Scheme and Solution

Internal Test – III, Nov. 2019

Sub: Automata Theory and Computability (17CS54/15CS54)

Dept. of CSE, CMRIT, Bangalore

USN | | | | | | | | | | |

Internal Assessment Test III –Nov. 2019

Sub:	Automata Theory and Computability					Sub Code:	17CS54/ 15CS54	Branch:	CSE	
Date:	16/11/19	Duration:	90 min's	Max Marks:	50	Sem/Sec:	5/CSE(A,B,C)		OBE	

Answer any FIVE FULL Questions

		MARKS	CO	RBT
1	Design a Turing Machine to accept Palindrome of strings over Σ= {a,b}.Draw the transition diagram and show the moves for the input string babab.	[10]	CO5	L3
2	Define PDA. Design a PDA which will accept the language $L=\{0^n1^{2n} \mid n>=1\}$. Show the moves of the PDA for the string 001111.	[10]	CO4	L3
3(a)	With a neat diagram explain Linear Bounded Automata.	[5]	CO5	L2
(b)	Explain Multi-tape Turing machine with a neat diagram.	[5]		
4	Covert the following grammar to PDA E→E+T | E - T | T T→T*F | F F→(E) | a	[10]	CO4	L3
5	Write Short notes on: (a) Post correspondence problem (b) Halting problem of TM	[2*5]	CO5	L2
6	Define a Turing machine. Explain the working of a basic TM with a neat diagram. Also define the language accepted by TM.	[10]	CO4	L1
7	Write Short notes on: (a) Quantum Computer(b) Class NP (c) Growth rate function (d) Undecidable Languages	[10]	CO6	L2

Q.1. Design a Turing Machine to accept Palindrome of strings over $\Sigma = \{a,b\}$.Draw the transition diagram and show the moves for the input string babab. **[5+5]**

Ans:

The transition function is given below.

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

$\delta(q_1,a)=(q_2,B,R)$

$\delta(q_1,b)=(q_5,B,R)$

$\delta(q_1,B)=(q_7,B,R)$

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

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

$\delta(q_2,B)=(q_3,B,L)$

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

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

$\delta(q_5,B)=(q_6,B,L)$

$\delta(q_3,a)=(q_4,B,L)$

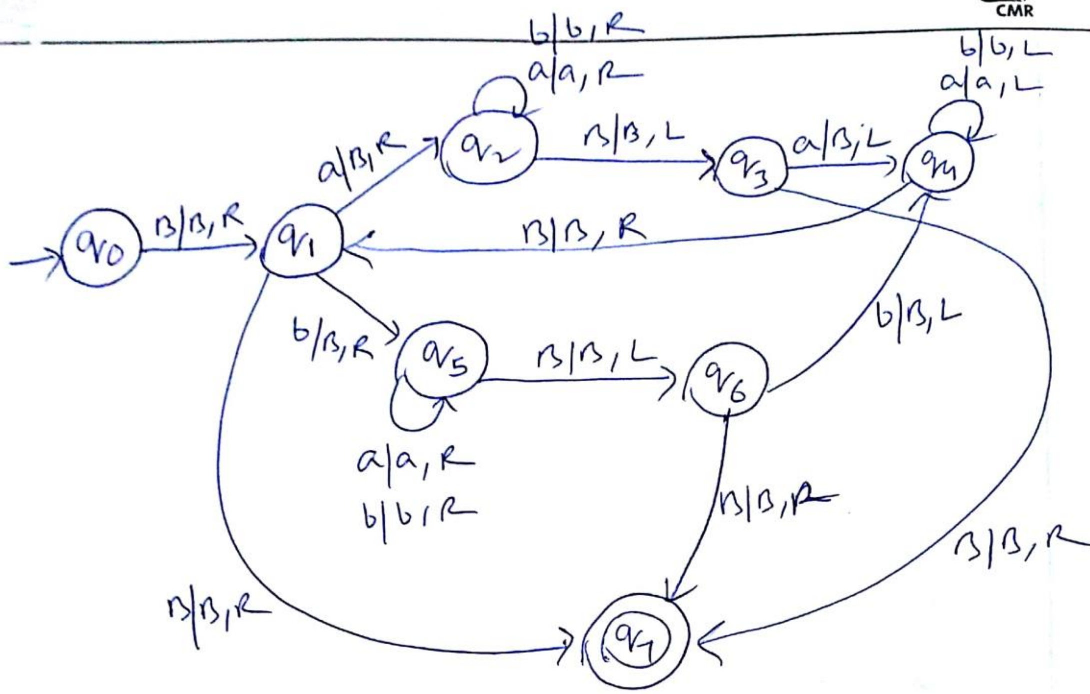
$\delta(q_3,B)=(q_7,B,R)$

$\delta(q_4,a)=(q_4,a,L)$

$\delta(q_4,b)=(q_4,b,L)$

$\delta(q_6,b)=(q_4,B,L)$

$\delta(q_6,B)=(q_7,B,R)$



q_0 B bababB ⊢ B q_1 bababB ⊢ BB q_5 ababB

⊢ BBa q_5 babB ⊢ BBab q_5 abB ⊢ BBaba q_5 bB

⊢ BBabab q_5 B ⊢ BBaba q_6 bB ⊢ BBab q_4 aBB

⊢ BBa q_4 baBB ⊢ BB q_4 abaBB ⊢ B q_4 BabaBB

⊢ BB q_1 abaBB ⊢ BBB q_2 baBB ⊢ BBBb q_2 aBB

⊢ BBBba q_2 BB ⊢ BBBb q_3 aBB ⊢ BBB q_4 bBBB

⊢ BB q_4 BbBBB ⊢ BBB q_1 bBBB ⊢ BBBB q_5 BBB

⊢ BBB q_6 BBBB ⊢ BBBB q_7 BBB

Accepted

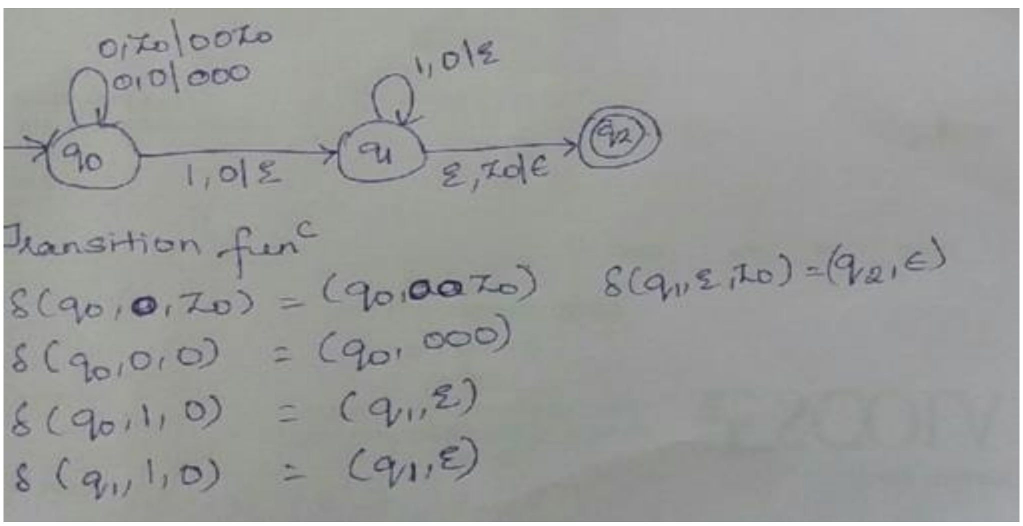
Q.2. Define PDA. Design a PDA which will accept the language $L=\{0^n1^{2n} | n>=1\}$. Show the moves of the PDA for the string 001111.

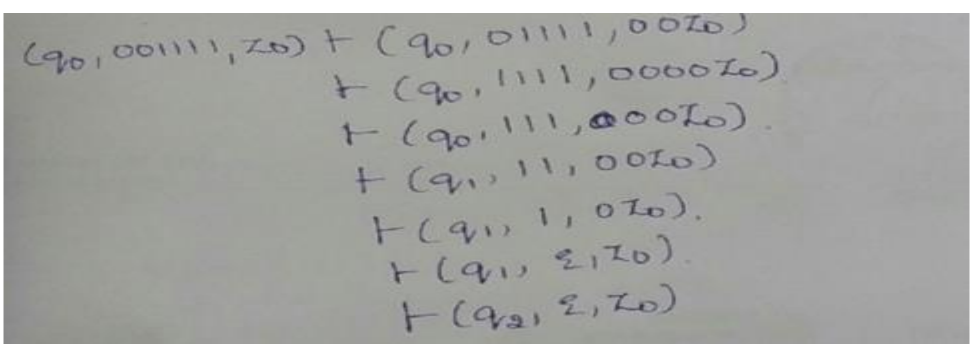
ANS

Pushdown Automata is a finite automata with extra memory called stack which helps Pushdown automata to recognize Context Free Languages.

A Pushdown Automata (PDA) can be defined as :

- Q is the set of states
- ∑is the set of input symbols
- Γ is the set of pushdown symbols (which can be pushed and popped from stack)
- q0 is the initial state
- Z is the initial pushdown symbol (which is initially present in stack)
- F is the set of final states
- δ is a transition function which maps Q x {Σ ∪ ∈} x Γ into Q x Γ*. In a given state, PDA will read input symbol and stack symbol (top of the stack) and move to a new state and change the symbol of stack.





Q.3.a. With a neat diagram explain Linear Bounded Automata. [2+3]

Ans: Alinear bounded automata (**LBA**) is a restricted form of Turing machinewith a tape of some bounded finite length. The computation is restricted to the constant bounded area. The input alphabet contains two special symbols which serve as left end markers and right end markers which mean **the transitions neither move to the left of the left end marker nor to the right of the right end marker of the tape.**

A linear bounded automaton can be defined as an 8-tuple (Q, ∑,Γ, q_0, M_L, M_R, δ, F) where −

- **Q** is a finite set of states
- Γ is the tape alphabet
- ∑ is the input alphabet
- $\mathbf{q_0}$ is the initial state
- $\mathbf{M_L}$ is the left end marker (ex: <)
- $\mathbf{M_R}$ is the right end marker(ex: >) where $M_R \neq M_L$
- **δ** is a transition function which maps each pair (state, tape symbol) to (state, tape symbol, Constant 'c') where c can be 0 or +1 or -1
- **F** is the set of final states

Working space

<	a	b	>

Left end marker Input string Right end marker

A deterministic linear bounded automaton is always **context-sensitive** and the linear bounded automaton with empty language is **undecidable.**

- The language accepted by LBA is called **context-sensitive language.**
- **Example:**

  L={$a^n b^n c^n$} and L={ $a^{n!}$ }

- It is more powerful that NPDA but less powerful that TM

Q.3.b. Explain the Multi tape Turing Machine with neat block diagram. [2+3]

Ans:

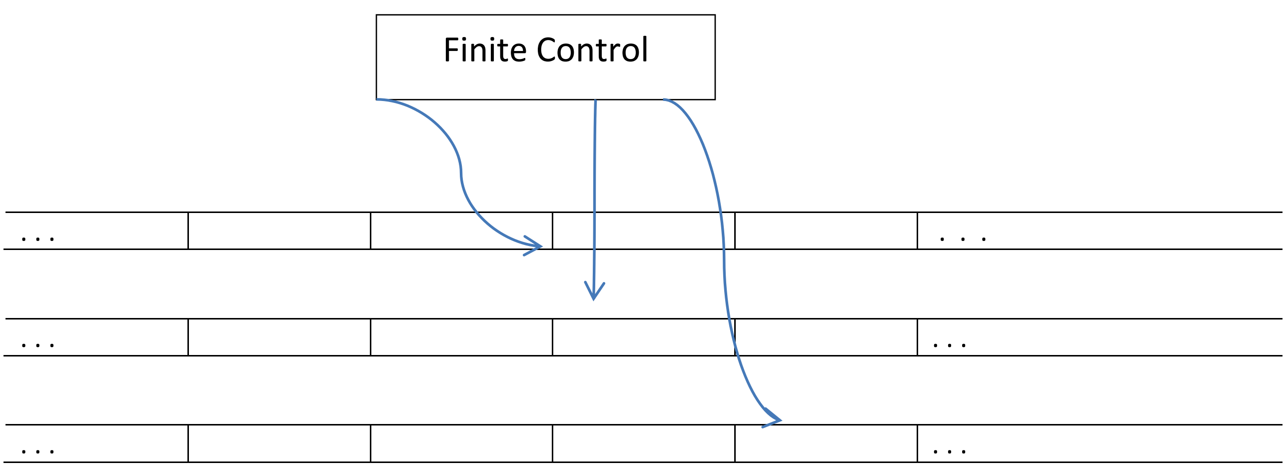
VARIANTS OF TURING MACHINE:

There are two new models of Turing machines:

1. MULTITAPE TURING MACHINE
2. NON-DETERMINISTIC TURING MACHINE

MULTITAPE TURING MACHINE

Multi-tape Turing Machines have multiple tapes where each tape is accessed with a separate head. Each head can move independently of the other heads. Initially the input is on tape 1 and others are blank. At first, the first tape is occupied by the input and the other tapes are kept blank. Next, the machine reads consecutive symbols under its heads and the TM prints a symbol on each tape and moves its heads.



A Multi-tape Turing machine can be formally described as a 7-tuple (Q,Σ,Γ, B, δ, q_0, F) where –

- **Q** is a finite set of states
- Σ is a finite set of inputs

- Γ is the tape alphabet
- **B** is the blank symbol
- **δ** is a relation on states and symbols where

  $\delta: Q \times \Gamma^k \rightarrow Q \times (\Gamma \times \{Left, Right, Stationary\})^k$

  where there is **k** number of tapes
- **q_0** is the initial state
- **F** is the set of final states

  In each move the machine M:

  (i) Enters a new state

  (ii) A new symbol is written in the cell under the head on each tape

  (iii) Each tape head moves either to the left or right or remains stationary.

Q.4 Covert the following grammar to PDA [10]

E→E+T | E - T | T T→T*F | F F→(E) | a

Ans:

δ (q,ε, E)= { (q, E+T), (q, E-T), (q, T) }

δ (q, ε,T)= { (q, T*F), (q, F) }

δ (q, ε,F)= { (q, (E)), (q, a) }

δ (q,+,+)=(q,ε)

δ (q,*,*)=(q,ε)

δ(q,-,-)= (q,ε)

δ (q,(,( )=(q,ε)

δ (q,),))=(q,ε)

δ (q,a,a)=(q,ε)

Q.5. (a) **Post Correspondence Problem:**

The Post Correspondence Problem (PCP), introduced by Emil Post in 1946, is an undecidable decision problem. The PCP problem over an alphabet ∑ is stated as follows −

Given the following two lists, **M** and **N** of non-empty strings over ∑ −

$M = (x_1, x_2, x_3, \ldots\ldots\ldots, x_n)$

$N = (y_1, y_2, y_3, \ldots\ldots\ldots, y_n)$

We can say that there is a Post Correspondence Solution, if for some $i_1, i_2, \ldots\ldots\ldots\ldots i_k$, where $1 \leq i_j \leq n$, the condition $x_{i1} \ldots\ldots x_{ik} = y_{i1} \ldots\ldots y_{ik}$ satisfies.

Example 1

Find whether the lists

M = (abb, aa, aaa) and N = (bba, aaa, aa)

have a Post Correspondence Solution?

Solution

	x_1	x_2	x_3
M	Abb	aa	aaa
N	Bba	aaa	aa

Here,

$x_2x_1x_3$ = 'aaabbaaa'

and **$y_2y_1y_3$ = 'aaabbaaa'**

We can see that

$x_2x_1x_3 = y_2y_1y_3$

Hence, the solution is **i = 2, j = 1, and k = 3.**

Example 2

Find whether the lists **M = (ab, bab, bbaaa)** and **N = (a, ba, bab)** have a Post Correspondence Solution?

Solution

	x_1	x_2	x_3
M	ab	bab	bbaaa
N	a	ba	bab

In this case, there is no solution because −

$| x_2x_1x_3 | \neq | y_2y_1y_3 |$ (Lengths are not same)

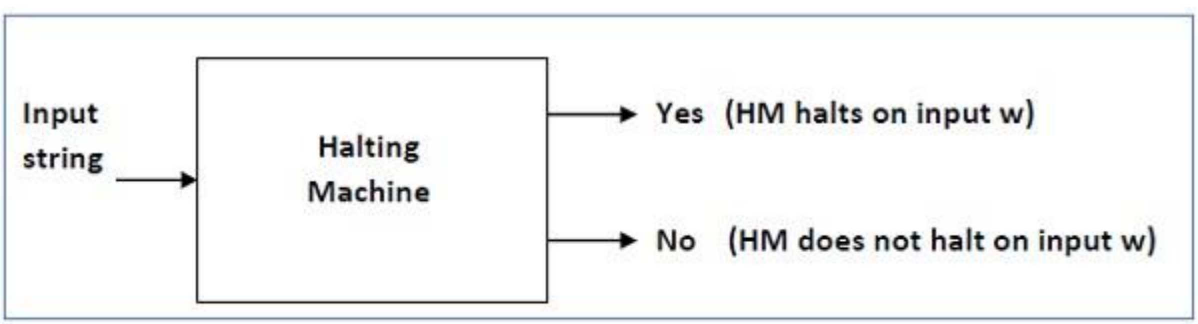
Hence, it can be said that this Post Correspondence Problem is **undecidable**.

(b) Halting Problem of TM:

Input − A Turing machine and an input string **w**.

Problem − Does the Turing machine finish computing of the string **w** in a finite number of steps? The answer must be either yes or no.

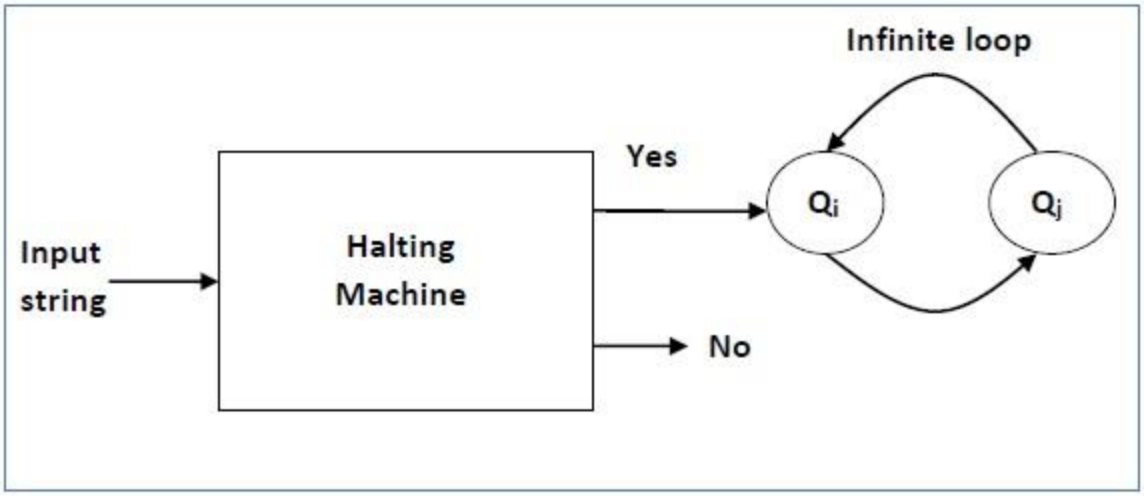
Proof − At first, we will assume that such a Turing machine exists to solve this problem and then we will show it is contradicting itself. We will call this Turing machine as a **Halting machine** that produces a 'yes' or 'no' in a finite amount of time. If the halting machine finishes in a finite amount of time, the output comes as 'yes', otherwise as 'no'. The following is the block diagram of a Halting machine −



Now we will design an **inverted halting machine (HM)’** as −

- If **H** returns YES, then loop forever.
- If **H** returns NO, then halt.

The following is the block diagram of an ‘Inverted halting machine’ −



Further, a machine **$(HM)_2$** which input itself is constructed as follows −

- If $(HM)_2$ halts on input, loop forever.
- Else, halt.

Here, we have got a contradiction. Hence, the halting problem is **undecidable**.

Q.6. Define a Turing machine. Explain the working of a basic TM with a neat diagram. Also define the language accepted by TM. [2+6+2]

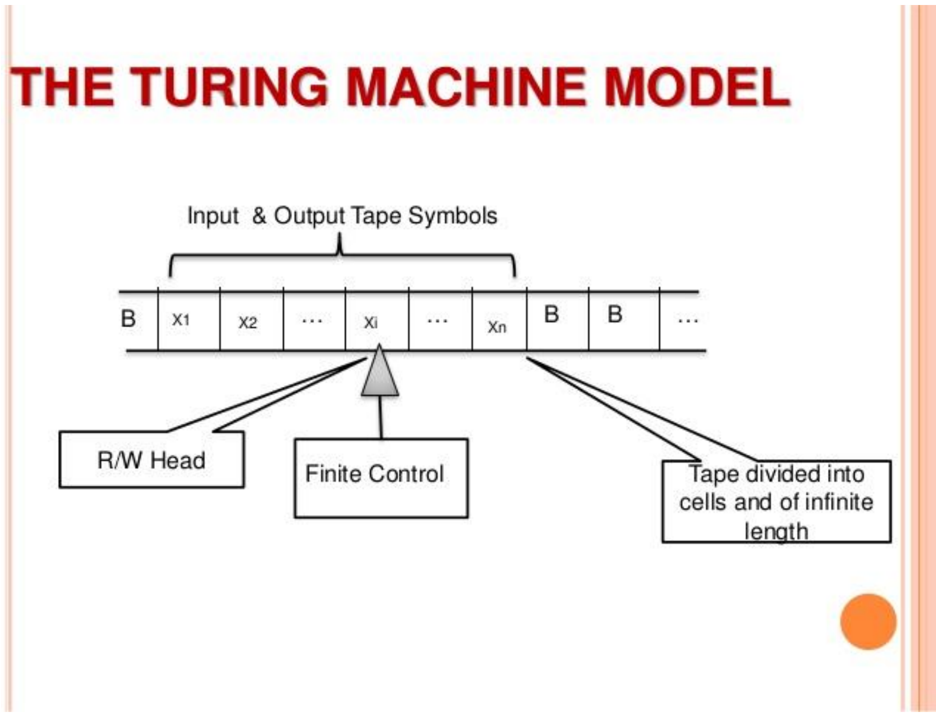
Ans: A Turing Machine is an accepting device which accepts the languages (recursively enumerable set) generated by type 0 grammars. It was invented in 1936 by Alan Turing.

Definition

A Turing Machine (TM) is a mathematical model which consists of an infinite length tape divided into cells on which input is given. It consists of a head which reads the input tape. A state register stores the state of the Turing machine. After reading an input symbol, it is replaced with another symbol, its internal state is changed, and it moves from one cell to the right or left. If the TM reaches the final state, the input string is accepted, otherwise rejected.

A TM can be formally described as a 7-tuple (Q, X, ∑, δ, q_0, B, F) where −

- **Q** is a finite set of states
- **X** is the tape alphabet
- **∑** is the input alphabet
- **δ** is a transition function; δ : Q × X → Q × X × {Left_shift, Right_shift}.
- **q_0** is the initial state
- **B** is the blank symbol
- **F** is the set of final states



A TM accepts a language if it enters into a final state for any input string w. A language is **recursively enumerable** (generated by Type-0 grammar) if it is accepted by a Turing machine.

A TM decides a language if it accepts it and enters into a rejecting state for any input not in the language. A language is **recursive** if it is decided by a Turing machine.

There may be some cases where a TM does not stop. Such TM accepts the language, but it does not decide it.

7) Short Notes:

The Classes P and NP :-

Class P :- A Language L is in Class P if there exists some polynomial T(n) such that L = T(M) for some deterministic Turing Machine M of time Complexity T(n).

Class NP :- A Language L is in Class NP if there exists a nondeterministic Turing Machine M and a polynomial time Complexity T(n) such that L = T(M) and M executes atmost T(n) moves for every input w of length n.

Growth Rate of functions

Definition: 1

Let $f, g: N \rightarrow R^+$ (R^+ being set of all +ve real Numbers) we say that $f(n) = O(g(n))$ if there exist positive integer C and N_0 such that

$$f(n) \le C\, g(n) \;,\; n \ge N_0$$

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

Prove that $f(n) = O(n^3)$

Assume $C = 5$, $N_0 = 10$

then $f(n) = 4n^3 + 5n^2 + 7n + 3 \le 5n^3$ for $n \ge 10$

when $n = 10$

$573 < 10^3$ so

$f(n) = O(g(n))$ Hence proved.

Definition: 2

An exponential function is a function $a: N \rightarrow N$ defined by $a(n) = a^n$ for $a > 1$

when n increases, each of $n, n^2, 2^n$ increases.

Definition: 3

If f and g are two functions and $f = O(g)$ but $g \ne O(f)$ we say that growth rate of g is greater than that of f.

Quantum Computation

Quantum Computers :

We know that a bit 0 or 1 is the fundamental concept of classical computation and information. Classical computer is a system built from the electronic circuit containing wires and logical gates. Let's discuss about quantum bits and quantum circuit.

Quantum bit (or) qubit is described mathematically as

$$|\psi> = \alpha|0> + \beta|0>$$

Qubit is defined as, classical bits has the two states 0 and 1. Two possible states for qubit are $|0>$ and $|1>$. Instead of

reya a.K 17CS11

classical bit, qubit can be in infinite no. of states other than $|0>$ and $|1>$ that is described as $|\psi> = \alpha|0> + \beta|0>$, where α and β are complex numbers such that $|\alpha|^2 + |\beta|^2 = 1$. 0 and 1 are known as computational basis of states and $|\psi>$ is known as superposition.

Multiple qubit also defined in similar way for two qubit, there are four computional basis states $|00>, |01>, |10>,$ and $|11>$ and quantum states $|\psi> = \alpha_{00}|00> + \alpha_{01}|01> + \alpha_{10}|10> + \alpha_{11}|11>$ with $|\alpha_{00}|^2 + |\alpha_{01}|^2 + |\alpha_{10}|^2 + |\alpha_{11}|^2 = 1$

For qubit gates, in classical NOT gate interchanges 0 and 1, but qubit NOT gate interchanges $\alpha|0> + \beta|1>$ to $\alpha|1> + \beta|0>$.

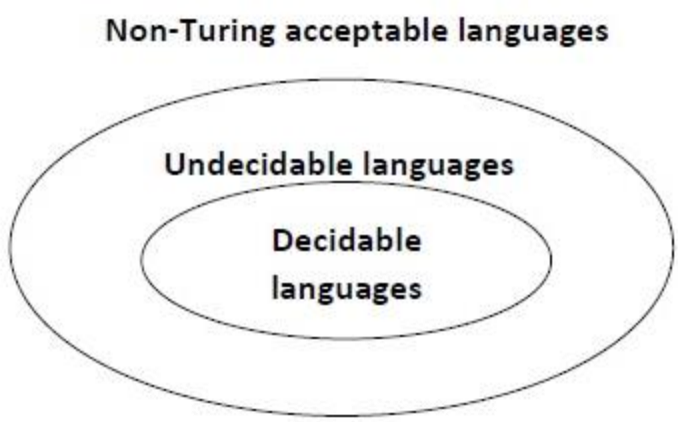
Qubit NOT gate is linear in two dimentional complexity vector space

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

$\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$ is the unitary matrix.

Undecidable language

For an undecidable language, there is no Turing Machine which accepts the language and makes a decision for every input string **w** (TM can make decision for some input string though). 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 sometimes, they may be recursively enumerable languages.



Example

- The halting problem of Turing machine
- The mortality problem
- The mortal matrix problem
- The Post correspondence problem, etc.