Volume 3, No. 5, May 2012
Journal of Global Research in Computer Science
ISSN-2229-371X
REVIEW ARTICLE
Available Online at www.jgrcs.info

COMPARATIVE ANALYSIS ON TURING MACHINE AND QUANTUM TURING MACHINE

Tirtharaj Dash[*1] and Tanistha Nayak[2]
[*1, 2]Department of Information Technology
National Institute of Science and Technology Berhampur-761008, India
tirtharajnist446@gmail.com
tanisthanist213@gmail.com

***Abstract*—** Now-a-days every computing device is based on the Turing Machine. It is known that classical physics is sufficient to explain macroscopic phenomena, but is not to explain microscopic phenomena like the interference of electrons. In these days, speed-up and down-sizing of computing devices have been carried out using quantum physical effects; however, principles of computation on these devices are also based on classical physics. This paper tries to analyze mathematically a possibility that the Universal Quantum Turing Machine (UQTM) is able to compute faster than any other classical models of computation. Basically we focused on comparative study on computation power of Universal Turing Machine (UTM) and UQTM. Namely, in the equal, we tried to show that the UQTM can solve any NP-complete problem in polynomial time. The result analysis showed that UQTM is faster for any computation.

Keywords- Quantum; Quantum Computer; Quantum Turing Machine; Universal Turing machine; Universal Quantum Turing Machine.

INTRODUCTION

Quantum is a discrete quantity of energy proportional in magnitude to the frequency of the radiation it represents [1]. An analogous discrete amount of any other physical quantity, such as momentum or electric charge is known as Quantum [1,3]. Bit (Binary Digit) is the smallest unit of information within computer, the only thing that computer can understand Bit is basic unit of Classical Computer. Classical Computer only works binary values called Bits, each having a value usually denoted by 0 or 1. One of the most intuitive representation of bit is an open or closed switch or by extension, and 'on' or 'off' portion of the circuit .In today's modern computer ,this representation remains in transistors, with a high voltage possibly denoting a 1 and low voltage possibly denoting a zero. A two state system (0 →1) is the building block of classical computational device. Quantum bit (Qubit) is a unit of quantum information. Qubit represents both the state memory and the state of entanglement in a system. Quantum entanglement is experimentally verified property of nature.

Quantum Entanglement occurs when the particles such as electron, photon, molecules interacts physically and then become separated. This interaction is called entanglement. Quantum bit (Qubit) can exist in a superposition of states which can be represented as $\alpha|0\rangle+\beta|1\rangle$ where α, β represents complex number satisfying $|\alpha|^2+|\beta|^2=1$.Any state measurement results in $|0\rangle$ with probability $|\alpha|^2$ and $|1\rangle$ with probability $|\beta|2$.A n-Qubit system can exist in any superposition of the 2 basis states. Quantum entanglement is a form of quantum superposition. A Quantum Computer also works with two Eigen values, a 0 and 1, but these units are termed as Qubit or Quantum Bits due to their non classical behavior. A Qubit can take any value i.e. linear combination of Qubit 0 or Qubit 1.

Therefore while classical bits can take any value of north and south and any orientation would be illegal. A Quantum bits (Qubit) can take the value of northeast [2]. However upon measurement, only one of these values, north or south is read. To determine the coefficient of the linear combination of the relative portions of the amount read being north and south must be found out through repeated measurements of the original Qubit state. Only something that exhibit quantum phenomena that can be used to hold a Qubit. It is impossible for a simple on/off switch to hold a linear combination of on or off. On the other hand electron or nuclear spin and polarization state of photon make natural candidates. These quantum states either spin or polarization must also somehow be able to manipulate so that that value could be assigned to each bit [3]. Figure1 shows a binary transition of states incorporating physical phenomenon.

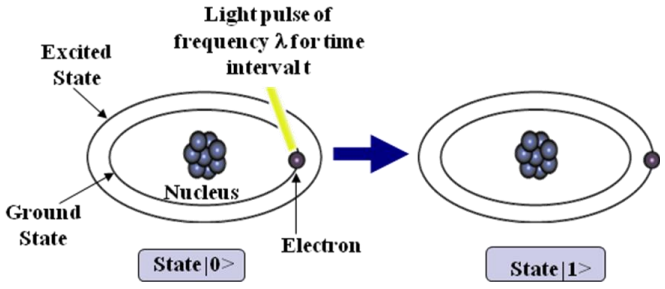


Figure 1 Transition of states in an atom

TURING MACHINE

We are interested in designing an automaton (machine) that can solve our two objectives. *(i)Recognizing (ii) Computation.* A Turing Machine (TM) is a generalization of Pushdown Automata (PDA) a **tape** instead of a tape and stack. The

© JGRCS 2010, All Rights Reserved

potential length of the tape is assumed to be infinite and divided into cells, and one cell hold one input symbol. The head of TM is capable to read as well as write on the tape and can move left or right or can remain static. Turing Machine(TM) is an abstract version of a computer; this has been used to define formally what is computable. The Church's Thesis [4] is supported the fact that a Turing Machine(TM) can stimulate the behavior of general purpose computer system. Alan Turing, an English mathematician in 1936, suggests the concept of Turing Machine. Turing Machine accepts the language defined by the grammars that are called Type 1 languages or Recursively Enumerable languages. Turing Machine is idealized in that they have an infinite memory, which comes in the form of an "infinite tape" (infinite on both the sides). The idea is that this tape consists of cells each of which can take one symbol from some underlying alphabet Σ, and that there is a 'read/write head' which moves over the tape one position at a time and can either read from or write to the tape. It is practical to assume that the input string is written on the tape (from left to right).

When the machine starts the read/write head is positioned on the cell holding the first symbol of the input string. We further assume that the tape contains no further symbols that are all remaining cells are empty. In order to describe what the machine does (that is, under which circumstances it moves the head one position to the left or the right, or it reads from the tape, or writes to the tape) we need to have a notion of state, just as we had with the other automata. As before there is a specific start state, and this time we need an action stop to tell us when the machine has finished its computation.

Abstract Model of a UTM:

A Turing Machine (TM) consists of a finite control, which can be in any of a finite set of states. There is a tape divided into square or cells; each cell can hold any one of a finite number of symbols. Figure 2 shows a model of TM and its components.

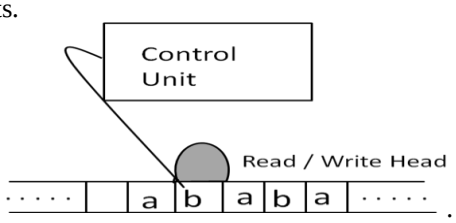


Figure 2: Turing Machine Model

Initially, the input, which is a finite-length string of symbols chosen from the *input alphabet*, is placed on the tape. All other tape cells, extending infinitely to the left and right, initially hold a special symbol called the blank. The blank is a tape symbol but not an input symbol, and there may be other tape symbols besides the input symbols and the blank, as well. There is a tape *head* symbol scanned. In one move, the Turing Machine will work as following steps.

a. The next state optionally may be the same as the current state.
b. Write a tape symbol in the cell scanned. This tape symbol replaces whatever symbol was in that cell. Optionally, the symbol written may be the same as the symbol currently there.
c. Move the tape head left or right. In our formalism we require a move, and do not allow the head to head to remain stationary. This restriction does not constrain what a Turing Machine(TM) can compute, since any sequence of moves with a stationary head could be condensed, along with the next tape head could be condensed, along with the next tape Head move, into a single state change, a new tape symbol, and a move left or right.

Mathematical Description of Turing Machine:

Turing Machine has seven tuples and is defined as,

$$\mathbf{M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)}$$

Where,

Q = the finite set of state the finite control.

Σ = the finite set of state the input symbols.

Γ = the complete set of tape symbols; Σ is always a subset of Γ.

δ: The Transition Function. The arguments of δ(q, X) are a state q and a tape symbol X. The value of δ(q, X) ,if it is defined, is a triple(*p, Y, D*) where

a. p is the next state in Q.
b. Y is the symbol, in Γ, written in the cell being scanned, replacing whatever symbol was there.
c. D is a direction, either L or R, standing for 'left' or 'right' respectively, and telling us the direction in which the head moves.

q_0 = a start State, a member of Q, in which the finite control is found initially.

B = blank symbol.

F = a set of final or accepting states, a subset of Q.

Depending upon the number of moves in transition, a TM may be deterministic or non-deterministic. If TM has at most one move in transition, then it is called Deterministic Turing Machine (DTM). If one or more than one move is possible for single symbol then the TM is Nondeterministic (NDTM). Figure-3 shows a possible NDTM model where each state gives rise to more than one state for a single input symbol [4,5,6].

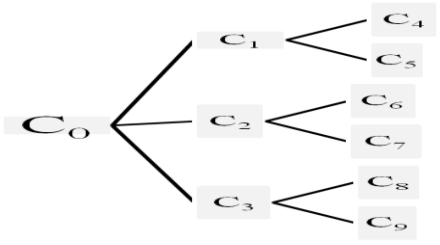


Figure 3: An abstract model of NDTM

A standard TM reads one cell of the tape and moves left or right one cell. The difference between TM and two way automatons is writing capability, not in the movement of the control head.

The transition function δ is defined as δ(p, a) =(q, b, D), where P is present state; q is next state; a,b ∈ Γ and D is the movement (left or right).

© JGRCS 2010, All Rights Reserved

a) After reading an input a ∈ Γ. TM does the following thing. (i) Replace a by b and move right (R) as δ(p, a) =(q, b, R)
b) Without writing move right (R) as δ(p, a) =(q, b, R)
c) Replace a by b and move left(L) as δ(p, a) =(q, b, L)
d) Without write, move left(L) as δ(p, a) =(q, a, L).

Computational power of UTM:

If c_0 is the configuration in the starting state, with w on the tape and the tape head at the left end of the string, w is accepted if the computation $c_0 \rightarrow c_1 \rightarrow \cdots \rightarrow c_f$ eventually reaches an accepting state. If the length of the computation is bounded by a polynomial in the length of x, the language accepted by the machine is in P. The action of the Turing machine can equivalently be described as a linear operator M on an infinite-dimensional space. The set of configurations form a basis for the space.

X_1 q X_2 is the instantaneous description or configuration of a TM which is in state q with X_1 being the contents of the tape to the left of the read write head and X_2 being the contents of the tape to the right of and including the current read write head tape position.

Representation of Language **L = {$a^n b^n c^n$| n>=0}**

Transition state of the above language is given below.

δ (q_0,a)= (q_1,x,R);
δ (q_1,a)= (q_1,a,R);
δ (q_1,y)= (q_1,y,R);
δ (q_1,z)= (q_1,z,R);
δ (q_1,b)= (q_2,y,R);
δ (q_2,b)= (q_2,b,R);
δ (q_2,z)= (q_2,z,R);
δ (q_2,c)= (q_3,z,L);
δ (q_3,a)= (q_3,a,L);
δ (q_3,y)= (q_3,y,L);
δ (q_3,b)= (q_3,b,L);
δ (q_3,z)= (q_3,z,L);
δ (q_3,x)= (q_0,x,R);
δ (q_0,y)= (q_4,y,R);
δ (q_4,y)= (q_4,y,R);
δ (q_4,z)= (q_5,z,R);
δ (q_5,z)= (q_5,z,R);
δ (q_5,$)= ($q_6$,$,R);

The transition diagram for the language L is given in Figure 4 below.

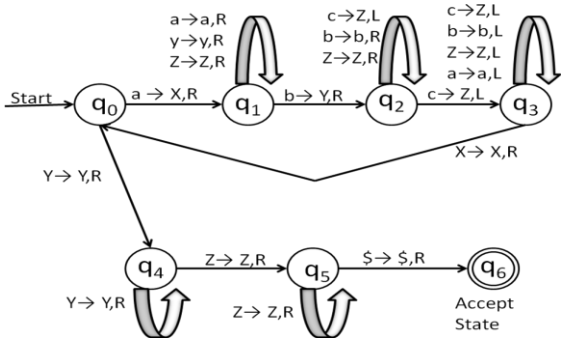


Figure 4: Transition diagram for **L = {$a^n b^n c^n$| n>=0}**

Probabilistic Machine:

With a probabilistic machine, δ defines, for each current state and symbol, a probability distribution over the possible next moves. The action of the machine can be defines as an infinite matrix, where the rows and columns are configuration, and each column adds up to 1.However, how much information can be encoded in a single entry? We require the entries α to be feasibly computable. That is, there is a feasibly computable f such that: $|\mathbf{f(n) - \alpha}| < \mathbf{2^{-n}}$

BPP:

BPP is collection of languages L for which there is a probabilistic machine M, running in polynomial time with

$$P(M \text{ accepts } w) = \begin{cases} \frac{2}{3}, & \text{if } w \in L \\ \frac{1}{3}, & \text{if } w \notin L \end{cases}$$

The class of languages is unchanged if we replace 2/3 and 1/3 by 1- ϵ and ϵ for any error indeed the set of all feasibly computable probabilities with {0, 1/2, 1}. The only inclusion relations we know are P⊆NP and P⊆BPP. *Primality testing*, long known to be in BPP was recently shown to be in P.

The Language Recognization and Turing Machine:

TM can be used as a language recognizer. Turing Machine recognizes all the languages, CFL, CSL, and Type 0. A string W is written on the tape, with blanks filling out the unused portions. The machine enters a initial state q_0, with the read write head position on the left most symbol of W. If after a sequence of moves, the Turing Machine enters the final state and halts, then w is considered to be accepted. A language is accepted by Turing Machine is known as *Recursively enumerable language (RE)*, being enumerable means, TM precisely list all the strings of that language. Although TM may precisely list all the strings of RE (Type 0) language, it may not be able to decide every such language. Deciding a language requires that TM halt on a final state for every w ∈ L.

And also halts on a non final state for every w ∉ L. Therefore we select subclass of type 0 languages, which are decidable by TM, are known as Recursive set of language. For a recursive language, a TM always halts in a final state for all its member strings and accepts them and also halts in a non final state and rejects others which are not member strings. Recursive sets ⊂ Recursively enumerable sets (RE). We have designed a TM for **L = {w; w∈ (a + b)* ending in substring abb}**. Figure 5 shows a TM which recognizes the language L.

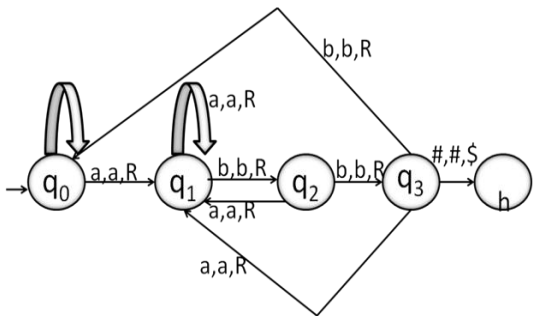


Figure 5: Turing machine for L

© JGRCS 2010, All Rights Reserved

TM can perform computation like addition, subtractions, multiplication, and division. Number on the tape is represented in form of Zero. Suppose i is a positive integer, then o^i represents this integer. It is assumed that a 1 separates two integer numbers. Suppose f is a computable function and has arguments a_1,a_2,a_3 such that f (a_1,a_2,a_3) =m, where a_1,a_2,a_3 and m are some natural numbers then f is said to be computed by TM M .Such a function f is said to be Turing Computable or simply computable. TM computes integer function as defined below.f1 $(a_1, a_2, a_3.....,a_n)\rightarrow m$. If f1 is defined for all arguments $a_1, a_2, a_3.....a_n$ then total function is analogous to recursive language. A function f1$(a_1, a_2, a_3.....,a_n)$ computed by a TM is known as partial recursive function, if f is defined for some, not for all values of $a_1, a_2, a_3.....,a_n$. The TM that compute this function halts for some and may not halts for other input like n!, $\log_2 n$ etc... total recursive function. Figure-6 shows a sample difference between Total Recursive Function and Partial Recursive Function.

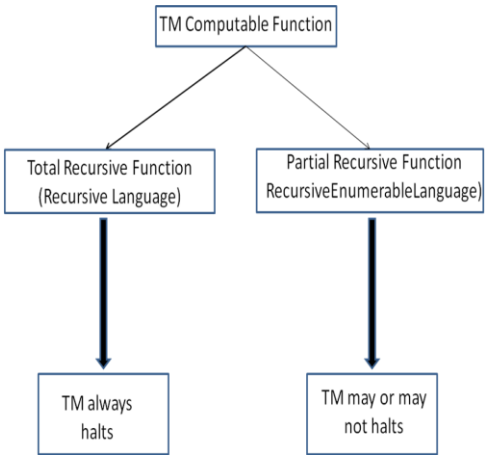


Figure 6: Computational ability of TM

Modification of Turing Machine:

There are many modification of Turing Machine.

a. Two-way infinite TM. L is recognized by a Turing Machine with a two-way infinite tape if and only if it is recognized by one-way infinite tape. Figure 7 shows this machine model.

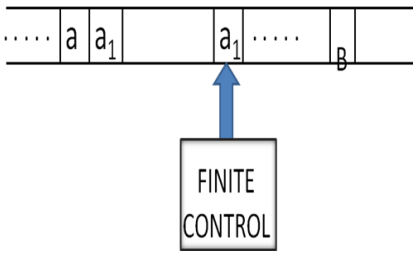


Figure 7: Two-way infinite TM

b. ***Multiple TM:*** If a language L is accepted by multiple tape Turing Machine, it is accepted by single tape Turing Machine. Figure 8 shows a theoritical structure of this type of TM.

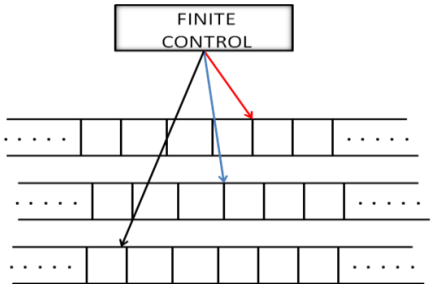


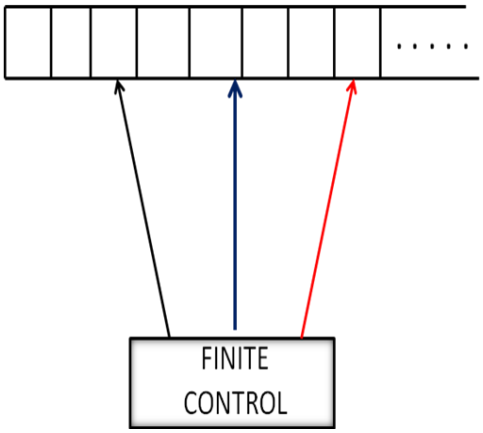
Figure 8: Multiple TM

c. ***Non-deterministic TM:*** If L is accepted by a non-deterministic Turing Machine(TM) M_1, then L is accepted by some deterministic Turing Machine M_2 (See Figure 3).

d. ***Multi Dimensional Turing Machine:*** In k-dimensional TM the tape consists of k-dimensional array to cells infinite in all 2k direction for some fixed k. If L is accepted by k-dimensional Turing Machine M_1, then L is accepted by some single tape Turing Machine M_2.

Multi Head Turing Machine:

A k-head TM has some fixed number k of head are numbered I through K and a move of TM depends on the state and on the symbol scanned by each head in one move the head may move independently left right or remain stationary. If L is accepted by some k-head by some k-head TM M_1, it is accepted by one head TM M_2. Figure 9 refers to K-Headed turing machine.



. Figure 9: K-head TM

Multi Track Turing Machine:

In this type of Turing machine the rope of the Turing Machine TM is divided into k tracks, for any finite k.

Turing Machine with stay option:

In these TM'S the read-write head can stay at the current position upon reading an input symbol without moving Left or right.

Off-line Turing Machine:

An off line Turing Machine(TM) is a multiple TM whose input tape is read-only. Usually we surround the input tape by end marker c on the left and s on right. The Turing Machine(TM) is not allowed to move the input tape head off

© JGRCS 2010, All Rights Reserved

the region between c and s, it should be obvious that the offline TM is just special case of multiple TM. An offline TM can simulate any TM by using one more tape than M. The first thing of offline TM does is copy its own input onto the extra tape and it then stimulates M as if extra tape were M's input.

QUANTUM TURING MACHINE

The Quantum Turing Machine (QTM) was defined by David Deutsch in 1985.QTM is a precise model of a quantum physical computer. There are two ways of thinking about QTM, (i)the quantum physical analogue of a Probabilistic Turing Machine; (ii) computation as transformation in a space of complex superposition of configuration. The QTM is the most general model for a computing device based on Quantum physic. Like an ordinary Turing machine. A Quantum Turing Machine M consists of a finite control, an infinite tape, and a tape head. a quantum Turing machine is defined to be a quantum system consisting of a processor, a moving head, and a tape, obeying a unitary time evolution determined by local interactions between its components, and allowing to be in a superposition of computational configuration. Deutsch pointed out that the global transition function between computational configurations should be determined by a local transition function which depends only on local configurations.. Bernstein and Vazirani found a simple characterization of the local transition functions for the restricted class of quantum Turing machines in which the head must move either to the right or to the left at each step. Since the above characterization constitutes an alternative definition of quantum Turing machines more tractable in the field of theoretical computer science, it is an interesting problem to find a general characterization valid even when the head is not required to move or more generally when the machines has more than one tape [7,8].

Mathematical Definition of QTM:

A Quantum Turing Machine (QTM) is a 7-tuples

$$\mathbf{M= (Q, \sum, \Gamma, U, } q_0 \mathbf{\ B, F)}$$

Where,

Q is a Finite number of states.
Γ is tape symbols.
$\sum \subseteq$ is an input symbol.
B $\in$ Γ is a blank symbol.
δ is a state transition function and is a q mapping from Q × Γ × Γ × Q × {L,R} to **C.**
$q_0 \in$ Q is a initial state.
F $\subseteq$ Q is a set of final states.

An expression δ(p, a, b, q, d)= **C** represents the following:
If M reads a symbol a in a state a, p (let c_1 be this configuration of M), M writes a symbol on the square under the tape head, changes the state into q , and moves the head one square in the direction denoted by d $\in$ {L, R} (let c_2 be this configuration of M), and c is called an amplitude of this event. Then we define the probability that M changes its configuration from c_1 to c_2 to be $|C|^2$. This state transition function δ defines a linear mapping in a linear space of superposition of M's configuration. This linear mapping is specified by the following matrix M's configuration. This linear mapping is specified by the following matrix M_6. Each row and column of M_6 is corresponds to configuration of M. Let c_1 and c_2 be configuration of M, then the entry corresponds to c_2 row and c_1 column of M_6 is δ evaluated at the tuples which transforms c_1 and c_2 in a single step. If no such tuples exists, then the corresponding entry will be Zero. We call this matrix M_6 *a time evolution matrix* of M. With a Quantum Turing machine, δ associates with each state and symbol, and each possible next move, a complex probability amplitude (which we require to be a feasible complex number). We also require that the linear transformation defined by the machine is unitary. BQP is the collection of languages L recognized by a Quantum Turing machine, running in polynomial time, under the bounded probability rule. The class BQP is not changed if we restrict the set of possible amplitudes to {0, ±2/3, ±4/5, 1} BPP ⊆ BQP. Shor has shown that the factorization problem is in BQP. It is not known to be in BPP.

QTM can be considered as a physical model:

A quantum Turing machine is a quantum system consisting of a processor, a bilateral infinite tape, and a head to read and write a symbol on the tape. Its configuration is determined by the processor configuration q from a finite set Q of symbols, the tape configuration T is represented by an infinite string from a finite set Q of symbols, the tape configuration T is represented by an infinite string from a finite set Σ symbols, and the discretized head position □. Figure 10 is a sample example of QTM considered as a physical system [6,7,8].

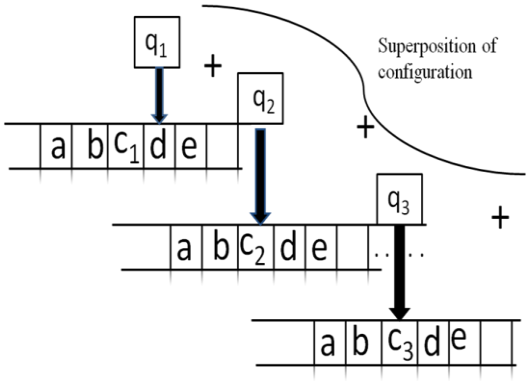


Figure 10: QTM as a physical model

However, various researches are going on throughout the globe to make this hypothesis of Quantum computing to implement it practically like development of a quantum computer. It is still in its infancy stage.

CONCLUSION

This paper presents a comparative analysis on the two computational models, 'Turing Machine' and 'Quantum Turing Machine'. The study is based on computational power of both the models. It shows that a Turing machine can be modified according to application. Based on this, some models have been proposed viz. single-tape, multi-tape, k-headed, multiple etc. However, Quantum Turing machine is better than

© JGRCS 2010, All Rights Reserved

any other computing models based on its computational speed and number of languages accepted. As future works, it would be interesting to work in depth of the Quantum Turing machine and find out its more advantages over traditional computational models.

REFERENCES

[1] Nayak T., Dash T., A Comparative Study on Quantum Pushdown Automata, Turing Machine and Quantum Turing Machine, International Journal of Computer Science and Information Technologies, Vol.-3(1), 2012, pp. 2932-2935.

[2] Nayak T., Dash T., Quantum Finite Automata, Quantum Pushdown Automata & Quantum Turing Machine: A Study, International Journal of Computer Science and Information Technologies, Vol.-3(2), 2012, pp. 3765-3769.

[3] Feynman R. P., Simulating physics with computers, International. J. Theoret. Phys., 21 (1982), pp. 467-488.

[4] Deutsch D., Quantum theory the Church-Turing principle and the universal quantum computer, Proc. Roy. Soc. London Ser. A, 400 (1985), pp. 97-117.

[5] Deutsch D., Quantum computational networks, Proc. Roy. Soc. London Ser. A, 425 (1989), pp. 73-90.

[6] Benioff P., The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines, J. Stat. Phys., 22 (1980), pp. 563-591.

[7] Moore C., Crutchfield J.P., Quantum automata and quantum grammars, Theoret. Comput. Sci. 237 (1–2) (2000) 275–306.

[8] Nielsen M., Chuang I., Quantum Computation and Quantum Information, Cambridge University Press, 2000.

© JGRCS 2010, All Rights Reserved