ISSN:0975-9646

Quantum Finite Automata, Quantum Pushdown Automata & Quantum Turing Machine: A Study

Tanistha Nayak, Tirtharaj Dash

National Institute of Science and Technology
Berhampur-761008, India

***Abstract*— An important question of quantum computing is that whether there is a computational gap between the models that is allowed to use quantum effect and a models that does not. Several types of Quantum computational models has been proposed including quantum finite automata, Quantum pushdown Automata, Quantum branching programs. It has been shown that some computational models are more powerful than classical counterpart. In this paper we have compared the power among Quantum finite automata, pushdown Automata, turing machine.**

***Keywords*— Quantum Finite Automata, Quantum Pushdown Automata, Quantum Turing Machine, Computational Model.**

I. INTRODUCTION

Bit(Binary Digit) is the smallest unit of information within computer ,the only thing that computer can understand.Bit is a basic unit of classical computer.Bit has one of two values i.e off(0),on(1).Bits are very exact.it is very easy to tell difference between two.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 exsist 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 exsist in any superposition of the 2 basis states.Quantum entanglement is a form of quantum superposition.

II. METHODOLOGY

A Quantum system that is not interact with the environment evolves an unitary system.Its evolution in time step is given by unitary linear operation.Such an operator is describe by a matrix U such that UU*=I where U* is the conjugate transpose of U.

Quantum Automata can be described as (i)quantum model of automata (ii)finite alphabet and finite states(iii)transition are unitary matrix(iv)superposition of states(v)bounded error-ϵ.Finite automata can be classified into four types.(i)Deterministic Finite Automata(DFA) (ii) 2-way deterministic finite automata(2DFA) (iii) Probabilistic Finite Automata(PFA) (iv) 2-way probabilistic finite automata(2PFA) (v) Nondeterministic Finite Automata(NFA). DFA can be described as basic computation model, read-only machine, finite alphabet, finite state, transition function, regular languages. Coming to 2-way deterministic finite automata (2DFA) is same as deterministic finite automata (DFA) but it has 2-way move head(left,right,stationary).It is more powerful than deterministic finite automata in case of simple computation .Probability finite automata(PFA) has following characteristics.(i)transition are probabilistic not deterministic (ii) Bounded error-□ (iii) strings are accepted or rejected with 1-□ probability (IV) any PDA with □ is equivalent to DFA. Another type of finite automata is 2-way probability finite automata (2-PFA).This language is detected by Freivalds machines.It has two way move head.2-Probabilistic Deterministic Automata (PDA) polytime is equivalent to regular language. In another study, 2-way Probabilistic Deterministic Automata (2PDA) can be considered more concise than 1-way probabilistic deterministic automata (1PDA).

1. Quantum finite Automata(QFA)

Quantum finite automata can be classified into three types. (1)1-way quantum finite automata (ii) 1.5 way quantum finite automata (iii) 2-way quantum finite automata.1-way quantum finite automata (1-QFA) has six (6) tuples. The first ever developed quantum automatons are 1-way quantum finite automaton and 2-way quantum finite automaton. It can be described as $M=(Q,\sum,\delta,q_0,q_{acc},q_{rej})$ where Qis a finite set of states, $\sum$ is an input alphabet,δ is transition function,q_0 is starting state,q_{acc} □ Q is accepting state,q_{rej} is reject state.1-way quantum finite automata is a very reasonable model of computation and also its implementation is very easy. The finite dimensional state-space of a QFA corresponds to a system with finitely many particles.Each letter has a corresponding unitary transformation the state-space. A device is used to read symbols from the input and apply the corresponding transformations to quantum mechanical part.Language accepted by 1-way quantum finite automaton has to obey two rules.(i)A 1-way quantum finite state automata(1-qfa) M is said to be accept the language L with cut-point λ if for all w □ L the probability of M accepting w greater than λ and for all w ∉ L the probability of M accepting

w is at most λ.(ii) A 1- way quantum finite automata(1-qfa) M accepts L with bounded error if there exists an □ > 0 such that for all w □ L the probability of M accepting w greater than λ + □ and for all w ∉ L, the probability of M accepting w is less than λ - □,here we call □ is the margin. .On a study it is found that the languages recognised by 1-way QFAs is a proper subset of regular language.If L is a language recognized by 1-way quantum finite automata(1-QFA) with N states than it can be recognized by a 1-way Deterministic Finite Automata(1-DFA) with $2^{o(N)}$ states. So transformation of 1-QFA into classical counterpart can cause exponentially increase in its size.thoughh the 1-QFA is exponentially smaller than its classical counterpart but the language recognised by 1-QFA is proper subset of regular language.Though space efficiengt 1-QFA is not much powerful than its classical part.In fact 1-QFA is strictly lass powerful than 1-way deterministic Finite Automata(1-DFA).Despite being limited some situations, it performs well in other situation.For removing the drawback of 1-QFA 2-way Quantum finite Automata is developed.Whereas 2-way quantum finite automata(2-QFA) consists of a finite control and a 2 way tape head which scans a read only tape. Formally it can be described as $M=(Q,\sum,\delta,q_0,q_{acc},q_{rej})$ where Qis a finite set of states, ∑ is an input alphabet,δ is transition function,q_0 is starting state,q_{acc} □ Q is accepting state, q_{rej} □ Q is reject state.Q_{non} is non halting state where Qnon = Q – (Qa U Qr).The tape alphabet Γ=∑ U {#,$} where # and $ are left and right end markers respectively.The transition function is δ:Q × T × Q ×{-1,0,1}→□ if q and q' □ Q, σ □T,d □ {-1,0,1}.Then δ(q,σ,q',d) represents the amplitude with which a machine in state q moves to state q' by scanning the symbol σ and with move direction d. The configurations of a 2qfa running on a string *w* are pairs of the form □*q, x*>, where *q* is the state and *x* is the head position. Initially, the head is on the left endmarker, and the machine is in the configuration □q_0, 0>. At later steps of the computation, due to its quantum nature, the machine may exist in superposition of more thanone configuration. It is sometimes useful to visualize such superposition of multiple configurations as a snapshot of the machine running in multiple parallel computationalpaths or branches. Since the coefficients (amplitudes) in this superposition can benegative or even complex numbers, these paths may interfere with each other in waysquite unlike what can happen with classical probabilistic machines. In each step, the machine makes two linear operations: The first one is a unitary transformation of the current superposition according to *δ*, and the second one is an orthogonal measurement of the new configuration to see whether the machine has accepted, rejected, or not halted yet. In all 2qfa's described here, every transition entering the same state involves the tape head moving in the same direction (left, right, or stationary).We represent this feature of q using appropriate one of the notations $\vec{q},\bar{q},\vec{q}$ for this state in this machine descriptions, With this description the syntactically correct 2qfa can be specified easily by just providing a unitary operator $V\sigma: l_2 \rightarrow l_2$ for each symbol σ ε Γ. More formally σ(q,σ,q',d)= □ q'□ **V** $_\sigma$□q> is the amplitude with which the machine currently in state q and is scanning the symbol σ will jump to state q' and more the head in the direction d.Here σ(q,σ,q',d)= □ q'□ **V** $_\sigma$□q> denotes the coefficient of σ(q,σ,q',d)= □ q'□ **V** $_\sigma$□q> and d ε{-1,0,1}in the direction of the tape head determined by q'. The observable describing the measurement process that is executed at each step of the computation is designed so that the outcome of any observation is \accept", \reject", or \non{halting". The machine continues running from normalized superposition from{halting configurations until no non{halting configurations remain. The overall acceptance probability of the input string is the sum of the probabilities of the computational branches which end with accepting configurations. For any regular language, there exist a 2qfa which recognises the language in linear time. For any bound □ >0 there exists a 2qfa which recognises the non-regular language L_{eq} with error bounded by □ in linear time. Let us consider a language L= $\{a^n b^m$ □ m, n>0}. Here the machine first try to validate the input string in form of $\{a^n b^n$ □ m, n>0}, rejecting state or halting otherwise. If this first stage is passed without rejection, the string is over right end marker $, and the computation branches to the N=1/□ paths, each of which the head travels on different speed on a's and b's towards the left end-marker. If number of a's and b's are not equal then the each computation parts arrives at □ at a different time and therefore branches to the accept state with 1/N probability and to some reject state with 1-1/N probability. Therefore the overall acceptance probability is 1/N and the overall rejection probability is 1-1/N. If number of a's and b's are qual then all paths reach □ at same time. The transitions implementing the QFT are arranged in such a way that, in this case, all the rejecting configurations have amplitude zero, and the machine accepts with probability 1.For error bound □ and input string w, a machine built according to this method has $O((\frac{1}{\epsilon^2}))$ and halts within O(1/ □|w|) steps.2-way quantum finite automata is remarkably more powerful than 1-way quantum finite automata and also can recognise context-free languages. We discuss about the first implementation of quantum computer then it may not be fully quantum mechanical. There may be a classical part and a quantum part where the quantum part is definitely more expensive than the classical part. But the model of 2-QFA is not quite consistent with this idea. It allows superposition where different parts of superposition have the head of QFA at different locations. Hence the number of quantum states necessary to implement a 2-QFA is not constant but grows when the size of the input grows.1.5-way quantum finite automata (1.5-qfa) is a special case of 2qfa which cannot move its head to left. 1.5-way quantum finite automata were defined by Amano and lawma. It was shown that 1.5-way can recognized some non-context free languages. A 1.5 quantum finite state automaton can be defined as having 6 tuples. $M=(Q,\sum,\delta,q_0,q_{acc},q_{rej})$ where Q is a finite set of states, ∑ is an input alphabet, δ is transition function,q_0 is starting state , q_{acc} □ Q is accepting state, q_{rej} is reject state. The powers of QFAs can be views as a van diagram as given in Fig-1.

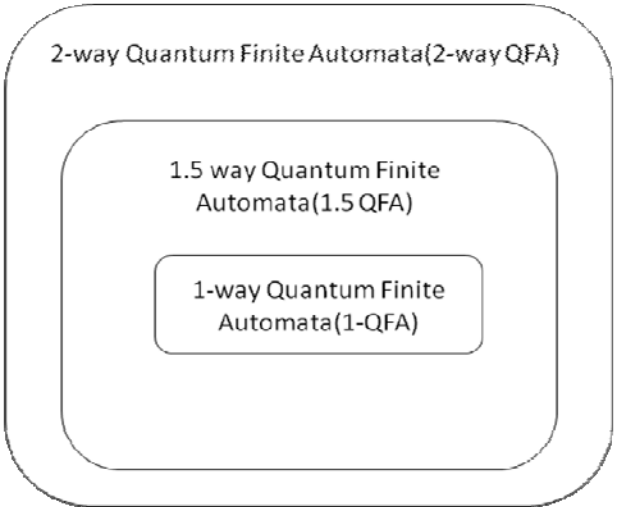

Fig 1: Comparsion among 1-way, 1.5-way and 2-way QFA

As discussed, the power among the languages,Let us discuss efficient the construction of two-way Quantum Finite Automata(2-QFA). There are various methods that will describe the efficient construction of 2-way quantum finite automata such as (i) N-way branching with a single pass (ii) two-way branching with multiple Pass (iii) two-way branching with multiple Pass with collision Avoidance .

2. Quantum Pushdown Automata (QPDA)

One of important questions of quantum computing is that whether there is a computational gap between the models that is allowed to use quantum effect and the models that is not. Several types of Quantum computational models has been proposed including quantum finite automata,Quantum pushdown Automata , Quantum branching programs. It has been shown that some computational models are more powerful than classical counterpart.There are some models which are not more powerful than classical counterpart since they have to obey some restrictions such as reversible state transition. It has been shown that Quantum Pushdown Automata can recognize every regular language and some contextfree language. For this they have a quantum tape head, a quanutm stack and needs O(n) qubits for realization,where n is an execution time. There are several types of Quantum pushdown Automata(QPDA) such as

(i) Quantum Pushdown Automata with 1-counter

(ii)Quantum Pushdown Automata with multiple counter

(iii)Quantum Pushdown Automata with classical stack (QCPA). In this following section we discuss briefly about Quantum Pushdown Automata(QCPA) . QCPA is a quantum automaton whose stack is implemented as classical device. It needs $\lceil \log m \rceil$ qbits for specifying the position of head and also the constant number of qbits for representing finite state control where m is input length. The Quantum part of Quantum Pushdown Automata with classical counterpart is known as 1.5-way quantum finite Automata.1.5-way quantum finite automata can recognise one of non-context free language with probability less than $\frac{2}{3}$. QCPA recognise some non- contextfree language as well as every deterministic contextfree language. L_1={ $a^n db^n dc^n$ | n >0} and L_2={ $a^i db^j dc^k$ | i ≥ j =k}. L_1 and L_2 are not context free languages by pumping lemma. QCPA can also recognise Deterministic Pushdown Automata. Let M=(Q,∑,Γ,δ,q0,qacc) be Deterministic Pushdown Automata where Q is the set of states, ∑ is the set of input symbols, Γ is the set of stack symbols, δ is the transition function where $\delta: Q\times\sum\times\Gamma \rightarrow Q\times\Gamma^*\times\{0,1\}$, q_0 is the initial state and q_{acc} is the set of accepting state. Since M is a Deterministic Pushdown Automata, tape head moves to right on each step. We make a reversible pushdown Automaton by adding extra states and stack symbol where reversible means each state has one incoming transition for incoming transition for each stack symbol. Suppose that a state q has multiple incoming transition $t:\delta(q',a,\$) = (q_t,w,1)$.Here we define a new state q_t and also a new stack symbol s_t to δ. We also define δ as $\delta(q',a,\$)=(q_t,s_t,0)$ and $\delta(q_t,s_t,0) = (q_t,s_t,1)$.thus the resulting pushdown automata is reversible. And that's why all deterministic pushdown Automata is accepted by QCPA. Quantum Pushdown Automata contains seven tuples. A = (Q, Σ, q_0, Q_{acc}, Q_{rej}, δ) Where Q: A finite of state, like the states of a finite automaton. Σ: A finite of input symbols, also analogous to the corresponding component of a finite automaton. Δ=T U {Z0} is the working set alphabet of A and Z0 ⊄ T is the stack base symbol {↓, →} = is the set of direction of input tape head. The automaton must satisfy condition of well-formed ness which will be expressed below. Furthermore the transition function is restricted to further requirement.

Conditions for Quantum Pushdown Automata:

If δ (q, α, β, q', d, ω) ≠ 0, then 1.| ω|<=2; 2. If |ω| =2, then ω1=β; 3. If β=Z_0, then ω ε (Z_0T)*; 4. If β ≠ Z_0, then ω ε T*.

Ŕ: A finite stack alphabet. This component, which has no finite automaton analogy, is the set of symbols that we are allowed to push onto the stack. δ: Q × Γ × Δ ×Q×{↓,→} × Δ* → ⊄ [0,1]; Where Γ= Σ U { # , $} is the tape alphabet of A #, $=end markers not in the Γ. Generally in Pushdown Automata (PDA), we read all the characters in a string and after that we get we can know that whether that particular string will accepted by PDA automaton or not, but in QPDA we generally have states in such a way that after reading one or two character in a string we can know that whether that particular string will be accepted or not which shows that it will consume less time and effective in nature. The main difference between PDA and QPDA is that we have direction in case QPDA for which we know easily whether the particular string will be accepted by the automaton with less time.

3. Quantum Turing Machine (QTM)

A QTM is an abstract machine used to model the effect of a quantum computer. Normal Turing machine can only

perform one calculation at a time whereas a QTM can perform multiple calculations at a time. Normal computer works by manipulating bits in which there exists two states (0 or 1), but Quantum Computers are not limited to two states. They encode information as Quantum Bits (Qubit) which exist in superposition. Qubit represent atoms, ions, photons, electrons. The quantum Turing machine (QTM) has been introduced by Deutsch as the very first model of quantum computation. A quantum Turing machine can be seen as a generalisation of a probabilistic Turing machine where the probabilities which are associated with any transition are replaced by *amplitudes* i.e., complex numbers. The use of complex numbers leads to model fundamental phenomena of quantum mechanics like interferences and entanglement. A probabilistic Turing machine is submitted to well-formed ness conditions ensuring that for any configuration, the probabilities of all the possible evolutions sums to one a quantum Turing machine is submitted to similar well-formed ness conditions. These well-formed ness conditions ensures that the evolution of a quantum Turing machine (i) does not violate the law of quantum mechanics; (ii) is reversible. This later condition can be rephrased in more physical terms as an isolation of the quantum Turing machine from its environment during the computation. The reversibility assumption of quantum Turing machines is questionable for several reasons, including for instance the emergence of quantum computing models based on non reversible evolutions, like the one-way model or more generally the measurement-based quantum computations, which point out that a quantum computation is not necessary reversible. Moreover, the isolation assumption leads to technical issues like the impossibility to know whether a running computation is terminated or not i.e., whether the halting state is reached or not. Finally, due to the isolation assumption, quantum Turing machines are the natural quantum versions of reversible Turing machines. But the natural embedding of any reversible Turing machine into a quantum Turing machine cannot be extended to non-reversible and probabilistic Turing machines.

Computational power of QTM:

Let M a stationary, normal form, multi-track QTM whose last track has alphabet {#, 0, 1}. If we run M with string x in the first track and the empty string elsewhere, wait until M halts and then observe the last track of the start cell: we will see a 1 with probability p. We will say that M accepts x with probability p and rejects x with probability 1 − p. We say that a QTM M accepts a language L with probability p, if M accepts with probability at least p every string x □ L, and rejects with probability at-least p every string x ∉ L.

The power of acceptance of the three machines (QFA, QPDA and QTM) is being compared above and this can be represented as a van diagram. The representation is given in Fig-2. The figure shows that QTM is capable of accepting more number of languages than QPDA or QFA. The diagram resembles with the Chomsky hierarchy of computation.

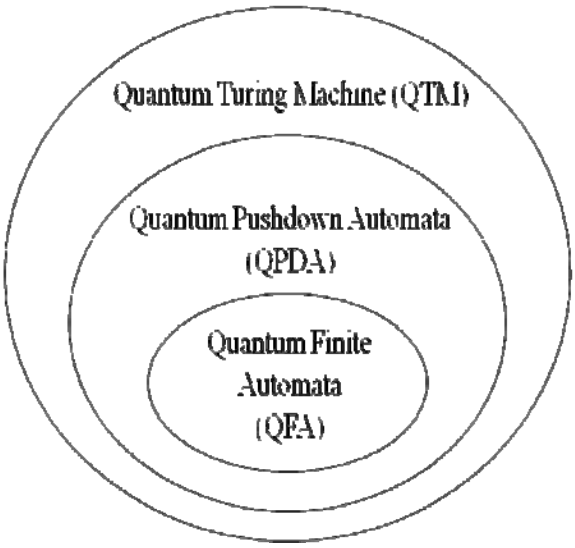

Fig 2: Comparison of Power among QFA, QPDA, QTM.

III. DISCUSSION

As this work is a comparative study on three powerful machines it will be worth to compare the three machines based on the acceptance timing. For this purpose, a commonly accepted language L={$a^n b|n\geq 0$} has been taken, because this language can be accepted by different values of n. For different values of n, starting with n=0 to n=10000 are being tested. It should be noted that the coding for all the machines (QFA, QPDA and QTM) has been done in C-language and executed in a system with 1GB RAM. The processor is a dual-core and the environment is LINUX. The results obtained from the above are being tabled as in Table-1. It is worth to note that the time is calculated as the average of 20 runs for each of the machines.

Table-1: Result from the testing of L= {$a^n b|n\geq 0$}

Value of **n**	Time of Execution (sec.)		
	QFA	QPDA	QTM
25	0.00	0.00	0.00
50	0.00	0.00	0.00
150	2.88	3.01	2.56
200	3.05	3.44	2.78
300	5.33	5.18	3.92
1000	10.27	14.21	9.77
2000	23.81	25.11	24.19
5000	37.16	35.66	34.29
10000	87.73	96.18	64.95

The table shows that till n=1000, all the three acceptors works fine with little computation time difference between them. When the value of n increases to 10000 through 2000, the computation times of the three language acceptors are making a significant difference. A plot has been given in figure-3 to view the difference in time of execution.

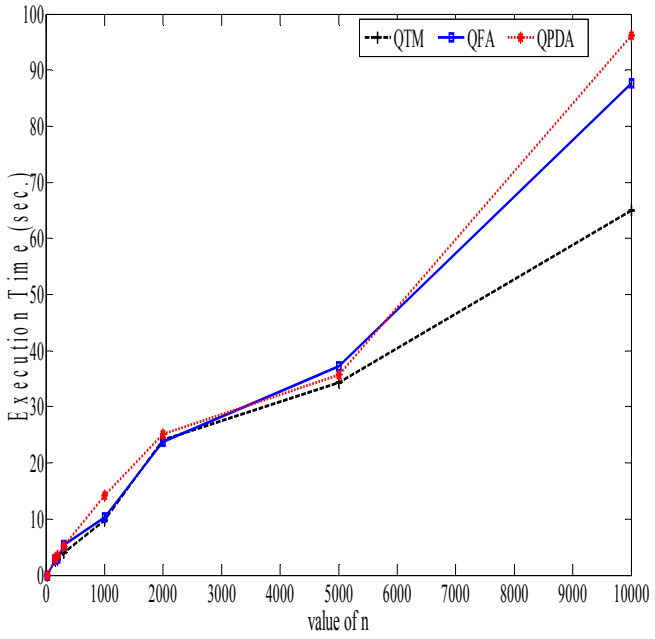

Fig-3: Plot showing time taken by QFA, QPDA and QTM to recognize the language L= $\{a^n b | n \geq 0\}$

From the above plot, it is clear that from n=5000 to n=10000, QTM is performing better as compared to other twos. QPDA is taking some more time to recognize the same language as compared to that of QFA. This is because the push and pop operation of the QPDA. It will be a good approach to optimize all the code manually. For automatic optimization by the compiler '–g' directive is used in the compilation command.

IV. CONCLUSION AND FUTURE WORKS

This work focused on a detailed analysis of three powerful language acceptors being Quantum Finite Automata (QFA), Quantum Pushdown Automata (QPDA) and Quantum Turing Machine (QTM). It was observed that the third type of language acceptor i.e. QTM is capable of accepting more number of language in a comparatively smaller amount of time. QPDA performs better in accepting language than the QFA. All these machines are seen to resemble with the Chomsky hierarchy of computation.

As our future work, it will be interesting to work on these quantum machines using a newly evolving language called as QCL (Quantum Computation Language). It will also be a good approach to compute some algebraic operation using these quantum computing machines.

V. REFERENCES

[1] T. Nayak, T. Dash, 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, 2932-2935.

[2] Olaf Nairz, Markus Arndt, and Anton Zeilinger, "Quantum interference experiments with large molecules", American Journal of Physics, 71 (April 2003) 319-325.

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

[4] Yakaryilmaz A.Cem Say A,C. "Succinct of twoway probabilistic and Quantum finite Automata" computational complexity, 22 dec 2009.

[5] Ambainis A. ,Freivalds R."1-way Quantum finite Automata: strengths, weaknesses and generalization", IEEE Computer Society Washington, DC, USA, ISBN: 0-8186-9172-7.

[6] A. Kondacs, J. Watrous, On the power of quantum finite state automata. Proceedings of the 38th IEEE Conference on Foundations of Computer Science, 66-75, 1997.

[7] D.Â Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer. *Proc. R. Soc. Lond. A*, 400:97–117, 1985.

[8] M.Â A. Nielsen. Universal quantum computation using only projective measurement, quantum memory, and preparation of the 0 state. *Phys. Rev. A*, 308:96–100, 2003.

[9] S.Â Perdrix. Partial observation of quantum Turing machine and weaker well-formedness condition. In *proceedings of Joint Quantum Physics and Logic and Development of Computational Models* (Joint 5th QPL and 4th DCM), 2008.

[10] R.Â Raussendorf, D.Â E. Browne, and H.Â J. Briegel. The one-way quantum computer - a non-network model of quantum computation. *Journal of Modern Optics*, 49:1299, 2002.

[11] A. Kondacs, J.H.Watrous, On the power of quantum finite state automata, in: 38th Annual Symp. Foundations of Computer Science, 1997, pp. 66–75.

[12] . Ambainis, J.H. Watrous, Two-way finite automata with quantum and classical states, Theoret. Comput. Sci. 287 (1) (2002) 299–311.

[13] A. Ambainis, R. Freivalds, 1-way quantum finite automata: strengths, weaknesses and generalizations, in: 39th Annual Symp. Foundations of Computer Science, 1998, pp. 332–341.

[14] T.Yamasaki, H.Kobayashi, H.Imai, "*Quantum versus deterministic counter automata*", Theoret. Comput. Sci. 344 (1) (2005) 275–297.

[15] A. Nayak, Optimal lower bounds for quantum automata and random access codes, in: 40th Annual Symp. Foundations of Computer Science, 1999, pp. 369–376.

[16] M. Kravtsev, Quantum finite one-counter automata, in: SOFSEM □99: Theory and Practice of Informatics, 26th Conf. Current Trends in Theory and Practice of Informatics, Vol. 1725, Lecture Notes in Computer Science, 1999, pp. 431–440.

BIOGRAPHY

Ms. Tanistha Nayak is a final year student of Bachelor of Technology in Department of Information Technology (IT) at National Institute of Science and Technology (NIST), Berhampur, Odisha, India. She is a Research Scholar and her research interests lie in the area of Quantum Computing and Image Processing.

Mr. Tirtharaj Dash is a final year student of Bachelor of Technology in Department of Information Technology (IT) at National Institute of Science and Technology (NIST), Berhampur, Odisha, India. He is a Research Scholar. His research interests lie in the area of Soft Computing, High Performance Computing, Parallel and Distributed Systems, Quantum Computing, Algorithm Design and Analysis, Artificial Intelligence and Pattern Recognition.