ISSN:0975-9646

A Comparative Study on Quantum Pushdown Automata, Turing Machine and Quantum Turing Machine

TANISTHA NAYAK[1], TIRTHARAJ DASH[2]
National Institute of Science and Technology
Berhampur, India

Abstract— **An automaton is a simple model of computer. There are various automata each with its formal definition. Generally an automaton has some form of input, some form of output, internal states may or may not have some from of storage, hardwired not programmable. In this paper we have discussed about Quantum Pushdown Automata, Turing Machine and Quantum Turing Machine and compared the power among these and by taking some interesting examples. Our main objective is to study about these three machines.**

Keywords—**Quantum Pushdown Automata (QPDA), Quantum Turing Machine(QTM), Turing Machine (TM), Quantum computer, Quantum bit (Qubit).**

I. INTRODUCTION

QUANTUM is a discrete quantity of energy proportional in magnitude to the frequency of radiation it represents. Also it can be defined as an analogous discrete amount of any other physical quantity such as momentum or electric charge.

Quantum computer is a device for computation that makes direct use of quantum mechanical phenomena, such as superposition and entanglement, to perform operations on data. Quantum computers are different from traditional computers based on transistors. The basic principle behind quantum computation is that quantum properties can be used to represent data and perform operations on these data. A theoretical model is the quantum Turing machine, also known as the universal quantum computer. Here, we share theoretical similarities with non-deterministic and probabilistic computers, like the ability to be in more than one state simultaneously.

A quantum computer with a given number of qubits is fundamentally different than a classical computer composed of the same number of classical bits. For example, to represent the state of an n-qubit system on a classical computer would require the storage of 2^n complex coefficients. Although this fact may seem to indicate that qubits can hold exponentially more information than their classical counterparts, care must be taken not to overlook the fact that the qubits are only in a probabilistic superposition of all of their states. This means that when the final states of the qubits are measured, they will only be found in one of the possible configurations they were in before measurement. Moreover, it is incorrect to think of the qubits as only being in one particular state before measurement since the fact that they were in a superposition of states before the measurement was made directly affects the possible outcomes of the computation.

Here our objective is to solve some problems of language recognition using QPDA, TM and QTM and compare their time complexity and output.

II. WHAT IS A QUANTUM BIT (QUBIT)?

Qubit is a unit of quantum information. Qubit represents both the state memory and the state of entanglement in a system. As a bit is the basic unit of information in classical computer, a qubit is the basic unit of information in quantum computer. In a quantum computer a number of elemental particles, such as electrons or photons can be used with either their charge or polarization acting as representation of ‘0’ or ‘1’. This particle is known as a qubit. The pure qubit state is a linear superposition of basis state i.e. the qubit can be represented as combination of |0> and |1>.

III. QUANTUM PUSHDOWN AUTOMATA

Quantum Pushdown Automata contains seven tuples.

$$A = (Q, \Sigma, q_0, Q_{acc}, Q_{rej}, \delta)$$

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.

$\Delta = T \cup \{Z0\}$ is the working set alphabet of A and

$Z0 \not\subset T$ is the stack base symbol

$\{\downarrow, \rightarrow\}$ = is the set of direction of input tape head.

The automaton must satisfy condition of well-formedness which will be expressed below. Further more the transition function is restricted to further requirement.

If $\delta(q, \alpha, \beta, q', d, \omega) \neq 0$, then

1. $|\omega| <= 2$;
2. If $|\omega| = 2$, then $\omega 1 = \beta$;

3. If $\beta=Z_0$, then ω ε $(Z_0T)^*$; 4. If $\beta \neq Z_0$, then ω ε T^*.

Ѓ: A finite stack alphabet. This component, which has no finite automaton analog, is the set of symbols that we are allowed to push onto the stack. δ: $Q \times \Gamma \times \Delta \times Q \times \{\downarrow,\rightarrow\} \times \Delta^* \rightarrow \not\subset [0,1]$; Where $\Gamma= \Sigma \cup \{ \#, \$\}$ is the tape alphabet of A #, \$=end markers not in the Γ. We have solved some examples in QPDA which are given below:

Example 1: Let us consider a language L= $\{a^nb^n| n>=0\}$ Q = (q_0; q_1; q_2; q_3; q_4), Qacc = {q_4}, Qrej = {q_2}, Σ= {a; b}; T = {a; b}.Here we have taken the following assumption:

1. D(q_0)=→ D(q_1)=↓ D(q_2)=↓ D(q_3)=→ D(q_4)=↓

If D='→' moves the input tape head one cell forward. If D='↓' then the input tape head remain in the current state.

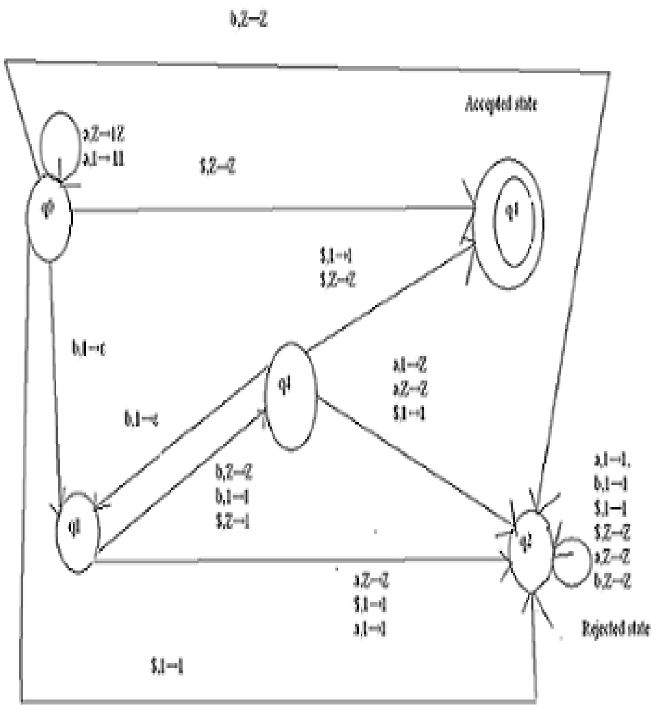

Fig. 1 Representation of L= $\{a^nb^n| n>=0\}$

The transition states of the following problem are mentioned below:

δ (q_0,a,Z) = (q_0,1Z);δ (q_0,a,1) = (q_0,11);δ (q_0,b,Z) = (q_2,Z);δ (q_0,\$,Z) =(q_4,z);δ (q_0,b,1) = (q_1, ε); δ (q_0,\$,1) = (q_2,1);δ (q_1,b,z)=(q_3,z);δ (q_1,b,1)=(q_3,1);(q_1,\$,z)=(q_3,1);δ (q_1,a,z)=(q_2,z);δ (q_1,\$,1)=(q_2,1);δ (q_1,1,1)=(q_2,1);δ (q_2,\$,1)=(q_2,1);δ (q_2,\$,z)=(q_2,z);δ (q_2,a,1)=(q_2,1);δ (q_2,b,1)=(q_2,1);δ (q_2,a,z)=(q_2,z);δ (q_2,b,z)=(q_2,z);δ (q_3,\$,1)=(q_4,1);δ (q_3,\$,Z)=(q_4,Z);δ (q_3,\$,1)=(q_2,1);δ (q_3,a,Z)=(q_2,Z);δ (q_3,a,1)=(q_2,Z);δ (q_3,b,1)=(q_1,ε);

Example 2: L= {w Є (a, b)* ┃ |w | a = |w | b}

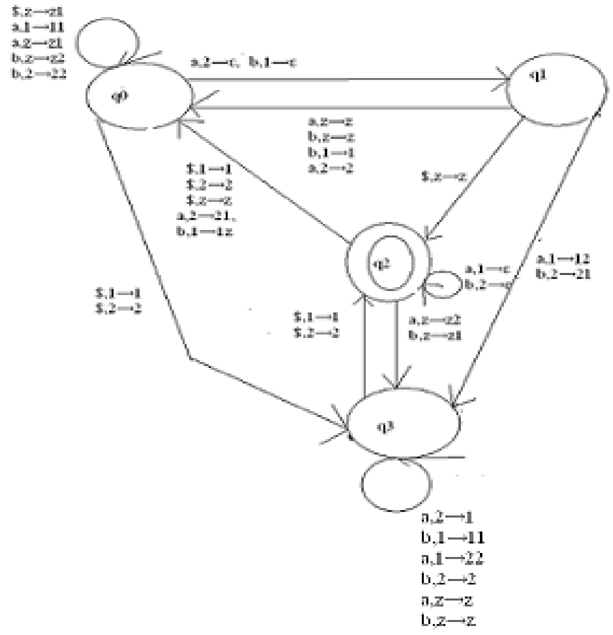

Fig 2: Representation of L={w Є (a, b)* ┃ |w| a = |w | b}

Q = (q_0; q_1; q_2; q_3), Q_{acc} = {q_2};

Q_{rej} = {q_3};

Σ= {a; b};

T = {0; 1}

δ (q_0, a, Z) = (q_0, Z1);

δ (q_0,a,1) = (q_0,11);

δ (q_0,b,Z) = (q_0,Z2);

δ (q_0,b,2) = (q_0,22);

δ (q_0,\$,Z) = (q_0,Z1);

δ (q_0,a,2) = (q_1,□);

δ (q_0,\$,1) = (q_3 ,1);

δ (q_0,\$,2) = (q_0,2);δ (q_1,a,Z) = (q_0,Z); δ (q_1,b,Z) = (q_0,Z); δ (q_1,b,1) = (q_0,1);δ (q_1,a,2) = (q_0,2);δ (q_1,\$,Z) = (q_2,Z); δ (q_1,a,1) = (q_3,12); δ (q_1,b,2) = (q_3,21);δ (q_2,\$,1) = (q_0,1); δ (q_2,\$,2) = (q_0,2);δ (q_2,\$,Z) = (q_0,Z); δ (q_2,a,2) = (q_0,21); δ (q_2,b,1) = (q_0,12); δ (q_2,a,Z) = (q_3,Z2); δ (q_2,b,Z) = (q_3,Z1);δ (q_2,a,1) = (q_2,□);δ (q_2,b,2) = (q_2, □); δ (q_3,\$,1) = (q_2,1);δ (q_3,\$,2) = (q_2,2); δ (q_3,a,2) = (q_3,1);δ (q_3,b,1) = (q_3,11); δ (q_3,a,1) = (q_3,22); δ (q_3,b,2) = (q_3,2);δ (q_3,a,z) = (q_3,z);δ (q_3,b,z) = (q_3,z);

Here we have taken the directions as follows.

1. D (q_0) =→
2. D (q1) =↓
3. D (q_2) =↓
4. D (q_3) =↓

IV. TURING MACHINE

Turing Machine was developed by Alan Turing in 1930. A Turing Machine has a finite state controller, a tape that it can read and write.

i. the tape is unbounded to right
ii. the tape initially holds the input string
iii. the tape beyond the input string is initially filled with an infinite string of blank

The finite state controller has two special states

q_{accept} :if the machine ever reaches the state ,it halts and accept the state
q_{reject}: If the machine ever reaches this state, it halts and rejects the string.
If M is a Turing Machine, then the language recognized by M is written L(M)
and is the set of all strings for which the TM reaches the q_{accept} state.

Turing Machine reads the symbol at its current position on the tape.

i. Based on that symbol and it current state, the machine:
ii. Writes a symbol at the current position;
iii. Transitions to a new state; and
iv. Moves one square to the left or right.

A Turing machine is a 7-tuple (Q, ∑,Γ,q_0, q_{accept} ,q_{reject}) where
Q is a finite set, the states.
∑ is a finite set, the input alphabet.
Γ □ ∑ is a finite set, the tape alphabet.
δ : (Q × Γ) → (Q × Γ × {L,R}) is the transition function.
Q_0 □ Q is the initial state.
q_{accept} □ Q is the accepting state.
q_{reject} □ Q is the rejecting state.

Let us take a language to check the acceptance and time taken by a Turing machine which is given in Figure-3.

$$L=\{ a^n b^n c^n \mid n>=1\} \qquad (1)$$

Input alphabet: _ = {a, b, c}.
States: Q = {q0, q1, q2, q3, q4, q_{accept} , q_{reject} }.
Tape alphabet: = {a, b, c, A, B, C, $}.
Transitions:
q0 :
(q0, a) → (q1, A, right), (q0,$) → (qaccept ,$, right), (q0, B) → (q0, R, right**)**, (q0, C) → (q0, R, right**)**,(q0, other) → (qreject , R, right).
q1 :
(q1, a) → (q1, R, right), (q1, **B**) → (q1, R, right), (q1, b) → (q2, B, right)
(q1, other) → (qreject , R, right).
q2 :
(q2, b) → (q2, R, right), (q2, c) → (q2, R, right) (q2, c) → (q3, C, left)(q2, other) → (qreject , R, right)
q3 :
(q3, $ − {A} → (q3, R, left) (q3, A) → (q0, R, right).

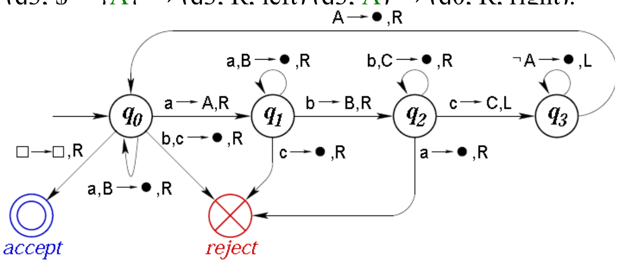

Figure-3: TM for L

An algorithm which is designed to accept L is given by the following Pseudo-code.

```
state = q[0];
while(TRUE)
do
  switch(state)
  do
    case q[0]:
      switch(Current Read Symbol)
      do
        case a:
          write(a);
          state=q[1];
          movement(RIGHT);
          break;
        case b:
          write(b);
          state=q[0];
          movement(RIGHT);
          break;
        case $:
          write($);
          state=q(reject);
          movement(right);
          break;
      end
      break;
    case q[1]:
      /* switch case operations */
    case q[2]:
      /* switch case operations */
    case q(accept):
      accept();
    case q(reject):
      reject();
  end
end
```

A data showing value of n and time taken by the algorithm to recognize the language is given below in Table-1.

TABLE-1 ACCEPTANCE TIME

$a^n b^n c^n$ (Input *n*)	Time Taken (sec)
1	0.000000
10	0.000142
50	0.301357
100	3.589010
150	10.670100
200	23.010067
400	60.090127
500	89.107601

We see that when n is increasing, the complexity of the machine becomes more and it takes much more time to recognize the language. When n is 500 it takes about 90 sec. recognize in a Dual Core processor with 2GB

RAM. It becomes impossible by the machine to accept the same language with n=10000, 20000 and more. A plot has been given in Figure-4 to see how the time changes with **n.**

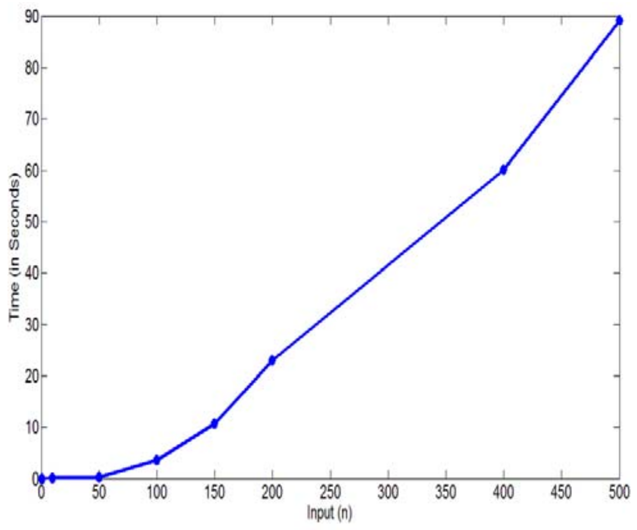

Figure-4: Plot *n* Vs. *Time*

V. QUANTUM TURING MACHINE

A **quantum Turing machine** (**QTM**) is an abstract machine used to model the effect of a quantum computer. It provides a very simple model which captures all of the power of quantum computation. Any quantum algorithm can be expressed formally as a particular quantum Turing machine.

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. Qubits represent atoms, ions, photons, electrons.

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

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

Where,

Q is a Finite number of states.

Γ is tape symbols.

$\sum \ \subseteq$ is an input symbol.

B □ 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 □ Q is a initial state.

$F \subseteq Q$ is a set of final states

If we run the same program which is given by (1), it may take lesser time as compared to the traditional Turing machine.

VI. CONCLUSION AND FUTURE WORKS

In this research work, we studied the three types of language recognizers i.e. Quantum Pushdown Automata, Turing Machine, Quantum Turing Machine and worked out some examples by showing performance of Turing Machine.

As our Future work, we will be working on Quantum Turing Machine to see the performance of it by taking more complex languages.

REFERENCES

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

[2] 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.

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

[4] 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.

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

[6] M. Golovkins, On Quantum Pushdown Automata, School of computing, LNCS, 2000, vol. 1963, pp. 336-346

[7] D. Qiu, P. Mateus, X. Zou, A .Sernadas, One-way quantum finite automata together with classical states: Equivalence and minimization of Computer Science.

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

[9] A. Kitaev, A. Shen, and M. Vyalyi. *Classical and Quantum Computation*, volume 47 of *Graduate Studies in Mathematics*. American Mathematical Society, 2002.

[10] A. Ambainis, R. Bonner, R. Freivalds, A. K¸ ikusts: Probabilities to Accept Languages by Quantum Finite Automata. Lecture Notes in Computer Science, 1999, Vol. 1627, pp. 174-183.

[11] A. R. Calderbank, E. M. Rains, P. W. Shor, N. J. A. Sloane: Quantum error correction via codes over GF(4). IEEE Transactions on Information Theory, 1998, vol. 44, p. 1369-1387.

[12] Deutsch, David (July 1985). "Quantum theory, the Church-Turing principle and the universal quantum computer". *Proceedings of the Royal Society of London; Series A, Mathematical and Physical Sciences* **400** (1818): pp. 97–117.

[13] Bonsor, Kevin, and Jonathan Strickland. "How Quantum Computers Work" 08 December 2000. HowStuffWorks.com. <http://computer.howstuffworks.com/quantum-computer.htm> 05 January 2012.