>$	-	-	$<q_4, B, R>$
q_1	$<q_2, x, R>$	$<q_1, y, R>$	-	-	$<q_4, B, R>$
q_2	$<q_4, 0, R>$	$<q_4, 1, R>$	-	-	$<q_3, B, L>$
q_3	-	-	-	-	-
q_4	-	-	-	-	-

5) Design a turing machine for language $L=\{a^nb^n / n\geq 1\}$

$L=\{ab, aabb, aaabbb, \cdots\}$



Tape Symbol / State	a	b	x	y	B
$\rightarrow q_0$	$<q_1, x, R>$	—	—	$<q_3, y, R>$	—
q_1	$<q_1, a, R>$	$<q_2, y, L>$	-	$<q_1, y, R>$	-
q_2	$<q_2, a, L>$	—	$<q_0, x, R>$	$<q_2, y, L>$	-
q_3	—	-	—	$<q_3, y, R>$	$<q_A, B, L>$
$*q_{Accept}$	-	-	-	-	—

~~transition functions~~

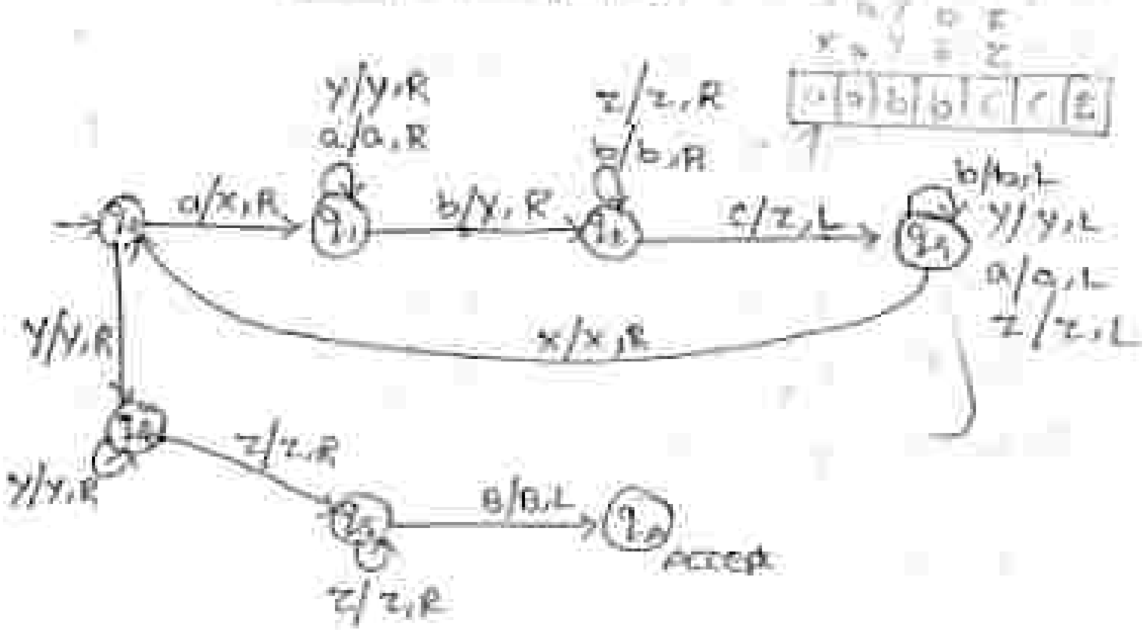
8

Design turing machine for $L = \{a^n b^n c^n / n \geq 1\}$

$L = \{abc, aabbcc, aaabbbccc, \ldots\}$

$TM = \{Q, \Sigma, \delta, \Gamma, F, q_0, B\}$

x x y y z z B
a a b b c c B ....



i/p symbol state	a	b	c	x	y	z	B
$\rightarrow q_0$	$<q_1, x, R>$	—	—	—	$<q_4, y, R>$	—	—
q_1	$<q_1, a, R>$	$<q_2, y, R>$	—	—	$<q_1, y, R>$	—	—
q_2	—	$<q_2, b, R>$	$<q_3, z, L>$	—	—	$<q_2, z, R>$	—
q_3	$<q_3, a, L>$	$<q_3, b, L>$	—	$<q_0, x, R>$	$<q_3, y, L>$	$<q_3, z, L>$	—
q_4	—	—	—	—	$<q_4, y, R>$	$<q_5, z, R>$	—
q_5	—	—	—	—	—	$<q_5, z, R>$	$<q_A, B, L>$
q_A	—	—	—	—	—	—	—

TM: $M = \{Q, \Sigma, \Gamma, \delta, q_0, B, F\}$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_h\}$

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

$\Gamma = \{a, b, c, x, y, z, B\}$

$\delta(q_0, a) = (q_1, x, R)$ $\qquad$ $\delta(q_3, b) = (q_3, b, L)$

$\delta(q_0, y) = (q_4, y, R)$ $\qquad$ $\delta(q_3, x) = (q_0, x, R)$

$\delta(q_1, a) = (q_1, a, R)$ $\qquad$ $\delta(q_3, y) = (q_3, y, L)$

$\delta(q_1, b) = (q_2, y, R)$ $\qquad$ $\delta(q_3, z) = (q_3, z, L)$

$\delta(q_1, y) = (q_1, y, R)$ $\qquad$ $\delta(q_4, y) = (q_4, y, R)$

$\delta(q_2, b) = (q_2, b, R)$ $\qquad$ $\delta(q_4, z) = (q_5, z, R)$

$\delta(q_2, c) = (q_3, z, L)$ $\qquad$ $\delta(q_5, z) = (q_5, z, R)$

$\delta(q_2, z) = (q_2, z, R)$ $\qquad$ $\delta(q_5, B) = (q_h, B, L)$

$\delta(q_3, a) = (q_3, a, L)$

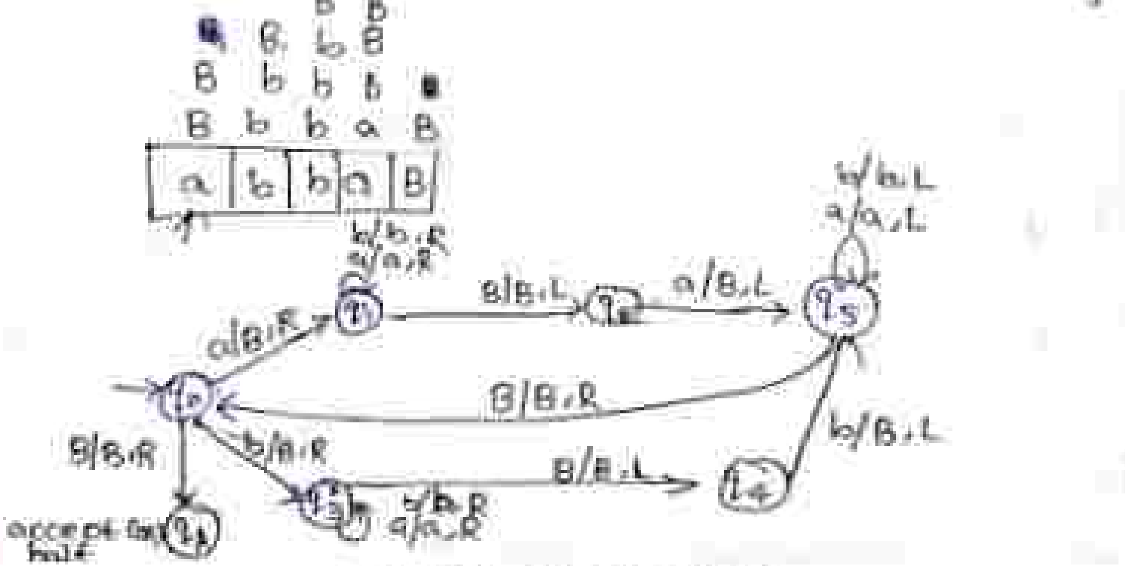
$F = \{q_h\}$

9) Design turing machine for $L = \{ww^R / w \in (a,b)^*\}$

Sol: $L = \{ww^R / w \in (a,b)^*\}$

It is a even length palindrome

$L = \{aa, bb, abba, baab, abbba, abaaba, \ldots\}$



i/p symbol / State	a	b	B
$\rightarrow q_0$	$<q_1, B, R>$	$<q_3, B, R>$	$<q_A, B, R>$
q_1	$<q_1, a, R>$	$<q_1, b, R>$	$<q_2, B, L>$
q_2	$<q_5, B, L>$	-	-
q_3	$<q_3, a, R>$	$<q_3, b, R>$	$<q_4, B, L>$
q_4	-	$<q_5, B, L>$	-
q_5	$<q_5, a, L>$	$<q_5, b, L>$	$<q_0, B, R>$
$* q_A$	-	-	-

ii) abba

q_0 a b b a B

$\vdash$ B q_1 b b a B

$\vdash$ B b q_1 b a B

$\vdash$ B b b q_1 a B

$\vdash$ B b b a q_1 B

$\vdash$ B b b q_2 a B

$\vdash$ B b q_5 b B B

$\vdash$ B q_5 b b B B

$\vdash$ q_5 B b b B B

$\vdash$ B q_0 b b B B

$\vdash$ B B q_3 b B B

$\vdash$ B B b q_3 B B

$\vdash$ B B q_4 b B B

$\vdash$ B q_5 B B B B

$\vdash$ B B q_0 B B B

$\vdash$ B B B q_A B B

accept

Design turing machine for 'parity counter' that outputs '0' (or) '1' depending (no) on ω

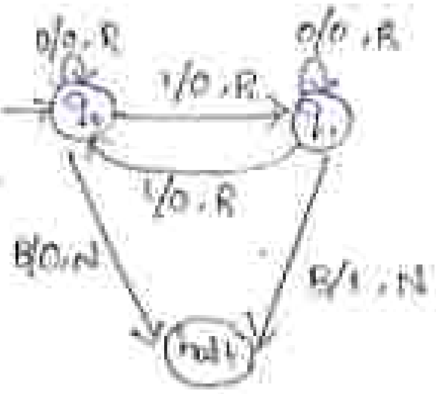
1's odd - 1

B even - 0

0	0	0	1
0	1	0	B

transition Table

0	0	0	0	0	0	0
0	1	1	0	1	1	B



odd 1's 010

$\vdash q_0 010 B$

$\vdash 0 q_0 10 B$

$\vdash 00 q_1 0 B$

$\vdash 000 q_1 B$

$\vdash 0001$ halt

even 1's 1010

$\vdash q_0 1010 B$

$\vdash 0 q_1 010 B$

$\vdash 00 q_1 10 B$

$\vdash 000 q_0 0 B$

$\vdash 0000 q_0 B$

$\vdash 00000$ halt

Design TM for 2's complement

I/P 10010

1's 01101

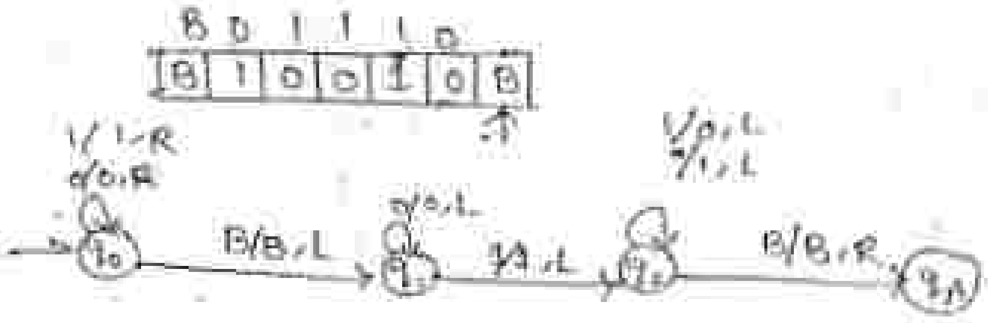
+1 1

2's 01110

	MSB				LSB	
B	1	0	0	1	0	B

accept
halt

First of all we have to move LSB then upto



q_0:-B0010B

$\vdash$B$q_0$10010B

$\vdash$B1$q_0$0010B

$\vdash$B10$q_0$010B

$\vdash$B100$q_0$10B

$\vdash$B1001$q_0$0B

$\vdash$B10010q_0B

$\vdash$B1001$q_1$0B

$\vdash$B100$q_1$10B

$\vdash$B10$q_2$010B

$\vdash$B1$q_2$0110B

$\vdash$B$q_2$11110B

$\vdash$BB01110B

$\vdash$$q_2$B01110B

$\vdash$B$q_2$01110B

Addition of two integers. Design TM)

Design turing Machine for to accept set of all the palindromes.

Sol:

a	b	a	B

B b a B

B b B B

B B B

halt

b	a	b	B

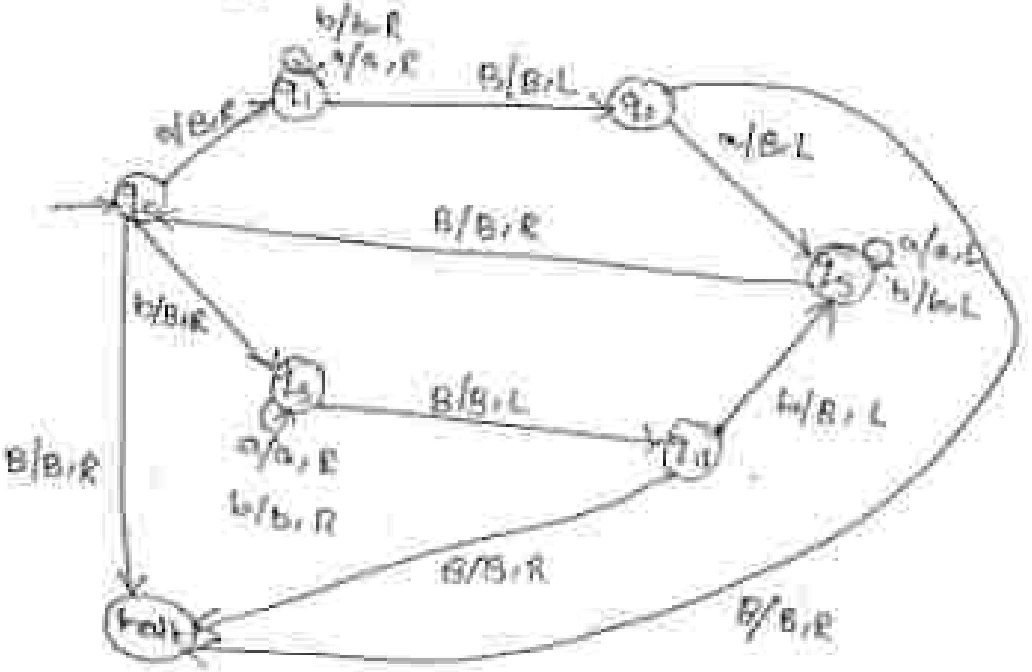
B a b B

B a B B

B B B

B B

halt



bab

⊢ $q_0$babB

⊢ B$q_3$abB

⊢ Baq_3bB

⊢ Babq_3B

⊢ Baq_4bB

⊢ Bq_5aBB

⊢ q_5BaBB

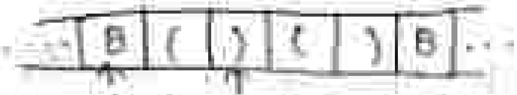
⊢ B$q_0$abB

⊢ BBq_1BB

⊢ Bq_2BBB

⊢ BBq_hBB

Design turing Machine for parenthesis checking



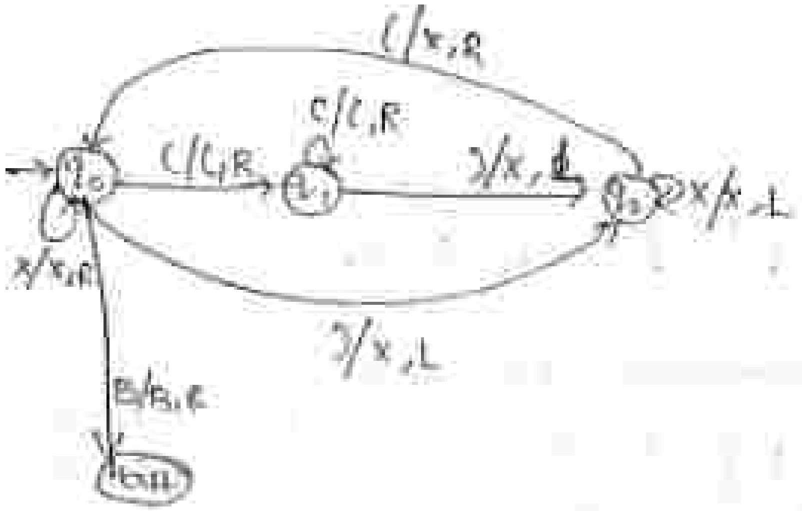
B (x ( ) B

B x x ( ) B

B x x (x B

B x x x x B

↑ B



⊢ q_0(())B

(q_1())B

((q_1))B

((q_2x)B

(q_2(x)B

(x q_0x)B

(x x q_0)B

(x q_0 x x B

(q_2 x x x B

q_2 (x x x B

x q_0 x x x B

x x q_0 x x B   x x x x B H

x x x q_0 x B

x x x x q_0 B

Types of Grammars - Chomsky Hierarchy:

Linguist Noam Chomsky defined a hierarchy of languages, in terms of complexity. This four level hierarchy, called the Chomsky hierarchy, corresponds to four classes of machines.

The Chomsky hierarchy classifies grammars according to form of their productions into the following four levels.

(1) Type 0 grammars - unrestricted grammars
(2) Type 1 grammars - Context sensitive grammars
(3) Type 2 grammars - Context free grammars
(4) Type 3 grammars - regular grammars.

(1) Type - 0 grammars - Unrestricted Grammars: (URG)

These grammars include all formal grammars. In URG, all the productions are of the form $\alpha \rightarrow \beta$, where α and β may have any number of terminals and non-terminals. ie, no restrictions on either side of productions. Every grammar is included in it if it has atleast one non-terminal on the left hand side.

Ex: $aA \rightarrow abCB$ ie $(\alpha \in V)^+$
$aA \rightarrow bAA$
$bA \rightarrow a$
$S \rightarrow aAb \mid \epsilon$

They generate exactly all languages that can be recognized by a turing machine. The language that is recognized by a Turing machine is defined as set of all the strings on which it halts. These Languages are also known as the recursively enumerable languages (2)

(2) Type 1 grammar – Context sensitive Grammars: (CSG)

These grammars define the context-sensitive languages. In context sensitive grammar, all the productions of the form $\alpha \rightarrow \beta$, where length of α is lessthan or equal to β the length β. ie $|\alpha| \le |\beta|$. α and β are may have any number of terminals and non-terminals. These languages are as exactly all the languages that can be recognized by linear bound automata. (2)

Ex:- $|\alpha| \le |\beta|$ $\alpha \rightarrow \beta$

$aAbcD \rightarrow abcDbcD$

(3) Type II Grammar – Context-free Grammar (CFG)

These grammars define the context free languages. These are defined by rules of the form $\alpha \rightarrow \beta$ with $|\alpha| \le |\beta|$ where $|\alpha| = 1$ and α is non-terminal and β is a string of terminals and non-terminals. we can replace α by β regardless of where it appears. Hence the name context free grammar. These languages are exactly those languages that can be recognized by a pushdown automaton. CFG Context free languages defines the syntax of all programming languages.

Ex:- (i) $S \rightarrow aS \mid Sa \mid a$
(ii) $S \rightarrow aAA \mid bBB \mid \epsilon$.

(4) Type 3 grammars - regular grammars:

These grammars generate the regular languages. Such a grammar restricts its rules to a single non-terminal on the LHS. The RHS consists of either a single terminal or string of terminals with single non-terminal on left or right end.

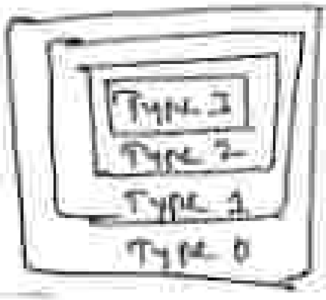
$\alpha \to \beta$, $\alpha = \{V\}$
$\beta = V\{T\}^*$ + $\{T\}^* V$ + T^*

Ex: $A \to aA|a$ — right linear grammar
$A \to Aa|a$ — left linear grammar.

* Every regular language is context free, every context-free language is context-sensitive and every context sensitive language is recursively enumerable.

Table: Chomsky's Hierarchy

Grammar	language	Automaton	production rules
Type 0	Recursively enumerable	Turing machine	$\alpha \to \beta$ no restriction on β, α should have at least one non terminals
Type 1	Context sensitive	Linear bounded Automata	$\alpha \to \beta$ $\|\alpha\| \le \|\beta\|$
Type 2	Context free	push down automata	$\alpha \to \beta$ $\|\alpha\| = 1$
Type 3	Regular	finite state automaton	$\alpha \to \beta$. $\alpha = \{V\}$ $\beta = \{V\}T^*$ or $= T^*\{V\}$ $= T^*$



(3)

chomsky hierarchy of grammars

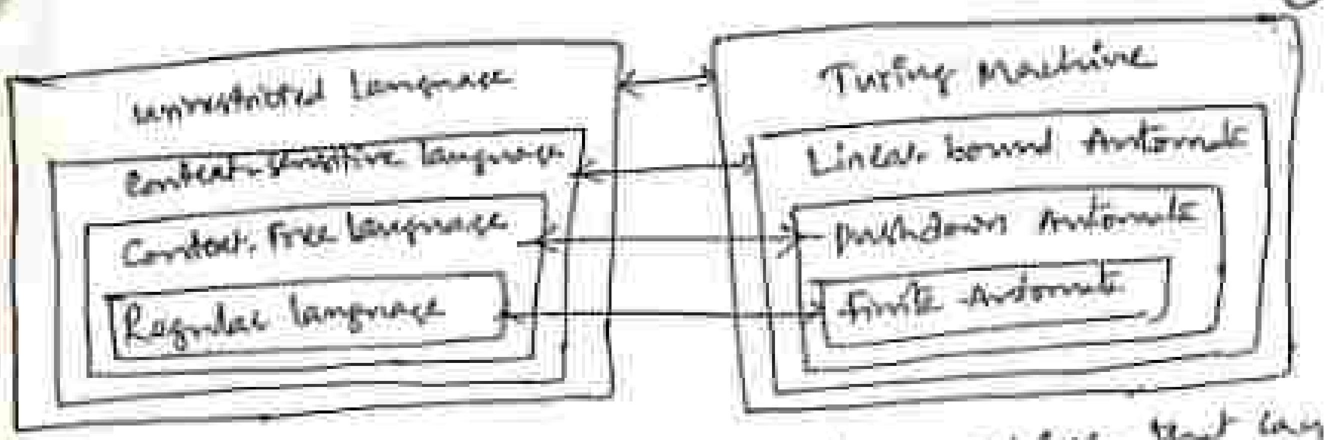


fig: The hierarchy of languages and the machine that can recognize the same is shown above fig.

. Every RG is context free, every CFG is context sensitive and every CSG is unrestricted.& So the family of regular language can be recognized by any machine.

. CFLs are recognized by pushdown automata, Linear bounded automata and Turing machines.

. CSLs are recognized by Linear bounded automata and Turing machines

. Unrestricted languages are recognized by only Turing machine

(G)

\

Push Down Automata (PDA)

(68)

PDA – $\boxed{\text{FA + Stack}}$
memory element

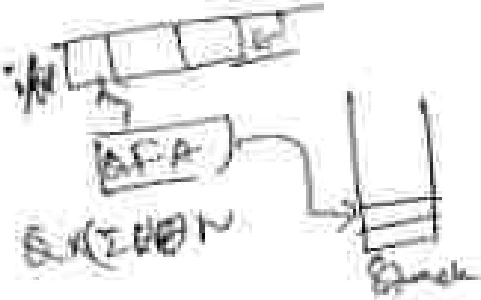
PDA: $(Q, \Sigma, \delta, q_0, Z_0, F, \Gamma)$
(7 tuple)

PDA
- DPDA
- NDPDA

Q = finite set of states
Σ = input symbol
δ = Transition function
q_0 = initial state
Z_0 = Bottom of the stack
F = set of final states
Γ = stack alphabet.

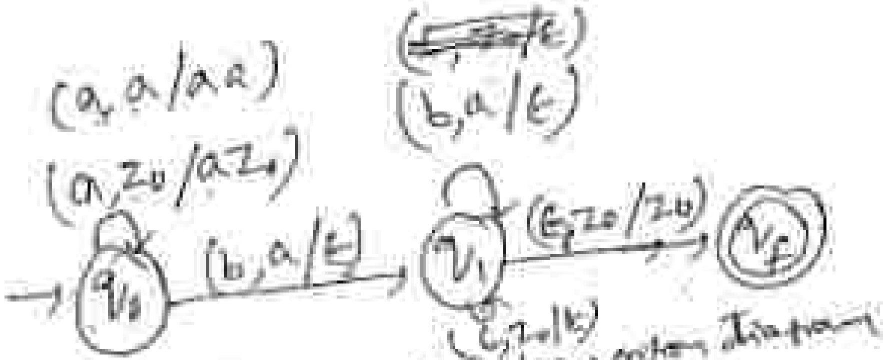
DPDA: $\delta : \underbrace{Q}_{\text{State}} \times \underbrace{\{\Sigma \cup \epsilon\}}_{\text{Input}} \times \underbrace{\Gamma}_{\text{Top}} \to \underline{Q \times \Gamma}$

NDPDA $\delta : \underbrace{Q}_{\text{State}} \times \underbrace{\{\Sigma \cup \epsilon\}}_{\text{input}} \times \underbrace{\Gamma}_{\text{Top of stack}} \to \underline{2}^{(Q \times \Gamma)^*}$



Ex: $\underline{a^n b^n \mid n \geq 1}.$

a a b b ε
↑ ↑ ↑↑ ↑



State transition diagram

a a b b ε
↑ ↑ ↑↑ ↑

$\delta(q_0, a, Z_0) = (q_0, aZ_0)$
$\delta(q_0, a, a) = (q_0, aa)$
$\delta(q_0, b, a) = (q_1, \epsilon)$
$\delta(q_1, b, a) = (q_1, \epsilon)$
$\delta(q_1, \epsilon, Z_0) = \underbrace{(q_f, Z_0)}_{\text{accepting by final state}}$ or $\underbrace{(q_1, \epsilon)}_{\text{Acceptance by empty stack}}$

DPDA

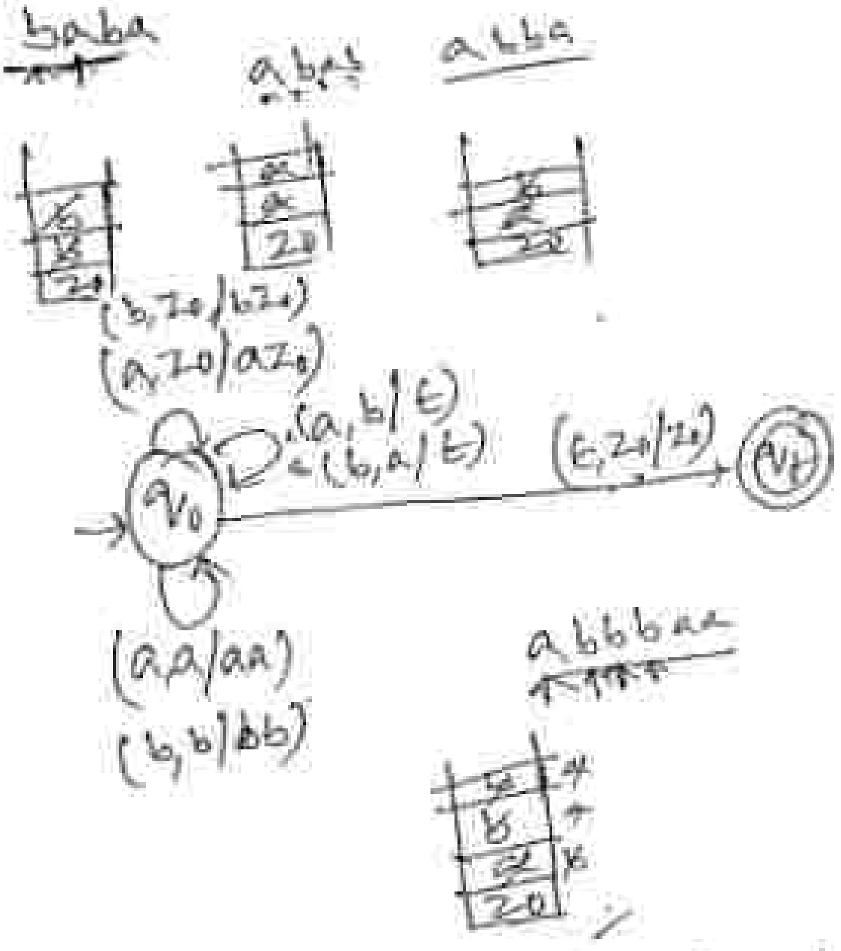
transition function

PDA
- final state
- Empty stack.

$\{w \mid n_a(w) = n_b(w)\}$ [number of a's must be equal].

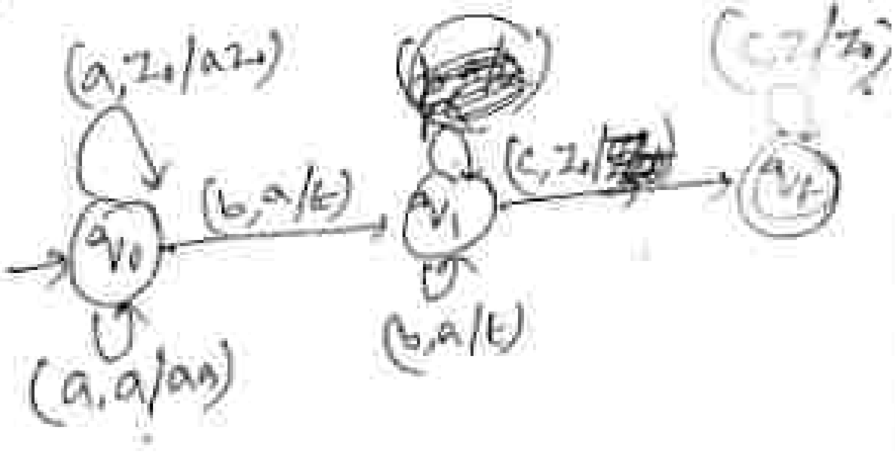
DPDA

Eg: ab bbaa
aabb baba
abab



$a^n b^n c^m \mid n, m \geq 1$.

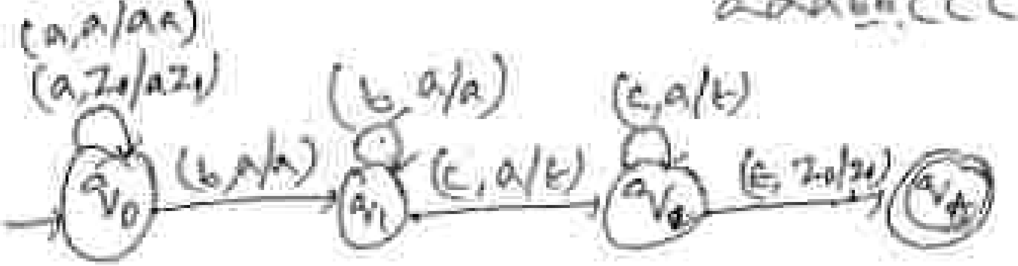
Idea
$a^n \to$ push
$b^n \to$ pop
$c^m \to$ no change



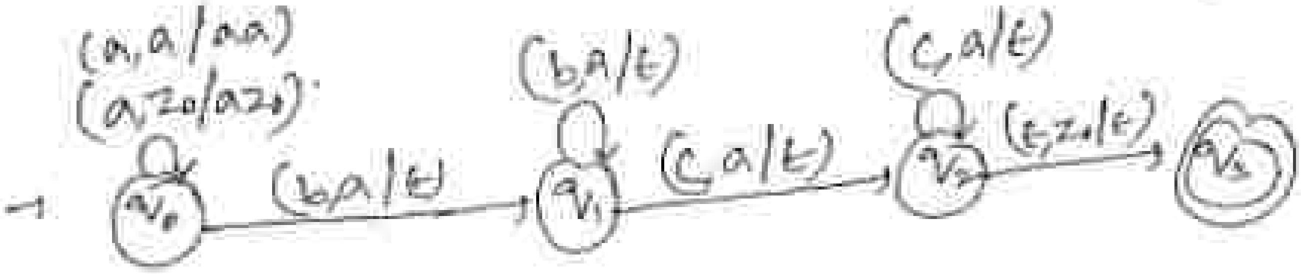
Q) $a^n b^m c^n \mid n, m \geq 1$ don't want count b

aaabbccc

Sol

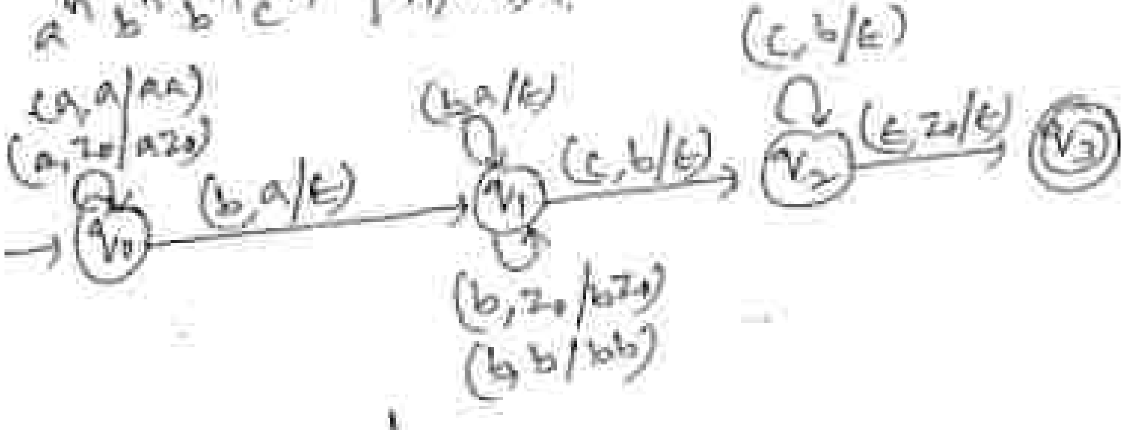


Q) $a^{m+n} b^m c^n \mid m, n \geq 1$



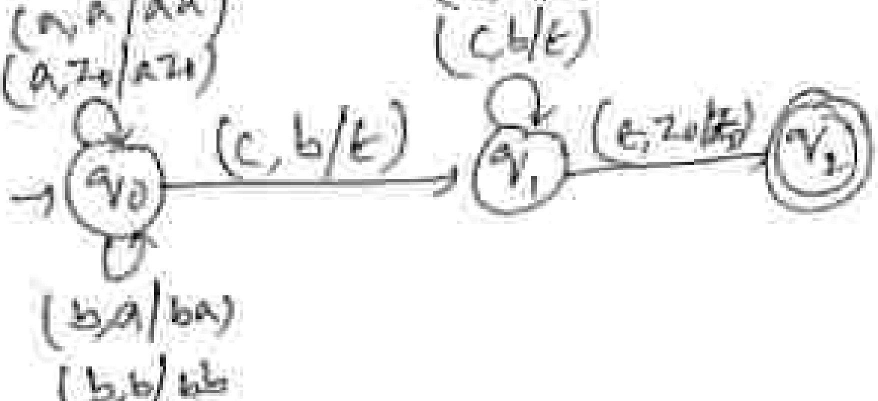
Q) $a^n b^{m+n} c^m \mid n, m \geq 1$

Sol) $a^n b^n b^m c^m \mid n, m \geq 1.$



Q) $a^n b^m c^{n+m} \mid n, m \geq 1.$

Sol



$a^n b^n c^m d^m \mid n,m \geq 1$

$a^n b^m c^m d^n \mid n,m \geq 1$

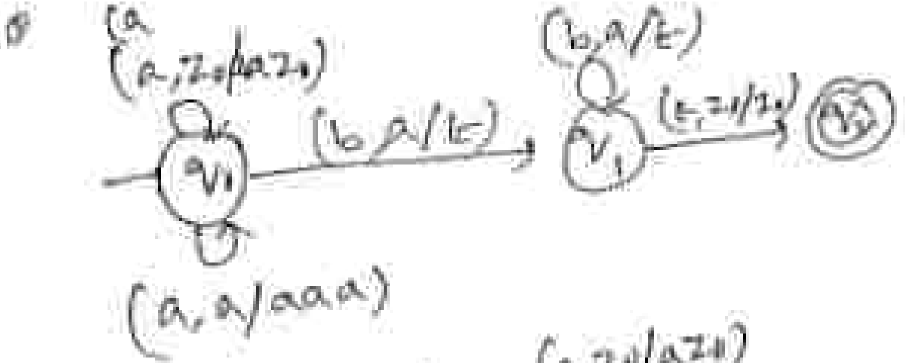
$a^n b^m c^n d^m \mid n,m \geq 1$ × is not CFG not PDA

$a^n 2^n \mid n \geq 1$

$a^n b^{2n} \mid n \geq 1$

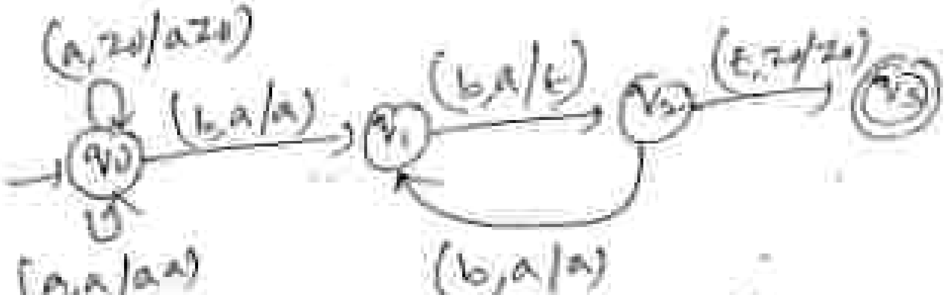
1. abb, aabbbb, aaabbbbbb . . .

two solutions < for every a — push two a's. / for every b — pop two b's



bb, bb,

for every b

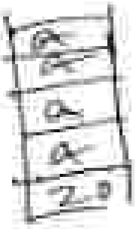


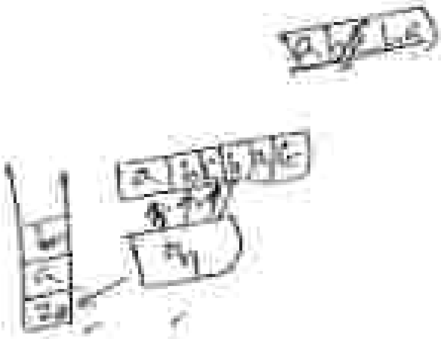
aabbbbε
a
a
Z0

Ex:- $a^n b^n c^n \mid n \geq 1$ X not PDA

one problem

X aabbcc

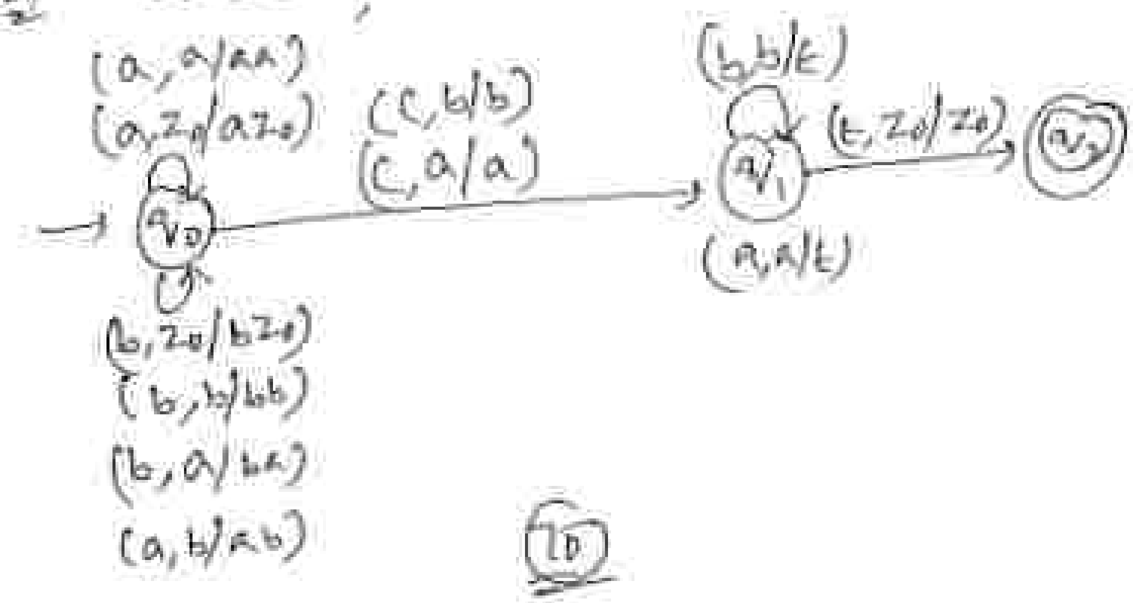




for every a → two a's push

Ex: $wcw^R \mid w \in (a,b)^+$

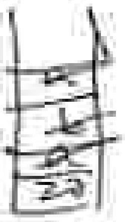
Sol: abcba, abbcbba



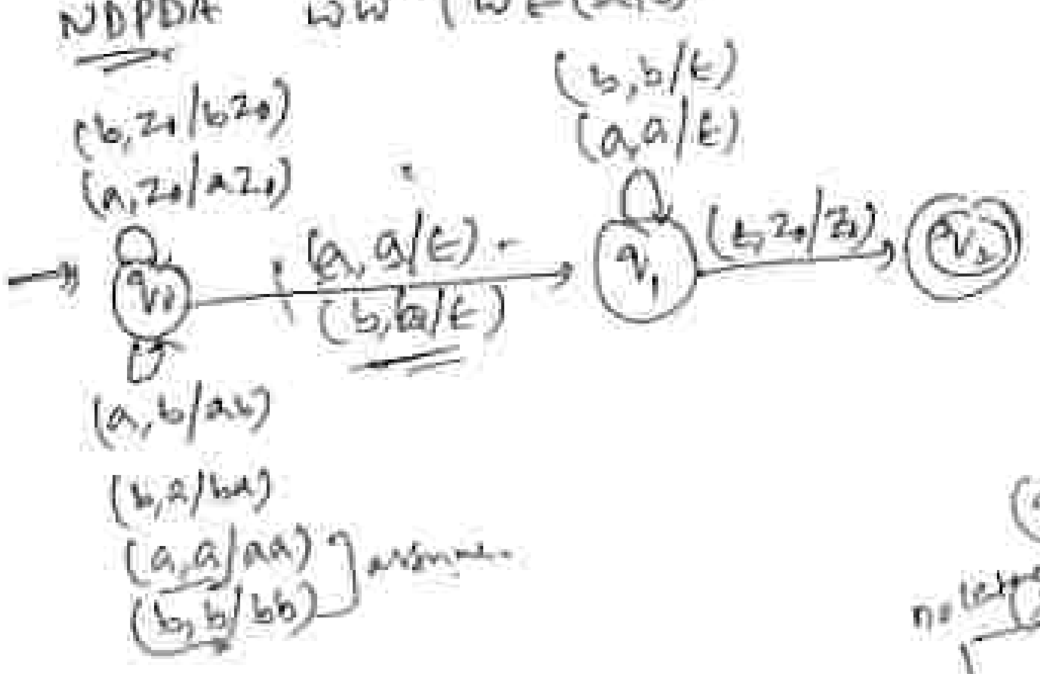
ID

Ex: $ww^R \mid w \in (a+b)^+$ → whenever

Sol: $\underbrace{aba}_{w}\underbrace{aba}_{w^R}$



NDPDA $ww^R \mid w \in (a+b)^+$

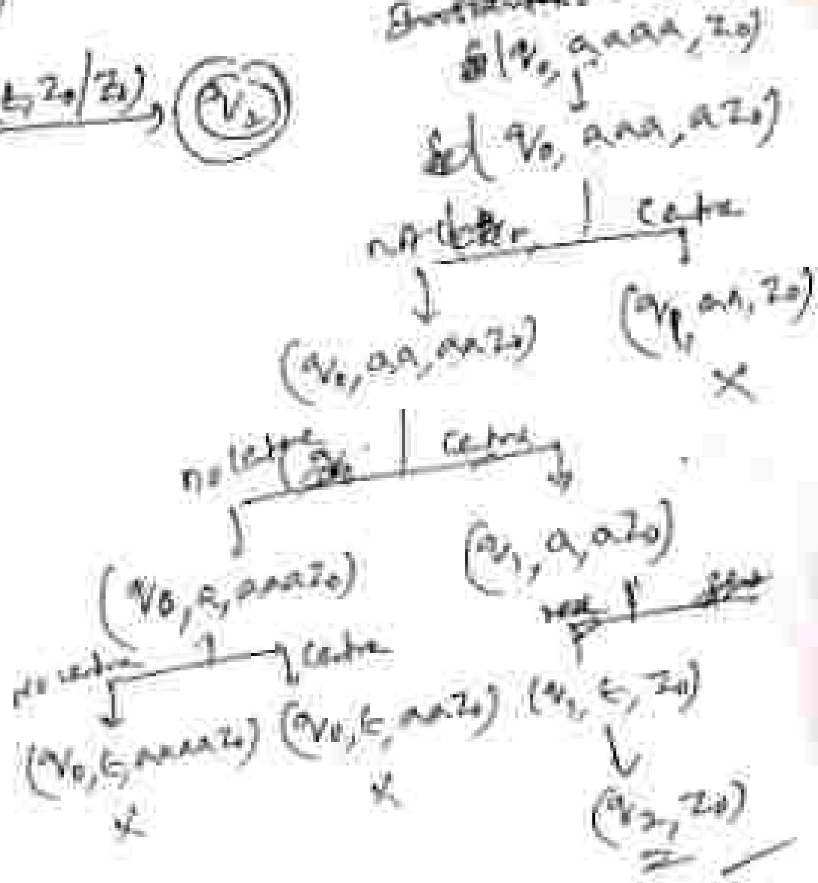


abba

Instantaneous Desc

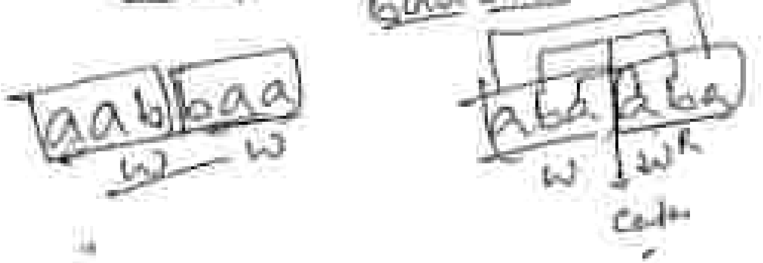
aaaa

~~Instantaneous description~~

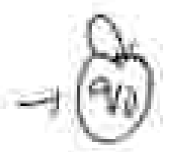


abbaab

baaaab



aa|aa
W
WR
NDPDA
Centre has come
pop
Centre not come
push



Chance of getting a centre

ab|ba.

UNIT-VI

language decidability

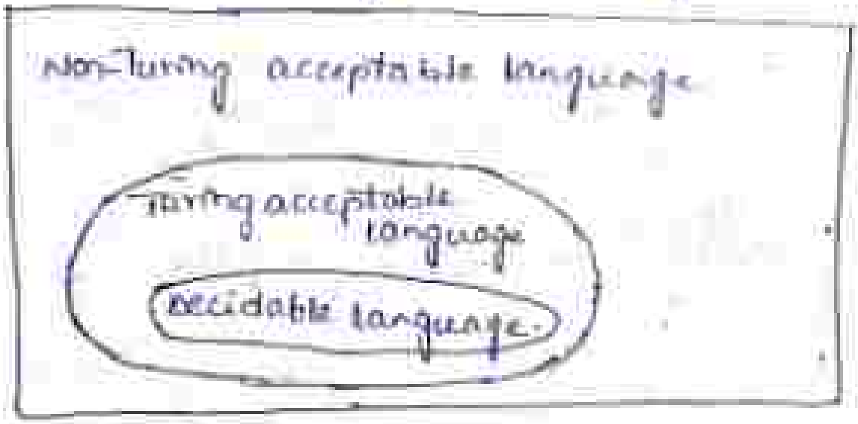
- Introduction
- Examples

Introduction:-

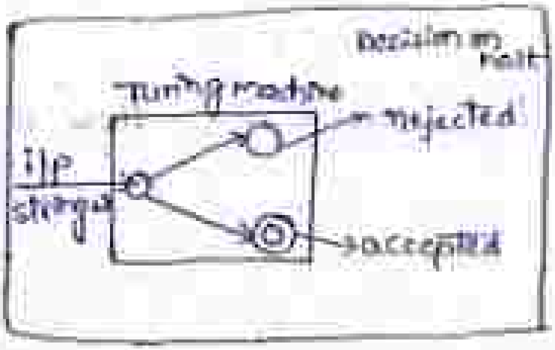
- **Decidable problem:—**

- A language is called Decidable (or) recursive if there is a turing machine which accepts and halts on every i/p string "w".

- Every decidable language is a turing acceptable



- A decision problem "p" is decidable if the language "L" of all "yes" instances to "p" is decidable.
- for a decidable language, for each i/p string, the turing machine halts either at the accept (or) the reject state



Examples:-

1) find out whether the following problem is decidable (or) not:

Is a number "m" prime ?

Sol:- Prime numbers = {2,3, 5,7, 11,13,17,19, ----}

divide the number "m" by all the numbers b/w

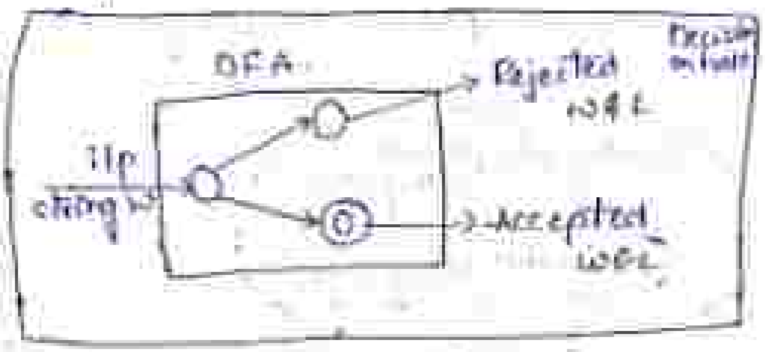
5 and m/2 starting from 2.

If any of these numbers produce a remainder 0 then it goes to the rejected state otherwise it goes to the accepted state. so, here the answers could be mainly yes (or) NO.

Hence, it is a decidable problem.

3) Given a Regular language 'L' and string 'w', how can we check if w∈L.

sol:- Take the DFA that accepts 'L' and check if is is accepted.



Some more decision problems are

i) Does DFA accept the empty language

ii) Is $L_1 \cap L_2 = \phi$ for regular sets

iii) If a language L is decidable then its complement L' is also decidable.

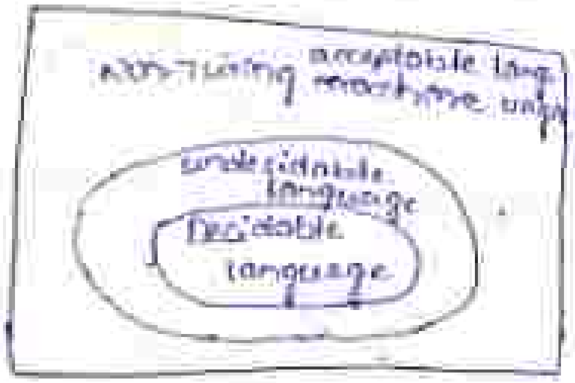
iv) If a language is decidable then there is a turing machine for it.

Undecidable problems:-

Introduction:-

- for an undecidable language there is no TM which accepts the language and makes a decision for every i/p string 'w'.
- A decision problem 'p' is called undecidable if the language 'L' of all 'yes' instances to 'p' is not decidable.

undecidable languages are not recursive languages but, sometimes they may be recursive enumerable language



Examples :-

i) The halting problem of turing machine

ii) The mortality problem.

iii) The mortal matrix problem.

iv) The post correspondence problem [pcp]

i) The halting problem :

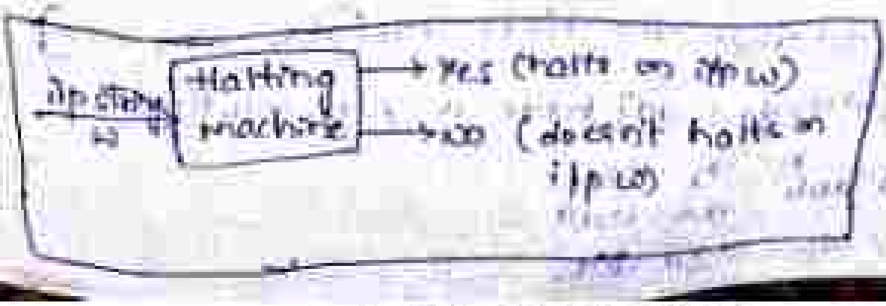
The halting problem i/p : a turing machine and the i/p string w

problem : Does the turing machine finish computing of the string 'w' in a finite no. of steps ? The answer must be either yes (or) no

proof :- At first, we will assume that a turing machine exists to solve the problem we will show and then it is contradicting it self.

we will call this turing machine as a halting machine that produces a yes (or) no in a finite amount of time

If the halting machine finishes in a finite amount of time then the o/p comes as yes. otherwise, as no

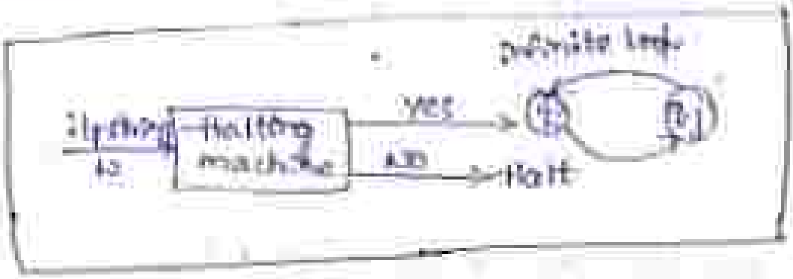


Now, we will design an inverted halting machine as

i) If HM returns yes then loop forever.

ii) If HM returns no then halt.

The following block diagram show the inverted halting machine.



-Further, a machine "HM" which ip itself is constructed as follows

i) If HM halts on ip loop forever.

ii) Else Halt.

∴ Here we have got a Contradiction, Hence, the halting problem is undecidable.

Post correspondence problem (pcp):—

- It was introduced by "Emilpost" in 1946 is as undecidable decision problem.

The pcp problem over an ip alphabet "Σ" is stated as follows.

Given the following two lists, M and N, of non-empty strings over Σ

$M = (x_1, x_2, x_3 \cdots x_n)$

$N = (y_1, y_2, y_3 \cdots y_n)$

we can say that there is a pcp solution if for some $i_1, i_2, i_3 \cdots i_k$ where $1 \leq i_k \leq n$, The condition.

$\boxed{x_{i_1} x_{i_2} x_{i_3} \cdots x_{i_k} = y_{i_1} y_{i_2} y_{i_3} \cdots y_{i_k}}$ satisfies

Ex:- Q find whether the lists M = {abb, aa, aaa} and N = [bba, aaa, aa] have a pcp solution.

Sol:-

	x_1	x_2	x_3
M	abb	aa	aaa
N	bba	aaa	aa

Here $x_2 x_1 x_3$ = aaabbaaa

$y_2 y_1 y_3$ = aaabbaaa

we can see that $x_2 x_1 x_3 = y_2 y_1 y_3$

Hence, the solution is i=2, j=1, k=3

2) find whether the list M = [ab, bab, bbaaa] and N=[a, ba, bab] have a pcp solution

sol:-

	x_1	x_2	x_3
M	ab	bab	bbaaa
N	a	ba	bab

In this case, there is no solution because

$|x_2 x_1 x_3| \neq |y_2 y_1 y_3|$ lengths are not same

Hence, it can be said that this pcp is a undecidable problem.

Modified post correspondence problem:-

Given two lists $M = x_1 x_2 x_3 \cdots x_n$ and $N = y_1 y_2 y_3 \cdots y_n$

Given a set of pair of strings $\{(x_1, y_1)(x_2, y_2) \cdots (x_n, y_n)\}$ then the solution is an instance such that,

$$\boxed{x_1 x_{i_1} x_{i_2} \cdots x_{i_n} = y_1 y_{i_1} y_{i_2} \cdots y_{i_n}}$$

that means the pair (x_1, y_1) is forced to be at the begning of the strings

Ex:-

	x_1	x_2	x_3
M	11	100	111
N	111	001	11

Sol:- Then the solution is $x_1 x_2 x_3 = y_1 y_2 y_3$

$x_1 x_2 x_3$ = 11100111

$y_1 y_2 y_3$ = 11100111

That means it is essential to have x_1 & y_1 at the begining of ...

P and Np classes :-

- p-problems
- Np-problems
- P Vs Np

- P-problems :-

- P is the class of problems that can be solved by deterministic algorithm in a polynomial type time p(n), when n is the size of i/p string.
- p-problem consist of a language accepted by deterministic Turing machine that runs in polynomial amount of time.

Ex :- 1) shortest path problem
2) Equivalence of NFA and DFA
3) shortest cycle in a graph.
4) sorting algorithms

- Np-problems :-

- Np-problem is a class of problems that can be solved by Non-deterministic algorithms in a polynomial time p(n) where 'n' is the size of i/p string.
- Np-problems consists of a language accepted by Non-deterministic turing machine that runs in a polynomial amount of time.

Ex - 1) Travelling sales man problem.
2) subgraph isomorphism

Np-problem classified into two types.
i) Np- hard problem.
ii) Np- complete problem.

- Np-hard problem :-

If there is a language x such that every language y in Np can be polynomially reducible to x and we cannot prove that x is in Np then x is said to be Np-hard problem.

Ex: Turing machine halting problem.

- Np- Complete problem :-

If there is a language x such that every language y in NP can be polynomially reducable to x and we can prove that x is in NP then x is said to be NP-complete problems.

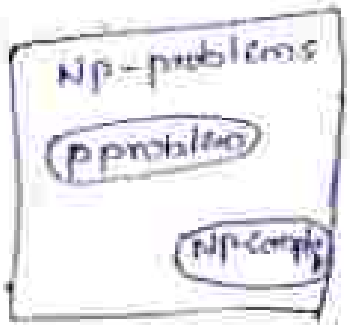
Ex:- 1) Travelling salesman problem

2) subgraph isomorphism

P vs NP:-

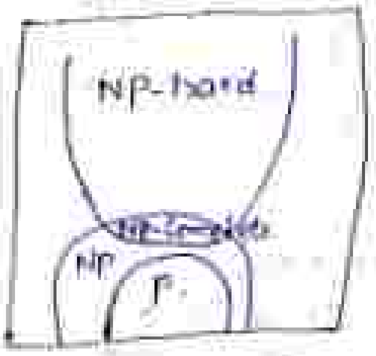
1. Ladner's theorem:

(a) P≠NP

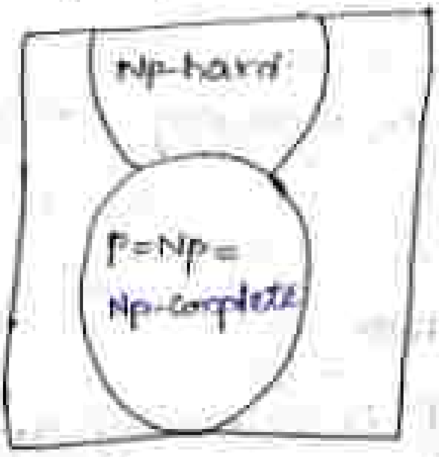


2. Euler's theorem:-

(a) P≠NP



(b) P=NP


NP-hard
P=NP=
NP-complete