Vimal Singh
CSE Department

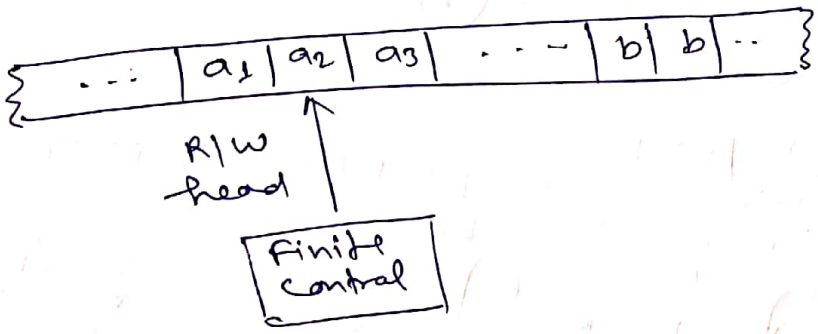
UNIT - V

Turing Machine (TM): Basic Model, definition and representation, Instantaneous Description, Language accepted by TM, Variants of TM, TM as Computer of Integer Functions, Universal TM, Church's Thesis, Recursive and Recursive Enumerable languages, Halting Problem.

Introduction to Undecidability, Undecidable problem about TM, Post Correspondence Problem (PCP), Modified PCP, Introduction to recursive function theory.

Turing Machine:- TM is a mathematical model of computation that abstract computer, was invented in '1936' by Alan Turing.

Model:- The TM can be thought of as finite control connected to a R/W head. It has one tape which is devided into a number of cells.



Tape devided into cells and of infinite length.

TM Model

TM has seven tuples.

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

where Q: Set of states
Σ: I/p alphabets
Γ: tape alphabets
B: Blank symbol
q_0: initial state $q_0 \in Q$
F: Set of final state $F \subseteq Q$

δ: Transition function

$$Q \times \Gamma \rightarrow Q \times \Gamma \times (L/R)$$

Note: TM = PDA + stack
TM = FA + 2 stack

Instantaneous Description :- of TM can be given by a triplet i.e.

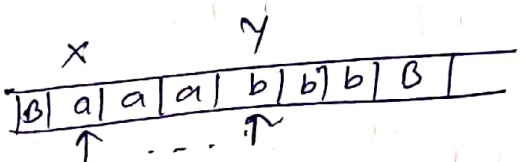
$$(q_0, a) \rightarrow (q_1, A, L)$$

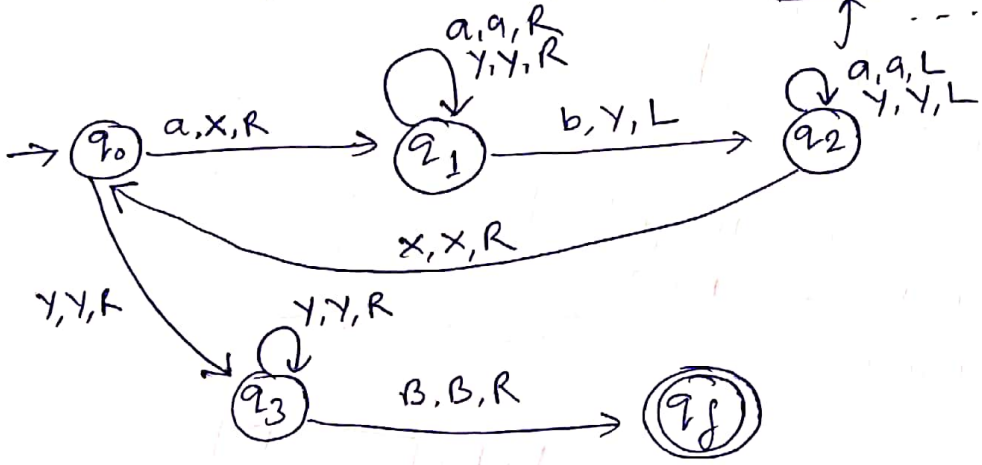
It means currently we are reading the input symbol 'a' and at state q_0 then goto state q_1 by writing 'A' at input 'a' and final move in left direction.

~~Language A~~

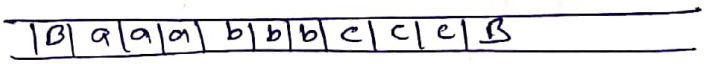
Design of TM :-

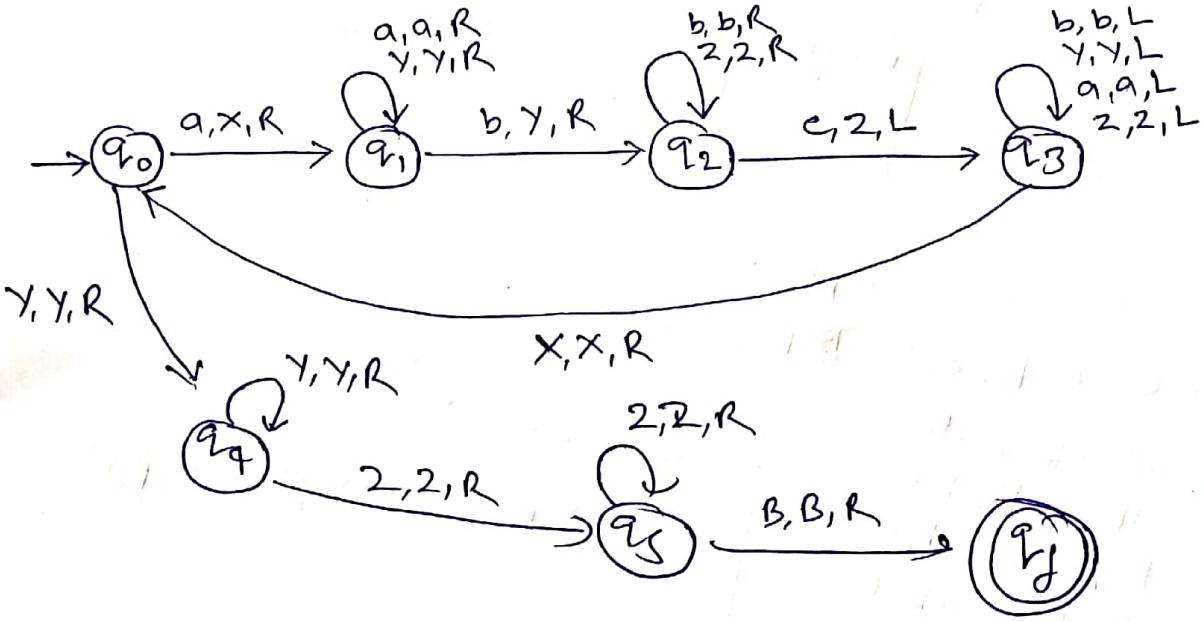
1. $L = \{a^n b^n \mid n \geq 1\}$





2. $L = \{a^n b^n c^n \mid n \geq 1\}$





Languages accepted by TM :-

Let us consider the TM $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$.
A string w in Σ^* is said to be accepted by M if $q_0 w \vdash^* \alpha_1 p \alpha_2$ for some $p \in F$ and $\alpha_1, \alpha_2 \in \Gamma^*$.

M does not accept w if the machine M either halts in a nonaccepting state or does not halt.

Variants of TM :-

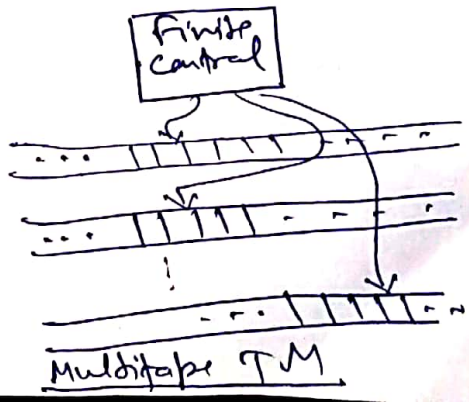
The TM we have seen has a single tape. $\delta(q,a)$ is either a single triple $(q', A, L/R)$ or is not defined. We introduce two new model of TM.

(i) **Multitape TM** :- A multitape TM has a finite set Q of states, an initial state q_0, a subset F of Q called the set of final states, a set P of tape symbols, a new symbol b, not in P called blank symbol. (here $\Sigma \subseteq \Gamma$ and $b \notin \Sigma$).

There are k tape, each devided into cells. The first tape holds the input string w. Initially all the other tapes hold the blank symbol.
Initially the head of the first tape is set the left end of the input w. All the other head can be placed at any cell initially.

δ is a partial function from $Q \times \Gamma^k \longrightarrow Q \times \Gamma^k \times (L,R,S)^k$

The typical move:
(i) M entry a new state
(ii) On each tape a new symbol is written in the cell under the head.
(iii) Each tape head moves to the Left or Right or remains stationary. (Head move independently) some to L, some R, & son S.



(ii) Nondeterministic Turing Machines :-

In case of standard TM (Deterministic TM), $\delta(q_1, a)$ defined (for some elements $Q \times \Gamma$) as an element of $Q \times \Gamma \times (L, R)$.
In nondeterministic TM, $\delta(q_1, a)$ is defined as a subset of $Q \times \Gamma \times \{L, R\}$.

A NDTM has 7-tuples $(Q, \Sigma, \Gamma, \delta, q_0, b, F)$ where

Q: set of state
Γ: set of tape symbols
b: $b \in \Gamma$ call blank symbol
Σ: input symbols, assume $b \notin \Sigma$
q_0: initial state
F: $F \subseteq Q$ set of final state
δ: δ is a partial function from $Q \times \Gamma$ into the power set of $Q \times \Gamma \times \{L, R\}$.

Turing Machine as Computer of Integer Functions :-

A basic TM is a model of studying computation. TM can solve decision problem and compute results based on inputs. When studying computation we usually restrict our attention to integers.

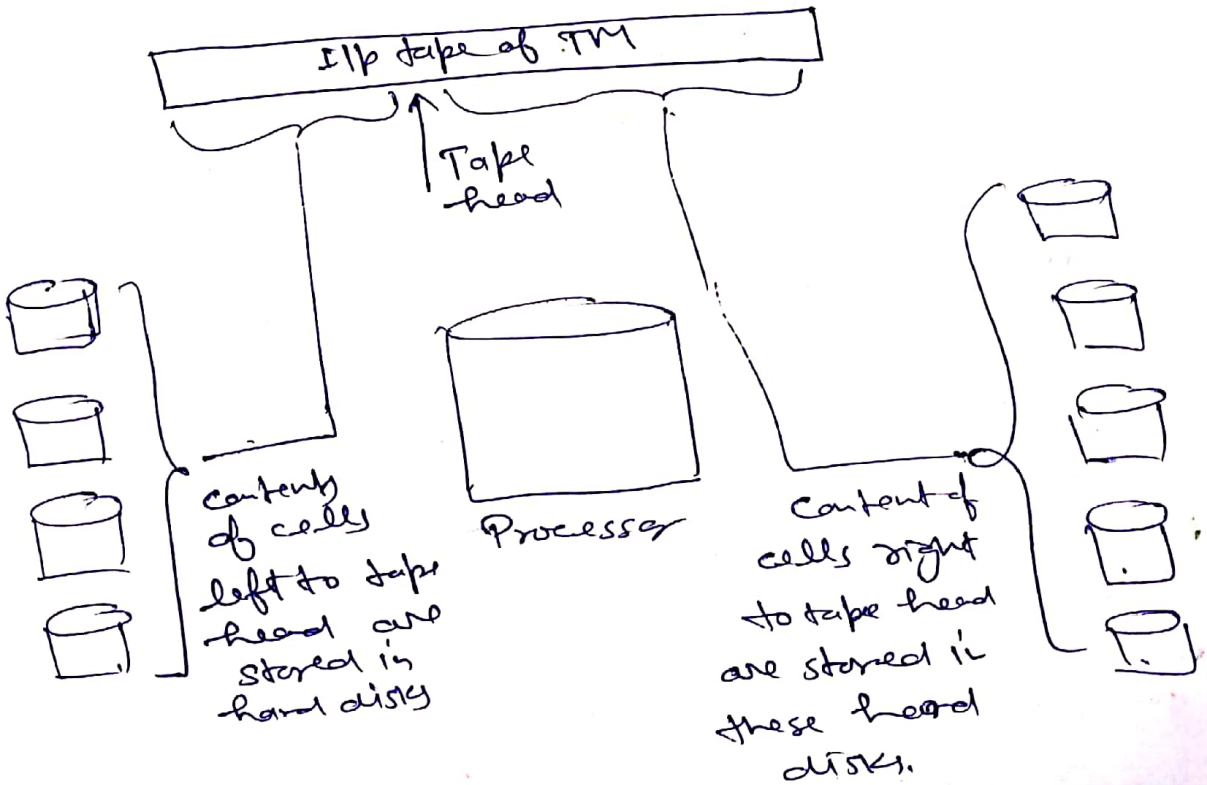
(1) Simulation of TM by computer :-

The model of TM consists of two main things - finite control and input tape.

For TM, a computer program can be written which act as a finite control. There are finite number of states and finite no. of transition rules.

For input tape of TM, storage device such as hard disk or magnetic tapes can be used.

Now the problem is how can we simulate an infinite tape with fixed amount of memory? Here we may require larger no. of memory and content need to be swapped whenever required.



TM by Computer

(ii) Simulating Compute by TM :-

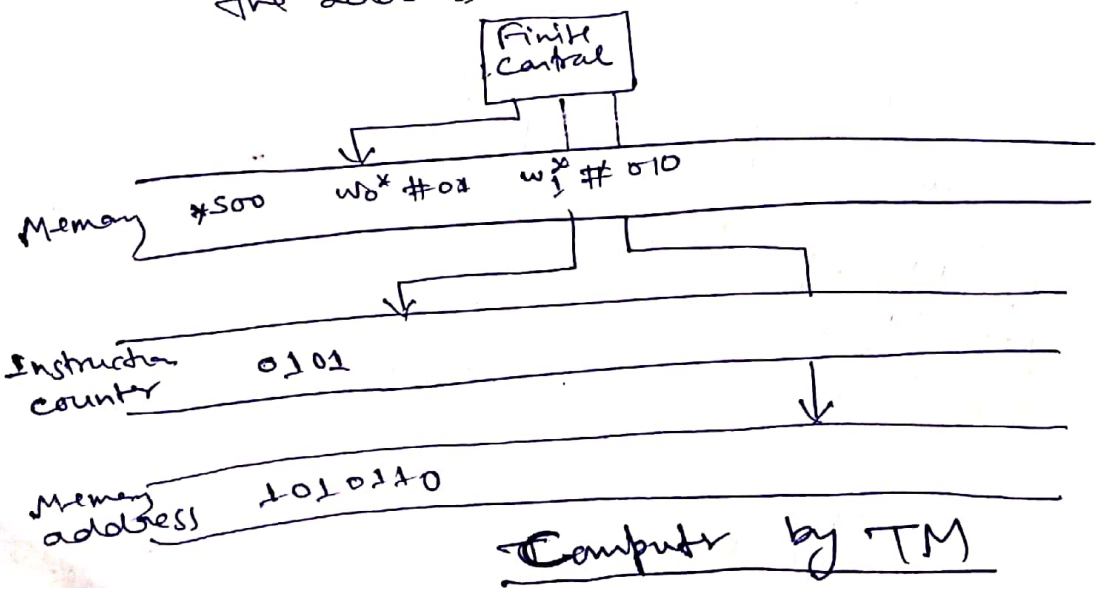
Here multitape TM is used to store the contents and address. The first tape represents the memory and on this tape contents along with their address will be stored.

The second tape represents instruction counter. This is a binary value which represent some integer. This value represent the memory locata of the next instruction to be executed.

The third tape represents memory address. Generally this address is of the contents that are stored in tape 1.

Steps for simulation :

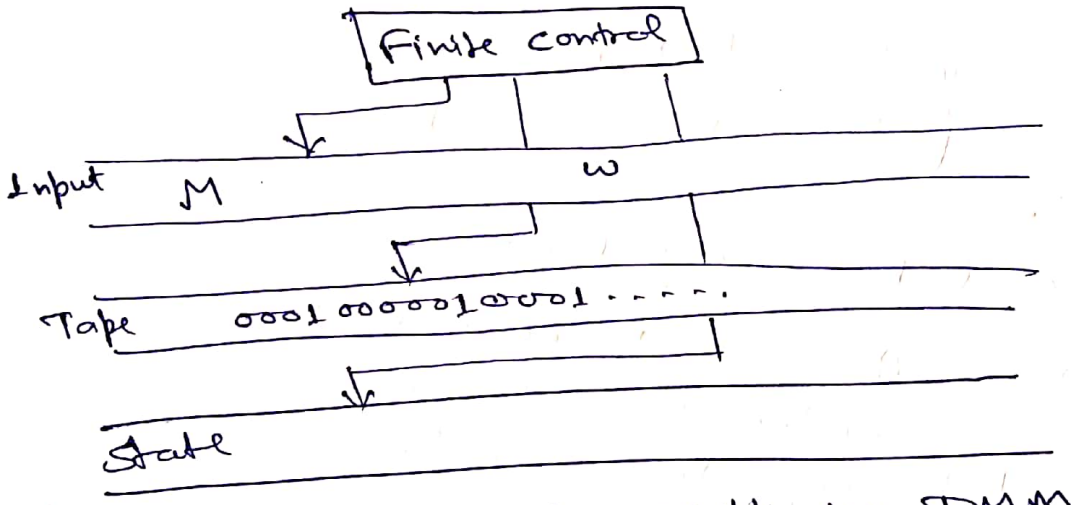
1. On tape same memory address is stored which indicate the instruction number. Read tape 1 and go on comparing each address with address stored on tape 2 , move right each time
2. When a match is found then read that instruction then read that instruction . If the instruction required the value of same address then the address will be part of instruction. Copy that address on to third tape.
3. Execute the instruction
4. We can go back to this instruction by using the address on tape 3.



Computer by TM

Universal Turing Machine :-

The universal language L_u is a set of binary strings which can be modeled by a turing machine. The universal language can be represented by a pair (M, w), where M is a TM that accept this language and w is a binary string in $(0+1)^*$ such that w belongs to $L(M)$. This we can say that any binary string belongs to a universal language. From the concept of universal turing machine came up. It can be represented by $L_u = L(U)$, where U is a universal turing machine.



(I) The universal TM U accepts the TM M.

(II) The transition of M are stored initially on first tape along with the string w.

(III) On the second tape the simulated tape of M is placed.

(IV) On the third tape various states of machine M are stored. The state q_i is represented by i 0's.

Church's Thesis :-

Church - Turing Thesis which asserts that any algorithm that can be performed on any computing machine can be performed on a Turing Machine as well.

Anything that can be done by current digital computer can also be done by a TM.

currently there is no problem which can be solve by digital computer and can not be solve by TM.

Many mathematical models are suggested but no one of them is more powerfull than the TM.

Recursive & Recursively Enumerable :-

Recursive :- A set X is recursive if we have an algorithm to determine whether a given element belongs to X or not.

Recursively Enumerable :- A RE set is a set X for which we have a procedure to determine whether a given element belongs to X or not.

It is clear that a recursive set is recursively enumerable.

→ A language $L \subseteq \Sigma^*$ is RE if there exist a TM M such that $L = T(M)$.

→ A language $L \subseteq \Sigma^*$ is recursive if there exist some TM M that satisfies the following two condition.

(i) If $w \in L$ then M accepts w (i.e reches an accepting state on processing w) and halts.

(ii) If $w \notin L$ then M eventually halts, without reaching an accepting state.

Decidability / Undecidability: (Solvable / Unsolvable Problems):-

A problem with two answers (Yes/No) is decidable if the corresponding language is recursive. And language is also called decidable.

A problem/language is undecidable if it is not decidable.

Note: A decidable problem is called a solvable problem and an undecidable problem is unsolvable problem.

Halting Problem of Turing Machine:-

The halting problem of TM is undecidable.

Halting:- means program on certain input will accept it and halt after a finite number of steps or reject it and halt after a finite no. of states. (Never goes in infinite loop)

No Halt:- Machine never reaches to halt state and remain in infinite loop.

Theorem:- $Halt_{TM} = \{(M,w) \mid$ The turing machine M halts on input w$\}$ is undecidable.

Proof: Assume $Halt_{TM}$ is decidable. (We get a contradiction)

Let M_1 be a TM such that $T(M_1) = Halt_{TM}$ and M_1 halt eventually on all (M,w), then we construct TM M as follows:

(I) for M_2, (M,w) is an input

(II) The TM M_1 acts on (M,w)

(III) If M_1 rejects (M,w) then M_2 reject (M,w)

(IV) If M_1 accepts (M,w), simulate the TM M on the input string w until M halts.

(V) If M has accepted, M_2 accepts (M,w) otherwise M_2 reject (M,w).

Post Correspondence Problem (PCP):

Emil

The PCP was first introduced by ~~Emil~~ Post in 1946. Later the problem was found to have many application in the theory of formal languages. The problem over an alphabet Σ belongs to a class of yes/no problems and is stated as follows:

Consider two lists $x = (x_1, x_2 \dots x_n)$

$y = (y_1, y_2 \dots y_n)$ of non-empty strings over an alphabet $\Sigma = (0, 1)$.

The PCP is to determine whether or not there exist $i_1, \dots, i_m$ where $1 \le i_j \le n$ such that $x_{i_1} \dots x_{i_m} = y_{i_1} \dots y_{i_m}$

Note: i_j's need not distinct, m may greater than n.

If there exist a solution to PCP, there exist infinitely many solutions.

Q: Does the PCP with two list $x = (b, bab^3, ba)$, $y = (b^3, ba, a)$ have a solution?

Sol: We need to determine whether or not there exist a sequence of substrings of x such that the string formed by this sequence and the string formed by the sequence of corresponding substring of y are identical.

The sequence is given by

$i_1 = 2, i_2 = 1, i_3 = 1, i_4 = 3$ ie $(2, 1, 1, 3)$ $\therefore m = 4$

The corresponding strings are

bab^3	b	b	ba	=	ba	b^3	b^3	a
x_2	x_1	x_1	x_3		y_2	y_1	y_1	y_3

Thus PCP has a solution.

Q. Does PCP with two list $X = (10, 011, 101)$, $Y = (101, 11, 011)$ have a solution?

Sol:- Consider a sequence

	1	3	2
X:	10	101	011

	1	3	2
Y:	101	11	011

These two string are not equal. If we try various combination for both the sets to find the unique sequence, we could not get a sequence hence there is no solution.

Modified PCP :-

The PCP may be thought of as a game of dominoes. Let each domino contain some x_i in upper half and corresponding substring of y in the lower half, domino shown as

x_i	upper-half
y_i	Lower-half

The PCP is equivalent to placing the dominoes one after another as a sequence (Repetition of dominoes allowed). To win the game, same string should appear in lower half and in the upper half.

Q.

i	List A w_i	List B x_i
1	11	111
2	100	001
3	111	11

Is there any possible solution ?

Sol. $\frac{w_1}{x_1} = \frac{11}{111}$ $\frac{w_2}{x_2} = \frac{100}{001}$ $\frac{w_3}{x_3} = \frac{111}{11}$

Now $\frac{11}{111}\ \frac{100}{001}\ \frac{111}{11}$ $\frac{w_1\ w_2\ w_3}{x_1\ x_2\ x_3}$ then is the solution.

Q. $\frac{B}{CA}$ $\frac{A}{AB}$ $\frac{CA}{A}$ $\frac{ABC}{C}$

We need to find a sequence of dominos such that the top and bottom string are the same.

Sol: $\frac{A}{AB}\ \frac{B}{CA}\ \frac{CA}{A}\ \frac{A}{AB}\ \frac{ABC}{C}$ $\equiv$ A B CA A ABC
A B CA A A BC

Hence we have a solution.

Q.

	A	B
1.	1	111
2.	10111	10
3.	10	0

① $\frac{1}{111}$ ② $\frac{10111}{10}$ ③ $\frac{10}{0}$

Now: ② ① ① ③
$\frac{10111}{10}\ \frac{1}{111}\ \frac{1}{111}\ \frac{10}{0}$

Hence we have solution.

Q.

	A	B
1.	10	101
2.	011	11
3.	101	011

① ②

Dominos are

$\frac{10}{101}$ ① $\frac{011}{11}$ ② $\frac{101}{011}$ ③

Sol:

(i) $\frac{10}{101}\ \frac{10}{101}$ further string not possible

(ii) $\frac{10}{101}\ \frac{101}{011}\ \frac{101}{011}\ \frac{10}{101}$

(iii) $\frac{10}{101}\ \frac{011}{11}$ ×

(iv) $\frac{10}{101}\ \frac{101}{011}\ \frac{101}{011}\ \frac{101}{011}$ ---- till ∞ hence no solution

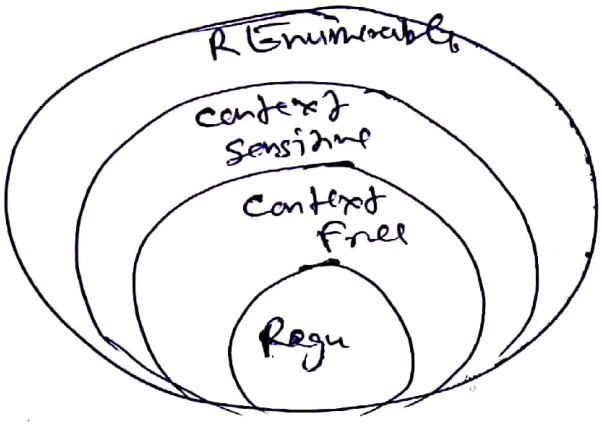
① ③

Note: PCP, there is no any specific algo to solve PCP problem hence PCP is undecidable problem (Unsolvable problem).

Generalized algo

Chomsky's Hierarchy :-

In TAFL, the chomsky's Hierarchy is a containment hierarchy of classes of formal grammars. This hierarchy of grammars was designed by Noam Chomsky in 1956.



Languag class	~~class~~ Languag	Gramma	Machine	Example
Type 3	Regural	Regular Gramma	FSM, NFA DFA	$a^* b^*$
Type 2	Context Free	CFG	PDA,	$a^n b^n$
Type 1	Context sensitive	Context Sensitive Gramma	Linear Bounded Automata	$a^n b^n c^n$
Type 0	Rec Enumerable	Unrestricted Gramma	Turing Machine	$n!$