3/3/18

UNIT-5

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

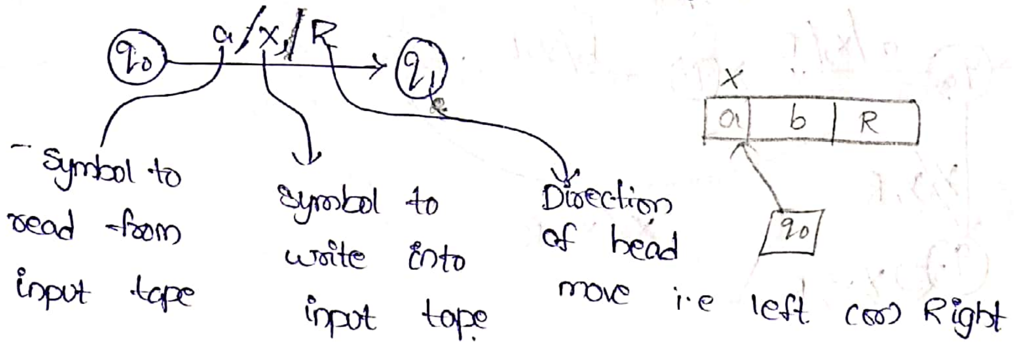
q_0 = initial state

F = final state

B = Blank symbol.

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

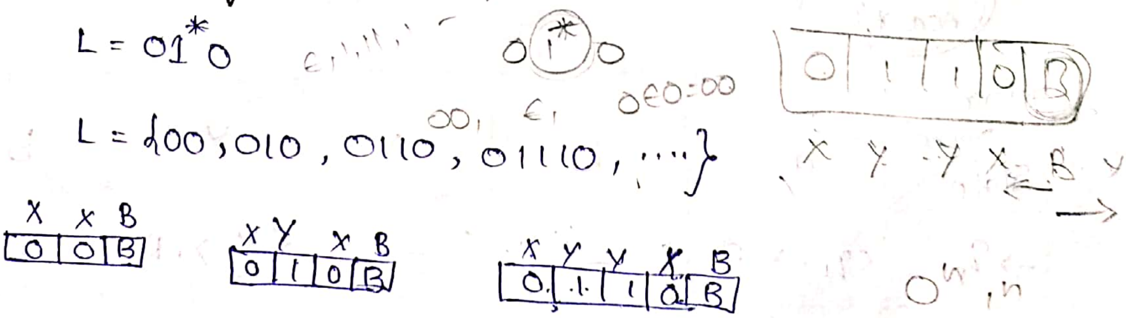
The Transition Diagram:- The transition function can be represented in the form of graphical notation.

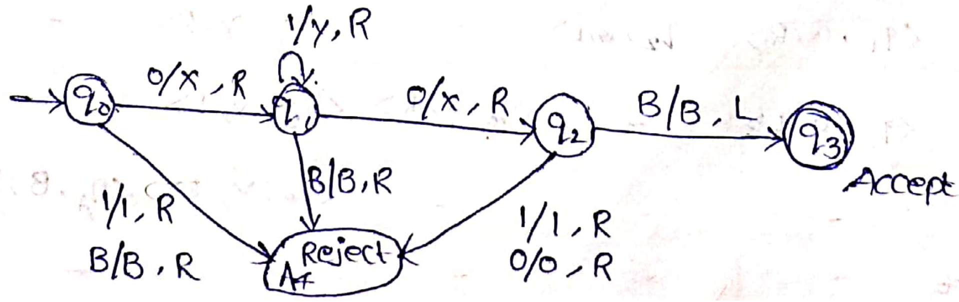


1) Design a turing machine for $L = 01^*0$

$L = 01^*0$

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

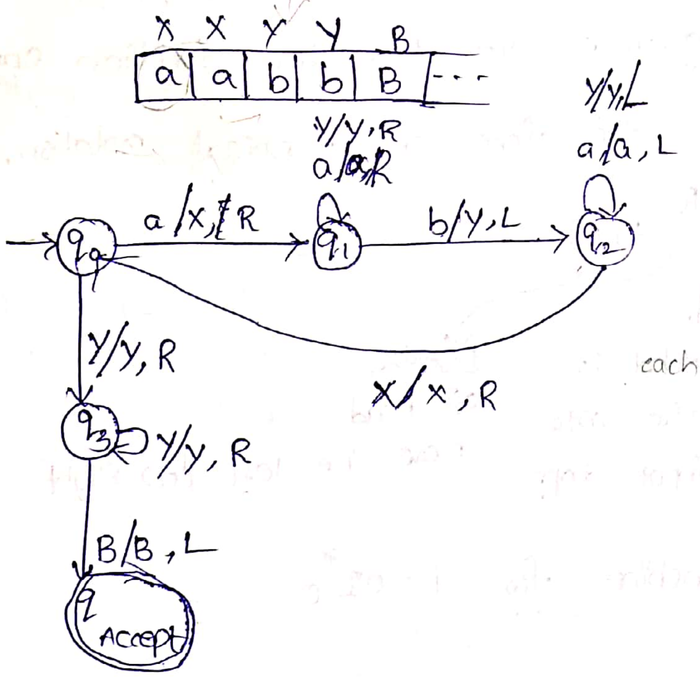


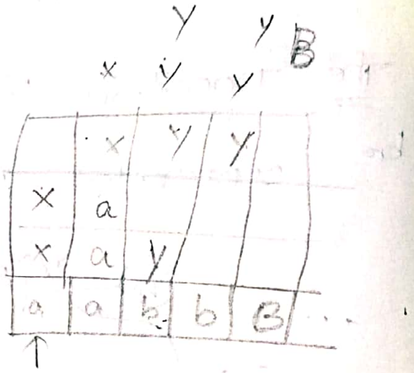


Tape Symbol / State	0	1	X	Y	B
q_0	$\langle q_1, X, R\rangle$	$\langle q_4, 1, R\rangle$	-	-	$\langle q_4, B, R\rangle$
q_1	$\langle q_2, X, R\rangle$	$\langle q_1, Y, R\rangle$	-	-	$\langle q_4, B, R\rangle$
q_2	$\langle q_4, 0, R\rangle$	$\langle q_4, 1, R\rangle$	-	-	$\langle q_3, B, L\rangle$
q_3	-	-	-	-	-
q_4	-	-	-	-	-

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
$\rightarrow q_0$	$\langle q_1, X, R\rangle$	—	—	$\langle q_3, Y, R\rangle$	—
q_1	$\langle q_1, a, R\rangle$	$\langle q_2, Y, L\rangle$	-	$\langle q_1, Y, R\rangle$	-
q_2	$\langle q_2, a, L\rangle$	-	$\langle q_0, X, R\rangle$	$\langle q_2, Y, L\rangle$	-
q_3	-	-	-	$\langle q_3, Y, R\rangle$	$\langle q_A, B, L\rangle$
* q_{Accept}	-	-	-	-	-

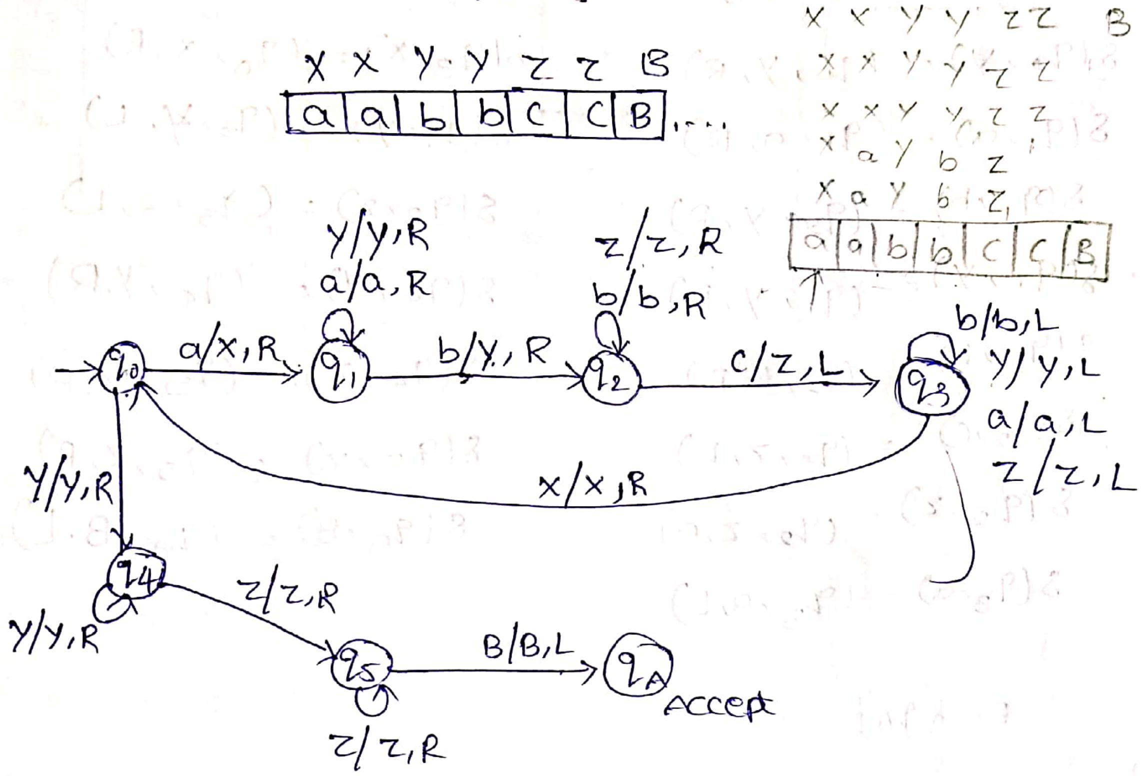
~~Transition functions~~

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
$\rightarrow q_0$	$\langle q_1, X, R\rangle$	–	–	–	$\langle q_4, Y, R\rangle$	–	–
q_1	$\langle q_1, a, R\rangle$	$\langle q_2, Y, R\rangle$	–	–	$\langle q_1, Y, R\rangle$	–	–
q_2	–	$\langle q_2, b, R\rangle$	$\langle q_3, Z, L\rangle$	–	–	$\langle q_2, Z, R\rangle$	–
q_3	$\langle q_3, a, L\rangle$	$\langle q_3, b, L\rangle$	–	$\langle q_0, X, R\rangle$	$\langle q_3, Y, L\rangle$	$\langle q_3, Z, L\rangle$	–
q_4	–	–	–	–	$\langle q_4, Y, R\rangle$	$\langle q_5, Z, R\rangle$	–
q_5	–	–	–	–	–	$\langle q_5, Z, R\rangle$	$\langle q_A, B, L\rangle$
q_A	–	–	–	–	–	–	–

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

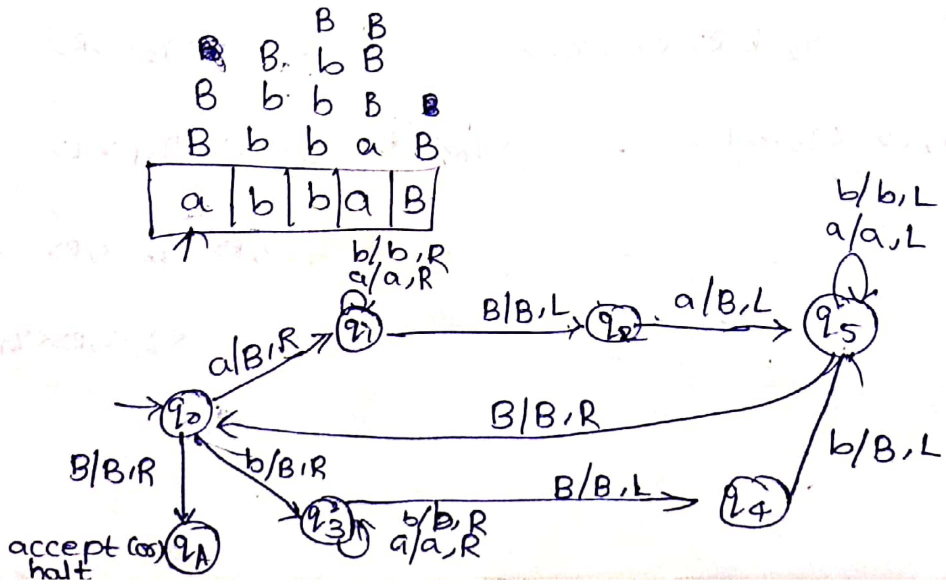
$F = \{q_A\}$

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 Symbols / 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 (on) on w

1's odd - 1

1's even - 0

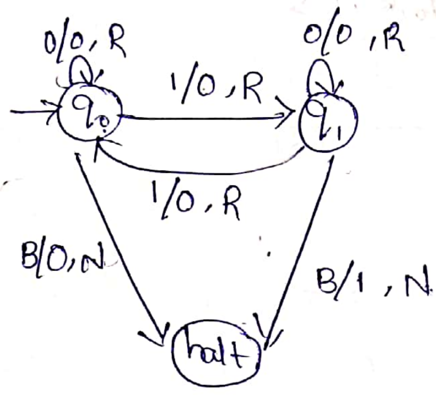
0 0 0 1

0	1	0	B

Transition Table

0 0 0 0 0 0 0

0	1	1	0	1	1	B



odd 1's 010

$\vdash q_0$ 0 1 0 B

$\vdash$ 0 q_0 1 0 B

$\vdash$ 0 0 q_1 0 B

$\vdash$ 0 0 0 q_1 B

$\vdash$ 0 0 0 1 halt

even 1's 1010

$\vdash q_0$ 1 0 1 0 B

$\vdash$ 0 q_1 0 1 0 B

$\vdash$ 0 0 q_1 1 0 B

$\vdash$ 0 0 0 q_0 0 B

$\vdash$ 0 0 0 0 q_0 B

$\vdash$ 0 0 0 0 0 halt

Design TM for 2's complement

I/P 10010

1's 01101

+1 1

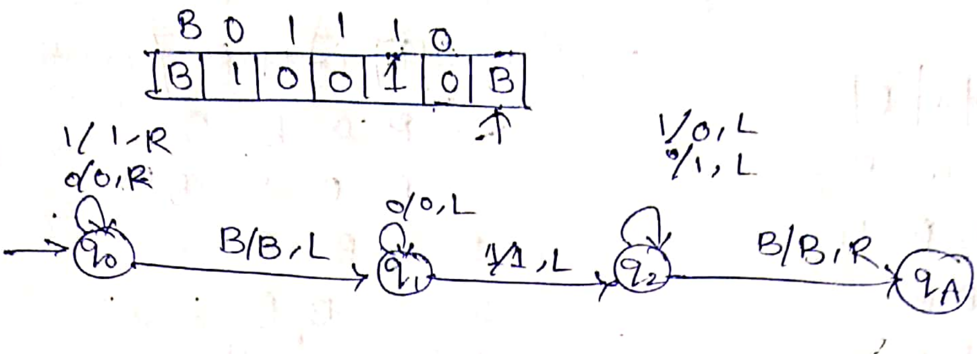
2's 01110

MSB LSB

B	1	0	0	1	0	B

accept
halt

First of all we have to move LSB then upto



ID:- B10010B

$\vdash Bq_0 10010B$

$\vdash B1q_0 0010B$

$\vdash B10q_0 010B$

$\vdash B100q_0 10B$

$\vdash B1001q_0 0B$

$\vdash B10010q_0 B$

$\vdash B1001q_1 0B$

$\vdash B100q_1 10B$

$\vdash B10q_2 010B$

$\vdash B1q_2 0110B$

$\vdash Bq_2 11110B$

$\vdash BB 01110B$

$\vdash q_2 B01110B$

$\vdash B q_2 01110B$

(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

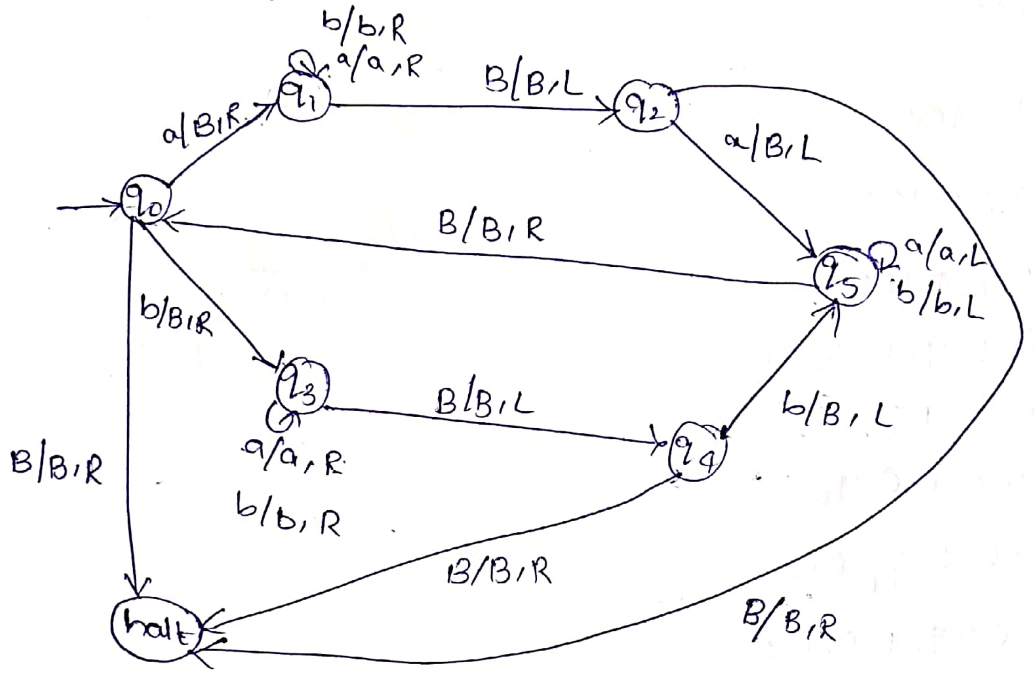
B a b B

B a B B

B B B

B B

halt



bab

$\vdash q_0 babB$

$\vdash B q_3 abB$

$\vdash B a q_3 bB$

$\vdash B a b q_3 B$

$\vdash B a q_4 b B$

$\vdash B q_5 a B B$

$\vdash q_5 B a B B$

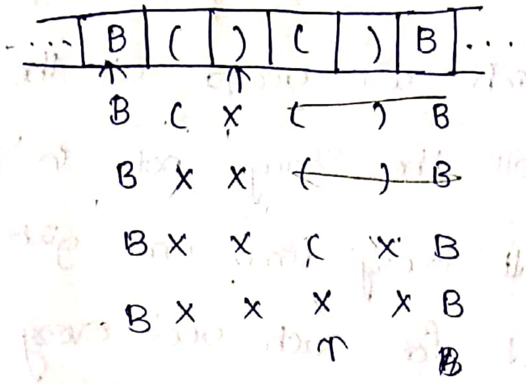
$\vdash B q_0 a b B$

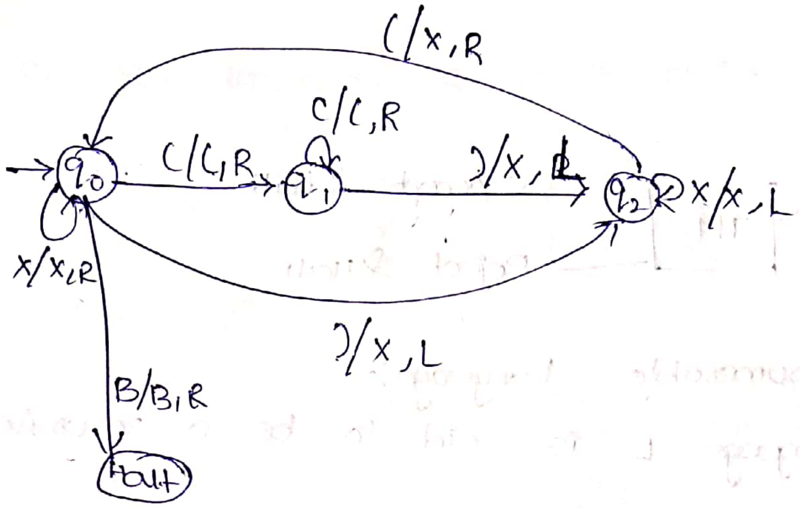
$\vdash B B q_1 B B$

$\vdash B q_2 B B B$

$\vdash B B q_A B B$

Design Turing Machine for paranthesis checking





⊢ 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 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.

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 $\alpha \to \beta$, 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:-
$aA \to abCB$
$aA \to bAA$
$bA \to a$
$S \to aAb \mid \epsilon$

ie ~~(N ∪ 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 $|\alpha| \leq |\beta|$. 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:- $|\alpha| \leq |\beta|$ $\alpha \to \beta$

$aAbcD \to abcDbcD$

(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 $|\alpha| \leq |\beta|$ 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) $S \to aS \mid Sa \mid a$

(ii) $S \to aAA \mid bBB \mid \epsilon$.

Scanned by CamScanner

(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^*

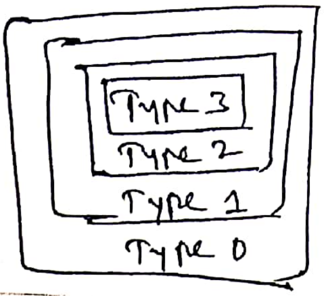
Ex: $A \to aA|a$ — right linear grammar

$A \to Aa|a$ — 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 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 $= T^*\{V\}$ $= T^*$



3

Chomsky hierarchy of grammar

Scanned by CamScanner

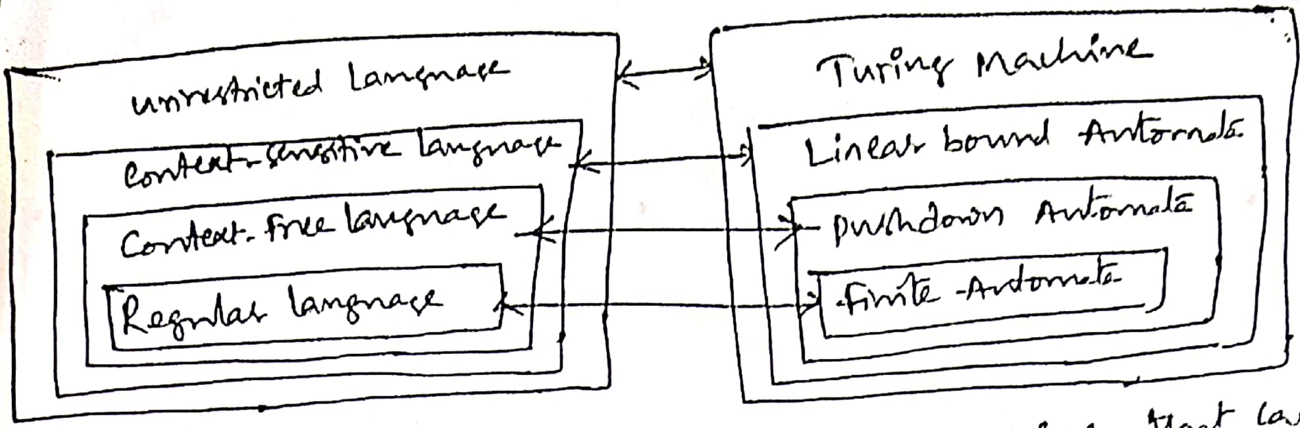


fig: The hierarchy of languages and the machine that can recognize the same is shown above fig.

- 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 push down 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

④

Scanned by CamScanner

Push Down Automata (PDA)

68

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

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

Q = finite set of states

Σ = input symbol

δ : Transition ~~function~~ function

q_0 = initial state

Z_0 = Bottom of the stack

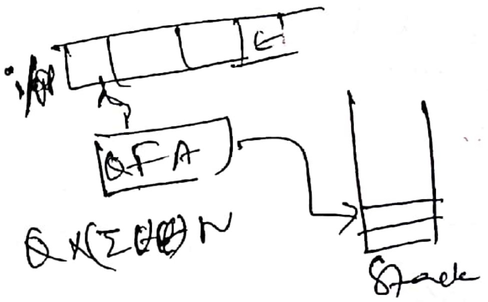
F = set of final states

Γ = ~~stack~~ stack alphabet.

PDA → DPDA, NDPDA

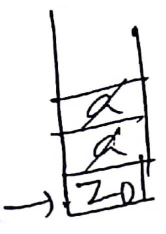
DPDA: $\delta : \underbrace{Q}_{State} \times \underbrace{\{\Sigma \cup \epsilon\}}_{Input} \times \underbrace{\Gamma}_{Top} \rightarrow \underline{Q \times \Gamma}$

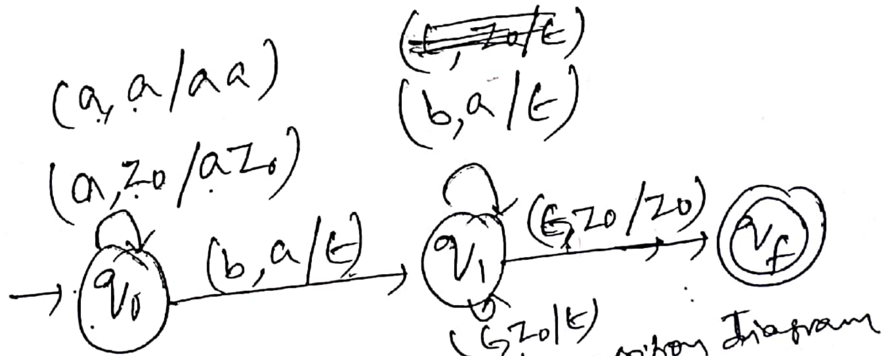
NDPDA $\delta : \underbrace{Q}_{State} \times \underbrace{\{\Sigma \cup \epsilon\}}_{input} \times \underbrace{\Gamma}_{Top\ symbol} \rightarrow 2^{(Q \times \Gamma)^*}$



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

aabb ϵ





State transition diagram

aabb ϵ

DPDA

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

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

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

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

$\delta(q_1, \epsilon, Z_0) = \underbrace{(q_f, Z_0)}_{accept\ by\ final\ state}$ or $\underbrace{(q_1, \epsilon)}_{Acceptance\ by\ empty\ stack}$

transitions functions

PDA → final state, Empty stack.

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

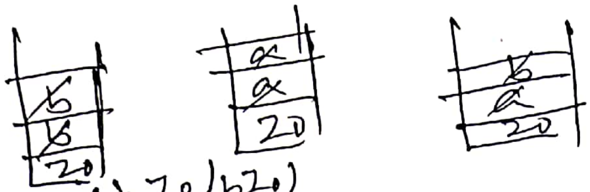
DPDA

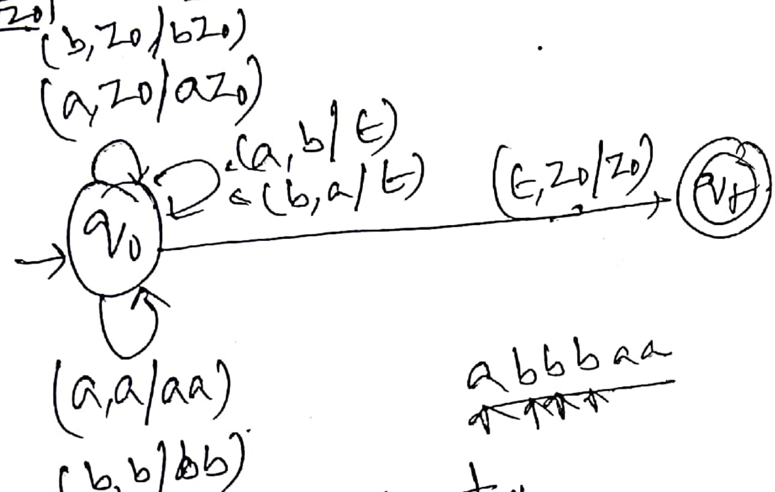
Eg: ab   bbaa

aabb   baba

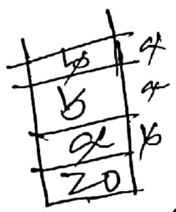
abab

baba   abab   abba





abbbaa



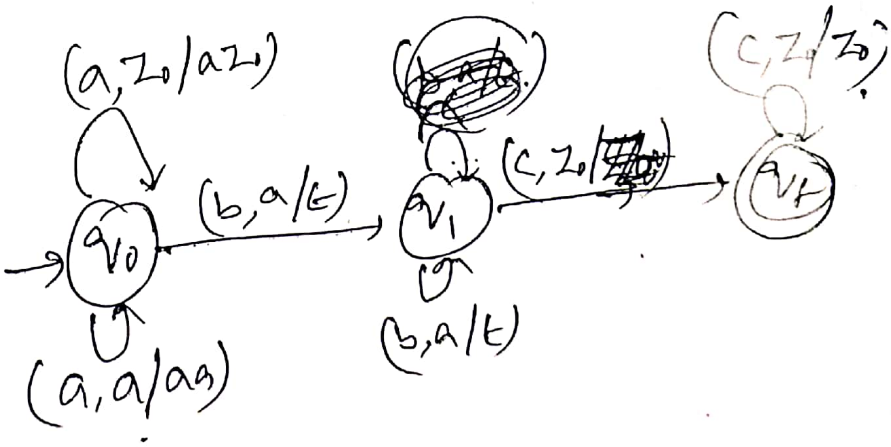
$a^n b^n c^m \mid n, m \geq 1$.

PDA

a^n → push

b^n → pop

c^m → in final state

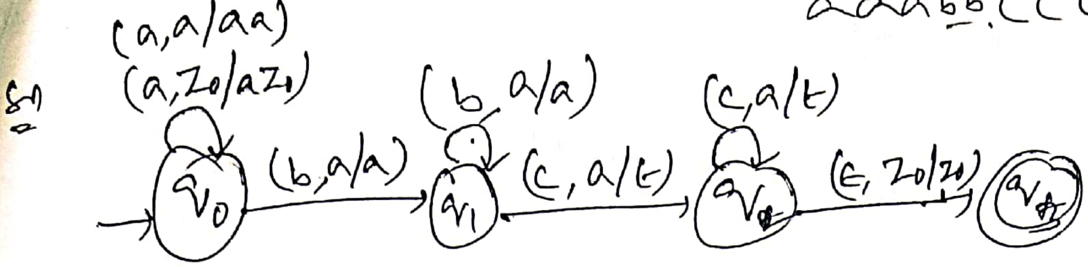


$a^n b^m c^n \mid n,m \geq 1$

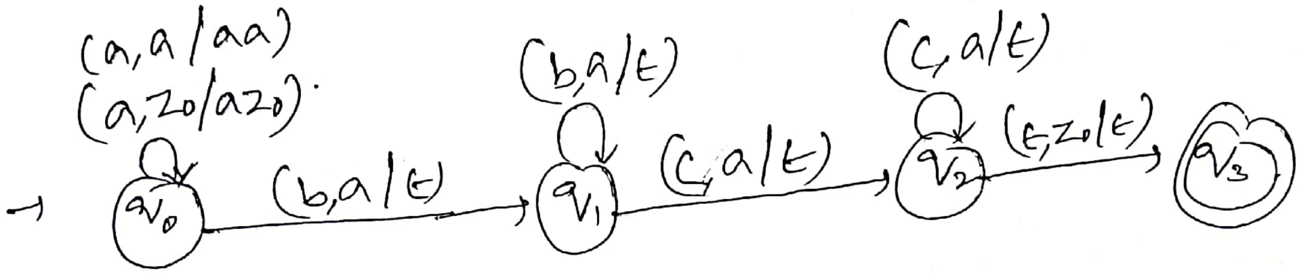
dont want cont b

aaabb.ccc

Sol

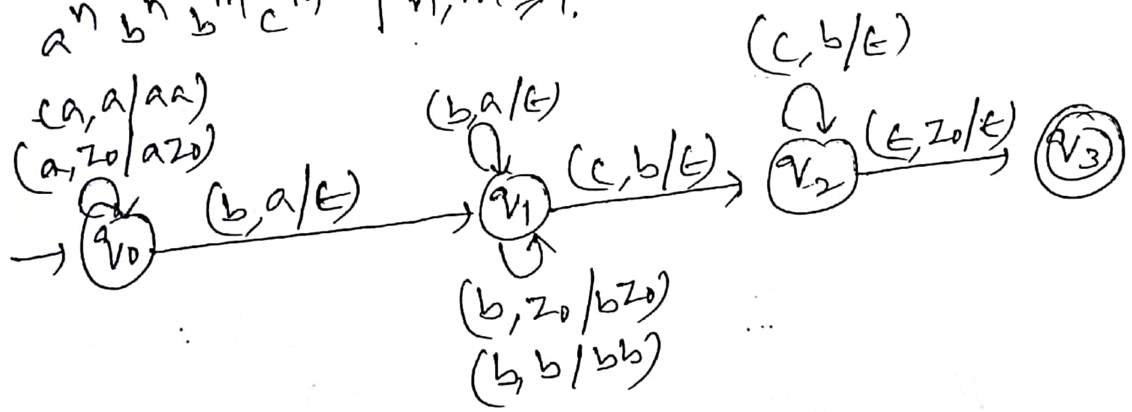


$a^{m+n} b^m c^n \mid m,n \geq 1$



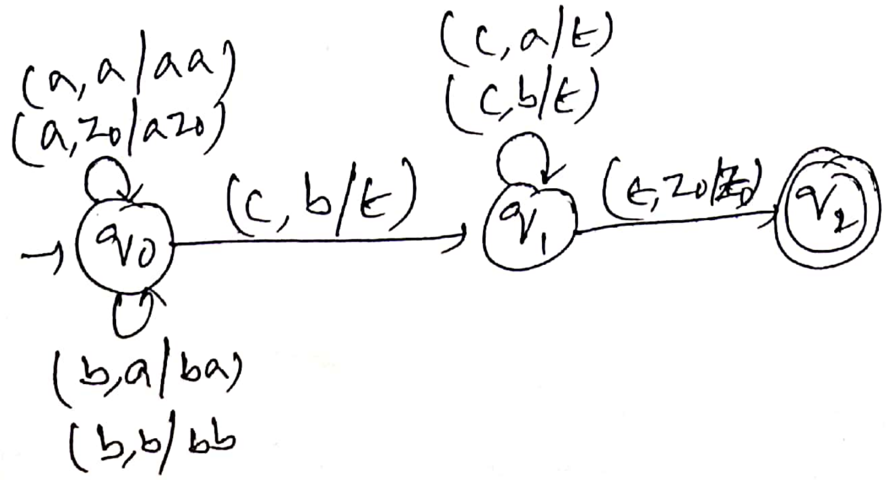
$a^n b^{m+n} c^m \mid n,m \geq 1$

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 CFL
not PDA

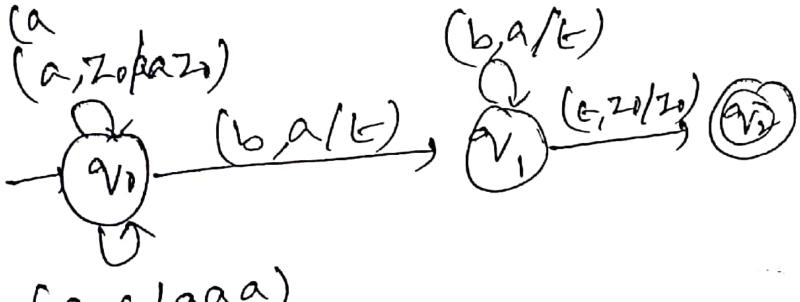
~~$a^n 2^n \mid n \geq 1$~~

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

abb, aabbbb, aaabbbbbb, . . .

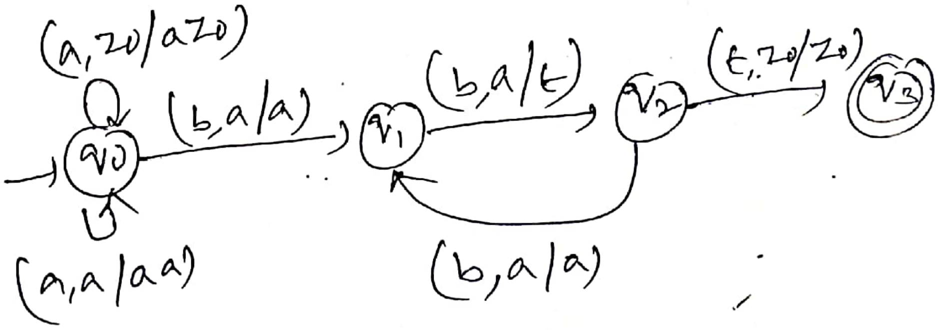
1. two solutions → for every a — push two a's.
→ for every b — pop two b's

①

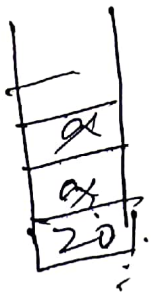


bb, bb,

for every b



aabbbb ε

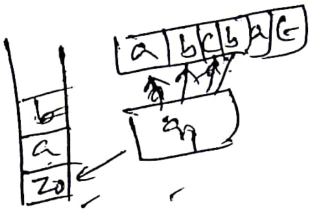


Ex:- $a^n b^n c^n \mid n \geq 1$ ✗ not a PDA

one possibility

✗ aabbcc

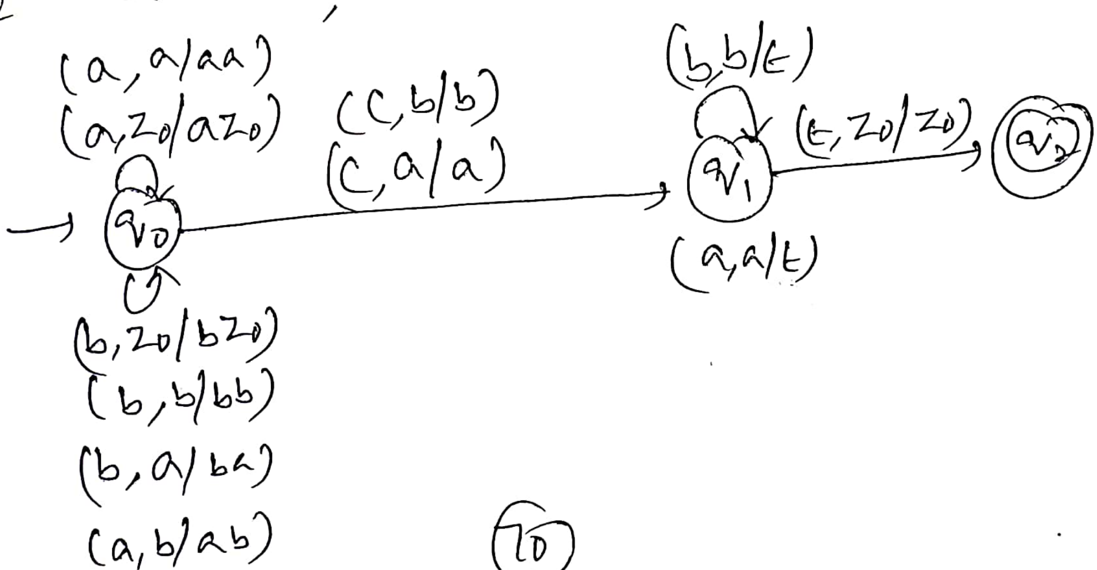
$\cancel{abb}ba)$



for every a → two a's push.

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

Sol: abcba, abbcbba

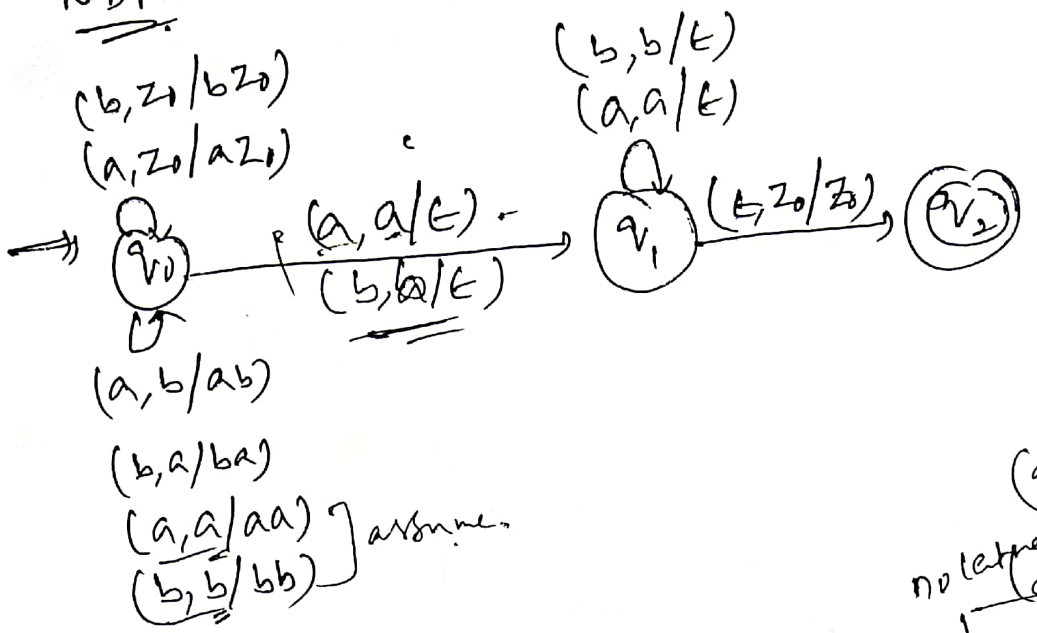


ID

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 Description

aaaa

~~Instantaneous description~~

$(q_0, aaaa, Z_0)$

(q_0, aaa, aZ_0)

no centre | centre

(q_0, aa, aaZ_0) (q_1, aa, Z_0) ×

no centre | centre

$(q_0, a, aaaZ_0)$ (q_1, a, aZ_0)

no centre | centre

$(q_0, \epsilon, aaaaZ_0)$ × (q_0, ϵ, aaZ_0) × (q_1, ϵ, Z_0)

(q_2, Z_0)

abbaab

baa aab

aab|baa

w w

aba|aba

w | w^R

Center

aa|aa

w | w^R

NDPDA

Center has come — pop

Center not come — push

Chance of getting a centre

ab|ba.

Language decidability

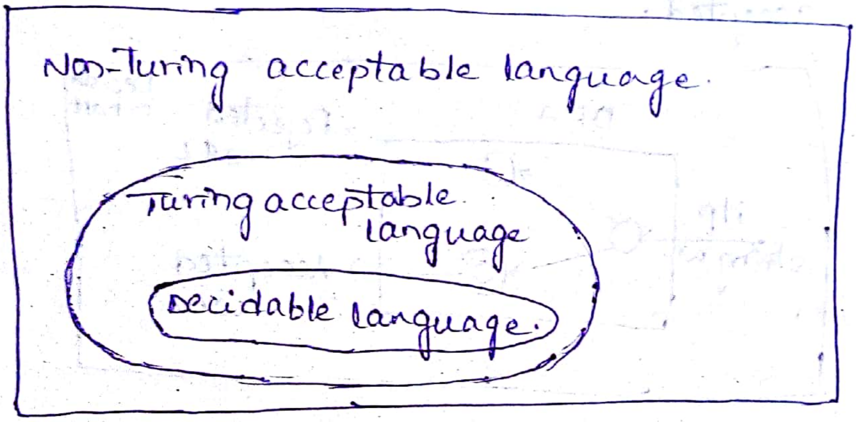
* Introduction
* Examples.

Introduction:-

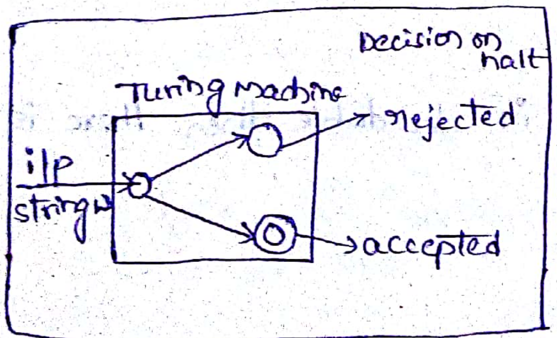
• Decidable problem:—

* 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

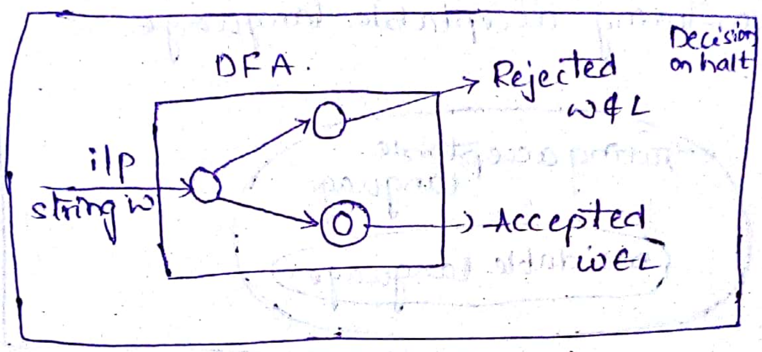
2 and m/2 · starting from 2.

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 answer Could be made by YES (or) NO.

Hence, it is a decidable problem.

2) Given a Regular language 'L' and string 'w', how can we check if $w \in L$.

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

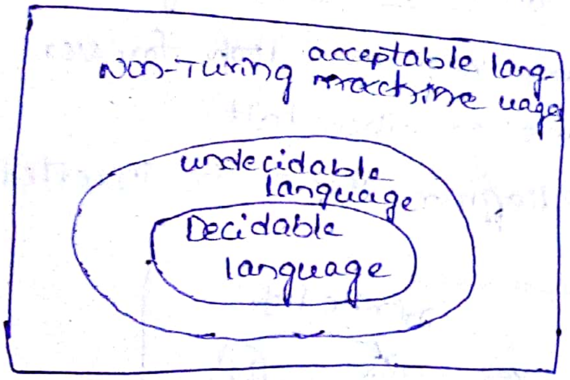
Undecidable problems:-

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, some times they may be recursive enumerable languages.



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:

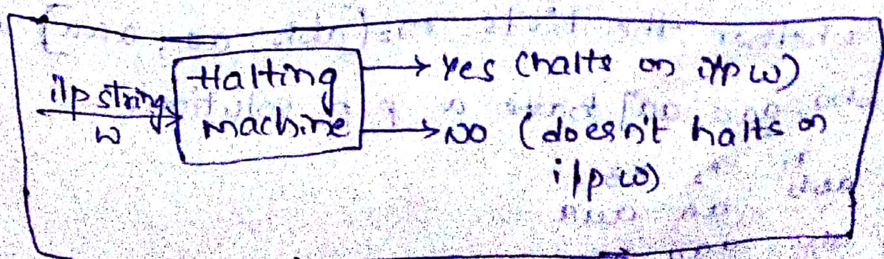
The halting problem i/p: a turing machine and the i/p string w.

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

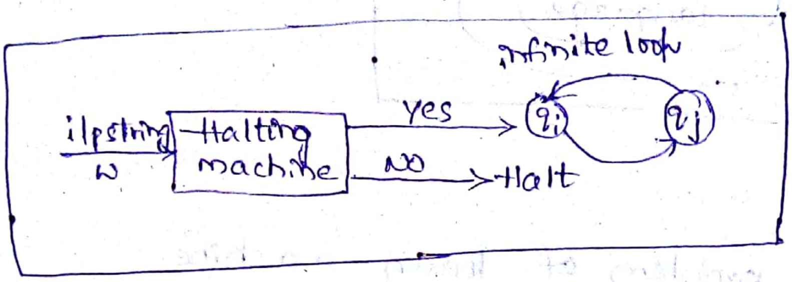


Now, we will design an inverted halting machine as.

i) If HM returns YES then loop forever.

ii) If HM returns NO then halt.

The following block diagram shows the inverted halting machine.



further, a machine "HM" which i/p itself is constructed as follows.

i) IF HM halts on i/p 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 "Emil post" in 1946 is as undecidable decision problem.

The pcp problem over an i/p 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 $1 \le i_k \le n$. 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:— 1) 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 $|x_2 x_1 x_3| \neq |y_2 y_1 y_3|$ length's 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 pairs 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 begining 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 list

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 $p(n^k)$ where '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 $p(n^k)$ 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 reduceable 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 reduceable to x and we can prove that x is in Np then x is said to be Np-complete problems.

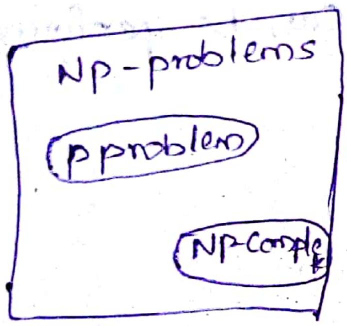
Eg:- 1) Travelling salesman problem.

2) subgraph isomorphism.

P vs NP :-

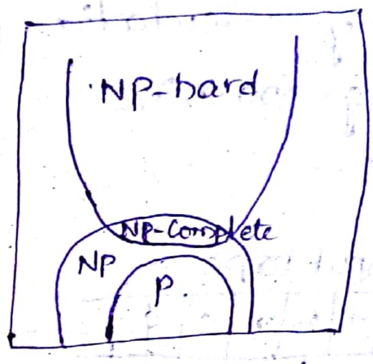
1. Ladner's theorem:

(a) P≠NP



2. Euler's theorem:-

(a) P≠NP



(b) P=NP