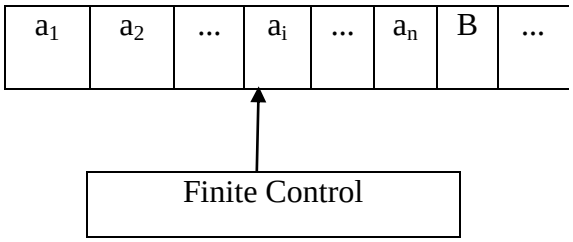
UNIT – V

TURING MACHINES

- The basic model of TM has a finite set of states, a semi-infinite tape that has a leftmost cell but is infinite to the right and a tape head that can move left and right over the tape, reading and writing symbols.
- For any input w with |w|=n, initially it is written on the n leftmost (continguous) tape cells. The infinitely many cells to the right of the input all contain a blank symbol, B whcih is a special tape symbol that is not an input symbol. The machine starts in its start state with its head scanning the leftmost symbol of the input w. Depending upon the symbol scanned by the tape head and the current state the machine makes a move which consists of the following:
    i. writes a new symbol on that tape cell,
    ii. moves its head one cell either to the left or to the right and
    iii. (possibly) enters a new state.
- The action it takes in each step is determined by a transition functions. The machine continues computing (i.e. making moves) until
    i. it decides to "accept" its input by entering a special state called accept or final state.
    ii. halts without accepting i.e. rejecting the input when there is no move defined.
- On some inputs the TM many keep on computing forever without ever accepting or rejecting the input, in which case it is said to "loop" on that input.
- A Turing Machines is an automaton whose temporary storage is a tape. This tape is divided into cells, each of which is capable of holding one symbol. Associated with the tape is a read-write, read that can travel right or left on the tape and that can read and write a single symbol on each move.
- A diagram giving an intuitive visualization of a Turing Machine is



- A Turing machine is a seven tuple notation i.e M= (Q, Σ, Γ, δ, q_0, B, F)
    1. Q is the finite set of states of the finite control.
    2. Σ is the finite set of input symbols.
    3. Γ is the set of tape symbols; Σ is a subset of Γ.
    4. δ is the transition function. It maps (Q × Γ) to subsets of (Q × Γ × {L,R}). If (p, Y, D) is in δ(q, X) and M is in state q reading the symbol X on the input tape, then M can
        - go from state q to state p,
        - replace the symbol X on the input tape by the symbol Y, and
        - move its input head one square in the direction D where D can be either L (for left) or R (for right).

       M is deterministic if there is at most one element in δ(q, X) for any state q and tape symbol X.
    5. q_0 is the start state.
    6. B is the blank symbol. B is in Γ but not in Σ.
    7. F, a subset of Q, is the set of final accepting states. We assume there are no transitions from a final state so that when M enters a final state it halts.

Instantaneous Description (ID):

- The ID (instantaneous description) of a TM capture what is going out at any moment i.e. it contains all the information to exactly capture thje "current state of the computations".
- It contains the following:
    - The current state, q

- The position of the tape head,
- The constants of the tape up to the rightmost nonblank symbol or the symbol to the left of the head, whichever is rightmost.

➢ Note that, although there is no limit on how far right the head may move and write nonblank symbols on the tape, at any finite tim, the TM has visited only a finite prefix of the infinite tape.

➢ An ID (or cinfiguration) of a TM M is denoted by $\alpha q \beta$ where $\alpha, \beta \in \Gamma^*$ and

- α is the tape contents to the left of the head
- q is the current state.
- β is the tape contents at or to the right of the tape head.

➢ That is, the tape head is currently scanning the leftmost tape symbol of β. (Note that if $\beta = \in$, then the tape head is scanning a blank symbol).

➢ If q_0 is the start state and w is the input to a TM M then the starting or initial configuration of M is onviously denoted by $q_0 w$.

Moves of Turing Machines:

➢ To indicate one move we use the symbol $\vdash$. Similarly, zero, one, or more moves will be represented by $\vdash$. A move of a TM M is defined as follows.

- Let $\alpha Z q X \beta$ be an ID of M where $X, Z \in \Gamma$, $\alpha, \beta \in \Gamma^*$ and $q \in Q$.
- Let there exists a transition $\delta(q, X) = (p, Y, L)$ of M.

➢ Then we write $\alpha Z q X \beta \vdash_M \alpha q Z Y \beta$ meaning that ID $\alpha Z q X \beta$ yields $\alpha Z q Y \beta$

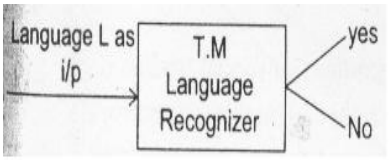
- Alternatively , if $\delta(q, X) = (p, Y, R)$ is a transition of M, then we write $\alpha Z q X \beta \vdash \alpha z Y p \beta$ which means that the ID $\alpha z q X \beta$ yields $\alpha z Y p \beta$.

Language Acceptance:

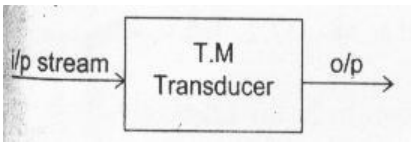
➢ L(M), the language accepted by M, is the set of strings w in Σ* such that $q_0 w \vdash^* \alpha p \beta$ for some state p in F.

Functions of Turing Machine

(1) Turing machine as a language recognizer**.**



(2) Turing Machine as a Transducer



Halt state:

It is a state from which no further transitions can be seen.

(a) Halt final state:
If at all string is accepted,T.M goes to halt final state.

(b) Halt non final state:
Turing Machine may goes to Halt non final state,if the string is rejected,when it reads invalid string.

Example:

1. Construct a TM is accept the languages $\{0^n1^n | n>1\}$

Solution:

The sample string is 0011 for n=2

This string is stored on the tape

…	B	0	0	1	1	B	…

First the finite control read the symbol '0' and replaced by 'x'. Now it will verify for one. Until encounter the symbol '1' the tape will move towards right by reading one input symbol at a time and all the zeros are replaced by 0's again. When the 1st '1' is encountered, this is replaced by symbol 'y', now again head movement is towards left side unit '1' encounter the 'x'. This process is repeated.

$\delta(q_0,B)=(q_0,B,R)$

$\delta(q_0,0)=(q_1,X,R)$

$\delta(q_1,0)=(q_1,0,R)$

$\delta(q_1,1)=(q_2,Y,L)$

$\delta(q_2,0)=(q_2,0,L)$

$\delta(q_2,X)=(q_0,X,R)$

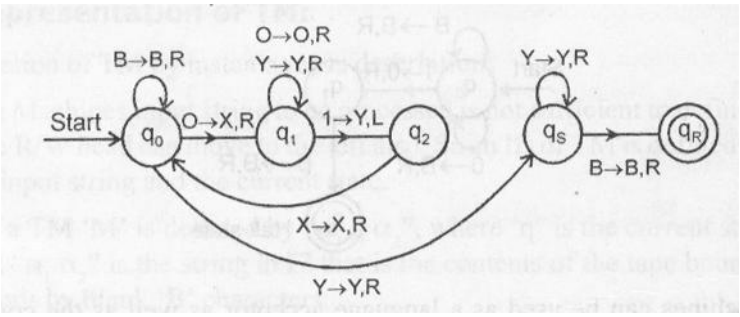
$\delta(q_1,Y)=(q_1,Y,R)$

$\delta(q_2,Y)=(q_2,Y,L)$

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

$\delta(q_3,Y)=(q_3,Y,R)$

$\delta(q_3,B)=(q_f,B,R)$



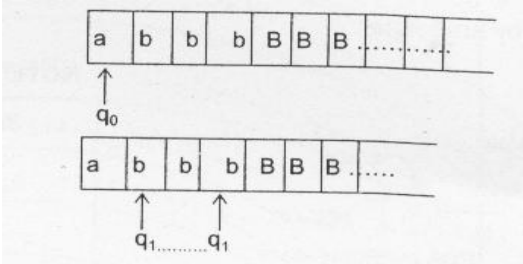
Let us consider a language

$L=\{ab^n/n\geq 0\}$

The possible strings in this language are

$L=\{a,ab,abb,\ldots\ldots\}$

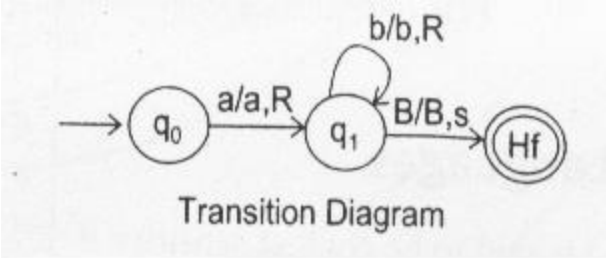
If we consider the T.M for this language



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

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

$\delta(q_1,B)=\delta(H_f,B,S)$



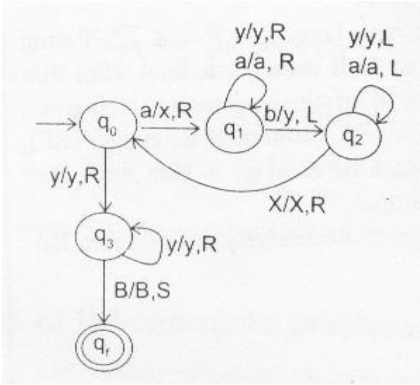
Transition Diagram

Note:

By reading an i/p symbol, Turing Machine may go to either left or right or stays at that state only.

Let us consider the language $L=a^n b^n$ where $n \geq 1$.

To construct the Turing Machine for this language,the transition diagram is



For the input string w=aaabbb

Instantaneous description is

1st

$$\begin{cases} Q_0aaabbb \rightarrow xq, aabbb \\ xaq_1abbb \\ xaaq_1bbb \\ xaq_2aybb \\ xq_2aaybb \\ q_2xaaybb \end{cases}$$

2nd

$$\begin{cases} xQ_0aaabbb \rightarrow Xxq_1, aybb \\ Xxaq_1ybb \\ xxayq_1bb \\ xxayq_1bb \\ xxaq_2yyb \\ xxq_2ayyb \\ Xq_2xayyb \end{cases}$$

3rd

$$\begin{cases} xxq_0aybb \rightarrow xxxq_1yyb & xxxq_0yyy \\ xxxyq_1yb & xxxyq_3yy \\ xxxyyq_1b & xxxyyq_3y \\ xxxyq_2yy & xxxyyyq_3 \\ xxxq_0yyy & xxxyyyq_f \\ xxxq_2yyy \\ xxq_2xyyx \end{cases}$$

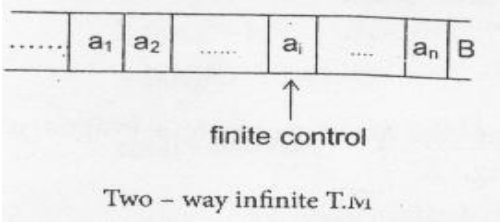
The Transaction table is

	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_f,B,S)
$*q_f$	-	-	-	-	-

2. MODIFICATIONS FOR TURING MACHINES

1. Two-way Infinite Turing Machine

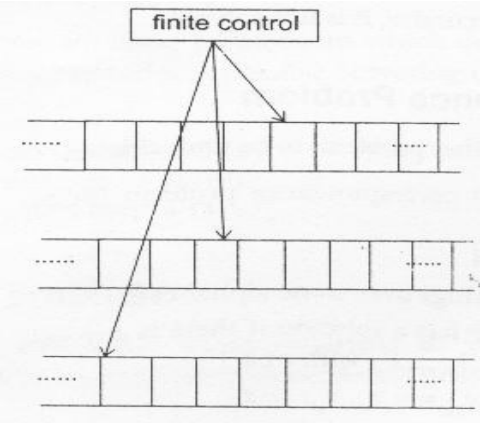
- Language L is recognized by a Turing Machine with a two-way infinite tape if and only if it is recognized by a Turing Machine with a one-way infinite tape.



Two – way infinite T.M

2. Multi-tape Turing Machine

- If a language L is accepted by a multi-tape Turing Machine,it is accepted by a single tape Machine.

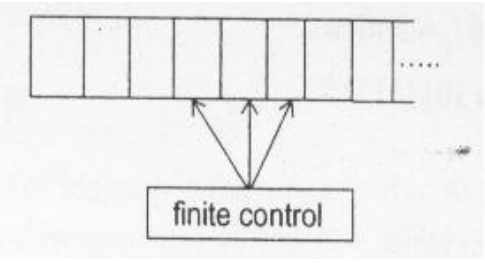


3. Non-deterministic Turing Machine

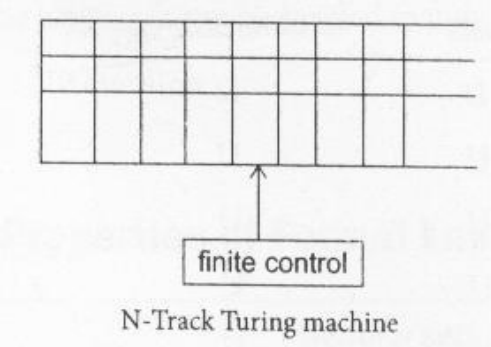
- If L is accepted by a non-deterministic Turing Machine M1,Then L is accepted by some deterministic Turing Machine,M_2.

4. Multi-head Turing Machine

- An N-head Turing Machine has some fixed number,N,of heads.The heads are numbered I through N,and a move of the Turing Machine depends on the state and on the symbol scanned by each head.



5. Multi-track Turing Machine



N-Track Turing machine

6. Turing Machine with Stay option

- In these,the read-write head can stay at the current position,upon reading on i/p symbol without moving left or right.

3. CONTEXT SENSITIVE GRAMMAR

- A grammar G=(V,T,P,S) is said to be context sensitive if all the productions are of the form

  $\alpha \rightarrow \beta$

  Where $\alpha,\beta \in (V \cup T)^+$ and $|\alpha| \leq |\beta|$

Note: The length of successive sentential forms can never decrease.

Context Sensitive Language: A language is said to be Context Sensitive if there exist a Context Sensitive grammar G, such that L=L(G).

Note: Context Sensitive grammar can never generate a language containing the empty string.

Some examples of context sensitive languages

1) $L=\{a^n\ b^n\ c^n / n \geq 1\}$
2) $L=\{a^n\ b^n\ c^{2n} / n \geq 1\}$
3) $L=\{a^n\ b^m\ c^9\ d^m / n \geq 1, m \geq 1\}$
4) $L=\{ww / w \in \{a,b\}^+\}$

Context sensitive Languages are closed under the operations

(1) ˜Ā10¯n

(2) Concatination

(3) Positive clouser

(4) $\in$-free homomorphism

(5) Inverse homomorphism

(6) Intersection with Regular sets

(7) Substitution

(8) Reversal

(9) Intersection

Note:

1. Whether Context sensitive Languages are closed under complementation or not is an open question.

- If language 'L' is a context sensitive language and 'w' is a string,we can find whether
  $w \in L$ or not algorithmetically i.e .,there is a 'membership algorithm' for context sensitive languages.

4. LINEAR BOUNDED AUTOMATION

A Linear Bounded Automation (LBA) M accepts a string w if after starting at the initial state with R/W head reading the left-end-marker, M halts over the right-end-marker in a final state.

Otherwise 'w' is rejected.

- A Linear Bounded Automation (LBA) is a nondeterministic Turning Machine(TM) satisfying the following two conditions.
  (1) Its i/p alphabet includes two special symbols $\subset and \$$,the left and right end marker respectively.

(2) The LBA has no more moves from $\subset$ or right from \$,not it may print another symbol over $\subset or \$$.An LBA will be defined as M=$\{Q, \Sigma, T, \delta, q_{0,} \subset, \$, F)$

Where $\subset$ & \$ are symbols in Σ,the left & right side markers.

Note:

(1) If L is CSL,then L is accepted by some LBA.

(2) If L=(M) for LBA, M=$(Q, \Sigma, \Gamma, \delta, q_0, \subset, \$, F)$ Then L-$\{\in\}$ is a CSL.

(3) Every CSL is a recursive but converse is not true.

5. UNDECIDABILITY

Recursive Languages:

- A language L over the alphabet Σ is called recursive language if there exist a Turing Machine M that accepts every word in L and rejects every word in L(The complement of L).

Recursively Enumerable Languages:

- A language L over the alphabet Σ is called recursively enumerable if there exist a Turing Machine T that accepts every word in L and either rejects or loops for every word in the language L.
  - In the case of recursive language $\forall x \in \Sigma^x$, Turing Machine must go to Halt state(Halt final,Halt non final) i.e.,no chance of infinite loop for any string.
  - In the case of Recursively enumerable languages (RE), $\forall x \in \Sigma^x$, Turing Machine must go to Halt final,Halt non final,Infinite loop.
  - Recursive languages $\subseteq$ recursively enumerable languages.

Note:

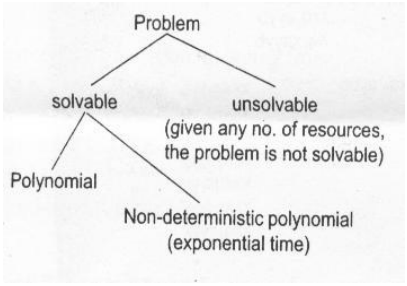
(1) Recursive languages are closed under complementation.

(2) Recursively enumerable languages are not closed under complementation.

(3) If language L is Recursively enumerable and $L^1$is also Recursively enumerable then the language L should be Recursively language.

6. UNDECIDABILITY AND NP COMPLETENESS

For any problem P if algorithm is not available,then the problem P is undecidable.



Note:

(1) If Problem is recursive, it is solvable.

(2) If Problem is not recursive, it is not solvable.

7. POST CORRESPONDENCE PROBLEM

It is a tool in establishing other problems to be undecidable.

- An instance of post correspondence problem (PCP) consists of two lists.
  $A=w_1,w_2,.....w_k$ and
  $B=x_1,x_2,.....x_k$ of strings over some alphabet Σ.
- This instance of PCP has a solution if there is any sequence of integers $i_1,i_2,....i_m$ with $m \geq 1$
  Such that $w_{i1},w_{i2},.....w_{im}==x_{i1},x_{i2},.....x_{im}$
  The sequence $i_1,i_2,....i_m$,ia a solution to this instance of PCP.
  Let us consider the following instance of PCP.

	List A	**List B**
i	W_i	X_i
1	1	111
2	10111	10

3	10	0

Let M=4, $I_1=2, i_2=1, i_3=1$ and $i_4=3$ then $w_2w_1w_1w_3=x_2x_1x_1x_3=101111110$

In the following table: D-Decidable,U-Undecidable,?-open question,T-Trivially decidable.

Question	Regular sets	DCFL's	CFL's	CSL's	Recursive sets	Recursively enumerable sets
(1) Member ship problem?	D	D	D	D	D	D
(2) Emptyness problem?	D	D	D	U	U	U
(3) Ompleteness problem is $L=\Sigma^+$?	D	D	D	U	U	U
(4) Equality Problem?	D	?	U	U	U	U
(5) Subset problem is $L_1 \subseteq L_2$?	D	U	U	U	U	U
(6) Is L Regular?	T	D	U	U	U	U
(7) Is the intersection of 2 languages, a lang,of the same type	T	U	U	T	T	T
(8) Is the complement of a lang,also a lang of the same type	T	T	U	?	T	U
(9) If L is finite or infinite	D	D	D	U	U	U

8. P-class, NP-class, NP-Hard, NP-complete

P-class problem

P class problems are those problems to which deterministic polynomial time algorithm is possible (covering every possibility).

Example: Linear search $\rightarrow$ O(n)

Bubble sort $\rightarrow$ O (n^2)

Selection sort $\rightarrow$ O (n^2)

NP-class problem

NP class problems are those problems to which non-deterministic polynomial time algorithm is possible.

NP-Hard problem

If there is a language L such that every language L is NP,can be polynomially reducible to L and we can't prove that L is in NP,then L is said to be **np-hard problem.** i.e.,'L' is polynomially reduced to all NP problems.

NP-complete problem

If we can prove that L is in NP and everyNP problem can be polynomially reducible to L then L is said to be NP-Complete problem.

P verse NP

NP is the class of languages that are solvable in polynomial time on a non-deterministic Turing Machine.

(or)

Equivalently,it is the class of language where the membership in the languages can be verified in polynomial time.

$\Rightarrow$The P=NP question.

We are unable to prove the existence of single language in NP that is not in P.

$\Rightarrow$P=NP is one of the greatest unsolved problem.

(1) If P_1 is NP-complete and there is a polynomial-time reduction of P_1toP_2 then P_2 is NP-complete.

(2) If some NP-complete problem is in P then P=NP.

Examples of NP-complete problems

(1) Boolean satisfiability problem(B-SAT) is NP complete.

(2) C-SAT,3-SAT

(3) Travelling sales man problem.

(4) Vertex cover problem.

(5) Chromatic number problem.

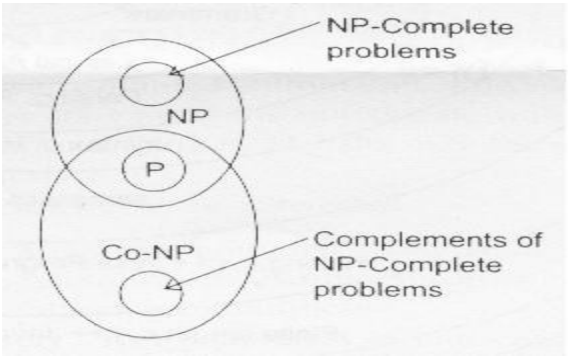
(6) The partition problem.

(7) A k-clique in a graph G.

(8) The edge cover problem.

Note:

(1) P is closed under complementation,but it is not known whether NP is closed under complementation.



(2) If P=NP then P,NP & complement NP all are same.

(3) NP=Co-NP iff there is some NP-complete problem whose complement is in NP.

9. Closure Properties of Formal Languages

	Regular sets	DCFL'S	CFL'S	CSL'S	Recursive sets	Recursively enumerable sets
1. Union	Y	N	Y	Y	Y	Y
2. Concatenation	Y	N	Y	Y	Y	Y
3. Kleene closure	Y	N	Y	N	N	Y

4. Intersection	Y	N	N	Y	Y	Y
5. Complementation	Y	Y	N	?	Y	N
6. Homomorphism	Y	N	Y	N	N	Y
7. Inverse Homomorphism	Y	Y	Y	Y	Y	Y
8. Reversal	Y	N	Y	Y	Y	Y
9. Substitution	Y	N	Y	Y	N	Y
10. Intersection with regular sets	Y	Y	Y	Y	Y	Y