Turing Machine Simulations: Classic and Quantum

Hafsa Yazdani, Fatemah Al Zayer, Naya Nagy, and Marius Nagy

***Abstract*—The simulation software presented in this paper visualizes abstract machines and concepts. It simulates five different Turing machine (TM) types of two categories: classical Turing machines and quantum Turing machines. The quantum Turing machines implement superposition bothfor the tape cells and for the state of the read/write head. Studentsprogramming our TMs get hands-on experience of naturally non-intuitive quantum theory concepts: superposition, quantum measurement,state collapse. Additionally, the quantum head TM described in this paper is the first TM definition that implements John Deutsch's multi-verse theory. To the best of our knowledge, our program is the firstthat gives users the experience of programming their own multiverses.**

***Index Terms*—Classical turing machines, multiverse theory, quantum turing machine, simulation.**

I. Introduction

In 1936 Alan Turing, a British mathematician and computer scientist, published a paper to present the Turing machine as a model that can translate symbols using a set of rules [1]. The machine contains the elements every modern computer is based upon today, and provides a mathematical base for solving problems using given sets of rules, also known as algorithms [2]. Additionally, any problem that can be solved by computers can be represented and solved by a Turing machine as well. Before the advent of hypercomputation, a Turing machine had thus been considered a stable and maximal class of computational devices that can take different algorithms for computation [3].

Quantum Turing Machines (QTM) have a more recent history. First proposed in 1985 by Oxford physicist David Deutsch, they expand upon the principle of classical Turing machines by allowing quantum transition functions and super-positioned configurations [4]. These Turing machines are used to abstract the model of a quantum computer. Such computers are vastly more efficient that classical computers in performing certain computational tasks because they rely on the power of natural quantum phenomena that have no classical analogues [5]. Complex problems such as integer factorization, used in many encryption technologies today, can be solved using quantum computation in polynomial time.

A key point to be noted is that quantum computers are faster than classical computers because of inherent parallel processing; that is, the power to explore several possible solutions to a problem simultaneously. Quantum computers and their abstraction as quantum Turing machines are intriguing in their identity. As some [6] have argued, the computational power of a QTM is not above a classical TM, as a classical TM could simulate *any* QTM computation. Opposite to the above, the attractive idea about QTMs is that some problems may be solved only on a QTM and *cannot* be simulated by a classical TM [7]. Thus a QTM has an inherently different computational identity than a classical TM. It is a member of the field of hypercomputation [8], [9]. We hope that experimenting with our Turing machine simulator allows the user to feel the different capabilities of the TM variants.

Section II describes classical and quantum Turing machines in more detail. Section III A presents two simulations of classical Turing machines: the first with one memory tape and the second with two tapes. The simulations come with demos for the unary addition. Users can define their own programs. Section III B refers to the two QTM simulations with one quantum memory tape and with two quantum memory tapes. Probabilistic binary addition is demonstrated on these machines.

Finally, the fact that quantum computers exist and quantum computation can be performed in parallel is suggested as the evidence for the Multi-verse theory. It is argued that quantum algorithms can perform more intermediate calculations that give a single result than the number of atoms in our universe, and that if there exists only one universe around us, then where were the computations performed? A number of alternate universes, as suggested by this theory, seems to answer this question [5]. In section III C we propose a new model of QTM that has a quantum Read/Write head. This model elegantly simulates a two-verse. The two-verse can be experienced by the user as two parallel universes that coexist and can influence each other. The user, or measuring identity, may be located in one universe or the other, but cannot effectively see (measure) both universes at the same time.

II. Types of Turing Machines

A Classical Turing Machine, as described in the previous section, is an abstract model of computation. In simple terms, a classical Turing Machine consists of a set of rules, a read-write head, and a tape that the machine reads and writes on. The read and write head of the Turing Machine can either read, write or move one cell to the right or left depending on the state that the machine is in and the rule specifies the action to be taken by the machine [10]. The machine moves from one system state to another according to the set of rules in the table. This set of rules makes up the program that will be executed by the Turing Machine. A single rule could be described by several parameters: the current state, the contents of the cell, the next state the

Manuscript received April 10, 2013; revised July 17, 2013.

The authors are with Prince Mohammad Bin Fahd University, KSA (e-mail: falzayer@hotmail.com).

machine must move into, and the action it must perform.

Variations of classical Turing machines exist; they may have multiple input tapes, or may read and write to two input tapes simultaneously. Some of these formats [3] are described as follows:

1) **Multiple Tapes.** This type of Turing machine can take k number of input tapes for the same set of rules. There will be k heads to read and write to the multiple tapes for the machine. Configurations for this type of Turing machine will need to keep track of the state, tape content and the head position for each input tape.
2) **Two Way Infinite Tape.** In this scenario, the tape is infinite in both the right and left direction.
3) **Multiple Heads.**It is also possible to have a machine with one tape and multiple heads.

Quantum Turing machines (QTM) are used to abstract the model of a quantum computer. They have the same building blocks as classical Turing machines. They, too, consist of one or more infinite tapes that can extend in one or both directions, with a head that can read and write on the tape.

In a QTM the memory cells do not hold bits, but quantum bits, inshort qubits. A qubit can hold the value 0, the value 1, or a superposition of both values 0 and 1 [11]. A qubit is described algebraically as

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

where $|\alpha|^2 + |\beta|^2 = 1$. Thus, q is in a superposition of 0 and 1, with a $|\alpha|^2$ probability to be 0, and a $|\beta|^2$ probability to be 1. For the purpose of our simulation, we will give the probabilities as percentages. For example, a qubit can be 25% 0 and 75% 1. Therefore, a memory cell is in a quantum state and submits to the laws of quantum physics. The superposition of a memory cell is maintained as long as the system remains unobserved. When the cell is read, the state of the qubit collapses to a classical value of either 0 or 1. The value collapses to one value or the other according to the $|\alpha|^2$ and $|\beta|^2$ probability.

In section III C we define a new version of a Turing machine: the Multiverse Quantum Turing Machine (MQTM). Additionally to having memory cells in quantum states, we also define the read/write head as being defined by a quantum state. Consider a QTM with two tapes. Previously, these tapes required two read/write heads that work independently. The new multiverse QTM has only one quantum read/write head that serves both tapes. The tape can be positioned either on the first tape, on the second tape or it can be positioned in a superposition of both tapes. When the head reads or writes to a cell, the head will collapse to one tape or the other according to the probabilities of the superposition.

The quantum Turing machine outperforms its classical counterpart by performing the computations in parallel and thus reducing the running time significantly. For example, integer factorization is widely used in cryptography as the best known algorithms need exponential time.With quantum computation, however, this problem has a polynomial time solution given byShor's algorithm [12].

III. Implementation

For our simulation program, we used a Microsoft Visual Studio Windows Presentation Foundation (WPF) project coded in the object-oriented language C#. The program is also connected to the Oracle 10g Express database to store user preferences and Turing machines (programs). When run, the user has the following options:

1) Start a demo of existing Turing machines (both classical and quantum), such as the addition of unary numbers. This demo includes the options to move between each state step-by-step, or to view the entire working of the machine continuously.
2) Test out a Turing machine of the users own creativity by entering its set of instructions into a table and subsequently running them. A mechanism is set in place to detect whether the program entered has worked correctly.
3) Study the database for programs used in the past. Users have an option to save their program in the database for later reuse.
4) While a Turing machine is running, a History file records and tracks every step made by the program. This history can besaved in the database for later reference.

Our software defines abstract classes for a generic Turing machine, a generic step of computation and generic rules. For each particular Turing machine, these classes have their particular instances. We simulate five Turing machines:

- Classical Turing machine with one tape.
- Classical Turing machine with two tapes.
- Quantum Turing machine with one quantum memory tape with qubit cells.
- Quantum Turing machine with two quantum tapes.
- Quantum Turing machine with two quantum tapes and a quantum read/write head.

A. Simulations of Classical Turing Machines

We implemented two classical TM simulators: with one memory tape and with two memory tapes.

The one tape classical TM simulator has one tape that extends to infinity to the right. The user may set the content of the tape as wished. New memory cells will be created by the program to the right according to needs.

The demo for this machine performs a unary addition. Unary addition adds two unary numbers by concatenating them. For example: 11 + 111 = 11111. The tape of the TM simulator initially contains the two numbers separated by a space (Fig. 1).

Fig. 1. One tape unary addition before the execution of the program. The two unary numbers are written on the tape with a space in between.

The software allows users to change this program to see different possible behaviours of the machine. Users can also write and enter into the TM their own program.

The two tape classical TM has two tapes and two read/write heads. In each computation step both tapes are taken into consideration. As such, the rule to be applied next is defined by the state of the machine and the contents of the two cells to which the two read/write heads point to.

Initially the memory tapes contain one of the operands each. Fig. 2 shows an example of how the tapes may look. The first operand is 111 on the first tape, and the second operand is 1111 on the second tape.

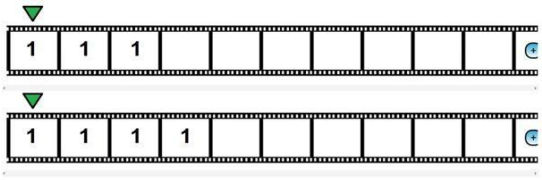

Fig. 2. Two tape unary addition before the execution of the program. Each tape contains one of the two operands to the addition.

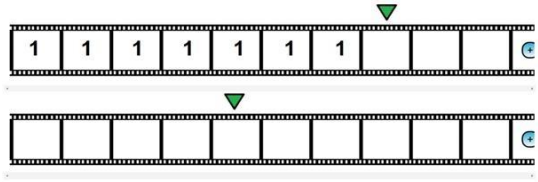

Fig. 3. Two tape unary addition after the execution of the program.

As the execution of the program completes, the result of the addition is stored on the first tape, while the second tape is empty. This can be seen in Fig. 3 where the result is 1111111.

Note that the number of computation steps that the two tape TM has to perform is less than the number of computation steps for the one tape TM.

B. Simulations of Quantum Turing Machines

The quantum TMs differ from the classical TMs described above in the definition of the memory tape. A quantum memory tape is still an infinite tape consisting of memory cells. In our simulations we consider only one-way infinite tapes. The content of a QTM memory cell is different from a classical TM cell. It contains a qubit. As such, the value of a QTM cell can be 0, 1, or a superposition of 0 and 1. Fig. 4 shows a possible quantum memory tape. Note the difference in color: 1 is a fully white cell, and 0 is a fully black cell. The value of the superposition of a cell is shown by the gradient from white to black. For example, the second cell has a 75% 0 and 25% 1. The third cell, has a 10% 0 and 90% 1. The value of the superposition can be set by the user and can be checked later, by clicking on the memory cell.

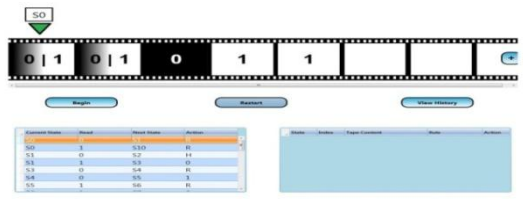

Fig. 4. Quantum binary addition before the start of the program. The first operand is a two digit quantum number. Depending on the superposition, this first operand may collapse to different values at reading. The second operand is the classical 11.

We simulate QTM with one tape and with two tapes. On each QTM we have a demo of a quantum binary addition. Quantum binary addition will have as operands binary numbers in superposition. Thus, the value of the operand depends probabilistically on the reading, when the superposition collapses to a classical binary value.

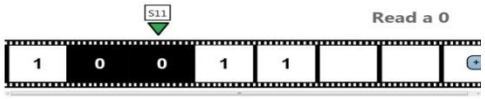

Fig. 5. The tape of the quantum binary addition after the first operand was read. The first operand has collapsed to the classical value 10.

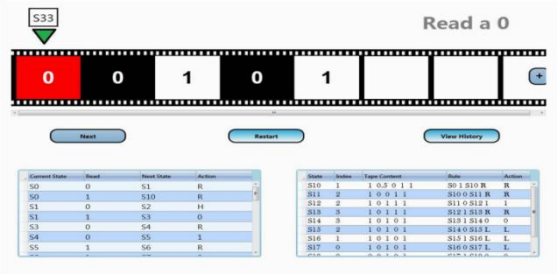

Fig. 6. The QTM halted after binary adding 10 and 11.

The figure shows the QTM before the start of the program. The program adds two numbers of two qubits each. The first operand is a duplet of two qubits in superposition, while the second operand is, for simplicity, the classical binary number 11. Note that the first two cells of the memory tape have a gradient of black and white. The superposition for both qubits is 50% 0 and 50% 1. When the first operand is read it can collapse to four possible binary numbers: 00, 01, 10, or 11. Therefore the outcome of the addition will be one of the four possible additions. We see here an example of quantum parallelism where all possible operands are considered initially, but only one is actually computed at one execution of the program. The program may have different outputs for different executions.

Fig. 5 shows a possible collapse of the first operand. Note that the machine reached state eleven (*S*11). The first operand has probabilistically collapsed to the classical value 10. Fig. 6 shows the QTM after halting. The result of the binary addition is 10 + 11 = 101. Note the History of the program on the lower right hand side of the screen.

The two tape quantum Turing machine has two identical quantum tapes. It can access, read and write, both tapes in one computation step, andtherefore the computations are faster.

An example for our two-tape QTM involves thebinary addition of two numbers, each two digits long. The first operand is on the first tape, a classical 00, while the second operand is on the second tape, a duplet of two qubits in superposition. Fig. 7 shows the tapes before the computation. The binary addition with two tapes perform less computation steps than its one tape counterpart.

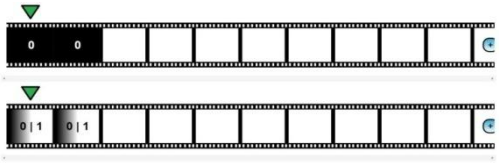

Fig. 7. The two tapes of the QTM before they perform a binary addition on two two-digit binary numbers. The first operand is 00 on the first tape, the second operand is on the second tape, a superposition of two qubits.

C. The Multiverse Quantum Turing Machine

The new model we propose has a read/write head with quantum properties. Because the definition of this Turing machine is directly linked to the multiverse theory we call the model the Multiverse Quantum Turing machine (MQTM). To the best of our knowledge, the read/write head has not been defined with quantum properties before.

All Turing machines described in this paper before had one read/write head per tape. At any moment, a read/write head had a definite position, pointing to exactly one cell. In one computation step, the head could read the cell, write the cell, and possibly move either left or right to the next cell.

The MQTM has two quantum tapes, identical to the tapes defined in the previous subsection. Nevertheless, these two tapes are served only by one head. This quantum head moves to the left and right among the cells like a classical head. The difference is that it can point to the first memory tape, to the second memory tape, or its state can be a superposition of pointing to both tapes. Like every quantum state, the pointing direction of the head is defined as a percentage pointing to the first tape and the rest pointing to the second tape. Fig. 8 shows the two quantum tapes with one (green) head pointing in both directions.

As long as the head only moves left or right, meaning it does not read, nor write, the pointing state is maintained in superposition. When the head reads or writes, the system performs a measurement and the pinpointing of the head collapses to either one tape or the other. The head will stay in the collapsed state until its pointing state is explicitly set by the program. The quantum head can perform the following operations:

1) Move left or right.
2) Read the content of a cell. In this case the pointing of the head collapses to one tape or the other.
3) Write the content of a cell. In this case the pointing of the head also collapses.
4) Set the pointing of the head to an arbitrary superposition of first and second tape.

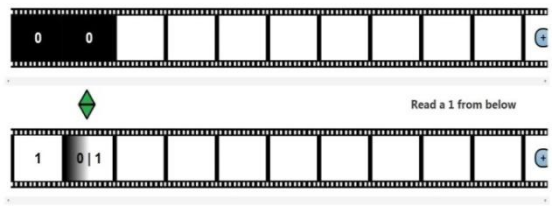

Fig. 8. Two quantum tapes with a quantum read/write head.

Note that the tape can see only one tape at a time. If we view the memory tape as a universe of information, then the head can measurably exist only in one universe. Also, the head can act, meaning write only in one universe. Nevertheless the head can probabilistically shift from one universe to the other if its state is in superposition.

This model is in an experimental stage.

IV. Conclusion

This paper showed simulations of various Turing machines models. In particular, we simulate two classical Turing machine models and three quantum Turing machine models. The multiverse quantum Turing machine is the first of its kind. By allowing the read/write head to have quantum properties, the multiverse QTM simulates the multiverse theory.

The simulations are designed to be user friendly. The following are advantages of our software:

- The user can run demos on the machines, but also define their own program.
- Abstract concepts of quantum mechanics are presented in a tangible form.
- The program can be executed step by step or in one go.
- The machine has a graphically pleasing aspect with the current state of the machine clearly visible.
- The program that the machine runs is visible on the screen at all times.
- The software keeps track of the history of the execution.
- Programs can be stored for later usage.

This software is suitable for students attempting to learn Turing Machines and to experience various common and uncommon Turing machine types. We feel that the main advantage of our software is to present the difference between classical and quantum principles as applied to Turing machines in a user-friendly way. It is a well known fact that quantum mechanical laws are non-intuitive and difficult to grasp. Therefore a graphical user interface may be a legitimate choice for understanding.

Extensions to our implementation are possible in several directions. Our software could be extended towards other classical Turing machine types: multidimensional tapes rather than linear tapes, multiple heads per tape, etc. Quantum Turing machines also allow for the same extensions as for the classical TMs. Additionally, quantum properties were defined for the memory cells and for the pointing of the read/write head. It would be interesting to experiment with other components of the Turing machine with quantum added functionality. Also in this paper we considered superposition as the only quantum property. Quantum entanglement is another option to be explored.

References

[1] A. M. Turing, "On computable numbers, with an application to the Entscheidungsproblem," in *Proc. the London Mathematical Society*, vol. 42, no. 2, pp. 230–265, 1937.
[2] G. Benenti, "Introduction to classical computation," in *Principles of Quantum Computation and Information*, pp. 9 – 64, *World Scientific*, River Edge, NJ, USA, 2004.
[3] H. Lewis and C. Papadimitriou, "Turing machines," Alan Apt Ed., ch. 4, *Elements of the Theory of Computation*, pp. 179 – 215., USA, 1998.
[4] M. Giunti, "Generalized computational systems," ch. 2, Cary Ed., *Computation, Dynamic, and Cognition*, pp. 55 – 112, Oxford University Press, NC, USA, 2997.
[5] S. Perdrix and P. Jorrand, "Turing machines," First International Workshop on Developments in Computational Models, *Mathematical Structures in Computer Science*, pp. 86 – 90, Lisbon, Portugal, 2005.
[6] D. Deutsch, "Quantum theory, the Church Turing principle, and the universal quantum computer," in *Proc. the Royal Society of London*, pp. 97–117, 1985.
[7] S. G. Akl, "Three counterexamples to dispel the myth of the universal computer," *Parallel Processing Letters*, vol. 16, no. 3, pp. 381–403, September 2006.

[8] T. D. Kieu, "Computing the non-computable," *Contemporary Physics*, vol. 44, no. 1, pp. 51–71, 2003.
[9] A. Syropoulos, *Hypercomputation: Computing beyond the ChurchTuring Barrier*, New York: Springer, 2008.
[10] S. P. E. Xavier, "Turing machines," *Theory of Automata, Formal Languages and Computation*, pp. 187 – 200, Delhi, IND, New Age International, 2004.
[11] S. Dasgupta, C. Papadimitriou, and U. Vazirani, "Quantum algorithms," ch. 10, *Algorithms*, pp. 311 – 320, McGraw Hill, 2006.
[12] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," *FOCS*, pp. 124–134, IEEE Computer Society, 1994.

Hafsa Yazdani was born in Hyderabad, India in the year 1990. After graduating from Prince Mohammad Bin Fahd University in Khobar, Saudi Arabia in 2012 with a bachelor's degree in Computer Science, she completed a summer internship at Schlumberger Dhahran Carbonate Research Center (SDCR) and is currently working as a junior software developer in Dammam, Saudi Arabia. Her interests are centred in theoretical computer science, and she particularly enjoys delving into the fields of algorithm analysis, formal languages and 3D visualization.

Fatemah Al Zayer was born in Dhahran, Saudi Arabia in 1990. She eceived a Bachelor's of Science degree in Computer Science from Prince Mohammad Bin Fahd University in Khobar, Saudi Arabia in 2012. She completed an internship at Saudi Aramco in the Plant Maintenance and Quality Management Division under the Hydrocarbon Application Department. She is currently working as an English teacher. Her research interests include databases, artificial intelligence, and Android application development.

Naya Nagy is an assistant professor at Prince Mohammad Bin Fahd University, Saudi Arabia. She received her PhD from Queen's University Canada. Her research interests are quantum computation, cryptography, security, and parallel and distributed algorithms. She is currently the Associate Chair of the College of Computer Engineering and Science at Prince Mohammad Bin Fahd University.

Marius Nagy obtained his M.Sc. and Ph.D. degrees in Computer Science from Queen's University at Kingston, Ontario in 2001 and 2007, respectively. His Ph.D. thesis focuses on novel applications of quantum mechanics into information processing and data security.

Dr. Nagy continued as a post-doctoral fellow in the School of Computing at Queen's University until the end of 2008. In 2009, he took a position as sessional instructor in the Royal Military College of Canada, Department of Math and Computer Science. Since February 2010, Dr. Nagy was appointed as Assistant Professor in the College of Computer Engineering and Science at Prince Mohammad Bin Fahd University, Al Khobar, Saudi Arabia.