TURING MACHINE

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

where Q = set of states

Σ = set of input symbols

Γ = set of input tape symbols

δ = Transition function

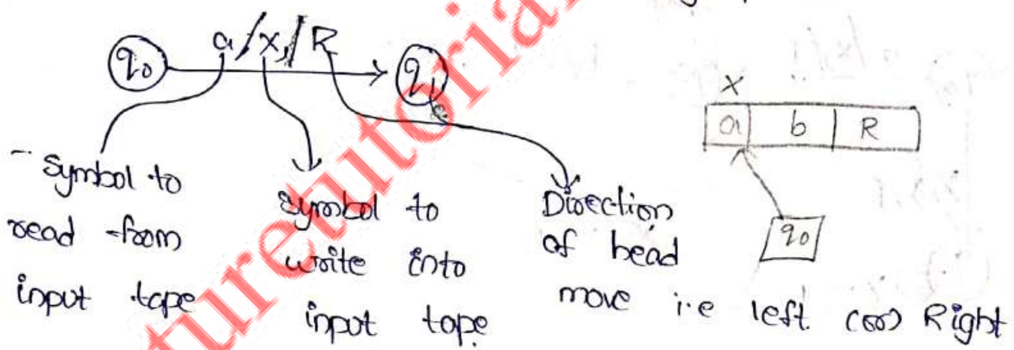
q_0 = initial state

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.



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

$L = 01^*0$ $\quad 0\,1^*\,0$

0	1	1	0	B

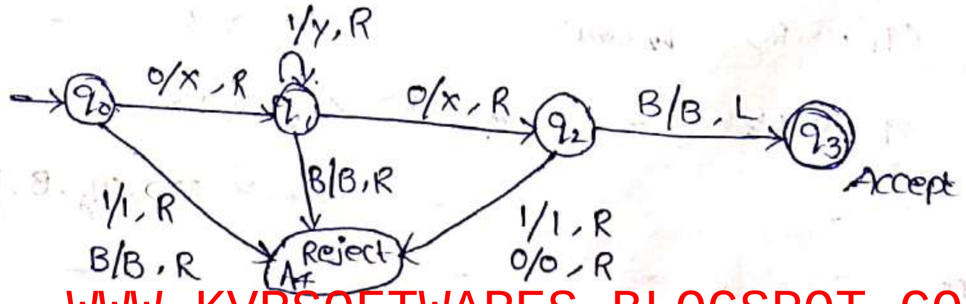
X Y Y X B

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

X		X	B
0		0	B

X	Y	X	B
0	1	0	B

X	Y	Y	X	B
0	1	1	0	B

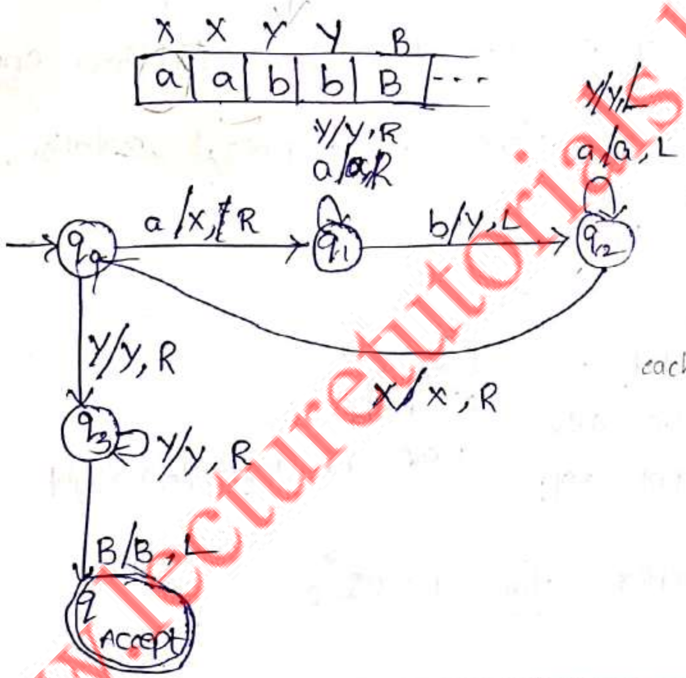


Tape Symbol / State	0	1	X	Y	B
q_0	$<q_1, X, R>$	$<q_4, 1, R>$	-	-	$<q_4, B, R>$
q_1	$<q_2, X, R>$	$<q_1, Y, R>$	-	-	$<q_4, B, R>$
q_2	$<q_4, 0, R>$	$<q_4, 1, R>$	-	-	$<q_3, B, L>$
q_3	-	-	-	-	-
q_4	-	-	-	-	-

*** M

2) Design a Turing Machine for language $L = \{a^n b^n / n \geq 1\}$

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



each 'a' is compared to 'b'

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}	-	-	-	-	—

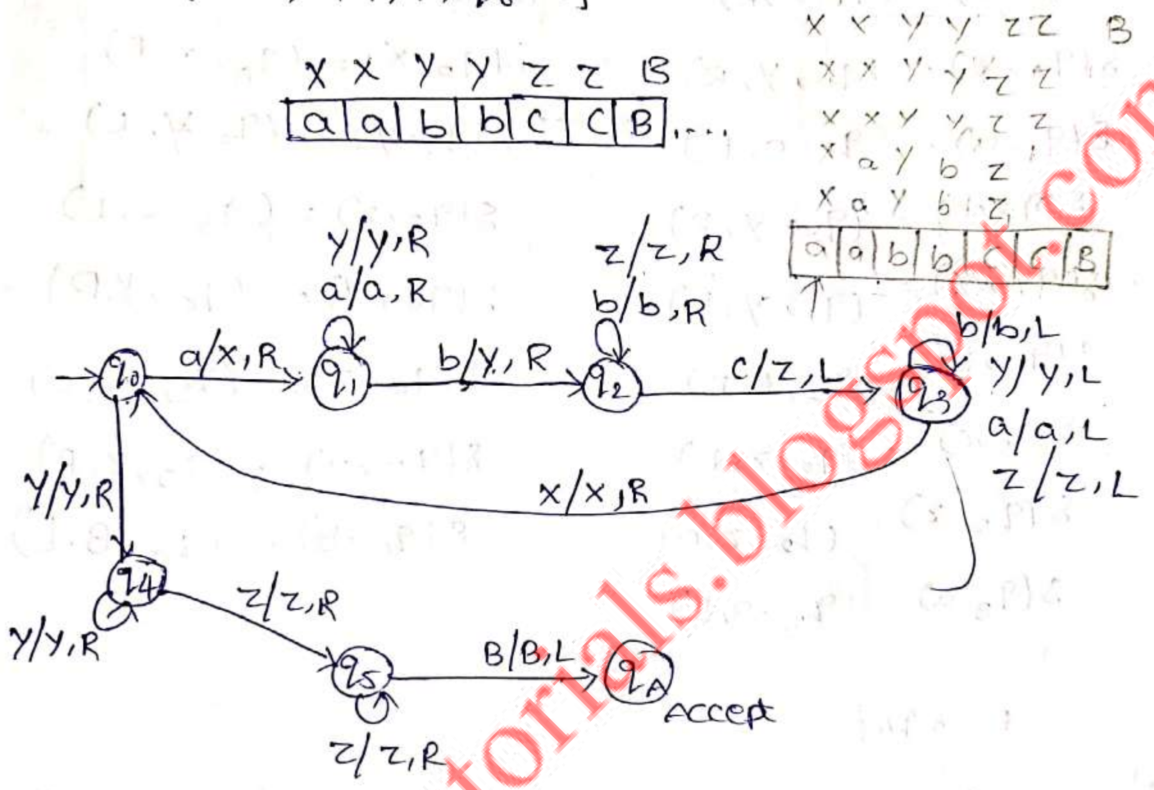
~~Transition functions~~

8

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

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

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



Tape symbol / State	a	b	c	X	Y	Z	B
→ 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: $M = \{Q, \Sigma, \Gamma, \delta, q_0, B, F\}$

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

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

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

$\delta(q_0, a) = (q_1, x, R)$

$\delta(q_0, y) = (q_4, y, R)$

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

$\delta(q_1, b) = (q_2, y, R)$

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

$\delta(q_2, b) = (q_2, b, R)$

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

$\delta(q_2, z) = (q_2, z, R)$

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

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

$\delta(q_3, x) = (q_0, x, R)$

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

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

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

$\delta(q_4, z) = (q_5, z, R)$

$\delta(q_5, z) = (q_5, z, R)$

$\delta(q_5, B) = (q_A, B, L)$

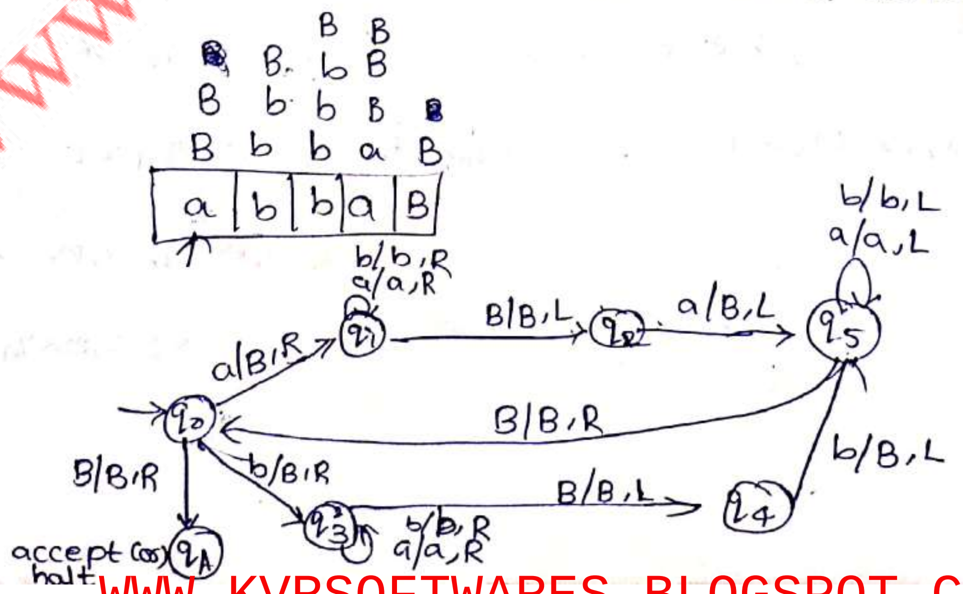
$F = \{q_A\}$

4) Design Turing Machine for $L = \{ww^R / w \in (a, b)^*\}$

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

It is a even length palindrome

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



Tape Symbol / State	a	b	B
$\rightarrow q_0$	$\langle q_1, B, R\rangle$	$\langle q_3, B, R\rangle$	$\langle q_A, B, R\rangle$
q_1	$\langle q_1, a, R\rangle$	$\langle q_1, b, R\rangle$	$\langle q_2, B, L\rangle$
q_2	$\langle q_5, B, L\rangle$	-	-
q_3	$\langle q_3, a, R\rangle$	$\langle q_3, b, R\rangle$	$\langle q_4, B, L\rangle$
q_4	-	$\langle q_5, B, L\rangle$	-
q_5	$\langle q_5, a, L\rangle$	$\langle q_5, b, L\rangle$	$\langle q_0, B, R\rangle$
$*q_A$	-	-	-

ID 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

5) Design turing Machine for 'parity Counter' that outputs '0' (or) '1' depending ...

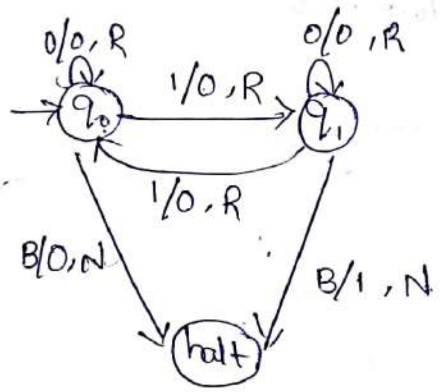
1's odd - 1

1's 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

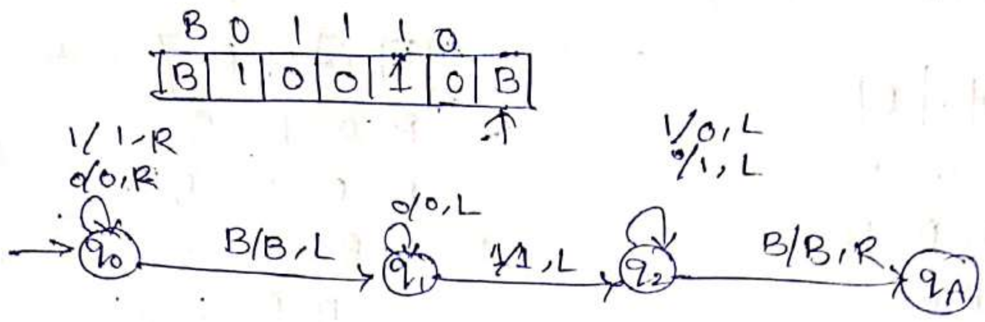
2's 01110

MSB LSB

B	1	0	0	1	0	B

accept
halt

First [illegible] we have to move LSB then upto



ID:- B1 0 0 1 0 B

⊢ B q_0 1 0 0 1 0 B

⊢ B 1 q_0 0 0 1 0 B

⊢ B1 0 q_0 0 1 0 B

⊢ B1 0 0 q_0 1 0 B

⊢ B1 0 0 1 q_0 0 B

⊢ B1 0 0 1 0 q_0 B

⊢ B1 0 0 1 q_1 0 B

⊢ B1 0 0 q_1 1 0 B

⊢ B 1 0 q_2 0 1 0 B

⊢ B 1 q_2 0 1 1 0 B

⊢ B q_2 1 1 1 1 0 B

⊢ B B 0 1 1 1 0 B

⊢ q_2 B 0 1 1 1 0 B

⊢ B q_2 0 1 1 1 0 B

(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

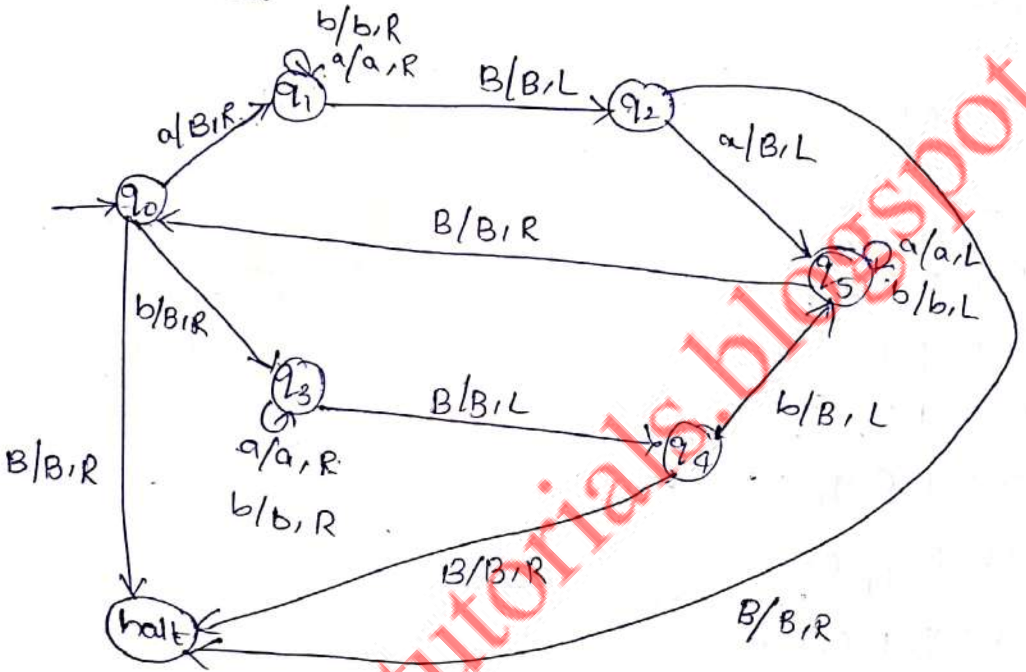
B a b B

B a B B

B B B

B B

halt



bab

$\vdash q_0 babB$

$\vdash Bq_3 abB$

$\vdash Ba q_3 bB$

$\vdash Bab q_3 B$

$\vdash Ba q_4 bB$

$\vdash B q_5 a BB$

$\vdash q_5 BaBB$

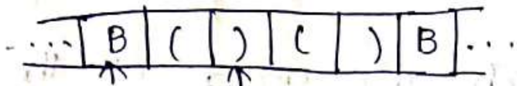
$\vdash B q_0 a b B$

$\vdash BB q_1 BB$

$\vdash B q_2 BBB$

$\vdash BB q_A BB$

Design Turing Machine for paranthesis checking



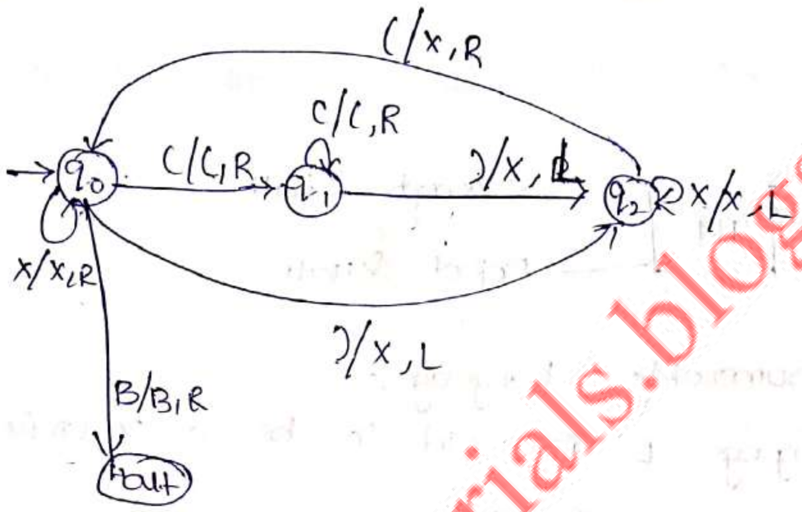
B (X ( ) B

B X X ~~( ) B~~

B X X (X B

B X X X X B

B



$\vdash$ q_0 (( )) B

(q_1 ( )) B

((q_1)) B

((q_2 X) B

(q_2 (X) B

(X q_0 X) 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 q_0 X B

X X X X q_0 B

X X X X B H

Types of Grammars - Chomsky Hierarchy:

Linguist Noam Chomsky defined a hierarchy of language 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.

Type 0
Type 1
Type 2
Type 3

(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 URGs, all the productions are of the form $\alpha \rightarrow \beta$, where α and β may have any number of terminals and non-terminals. i.e., 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:-
$aA \rightarrow abCB$
$aA \rightarrow bAA$
$bA \rightarrow a$
$S \rightarrow aAb \mid \epsilon$

ie ~~(V+T)+~~

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

Scanned by CamScanner

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 $\alpha \to \beta$, where length of α is less than or equal to the length β. i.e $|\alpha| \le |\beta|$, α and β are may have any number of terminals and non-terminals. These languages are exactly all the languages that can be recognized by linear bound automata.

(2)

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

aAbcD → abcDbcD

(3) Type 2 Grammar – Context-free Grammar (CFG)

These grammars define the context-free languages. These are defined by rules of the form $\alpha \to \beta$ 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. Context free languages defines the syntax of all programming languages.

Ex: i) S → aS | Sa | a
ii) S → aAA | bBB | ε.

(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.

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

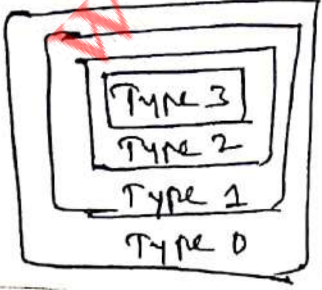
Ex: $A \to aA \mid a$ — right linear grammar

$A \to Aa \mid a$ — 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 restrictions on β, α; α should have at least one non-terminals
Type 1	Context-sensitive	Linear bounded Automata	$\alpha \to \beta$, $\lvert\alpha\rvert \le \lvert\beta\rvert$
Type 2	Context-free	push down automata	$\alpha \to \beta$, $\lvert\alpha\rvert = 1$
Type 3	Regular	Finite state automaton	$\alpha \to \beta$, $\alpha = \{V\}$, $\beta = \{V\}T^*$ or $= T^*\{V\}$ $= T^*$



chomsky hierarchy of grammars

(3)

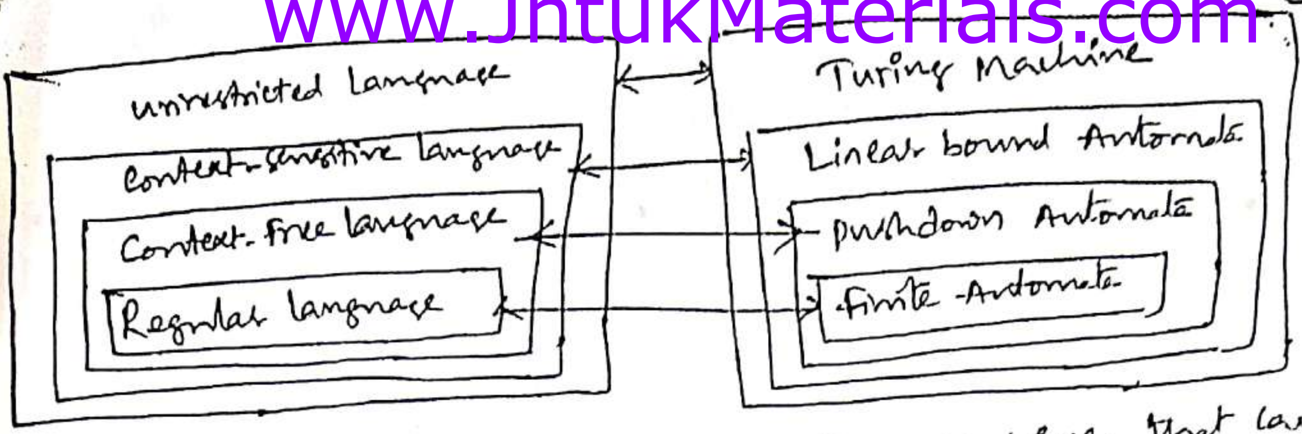


fig: The hierarchy of languages and the machine that can recognize the same is shown above fig.

- 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 machines

Push Down Automata (PDA)

(68)

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

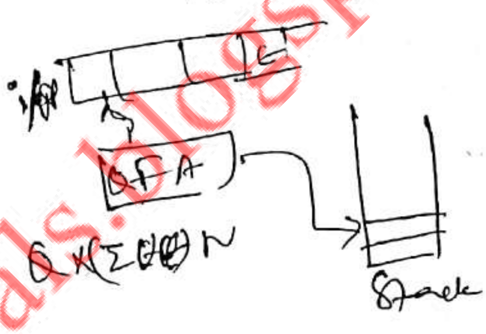
PDA = $(Q, \Sigma, \delta, q_0, Z_0, F, \Gamma)$ (7 tuple)

- 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.

PDA → DPDA, NDPDA

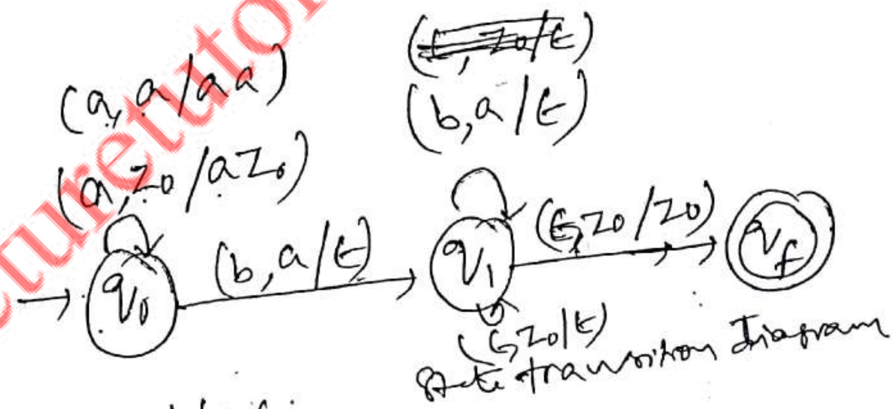
DPDA: $\delta : \underset{\text{state}}{Q} \times \underset{\text{input}}{\{\Sigma \cup \epsilon\}} \times \underset{\text{top}}{\Gamma} \rightarrow Q \times \Gamma$

NDPDA: $\delta : \underset{\text{state}}{Q} \times \underset{\text{input}}{\{\Sigma \cup \epsilon\}} \times \underset{\text{top of the stack}}{\Gamma} \rightarrow 2^{(Q \times \Gamma)^*}$



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

aabbε



aabbε

DPDA

transitions functions

$\delta(q_0, a, Z_0) = (q_0, aZ_0)$

$\delta(q_0, a, a) = (q_0, aa)$

$\delta(q_0, b, a) = (q_1, \epsilon)$

$\delta(q_1, b, a) = (q_1, \epsilon)$

$\delta(q_1, \epsilon, Z_0) = (q_f, Z_0)$ or (q_1, ϵ)

accept by final state · Acceptance by empty stack

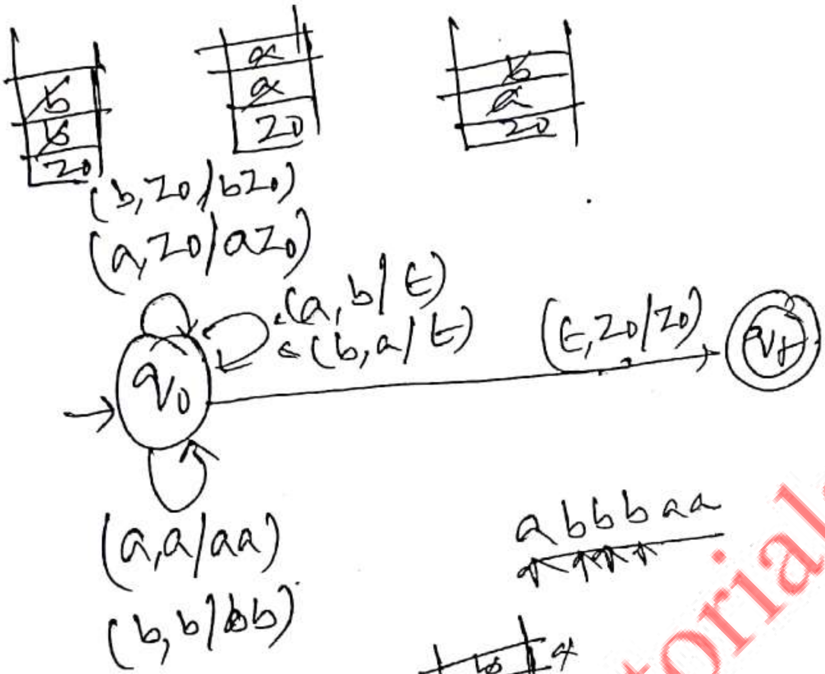
PDA → final state, Empty stack.

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

DPDA

Eg: ab, aabb, bbaa, baba, abab

baba abab abba



abbbaa

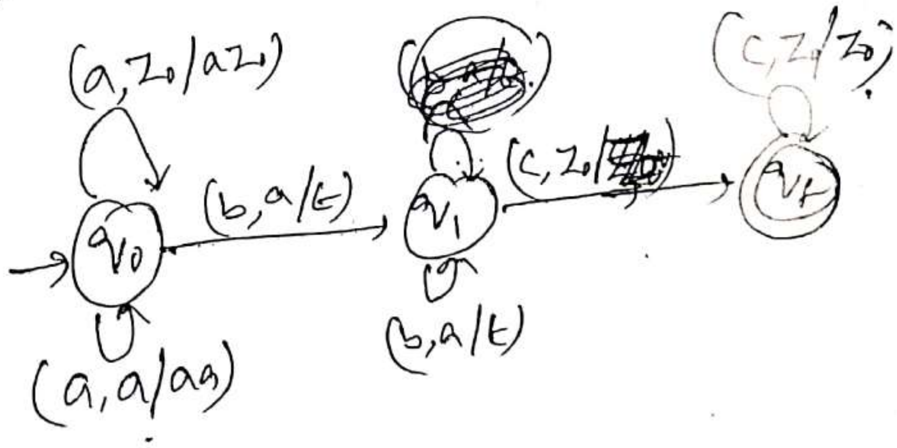
$a^n b^n c^m \mid n, m \geq 1$.

DPDA

$a^n \rightarrow$ push

$b^n \rightarrow$ pop

$c^m \rightarrow$ in final state

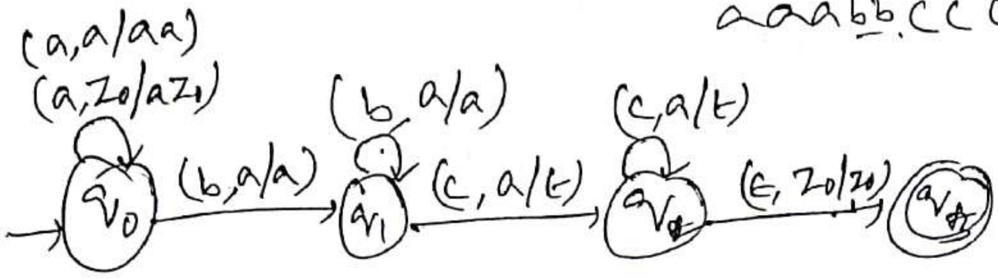


$a^n b^m c^n \mid n,m \geq 1$

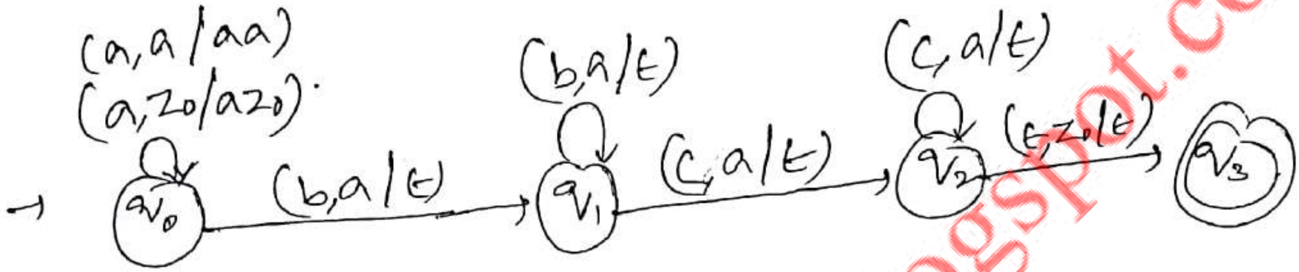
dont want count b

aaabbccc

Sol

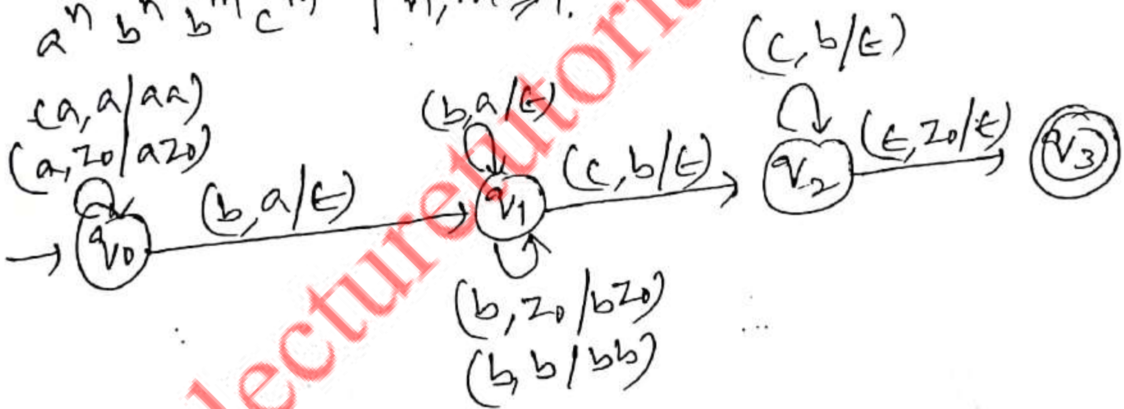


$a^{m+n} b^m c^n \mid m,n \geq 1$



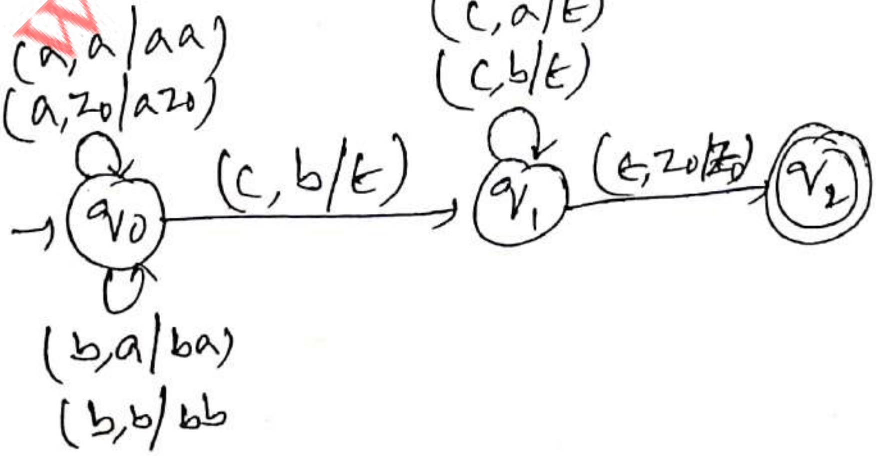
$a^n b^{m+n} c^m \mid n,m \geq 1$

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



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

Sol



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

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

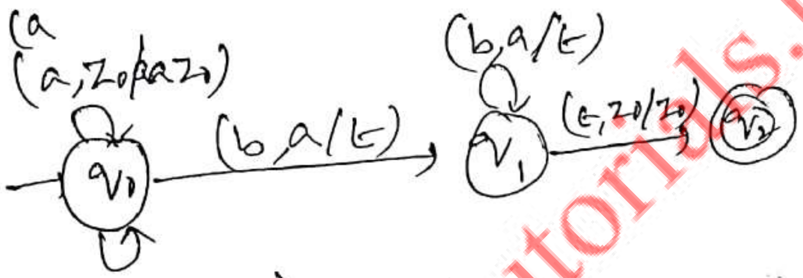
* $a^n b^m c^n d^m \mid n, m \geq 1$ ✗ is not CFG not PDA

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

1. abb, aabbbb, aaabbbbbb, ...

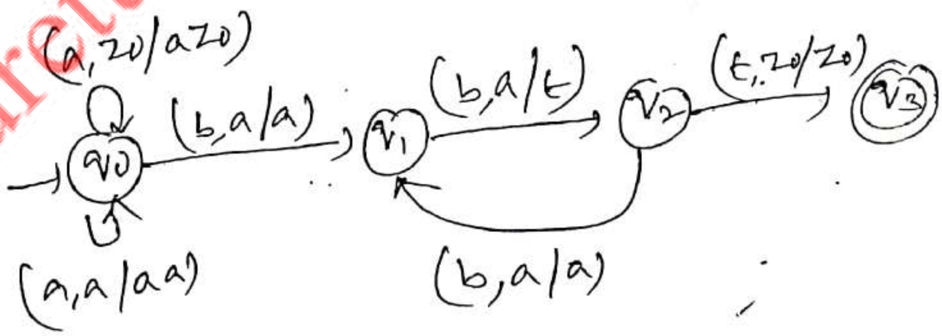
two solutions: for every a – push two a's; for every b – pop two b's

①

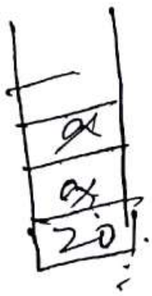


bb, bb,

for every b



aabbbbε

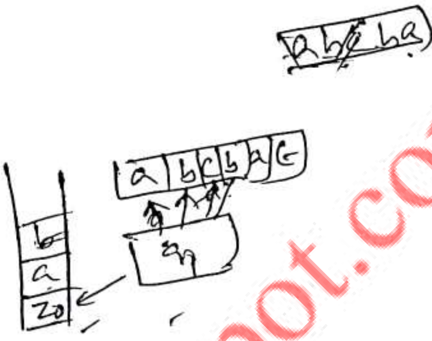


Ex:- $a^n b^n c^n \mid n \geq 1$ ✗ ~~PDA~~

One possibility

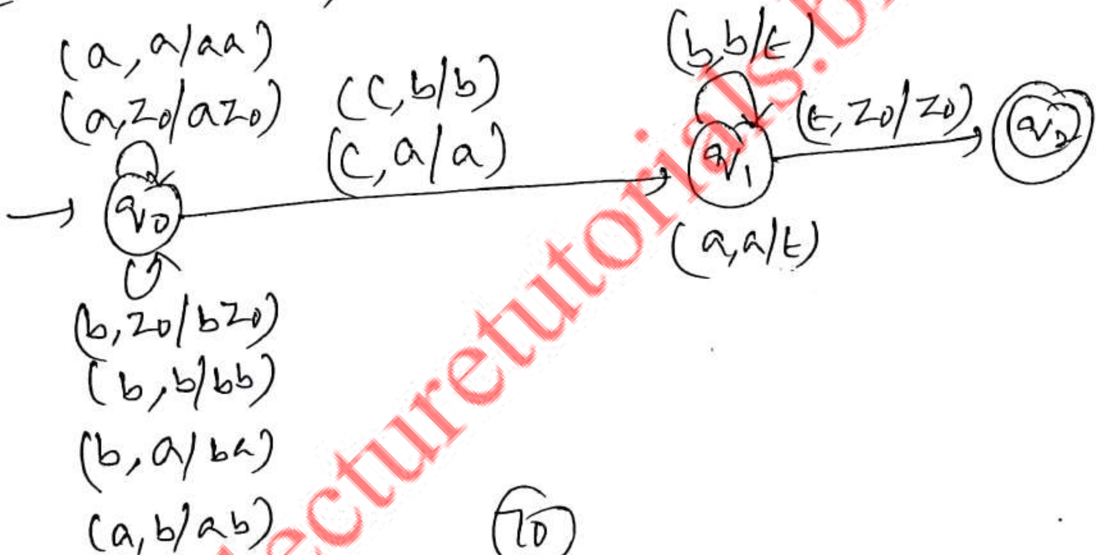
✗ aa bbcc

for every a → two a's push.



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

Sol: abcba , abbcbba

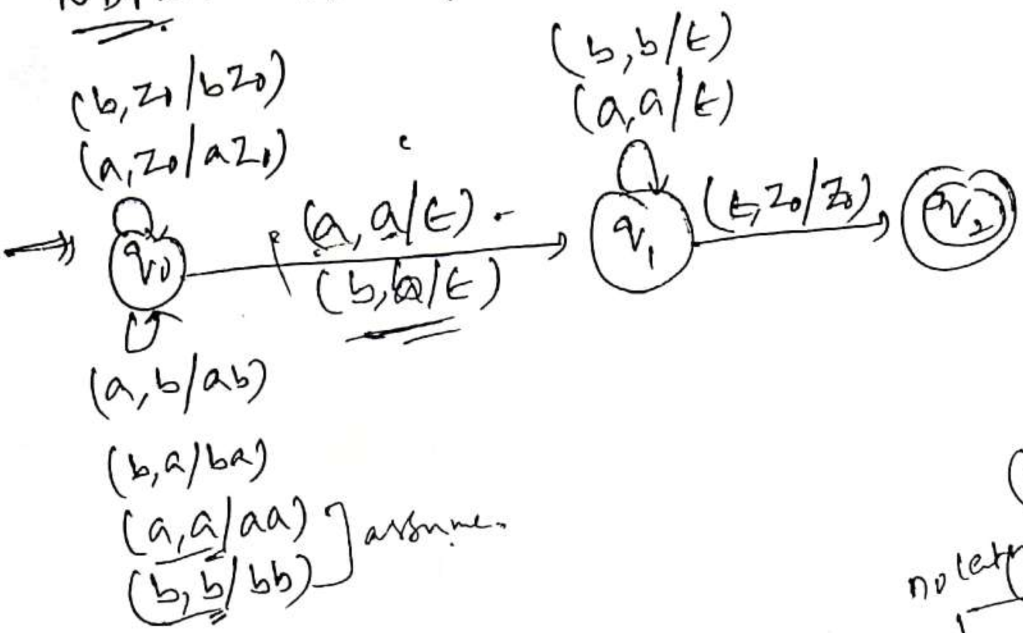


ZD

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

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

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



Instantaneous Description