Decidable and Undecidable language

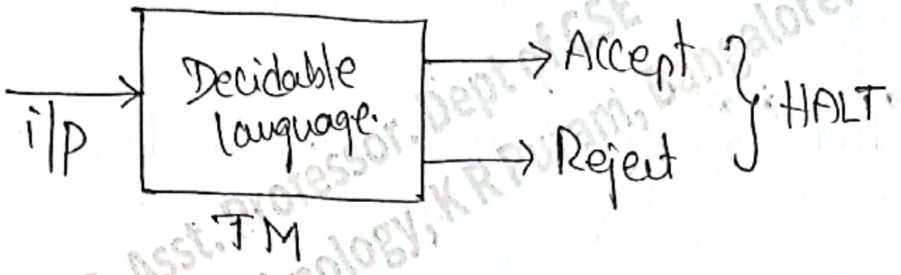
Decidability, is the process which helps us to know whether a TM halts (or) not.

- As an algorithm halts, eventually the TM also halts.
- The TM halts in following conditions:
  a) when TM reaches final state it halts.
  b) when TM reaches a state 'q' when input symbol scanned is 'a' and if $\delta(q,a)$ is not defined it halts.
- There are some TM which will not halt on an input string in any of the above situation.
- Hence we need to make a distinction between language accepted by TM and that are not.
- This is referred to as Decidability.

Decidable and Undecidable languages

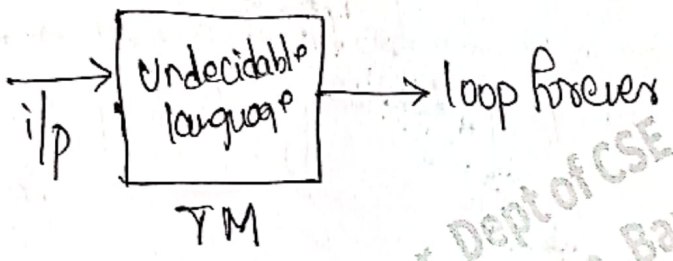
If a language is recursive then it is called as decidable languages.

- For all decidable languages, TM Halts with accept | reject output.



If a language is not recursive then it is called as undecidable language.

- For an undecidable language, TM does not Halt. Infact there is no TM.



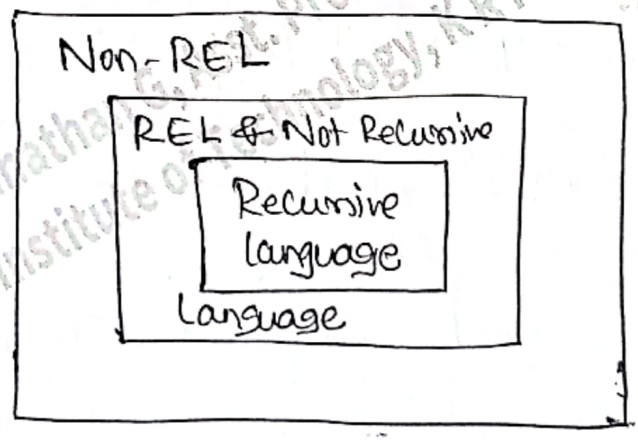


fig: Relation between languages.

Post Correspondence Problem

- Undecidability of strings is determined with the help of Post Correspondence Problem (PCP).

The post correspondence problem consists of two lists of strings that are of equal length over the input Σ.

If the two lists are

$A = w_1, w_2, w_3, \ldots w_n$ and

$B = x_1, x_2, x_3, \ldots x_n$

then there exists a non empty set of integers $i_1, i_2, i_3, \ldots i_n$ such that

$$w_1, w_2, w_3 \ldots w_n = x_1, x_2, x_3 \ldots, x_n$$

- To solve the PCP we try all the combinations of $i_1, i_2, i_3, \ldots i_n$ to find the

$$w_i = x_i$$

then we say that PCP has a solution.

Ex: Consider the correspondence system as follows:

$A = (1; 0; 010; 11)$ $B = \{10; 10; 01; 1)$

over $\Sigma = \{0, 1\}$

The solution for the above system is following sequence:

1; 2; 1; 3; 3; 4

ie., $w_1 w_2 w_1 w_3 w_3 w_4$ & $x_1 x_2 x_1 x_3 x_3 x_4$

ie., 10101001011

Quantum Computers

- A classical computer has a memory made up of bits.
  ie., either a '0' or '1'.

The Quantum computers maintain a sequence of qubits.

- The quantum bit or simply qubit can be mathematically described as:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

- Two possible states for a qubit are the states: $|0\rangle$ and $|1\rangle$.
  Here, notation "$|\ \rangle$" represents qubit.

α and β are complex numbers such that

$$|\alpha|^2 + |\beta|^2 = 1$$

The '0' and '1' are called the Computational basis states.

$|\psi\rangle$ is called Superposition.

- The notation, $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ is quantum state.

- It is possible to define multiple qubits.

  Ex: two qubit system has four basis states

  $|00\rangle, |01\rangle, |10\rangle, |11\rangle$

* The quantum states can be: (for two qubit)

$$|\psi| = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$$

$$|\alpha_{00}|^2 + |\alpha_{01}|^2 + |\alpha_{10}|^2 + |\alpha_{11}|^2 = 1$$

- Logic gates also can be defined using qubit.

  Ex: NOT gate: $\alpha|0\rangle + \beta|1\rangle \rightarrow \alpha|1\rangle + \beta|0\rangle$

* action of not gate can be described using:

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix}$$

A quantum computer can be defined as:

A system built from quantum circuits, containing wires and elementary quantum gates, to carry out manipulation of quantum information.

The Class of P and NP

- Problems can be classified into two groups:
  * Problems that can be solved in polynomial time.
  * Problems that can be solved in non-deterministic polynomial time.

- Problems solvable in polynomial time include searching of element, sorting of element etc.
- Problems solvable in Non-deterministic ~~solva~~ Polynomial time includes knapsack problem, Travelling salesperson problem etc.

- The problems that can be solved in polynomial time is refered to as Class P problems.
- Whereas the problems that are solved in non-deterministic polynomial time is refered as Class NP problems.

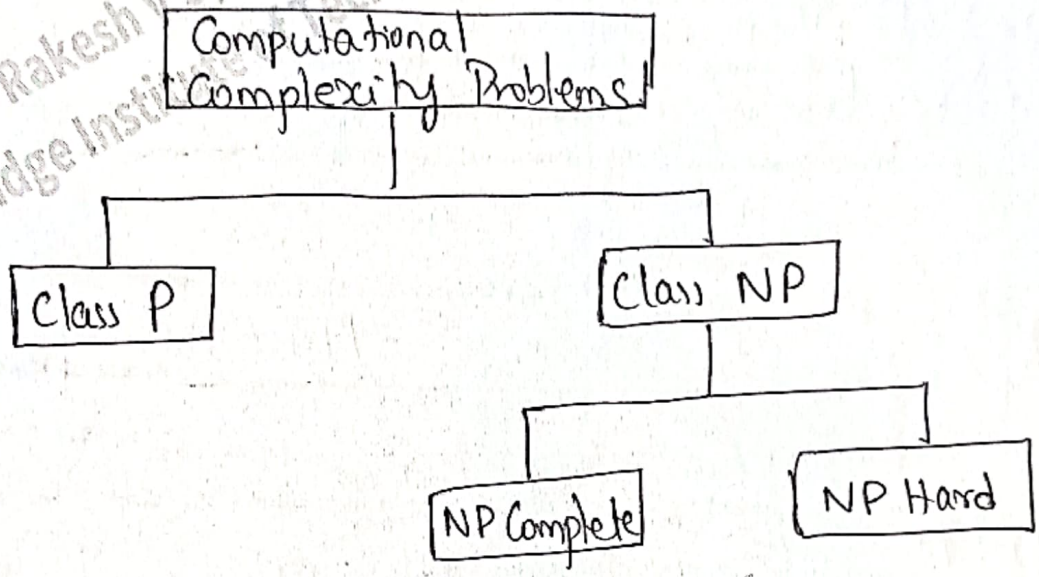


fig: Complexity Classes.

Note: All NP-Complete problems are NP-Hard but not viceversa.

- In the Computational Complexity, if the solution to the problem is "Yes" or "No" then it is called as decision problem.

Ex: Class P problems.

- If the solution exits for every input, then such type of problems are function problem.

The Church - Turing Thesis

- Alan turing proposed logical Computing Machines (LCMs), ie., Turing's expression for turing machines.
- This was made to define the algorithm properly.
- So, Church made a mechanical method named as 'M' for manipulation of strings by using logic and mathematics.
- This method 'M' must pass the following statements:
  - * No of instructions in M must be finite.
  - * Output should be produced after performing finite number of steps.
  - * It should not be imaginary.
  - * It should not require any complex understanding.

- Using these statements, Church proposed a hypothesis called Church's Turing thesis.

It states that, "The assumption that the intuitive notion of computable functions can be identified with partial recursive functions."

- However, this hypothesis cannot be proved.
- The computability of recursive functions is based on following assumptions:
  * Each elementary function is computable.
  * Let 'f' be the computable function and 'g' be another function obtained by applying elementary operation to 'f', then 'g' becomes computable function.
  * Any function obtained by above two rules are computable.

Linear Bounded Automata

- Linear bounded automaton is restricted form of non-deterministic Turing machine.
- It is originally developed as model for actual Computers.

LBA, is a multitrack Turing machine which has only one tape and it of length exactly same as input.

- Acceptance of string is same as that of Turing machine.
- In case of LBA, halting means accepting.
- the strength of LBA is as follows:

$$NPDA < LBA < TM$$

LBA model

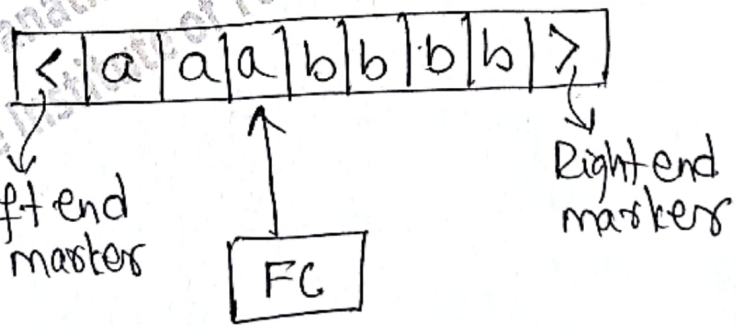


fig: LBA

- The input placed in the input tape is between the end markers
  ie, "< >"

Formal Definition

- It is a 7-tuple Nondeterministic TM.
  ie, $M_{LBA} = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$

  * The endmarkers are not part of Γ.
  * Input always lies between end markers.
  * The TM cannot replace endmarkers with anything else.

Turing Machine Model

- The turing machine is a collection of 7-tuples.
  ie, $M_{TM} = (Q, \Sigma, \Gamma, \delta, q_0, \Delta, F)$

  Here,
  Q: Finite set of all states.
  Σ: Input Alphabet
  Γ: set of all tape symbols.
  δ: Transition function

i.e., $Q \times \Sigma \rightarrow Q \times \Gamma \times R/L$

q_0: Start state.

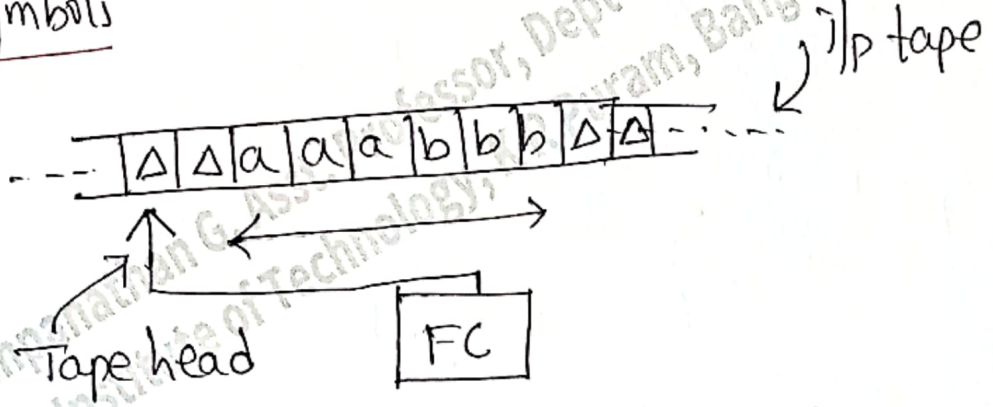
Δ: Blank Symbol.

R/L: Right / left movement of tape head.

F: Set of all possible final states.

Working Principle

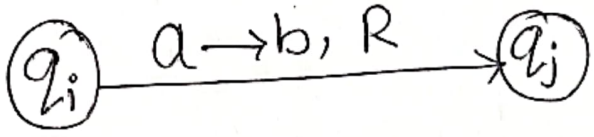
- The input tape is having an infinite number of cells.
- Each cell consists on exactly one input symbol.
- The empty tape is filled with blank symbols



- Finite control contains transitions defined.
- The tape head along with finite control reads the input.

- The tape head is capable of moving left (or) right.

- Various operations that can be performed include:
  * Read / scan the symbol below tapehead.
  * Update / write a symbol below tape head.
  * Move the tape head one step left.
  * Move the tape head one step right.

ie., 

Here on moving from 'q_i' to 'q_j', the scanned input 'a' is replaced by 'b' and tape head moves right.

Representation of Turing Machine

- Apart from the usual representation of TM, it can be represented using:
  a) Instantaneous Description.
  b) Transition table.
  c) Transition Diagram.

Transition Table

- Movement from one state to another state can be easily given in table.
- This table consists of Input alphabets and tape symbols as well.

Ex: TM to accept even number of ones.

	0	1	Δ
q_0	$(q_0, 0, R)$	$(q_1, 1, R)$	Halt
q_1	$(q_1, 0, R)$	$(q_1, 1, R)$	–
q_2	–	–	–

Transition Diagram

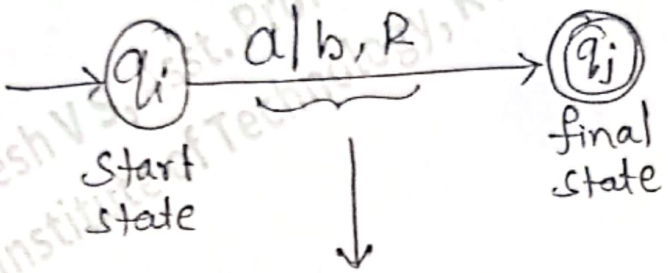
- This consists of states represented by small circles.
- The path between two states are represented by a line.
- On this line we represent the following:

$(a/b, R/L)$

a: Current Symbol b: Symbol to replace (tape symbol)
R/L: Left/right movement.

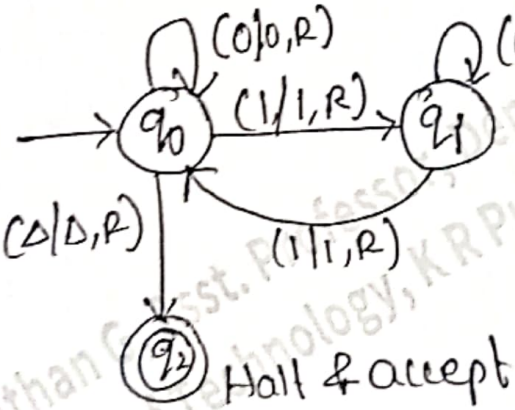
- Initial state has an incoming arrow. where as final state will be two concentric circle.

- In general, path between two states would be as follows:



on input 'a', we move from q_i to q_j by replacing 'a' with 'b'. Tape head moves one cell right.

Ex: For above exam,



Instantaneous Description

- This is another way of representing the operation of turing machine.

- ID of a TM can be given by a triplet.

- It is represented as:

$(\alpha_1 q \alpha_2)$

α_1: string scanned till q:

q: Current state

α_2: Remaining part of the string.

ie., $\underbrace{X_1 X_2 \cdots X_{i-1}}_{\alpha_1} q \underbrace{X_i X_{i+1} \cdots X_n}_{\alpha_2}$

→ The ID for above:

$$\delta(q, X_i) \vdash (q_1, Y, R)$$

ie., $X_1 X_2 \cdots X_{i-1} Y q_1 X_i X_{i+1} \cdots X_n$

Here, in state 'q' on seeing the symbol X_i, we moved to state q_1 replacing 'X_i' with 'Y' and to one cell right.

Ex: For above problem, The ID for input string 0110 is as follows:

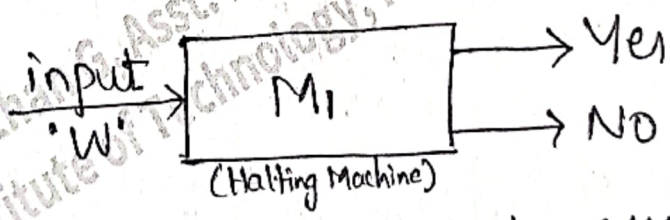
$\delta(q_0, 0110) \vdash (0 q_0 110) \vdash 01 q_1 10 \vdash$

$011 q_0 0 \vdash 0110 q_0 \Delta \vdash 0110 \Delta q_2 \Delta$ (Halt & accept)

Halting Problem in TM

- A Halting problem is an Undecidability.
- It is actually not a problem, instead it is the question that, "Is it possible to tell whether a given machine will halt for some given input".
- Let us assume that there exists a TM which solves a given problem.
  - ie., we consider a Halting machine.

    The m/c that produces "Yes" or "No" in finite amount of time.
- This machine can be represented as follows:



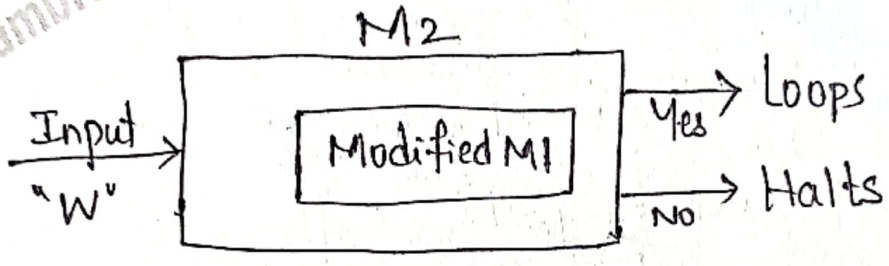
Yes: Machine halts by accepting string.

No: Machine halts by rejecting string.

→ Now let us consider another machine M_2 as follows:

* If M1 returns Yes, then M_2 loops forever
* If M1 returns No, then M_2 halts.

→ That is, this M_2 is a modified machine with modification stated as above.



→ We can observe that, M_2 itself is a turing machine which works as:

* If M_2 halts on an input, it loops forever.
* Else Halts.

→ Here we get contradiction to the assumption.

Hence the Halting Problem is Undecidable.

Variants of turing machine

- The two main variants of turing machine are:
  a) TM with more than one tape called as multitape TM and
  b) TM where $\delta(q,a) = \{(P_1, y_1, D_1), (P_2, y_2, D_2), \ldots, (P_\delta, y_\delta, D_\delta)\}$ called as nondeterministic TM.

Multi-tape turing machine

- 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 Δ, not in P called the blank symbol.

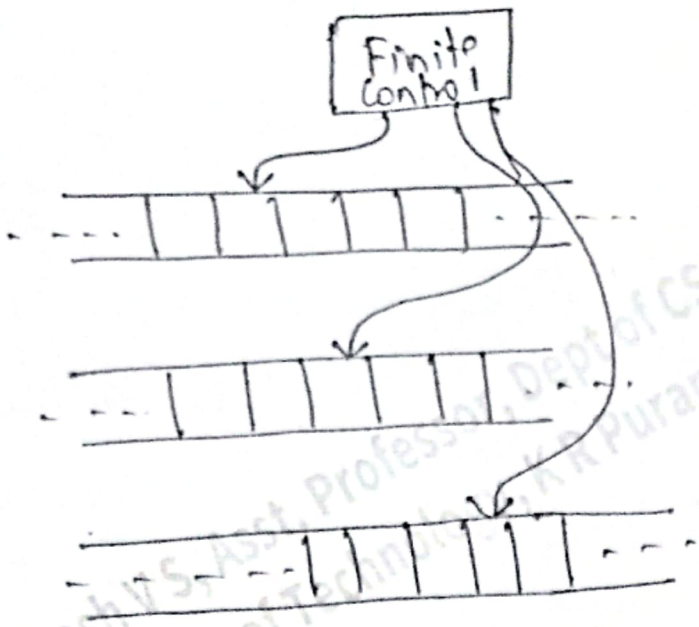


fig: Multitape TM

- There are k-tapes, each divided into cells.

- Initially,
  * First tape holds input string 'w'.
  * Other tapes will have Δ (blank) symbol.
  * Head of the first tape is at left end of input 'w'.
  * Heads of other tapes can be at any cell.

- "δ" is a partial function:

$$Q \times \Gamma^k \longrightarrow Q \times \Gamma^k \times \{L, R, S\}^k$$

- A particular move depends on:
  * The current state and
  * k tape symbols under k tape heads.

- In a typical move,
  * Machine enters new state.
  * On each tape, a new symbol is written in the cell under the head.
  * Each tape head moves to left, right (or) remains stationary.

  Note: Movement is independent.

- Every language accepted by a multitape TM is accepted by single tape TM.

Non Deterministic TM

- A Non Deterministic TM is a 7-tuple representation.

$$M_{NDTM} : (Q, \Sigma, \Gamma, \delta, q_0, \Delta, F)$$

Here,

Q ← Finite non-empty set of state.

Γ ← Finite non-empty set of tape symbols.

$\Delta \in \Gamma$ ← Blank Symbol.

Σ ← Non empty subset of Γ.

$\Delta \notin \Sigma$

q_0 ← Initial state

$F \subseteq Q$ ← set of final states.

δ ← Transition function

$$Q \times \Gamma \longrightarrow (Q \times \Gamma \times \{L, R\})^{2Q}$$

- If M is a nondeterministic TM, then there is a deterministic TM M1 such that $T(M) = T(M_1)$.

Other variants of TM include:

a) Multiple track TM.
b) Two way infinite tape TM.
c) Multitape Multihead TM.
d) Multi Dimensional TM.
e) Off line TM.

Multiple track TM

- It has k-Tracks.
- One R/W head, that reads and writes all of them one by one
- It can be simulated by a single track TM.

Two way infinite tape TM

- It is bounded in both directions.
  ie., left and Right.
- It can be simulated by one-way TM.

Multi Tape Multi Head TM

- It has multiple heads and multiple tapes.
- Each tape is controlled by seperate head.
- It can be simulated by standard TM.

Multi Dimensional TM

- It has multi dimensional tape.
- Tape head can move in any direction.
  ie., left, Right, Up and Down.
- It can be simulated by a 1D-TM.

Off Line TM

- It is similar to multi tape TM, but it's input is read only.
- It has two end markers for input on input tape.
- Remaining tapes are two-way infinite tape.

Recursive and Recursively Enumerable language.

Given a TM, there are the possibilities that exists:

a) It halts and accepts the string.
b) It halts and rejects the string.
c) It never halts.

Recursive language

* If there exists a turing machine, then that language 'L' is recursive.
* For every string that is present in language, the machine accepts it.
* whereas, it rejects all strings not there in the language.
* Conider that there is a language 'L' and it have problem.
* The problem is:
  a) Decidable, if 'L' is recursive. and
  b) Undecidable, other wise.

Recursive Enumerable language

* If there exits a TM, then the language accept it such language are recursive enumerable language.

* There are the languages for which if problem exist, then those problems are decidable.

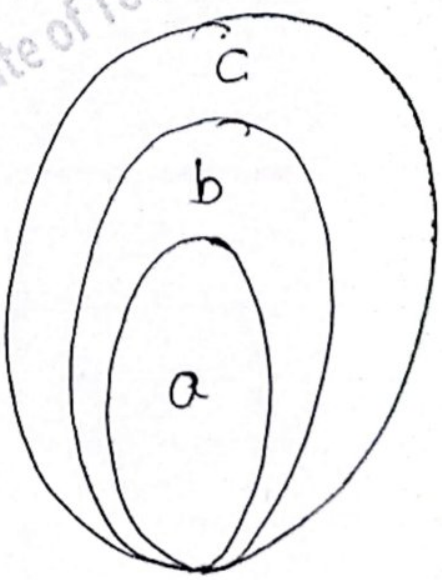


fig: Relationship b/w languages.

Here:

a - Recursive language
b - Recursively Enumerable language
c - Non-Recursively Enumerable language.

Techniques for TM Construction

- High level Conceptual tools makes the construction of TMs easier.
- There are four techniques. They are:

a) Turing machine with stationary head.
b) Storage in the state.
c) Multiple track turing machine.
d) Subroutines.

Turing machine with stationary head

- In the formal definition of TM, 'δ' is defined as:

$$\delta(q,a) \longrightarrow (q_1, Y, D)$$

Here, D is Left/Right.

- Suppose we want to include the option that the head can continue to be in same cell for some input.
- This can be done as follows.

$$\delta(q,a) \longrightarrow (q_1, Y, S)$$

Here, S is stationary.

i.e., TM on reading 'a' in state 'q', it changes to 'q_1' and writes 'Y' in current cell and remains there only.

- This can be shown as follows:

$$wqax \vdash wq'yx.$$

- Now the 'δ' function in formal definition would change as follows:

$$\delta(q,a) \rightarrow (q_1, Y, D)$$

Here, D is left | Right | Stationary.

Storage in the state

- The machines that we have studied so far ie., FA, PDA (or) TM has states.
  - These states are used to remember things.
  - We can also use states to store a symbol as well.
  - In that case, it becomes a pair (q,a).

    Here, q: The state

    a: Tape symbol stored.
  - The new set of states now become:

    $Q \times \Gamma$.

Multiple track turing machine.

- In a multiple track TM, a single tape is assumed to be divided into several tracks.
- In this case, Γ, tape alphabet is required to consists of k-tuples.
  k: Number of tracks.

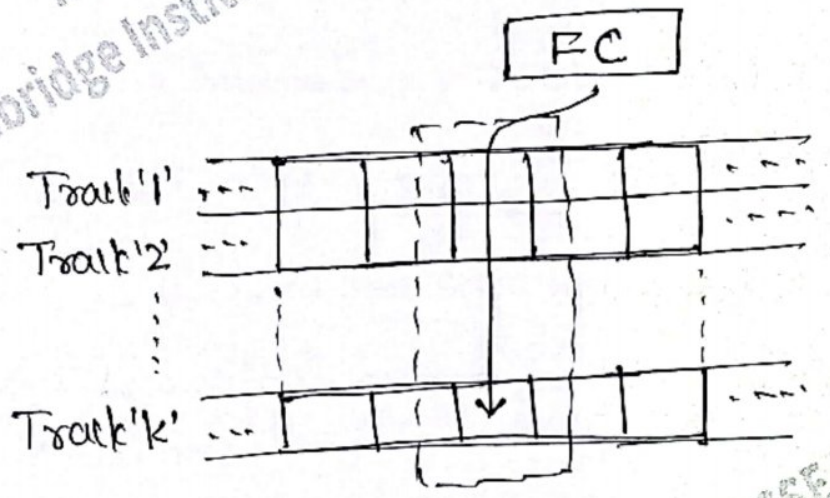


fig: Multiple track TM

- The only difference when compared with Standard TM is set of tape symbols.
- Tape Symbols,

  Standard TM: Γ

  Multiple track TM: Γ^*

Subroutines

- When certain task need to be executed repeatedly, we use subroutines.
- TM for subroutine has:
  a) Initial state and
  b) Return state.
- After reaching the return state, there is a temporary halt.
- For using subroutines, new states are introduced.
- When there is need for calling the subroutine, moves are affected such that we enter initial state of subroutine.

Growth rate of function

- When we have two algorithms for the same problem, we may require a comparison between the running time of these two algorithms.
- Let us consider a set of natural numbers.
- The growth rate of functions defined on set of natural numbers N is:

- Let $f, g : N \longrightarrow R^+$

  We say that, $f(n) = O(g(n))$ if there exists positive integer 'C' and 'N_0' such that

$$f(n) \leq Cg(n) \quad \forall\ n \geq N_0.$$

- We say that, 'f' is of the order of 'g'.

  ie., f is big oh (O) of g.

Ex: $f(n) = 4n^3 + 5n^2 + 7n + 3$

Let us take $C = 5$ and $N_0 = 10$

ie., $f(n) = 4n^3 + 5n^2 + 7n + 3 \leq 5n^3$ for $n \geq 10$

when $n = 10$,

$$5n^2 + 7n + 3 < 10^3$$

for $n > 10$,

$$5n^2 + 7n + 3 < n^3$$

$\therefore f(n) = O(n^3)$

Theorem: Growth rate of any exponential function is greater than that of any polynomial.

Proof: Let,

$$p(n) = a_k n^k + a_{k-1} n^{k-1} + \cdots + a_1 n + a_0$$

$$q(n) = a^n \text{ for some } a > 1$$

We have to prove that

$$n^k = O(a^n) \text{ and}$$

$$a^n \neq O(n^k)$$

Consider, $\log n = \log_e n$

By L'hospitals rule, $\frac{\log_e n}{n}$ tends to 0 as $n \to \infty$.

If $z(n) = \left[e^{k\left(\frac{\log_e n}{n}\right)}\right]$

then,

$$(z(n))^n = \left[e^{k\left(\frac{\log_e n}{n}\right)}\right]^n$$

$$= e^{k \cdot \log_e n} = e^{\log_e n^k}$$

$$= n^k$$

Choose No such that,

$$z(n) \leq a \quad \forall n \geq No$$

$$\therefore n^k = z(n)^n \leq a^n$$

$$\therefore n^k = O(a^n) \quad \text{---} (1)$$

Consider, $\frac{a^n}{n^k}$. we know that $n^k < a^n$ for large 'n' and positive integer 'k'

$$n^{k+1} \leq a^n \text{ (or) } \frac{a^n}{n^{k+1}} \geq 1.$$

multiply by 'n' both side

ie., $n\left(\frac{a^n}{n^{k+1}}\right) \geq n$

ie., $\frac{a^n}{n^k}$ is unbound

$$\therefore a^n \neq O(n^k) \quad \text{---} (2)$$

from (1) and (2),

Growth rate of any exponential function is greater than that of any polynomial.