On Some Aspects of Quantum Computational Models

A Thesis

Submitted in fulfillment of the requirement

for the award of the degree of

DOCTOR OF PHILOSOPHY

IN

COMPUTER SCIENCE & ENGINEERING

By

Amandeep Singh Bhatia
(Registration No. 901503002)

Under the guidance of

Dr. Ajay Kumar
(Associate Professor, CSED)

Computer Science and Engineering Department
Thapar Institute of Engineering and Technology
Patiala - 147004, India

To my family members
for their love, support and encouragement

Certificate

This is to certify that the Thesis entitled "**On Some Aspects of Quantum Computational Models**", submitted by **Amandeep Singh Bhatia**, a research scholar in the *Computer Science and Engineering Department, Thapar Institute of Engineering and Technology, Patiala*, for the award of the degree of **Doctor of Philosophy**, is a record of an original research work carried out by under the supervision of Dr. Ajay Kumar and refers work of other researchers which are duly listed in reference section. The Thesis has fulfilled all requirements as per the regulations of the Institute and in our opinion has reached the standard needed for submission.

The results embodied in this Thesis have not been submitted to any other University or Institute for the award of any degree.

Amandeep Singh Bhatia
Registration No. 901503002

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge and belief.

Dated: 16/7/2020
Place: Patiala

Dr. Ajay Kumar
Associate Professor,
Computer Science and Engineering Department,
Thapar Institute of Engineering and Technology, Patiala,
India - 147004

Acknowledgements

I would like to express my sincere thanks to all those people who made this dissertation possible. First and foremost, I would like to express my profound respect and gratitude to my supervisor, Dr. Ajay Kumar, who has been guiding force behind this work. I am greatly bounded for his constant encouragement, invaluable guidance, and his valuable comments on my work. More importantly, I would like to thank for the patience he has shown in carefully reading and commenting on the manuscripts, and countless revisions of this dissertation. His commitments and dedication to research have been and will continue to be a constant source of inspiration for me. I am fortunate enough to have such an advisor who gave me the freedom to think independently and explore new ideas. I have no doubts that finishing my degree in proper and timely manner was impossible without his help. I am highly privilege to have got an opportunity to work with such a wonderful person.

I would also like to thank my doctoral committee members Dr. Maninder Singh, Dr. Shalini Batra, Dr. Parteek Bhatia and Dr. Kulbir Singh for their invaluable suggestions, encouragements, and moral supports that helped me to improve my research work. I am also thankful to the Head of the Department, other faculty members and staffs for their kind help carried out during my academic studies.

My special thanks to my friend cum moral adviser Dr. Balwinder Sodhi, for his guidance, constant support and insightful comments during the entire journey of my PhD life.

On a personal note, I would like to thank my wife Mandeep Kaur Saggi for her constant support and being with me in every aspect of my life. My special thanks go to, Dr. Sunita Garhwal, for her motivation and support. I had a great time with my many friends at Thapar institute of engineering & technology. I would like to thanks them for their support and encouragement. An special appreciation goes to Dr. Ajay Kumar for correcting the Thesis and giving critical and valuable suggestions.

I am grateful to my parents and in-laws, whose love, blessings, care, encouragement, and support made this research work possible. I would especially thank my Bhaiya, Bhabhi and my most heartfelt love to their children: *Sehajveer*, who is the joy of my life.

I would like to thank the members of CSED Department, faculty members, staff members and my seniors who were always there at need of the hour and provided with the help and facilities, which I required for the completion of my Thesis.

I gratefully acknowledge the funding received towards my PhD from the Maulana Azad National Fellowship by Ministry of Minority Affairs, Government of India. Finally, I would like to thank the Almighty God for bestowing me this opportunity and showering his blessings on me to come out successful against all odds.

Amandeep Singh Bhatia

Abstract

Quantum computing is concerned with computer technology based on the principles of quantum mechanics, in which operations are performed at the quantum level. Quantum computational models make it possible to analyze the resources required for computations. Quantum automata can be classified thusly: quantum finite automata, quantum sequential machine, quantum pushdown automata, quantum Turing machine, and orthomodular lattice-valued automata. These models are useful for determining the expressive power and boundaries of various computational features. In many cases, quantum models are more superior to classical models in terms of language recognition. Thus, mathematical models of quantum computation can be view as generalizations of its physical models. Motivated from the above-mentioned facts, we proposed a variant of two-way quantum finite automata named two-way multihead quantum finite automata and determined its language recognition capability.

Moreover, we have proposed a quantum analogue of classical queue automata by using the definition of the quantum Turing machine and quantum finite-state automata. Further, we have also introduced a generalization of real-time deterministic queue automata, the real-time quantum queue automata which work in real-time i.e., the input head can move towards the right direction only and takes precisely one step per input symbol. We have proved that real-time quantum queue automaton is more superior than its real-time classical variants by using quantum transitions.

Classical systems are not robust and capable of describing quantum systems. Some tasks that are impossible in classical systems can be realized in quantum systems. The most significant property 'entanglement' separates the classical world from the quantum world. It is one of the most central topics in quantum information theory. In fact, quantum many-body systems cannot be simulated by classical systems because of the increase in size of Hilbert space. In this Thesis, we have efficiently simulated well-known examples of matrix product state and designed their quantum finite-state machines (QFSMs) using unitary criteria. Furthermore, the proposed unitary criterion is used to analyze the dynamic behavior of matrix product states with quantum weightless neural networks.

The relationship between the theory of automata and logic had a great influence on computer science. Linear temporal logic is a widely used method for verification of model checking and expressing the system specifications. We have investigated the relationship between quantum finite automata and linear temporal logic. It has a variable free compact syntax, intuitive semantics, and it can be transformed into automata more efficiently. Further, we have characterized the class of languages accepted by quantum finite automata using linear temporal logic formulas.

In the past few years, the research has been focused on applying assets of quantum computing in various areas such as quantum cryptography, quantum machine learning, quantum neural computation, tensor network theory, quantum dynamics to study behavior of biological sequences, weather forecasting, quantum optics, quantum chemistry as well as others. We have modeled various ribonucleic acid

(RNA) secondary structure loops such as hairpin loop, internal loop, and double helix using two-way quantum finite automata. It has been proved that two-way quantum finite automata can be designed for such structure loops in linear time with bounded error.

Next, we have explored and examined the implications of the most successful McEliece cryptosystem in post-quantum cryptography. The proposed cryptosystem is based on extended Golay code in place of the usual Goppa code. Furthermore, we have implemented a McEliece cryptosystem based on extended Golay code using MATLAB.

In recent years, quantum computation is receiving much attention for its capability to solve difficult problems efficiently contrast to classical computers. Specifically, some well-known public-key cryptosystems depend on the difficulty of factoring of large numbers takes a long time. Furthermore, a quantum variant of classical factorization algorithm named Quadratic Sieve has been designed and its simulation framework using high-level programming language Mathematica is constructed. Further, its simulation results are examined on a classical computer to get a feel of the quantum system.

Contents

List of Figures

List of Tables

List of Acronyms

QFT	Quantum Fourier Transform
1QFA	One-way Quantum finite automata
DFA	Deterministic finite automata
PFA	Probabilistic finite automata
2PFA	Two-way probabilistic finite automata
MO-1QFA	Measure-once quantum finite automata
MM-1QFA	Measure-many quantum finite automata
LQFA	Latvian quantum finite automata
CL-1QFA	Quantum finite automata with control languages
1QFAC	One-way finite automata with quantum and classical states.
GQFA	General quantum finite automata
MO-1gQFA	Measure-once general quantum finite automata
MM-1gQFA	Measure-many general quantum finite automata
QCPA	Quantum pushdown automata with the classical stack
1.5 QFA	1.5 quantum finite automata
2QFA	Two-way quantum finite automata
2QCFA	Two-way finite automata with quantum and classical states
rtQ1BCA	Real-time quantum automata with one blind counter
1DkBCA	One-way deterministic automata with k blind counter
1D1BCA	One-way deterministic automata with one blind counter
1Pk1BCA	One-way probabilistic automata with k blind counter
1P1BCA	One-way probabilistic automata with one blind counter
2TFA	Two-tape finite automata
kTQCFA	k-tape automata with quantum and classical states
2TQCFA	Two-way two tape finite automata with quantum and classical states
PAFA	Real-time private alternating finite automata
1Q1CA	One-way quantum one-counter automata
2-1DFA	One-way deterministic two-head finite automata
2-2T1QFA	Two-tape one-way quantum finite automata with two-head
QFSM	Quantum finite-state machine
DMFA	Two-way deterministic multi-head finite automata
k-2MQFA	Two-way k-multihead quantum finite automata
DQA	Deterministic queue automata
QQA	Quantum queue automata
QRA	Quantum Rabin automata
QBA	Quantum Büchi automata
QSA	Quantum Streett automata
RNA	Ribonucleic acid

List of Publications

Journal Publications

1. **Amandeep Singh Bhatia** and Ajay Kumar, Quantifying matrix product state, *Quantum Information Processing*, 17(3), 2018. [SCIE Indexed, Impact Factor 2.283].

2. **Amandeep Singh Bhatia** and Ajay Kumar, Neurocomputing approach to matrix product state using quantum dynamics, *Quantum Information Processing*, 17(10), 2018. [SCIE Indexed, Impact Factor 2.283].

3. **Amandeep Singh Bhatia** and Ajay Kumar, Modeling of RNA secondary structures using two-way quantum finite automata, *Chaos, Solitons & Fractals*, 116, 332-339, 2018. [SCIE Indexed, Impact Factor 2.213].

4. **Amandeep Singh Bhatia** and Ajay Kumar, Quantum ω-automata over Infinite Words and their Relationships, *International Journal of Theoretical Physics*, 1-12, 2019. [SCIE Indexed, Impact Factor 0.968].

5. **Amandeep Singh Bhatia** and Ajay Kumar, On the Power of Two-way Multihead Quantum Finite Automata, *RAIRO - Theoretical Informatics and Applications*, 53(1-2), 2019 [SCIE Indexed, Impact Factor 0.350].

6. **Amandeep Singh Bhatia** and Ajay Kumar, On Relation between Linear Temporal Logic and Quantum Finite Automata, *Journal of Logic, Language and Information*, 2019 [SCIE Indexed, Impact Factor 0.536].

7. **Amandeep Singh Bhatia**, Mandeep Kaur Saggi, Ajay Kumar, Sushma Jain, Matrix Product State based Quantum Classifier, *IEEE Neural Computation*, 31(7), 2019 [SCIE Indexed, Impact Factor 1.938].

Book Chapter

1. **Amandeep Singh Bhatia** and Ajay Kumar, "Post Quantum Cryptography", In: "*Emerging Security Algorithms and Techniques*", CRC Press (Taylor & Francis), 1st edition, New York, 2019.

Preprint Papers

1. **Amandeep Singh Bhatia** and Ajay Kumar, On the power of quantum queue automata in real-time. arXiv preprint: 1810.12095, 2018.

2. **Amandeep Singh Bhatia** and Ajay Kumar, McEliece Cryptosystem Based On Extended Golay Code. arXiv preprint: 1811.06246, 2018.

3. **Amandeep Singh Bhatia** and Ajay Kumar, Quantum finite automata: survey, status and research directions. arXiv preprint: 1901.07992, 2019.

4. **Amandeep Singh Bhatia** and Ajay Kumar, Quantized Quadratic Sieve Algorithm & its Simulation. arXiv preprint: 2005.11668v1, 2020.

Chapter 1

Introduction

Over the last three decades, quantum computation is a promising research area that deals with the visionary ideas of Computer Science, Physics, and Mathematics. It promises to deliver fundamentally different, radically more powerful computers of the future. Earlier, quantum computing was thought to be only a theoretical possibility, but research has evolved, such as to make quantum computing applications a realistic possibility.

1.1 Quantum Computing

In the early 20th century, the invention of quantum physics forced scientists to reconsider many cherished ideas from classical physics, leading to revolutionary changes in our scientific and philosophical understanding of the universe. The field of quantum computing concerns with the behaviour and nature of energy at the quantum level to improve the efficiency of computations. Initially, Feynman [1] observed an interesting relationship between computers and physics to propose the idea of quantum computing in 1982. Further, Feynman stated that quantum computers could simulate quantum mechanical systems exponentially faster than with classical computers. It has been observed that a classical computer computes a function on some set of inputs and results in some output. However, a quantum computer produces many output values on giving single input. Thus, it operates differently, i.e., it can compute a solution to a problem simultaneously. Quantum mechanics describes our universe at its most fundamental level. Therefore, harnessing the power of quantum mechanics can transform the way of living, working, and communicating.

Quantum computing relies upon the quantum phenomena of superposition, and entanglement to perform operations. Such features do not exist in classical physics. It allows different computation paths in parallel and generates final output based on positive or negative interference between them. There exist quantum algorithms that outperformed the leading classical algorithms known in terms of accuracy for certain tasks. In 1994, Shor [2] designed a quantum algorithm for calculating the factor of a large number n with space complexity $O(logn)$ and time complexity $O((logn)^2 * loglogn)$ on a quantum computer, and then perform $O(logn)$ post-processing time on a classical computer, which could be applied in breaking several cryptosystems, such as RSA algorithm, Buchmann-Williams key-exchange, and elliptic curve cryptography. Through the impetus provided by Shor's algorithm, quantum computational complexity is an exhilarating area that transcends

the boundaries of quantum physics and theoretical computer science.

Theoretical research into quantum computing, along with experimental efforts to construct a quantum computer, has gained a lot of attention. In 1994, Shor [2] introduced the concept of first quantum error correction code by representing the information of one qubit into a highly entangled state of nine qubits. Since the introduction of Shor's algorithm, many quantum algorithms have been introduced. In 1994, Wineland [3] introduced the recognition of Controlled Notgate, using the two lowest energy levels of ions to realize the concept of two-qubit states. In 1996, Grover [4] designed a searching algorithm with great quadratic gain for finding an element in an unstructured set of size n in $\sqrt{n}$ operations approximately. In 1998, Kwiat et al. [5] implemented Grover's algorithm at Los Alamos National Laboratory using conventional optical interferometers. Quantum parallelism is one of the central features of quantum computation which allows quantum algorithms to gain speed up as compared to classical algorithms. Younes [6] investigated the performance of Grover's algorithm over the whole search space. It has been stated that it gives an optimal solution for single match and accuracy can be lower for the number of matches, i.e., the behavior of an algorithm is not reliable.

1.2 Quantum Automata

Models of finite automata are abstract computing devices, which play a crucial role to solve computational problems in theoretical computer science. Thus, it is a natural goal to study quantum variants of classical automata models, which can play an important role in quantum information processing. Quantum finite automata blend quantum mechanics with classical finite automata. It is a theoretical model with finite memory for quantum computers, which plays a vital role in performing real-time computations. The theory of quantum automata has been developed using the principles of classical automata and quantum mechanics. Quantum automata lay down the vision of quantum processor for performing the quantum actions on reading the inputs [7]. Therefore, investigating the quantum automata models impact to gain an insight into their computational power and limitations.

In 1961, Landuer [8] articulated a concept of reversibility in quantum computing. In 1985, Deutsch [9] described a quantum Turing machine and determined the certainty of a universal quantum Turing machine based on the Church-Turing-Deutsch principle. Calude et al. [10] constructed a quantum device to solve the halting problem of Turing machine, which is performed on test-vector selected randomly. Its evolution is shown to be an unbounded and exponentially growing semigroup. Ghosh et al. [11] proposed the concept of parallel memory based on 2 Dot 1 electron quantum cellular automata, which is efficient and compact in nature. Furthermore, in 1993, Yao [12] determined that if a particular function can be computed polynomially with the quantum Turing machine, then a quantum circuit of polynomial-size could be designed for the function. The field of quantum computation and information processing has subsequently made a significant impact on the academic and research community alike.

Soon after the brainstorm of Shor's factorization algorithm [2], the first quantum finite automata (QFAs) models have been introduced. The concept of quantum automata was first introduced by Moore and Crutchfield [13] and Kondacs and Watrous [14] separately. In 1997, Kondacs and Watrous [14] proposed a variant of

quantum automata: *measure-many one-way quantum finite automata* (MM-1QFA). In 2000, Moore and Crutchfield [13] proposed another variant of the quantum model: *measure-once one-way quantum finite automata* (MO-1QFA). MO-1QFA produces the output accept or reject after reading the last symbol of an input string; whereas MM-1QFA results in the output reject, accept, or the continuation after reading each symbol of the input tape.

There is diversity of quantum automata models have been studied since then and investigated in various directions such as quantum finite automata (QFA): one-way QFA (1QFA), Latvian QFA (LQFA), 1.5 way QFA (1.5QFA), two-way QFA (2QFA), quantum Turing machine (QTM), quantum pushdown automata (QPDA), quantum multicounter machines (QMCM), quantum multihead finite automata (QMFA), quantum finite automata with classical stack (QFACS) and many more since last two decades [7] [15–17]. These quantum computing models have got overwhelming response among research communities. Moreover, various researchers studied their computational power, various closure properties, pumping lemmas and rational generating functions of some aforementioned quantum automata models in parallel to classical automata theory.

1.3 Organization of the Thesis

This Thesis is devoted to the design and development of a variant of quantum automata model to represent various languages and to apply the concept of quantum computing in various applications. It comprises of eleven chapters including, the present one.

In chapter 2, we provide a comprehensive and systematic analysis of quantum finite automata models, quantum pushdown automata, quantum Turing machine, quantum finite automata models with density operators and quantum finite automata models with classical states as described in the literature. The aforementioned quantum finite automata models have been compared based on the closure properties and the inclusion relationship among different models has been shown based on language recognition capability. Further, the statistical results (yearly publications, top cited papers, list of authors publications) related QFA papers are shown.

In chapter 3, a variant of two-way quantum finite automata named two-way multihead quantum finite automaton (2MQFA) has been introduced by exploiting the superposition property of quantum automata. We have investigated its language recognition capability and its comparison with classical and quantum counterparts. We have shown that k-head two-way quantum finite automata are more powerful than $(k+1)$-head two-way quantum finite automata. Furthermore, it has been investigated that quantum version of two-way deterministic multihead finite automaton takes less number of heads to recognize a language containing of all words of prime length.

In chapter 4, a notion of quantum queue automata using unitary criteria and its well-formedness conditions have been introduced. We have also introduced a generalization of real-time deterministic queue automata, the real-time quantum queue automata which work in real-time i.e., the input head can move towards the right direction only and takes exactly one step per input symbol. We have proved

that real-time quantum queue automaton is more superior than its real-time classical variants by using quantum transitions.

In chapter 5, the quantum finite-state machines (QFSM) of one-dimensional matrix product state (MPS) representations for quantum spin systems have been designed. Further, the proposed unitary criterion is used to investigate the dynamics of MPS with quantum weightless neural networks, where the output qubit is extracted and fed back (iterated) to input and Von Neumann entropy to measure the possible entanglement of output quantum state. It has been shown that GHZ and AKLT state exhibit undamped and under damping behavior, respectively.

In chapter 6, we have studied the Muller automata, Rabin automata, and Streett automata with quantum acceptance conditions. We have investigated the classes of quantum ω-automata from two aspects: language recognition and their closure properties. It has been proved that quantum Muller automaton is more dominant than quantum Büchi automaton. Furthermore, we have demonstrated the languages recognized by one-way quantum finite automata with different quantum acceptance conditions and proved their closure properties.

In chapter 7, we have shown the relationship between quantum finite automata and linear temporal logic (LTL). We presented the construction of quantum finite-automata (MO-1QFA, MM-1QFA, and LQFA) on finite words from linear-time temporal logic formulas. It has been shown that the class of languages can be definable in LTL, which can be recognized by QFA, except for MO-1QFA.

In chapter 8, the concept of quantum computing has been applied in various areas. Two-way quantum finite automata (2QFA) are more dominant than classical models in language recognition. We have modeled ribonucleic acid (RNA) secondary structure loops such as internal loop and double helix loop using two-way quantum finite automata. It has proved that 2QFA can parse these sequences in linear time by one-sided bounded error. In classical theory, two-way probabilistic finite automaton (2PFA) takes exponential time, whereas two-way deterministic finite automata (2DFA) cannot represent RNA secondary structure loops.

In chapter 9, we proposed a quantum variant of classical factorization algorithm named "Quadratic Sieve" using the quantum principle of superposition and entanglement. We have examined its simulation results on a classical computer to get a feel of the quantum system and proved that it is more efficient than its classical variants from a computational complexity point of view.

In chapter 10, the most successful and undefeated McEliece cryptosystem algorithm in post-quantum cryptography is explored using extended Golay code. The significance of using an extended Golay code instead of the usual Goppa code is examined. Furthermore, the proposed McEliece cryptosystem is simulated using MATLAB.

Finally, **chapter 11** is the conclusion of the Thesis. We present a technical summary of all results and outline current limitations together with future research directions.

Chapter 2

Review of Quantum Computational Models

In this chapter, we give some basics notations of quantum computation, definitions of quantum computational models, and its results which will be used later in the rest of the Thesis. Further, we have compared the quantum computational models, and the inclusive relation between them is shown based on language recognition power. Moreover, bibliometric perspective, including statistical results (yearly publications, top-cited papers, list of author's publications) related QFA papers, are shown.

2.1 Preliminaries

Throughout the Thesis, an alphabet is any finite set Σ of elements called symbols. A word on Σ is a sequence $w = \sigma_1\sigma_2...\sigma_n$ with $\sigma_i \in \Sigma$ being its ith symbol.The length of a string w is the number of characters in w, and it is denoted by $|w|$ [**?**]. We let ϵ be the empty word satisfying $|\epsilon| = 0$. The set of all words (including ϵ) on Σ is denoted by Σ^*, and we let $\Sigma^+ = \Sigma^*/\{\epsilon\}$. A language L on Σ is any subset of Σ, i.e., $L \subseteq \Sigma^*$. The concatenation operation of languages U and V, denoted $U.V$ or just UV, is the set of strings that can be formed by taking any string of U immediately followed by any string in V.

We use the elements of linear algebra and notations of quantum mechanics to define the model of quantum finite automata. Given a set $Q = \{q_1, q_2, ..., q_n\}$ of basis states, every q_i can be represented by its characteristic vector $v_j \in \{0,1\}^n$ having 1 at jth position and 0 elsewhere. A quantum state on Q is a superposition $\psi \in \mathbb{C}^n$ of basis states of the form $\psi = \sum_{j=1}^{n} \alpha_j v_j$, where coefficients α_j's are complex amplitudes satisfying $||\psi|| = 1$ [19]. Consider an input alphabet $\Sigma = \{x_1, x_2, ..., x_k\}$ of events, where a unitary transformation is associated with each symbol x_j such that $U(x_j : \mathbb{C}^n \to \mathbb{C}^n)$. Suppose a quantum system is described by the quantum state (ϕ) at any given instant of time. The evolution of any event x_j is occurred as $\phi^{'} = \phi U(x_j)$. The dynamics are reversible in nature such that $\phi = \phi^{'}\overline{U}(x_j)$, where $\overline{U}$ is a conjugate matrix. The most of the definitions and notations of quantum mechanics we refer to [7, 20] and classical automata theory [21]. The following concepts of linear algebra are used in quantum automata theory:

- Linear vector space [7]: It is defined as a set of elements, called vectors. If any two vectors $|\psi\rangle$ and $|\varphi\rangle$ are the part of it, then $|\psi\rangle + |\varphi\rangle$ belongs to vector

space. It is closed under multiplication and addition by scalars.

- Bra-ket notation [22]: It is defined as criterion for unfolding quantum states, composed of angle brackets and vertical bars.

$$|u\rangle = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}, \langle v| = \begin{bmatrix} a_1^* & a_2^* & a_3^* \end{bmatrix}, |u\rangle \langle v| = \begin{bmatrix} a_1 a_1^* & a_1 a_2^* & a_1 a_3^* \\ a_2 a_1^* & a_2 a_2^* & a_2 a_3^* \\ a_3 a_1^* & a_3 a_2^* & a_3 a_3^* \end{bmatrix} \tag{2.1}$$

a_i^* signifies the conjugate of a complex number a_i. The ket $|u\rangle$ is a column vector, and its conjugate transpose bra $\langle v|$ is a row vector. The bra-ket notation is also known as Dirac notation.

- Qubit: It is defined as a quantum bit that exists in more than one state at a time, i.e., superposition. It can be represented as a state vector having two basis states labeled $|0\rangle$ and $|1\rangle$. In general,

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle \tag{2.2}$$

where α and β are complex numbers. The state $|0\rangle$ occurs with probability $|\alpha|^2$ and $|1\rangle$ with probability $|\beta|^2$. Since the absolute squares of the amplitudes equate to probabilities, it follows that $|\alpha|^2 + |\beta|^2 = 1$. One qubit represents two complex amplitudes (α and β), similarly, n qubits represent 2^n complex amplitudes [23]. The Bloch sphere is used to represent the qubit states geometrically on the surface of a unit sphere. It represents the qubit as a three-dimensional object. The north and south poles are selected corresponding to the standard basis vectors $|0\rangle$ and $|1\rangle$, respectively. Fig 2.1 represents the Bloch sphere.

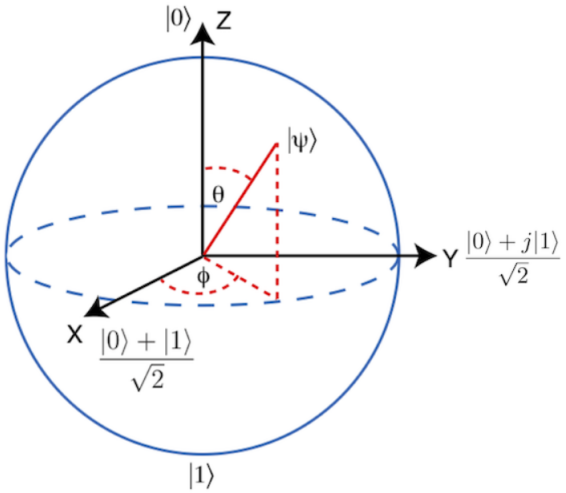


Figure 2.1: Representation of a Bloch sphere

The single qubit quantum state can be represented in convenient way as

$$|\psi\rangle = cos(\theta/2) |0\rangle + sin(\theta/2) e^{\iota\phi} |1\rangle \tag{2.3}$$

where the real α ad β are represented in terms of θ, i.e. $0 \leq \theta \leq \pi$ and $0 \leq \phi \leq 2\pi$ is the quantum phase. Any quantum state can be plotted on

the surface of a sphere called Bloch sphere. The amplitude of the $|1\rangle$ state is $e^{\iota\phi}sin(\theta/2)$ and the amplitude of the $|0\rangle$ state is $cos(\theta/2)$. The length of the vector is equal to $|cos(\theta/2)|^2 + |sin(\theta/2)|^2 = 1$. For instance, $\phi = 0$ and $\theta = \pi/2$. Fig 2.2 represents the qubit state on a Bloch sphere.

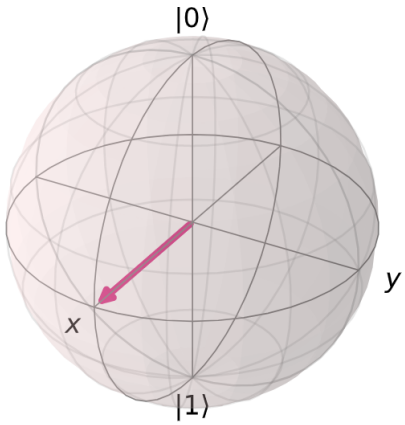


Figure 2.2: Visually representation of a qubit state

- Quantum state [22]: A quantum state $|\phi\rangle$ is a superposition of classical states,

$$|\phi\rangle = \alpha_1 |x_1\rangle + \alpha_2 |x_2\rangle + ... + \alpha_n |x_n\rangle \tag{2.4}$$

where $|x_i\rangle$'s are classical states for $1 \leq i \leq n$, α_i's are complex numbers called amplitudes and $|\alpha_1|^2 + |\alpha_2|^2 + ... + |\alpha_n|^2 = 1$, where $|\alpha_i|^2$ is the squared norm of the corresponding amplitude. Quantum state can also be seen as n-dimensional column vector.

$$\begin{bmatrix} \alpha_1 \\ \alpha_2 \\ ... \\ \alpha_3 \end{bmatrix} \tag{2.5}$$

- Hilbert space $\mathcal{H}$: It is a mathematical framework for describing the principles of the quantum system [22]. It is a complex vector space. An inner product space on Hilbert space is associated with the inner product of vectors $\langle u|v\rangle : \mathcal{H} \times \mathcal{H} \to \mathbb{C}$, fulfil the following properties, such that $u, v, w \in \mathcal{H}$, and $x, y \in \mathbb{C}$ (a set of complex numbers).
  - Linearity: $(x \langle u| + y \langle v|) |w\rangle = x \langle u|w\rangle + y \langle v|w\rangle$
  - Symmetric: $\langle u|v\rangle = \langle v|u\rangle$
  - Definite Positivity: For any $u \in \mathcal{H}, \langle u|u\rangle \geq 0$ and $\langle u|u\rangle = 0$ iff $u = 0$.
- Vector norm: A vector norm ($\|v\|$) is defined as a mapping from R^n to R (Euclidean space) with the following properties:
  - $\|v\| > 0$, if $v \neq 0$
  - $\|\alpha v\| = |\alpha|\|v\|$, for any $\alpha \in R$
  - $\|u + v\| \leq \|u\| + \|v\|$, for any $u, v \in R^n$

- Quantum registers: An n-qubit quantum register is an ordered sequence of n quantum qubit systems. It is a quantum analog of a classical register. The complex Hilbert space is composed of the n-fold tensor product of the two-dimensional Hilbert spaces $\mathcal{H}_2^n$ representing the qubits.

- Density matrix: It is an alternate representation of the state of a quantum system. It is used to describe the describe the statistical state of a quantum system. The density matrix ρ for the pure state $|\phi\rangle$ is given as

$$\rho = |\phi\rangle \langle\phi| = \begin{bmatrix} \alpha_1\alpha_1^* & \alpha_1\alpha_2^* & ... & \alpha_1\alpha_n^* \\ \alpha_2\alpha_1^* & \alpha_2\alpha_2^* & ... & \alpha_2\alpha_n^* \\ ... & ... & ... & ... \\ \alpha_n\alpha_1^* & \alpha_n\alpha_2^* & ... & \alpha_n\alpha_n^* \end{bmatrix} \tag{2.6}$$

  It satisfies the following properties:

  - Projector: $\rho^2 = \rho$
  - Normalization: Tr(ρ)=1
  - Hermiticity: $\rho^\dagger = \rho$
  - Positivity: $\rho \geq 0$

- Mixed quantum state: It is defined as a probabilistic distribution of pure quantum states. The quantum state which cannot be described by single ket vectors, but it is a mixture of ket vectors [24]. The density matrix ρ assigned to a mixed state results from the sum of density matrices assigned to pure quantum states $|\phi_i\rangle$, weighted by their respective probabilities Pr_i [25].

$$\rho = \sum_i Pr_i |\phi_i\rangle \langle\phi_i| \tag{2.7}$$

- Projective measurement: A quantum state $|\phi\rangle \in \mathcal{H}_n$can be measured with respect to an observable i.e. $\mathcal{H}_n$ is decomposed into orthogonal subspaces [24]. A projective measurement of $|\phi\rangle$ produces two outcomes: (i) It is collapsed into a new quantum state $|\phi'\rangle$. (ii) The subspace projection of a state $|\phi\rangle$ was made in classical information.

- Orthogonal and orthonormal vectors [22]: Two vectors $|v\rangle$ and $|u\rangle$ are orthogonal, if they are perpendicular to each other, i.e., the inner product of vectors $\langle v|u\rangle = 0$. It is represented by set of vectors $U = \{u_1, u_2, ..., u_n\}$ are mutually orthogonal if every pair of vectors are orthogonal, i.e. $\langle u_i|u_j\rangle = 0$, $\forall i \neq j$. A set of vectors U is orthonormal if they are mutually orthogonal and each vector in U is a unit vector.

- Unitary evolution: In quantum systems, Markov matrices are replaced by matrices with complex number entries for the time evaluation of probabilistic systems, by maintaining the condition $\sum_{i=1}^n |\alpha_i|^2$. Therefore, consider a quantum system state at time t_0: $|\phi(t_0)\rangle = \alpha_1 |x_1\rangle + \alpha_2 |x_2\rangle + ... + \alpha_n |x_n\rangle$ changes into the state at time t: $|\phi'(t)\rangle = \alpha_1' |x_1\rangle + \alpha_2' |x_2\rangle + ... + \alpha_n' |x_n\rangle$, where amplitudes $\alpha_1, \alpha_2, ..., \alpha_n$ and $\alpha_1', \alpha_2', ..., \alpha_n'$ are related by $|\phi'(t)\rangle = U(t - t_0) |\phi(t_0)\rangle$, where

$U(t-t_0)$ is a time dependent unitary operator such that $(U(t-t_0))^*U(t-t_0) = 1$, α_{ij}'s are its entries for $1 \leq i, j \leq n$

$$\begin{bmatrix} \alpha_{11} & \alpha_{12} & ... & \alpha_{1n} \\ \alpha_{21} & \alpha_{22} & ... & \alpha_{2n} \\ ... & ... & ... & \\ \alpha_{n1} & \alpha_{n2} & ... & \alpha_{nn} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ ... \\ \alpha_n \end{bmatrix} = \begin{bmatrix} \alpha'_1 \\ \alpha'_2 \\ ... \\ \alpha'_n \end{bmatrix} \tag{2.8}$$

and $\sum_{i=1}^{n} |\alpha_i|^2 = \sum_{i=1}^{n} |\alpha'_i|^2 = 1$. Therefore, evaluation of a quantum system at any time must be unitary [22].

- Language recognition with unbounded and bounded error: If any language L is recognized by QFA with probability greater than cut-point λ, $\forall x \in L$, then it is called to be recognized with λ. The probability of recognition by QFA is at most λ, $\forall x \notin L$. A language is said to be accepted with bounded error if there exists $\epsilon > 0$ such that $x \in L$, the acceptance probability is greater than $\lambda + \epsilon$; otherwise, the acceptance probability is less than $\lambda - \varepsilon$ for all $x \notin L$. A language L is recognized by QFA with λ and without a bounded error, then L is said to be recognized with an unbounded error [7].

2.2 Classical Finite State Automata

In classical computation theory, several finite state automata models have been proposed and investigated from various aspects such as computational power, closure properties, and pumping lemmas. In this section, we have considered the basic notations and definitions of the following computational models:

Definition 2.2.1. [21] *DFA is defined as a quintuple $(Q, \Sigma, \delta, q_0, Q_a)$, where*

- *Q is a finite set of states,*
- *Σ is an input alphabet,*
- *δ is a transition function $\delta : Q \times \Sigma \to Q$,*
- *q_0 is a starting state,*
- *$Q_{acc} \subset Q$ is a set of accepting states.*

The computation process of DFA begins with the initial state q_0 and executes the δ according to current state and input symbol ($\sigma \in \Sigma$) under the tape head. An input string is said to be recognized by DFA if the state is changed into an acceptance state belongs to Q_a after reading the last symbol.

Definition 2.2.2. [21] *NDFA is defined as a quintuple $(Q, \Sigma, \delta, q_0, Q_a)$, where*

- *Q is a finite set of states,*
- *Σ is an input alphabet,*
- *A transition function $\delta : Q \times (\Sigma \cup \{\epsilon\}) \to P(Q)$, where $P(Q)$ signifies the power set of Q,*

- q_0 *is an initial state,*
- $Q_{acc} \subset Q$ *is a set of accepting states.*

There can be zero, one, two, or more transitions related to a symbol for each state in NDFA. Thus, all feasible transitions are performed simultaneously from a present state like in the case of parallel computing. Finally, if at least one of state is an acceptance state, the whole computation process is said to be accepted.

Definition 2.2.3. [22] *PFA is represented by quintuple* $(Q, \Sigma, \delta, q_0, Q_a)$*, where*

- Q *is a finite set of states,*
- Σ *is a set of input symbols,*
- *The transition function* $\delta : Q \times (\Sigma \cup \{\#, \$\}) \times Q \to \mathbb{R}_{[0,1]}$*, where \$ and # denote the right and left-end markers denoting starting and ending of the input string,*
- q_0 *is an initial state,*
- $Q_{acc} \subset Q$ *is a set of accepting states.*

In PFA, the computation begins with the q_0 starting state, and δ is performed corresponding to the current input symbol. For each input symbol $\sigma \in \Sigma$, the transition function is defined as a Markov matrix such that $V_\sigma = [v_{ij}]$, where $v_{ij} = \delta(q_i, \sigma, q_j)$, i.e., the probability after reading a symbol σ, the current state q_i is transformed to the state q_j.

2.3 Quantum Circuit

A quantum circuit is a computational routine consisting of coherent quantum operations on quantum data, such as qubits, and concurrent real-time classical computation. It is an ordered sequence of quantum gates, measurements and resets. In general, a quantum circuit may have n input qubits and m output qubits, where n, $m \geq 0$. It induces some quantum operation from n qubits to m qubits, determined by composing the actions of the individual gates in the appropriate way. The size of a quantum circuit is the total number of gates plus the total number of wires in the circuit.

A unitary quantum circuit is a quantum circuit in which all of the gates correspond to unitary quantum operations. A set of quantum gates is said to be universal if any unitary transformation of the quantum data can be efficiently approximated arbitrarily well as sequence of gates in the set. Any quantum program can be represented by a sequence of quantum circuits and non-concurrent classical computation. The quantum circuits will be assumed to be composed of gates from the following list (representing a standard choice for a gate set):

- Identity gate: It has no effect on the value of the qubit it operates on; equivalent to multiplying a value by one.

$$\alpha\,|0\rangle + \beta\,|1\rangle \;—\boxed{ID}—\; \alpha\,|0\rangle + \beta\,|1\rangle \qquad ID = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

- Pauli-X: It flips the qubit i.e. $|0\rangle$ to $|1\rangle$, and vice versa. It is equivalent to rotate to qubit with the angle π.

$$\alpha|0\rangle + \beta|1\rangle \text{ —}\boxed{X}\text{— } \beta|0\rangle + \alpha|1\rangle \qquad X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

- Pauli-Y: It rotates the single qubit through π radians around the Y-axis.

$$\alpha|0\rangle + \beta|1\rangle \text{ —}\boxed{Y}\text{— } \alpha|0\rangle - \iota\beta|1\rangle \qquad Y = \begin{bmatrix} 1 & 0 \\ 0 & -\iota \end{bmatrix}$$

- Pauli-Z: It is π rotation around the Z-axis and has the property that X $\rightarrow$ -X, Z $\rightarrow$ Z. It is also called as Phase flip.

$$\alpha|0\rangle + \beta|1\rangle \text{ —}\boxed{Z}\text{— } \alpha|0\rangle - \beta|1\rangle \qquad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

- Hadamard gate (H): It generates superposition of states with equal probability in computational basis.

$$\alpha|0\rangle + \beta|1\rangle \text{ —}\boxed{H}\text{— } \frac{\alpha+\beta}{\sqrt{2}}|0\rangle + \frac{\alpha-\beta}{\sqrt{2}}|1\rangle \qquad H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

- Controlled-NOT (CNOT) gate: It has two inputs as well as outputs. The flip operation is performed over the target qubit when the first qubit is 1.

$$U_{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

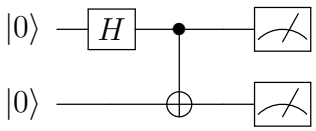
- T gate: It is equivalent to $\pi/4$ rotation around Z-axis. It is needed for universal control.

$$\alpha|0\rangle + \beta|1\rangle \text{ —}\boxed{T}\text{— } \alpha|0\rangle + e^{\iota\pi/4}\beta|1\rangle \qquad T = \begin{bmatrix} 1 & 0 \\ 0 & e^{\iota\pi/4} \end{bmatrix}$$

- SWAP gate: It simply swaps the states of two qubits.

$$U_{SWAP} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The Bell states are specific quantum states of two qubits that represent the simplest and maximally entangled state as $|\phi^+\rangle = (|00\rangle + |11\rangle)/\sqrt{2}$ .The quantum circuit of Bell state is designed, which takes the two qubit input $|00\rangle$ and transforms it to the Bell state. Explicitly, the Hadamard gate transforms $|00\rangle$ into a superposition of $(|0\rangle + |1\rangle)/\sqrt{2}$. Now, CNOT gate is applied, which only inverts the target (the second qubit) when the control (the first qubit) is 1. Thus, the CNOT gate transforms the second qubit and meter symbol represents the measurement. The resultant state is transformed as Bell state.



2.4 Quantum Finite State Automata (QFA)

A quantum finite automaton is a quantum counterpart of a classical finite automaton. In quantum finite automaton, quantum actions are performed on reading the symbols from the inputs tape. Quantum finite automata can be classified into one-way quantum finite automata, 1.5-way quantum finite automata, two-way quantum finite automata, quantum Turing machine, quantum pushdown automata, quantum sequential machines, orthomodular lattice-valued automata, and quantum circuit model. These automata act as a model of quantum processors. In this chapter, we address the quantum finite automata models with finite memory [13,14] [17], [26–28]; quantum automata models with density operators [29–32] and quantum automata with both quantum and classical states [33,34].

Definition 2.4.1. [13] *A real-time quantum finite automaton is defined as a quintuple* $(\mathcal{H}, \Sigma, s_{init}, P_{acc}, U_\sigma)$*, where*

- *An input alphabet* Σ*,*
- *Hilbert space* $\mathcal{H}$*, an initial state vector* $s_{init} \in \mathcal{H}$ *with* $|s_{init}|^2 = 1$*,*
- *A subspace* $\mathcal{H}_{accept} \subset \mathcal{H}$ *and an operator* P_{acc} *which project on it,*
- *A unitary transition matrix* U_σ *for each input symbol* $\sigma \in \Sigma$*.*

The quantum language recognized by QFA as a function $f_{QFA}(w) = |s_{init} U_w P_{acc}|^2$, where $U_w = U_{w_1} U_{w_2} ... U_{w_{|w|}}$. The process of computation of an input string w starts with the initial vector, apply the unitary matrix of each symbol, and measure the probability by applying projection operator such that the resultant state is in subspace $\mathcal{H}_{accept}$. It works in real-time since it takes exactly one step per input symbol and head moves only towards the right direction. A real-time quantum finite automaton does not need to store the input. The given input is fed to the real-time machine from left to right, symbol by symbol. Physically, it can be interpreted as follows. Suppose, we have prepared a quantum system in superposition of initial states.Then, we exhibit the system to different influences over time depending on the input symbols, exactly one time-step per input symbol. At the end of computation, a measurement is performed on the system and $f_{QFA}(w)$ is the probability of this measurement having an acceptable outcome, such as being in a given energy level.

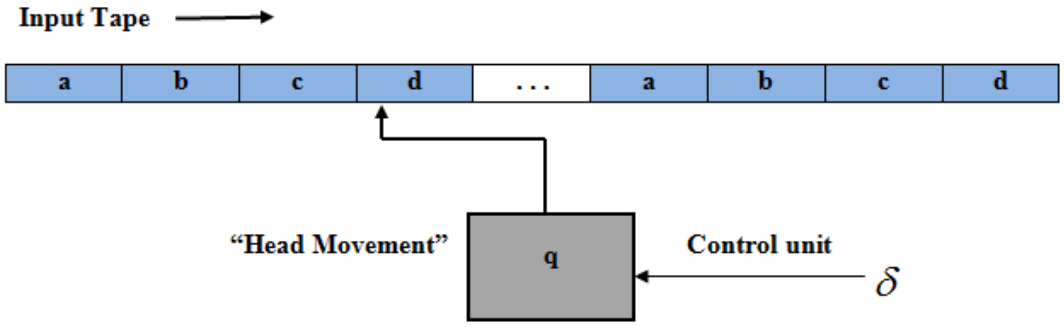


Figure 2.3: One-way quantum finite automata

Definition 2.4.2. [15] *1QFA is represented by sextuple* $(Q, \Sigma, \delta, q_0, Q_{acc}, Q_{rej})$, *where*

- Q *is a finite set of states,*
- Σ *is a set of symbols,*
- *A transition function* $\delta : Q \times \Sigma \times Q \to \mathbb{C}$, *where* $\mathbb{C}$ *is a complex number.*
- q_0 *is an initial state,*
- $Q_{rej} \subset Q$ *and* $Q_{acc} \subset Q$ *represent the set of rejecting and accepting states.*

In one-way QFA, the tape head moves only in the right direction after reading the symbols from an input tape. It produces a superposition of states. Based on the measurement, one-way quantum finite automata are divided into MO-1QFA and MM-1QFA. In Fig 2.3, the representation of one-way quantum finite automata is depicted.

Definition 2.4.3. [13] *MO-1QFA is defined as a sextuple* $(Q, \Sigma, \delta, q_0, Q_{acc}, Q_{rej})$, *where*

- Q *is a set of states,*
- Σ *is a set of input symbols,*
- q_0 *is a starting state,*
- δ: $Q \times \Sigma \cup \{\$\} \times Q \to \mathbb{C}$ *is a transition function fulfil the unitary condition:*

$$\sum_{p \in Q}^{\forall (q_1,\sigma),(q_2,\sigma) \in Q \times \Sigma} \overline{\delta(q_1,\sigma,p)}\delta(q_2,\sigma,p) = \left\{ \begin{array}{l} 1 \; q_1 = q_2 \\ 0 \; q_1 \neq q_2 \end{array} \right\}$$

- Q_{acc} *and* Q_{rej} *represent the set of accepting and rejecting states.*

The computation procedure of MO-1QFA consists of an input string $x = \sigma_1\sigma_2...\sigma_n\$$. For each symbol, transition function is represented as $U(\sigma)$ unitary matrix such that $U(\sigma)(i,j) = \delta(q_j, \sigma, q_i)$. The head reads x symbol by symbol from left to right side, and unitary matrices $U(\sigma_1), U(\sigma_2), ..., U(\sigma_n)$ of each symbol are performed on the current state, beginning with q_0. δ is indicated by a set of unitary matrices $\{V_\sigma\}_{\sigma \in \Sigma}$, where V_σ is a unitary evolution of MO-1QFA, defined as: $V_\sigma(|q\rangle) = \sum_{q' \in Q} (q, \sigma, q') |q'\rangle$. Finally, the projective measurement is performed by projection operator on the final state for checking whether the x is accepted or rejected. It allows the measurement to be made only after reading the last symbol of the input string. If after reading the last symbol, a superposition $\psi = \sum_{q_i \in Q_{acc}} \alpha_i |q_i\rangle + \sum_{q_j \in Q_{rej}} \beta_j |q_j\rangle$ is observed, then the input string is accepted with $\sum \alpha_i^2$ and rejected with $\sum \beta_j^2$. The probability of acceptance for MO-1QFA is calculated as

$$P(x) = \| P_{acc} U(\sigma_n)...U(\sigma_2)U(\sigma_1) |q_0\rangle \|^2$$

Definition 2.4.4. [35] *MM-1QFA is represented by sextuple* $(Q, \Sigma, \delta, q_0, Q_{rej}, Q_{acc})$, *where*

- *Q is a finite set of states, $Q = Q_{acc} \bigcup Q_{rej} \bigcup Q_{non}$, where $Q_{acc}, Q_{rej}, Q_{non}$ denotes the accepting, rejecting, and non-halting set of states correspondingly.*
- *Σ is an input alphabet,*
- *q_0 is a starting state,*
- *The transition function δ is defined by $Q \times \Sigma \cup \{\#, \$\} \times Q \to \mathbb{C}$, where $\#$ and $\$$ represent the left and right-end markers denoting starting and ending of input string. δ represents the amplitudes flow from one state to other after reading the symbol from the input tape. It must satisfy the unitary condition.*

The computation process of MM-1QFA consists of an input string $x = \#\sigma_1\sigma_2...\sigma_n \$$. The tape head reads x from left-end marker $\#$ to end of the string and transition function corresponding to each symbol is performed. The whole Hilbert space is divided into three subspaces: E_{non}, E_{rej}, and E_{acc} belong to sets of non-halting, rejecting, and accepting a set of states, respectively. There are three projectors $P_{non}, P_{rej}, P_{acc}$ on to the three subspaces. After each transition, MM-1QFA measures its state with reference to observable $E_{non} \bigoplus E_{acc} \bigoplus E_{rej}$. If the observed state is in E_{acc} or E_{rej} subspace, then MM-1QFA accepts or rejects the input string respectively; otherwise, the computation process continues. Therefore, after every step of measurement, the superposition of states ends with measured subspace.

Table 2.1: Comparison of MO-1QFA and MM-1QFA

Model properties	**MO-1QFA**	**MM-1QFA**
Introduced by	Moore and Crutchfield [13]	Kondacs and Watrous [14]
Computation process	Measurement is allowed only once after reading the last symbol of the input string.	Measurement is allowed after reading of each symbol by the head from the input tape.
Measurement result	Accept/ reject.	Accept/ reject/ continuation.
Language acceptance	It can accept only group languages.	It is strictly more powerful than MO-1QFA for bounded error acceptance.

Due to the non-zero probability of halting of MM-1QFA, it is convenient to have a path of the aggregate rejecting and accepting probabilities. Hence, the state of the automaton is denoted as $|\phi\rangle, P_{acc}, P_{rej}$, where P_{rej} and P_{acc} are the aggregate probabilities of rejecting and accepting. δ is defined as: $P_{non}|\phi'\rangle, P_{acc} + \| P_{acc}|\phi'\rangle \|^2, P_{rej} + \| P_{rej}|\phi'\rangle$, where $|\phi'\rangle = V_\sigma|\phi\rangle$. The probability of acceptance for MM-1QFA is calculated as $P(x) = \sum_{k=1}^{n+1} \| P_{acc} U(\sigma_k) \prod_{i=1}^{k-1} (P_{non} U(\sigma_i)) |q_0\rangle \|^2$.

Ambainis et al. [26] introduced a generalized version of MO-1QFA named Latvian quantum finite automata (LQFA). It works similarly as MO-1QFA except that the transition function (δ), consisting of projective measurement and unitary matrix [15]. This alteration increases the power of LQFA for acceptance of languages.

Definition 2.4.5. [7] *LQFA can be represented by septuple $(Q, \Sigma, \{A_\sigma\}, \{P_\sigma\}, q_0, Q_{rej}, Q_{acc})$, where*

- *Q is a finite non-empty set of states,*
- *Σ is a finite set of input symbols,*
- *A_σ denotes unitary matrix for each symbol, and P_σ is defined as a set of orthogonal subspaces,*
- *q_0 is an initial state,*
- *Q_{acc} and Q_{rej} represent the accepting and rejecting set of states.*

LQFA is closely related to probabilistic finite automata [27]. LQFA [7] accepts the language with a bounded error if it is in a Boolean form such that $G_0 b_1 G_1 ... b_k G_k$ where b_i's and G_i's represent the letter and group languages, respectively. Therefore, LQFA with bounded error cannot be designed for all set of regular languages [27]. The computing process of LQFA starts with an initial state q_0 for input string $x = \#\sigma_1\sigma_2...\sigma_n\$$. On reading each symbol $\sigma \in \Sigma$, unitary matrix and projective measurement are performed. Based on measurement $P_\$$ of the right-end marker, the input string is said to be accepted or rejected. Therefore, $P_\$ = E_{acc} \bigotimes E_{rej}$, $(E_{acc} = span\{|q\rangle : q \in Q_{Acc}\})$ and $(E_{rej} = span\{|q\rangle : q \in Q_{rej}\})$.

Nayak [30] and Ambainis et al. [29] studied the measure-many version of LQFA. This model is named as general quantum finite automata (GQFA). The definition is almost same as the MM-1QFA. It induces a transition function for any symbol $\sigma \in \Sigma$, that is a combination of projective measurement and unitary transformation instead of a unitary transformation only. Li et al. [32] studied the generalized variant of 1QFA, i.e., 1gQFA, in which σ prompts a quantum operation, preserving positive trace in place of unitary evolution. It is classified as MO-1gQFA and MM-1gQFA. Hirvensalo [31] introduced a generalized version of MO-1QFA in which transition function corresponding to σ preserves a positive trace.

Definition 2.4.6. [32] *MO-1gQFA is defined as a quintuple $(\mathcal{H}, \Sigma, \rho_0, \{\xi_\sigma\}_{\sigma\in\Sigma}, P_{acc})$, where*

- *$\mathcal{H}$ is a Hilbert space of finite dimension,*
- *Σ is an input alphabet,*
- *ρ_0 is a starting state (density operator on $\mathcal{H}$),*
- *ξ is a quantum operation preserving trace on $\mathcal{H}$, for each $\sigma \in \Sigma$,*
- *P_{acc} is projector called accepting subspace of $\mathcal{H}$, $P_{rej} = 1 - P_{acc}$, then the $\{P_{rej}, P_{acc}\}$ forms projective measurement on $\mathcal{H}$.*

Consider an input string $x = \sigma_1\sigma_2...\sigma_n$, firstly, the quantum operations $\xi_{\sigma 1}\xi_{\sigma 2}, ...\xi_{\sigma n}$ are accomplished on ρ_0 after reading the input symbols. In the end, projective measurement $\{P_{rej}, P_{acc}\}$ is applied to the final state. Based on the measurement, the input string is said to be accepted with a certain probability. Therefore, it induces a function such that $f_{MO-1gQFA} : \Sigma^* \to [0, 1]$ as

$$f_{MO-1gQFA}(x) : Tr(P_{acc}\xi_n \circ ... \circ \xi_2 \circ \xi_1(\rho_0))$$

where $\circ$ denotes the composition. Hence, $f_{MO-1gQFA}(x)$ defines the probability of acceptance for x.

Li et al. [32] studied the equivalence problem and language recognition capability of MM-1gQFA. It has been proved that the class of languages recognized by MM-1gQFA is regular languages with bounded error.

Definition 2.4.7. [32] *MM-1gQFA is a sextuple* $(\mathcal{H}, \Sigma, \rho_0, \{\xi_\sigma\}_{\sigma\in\Sigma}, H_{acc}, H_{rej})$, *where*

- $\mathcal{H}$ *is a Hilbert space,*
- Σ *is an input alphabet,*
- ρ_0 *is an initial state (density operator on* $\mathcal{H}$*)* ,
- ξ *is a quantum operation on* $\mathcal{H}$*, for each* $\sigma \in \Sigma$*,*
- H_{acc}, H_{rej} *is the accepting and rejecting subspaces of* $\mathcal{H}$*, respectively. Therefore,* $\{H_{acc}, H_{rej}, H_{non}\}$ *spans the full space of* $\mathcal{H}$*. There exists a set of projection operators* $\{P_{acc}, P_{rej}, P_{non}\}$*, where measurement is performed onto a subspace* H_{acc}, H_{rej}*, and* H_{non} *respectively.*

The input string x is written on an input tape with both end-markers. The computation procedure of MM-1gQFA is same as that of MM-1QFA. Firstly, a quantum operation is executed on current state ρ. Then, the resultant state is measured using a set of projectors $\{P_{acc}, P_{rej}, P_{non}\}$. If the observed state is in H_{acc} or H_{rej} subspace, then MM-1gQFA accepts or rejects the input string respectively; otherwise, the computation process continues with probability $Tr(P_{non}\xi_\sigma(\rho))$. To represent the total number of states of MM-1gQFA, we have defined $\nu : L(H)\times R\times R$. Thus, the current state of an automaton is described as a triplet $\{\rho, P_{acc}, P_{rej}\} \in \nu$. The evolution of MM-1gQFA on reading a symbol σ can be defined by T_σ operator on ν as

$$T_\sigma(\rho, P_{acc}, P_{rej}) = (P_{non}\xi_\sigma(\rho)P_{non}, Tr(P_{acc}\xi_\sigma(\rho)) + P_{acc}, Tr(P_{rej}\xi_\sigma(\rho)) + P_{rej})$$

Definition 2.4.8. [34] *1QFAC is defined as a nonuple* $(S, Q, \Sigma, \Gamma, s_0, |\psi_0\rangle, \delta, U, M)$, *where*

- S *is a finite set of classical states,*
- Q *is a finite set of quantum states,*
- Σ *is an input alphabet,*
- Γ *is an output alphabet,*
- s_0 *is an initial classical state,*
- $|\psi_0\rangle$ *represents an initial quantum state,*
- δ *is a transition function:* $S \times \Sigma \to S$*,*
- U *is a unitary operator for each* σ *and* s *such that* $U = \{U_{s\sigma}\}_{s\in S,\sigma\in\Sigma}$ *and* $U_{s\sigma} : \mathcal{H}(Q) \to \mathcal{H}(Q)$*,*
- $M = \{M_s\}_{s\in S}$*, where each* M_s *is a projective measurement over* (Q) *with outcomes in* Γ*.*

Therefore, each $M_s = \{P_s, \gamma\}_{\gamma\in\Gamma}$, where $P_{s,\gamma}P_{s,\gamma'} = \left\{ \begin{array}{ll} P_{s,\gamma} & \gamma = \gamma' \\ 0 & \gamma \neq \gamma' \end{array} \right\}$ and $\sum_{\gamma\in\Gamma} p_{s,\gamma} = I$. After reading the input string, if the classical state is s and quantum state is in $|\psi\rangle$. The probability of getting γ as a result of the input string is $\|P_{s,\gamma}|\psi\rangle\|^2$. In the above definition, $\Gamma = \{r, a\}$, where r, a signifies rejection, and acceptance of string respectively. Therefore, $M = \{P_{s,a}, P_{s,r}, s \in S\}$, where $P_{s,a}$ and $P_{s,r}$ are projection operators such that $P_{s,a} + P_{s,r} = I$ and $P_{s,a}P_{s,r} = 0$.

Qiu et al. [34] introduced a computational model of one-way finite automata with both classical and quantum states, i.e., 1QFAC. The measurement is performed at once for processing each input string, i.e., after reading the last symbol. The computation procedure of 1QFAC for an input string $x = \sigma_1, \sigma_2, ...\sigma_n \in \Sigma^*$ is described as follows: It starts with an initial classical state and quantum state representing s_0 and $|\psi_0\rangle$ respectively. On reading σ_1, the classical state changes into $\mu\sigma_1$ and quantum state becomes $U_{s_0\sigma_1}|\psi_0\rangle$. Further, on reading the next symbol, the classical state changes into $\mu(\sigma_1\sigma_2)$ and quantum state to the result of applying $U_{\mu(\sigma_1)\sigma_2}$ to $U_{s_0\sigma_1}|\psi_0\rangle$. This transformation of states occurs in succession until the last symbol. Thus, on reading the symbol σ_n, the quantum state becomes $U_{\mu(\sigma_1...\sigma_{n-2}\sigma_{n-1})\sigma_n}U_{\mu(\sigma_1...\sigma_{n-3}\sigma_{n-2})\sigma_{n-1}}...U_{\mu(\sigma_1)\sigma_2}U_{s_0\sigma_1}|\psi_0\rangle$ and the classical state is $\mu(x)$. Let $\mu(Q)$ represents unitary operators on $\mathcal{H}(Q)$. For sake of accessibility, a mapping $\nu : \Sigma^* \to \mu(Q)$ is defined as $\nu(\epsilon) : I$, where I signifies an identity operator on $\mathcal{H}(Q)$.

$$\nu(x) = U_{\mu(\sigma_1...\sigma_{n-2}\sigma_{n-1})\sigma_n}U_{\mu(\sigma_1...\sigma_{n-3}\sigma_{n-2})\sigma_{n-1}}...U_{\mu(\sigma_1)\sigma_2}U_{s_0\sigma_1}|\psi_0\rangle$$

Finally, the probability of 1QFAC on reading the input string x produces a result as: $P(x) = \|P_{\mu(x),\gamma}\nu(x)|\psi_0\rangle\|^2$.

Definition 2.4.9. [36] *1.5QFA is defined by sextuple $(Q, \Sigma, \delta, q_0, Q_{acc}, Q_{rej})$, where*

- *Q is a finite set of states,*
- *Σ is an input alphabet and $\Gamma = \Sigma \cup \{\#, \$\}$,*
- *Transition function (δ) satisfies the condition: $\delta(q, \alpha, p, \leftarrow) = 0$ for $p, q \in Q$, $\leftarrow$ is a head movement towards the left direction (not allowed) and $\alpha \in \Gamma$*
- *q_0 is an initial classical state,*
- *Q_{acc} and Q_{rej} represent the set of accepting and rejecting states.*

In 1.5-way quantum finite automata (1.5QFA), the tape head is enabled to remain stationary or move towards the right direction of the input tape, but it cannot move towards the left of input tape. Amano and Iwama [37] proved that if the input tape is circular, then 1.5QFA can be designed for non-context-free languages. They have not considered the right-end marker \$ on the input tape.

Two-way quantum finite automaton (2QFA) is a quantum version of a two-way finite automaton. In 2QFA model, the tape head can remain stationary or move either in the left or right direction. 2QFA is more dominant than the classical model. In Fig 2.4, the representation of two-way quantum finite automata is depicted.

Definition 2.4.10. [14] *Two-way quantum finite automaton is a sextuple $(Q, \Sigma, \delta, q_0, Q_{acc}, Q_{rej})$, where*

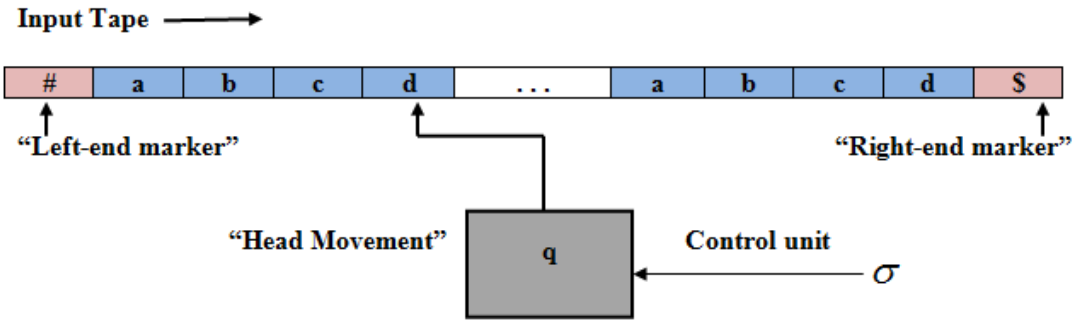


Figure 2.4: Two-way quantum finite automata

- *Q is a set of states. Moreover, $Q = Q_{acc} \cup Q_{rej} \cup Q_{non}$, where $Q_{acc}, Q_{rej}, Q_{non}$ represent the set of accepting, rejecting and non-halting states respectively.*
- *Σ is an input alphabet,*
- *Transition function δ is defined by $\delta : Q \times \Sigma \cup \{\#, \$\} \times Q \times D \to \mathbb{C}$, where $D = \{\leftarrow, \uparrow, \rightarrow\}$ represent the left, stationary, and right direction of the tape head. δ must fulfil the following conditions:*

  1 Local orthogonality and probability condition:

$$\sum_{(q',d)\in Q\times D}^{\forall (q_1,\sigma_1),(q_2,\sigma_2)\in Q\times\Sigma} \overline{\delta(q_1,\sigma,q',d)}\delta(q_2,\sigma,q',d) = \left\{ \begin{array}{l} 1 \ q_1 = q_2 \\ 0 \ q_1 \neq q_2 \end{array} \right\}$$

  2 First separability condition:

$$\sum_{q'\times Q}^{\forall (q_1,\sigma_1),(q_2,\sigma_2)\in Q\times\Sigma} \overline{\delta(q_1,\sigma_1,q',\rightarrow)}\delta(q_2,\sigma_2,q',\uparrow) + \overline{\delta(q_1,\sigma_1,q',\uparrow)}\delta(q_2,\sigma_2,q',\leftarrow) = 0$$

  3 Second separability condition:

$$\sum_{q'\times Q}^{\forall (q_1,\sigma_1),(q_2,\sigma_2)\in Q\times\Sigma} \overline{\delta(q_1,\sigma_1,q',\rightarrow)}\delta(q_2,\sigma_2,q',\leftarrow) = 0$$

A 2QFA is simplified, for each $\sigma \in \Sigma$, if there exists a unitary linear operator V_σ on the inner product space such that $L_2\{Q\} \to L_2\{Q\}$, where Q is the set of states and a function $D : Q \to \{\leftarrow, \uparrow, \rightarrow\}$. Define transition function as:

$$\delta(q,\sigma,q',d) = \left\{ \begin{array}{l|l} \langle q'|V_\sigma|q\rangle & \text{if } D(q') = d \\ 0 & \text{else} \end{array} \right\}$$

where $\langle q'|V_\sigma|q\rangle$ is the coefficient of $|q'\rangle$.

To process the input string by 2QFA, we assume that input string x is written on the input tape with both end-markers such that $\#x\$$. The automaton is in any state q, and the head is above the symbol σ. Then, it is changed into state q' with amplitude $\delta(q,\sigma,q',d)$ and changes the direction of tape head one cell towards left,

stationary and in right direction $d \in \{\leftarrow, \uparrow, \rightarrow\}$ accordingly. The automaton for processing an input x corresponds a unitary evolution in the inner-product space $\mathcal{H}_n$.

A computation of a 2QFA is a sequence of superpositions $c_0, c_1, c_2, ...$, where c_0 is an initial configuration. When the automaton is observed in a superposition state, for any c_i, it has the form $U_\delta |c_i\rangle \sum_{c \in C_n} \alpha_c |c_i\rangle$ where C_n defines the set of configurations, and the configuration c_i is associated with amplitude α_c. Superposition is valid; if the sum of the absolute squares of their probability amplitudes is unitary. The probability for a specified configuration is given by the absolute squares of amplitude associated with that configuration. Time evolution of quantum systems is given by unitary transformations.

$$U_\delta^x |q, k\rangle = \sum_{(q',d) \in Q \times D} \delta(q, x(k), q', d) |q', k + d mod |x|\rangle$$

for each $(q, k) \in C_{|x|}$, where $q \in Q, k \in Z_{|x|}$ and extended to $\mathcal{H}_n$ by linearity [14].

Definition 2.4.11. [33] *A 2QCFA is defined as a nonuple* $(S, Q, \Sigma, \Theta, \delta, q_0, s_0, S_{acc}, S_{rej})$, *where*

- S *is a finite set of classical states,*
- Q *is a finite set of quantum states,*
- Σ *is an input alphabet,*
- Θ *defines the evolution of the quantum part of an internal state,*
- δ *defines the evolution of the classical part,*
- s_0 *and* q_0 *are the initial classical and quantum states respectively,*
- S_{rej} *and* S_{acc} *signify the set of rejecting and accepting states respectively,* $(S_{acc}, S_{rej} \subseteq S)$,
- δ *is a transition function* δ*:* $S \times \Sigma \rightarrow S$, *(the classical transition map)*

Ambainis and Watrous [33] introduced 2QFA with classical states (2QCFA). In this model, the internal state may be a (mixed) quantum state, and the tape head position is classical. It is a central model between 1QFA and 2QFA.

Consider an input string x the computation procedure of 2QCFA is as follows: Initially, tape squares are indexed with $1, 2, ..., |x| = n$ consists $x_1, x_2, ..., x_n$ and squares are indexed with 0 to n+1 including right and left-end markers. On reading the symbol, σ the quantum state is changed according to $\Theta(s, \sigma)$ and further classical state and position of tape head is transformed according to $\delta(s, \sigma)$ and the outcome is achieved from a measurement of $\Theta(s, \sigma)$. Subsequently, the outcomes of each measurement are probabilistic, and the transitions for the classical state may also be probabilistic. Therefore, for any input string, the 2QCFA is said to be accepted with probability $S_{acc}(x)$ when the computation enters classical accepting state S_{acc}, otherwise rejected with probability $S_{rej}(x)$. Thus, the computation process is halted when it enters either classical accepting or rejecting state.

2.5 A Literature Review

Initial research into quantum automata was conducted by Moore et al. [13], Kondacs et al. [14], Ambainis et al. [10, 38], and Brosky et al. [35]. After that, a significant body of research has been produced in the field of quantum automata theory. As such, a systematic review of the literature to appraise the current state of quantum automata research. Moore et al. [13] introduced the concept of quantum finite and quantum pushdown automata. Their investigation revealed that quantum regular and context-free languages could be represented by quantum finite automata and quantum pushdown automata, respectively. The summary of various QFA models is described in Fig 2.5.

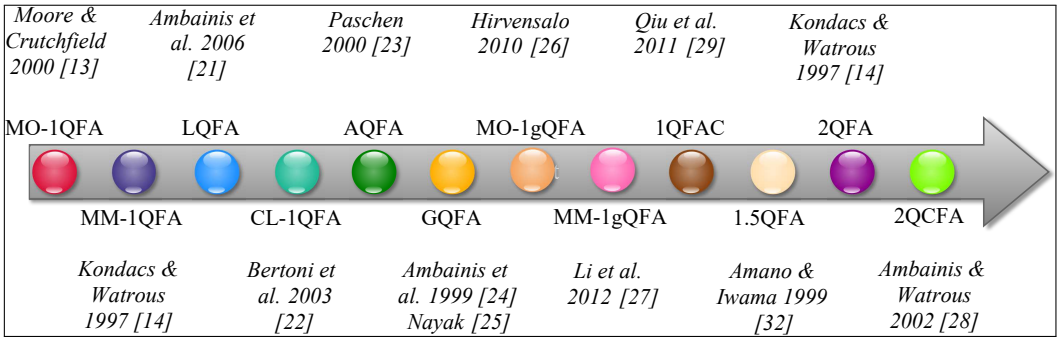


Figure 2.5: Summary of various quantum finite automata models

Brodsky et al. [35] identified the language of acceptance, closure properties, and equivalence in 1QFA models. MO-1QFA with bounded error is designed for group languages. Furthermore, an algorithm for the equivalence of two MO-1QFAs is introduced and proved that PFA can simulate the behaviour of MO-1QFA. Ambainis et al. [10] demonstrated that languages recognized by 1QFA are not closed under Boolean or union operations. Furthermore, they analyzed the necessary and sufficient conditions for 1QFA. Concerning bounded error acceptance mode, MM-1QFA is significantly more powerful than MO-1QFA [38]. In the case of MM-1QFA, measurements are taken after reading each symbol from the input tape causing the string to be rejected without processing the complete input string. Brodsky et al. [35] demonstrated that languages recognized by MM-1QFA are closed under word quotients, inverse homomorphisms, and complement operations, but are not closed under homomorphism.

Ambainis et al. [38] demonstrated that MM-1QFA could be designed for language with a probability of more than 7/9 if one-way reversible finite automata (1RFA) can be designed. It has been shown that if we allow smaller probabilities, MM-1QFA can be more powerful than 1RFA. MM-1QFA can recognize the language prime $L_p = \{a^p \mid where\ p\ is\ a\ prime\}$ with a probability close to 1, which is exponentially smaller than PFA is investigated. Ambainis et al. [29] shown that MM-1QFA can be exponentially outsize than its corresponding deterministic finite automata for a particular language. Kikusts [39] designed MM-1QFA for a language, which requires quadratically fewer states than its corresponding DFA.

Ambainis et al. [26] introduced the concept of LQFA, i.e., generalized form of MO-1QFA. In LQFA, δ is a blend of projective measurement and unitary matrix [7]. They demonstrated that the class of languages recognized by LQFA are closed under

inverse homomorphism, union, complement, and word quotients. Therefore, a MM-1QFA can be designed for all languages for which LQFA can be designed.

Paschen [28] introduced a computational model by appending some ancilla qubits to prevent unitarity condition. It can be executed by the addition of an output alphabet. Ciamarra [40] introduced a new model of 1QFA having computational power is at least equal to classical automata. Ancilla QFA and Ciamarra QFA are special MO-1gQFA. Diaz-caro et al. [41] proposed a quantum-like classical automaton named affine automata (AFA). Further, the proposed automata model is compared with QFA and PFA. It has been proved that AFA is more powerful than QFAs and PFAs. But, AFA is more powerful than PFA and equivalent to QFA in nondeterministic case based on language recognition.

The measure-many variant of LQFA is named as GQFA. It has been shown that GQFA [29, 30] takes the number of states than DFA to recognized $L = \{w0 \mid w \in \{0,1\}^* \mid w \leq m, m \geq 1\}$ exponentially. Nayak [30] showed that GQFA could not accept all regular languages with bounded error. It has been proved that GQFA can be designed for stochastic languages with unbounded error [42]. Hirvensalo [31] introduced a generalized model of MO-1QFA, in which each transition function of σ induces an entirely positive trace-preserving mapping [32].

Yakaryilmaz [43] conceptualized the idea of a blind counter with real-time quantum automata. Furthermore, the real-time quantum automaton is separated with a blind counter (rtQ1BCA) from one-way deterministic k-blind counter automata 1DkBCA by using the Kleene closure of language $L = \{a^n b^n \mid n \geq 0\}$. It has been shown that L can be recognized with negative one-sided error bound ϵ by real-time quantum blind counter automata. On the other hand, it has shown that L can be easily recognized by one-way deterministic with one blind counter automata (1D1BCA). Yakaryilmaz [43] also postulated that L^* could not be recognized by one-way probabilistic k-blind counter automata (1PkBCA). But recently, Nakanishi et al. [44] disapproved this conjecture. They provide an algorithm for 1P1BCA that recognizes L^*.

Amano and Iwama [37] introduced the concept of 1.5QFA, demonstrating that the problem of emptiness was unsolvable using this model. 1.5QFA can be designed for some languages, which cannot be recognized by 1QFA. Nakanishi et al. [11] demonstrated that non-regular language $L = \{a^m d b^n c b^n \mid n \geq 0, m \geq 0\}$ can be modeled by 1.5QFA. Further, it has been demonstrated that it can be designed for non-context-free languages with a probability of less than 2/3, which can be recognized by QCPA with larger probability. Yakaryilmaz [36] concluded that 1.5QFA could recognize non-stochastic languages.

Kondacs et al. [14] introduced the concept of two-way quantum finite automata (2QFA), demonstrating that it was more powerful than the classical two-way finite automata. In 2QFA, the tape head can move either left or right or may remain stationary. Furthermore, it has been shown that 2QFA can accept non-regular languages linearly with one-sided error. It has also been noted that 2QFA can accept non-context-free languages with a bounded error in linear time. Ambainis et al. [38] reported that 2QFA was difficult to implement because the quantum states are needed to keep track of position of the head, which grows with the size of an input string. Yakaryilmaz et al. [45] defined a more efficient probability amplification technique for $L = \{a^n b^n \mid n \in N\}$. Further, various techniques for reducing the state of probability amplification are introduced, and machines with less state complexity

are presented. In recent years, the research exertion on QFA models has decisive on one-way models. However, the study of 2QFAs is attracting less attention.

Ambainis et al. [33] described a concept of 2QFA with classical states, i.e., (2QCFA). It has been demonstrated that 2QCFA is more powerful and can be implemented with a constant size of a quantum part. It has also been proved that classical 2-way finite automata equipped with constant-size quantum registers can perform quantum transformations, measurements and can recognize non-regular language $L = \{a^n b^n \mid n \in N\}$. The size of quantum registers for 1.5QFA and 2QFA depends on the input length. Thus, it has been shown that 2QCFA is superior to these models that have a finite amount of quantum registers. Furthermore, Ambainis et al. [33] designed 2QCFA for the palindrome language, which cannot be recognized by two-way deterministic finite automata (2DFA). Qiu [46] demonstrated various Boolean operations over the class of languages recognized by 2QCFA. Zheng [47] proved that 2QCFA can be designed for $L = \{xcy \mid x, y \in \{a, b\}^*, c \in \Sigma, \mid x \mid = \mid y \mid\}$ in polynomial time with one-side error probability. But, it can be recognized by 2PFA in exponential time with bounded error.

2.5.1 Succinctness results

In recent years, state complexity advantages of QFA is a new direction. The power of quantum finite automata solving promise problems is a hot topic. It has got an enormous response from various researchers. In 2009, Ambainis and Nahimovs [48] designed the quantum automata using probabilistic argument. It has been shown that proposed automata can be designed for $L_p = \{a^i \mid i \text{ is divisible by } p\}$ with $4/\epsilon \log 2p$ states, whereas classical automata take p states. The exact quantum computation has been widely studied for promise problems. Ambainis and Yakaryilmaz [49] proved that promise problems could be recognized by 2QFA in real-time by just tuning the transition amplitudes. It has been shown that $A^k_{yes} = \{a^{i2^k} \mid i \text{ is a non-negative integer}\}$ and $A^k_{no} = \{a^{i2^k} \mid i \text{ is a positive odd integer}\}$ can be recognized by 2QFA in real-time, whereas any DFA takes at least 2^{k+1} states.

Bianchi et al. [50] shown the superiority of quantum variant over DFA by considering the promise problems $(A^{N,r_1}_{yes}, A^{N,r_2}_{no})$ such that $0 \leq r_1 \neq r_2$, where $A^{N,r_1}_{yes} = \{\sigma^n | n \equiv r_1 \ mod \ N\}$ and $A^{N,r_2}_{no} = \{\sigma^n | n \equiv r_2 \ mod \ N\}$ over unary alphabet σ. It has been shown that extended version of promise problem can be recognized by MM-1QFA exactly, but DFA takes d states such that $d|N$ and $d \nmid (r_2 - r_1) \ mod \ N$, where d is the smallest integer. Rashid and Yakaryilmaz [51] designed promise problems for which PFA cannot be designed with bounded error. Yakaryilmaz and Say [52] investigated that after setting the transition amplitudes, various regular languages of infinite families can be recognized by 2QFA with probability greater than 1/2, whereas the size of one-way variants (1PFA and 1QFA) increase without bound. Zheng et al. [53] examined the state succinctness of 2QCFA. It has been proved that $A^{eq}_{yes} = \{w = a^m b^m | w \in \{a, b\}^*\}$ and $A^{eq}_{no} = \{w \neq a^m b^m | w \in \{a, b\}^*, |w| \geq m\}$ can be recognized by a 2QCFA in a polynomial running time with O($\log 1/\epsilon$) classical states and a constant number of quantum states, whereas DFA takes at least $2m+2$ states and $\sqrt{m}$ for 2DFA and 2NFA respectively.

Gruska et al. [54] introduced two acceptance modes named recognizability and solvability, and explored various promise problems for classical, quantum and semi-quantum automata. It has been shown that quantum variants of classical automata

have significantly more power. Zheng and Qiu [55] presented a method for state succinctness results of QFA. It has been shown that state succinctness outcomes can be extracted from the results of query complexity. The simple quantum algorithm is given for partial function. It has been proved that promise problem $A(n) = \{A_{yes}(n), A_{no}(n)\}$, where $A_{no}(n) = \{x\#y\#\#x\#y|x, y \in \{0,1\}^n, H(x,y) = n/2\}$, and $A_{yes}(n) = \{x\#y\#\#x\#y|x, y \in \{0,1\}^n, H(x,y) \in \{0,1,n-1,n\}\}$ can be recognized by 1QCFA with $O(n^3)$ classical states and $O(n^2)$ quantum states, but the size of 1DFA is $2^{\Omega(n)}$. Zheng et al. [56] considered the concept of time-space complexity and proved that quantum computing models are more superior than classical variants in context of language recognition. The communication complexity results are used to derive time-space upper-bounds for 2QCFA and proved more desirable than a probabilistic Turing machine (PTM). Further, it has also been proved that 2PFA is more superior than DTM in terms of time-space complexity. Zheng [57] et al. examined the promise problems recognized by quantum, semi-quantum finite automata and classical automata. It has been proved that there is a promise problem that is recognized exactly by MO-1QFA, but cannot be recognized by DFA.

Zheng et al. [58] proposed 1QFA with classical states (1QCFA) and examined its closure properties. Further, the main succinctness result is derived, and the state complexity of 1QCFA is demonstrated. Bianchi [59] shown that 1QCFA with isolated cut-point can be designed for regular languages. Recently, Gainutdinova and Yakaryilmaz [60] investigated the computational power of PFA and QFA by studying promise problems based on unary languages. It has been proved that there exist unary promise problems which can be recognized by QFAs but cannot be recognized by PFAs with bounded error. Further, they have considered a promise problem with two parameters and shown that QFA is more succinct than PFA by adjusting one parameter, whereas PFA can be more concise than DFA exponentially after adjusting the other.

2.5.2 Other results

Interactive proof (IP) systems with verifiers are formed by QFA, where a powerful quantum prover communicates with a quantum-automaton verifier via a shared communication channel. The quantum prover is computationally strong (unlimited), whereas the power of verifier is limited. In 2009, Nishimura and Yamakami [61] explored a direct application of QFA to IP, where prover is the unitary operation and QFA is a verifier. The computational power of QFA models is studied by imposing restrictions. QIP(A) represents the class of languages recognized by QIP associated with verifier A. It has been shown that the language recognition power of 1QFA can be increased with an interaction of prover, i.e., QIP(1QFA) is equal to regular languages. Further, it has been proved that 2QFA is a proper subset of QIP(2QFA) in polynomial time for language $P = \{x\#x^R|x \in \{0,1\}^*\}$, where # is a separator. Furthermore, it has been demonstrated that the QIP(MO-1QFA) verifier is not closed under complementation and QIP(2QFA) is closed under union. Zheng et al. [62] investigated the power of QIP associated with 2QCFA verifier. It has been shown that the languages $L_m = \{xay|x, y \in \{a,b\}^*, |x| = |y|\}$ and $L_p = \{xax^R|x, \in \{a,b\}^*\}$ can be recognized by QIP(2QFA) in polynomial and exponential time with one-sided bounded error respectively.

A debate system is a generalized version of IP system, where two provers argue

over the belongingness of particular string (x) in a language (L). The prover prompts the verifier that $x \in L$ and refuter attempts to prove that x does not belong to L. Yakaryilmaz et al. [63] investigated that quantum model surpasses the classical ones when bounded to run in polynomial time. It has been proved that non-context free language $L^P = \{1^p | p \text{ is prime}\}$ and $L^m = \{1^{2^m} | m > 0\}$ can be recognized by 2QCFA in polynomial time debates using two qubits, $L^s = \{1^{m^2} | m > 0\}$ and $L^n = \{1^n | n \text{ is a fibonacci number}\}$ with three qubits respectively. Yamakami [64] introduced QFA with extra information known as advice, which depends on the length of an input string. It has been explored that QFA with advice cannot be designed for some regular languages. Yamakami [65] extended the single prover model, where more than one prover can interact with a verifier named quantum multiple prover interactive proof (QMIP) systems and the advantages of quantum computation over classical variants are shown. It has been proved that QIP(1QFA) $\neq$ QMIP(1QFA) and QIP(2QFA) $\neq$ QMIP(2QFA) in polynomial time. Scegulnaja-Dubrovska et al. [66] shown that palindrome language can be recognized by MM-1QFA with postselection, whereas it cannot be recognized by PFA with non-isolated cut-point 0. Yakaryilmaz and Say [67] proposed a quantum finite automata with postselection and proved that the computational power of real-time PFA and QFA can be increased with postselection. Further, the class of languages recognized by QFA with postselection are examined.

2.5.3 Comparison of various QFA models

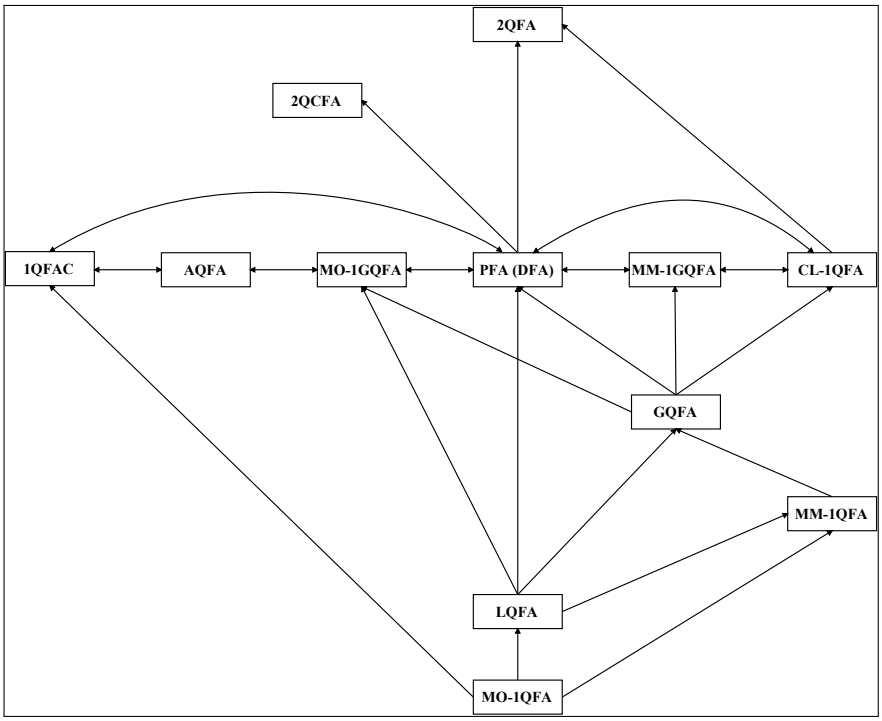


Figure 2.6: Bounded error inclusion relationship between the languages recognized by QFA models

In this section, we review comparative studies of various types of quantum finite automata. Fig 2.6 shows an inclusion relationship among the languages accepted

with a bounded error by 1QFAs, GQFAs, 2QFA, and 2QCFA. The acronyms of models used to indicate the class of languages recognized by them, e.g., "MO-1gQFA" depicts the class of languages identified by the MO-1gQFA with bounder error. One-directional lines show containment relation and bidirectional lines show equivalence relation. The relationship among the model shows: the languages recognized by MO-1QFA are contained in those identified by the MM-1QFA. Ancilla QFA, CL-1QFA and generalized versions of 1QFA accept a regular class of languages. MO-1gQFA can simulate the behaviour of DFA and PFA. Thus, both recognize exactly regular languages. 2QCFA is more powerful than two-way PFA because it can recognize regular languages with certainty and also some non-regular languages in polynomial time. MM-1gQFA, CL-1QFA, PFA, MO-1gQFA, AQFA, 1QFAC recognize the regular class of languages.

2.5.4 Closure properties

Table 2.2: Closure properties of various QFA models, Here, ✓, ×, − represents that a particular model is closed, not closed and undefined respectively.

Authors	**Models**	**Homomorphism**	**Inverse homomorphism**	**Union**	**Word quotients**	**Boolean operations**	**Complement**
Brodsky et al. [35]	MO-1QFA	×	✓	✓	−	✓	−
Ambainis et al. [10]	MM-1QFA	×	✓	✓	−	−	✓
Ambainis et al. [26]	LQFA	−	✓	✓	✓	✓	✓
Bertoni et al. [27]	CL-1QFA	−	−	−	−	✓	−
Macko [68]	1.5QFA	×	−	−	−	−	−
Macko [68]	2QFA	×	−	−	−	−	−
Qiu [46]	2QCFA	−	−	−	−	✓	✓

In this subsection, we describe a comparative study of various quantum finite automata based on their closure properties. Macko [68] demonstrated that the class of languages are closed under inverse length non-increasing homomorphism and non-erasing inverse homomorphism in case of 1.5QFA and 2QFA. Qiu [46] proved that 2QCFA with error probabilities is closed under intersection, complement and reversal operation. Further, 2QCFA with a certain restricted condition is closed under concatenation operation. Table 2.2 illustrates a comparative study of various quantum models based on their closure properties.

2.5.5 Simulations of classical counterparts

It is known that computation process of most classical models is either deterministic or probabilistic. Therefore, investigate the power of PFA to their quantum counterparts is a natural goal. Till now, there are several QFA models to simulate the classical automata models exactly.

Theorem 2.5.1. *Let L be a language recognized by MO-1QFA with cut-point λ, then there exists a PFA recognizing the same language with λ'.*

The proof of the theorem is given in [31] [35] [69]. It is also valid for probabilistic and quantum Turing machines [70]. Therefore, it can be easily checked that any PFA can be converted into an equivalent QFA model.

Theorem 2.5.2. *Let L be a language accepted by n-state 1QFA, then GFA can be designed with n^2-state for the same language.*

Proof. The proof has been shown in [69] [71]. □

Theorem 2.5.3. *If any language L is recognized by n-state 1QFA (pure states) with bounded error, then 1DFA can be designed with $2^{O(n)}$-states.*

Proof. The proof of the theorem is given in [14,38]. □

There exists a more powerful generalization of 1QFA model that allows mixed states named GQFA. Any 1DFA can be easily simulated by a GQFA.

Theorem 2.5.4. *There exists a MO-1gQFA for every regular language recognized by PFA.*

Proof. MO-1gQFA can simulate the behaviour of DFA exactly. The proof of the simulation process is shown in [31]. However, the detailed proof is given by Li et al. [32]. □

Theorem 2.5.5. *If any language L is recognized by n-state 1QFA (mixed states) with bounded error, then 1DFA can be designed with $2^{O(n^2)}$-states.*

Proof. Simulation process of 1DFA by MO-1gQFA has been shown in various papers. The simulation by considering the upper bound on the number of states is given in [16]. □

2.6 Some Open Problems

Based on literature survey, we address some open problems for further research [7,10,13–15,32,34,40,42,72].

Open problems of 1QFA
• Concerning the bounded error acceptance mode, characterize the set of languages recognized by MM-1QFA.
• Investigate the relation between the state complexities of multi-letter QFAs with MO-1QFAs for unary regular languages.
• To determine the language recognition power of MO-1QFA with unbounded error acceptance mode.

Open problems of 1.5QFA
• Concerning the bounded error acceptance mode, determine whether 1.5QFA can be designed for regular languages.
• To compare state complexity of 1QFA and 1.5QFA.
• To determine the various closure properties of 1.5QFA.
Open problems of 2QFA
• To investigate a language for which 2QFA with bounded error takes polynomial time, whereas 2PFA with a bounded error of the same languages take exponential time.
• To determine the decidability of 2QFA and 1.5QFA equivalence.
• To determine whether there exists a non-stochastic language which can be represented by 2QFA, but not by 1.5QFA?
• To find out whether 2QFA can accept any non-stochastic languages with bounded error mode.
•To determine the various closure properties of 2QFA.
• To determine whether 2QFA is more powerful than corresponding classical automata if it is restricted to a particular measurement after a specified time.
Miscellaneous models
• To compare the state complexity of 1QFAC with 1QFA models.
• To determine the simulation of 1QFAC by 1QFA models.
• To investigate whether GQFA with unbounded error can be designed for more languages strictly as compared to MM-1QFA.
• To check whether there exists any algorithm to solve the equivalence between 2QCFA polynomially
Other promising research directions
• To study the computational power of QFA with algebraic methods.
• To investigate the power of other QFA models with advice from a computational complexity point of view.
• To explore more languages for which QFA can be designed with less number of states as compared to PFA and DFA.
• To determine more promise problems to show separations between computational models.

2.7 A Bibliometric Perspective of QFA

Sixty-two journal papers, 23 conference papers, and 3 technical reports have been evaluated in this review systematically. Fig. 2.7 shows the number of papers published from the year 1997 to 2018 in field of quantum finite automata, where publication years and the number of published papers in review are on the x and y-axis respectively.

Sixty percentage of papers are journal papers, 29% of papers are conference proceedings, 5% are workshop papers, 4% of papers are technical reports, and 2% are Thesis.

Table 2.4: List of sources publishing the top 15 quantities of articles of quantum automata

Ranking	Publication sources	Quantity

1	Theoretical Computer Science	38
2	Lecture Notes In Computer Science	28
3	International Journal of Foundations of Computer Science	17
4	International Journal of Theoretical Physics	11
5	Information And Computation	6
6	Journal of Computer And System Sciences	4
8	RAIRO- Theoretical Informatics And Applications	5
9	Natural Computing	5
10	Quantum Information Computation	5
11	Journal of Statistical Physics	4
12	Physical Review E	4
13	Information Processing Letters	4
14	Physical Review A	4
15	Information Sciences	3

Table 2.4 shows the publication sources which have published the top 15 quantities of contributions relating quantum finite automata. Fig 2.8 depicts the number of publication by various authors. There is a lot of literature on all sorts of quantum finite automata, particularly from the University of Latvia (Ambainis, Freivalds, Yakaryilmaz, etc.). Table 2.5 depicts the top 15 highly influential and cited paper related to quantum automata (source: https://scholar.google.co.in, accessed Jan 8, 2019).

Table 2.5: Top 15 cited papers

Year	**Title**	**Journal/ Proceedings**	**Number of citations**
2000	Quantum automata and quantum grammars [13]	Theoretical Computer Science	430
1997	On the power of quantum finite state automata [14]	Foundations of Computer Science	429
1998	1-way quantum finite automata: Strengths, weaknesses, and generalizations [38]	Foundations of Computer Science	307
2002	Characterizations of 1-way quantum finite automata [35]	SIAM journal on computing	182
2002	Dense quantum coding and quantum finite automata [29]	Journal of the ACM	193
2000	Two-way finite automata with quantum and classical states [33]	Theoretical Computer Science	176
2003	Quantum computing: 1-way quantum automata [27]	Developments in Language Theory	91
1999	Undecidability on quantum finite automata [37]	ACM Symposium on Theory of Computing	69
2000	On the class of languages recognizable by 1-way quantum finite automata [10]	Theoretical Aspects of Computer Science	59
2006	Algebraic results on quantum automata [26]	Theory of computing systems	58

2010	Succinctness of Two-way probabilistic and quantum finite automata [52]	Discrete Mathematics and Theoretical Computer Science	60
1999	Probabilities to accept languages by quantum finite automata [73]	International Computing and Combinatorics Conference	50
2012	One-way finite automata with quantum and classical states [58]	Languages alive (lecture notes in computer science)	31
2013	State succinctness of two-way finite automata with quantum and classical states [53]	Theoretical Computer Science	25
2014	On the state complexity of semi-quantum finite automata [74]	RAIRO-Theoretical Informatics and Applications	25

Table 2.6: Journals/Conferences reporting most related research

Publication source	**J/C/S/W**	**#**	**N**
IEEE International Symposium on Foundations of Computer Science (FOCS)	S	4	4
Theoretical Computer Science	J	23	15
ACM Symposium on Theory of Computing (STOC)	S	3	3
International Computing and Combinatorics Conference (COCOON)	C	5	3
Journal of Theoretical Physics	J	7	2
Journal of Foundations of Computer Science	J	5	4
Proceeding of Quantum Computation and Learning Workshop	W	4	2
RAIRO: Theoretical Informatics and Applications	J	7	7
Symposium on Fundamentals of Computation	S	3	1
SIAM Journal on Computing	J	2	1
Quantum Information and Computation	J	3	3
International Conference on Developments in Language Theory (DLT)	C	4	4
Symposium on Fundamentals of Computer Theory (FCT, Springer)	S	3	1
Journal of Natural Computing	J	5	2
Information Processing Letters	J	5	5
International Symposium on Algorithms and Computation (ISAAC, Springer)	S	1	1
arXiv: preprint	J	18	8

Table 2.6 lists the Journals and Conferences publishing quantum finite automata related research, where (J, Journal; C, Conference; S, Symposium; W, Workshop; N, number of studies reporting related research as prime study; #, the total number of articles investigated). We observed that conferences like IEEE Symposium on Foundations of Computer Science (FOCS), ACM Symposium on Theory of Computing (STOC), International Conference on Computing and Combinatorics Conference (COCOON), Quantum Computation and Learning Workshop contribute a large part of research articles. Premier journals like Theoretical Computer Science, Journal of

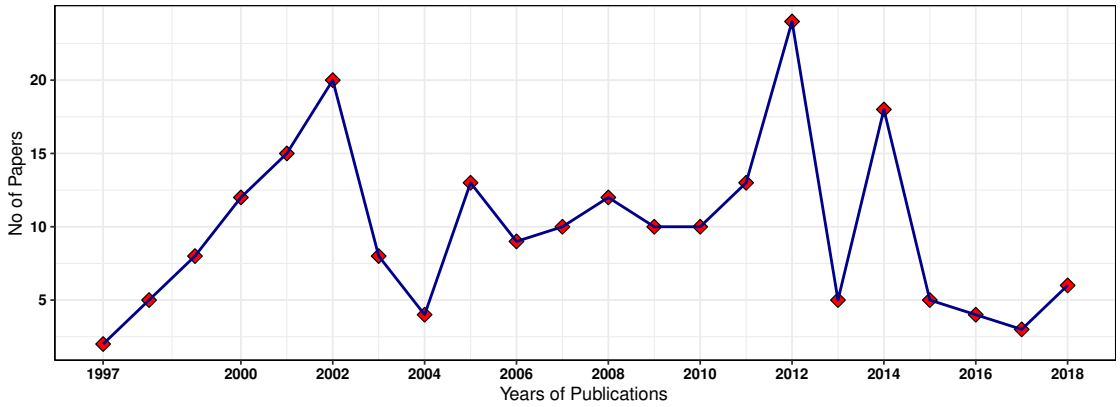


Figure 2.7: Number of papers per year in a review

Figure 2.8: Author's publications

Theoretical Physics, Journal of Foundations of Computer Science, RAIRO: Theoretical Informatics and Applications contributed significantly to our review area.

Chapter 3

Two-way Multihead Quantum Finite Automata

In this chapter, we proposed a variant of two-way quantum finite automata named two-way multihead quantum finite automata. A two-way quantum finite automaton is more powerful than its classical model. However, the generalizations of two-way quantum finite automata have not been defined so far as compared to one-way quantum finite automata model. We have investigated the newly introduced automata from two aspects: the language recognition capability and its comparison with classical and quantum counterparts. It has been shown that the class of languages recognized by k-head two-way quantum finite automata is a subset of the class of languages recognized by $(k+1)$-head two-way quantum finite automata. Furthermore, it has been investigated that quantum variant of two-way deterministic multihead finite automata takes less number of heads to recognize a language containing all words whose length is a prime number.

3.1 Introduction and motivation

In theoretical computational theory, research on two-way multihead finite automata has a long history. In the 1960s, Rabin and Scott [75] introduced a multi-tape or multihead finite state automata. Rosenberg [76] studied the basic properties of mutihead finite state automata and also examined their language recognition power. Kutrib and Malcher [77] introduced the variant of one-way reversible automata by introducing multiheads and investigated their properties. Furthermore, it has been proved that one-way two-head finite automata can recognize all uniletter regular languages. In 1970, Ibarra [78] introduced two-way multihead automata and proved that $(n+2)$-head deterministic two-way finite automata are more powerful than n-head deterministic two-way automata, where $n \geq 1$. Further, it has shown that the class of languages recognized by n-head deterministic/non-deterministic two-way pushdown automata is a subset of the class of languages recognized by $(n+1)$-head deterministic/non-deterministic two-way pushdown automata.

Morita [79] introduced a two-way reversible multi-head finite automaton (RMFA) and studied its language recognizing capability. It has shown that it can recognize various context-free and context-sensitive languages with less number of heads. After introducing the concept of two-way reversible multihead finite automata, Morita [80] investigated that deterministic two-way multihead finite automata (DMFA) can be

simulated by its reversible model with the same number of heads. It has been proved that DMFA(k) can be converted into RMFA(k). Further, it has been proved that a language containing all words of prime number can be recognized by DMFA and RMFA with three heads. Duris and Galil [81] designed a technique to prove that $L = \{x_0\#x_1\#...\#x_k \mid k \geq 1\}, x_j \in \{a,b\}^*$ for $0 \leq j \leq k$, with $x_i = x_0$ for some $1 \leq i \leq k$ cannot be recognized by any two-way simple deterministic two-head finite automata (2-2sDFA) and by any two-way deterministic counter machine. It is a special case of 2DFA with k-heads except that (k-1) heads are blind, they can read only end-markers.

In 1997, Kondacs and Watrous [14] defined the quantum analogue of 2-way deterministic finite state automata named *2-way quantum finite automata* (2QFA). They demonstrated that it was more powerful than the 2-way finite automata. In 2QFA model, the tape head can read the input tape bi-directionally, or it can be stationary. It is more dominant than the classical model because it allows quantum parallelism with the superposition of states on the input tape. Moreover, 2QFA recognizes regular languages, some context-free languages with one-sided error and also context-sensitive languages linearly.

Till now, in quantum automata theory, multihead variants of various quantum automata models have been proposed, and their language recognition capabilities have been examined. Ambainis and Watrous [33] introduced two-way quantum finite automata with classical states (2QCFA). Jeronimo and Moura [82] introduced a variant of 2QCFA model based on markers named k-marker two-way finite automata with quantum and classical states (k-M2QCFA). It has been investigated that k-M2QCFA is more powerful than its classical counterparts concerning computation and complexity. Further, it has shown that a language $L_{pow(k)} = \{a^n b^{n^k} \mid n > 1, k \in N\}$ cannot be used to attain a computability separation between quantum and classical multihead automata for $k \geq 3$.

Recently, Ganguly et al. [83] proposed one-way multihead quantum finite automata (k-1QFA) and proved that it could recognize all unary languages. It has been shown that a language $L = \{w \neq w^R \mid w \in \{a,b\}^*\}$ cannot be recognized by 1-way deterministic 2-head finite automata (2-1DFA), but can be recognized by 2-1QFA. Furthermore, it has been investigated that it is more powerful than 1-way reversible 2-head finite automata. Ganguly et al. [84] introduced two-tape one-way quantum finite automata (2-2T1QFA) and claimed that it could recognize all regular languages. It has been investigated that a language which cannot be recognized by any deterministic multihead finite automata can be recognized by 2-2T1QFA.

Although, the various generalizations of 2QFA such as multi-dimensional 2QFA, multihead 2QFA, and 2QFA with more observables can be defined. In this chapter, we started the investigation of 2QFA by introducing multiheads for refining the language recognition power. It has been proved that 2-2MQFA has more language recognizing power than 1-2QFA and 2QCFA. We have shown that 2-2MQFA can recognize context-free and context-sensitive languages such as $L = \{ww \mid w \in \{a,b\}^*\}, L = \{xay \mid x,y \in \{a,b\}^*, \mid x \mid = \mid y \mid\}$ and $L = \{www \mid w \in \{a,b\}^*\}$. Although, we find that in a 2-2MQFA where transition matrices are only 0 and 1, it is a two-way reversible multihead finite automata (RMFA). So, 2MQFA can recognize all the languages recognized by RMFA. Morita [12] claimed that RMFA takes few heads to recognize various non-regular and non-context free languages. We have proved that 2MQFA takes a less number of heads as compared to RMFA

to recognize a prime language by exploiting quantum parallelism.

3.1.1 Motivation

- 1QFA models are simpler and widely studied by various researchers. Its different variants have been proposed such as k-letter 1QFA, multihead 1QFA, generalized 1QFA and many more.

- 2QFA models have not been extensively examined as compared to one-way models [16]. 2QFA is more dominant than classical 2DFA and two-way probabilistic finite automata (2PFA). A language is recognized by 2PFA in exponential time, which can be recognized by 2QFA with bounded error in linear time.

- A multihead finite automaton can be considered as a system of finite automata which read the same input string simultaneously. Any two automata can see each other if and only if they reading the symbol at similar index of an input string. Due to common finite state control, each automaton knows the state of any other automaton in the present step. Following the superposition principle, the tape head moves deterministically in all computational paths and one head does not know the exact position of head in another path. Although, all computational paths can be evaluated simultaneously. Motivated by the computational power of superposition principle and entanglement, it is natural to evaluate the language recognition capability of quantum finite automata with multiheads.

- By exploiting the superposition property of quantum automata, we have introduced the two-way multihead quantum finite automata and determined its language recognition power.

3.2 Definitions

In this section, we give results and formal definitions of two-way deterministic finite automata, two-way reversible finite automata, two-way deterministic multihead finite automata, and two-way multihead quantum finite automata.

Definition 3.2.1. [85] *A two-way deterministic finite automaton M_{2DFA} is defined as octuplet $(Q, \Sigma, \#, \$, s, t, r, \delta)$, where*

- *Q is a set of states.*

- *Σ is an input alphabet,*

- *$\#$ is a left-end marker ($\# \notin \Sigma$),*

- *$\$$ is a right-end marker ($\$ \notin \Sigma$)*

- *s is a starting state $s \in Q$,*

- *t is an accepting state such that $t \subset Q$,*

- *r is a rejecting state $r \subset Q$ such that $s \neq t$,*
- *Transition function δ is defined by $\delta : Q \times (\Sigma \cup \{\#, \$\}) \longrightarrow Q \times (\leftarrow, \rightarrow)$,where $(\leftarrow, \rightarrow)$ represents the head function for the left and right direction of tape head.*

At any instance, 2DFA is in any state q and read some symbol a or an end-markers, based on q and the current symbol, it will move its head one cell in direction $(\leftarrow, \rightarrow)$ and returns a new resultant state p. The tape head does not move beyond the end-markers.

Definition 3.2.2. [86] *A two-way reversible finite automaton is defined as octuplet $(Q, \Sigma, \#, \$, s, t, r, \delta)$. The definition of 2RFA is same as 2DFA; the only difference lies in the pair of transitions (i.e., the transitions from two different states reading the same symbol cannot lead to single state such that $\delta(q_1, a) = (p, \rightarrow), \delta(q_2, a) = (p, \rightarrow)$ the states q_1 and q_2 must be the same.*

It is a special case of two-way deterministic finite automata, in which every step of computation is logically reversible, i.e., every configuration induces a partial one-to-one mapping from the set of states into itself. It has been investigated that 2RFA is equivalent to 1DFA and 2DFA in language recognition power [14]. The transition matrices of reversible automata consist only 0 and 1. Each row and column has exactly only one entry 1. The transition matrix of two-way reversible finite automata (2RFA) must satisfy the following two properties: (1) The dot product of any two rows of transition matrix is zero, (2) There should not be any cycle within the transition matrix that cannot be accessed from one of the input states. To illustrate its working, a 2RFA has been designed for the language $L = \{a^*b^*\}$ can be defined.

Define $M_{2RFA} = (Q, \Sigma, \#, \$, s, t, r, \delta)$ as follows. Let $Q = \{q_0, q_1, q_2, q_3, q_4\}, \Sigma = \{a, b\}, s = \{q_0\}$, $t = \{q_3\}, r = \{q_4\}$. Define transition function as:

$$\begin{aligned} M_\sigma(i, j) &= 1, \text{ if } (q_i, \sigma) \rightarrow q_j, \\ &= 0, \text{ otherwise} \end{aligned}$$

where $\sigma \in (\Sigma \cup \{\#, \$\})$.

$$M_a = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}, M_b = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Take an example of the input string $\{w = \#aab\$\} \in L$ enclosed with both end-markers. On reading the left-end marker, the head moves toward the right direction. On reading the input symbol $a's$, the state of M_{2RFA} remains q_0. On reading the symbol b, the state is transformed into q_1 and tape head moves one step backward. Further, the tape head reads the last symbol a and state is changed into q_2. On reading the symbol b, the state remains same. Finally, on reading the right-end marker \$, the automaton goes to acceptance state $|q_3\rangle$ and the input string is said to be recognized by M_{2RFA}.

Definition 3.2.3. [80] *A two-way multihead finite automaton M_{2MFA} is defined as nonuple $(Q, \Sigma, k, \#, \$, q_0, A, R, \delta)$, where*

- *Q is a non-empty finite set of states,*
- *Σ is an input alphabet,*
- *k is a number of heads ($k \in 1, 2, ..$),*
- *$\#$ is a left-end marker ($\# \notin \Sigma$),*
- *$\$$ is a right-end marker ($\$ \notin \Sigma$)*
- *q_0 is a starting state,*
- *A is an accepting state such that $A \subset Q$,*
- *R is a rejecting state $R \subset Q$ such that $A \cap R = \phi$,*
- *Transition function δ is a subset of $Q \times ((\Sigma \cup \{\#, \$\})^k \cup \{\leftarrow, \uparrow, \rightarrow\}^k \times Q)$, where $(\leftarrow, \uparrow, \rightarrow)$ represent the left, stationary, and the right direction of tape head.*

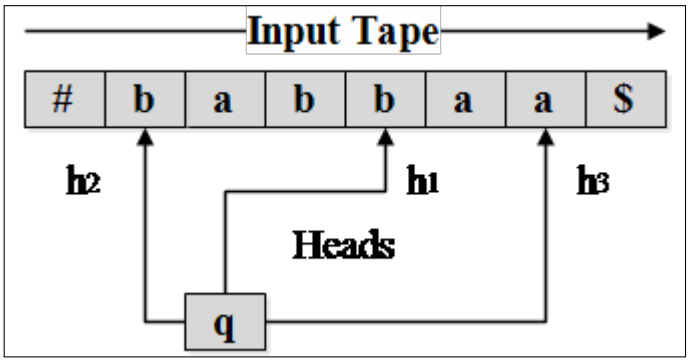


Figure 3.1: Two-way multihead finite automata

The right and left-end markers are not included in Σ. To process the input string by M_{2MFA}, we assume that input string x is written on input tape with both end-markers such that $\#x\$$. The automaton is in any state q and for any $h \in \{0, 1, ..., \mid x \mid +1\}^k$. A triplet $\{\#x\$, q, h\}$ is called configuration of 2MFA on x. It defines a function $s_x : \{0, 1, ..., \mid x \mid +1\}^k \rightarrow (\Sigma \cup \{\#, \$\})^k$. It generates a k-tuple of symbols in $\#x\$$ which are read by k-heads of 2MFA at the position h. Now, we define the notion of deterministic and reversibility for multihead finite automata.

A multihead finite automata [80] M_{MFA} is said to be deterministic if it holds the following condition:

$$\forall r_1[q_1, s_1, d, q_1^{'}] \in \delta, \forall r_2[q_2, s_2, d, q_2^{'}] \in \delta((q_1 = q_2 \wedge s_1 = s_2) \implies (r_1 = r_2)) \quad (3.1)$$

The q_1 state of M_{MFA} moves the tape head according to the $d = (\leftarrow, \uparrow, \rightarrow)$ on reading the symbol s_1 and the resultant state is uniquely determined. Similarly, M_{MFA} is said to be reversible if the following condition holds:

$$\forall r_1[q_1, s_1, d, q_1^{'}] \in \delta, \forall r_2[q_2, s_2, d^{'}, q_2^{'}] \in \delta((q_1^{'} = q_2^{'} \wedge r_1 \neq r_2) \implies (d = d^{'} \wedge s_1 \neq s_2)) \quad (3.2)$$

Theorem 3.2.1. *For any two-way deterministic multihead finite automata, we can design two-way reversible finite automata with the same number of heads which recognizes the same language as the former.*

Proof. The proof has been shown in [80]. □

Theorem 3.2.2. *Two-way quantum finite automata can recognize all regular languages.*

Proof. The proof has been shown in [14]. □

Definition 3.2.4. *A two-way multihead quantum finite automaton $M_{k-2MQFA}$ is a sextuple $(Q, \Sigma, q_0, Q_{acc}, Q_{rej}, \delta)$, where*

- *Q is a set of states. Moreover, $Q = Q_{acc} \cup Q_{rej} \cup Q_{non}$, where $Q_{acc}, Q_{rej}, Q_{non}$ represent the set of accepting, rejecting, and non-halting states, respectively.*
- *Σ is an input alphabet,*
- *Transition function δ is defined as $\delta : Q \times \Gamma^k \times Q \times D^k \to \mathbb{C}$, where $\Gamma \in (\Sigma \cup \{\#, \$\})^k$, $D = \{\leftarrow, \uparrow, \rightarrow\}$ represent the left, stationary, and right direction of tape head. The computation process of $M_{k-2MQFA}$ is as follows:*

Consider a $M_{k-2MQFA}$, at instance, it is in any state q and with the amplitude $\delta(q, \sigma_1, \sigma_2, ..., \sigma_k, q^{'}, d_1, d_2, ..., d_k)$ moves to state $q^{'}$ after reading the input symbols $(\sigma_1, \sigma_2, ..., \sigma_k)$ by k-heads and moves the heads according to $(d_1, d_2, ..., d_k)$ respectively. Similar to simplified 2QFA, for each $(\sigma_1, \sigma_2, ..., \sigma_k) \in \Gamma$, if there exists a unitary linear operator $V_{(\sigma_1,\sigma_2,...,\sigma_k)}$ on the inner product space such that $L_2(Q) \to L_2(Q)$, where Q is the set of states and a function $D : Q \to \{\leftarrow, \uparrow, \rightarrow\}$. Define transition function as

$$\delta(q, \sigma_1, \sigma_2, ..., \sigma_k, q^{'}, d_1, d_2, ..., d_k) = \begin{cases} \left\langle q^{'} \middle| V_{(\sigma_1,\sigma_2,...,\sigma_k)} \middle| q \right\rangle & \text{if } D(q^{'}) = (d_1, d_2, ..., d_k) \\ 0 & \text{else} \end{cases} \tag{3.3}$$

where $\left\langle q^{'} \middle| V_{(\sigma_1,\sigma_2,...,\sigma_k)} \middle| q \right\rangle$ is a coefficient of $|q^{'}\rangle$ in $V_{(\sigma_1,\sigma_2,...,\sigma_k)} |q\rangle$ is said to be well-formed iff

$$\sum_{q^{'} \in Q} \left\langle q^{'} \middle| V_{(\sigma_1,\sigma_2,...,\sigma_k)} \middle| q_1 \right\rangle \left\langle q^{'} \middle| V_{(\sigma_1,\sigma_2,...,\sigma_k)} \middle| q_2 \right\rangle = \left\{ \begin{array}{l} 1 \; q_1 = q_2 \\ 0 \; q_1 \neq q_2 \end{array} \right\} \tag{3.4}$$

for each $V_{(\sigma_1,\sigma_2,...,\sigma_k)}$. Equivalently, it is well-formed when every $V_{(\sigma_1,\sigma_2,...,\sigma_k)}$ is unitary.

3.2.1 Language recognition

A two-way k-head quantum multihead finite automaton is defined as 2QFA in which k-heads reading the input string from single input tape and cannot move beyond the end-markers. We assume that 2MQFA has to be observed to produce information about its processing. Consider an observable O for finite-dimensional Hilbert space $\mathcal{H}_n$, which is decomposed into subspaces such as E_a, E_r, E_n refers to the subspace of 'accept', 'reject' and 'non-halting' respectively. Each of these subspaces is traversed by configurations.

We assume that the input string $x \in \Sigma^*$ with both end-markers such that $\#x\$$ is written on an input tape. The processing of an input string starts with an initial state and heads pointing towards the left-end marker. Each computation consists of two linear operations. Firstly, the unitary evolution operator is applied to the current state, and the second one is a measurement. It computes several paths simultaneously (quantum parallelism). Generally, N qubits form 2^N computational paths. In any mathematical operation, all 2^N paths can be involved simultaneously. In other words, the same operation is performed on all 2^N computational paths and result is evaluated simultaneously. This phenomenon is called quantum parallelism. Then, the result is observed by an observable O. During computation, it is the branching and quantum interference that creates parallelism and constructive/destructive superpositions of states result in amplifying or destruct the effects of some computation.

Suppose if the automaton is in a superposition state $|\phi\rangle = \alpha_1 |x_1\rangle + \alpha_2 |x_2\rangle + ... + \alpha_n |x_n\rangle$ where α_i's are amplitudes and $|\alpha_1|^2 + |\alpha_2|^2 + ... + |\alpha_n|^2 = 1$, then the superposition is projected into above-mentioned subspaces $E_j, j \in \{a, r, n\}$. The result of each observation will be either 'accept' or 'reject' or 'non-halting'. The probability of each outcome is determined by the amplitudes of an associated configuration in the present superposition. However, these amplitudes are complex numbers; parallel computation paths can interfere with each other (if their amplitudes cancel each other, i.e., destructive interference otherwise constructive). Interference is the act of quantum qubits operating as a wave, whose amplitude depends on the phase ($|\phi\rangle$). These waves can work in unison or opposite of each other. When the two waves are in-phase ($|\phi\rangle$=0), they interfere constructively and the result has twice the amplitude of the individual waves, creating constructive interference. When the two waves are out of phase ($|\phi\rangle$=180), their amplitudes cancel out, causing destructive interference. The computation is over if the observation results in the accepting or rejecting state. Sometimes, the computation paths may end in different perpendicular states during observation of some configuration. It results in constructive interference which holds $| \sum_{i=1}^{n} \alpha_i |^2 > \sum_{i=1}^{n} | \alpha_i |^2$. After each measurement, the superposition collapses to the measured subspaces. The outcome of the observation may result in a superposition of halting and non-halting states. Then, the computation continues and superposition collapses to non-halting state. If the amplitude of halting states in some superposition is zero, then the outcome of observation in non-halting subspace is measured with probability 1. Therefore, superposition remains unchanged by the observation.

3.3 Computational Power of 2MQFA

In this Section, we have shown the language recognition power of 2MQFA for various languages. We have shown that a 2QFA with $(k+1)$ heads is more powerful than k-head 2QFA for $k \geq 1$ and 2QCFA. Therefore, a language which cannot be recognized by 2QFA can be recognized by 2MQFA with two heads. First of all, we have shown the superiority of 2-2MQFA over DMFA and RMFA based on the number of heads.

Theorem 3.3.1. *$L_1 = \{w^n \mid n > 1, w \in \{a\}\}$ where the length of w is a prime number, can be recognized by 3-DMFA and 3-RMFA [79, 80]. However, it can be recognized by a two-way multihead quantum finite automaton with 2-heads.*

Proof. Let $M_{2-2MQFA} = (Q, \Sigma, q_0, Q_{acc}, Q_{rej}, \delta)$ be a 2MQFA with 2-heads, $Q = \{q_0, Q_{acc}, Q_{rej}\} \cup \{X_{s,i}, Y_{s,i} \mid 2 \leq s \leq 3 \mid 2 \leq i \leq n{+}3\} \cup \{q_{xi,i}, q_{yi,i}, q_{zi,i}, q_{bi,i}, q_{i,i}, z_{i,i}, q_{yi,i}, q_{ri,i}, q_{si,i}, q_{wi,i}, q_{ui,i}, q_{li,i}, q_{vi,i}, q_{ci,i}, q_{ti,i}, q_{gi,i}, f_{i,i}, g_{i,i}, q_{pi,k}, z_{i,k}, q_{ui,i}, q_{ui,i}, q_{ti,i}, q_{yi,i} \mid 2 \leq i \leq n, 1 \leq k \leq max(i-1)\} \cup \{q_{pi,0}, q_{ui,0}, q_{ti,0}, x_{i,0}, z_{i,0} \mid 2 \leq i \leq n\}, \Sigma = \{a\}, Q_{acc} = \{q_{acc}\}, Q_{rej} = \{q_{rej}\}$. Each $V_{(\sigma_1,\sigma_2,...,\sigma_k)}$ is unitary by inspection, so $M_{2-2MQFA}$ for L_1 is well-formed. The specification of transition functions and head functions are as follows:

Table 3.1: Details of the transition functions and head functions for L_1

$V_{\#,\#}\left\|q_0\right\rangle = \left\|X_{s,s}\right\rangle, V_{\#,a}\left\|X_{s,s}\right\rangle = \left\|Y_{s+1,s+1}\right\rangle, V_{a,a}\left\|Y_{s,k}\right\rangle = \left\|X_{s-1,k+1}\right\rangle,$
$V_{a,a}\left\|X_{s,l}\right\rangle = \left\|Y_{s,l+1}\right\rangle, V_{a,\$}\left\|Y_{s,l}\right\rangle = \left\|q_{s,l}\right\rangle, V_{a,\$}\left\|X_{s,l}\right\rangle = \left\|q_{s,l}\right\rangle$
$V_{a,a}\left\|q_{s,i}\right\rangle = \frac{1}{\sqrt{2}}\left\|q_{xi,i}\right\rangle + \frac{1}{\sqrt{2}}\left\|q_{yi,i}\right\rangle$, for $i = sk+1, k = 1, 2, 3, ...$
$V_{a,a}\left\|q_{s,i}\right\rangle = \frac{1}{\sqrt{2}}\left\|q_{yi,i}\right\rangle + \frac{1}{\sqrt{2}}\left\|q_{zi,i}\right\rangle$, for $i = (s+1)k, k = 1, 2, 3, ...$
$V_{a,a}\left\|q_{xi,i}\right\rangle = \left\|q_{pi,i}\right\rangle, V_{a,a}\left\|q_{pi,i-1}\right\rangle = \left\|q_{pi,k}\right\rangle, V_{a,a}\left\|q_{pi,k}\right\rangle = \left\|q_{pi,k-1}\right\rangle$
$V_{\#,\#}\left\|q_{pi,0}\right\rangle = \left\|z_{i,i+1}\right\rangle, V_{a,a}\left\|z_{i,k}\right\rangle = \left\|z_{i,k+1}\right\rangle, V_{a,a}\left\|z_{i,i}\right\rangle = \left\|f_{i,i}\right\rangle$
$V_{a,a}\left\|q_{yi,i}\right\rangle = \left\|x_{i,i}\right\rangle, V_{a,a}\left\|q_{y1,1}\right\rangle = \left\|q_{acc}\right\rangle, V_{a,a}\left\|q_{xi,i-1}\right\rangle = \left\|x_{i,k}\right\rangle, V_{a,a}\left\|q_{xi,k}\right\rangle = \left\|x_{i,k-1}\right\rangle$
$, V_{\#,a}\left\|x_{i,0}\right\rangle = \left\|z_{i,0}\right\rangle, V_{a,\#}\left\|z_{i,0}\right\rangle = \left\|z_{i,i+1}\right\rangle, V_{a,\#}\left\|z_{i,k}\right\rangle = \left\|z_{i,k+1}\right\rangle, V_{a,a}\left\|z_{i,k}\right\rangle = \left\|z_{i,k+1}\right\rangle$
$V_{a,a}\left\|z_{i,i}\right\rangle = \left\|g_{i,i}\right\rangle, V_{a,a}\left\|g_{i,i}\right\rangle = \left\|g_{i,i}\right\rangle, V_{a,\$}\left\|g_{i,i}\right\rangle = \left\|k_{i,i}\right\rangle, V_{a,a}\left\|k_{i,i}\right\rangle = \left\|f_{i,i}\right\rangle, V_{a,a}\left\|q_{zi,i}\right\rangle = \left\|q_{bi,i}\right\rangle, V_{a,a}\left\|q_{bi,i}\right\rangle = \left\|x_{i,i}\right\rangle$
$V_{a,a}\left\|f_{i,i}\right\rangle = \frac{1}{\sqrt{2}}\left\|q_{ri,i}\right\rangle + \frac{1}{\sqrt{2}}\left\|q_{si,i}\right\rangle$, for $i > 3$
$V_{a,\#}\left\|z_{i,k}\right\rangle = \frac{1}{\sqrt{2}}\left\|q_{acc}\right\rangle \pm \frac{1}{\sqrt{2}}\left\|q_{rej}\right\rangle, \forall k \neq i, 0 \leq k < 2$
$V_{a,\#}\left\|z_{i,k}\right\rangle = \pm\frac{1}{\sqrt{2}}\left\|q_{acc}\right\rangle + \frac{1}{\sqrt{2}}\left\|q_{rej}\right\rangle, \exists k = i, 2 \leq k \leq 3$
$V_{a,a}\left\|q_{ri,i}\right\rangle = \left\|q_{wi,i}\right\rangle, V_{a,a}\left\|q_{w(i-2),i-2}\right\rangle = \left\|q_{ui,i}\right\rangle, V_{a,a}\left\|q_{ui,i}\right\rangle = \left\|q_{ui,i-1}\right\rangle,$
$V_{a,a}\left\|q_{ui,k}\right\rangle = \left\|q_{ui,k-1}\right\rangle, V_{\#,a}\left\|q_{ui,0}\right\rangle = \left\|p_{i,i+1}\right\rangle, V_{a,a}\left\|p_{i,k}\right\rangle = \left\|p_{i,k+1}\right\rangle,$
$V_{a,a}\left\|p_{i,k}\right\rangle = \left\|q_{ui,i+1}\right\rangle$, for $k = i$,
$V_{a,\#}\left\|q_{ui,k}\right\rangle = \frac{1}{\sqrt{2}}\left\|q_{acc}\right\rangle + \frac{1}{\sqrt{2}}\left\|q_{rej}\right\rangle$, for $3 < i \leq n-2$
$V_{a,\#}\left\|q_{ui,k}\right\rangle = \frac{1}{\sqrt{2}}\left\|q_{acc}\right\rangle \pm \frac{1}{\sqrt{2}}\left\|q_{rej}\right\rangle$, for $2 \leq i \leq 3$
$V_{a,\#}\left\|p_{i,k}\right\rangle = \frac{1}{\sqrt{2}}\left\|q_{acc}\right\rangle + \frac{1}{\sqrt{2}}\left\|q_{rej}\right\rangle, V_{\#,\#}\left\|q_{ui,0}\right\rangle = \frac{1}{\sqrt{2}}\left\|q_{acc}\right\rangle \pm \frac{1}{\sqrt{2}}\left\|q_{rej}\right\rangle$
$V_{a,a}\left\|q_{si,i}\right\rangle = \left\|q_{li,i}\right\rangle, V_{a,a}\left\|q_{li,i}\right\rangle = \left\|q_{vi,i}\right\rangle, V_{a,a}\left\|q_{vi,i}\right\rangle = \left\|q_{ci,i}\right\rangle, V_{a,a}\left\|q_{ci,i}\right\rangle = \left\|q_{t(i-4),i-4}\right\rangle$
$V_{a,a}\left\|q_{ti,i}\right\rangle = \left\|q_{ti,i-1}\right\rangle, V_{a,a}\left\|q_{ti,k}\right\rangle = \left\|q_{ti,k-1}\right\rangle, V_{a,a}\left\|q_{ti,0}\right\rangle = \left\|q_{ri,k+1}\right\rangle,$
$V_{a,a}\left\|q_{ri,k}\right\rangle = \left\|q_{ri,i+1}\right\rangle, V_{a,\#}\left\|q_{ri,i}\right\rangle = \left\|q_{ti,i-1}\right\rangle, V_{a,a}\left\|q_{ri,k}\right\rangle = \left\|q_{yi,k+1}\right\rangle, 0 \leq k < i$
$V_{a,\#}\left\|q_{ti,k}\right\rangle = \left\|q_{yi,k+1}\right\rangle, k \neq 0,\ V_{\#,\#}\left\|q_{ti,0}\right\rangle = \left\|q_{yi,k+1}\right\rangle, V_{a,\#}\left\|q_{yi,k}\right\rangle = \left\|q_{yi,k+1}\right\rangle$
$V_{a,\#}\left\|q_{yi,i}\right\rangle = \left\|q_{fi,i}\right\rangle$, for $3 < i \leq n-2, V_{a,a}\left\|q_{fi,i}\right\rangle = \left\|q_{fi,i}\right\rangle, V_{a,\$}\left\|q_{fi,i}\right\rangle = \left\|q_{gi,i}\right\rangle, V_{a,a}\left\|q_{gi,i}\right\rangle = \left\|q_{fi,i}\right\rangle$
$V_{a,\#}\left\|q_{yi,k}\right\rangle = \frac{1}{\sqrt{2}}\left\|q_{acc}\right\rangle \pm \frac{1}{\sqrt{2}}\left\|q_{rej}\right\rangle, \forall k = i$, for $2 \leq i \leq 3$
$V_{a,\#}\left\|q_{yi,k}\right\rangle = \pm\frac{1}{\sqrt{2}}\left\|q_{acc}\right\rangle + \frac{1}{\sqrt{2}}\left\|q_{rej}\right\rangle, \exists k \neq i$, for $2 \leq i \leq 3$
Head Functions:

$D(q_0) = (\uparrow, \uparrow), D(X_{s,s}) = (\uparrow, \rightarrow)$, for $s = 1, D(Y_{s,k}) = (\rightarrow, \rightarrow), D(X_{s,l}) = (\rightarrow, \rightarrow)$, for $1 \leq l, k \leq n, D(q_{s,i}) = (\uparrow, \leftarrow)$, for $2 \leq i \leq n, D(q_{xi,i}) = (\uparrow, \uparrow), D(q_{yi,i}) = (\leftarrow, \uparrow), D(q_{zi,i}) = (\leftarrow, \uparrow), D(q_{bi,i}) = (\leftarrow, \uparrow), D(q_{pi,i}) = (\leftarrow, \leftarrow), D(q_{pi,k}) = (\leftarrow, \leftarrow), D(z_{i,0}) = (\uparrow, \uparrow), D(z_{i,k}) = (\rightarrow, \leftarrow)$, for $k \neq 0, D(f_{i,i}) = (\uparrow, \uparrow), D(x_{i,i}) = (\leftarrow, \leftarrow), D(x_{i,k}) = (\leftarrow, \leftarrow)$, for $k \neq 0, D(g_{i,i}) = (\uparrow, \rightarrow), D(k_{i,i}) = (\uparrow, \leftarrow), D(q_{ri,i}) = (\uparrow, \uparrow), D(q_{si,i}) = (\uparrow, \uparrow), D(q_{wi,i}) = (\leftarrow, \uparrow), D(q_{ui,i}) = (\leftarrow, \uparrow), D(q_{ui,k}) = (\leftarrow, \leftarrow), D(p_{i,k}) = (\rightarrow, \leftarrow), D(q_{li,i}) = (\leftarrow, \uparrow), D(q_{vi,i}) = (\leftarrow, \uparrow), D(q_{ci,i}) = (\leftarrow, \uparrow), D(q_{ti,i}) = (\leftarrow, \uparrow), D(q_{ti,k}) = (\leftarrow, \leftarrow)$, for $k \neq 0, D(q_{ti,0}) = (\uparrow, \uparrow), D(q_{ri,k}) = (\rightarrow, \leftarrow)$, for $k \neq i, D(q_{ri,i}) = (\uparrow, \uparrow), D(q_{yi,k}) = (\rightarrow, \uparrow), D(q_{fi,i}) = (\rightarrow, \rightarrow), D(q_{gi,i}) = (\uparrow, \leftarrow), D(q_{rej}) = (\uparrow, \uparrow), D(q_{acc}) = (\uparrow, \uparrow)$

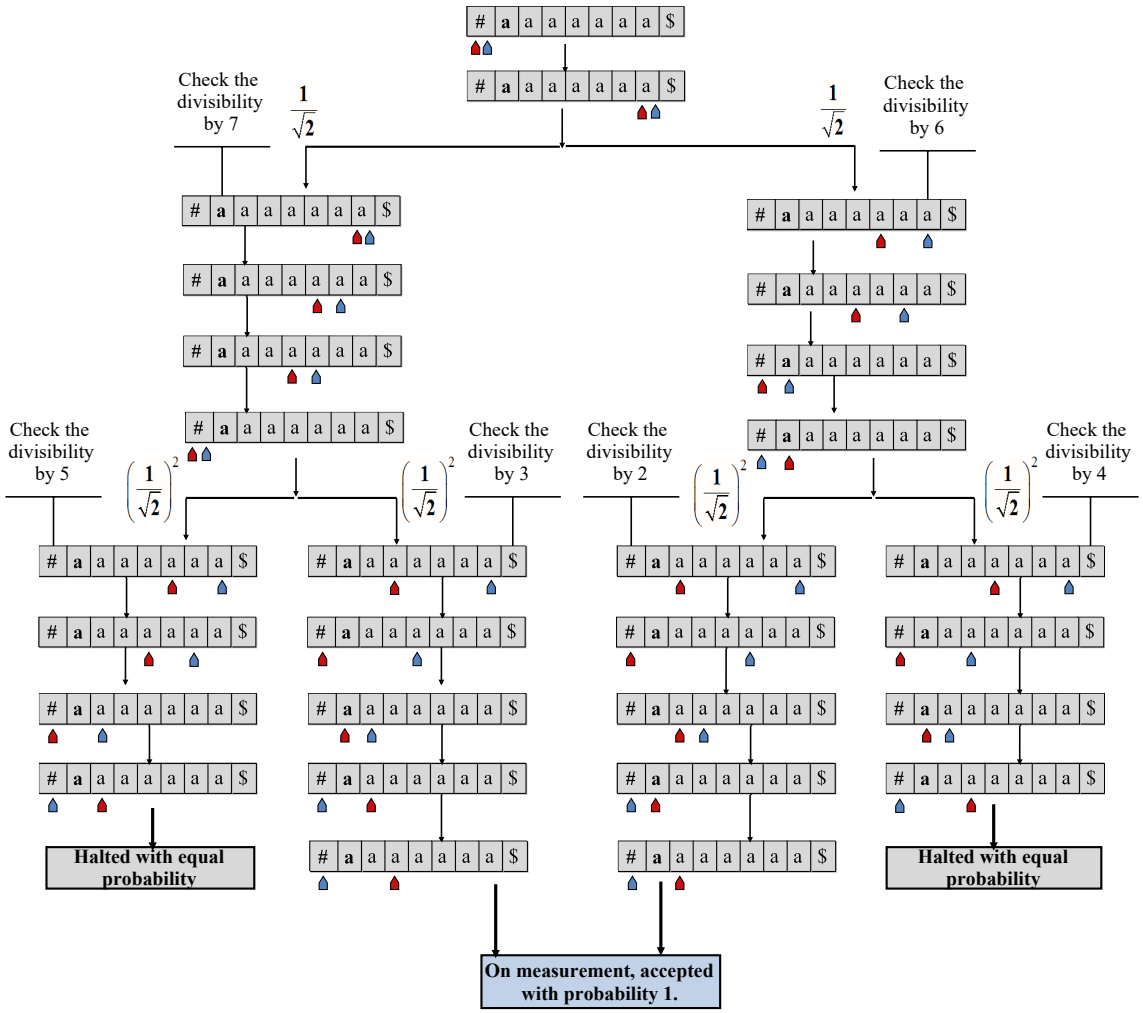


Figure 3.2: Computation process of language L_1 for the input $x \in L_1$

By the construction of 2-2MQFA for language L_1, we see that computation consists of various paths to check whether the input string x is of the prime number or not. An input string is said to be of prime number if its length is divisible by a number itself. During computation, we will divide the input string with the number $(2 \leq i \leq n)$ or $(2 \leq i \leq n-1)$ after checking its length of odd or even number, respectively. Thus, the computation is divided into two paths with equal $\frac{1}{\sqrt{2}}$ amplitude. Further, we considered an input string of length 7. Now, we will

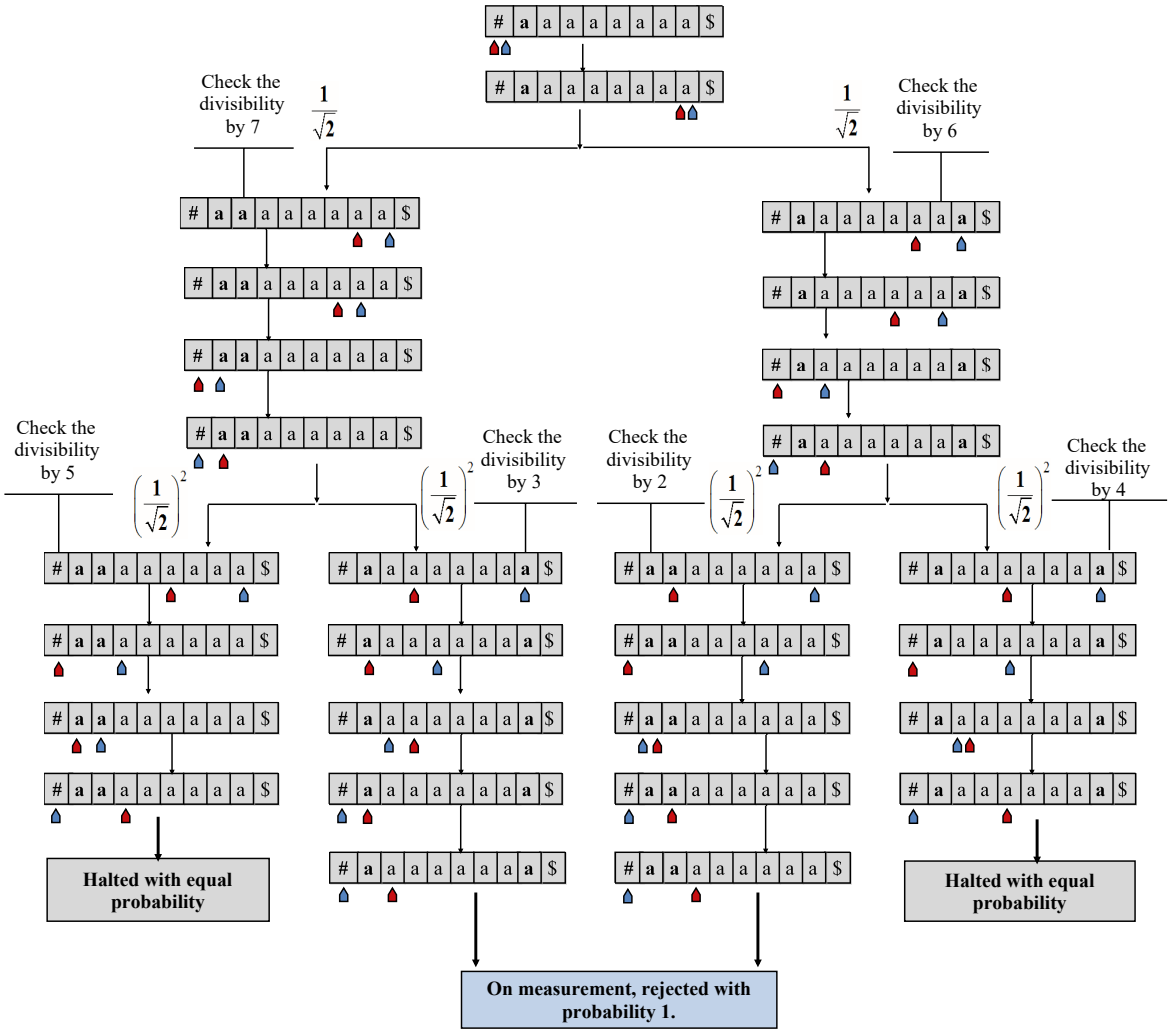


Figure 3.3: Computation process of language L_1 for the input $x \notin L_1$

check its divisibility from number 7 to 2. The computation process is further divided into equal parts on both sides. On observation, the computation paths are in a superposition of halting and non-halting states. Therefore, after checking divisibility with the input string, the computation paths are halted with equal probability in an acceptance and rejectance state. But, the computation remains to continue with the non-halting states. The input string is said to be accepted (prime number) when it is not divided by any of the numbers except itself. In the end, the second head of both computation paths read the left-end marker at the same time and other head points towards their respective position of a number. Thus, in case of a prime number, both computation paths leading to rejecting configuration interfere destructively, due to which their amplitude sums to 0. Hence, 2-2MQFA accepts the input string with probability 1. In case, if the left-end marker is read by the computation paths (both heads) at the same time, then the accepting configuration interferes destructively. It results in the rejection of an input string with probability 1.

Similarly, we consider an input string x of length 8. The computation process remains the same. But, the input string is divisible by more than one number. In the end, one head of computation reads the left-end marker at same time; but other head of one of the paths or both paths does not point towards their respective

position of a number. It shows that input string is divisible by (2 or/and 3). It leads to an accepting configuration interfere destructively. Thus, the string is said to be rejected with probability 1. Fig. 3.2, and Fig. 3.3 shows the pictorial representation of computation $x \in L_1$ and $x \notin L_1$. □

Theorem 3.3.2. *For k≥ 1, (k+1)-2MQFA is computationally more powerful than k-2QFA.*

Proof. Initially, we prove it for the case k=1. We claim that a language $L_2 = \{ww \mid w \in \{a,b\}^*\}$ cannot be recognized by k-2QFA, but it can be recognized by (k+1)-2MQFA (proved in Theorem 3.3.3). We assume that a language L_2 can be recognized by k-2QFA. Consider an input string $x \in L_2$, i.e., written on the input tape enclosed with end-markers such that $\#x\$$. On reading the left-end marker #, the computation is split in two paths $V_{\#}|q_0\rangle = \frac{1}{\sqrt{2}}|q_1\rangle + \frac{1}{\sqrt{2}}|q_2\rangle$. Now, in the first path along with state q_1, the tape head moves after remain stationary one time on reading each symbol. Meantime, in other paths, the head moves deterministically toward right-end marker \$. This process remains to continue until the head does not reach the right-end marker in second path. Suppose, along the second path, the head points towards the right-end marker, and simultaneously the head is somewhere in the middle of an input string in first path. Now, in order to match the contents, the tape head should be at the beginning of the second w in first path when the head is reading the right-end marker in the second path. However, the tape head moves deterministically in both paths and the first head does not know the position of head in another path. Regardless of the power of 2QFA, there are some languages which cannot be recognized by it. We get a contradiction that language L_2 is cannot be accepted by k-2QFA. But, it can be accepted by (k+1)-2MQFA (proved in Theorem 3.3.3).

Although, k-2QFA [5] with single head can recognize all regular languages, some non-regular languages $L = \{a^n b^n \mid n \geq 1\}$ and non-context free language $L = \{a^n b^n c^n \mid n \geq 1\}$. The power of 1-2QFA comes from the ability of their heads to be in superpositions of basis states corresponding to configurations with the heads in different positions, instead of one position as in the classical model. But, the relationship between 2QFA and its multihead variant is not known concerning language recognition capability.

Thus, it has been proved that 2-2MQFA is computationally more powerful than 1-2QFA. As a result, in Theorem 3.3.3, 3.3.4 and 3.3.5, we considered the languages $L_2 = \{ww \mid w \in \{a,b\}^*\}, L_3 = \{xay \mid x,y \in \{a,b\}^*, |x| = |y|\}$ and $L_4 = \{www \mid w \in \{a,b\}^*\}$ which cannot be recognized by simple 2QFA and 2QCFA. However, it has been proved that these languages can be recognized by 2QFA with 2-heads. Each transition $V_{(\sigma_1,\sigma_2,\ldots,\sigma_k)}$ is unitary and $M_{2-2MQFA}$ is well-formed for these languages. However, it is still an open problem to find more languages like these that witness the superiority of (k+1)-2MQFA over k-2QFA for $k\geq 2$. □

Theorem 3.3.3. *Language $L_2 = \{ww \mid w \in \{a,b\}^*\}$ is recognizable by 2MQFA with 2 heads.*

Proof. The idea of this proof is as follows. It consists of three phases. First, the initial state q_0 reads the first symbol, and both heads start moving towards the right-end marker \$. After reading a symbol, first head remains stationary at once and

other moves forward. If the input string $x \in L_2$, then first head is at the beginning of second w in the input string and the second head points towards the right-end marker \$. In the second phase, the symbols read by first head are getting matched with the symbols read by the second head and moves towards the left-end marker #. If any of the symbol read by both heads does not get matched, then it leads to rejecting state q_{rej}. Otherwise, L_2 is said to be accepted if all the symbols read by first head gets matched with the symbols read by the second head and first head is reading right-end marker # at the end. A 2-2MQFA for L_2 is defined as follows:

$$M_{2-2MQFA} = (Q, \Sigma, q_0, Q_{acc}, Q_{rej}, \delta),$$

where

- $Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$, where q_1, q_2, and q_3 are used to move the second head towards the \$ and first head remains stationary at once on reading the symbol, q_4 is used to match the symbols of both heads.
- $\Sigma = \{a, b\}$, q_0 is an initial state, $Q_{acc} = \{q_{acc}\}$ and $Q_{rej} = \{q_{rej}\}$.
- In transition functions $r, s \in \{a, b\}^*$, and its specifications are defined as:

Table 3.2: Details of the transition function and head function for L_2

$V_{\#,\#}\|q_0\rangle = \|q_0\rangle$	$V_{\#,r}\|q_0\rangle = \|q_1\rangle$	$V_{r,s}\|q_1\rangle = \|q_2\rangle$	$V_{r,\$}\|q_1\rangle = \|q_{rej}\rangle$
$V_{r,s}\|q_2\rangle = \|q_1\rangle$	$V_{r,\$}\|q_2\rangle = \|q_3\rangle$	$V_{a,a}\|q_4\rangle = \|q_4\rangle$	$V_{b,b}\|q_4\rangle = \|q_4\rangle$
$V_{r,\$}\|q_3\rangle = \|q_4\rangle$	$V_{\#,r}\|q_4\rangle = \|q_{acc}\rangle$	$V_{b,a}\|q_4\rangle = \|q_{rej}\rangle$	$V_{a,b}\|q_4\rangle = \|q_{rej}\rangle$
Head Functions:			
$D(q_0) = (\uparrow, \rightarrow), D(q_1) = (\rightarrow, \rightarrow), D(q_2) = (\uparrow, \rightarrow), D(q_3) = (\rightarrow, \uparrow),$ $D(q_4) = (\leftarrow, \leftarrow), D(q_{acc}) = (\uparrow, \uparrow), D(q_{rej}) = (\uparrow, \uparrow)$			

Each $V_{(\sigma_1,\sigma_2,...,\sigma_k)}$ is unitary, it obeying unitary time evolution defined by local interactions between its symbols which permitting being in a superposition of configurations for each $(\sigma_1, \sigma_2, ..., \sigma_k) \in \Gamma$. □

Theorem 3.3.4. *Language $L_3 = \{xay \mid x, y \in \{a, b\}^*, |x| = |y|\}$ is recognizable by 2MQFA with 2 heads.*

Proof. It consists of two phases. First, we keep the first head stationary at once and other moves forward on reading symbols towards the left-end marker \$. At the end of the first phase, if the central symbol is a, then it proceeds to second phase to check $|x| = |y$. Suppose, if the central symbol is not a, then L_3 is said to be rejected. In second phase, we keep the first head stationary for once, and other heads keep moving towards the left side. In the end, if both heads read the left-end marker at the same time, it is said to be accepted ($w \in L_3$). A 2-2MQFA for L_3 is defined as follows:

$$M_{2-2MQFA} = (Q, \Sigma, q_0, Q_{acc}, Q_{rej}, \delta),$$

where

- $Q = \{q_0, q_1, q_2, q_3, q_4, q_{acc}, q_{rej}\}$, where q_1 and q_2 are used to move the second head towards the \$ and first head remains stationary at once, q_3 and q_4 are used to move both heads towards left side after checking the middle symbol a. On reading the left-hand marker by both heads at the same time, the state q_4 is changed to q_{acc}.

– $\Sigma = \{a, b\}, q_0$ is an intial state, $Q_{acc} = \{q_{acc}\}$ and $Q_{rej} = \{q_{rej}\}$.

– In transition functions $r, s \in \{a, b\}^*$, and its specifications are defined as:

Table 3.3: Details of the transition function and head function for L_3

$V_{\#,\#}\|q_0\rangle = \|q_0\rangle$	$V_{r,r}\|q_0\rangle = \|q_1\rangle$	$V_{r,s}\|q_1\rangle = \|q_2\rangle$
$V_{a,\$}\|q_1\rangle = \|q_3\rangle$	$V_{r,s}\|q_3\rangle = \|q_4\rangle$	$V_{r,s}\|q_4\rangle = \|q_3\rangle$
$V_{b,\$}\|q_1\rangle = \|q_{rej}\rangle$	$V_{r,s}\|q_2\rangle = \|q_1\rangle$	$V_{\#,\#}\|q_4\rangle = \|q_{acc}\rangle$
Head Functions:		
$D(q_0) = (\uparrow, \uparrow), D(q_1) = (\uparrow, \rightarrow), D(q_2) = (\rightarrow, \rightarrow), D(q_3) = (\uparrow, \leftarrow),$ $D(q_4) = (\leftarrow, \leftarrow), D(q_{rej}) = (\uparrow, \uparrow), D(q_{acc}) = (\uparrow, \uparrow)$		

It is easy to verify that L_3 can be recognized by $M_{2-2MQFA}$. First phase checks whether the middle symbol is a or not. Finally, the second phase confirms the length $|x| = |y|$. □

Theorem 3.3.5. *Language $L_4 = \{www \mid w \in \{a, b\}^*\}$ is recognizable by 2-2MQFA.*

Proof. The $M_{2-2MQFA}$ to accept this language is identical to the one accepting L_2. The idea of this proof is as follows. It consists of four phases. First, we keep the first head stationary two times and other moves forward. This process remains to continue until the second head does not read right-end marker \$. If the input string $x \in L_4$, then the first head is at beginning of second w in the input string and the second head points towards the right-end marker \$. In the second phase, the symbols read by first head are getting matched with the second head and moves towards left-end marker #. If any of the symbol read by both heads does not get matched, then it leads to rejecting state q_{rej}.

Otherwise, it confirms that the contents of first and third w are same. In third phase, we keep the second head stationary two times, and other head moves forward towards \$. Finally, we repeat the second phase. The contents of all w's get matched with corresponding one respectively. If any of the symbols do not matched between both heads, then the input sting is said to be rejected. In the end, if the first read points the left-end marker #, which shows that all the contents get matched and the string is accepted. A 2-2MQFA for L_4 is defined as follows:

$$M_{2-2MQFA} = (Q, \Sigma, q_0, Q_{acc}, Q_{rej}, \delta),$$

where

– $Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_{acc}, q_{rej}\}$, where q_1, q_2, q_3, and q_8 are used to keep the first head stationary two times and second head to move towards the \$, states q_4 and q_7 are used to match the symbols between both heads, q_5, q_6 used in the third phase to move the first head towards the right side and second head remains stationary at once, q_4 and q_7 are changed to q_{rej} on reading the different symbols by both heads.

– $\Sigma = \{a, b\}, q_0$ is an initial state, $Q_{acc} = \{q_{acc}\}$ and $Q_{rej} = \{q_{rej}\}$.

– The specifications of transition function and head function are as follows, where $r, s \in \{a, b\}^*$.

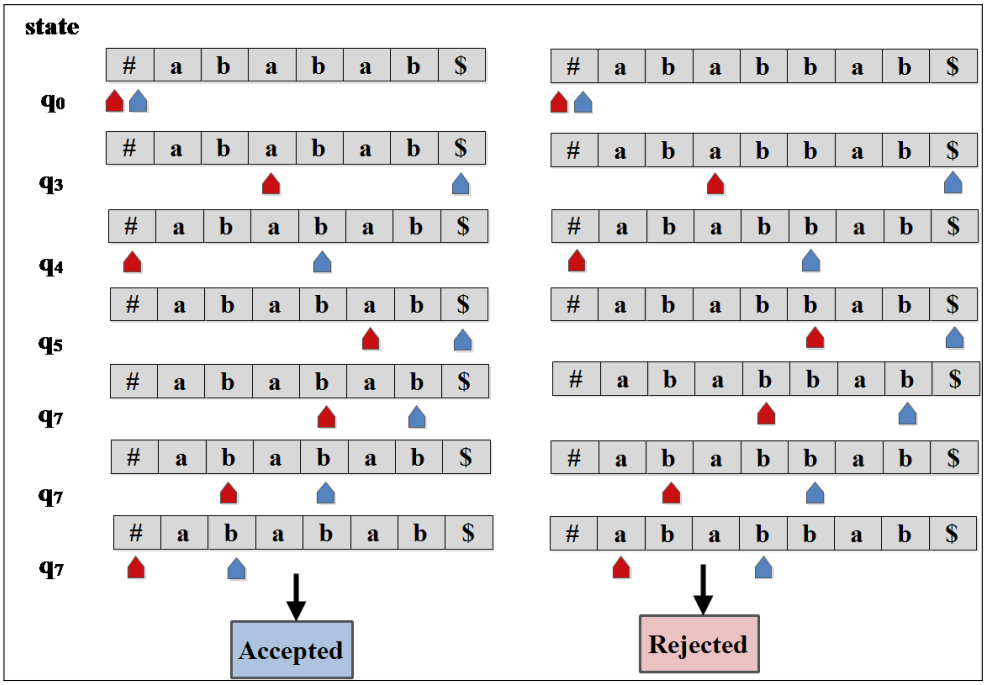


Figure 3.4: Computation process of language L_4 for inputs $ababab \in L_4$ (left) and $ababbab \notin L_4$ (right). In the second case, $M_{2-2MQFA}$ halts with the two heads leaving at the positions when the symbols do not match.

Table 3.4: Details of the transition function and head function for L_4

$V_{\#,\#}\|q_0\rangle = \|q_0\rangle$	$V_{\#,r}\|q_0\rangle = \|q_1\rangle$	$V_{r,s}\|q_1\rangle = \|q_2\rangle$	$V_{r,s}\|q_2\rangle = \|q_3\rangle$
$V_{r,s}\|q_3\rangle = \|q_1\rangle$	$V_{r,\$}\|q_3\rangle = \|q_8\rangle$	$V_{r,\$}\|q_8\rangle = \|q_4\rangle$	$V_{a,a}\|q_4\rangle = \|q_4\rangle$
$V_{a,b}\|q_4\rangle = \|q_{rej}\rangle$	$V_{b,b}\|q_4\rangle = \|q_4\rangle$	$V_{\#,r}\|q_4\rangle = \|q_5\rangle$	$V_{r,s}\|q_5\rangle = \|q_6\rangle$
$V_{r,s}\|q_6\rangle = \|q_5\rangle$	$V_{b,a}\|q_4\rangle = \|q_{rej}\rangle$	$V_{a,a}\|q_7\rangle = \|q_7\rangle$	$V_{b,b}\|q_7\rangle = \|q_7\rangle$
$V_{a,b}\|q_7\rangle = \|q_{rej}\rangle$	$V_{r,\$}\|q_5\rangle = \|q_7\rangle$	$V_{\#,r}\|q_7\rangle = \|q_{acc}\rangle$	$V_{r,\$}\|q_1\rangle = \|q_{rej}\rangle$
$V_{r,\$}\|q_2\rangle = \|q_{rej}\rangle$	$V_{b,a}\|q_7\rangle = \|q_{rej}\rangle$		
Head Functions:			
$D(q_0) = (\uparrow, \rightarrow), D(q_1) = (\rightarrow, \rightarrow), D(q_2) = (\uparrow, \rightarrow), D(q_3) = (\uparrow, \rightarrow),$ $D(q_4) = (\leftarrow, \leftarrow), D(q_5) = (\rightarrow, \rightarrow), D(q_6) = (\rightarrow, \uparrow), D(q_7) = (\leftarrow, \leftarrow)$ $D(q_8) = (\rightarrow, \uparrow), D(q_{rej}) = (\uparrow, \uparrow), D(q_{acc}) = (\uparrow, \uparrow)$			

It can easily verify that each $V_{(\sigma_1,\sigma_2)}$ is unitary and obeying unitary time evolution defined by local interactions between its symbols. Fig. 3.4 shows the computation process of $M_{2-2MQFA}$ for L_4. □

Chapter 4

Quantum Queue Automata

In this chapter, we have proposed a quantum analogue of classical queue automata by using the definition of the quantum Turing machine and quantum finite-state automata. However, quantum automata equipped with a storage medium of a stack have been considered, but the concept of quantum queue automata has not been introduced so far. The classical Turing machines can be simulated by classical queue automata. Motivated by the efficiency of the quantum Turing machine and nature of classical queue automata, we have introduced the notion of quantum queue automata using unitary criteria. Our contributions are as follows. We have also introduced a generalization of real-time deterministic queue automata, the real-time quantum queue automata which work in real-time, i.e., the input head can move towards the right direction only and takes exactly one step per input symbol. We have shown that real-time quantum queue automaton is more superior than its real-time classical variants by using quantum transitions. We have proved the existence of the language that can be recognized by real-time quantum queue automata and cannot be recognized by real-time deterministic (reversible) queue automata. Further, we have shown that there is a language that can be recognized by real-time quantum queue automata but not by real-time non-deterministic queue automata.

4.1 Introduction and motivation

Soon after the design of Shor's algorithm, interest in quantum computation and information has been increased significantly. Several different models, their language recognition capability, and properties have been introduced [7]. Till now, quantum versions for various classical automata's have been introduced such as quantum Turing machine (QTM) [37] of Turing machine; quantum finite automata (Moore and Crutchfield [13]; Kondacs and Watrous [14]) of deterministic finite automata; quantum pushdown automata (Moore and Crutchfield [13]; Golovkins [87]; Gudder [88]; Qiu [89]) of pushdown automata, quantum real-time one-counter automata (rtQ1CA) Say et al. [90] of classical real-time one-counter automata (rt1CA) and many more since last two decades. Some of these constructions are more powerful than their classical counterparts [43].

In 1936, the same fruitful year Turing introduced the Turing machine, Post [91–93] proposed the concept of Post machine (PM), which is equivalent to Turing machine model in computation. Later on, Manna [94] followed the approach of Post and defined an automaton equipped with data structure queue called Post machine.

Till now, it has been extensively studied from several perspectives. In 1980, Brandenburg [95] considered a queue automaton (called post machine) and investigated its features. It has also characterized the class of languages recognized by multi-reset machines and equality sets. The computation power of queue automata by putting a restriction on time has been investigated on several occasions. Cherubini et al. [96] considered the queue automata, which work in quasi real-time, i.e., limit the number of ϵ-transitions by a constant and proved that the emptiness problem is undecidable. In 2013, Jakobi et al. [97] introduced queue automata of constant length and studied its descriptional complexity. Recently, Kutrib et al. [98,99] introduced the concept of reversible queue automata. It has been shown that the class of languages recognized by reversible queue automata strictly consists of regular languages. It has been investigated that the computational power of reversible queue automata and Turing machines is an equivalent. Further, the working of reversible queue automata in quasi real-time has been shown. Moreover, the language recognition power of reversible queue automata is compared with reversible pushdown automata, input-driven queue automata and examined their closure properties.

Scegulnaja [100] introduced the concept of real-time quantum Turing machine and proved that a language $L = \{wcxcw^Rcx^R \mid w, x \in \{0,1\}^*\}$ can be recognized by real-time quantum Turing machine with single work-tape, but cannot be recognized by real-time deterministic Turing machine with single work-tape. It is known that each deterministic queue automaton (DQA) can be simulated by reversible queue automata (RDQA) without any time constraint. Thus, RDQA are powerful as Turing machine. Although, we find that in a quantum queue automation (QQA) where transition matrices consist 0 and 1, i.e., basically a reversible queue automaton (RDQA). So, QQA can recognize all the languages recognized by RDQA. Due to the computational universality of reversible queue automata, i.e., quantum queue automata are of natural interest to impose time restrictions to the computations. Given this result, it is interesting to consider restricted variants of the proposed quantum variant. In fact, one needs to take care the reversibility. In this chapter, we have introduced a notion of quantum queue automata. Further, we have shown that a language which cannot be recognized by any real-time deterministic queue automata can be recognized by quantum queue automata in real-time. Moreover, we have proved that real-time quantum queue automata are more powerful than real-time non-deterministic queue automata in language recognition.

4.1.1 Comparison with 2MQFA

In classical automata theory, queue automata can be simulated by Turing machine. Similarly, quantum queue automata can be simulated by quantum standard circuit model and quantum Turing machine. The languages $L_1 = \{w^n \mid n > 1, w \in \{a\}\}$, where $|w|$ is a prime number, $L_2 = \{ww \mid w \in \{a,b\}^*\}$ and $L_3 = \{www \mid w \in \{a,b\}^*\}$ can be recognized by two-way quantum multihead finite automata, which can also be recognized by quantum queue automata. However, it is still an open problem to find more languages like these that witness the superiority of quantum queue automata over two-way multihead quantum finite automata.

It has been proved that the languages $L_3 = \{ba^{n_1}ba^{n_2}ba^{n_3}...b\ a^{n_i}ca^{n_i}b \mid n_j \geq 0, 1 \leq j \leq i\}$ and $L_{xy} = \{xycyx \mid x \in \{a,b\}^*, y \in \{0,1\}^*\}$ can be recognized by quantum queue automata in real-time, which can also be recognized by two-

way quantum multihead finite automata with at least three heads. Although, the quantum queue automaton takes exactly time equal to the length of an input string. Under real-time conditions, the quantum queue automata can be less powerful than two-way multihead quantum finite automata.

4.1.2 Motivation

- It is known that deterministic finite automata equipped with queue can perform universal computations. It has been proved several times that the computational power of deterministic queue automata and the Turing machine is equivalent.

- QTM is more efficient than the classical Turing machine from a computational complexity point of view. For example, integer factorization and discrete logarithm problems are intractable on classical Turing machine, but they are tractable on QTM [101]. Moreover, real-time QTM is more powerful than real-time DTM [100].

- Motivated by the efficiency of QTM and classical queue automata, we investigate a quantum version of classical queue automata and its superiority over classical counterparts in real-time.

4.2 Preliminaries and definitions

In this section, we review some formal definitions. Throughout the chapter, we have used the following notations: the prefix "R", "D", "ND", "Q" and "rt" signify reversible, deterministic, non-deterministic, quantum and real-time, respectively. An input alphabet Σ does not contains left and right-end markers (#, \$), empty queue symbol $\perp$ is not an element of queue alphabet Σ_q. The length of an input string x is denoted by $|x|$.

Definition 4.2.1. *A deterministic queue automaton (DQA) is defined as a septuple* $(Q, \Sigma, \Gamma, \delta, q_0, \perp, F)$*, where*

- Q *is a set of states,*

- Σ *is an input alphabet,*

- Γ *is a finite set of queue symbols,*

- q_0 *is a starting state,*

- $\perp$ *is an empty queue symbol* $\perp \neq \Gamma$*,*

- *The transition function* δ *is defined by* $Q \times \Sigma \cup \{\lambda\} \times (\Gamma \times \Gamma) \cup (\{\perp\} \times \{\perp\}) \rightarrow Q \times \Gamma \cup \{\lambda\} \times \{keep, remove\}$*, where* λ *signifies empty symbol. It must never be used as an input symbol.*

- F *is a set of accepting states* ($F \subseteq Q$)*.*

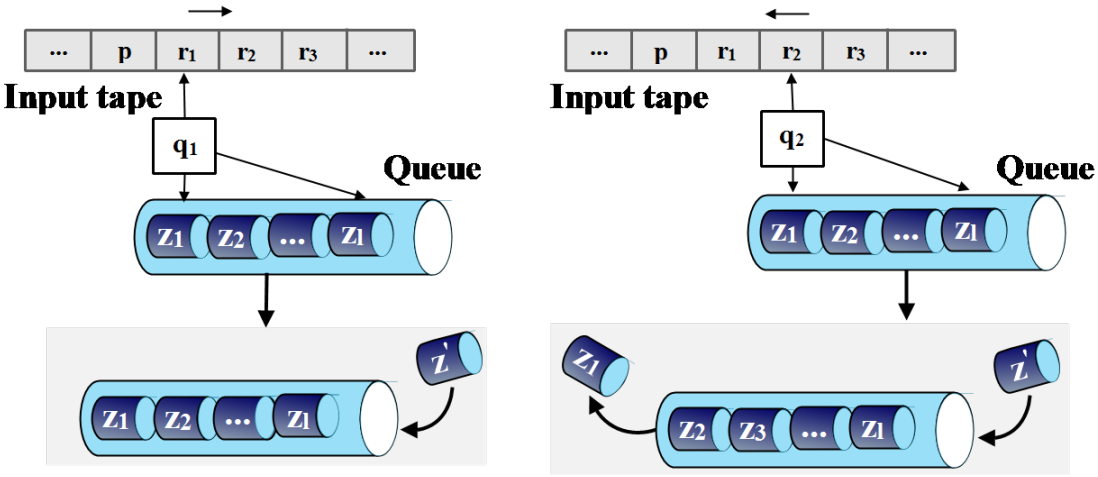


Figure 4.1: Resultant configurations of a DQA

In DQA, it is possible to enqueue the symbol at rear of the queue and dequeue (remove) the symbol at the front of the queue. To process the input string x by $M_{DQA} = (Q, \Sigma, \Gamma, \delta, q_0, \perp, F)$, we assume that x is written on the input tape employed with a queue. A computation process of M_{DQA} is a sequence of configurations $c_0, c_1, c_2, \ldots$, where c_0 is an initial configuration. The configuration of M_{DQA} is defined as a quadruple (p, q, r, s), where $p \in \Sigma^*$ denotes the already read part of input string, $q \in Q$ is the present state, $r \in \Sigma^*$ is unread part of x and s indicates the content of queue, where leftmost symbol is at the front of queue. Suppose the configuration of M_{DQA} is $c_1 = (p, q_1, r_1 r_2, z_1 z_2 \ldots z_l)$, where M_{DQA} is in state q_1 and R/W head is under the symbol r_1 and z_1, z_l are the symbols at the front and rear of the queue respectively, where $r_1 \in \Sigma \cup \{\lambda\}, p, r_2 \in \Sigma^*, z^{'} \in \Gamma \cup \{\lambda\}$ and $z_1, z_2, \ldots z_l \in \Gamma$ for $l \geq 1$. Thus, after reading the symbol r_1, the resultant configuration c_2 is as $c_2 = (pr_1, q_2, r_2, z_1 z_2 \ldots z_l z^{'})$. Thus, the above configuration c_1 is transformed into c_2 if the transition function is $\delta(q_1, r_1, z_1, z_l) = (q_2, z^{'}, \text{keep})$. Similarly, in order to remove the symbol from the head and put the other symbol at the rear, then the resultant configuration will be $c_2 = (pr_1, q_2, r_2, z_2 \ldots z_l z^{'})$ if transition function is $\delta(q_2, r_3, z_1, z_l) = (q_3, z^{'}, \text{remove})$. In case, the queue is empty at an initial stage; then the resultant configuration is computed by δ with empty queue symbol such as $\delta(q_1, r_1, \perp, \perp) = (q_2, z^{'}, \text{keep})$. Fig 4.1 shows the pictorial representation of above configurations of a DQA.

The definition of reversible DQA (RDQA) is same as above DQA. The only difference lies in the transitions, i.e., any configuration of RDQA must have one configuration, which can be computed by DQA. It has also to be deterministic backward and the symbols are enqueued at the front, and dequeued from the rear of queue. Any automaton is said to be real-time if it completes the computation in real-time i.e the running time of automaton on any input string $x \in \Sigma^*$ is less than equal to $|x|$. But, the notion real-time depends upon the model of automata studied. Kutrib et al. [98] studied the restricted versions of DQA by putting a restriction on time. RDQA is said to be real-time DQA (rtRDQA) if there are no λ-steps in computation. It is said to be quasi real-time if there is a constant number of λ-steps applicable for all computations.

4.3 Quantum queue automata

The concept of quantum queue automata is constructed similarly as quantum pushdown automata (QPDA). QQA employs a data structure *queue* referred to as First-In, First-Out (FIFO), whereas QPDA employs stack which referred to as Last-In, First-Out (LIFO) [87]. QQA consists of an input tape, queue, and finite state control. QQA can be defined as a modification of classical queue automata by adding weighted superposition to the configurations of classical queue automata in such a way that processing of an input string corresponds a unitary transformation. QQA choose a transition by considering the current state and the symbols at the front and end of the queue.

Definition 4.3.1. *A quantum queue automaton M_Q is defined as septuple $(Q, \Sigma, \Sigma_q, q_0, Q_{acc}, Q_{rej}, \delta)$, where*

- *Q is a set of states. Moreover, $Q = Q_{acc} \cup Q_{rej} \cup Q_{non}$, where $Q_{acc}, Q_{rej}, Q_{non}$ represents the set of accepting, rejecting, and non-halting states, respectively.*
- *Σ is an input alphabet,*
- *Σ_q is a queue alphabet such that $\Sigma_q = \Sigma \cup \{\#, \$\}$, where end-markers $\{\#,\$\}$ are not included in Σ. For convenience, we have used Σ_τ for $\Sigma_q \cup \{\tau\}$ such that τ represents an empty word and $\perp$ is the empty queue symbol such that $(\perp \notin \Sigma_q)$,*
- *Transition function δ is defined by $\delta : Q \times \Sigma \times \Sigma_q \times \Sigma_q \times Q \times \Sigma_\tau \times D \times X \to C$, where $D = \{\leftarrow, \uparrow, \rightarrow\}$ represents the head function for the left, stationary, and right direction of R/W head, $X \in \{\epsilon, \omega\}$ where ϵ, ω represent the dequeue and enqueue operations respectively. In the following conditions, we have used $z_1, z_2, ...z_l \in \Sigma_q$ for $l \geq 1$, where z_1, z_l signifies the first and last symbol of queue respectively. Transition function must satisfy the following conditions:*

  Well-formedness conditions:

  (i) Local probability condition:

$$\sum_{\substack{\forall(q_1,\sigma,z_1,z_l)\in Q\times\Sigma\times\Sigma_q\times\Sigma_q \\ (q,z',d,\omega)\in Q\times\Sigma_\tau\times D\times X}} | \ \delta(q_1,\sigma,z_1,z_l,q',z',d,\omega) \ |^2 = 1 \tag{4.1}$$

  (ii) Orthogonality condition:

$$\sum_{\substack{\forall(q_1,\sigma,z_1,z_l)\neq(q_2,\sigma,z_1,z_l) \ \in \ Q\times\Sigma\times\Sigma_q\times\Sigma_q \\ (q',z',d,\omega)\in \ Q\times\Sigma_\tau\times D\times X}} \overline{\delta(q_1,\sigma,z_1,z_l,q',z',d,\omega)} \ \delta(q_2,\sigma,z_1,z_l,q',z',d,\omega) = 0 \tag{4.2}$$

  (iii) Separability condition I:

$$\begin{aligned} &\sum_{\substack{\forall(q_1,\sigma_1,z_1,z_l),(q_2,\sigma_1,z_1,z_l)\in Q\times\Sigma\times\Sigma_q\times\Sigma_q \\ (q',z',d,\epsilon)\in Q\times\Sigma_\tau\times D\times X, \\ (q',z',d,\omega)\in Q\times\Sigma_\tau\times D\times X}} \overline{\delta(q_1,\sigma_1,z_1,z_l,q',z',d,\epsilon)} \ \delta(q_2,\sigma_1,z_1,z_l,q',z',d,\omega) \\ &+ \sum_{(q',z',d,\epsilon)\in Q\times\Sigma_\tau\times D\times X} \overline{\delta(q_1,\sigma_1,z_1,z_l,q',z',d,\epsilon)} \ \delta(q_2,\sigma_1,z_1,z_l,q',z',d,\epsilon) = 0 \end{aligned} \tag{4.3}$$

–

$$\sum_{\substack{(q',z',d,\epsilon)\in Q\times\Sigma_\tau\times D\times X,\\(q',z',d,\omega)\in Q\times\Sigma_\tau\times D\times X}} \overline{\delta(q_1,\sigma_1,z_1,z_l,q',z',d,\epsilon)}\ \delta(q_2,\sigma_1,z_1,z_l,q',z',d,\omega)=0 \tag{4.4}$$

(iv) Separability condition II:

$$\forall(q_1,\sigma_1,z_1,z_l),(q_2,\sigma_2,z_1,z_l)\in Q\times\Sigma\times\Sigma_q\times\Sigma_q \sum_{(q',z',\omega)\in Q\times\Sigma_\tau\times X} \overline{\delta(q_1,\sigma_1,z_1,z_l,q',z',\rightarrow,\omega)}\ \delta(q_2,\sigma_2,z_1,z_l,q',z',\uparrow,\omega)=0 \tag{4.5}$$

(v) Separability condition III:

–

$$\forall(q_1,\sigma_1,z_1,z_l),(q_2,\sigma_2,z_1,z_l)\in Q\times\Sigma\times\Sigma_q\times\Sigma_q,\forall d_1,d_2\in D:d_1\neq d_2 \sum_{\substack{(q',z',\epsilon)\in Q\times\Sigma_\tau\times X,\\(q',z',\omega)\in Q\times\Sigma_\tau\times X}} \overline{\delta(q_1,\sigma_1,z_1,z_l,q',z',d_1,\epsilon)}\ \delta(q_2,\sigma_2,z_1,z_l,q',z',d_2,\omega)=0 \tag{4.6}$$

–

$$\sum_{\substack{(q',z',\omega)\in Q\times\Sigma_\tau\times X,\\(q',z',\epsilon)\in Q\times\Sigma_\tau\times X}} \overline{\delta(q_1,\sigma_1,z_1,z_l,q',z',d_1,\omega)}\ \delta(q_2,\sigma_2,z_1,z_l,q',z',d_2,\epsilon)=0 \tag{4.7}$$

–

$$\forall(q_1,\sigma_1,z_1,z_l),(q_2,\sigma_2,z_1,z_l)\in Q\times\Sigma\times\Sigma_q\times\Sigma_q \sum_{\substack{(q',z',d,\epsilon)\in Q\times\Sigma_\tau\times D\times X,\\(q',z',d,\omega)\in Q\times\Sigma_\tau\times D\times X}} \overline{\delta(q_1,\sigma_1,z_1,z_l,q',z',d,\epsilon)}\ \delta(q_2,\sigma_2,z_1,z_l,q',z',d,\omega)=0 \tag{4.8}$$

To process the input string by M_Q, we assume that an input string x is written on the input tape enclosed with both end-markers such as $\#x\$$. It processes the input tape employed with a queue which is potentially infinite on the right-side. The automaton is in the state q, R/W head is above the symbol σ. Then, M_Q with the amplitude $\delta(q,\sigma,z_1,z_l,q',z',d,\omega)$, where z_1, z_l are the symbols at the front and end of queue respectively. It moves to state q', $d\in\{\leftarrow,\uparrow,\rightarrow\}$ moves the R/W head one cell towards left, stationary or in the right direction, and puts the symbol z' at the end of the queue. The automaton M_Q for processing an input x corresponds a unitary evolution in the inner-product space $\mathcal{H}_n$. Definition 4.3.1 utilizes the concept of deterministic queue automata [94, 98] and quantum pushdown automata [87, 89].

A computation of QQA M_Q is a sequence of superpositions $c_0, c_1, c_2, ...$, where c_0 is an initial configuration. When the automaton is observed in a superposition state, for any c_i, it has the form $U_\delta|c_i\rangle = \sum_{c\in C_n}\alpha_c|c_i\rangle$, where C defines the set of configurations, and the configuration c_i is measured with the probability α_c [5]. Superposition is valid; if the sum of the absolute squares of their amplitudes is unitary.

Time evolution of quantum systems is given by unitary transformations. Suppose if the system is in $|\psi\rangle$, then at a later time it will be $|\psi'\rangle = \hat{U}|\psi\rangle$, where $\hat{U}$ is unitary time evolution operator. If U is any linear transformation, then it will be unitary transformation if $\overline{U}U = I$ or $U\overline{U} = I$, where $\overline{U}$ is a conjugate transpose

of U. Therefore, evolution operator specifies how QQA M_Q will progress the input string. Each transition function δ induces a linear time evolution operator over the space H_n. Let $\sigma_1, \sigma_2, ..., \sigma_n \in \Sigma$ and $z_1, z_2, ..., z_l \in \Sigma_q$ for $l \geq 1$. For any configuration $c = |\sigma_1 q \sigma_2 \sigma_3 ... \sigma_n, z_1 z_2 ... z_l\rangle$, the evolution of a QQA M_Q is given by the linear operator U_δ such that

$$U_\delta |c\rangle = \sum_{(q',z',d,\omega) \in Q \times \Sigma_\tau \times D \times X} \delta(q, \sigma_2, z_1, z_l, q', z', d, \omega) \, |f(|c\rangle, d, q'), z', \omega\rangle \tag{4.9}$$

where $(q, \sigma, z_1, z_l) \in Q \times \Sigma \times \Sigma_q \times \Sigma_q$ and

$$f(|\sigma_1 q \sigma_2 ... \sigma_n, z_1 z_2 ... z_l z'\rangle, d, q') = \left\{ \begin{array}{l} q' \sigma_2, z_1 z_2 ... z_l z' \text{ if } d = \uparrow \\ q' \sigma_3, z_1 z_2 ... z_l z' \text{ if } d = \rightarrow \\ q^{'} \sigma_1, z_1 z_2 ... z_l z' \text{ if } d = \leftarrow \end{array} \right\} \tag{4.10}$$

When R/W head is stationary in equation (4.10), then the resultant state is reading the same symbol on input tape and enqueue the symbol z' at the end of queue. The resultant state q' reads the next symbol on moving the R/W head towards the right. Further, the resultant state reads the σ_1 on moving towards the left direction and puts the symbol z' at the rear of the queue.

Theorem 4.3.1. *Well-formedness conditions are satisfied iff the time evolution operator U_δ is unitary.*

Proof. For each input x, U_δ is unitary iff the vectors $U_\delta |c\rangle$ for $c \in C_x$ of the QQA evolution matrix are orthonormal. Condition (i), i.e. local probability condition is satisfied with the statement that $\|U_\delta |c\rangle\| = 1$ for each $c \in C_x$ configuration. Correspondingly, the column vectors of the QQA evolution matrix are orthogonal iff condition in Eq. (4.2) i.e. orthogonality condition is satisfied such that $U_\delta |c_1\rangle - U_\delta |c_2\rangle$, where $c_1 = |q_1 \sigma, z_1 z_2 ... z_l\rangle, c_2 = |q_2 \sigma, z_1 z_2 ... z_l\rangle$ for $q_1 \neq q_2$. Separability condition in Eqs (4.3-4.6) are satisfied in equivalent to the above statement. Separability condition (iv) is satisfied such that $U_\delta |c_1\rangle - U_\delta |c_2\rangle$, in which states q_1 and q_2 are reading the different symbols and resultant state is same, where $\{c_1 = |q_1 \sigma_1, z_1 z_2 ... z_l\rangle, c_2 = |q_2 \sigma_2, z_1 z_2 ... z_l\rangle \mid c_1, c_2 \in C_x\}$, where R/W head moves right and remains stationary for c_1, c_2 respectively according to the condition. Thus, the columns of the evolution matrix are orthogonal for each input x iff conditions in Eqs (4.2-4.6) are satisfied. We can say that, if U_δ is a unitary operator, transition function δ satisfies the well-formedness conditions.

It is not an easy task to check all the well-formedness conditions for trivial QQA (i.e. there is no other state that it can exist in). Consider a QQA, whose evolution matrix columns are orthonormal for each configuration, but the evolution is not unitary. $Q = \{q\}, \Sigma = \{a\}, \Sigma_q = \{A_0\}$ and transition function δ is defined as: $\delta(q, \#, A_0, A_0, q, A_0, \rightarrow, \omega), \delta(q, a, A_0, A_0, q, A_0, \rightarrow, \omega) = 1, \delta(q, \$, A_0, A_0, q, \$, \rightarrow, \omega) = 1$. The other values of the transitions $\delta = 0$. Therefore, we have defined the simple notation of QQA similar to 2QFA and QPA, by which well-formed machines can be more easily specified. The method is to decompose the transition function into transforming of states with queue automata operations (enqueue, dequeue) and other head functions. □

Definition 4.3.2. *A QQA is simplified, for each $\sigma \in \Sigma$, if there exists a function $D : Q \rightarrow \{\leftarrow, \uparrow, \rightarrow\}$ on the inner product space $L_2(Q) \rightarrow L_2(Q)$ such that where Q*

is the set of states, $X \in \{\epsilon, \omega\}$. Define transition function as

$$\left\{ \begin{array}{l|l} \varphi(q, \sigma, z_1, z_l, q', z', \epsilon) = \delta(q, \sigma, z_1, z_l, q', z', D(q'), \epsilon) & \textit{if } D(q') = d \\ 0 & \textit{else} \end{array} \right\} \tag{4.11}$$

where state q results in to q' on reading σ, dequeue a symbol z_1 from the head and puts the $z^{'}$ at end of the queue.

Theorem 4.3.2. *A simple QQA satisfies the well-formedness conditions if there exists a transition function $\varphi(q, \sigma, z_1, z_2, q', z', \omega)$ for any $\sigma \in \Sigma$, $D : Q \to \{\leftarrow, \uparrow, \rightarrow\}$ on the inner product space $L_2(Q) \to L_2(Q)$ and $X \in \{\epsilon, \omega\}$ such that*

$$\sum_{(q',z',\omega) \in Q \times \Sigma_\tau \times X}^{\forall (q_1,\sigma,z_1,z_l),(q_2,\sigma,z_1,z_l) \ \in \ Q \times \Sigma \times \Sigma_q \times \Sigma_q} \overline{\varphi(q_1, \sigma, z_1, z_l, q', z', \omega)} \ \varphi(q_2, \sigma, z_1, z_l, q', z', \omega) = \left\{ \begin{array}{l|l} 1 & q_1 = q_2 \\ 0 & q_1 \neq q_2 \end{array} \right\} \tag{4.12}$$

Proof. Firstly re-write the well-formedness conditions:

$$\sum_{(q',z',d,\omega) \in Q \times \Sigma_\tau \times D \times X} \overline{\delta(q_1, \sigma, z_1, z_l, q', z', d, \omega)} \ \delta(q_2, \sigma, z_1, z_l, q', z', d, \omega) =$$

$$\sum_{(q',z',d,\omega) \in Q \times \Sigma_\tau \times D \times X} \overline{\delta(q_1, \sigma, z_1, z_l, q', z', D(q'), \omega)} \ \delta(q_2, \sigma, z_1, z_l, q', z', D(q'), \omega) =$$

$$\sum_{(q',z',\omega) \in Q \times \Sigma_\tau \times X}^{\forall (q_1,\sigma,z_1,z_l),(q_2,\sigma,z_1,z_l) \ \in \ Q \times \Sigma \times \Sigma_q \times \Sigma_q} \overline{\varphi(q_1, \sigma, z_1, z_l, q', z', \omega)} \ \varphi(q_2, \sigma, z_1, z_l, q', z', \omega) = \left\{ \begin{array}{l|l} 1 & q_1 = q_2 \\ 0 & q_1 \neq q_2 \end{array} \right\} \tag{4.13}$$

□

The definition of a real-time quantum queue automaton (rtQQA) is same as QQA. The only difference lies in the movement of R/W head. It is only allowed to move in the right direction Thus, every step rtQQA reads a new symbol. On reading the right-end marker \$, the computation is finished. The input string is said to be recognized by rtQQA if the R/W reads the right-end marker \$ and queue is empty. Thus, rtQQA accepts the language L, if it takes time not more than $|w|$ for every word $w \in L$.

4.3.1 Language recognition

We assume that QQA has to be observed to produce information about its processing. Consider an observable O for finite-dimensional Hilbert space $\mathcal{H}_n$, which is decomposed into subspaces such as E_a, E_r, E_n refers to the subspace of 'accept', 'reject' and 'non-halting', respectively. Each of these subspaces is traversed by configurations such that $c_a = \{|q\sigma_1\sigma_2...\sigma_n, z_1z_2...z_l\rangle \in C \mid q \in Q_{acc}\}, c_r = \{|q\sigma_1\sigma_2...\sigma_n, z_1z_2...z_l\rangle \in C \mid q \in Q_{rej}\}$ and $c_n = \dfrac{C}{(c_a \cup c_r)}$, where $c_n \in C$.

We assume that an input string $x \in \Sigma^*$ is written on the input tape with both end-markers such that #x\$. It is equipped with a queue (i.e. initially empty). The processing of the input string starts with an initial state and R/W points towards the first symbol on input tape. The transition depends upon the symbol under

R/W head and the symbols at the front and end of the queue, respectively. Firstly, the evolution operator is applied to the current state and computes several paths simultaneously (quantum parallelism); however, as a result of measurement, it is possible to get the results of only one computation path. In meanwhile, each path performs the enqueue or dequeue operations on queue and moves the R/W head corresponding to the resultant state. Then the result is observed by an observable O. Suppose if the automaton is in a superposition state $|\phi\rangle = \alpha_1 |x_1\rangle + \alpha_2 |x_2\rangle + ... + \alpha_n |x_n\rangle$, where α_i are amplitudes and $|\alpha_1|^2 + |\alpha_2|^2 + ... + |\alpha_n|^2 = 1$, then the superposition is projected into the above-mentioned subspaces $E_j, j \in \{a, r, n\}$. The result is observed randomly, and each result j is realized with probability $\|\alpha_j\|^2$. The result of each observation will be either 'accept' or 'reject' or 'non-halting'. The processing remains continue until the observation does not undergo 'acceptance' or 'rejectance' state. Therefore, when the R/W head reaches at right end of the input tape and queue is empty, then the string is said to accepted otherwise the string is rejected after processing the input string.

4.4 The power of quantum queue automata

The computational power of automata employed with queue has been widely studied by various researchers. It is known that queue automata and Turing machines have the same computational power i.e capable of performing universal computations. Kutrib et al. [98] shown the several languages recognized by rtRDQA and proved its closure properties. It has been examined that languages $L_1 = \{ba^n ca^n b \mid n \geq 0\}$ and $L_2 = \{ba^n ba^m ca^m b \mid m, n \geq 0\}$ can be recognized by some reversible DQA in real-time. But, the union of $L_3 = L_1 \cup L_2$ i.e $L_3 = \{ba^{n_1} ba^{n_2} ba^{n_3} ... ba^{n_i} ca^{n_i} b \mid n_j \geq 0, 1 \leq j \leq i\}$ cannot be recognized by any rtDQA [98]. Further, it has been investigated that $L_{xy} = \{xycyx \mid x \in \{a, b\}^*, y \in \{0, 1\}^*\}$ cannot be recognized by any non-deterministic queue automata in real-time (rtNDQA) [99]. In this Section, we have investigated the computational power of real-time quantum queue automata. Thus, by Theorems 4.4.1 and 4.4.2, the real-time quantum queue automata have been shown to outperform its classical variants in the regime of language recognition by imposing same restrictions.

Theorem 4.4.1. *A language $L_3 = \{ba^{n_1} ba^{n_2} ba^{n_3} ... ba^{n_i} ca^{n_i} b \mid n_j \geq 0, 1 \leq j \leq i\}$ can be recognized by real-time quantum queue automata, but cannot be recognized by any deterministic queue automata in real-time.*

Proof. Let $M_{rtQQA} = (Q, \Sigma, \Sigma_q, q_0, Q_{acc}, Q_{rej}, \delta)$ be a real-time QQA, $Q = \{q_0, q_1, q_2, q_3, q_4, q_5, Q_{acc}, Q_{rej}\}, \Sigma = \{a, b, c\}, \Sigma_q = \{A, B\}, Q_{acc} = \{q_{acc_1}, q_{acc_2}\}, Q_{rej} = \{q_{rej_1}\}$. The transition function δ is defined in the manner as described in Section 4.3. It must be noted that the head moves always towards the right direction on reading a new symbol at each step. The specification of transition functions is defined as follows:

In Table 4.1, transition functions are applicable in the case where M_{rtQQA} is in state $q \in Q$ and R/W is above the symbol $\sigma \in \Sigma$ and z_1, z_l are the symbols at the front and end of queue respectively are represented as:

$$\varphi(q, \sigma, z_1, z_l) = \sum_{(q', z', \omega) \in Q \times \Sigma_\tau \times X} \varphi(q, \sigma, z_1, z_l, q', z', \omega)(q', z', \omega) \tag{4.14}$$

Table 4.1: List of transition functions for language $L_3 = \{ba^{n_1}ba^{n_2}b...ba^{n_j}ca^{n_j}b \mid n_k \geq 0, 1 \leq k \leq j\}$

$\varphi(q_0, \#, \perp, \perp) = (q_0, \tau, \omega)$ $\varphi(q_0, b, \perp, \perp) = \frac{1}{\sqrt{2}}(q_1, \tau, \omega) + \frac{1}{\sqrt{2}}(q_2, \tau, \omega)$	
$\varphi(q_1, a, \perp, \perp) = (q_1, A, \omega)$	$\varphi(q_2, a, \perp, \perp) = (q_2, \tau, \omega)$
$\varphi(q_1, a, A, A) = (q_1, A, \omega)$	$\varphi(q_2, b, \perp, \perp) = (q_1, \tau, \omega)$
$\varphi(q_1, b, A, A) = (q_r, \tau, \omega)$	$\varphi(q_2, b, \perp, \perp) = (q_2, \tau, \omega)$
$\varphi(q_1, c, A, A) = (q_3, \tau, \omega)$	$\varphi(q_2, c, \perp, \perp) = (q_4, \tau, \omega)$
$\varphi(q_3, a, A, A) = (q_3, \tau, \epsilon)$	$\varphi(q_4, a, \perp, \perp) = (q_4, \tau, \omega)$
$\varphi(q_3, a, \perp, \perp) = (q_{rej_1}, \tau, \epsilon)$	$\varphi(q_4, b, \perp, \perp) = (q_4, \tau, \omega)$
$\varphi(q_3, b, A, A) = (q_{rej_1}, \tau, \epsilon)$	$\varphi(q_3, b, \perp, \perp) = (q_5, \tau, \omega)$
$\varphi(q_5, \$, \perp, \perp) = (q_{acc_1}, \tau, \omega)$	$\varphi(q_4, \$, \perp, \perp) = (q_{acc_2}, \tau, \omega)$

.

M_{rtQQA} starts by splitting into two computational paths. Each path possesses an equal amplitude of $1/\sqrt{2}$. In the first path, state q_1 reads a and empty queue symbol $\perp$, then it enqueues symbol A at the rear of queue and moves the R/W towards the right. This process continues and the state remains same. On reading symbol b, the M_{rtQQA} changes its state to rejecting state q_r. The state q_1 is changed into q_3 on reading a symbol c from an input tape, and symbol A is at the front and end of queue. Further, it starts dequeue each A from the front of queue on reading a from tape. If in case, state q_3 reads a and queue gets empty or reads b symbol and there exists A is at the front and end of queue, then it goes to the rejectance state q_{rej_1}. Otherwise, the state q_3 is changed into q_5 on reading b and empty queue symbol $\perp$.

While in the second path, R/W keeps moving towards right of the input tape and neither enqueue nor dequeue any symbol from the queue. Whenever a state q_2 reads a symbol b, it splits the computation into two paths, where the first path follows the above procedure and another path follows the loop. But, on reading a symbol c and empty queue by tape heads, state q_2 is changed into state q_4 and moves to the right. If the first path finds the difference in the number of a's before the symbol c and after it, then it goes to rejectance state. Otherwise, on reading symbol \$ from the input tape and empty queue symbol, the working states q_5 and q_4 go to the accepting states q_{acc_1} and q_{acc_2}, respectively. In the end, the total amplitude can be written as the product of the amplitudes associated with each subpath. Thus, the number of subpaths depend upon the number of b's occur before the symbol c in the second path. If the input string $w \in L_3$, then both the paths read the right-end marker \$, i.e., the string is said to be accepted with probability at most 1. If $w \notin L_3$, then it is rejected by one of the paths and the input string is said to be rejected with probability at most $1/2$. Thus, the inputs which are not in L_1 are accepted with probability at most $1/2^i$, where i depends upon the number of b's occur before the symbol c. For instance, consider an input string written on input tape as $\#bcb\$$ enclosed with both end markers. On reading the first b, the computation is split into two paths with $\frac{1}{\sqrt{2}}$. In both paths, state q_1 and state q_2 read the next symbol c with an empty queue and changed into state q_3 and q_4 respectively. Finally, both paths go to the accepting states and string is said to be accepted with probability

1. If the input string is taken as w=#$bacb$\$, i.e., $w \notin L_3$, then it is rejected with probability 1/2. □

Theorem 4.4.2. *There exists a language $L_{xy} = \{xycyx \mid x \in \{a,b\}^*, y \in \{0,1\}^*\}$, that can be recognized by real-time quantum queue automata, but cannot be recognized by non-deterministic queue automata in real-time.*

Proof. Let $M_{rtQQA} = (Q, \Sigma, \Sigma_q, q_0, Q_{acc}, Q_{rej}, \delta)$ be a real-time QQA, $Q = \{q_0, q_1, q_2, q_3, q_4, Q_{acc}, Q_{rej}\}, \Sigma = \{a, b, 0, 1\}, \Sigma_q = \{A, B\}, Q_{acc} = \{q_{acc_1}, q_{acc_2}\}, Q_{rej} = \{q_r\}$. Each transition in Table 4.2 is unitary by inspection and the other transitions are impulsive so that the transformations are unitary. For convenience, we have used the symbols $z_1, z_l \in \{A, B\}$ to represent the symbols at the front and rear of the queue. The specification of transition functions is defined as follows:

Table 4.2: List of transition functions for language $L_{xy} = \{xycyx \mid x \in \{a,b\}^*, y \in \{0,1\}^*\}$

$\varphi(q_0, \#, \perp, \perp) = \frac{1}{\sqrt{3}}(q_1, \tau, \omega) + \frac{1}{\sqrt{3}}(q_2, \tau, \omega) + \frac{1}{\sqrt{3}}(q_r, \tau, \omega)$	
$\varphi(q_1, a, \perp, \perp) = (q_1, A, \omega)$	$\varphi(q_2, a, \perp, \perp) = (q_2, \tau, \omega)$
$\varphi(q_1, b, \perp, \perp) = (q_1, B, \omega)$	$\varphi(q_2, b, \perp, \perp) = (q_2, \tau, \omega)$
$\varphi(q_1, a, z_1, z_l) = (q_1, A, \omega)$	$\varphi(q_2, 0, \perp, \perp) = (q_2, A, \omega)$
$\varphi(q_1, b, z_1, z_l) = (q_1, B, \omega)$	$\varphi(q_2, 1, \perp, \perp) = (q_2, B, \omega)$
$\varphi(q_1, 0, z_1, z_l) = (q_1, \tau, \omega)$	$\varphi(q_2, 0, z_1, z_l) = (q_2, A, \omega)$
$\varphi(q_1, 1, z_1, z_l) = (q_1, \tau, \omega)$	$\varphi(q_2, 1, z_1, z_l) = (q_2, B, \omega)$
$\varphi(q_1, c, z_1, z_l) = (q_3, \tau, \omega)$	$\varphi(q_2, c, z_1, z_l) = (q_4, \tau, \omega)$
$\varphi(q_3, 0, z_1, z_l) = (q_3, \tau, \omega)$	$\varphi(q_4, 0, A, z_l) = (q_4, \tau, \epsilon)$
$\varphi(q_3, 1, z_1, z_l) = (q_3, \tau, \omega)$	$\varphi(q_4, 1, B, z_l) = (q_4, \tau, \epsilon)$
$\varphi(q_3, a, A, z_l) = (q_3, \tau, \epsilon)$	$\varphi(q_4, 0, B, z_l) = (q_r, \tau, \epsilon)$
$\varphi(q_3, b, B, z_l) = (q_3, \tau, \epsilon)$	$\varphi(q_4, 1, A, z_l) = (q_r, \tau, \epsilon)$
$\varphi(q_3, a, B, z_l) = (q_r, \tau, \epsilon)$	$\varphi(q_4, a, \perp, \perp) = (q_4, \tau, \epsilon)$
$\varphi(q_3, b, A, z_l) = (q_r, \tau, \epsilon)$	$\varphi(q_4, b, \perp, \perp) = (q_4, \tau, \epsilon)$
$\varphi(q_3, \$, \perp, \perp) = (q_{acc_1}, \tau, \epsilon)$	$\varphi(q_4, \$, \perp, \perp) = (q_{acc_2}, \tau, \epsilon)$

Similarly to Table 4.1, transitions functions are applicable in the case where M_{rtQQA} is in state $q \in Q$ and R/W is above the symbol $\sigma \in \Sigma$ and z_1, z_l are the symbols at the front and end of queue respectively are represented as:

$$\varphi(q, \sigma, z_1, z_l) = \sum_{(q', z', \omega) \in Q \times \Sigma_\tau \times X} \varphi(q, \sigma, z_1, z_l, q', z', \omega)(q', z', \omega) \tag{4.15}$$

For the construction of M_{rtQQA} for language L_{xy}, the computation process is split into three paths: one of which goes to the rejectance state q_r, and other two paths with states q_1 and q_2 compare the symbols representing x and y, respectively. In the first path, state q_1 puts the symbol A or B into the queue on reading reads a or b with an empty queue symbol $\perp$ respectively and keep moving towards the right. On reading $y \in \{0,1\}^*$, the state q_1 neither enqueue nor dequeue any symbol from queue and moves to the right. On reading c, the state q_1 is changed into q_3. The content of the queue remains same on reading y. On reading the symbols belong to x, it compares the symbols and dequeue A for each a and symbol B for each b from

the front of queue.
In the second path, state q_2 enqueues the symbol A or B into the queue on reading reads 0 or 1 with an empty queue symbol $\perp$ respectively and keep moving towards the right. On reading $x \in \{a, b\}^*$, the state q_2 neither enqueue nor dequeue any symbol from queue and moves to the right. On reading c, the state q_2 is changed into q_4. On reading the symbols belong to y, it compares the symbols and dequeue A for each 0 and symbol B for each 1 from head of the queue. The content of the queue remains same on reading $x \in \{a, b\}^*$ at the end.
If any mismatch occurs in a path on reading the symbols from an input tape and queue, then it goes to the rejecting state. Otherwise, on reading the right-end marker \$, both paths go to the acceptance states q_{acc_1} and q_{acc_2}, respectively. Thus, R/W moves always towards the right direction on reading the symbol, and M_{rtQQA} finishes its work on reading the right-end marker \$.
If the input string $w \in L_{xy}$, then both the paths read the right-end marker \$ and move to the accepting states i.e. the string is said to be accepted with probability $\frac{2}{3}$. If the input string $w \notin L_{xy}$, then it is rejected by at least one of the path and w is said to be rejected with probability greater than equal to $\frac{2}{3}$. □

Chapter 5

Quantifying Matrix Product State

In this chapter, we have investigated the relation between quantum finite state machines and matrix product state (MPS) of quantum spin systems. It plays a crucial role in the context of quantum information processing and is considered as an important asset for quantum communication and information purpose. It is an effective way to represent entangled states of quantum many-body systems. In this chapter, the quantum finite-state machines of one-dimensional MPS representations have been designed.

5.1 Introduction

The simulation of quantum many-body systems on a classical computer is a challenging task due to exponentially increase in the dimension of Hilbert space with the size of system. But, the development of quantum simulators allows us to simulate the dynamics of interacting quantum many-body systems. In 1982, Feynman [1] initially proposed the idea of quantum computing after performing a quantum mechanics simulation on a classical computer and observed that n qubits can simulate n spin-1/2 particles on a quantum computer. The most significant property 'entanglement' separates the classical world from the quantum world. It is one of the most central topics in quantum information theory. Quantum entanglement is a purely quantum mechanical correlation between two parts of the quantum system. In quantum many-body systems, it provides a new way to describe the correlation between two particles.

Devaraj et al. [102] proposed formal approaches to process non-preemptive aperiodic and sporadic tasks due to its ability to guarantee correctness and completeness. In recent years, the tensor network theory has become increasingly popular. Tensor network states are a new language, based on entanglement, for quantum many-body systems. It is widely used to simulate strongly entangled correlated systems [103]. The DNA is the fundamental building block of a person. Similarly, the tensor is the fundamental building block of a quantum state. Therefore, the tensor is the DNA of wave function in which properties of the quantum many-body states can be read from the small individual tensors alone. Such a structure is called a tensor network (TN). It includes more degrees of freedom to attach the different tensors. These extra degrees of freedom connecting indices between the tensors are called bond indices, and lines which do not connect one tensor to others are called open indices. Fig 5.1 shows the TN diagrams:

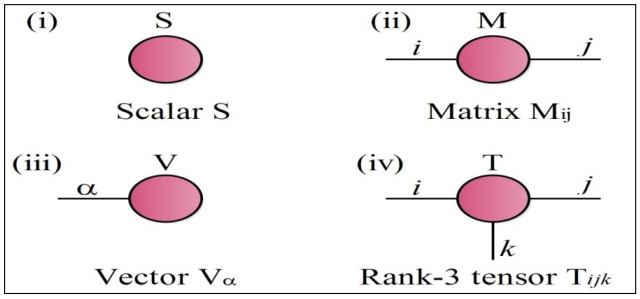


Figure 5.1: Tensor networks (i) Scalar, (ii) Vector, (iii) Matrix, (iv) Rank-3 tensor

Tensor network states can be classified based on dimensions along which the tensors are traversed. In 1993, White [104] introduced the density matrix renormalization group (DMRG), the most common TN method for simulation of one-dimensional strongly correlated quantum systems. It is based on the most well-designed class of TN states called matrix product state (MPS). In the last two decades, DMRG is considered as a method of reference to study the stationary properties of one-dimensional strongly correlated quantum systems. MPS provides an efficient approximation of realistic local Hamiltonians and can be generated by the sequential generation of tensors.

Regardless of success, restrictions remaining in the dimensions, and classes of Hamiltonians that can be simulated with MPS-based methods. Various new algorithms have been proposed based on different types of TN states to overcome such restrictions. The first one was the projected entangled pair states (PEPS), which is a generalization of MPS to two and higher dimensions, tree tensor network (TTN), which has a real space renormalization group structure and has tree-like structures with no loops, and the multiscale entanglement renormalization ansatz (MERA), which removes a short-range entanglement [105].

5.2 Definitions

In this section, we review some formal definitions and related properties that will be used in this chapter.

Definition 5.2.1. [21] *A finite-state machine (FSM) is defined as a quintuple* $(Q,\ \Sigma,\ \delta, q_0,\ F)$, *where*

- Q *is a finite set of states,*
- Σ *is a finite set of alphabets,*
- δ *is a transition function:* $Q \times \Sigma \to Q$,
- $q_0 \in Q$ *is an initial state,*
- $F \subseteq Q$ *is a set of final states.*

Definition 5.2.2. [106, 107] *A stochastic finite-state machine (SFSM) is defined as a triple* $(S,\ X,\ Y, T^{(y)} : y \in Y)$, *where*

- S *is a finite set of states,*

- *X and Y are input and output alphabets, respectively,*
- *$T^{(y)}$ is sub-stochastic matrix such that $T = \sum_{y \in Y} T^{(y)}$, where T is a stochastic matrix and $T^{(y)}$ is a probability of transition from one state to other.*

To process the languages with SFSM, we adopt Dirac's bra–ket notation. It is closed under addition and multiplication by scalars. Using Dirac's notation, the vectors are denoted by kets $|u\rangle$ (row vectors). We can associate with each ket a vector in the dual space called bra $\langle v|$ (column vectors). In SFSM, the probability associated with words $w_1 w_2 w_3 \ldots w_L \in Y^L$ of length L is computed as:

$$\Pr(w_L) = \langle \pi | T^{(w_L)} | \eta \rangle, \tag{5.1}$$

where $\langle \pi|$ is stationary probability density such that $\langle \pi| = (\pi_1,\ \pi_2,\ \pi_3, \ldots, \pi_S)$ satisfies $\pi_1 \geq 0; \sum_i^S \pi_i = 1$; $T^{(w_L)} = \ T^{(w_1)} = T^{(w_1)} T^{(w_2)} T^{(w_3)} \ldots \ T^{(w_L)}$ are transition matrices; and $|\eta\rangle = (1,\ 1,\ 1, \ldots, 1)^T$ is a column vector with $\mid S \mid$ components. We define a QFSM that relates to the standard quantum mechanical explanation of a physical experiment.

Definition 5.2.3. *A quantum finite-state machine (QFSM) is defined as a quintuple $\left(Q,\ \langle \Psi|, X, Y, P^{(y)} : y \in Y, U(y)\right)$,where*

- *Q is a finite set of states,*
- *$\langle \Psi|$ is a state vector belongs to n-dimensional Hilbert space $\mathcal{H}_n$,*
- *X and Y are input and output alphabets, respectively,*
- *$P^{(y)}$ is a mutually orthogonal projection operator such that $\sum_{y \in Y} P^{(y)} = 1$,*
- *U (y) is a transition matrix such that $U(y) = U.P^{(y)}$, where U is the unitary matrix and $P^{(y)}$ is an orthogonal projection operator.*

A quantum deterministic finite-state machine is a quantum finite-state machine in which each matrix U (y) has at most one nonzero entry per row. A QFSM is said to be quantum transducer with $|X| = 1$ [107]. In QFSM, there exists a basis vector v_i for each quantum state $q_i \in Q$, having single nonzero entry of 1 at i^{th} coordinate. The set of basis vectors span the Hilbert space $\mathcal{H}$. If a state $q_i \in Q$ is an incoming transition which outputs symbol y_1, then $P^{(y_1)} v_i = v_i$ [106]. Consider another incoming transition that outputs symbol y_2, and then $P^{(y_2)} v_i = v_i$. Subsequent to $y_1 \neq y_2$, by mutual orthogonal of projections, it satisfies that $P^{(y_1)} P^{(y_2)} = 0$.

To measure the probability of words for QFSMs, we define density operator ρ on a finite set of states that satisfies $\{\Psi_1, \ldots, \Psi_k\}$ as

$$\rho = \sum_i^k \rho_i \left|\Psi_i\right\rangle \left\langle \Psi_i\right|, \tag{5.2}$$

where ρ_i is the probability for the system in the state of Ψ_i, and Ψ_i's are the diagonal basis for ρ. The density operator on Hilbert space must satisfy the trace condition, i.e., $Tr(\rho) = 1 \ | \rho \geq 0$, where $Tr(\ )$ refers to the sum of the diagonal elements

of matrices. Similar to the stationary density operator of SFSMs, the stationary density operator ρ for QFSMs is defined as

$$\rho = \sum_{y \in Y} P^{(y)} U^* \rho \, U P^{(y)}. \tag{5.3}$$

It is invariant under unitary evolution. The stationary density operator of QFSM is $\rho = |Q|^{-1} \cdot 1$ (proved in [106]). In QFSM, the probability of single character $y \in Y$ depends on dimension of projection operators and computed as

$$Pr\,(y) = |Q|^{-1} \cdot \dim\left(P^{(y)}\right). \tag{5.4}$$

The probability associated with words $w_1 w_2 w_3 \ldots w_L \in Y^L$ of length L is computed as

$$\Pr\,(w_L) = Tr\,(U^*(w_L)\rho U(w_L)), \tag{5.5}$$

where $\rho = |Q|^{-1} \cdot 1$ is stationary density operator and $U\,(w_L) = U(w_1)\; U(w_2)\; U(w_3) \ldots U(w_L)$.

Definition 5.2.4. [106] *Any matrix $A^{n \times n}$ is called bistochastic if it contains non-negative real numbers, and every row and column sums to 1. It is called unistochastic if there exists a unitary matrix U such that $A_{ij} = |U_{ij}|^2$.*

Definition 5.2.5. *Von Neumann entropy [108]: It plays a crucial role in measuring quantum information and consideration of entanglement. The Von Neumann entropy is defined as $S(\rho)$=-$Tr(\rho log_2 \rho)$ where ρ is the density operator associated with the state, Tr denotes the trace, "log" base 2 shows that the units are bits and ρ is written in terms of its eigenvectors. Generally, it measures the point of mixture of a system.*

There are some operators used over single qubit in quantum computation. Quantum operators can be represented in a quantum circuit using quantum gates. The controlled NOT operator (CNOT) has two inputs a, b and two outputs [109]. It performs a flip operation over the target qubit b if the first is 1. Other extensively used quantum operators are as identity operator (I): it produces an output qubit as it is (does nothing), Hadamard operator (H): it produces superposition of states with equal probability in the computational basis, flip operator (X): it acts as a classical NOT i.e. flips the qubit and their matrix representations are given in equations (5.6-5.9), respectively.

- CNOT operator

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{5.6}$$

- Identity operator (I):

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \begin{matrix} I\,|0\rangle = |0\rangle \\ I\,|1\rangle = |1\rangle \end{matrix} \tag{5.7}$$

- Hadamard operator (H):

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad \begin{matrix} H\,|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\ H\,|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \end{matrix} \tag{5.8}$$

- Flip operator (X):

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad \begin{matrix} X|0\rangle = |1\rangle \\ X|1\rangle = |0\rangle \end{matrix} \tag{5.9}$$

5.3 Matrix product state

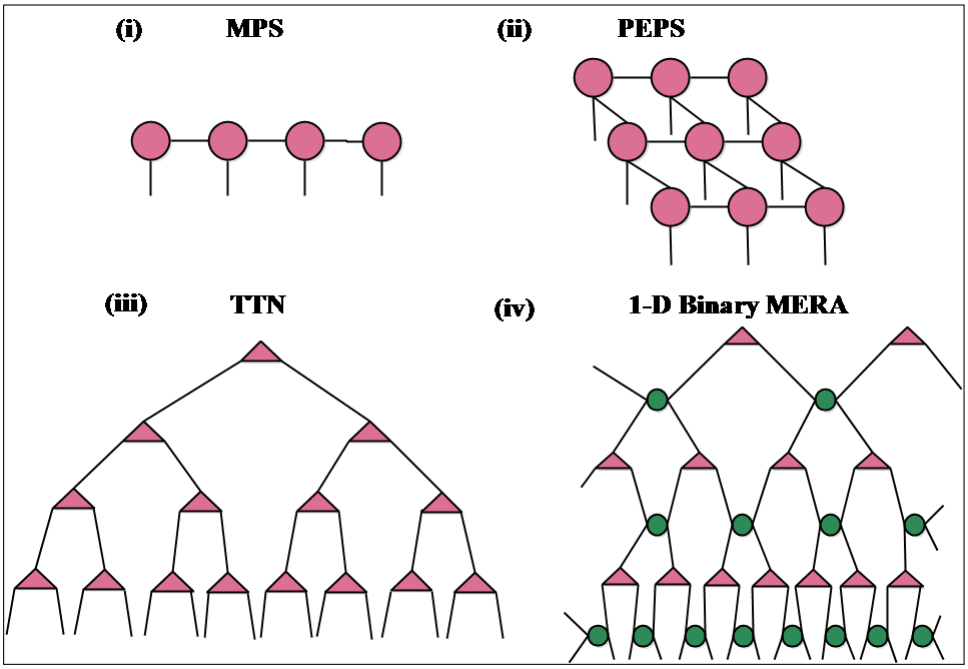


Figure 5.2: Tensor network states (i) MPS, (ii) PEPS, (iii) TTN, (iv) 1-D binary MERA

The family of MPS is the most prominent example of TN states. There are various powerful methods such as density matrix renormalization group (DMRG) algorithm, power wave function renormalization group (PWFRG) and time-evolving block decimation (TEBD) based on MPS to simulate one-dimensional quantum many-body systems [110]. Fig 5.2 (i) shows the MPS as a one-dimensional array of tensors and an example of a finite system of 4 sites. There exists one tensor for each site in quantum many-body systems. MPS shows a certain geometry that is relevant to experimental and quantum information theoretic applications. The family of MPS consists of following non-trivial states:

5.3.1 GHZ state

A Greenberger–Horne–Zeilinger (GHZ) state is an entangled quantum state [103, 111] which has many non-classical properties. The GHZ state of N-spins 1/2 is defined as

$$|GHZ\rangle = \frac{1}{\sqrt{2}} \left(|0\rangle^{\otimes N} + |1\rangle^{\otimes N} \right). \tag{5.10}$$

It can be represented independently by the matrices:

$$A^0 = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \quad A^1 = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}. \tag{5.11}$$

The GHZ state for 3-qubit is $\frac{1}{\sqrt{2}}(|000\rangle + |111\rangle)$. It is highly entangled quantum state of N-spins, which has some non-trivial entanglement properties [112].

5.3.2 AKLT state

In 1987, Affleck–Kennedy–Lieb–Tasaki [113] state is introduced as an extension of quantum Heisenberg spin model. It is one of the most interesting quantum states in correlation physics, which is a ground state of quantum spin chain of spin-1. It is given by the Hamiltonian:

$$H = \sum_i \left(S_i \cdot S_{i+1} \; + \; \frac{1}{3} (S_i \cdot S_{i+1})^2 \right). \tag{5.12}$$

In MPS representation of AKLT state, each spin-1 is replaced by a pair of symmetrized spin-1/2. Consider triplet states represented as spin-1 states

$$|+\rangle = |\uparrow\uparrow\rangle, |0\rangle = \frac{|\uparrow\downarrow\rangle + |\downarrow\uparrow\rangle}{\sqrt{2}}, |-\rangle = |\downarrow\downarrow\rangle, \tag{5.13}$$

whereas adjacent pairs of spin-1/2 are linked in a singlet state such as $\frac{|\uparrow\downarrow\rangle - |\downarrow\uparrow\rangle}{\sqrt{2}}$. The normalized MPS representation of N-chained AKLT state is:

$$|\psi\rangle \; = \sum_{\sigma}^{N} Tr\left[A^{\sigma_1} A^{\sigma_2} \dots A^{\sigma_N}\right] |\sigma_1 \sigma_2 \dots \; \sigma_N\rangle. \tag{5.14}$$

Following are the three matrices categorized with Pauli matrices indices as σ_i's [104].

$$A^+ = \begin{pmatrix} 0 & \sqrt{\frac{2}{3}} \\ 0 & 0 \end{pmatrix}, \quad A^0 = \begin{pmatrix} \frac{-1}{\sqrt{3}} & 0 \\ 0 & \frac{1}{\sqrt{3}} \end{pmatrix}, \quad A^- = \begin{pmatrix} 0 & 0 \\ -\sqrt{\frac{2}{3}} & 0 \end{pmatrix}. \tag{5.15}$$

It can be computed as

$$\begin{aligned} \langle\psi|\psi\rangle = & \sum_{\sigma_i,\, \sigma_i'}^{N} \langle\sigma|\sigma'\rangle \; Tr\left[A^{\sigma_1 '} A^{\sigma_2 '} \cdots A^{\sigma_N '}\right] Tr\left[A^{\sigma_1} A^{\sigma_2} \dots A^{\sigma_N}\right] \\ = & Tr \left(\sum_{\sigma_1} A^{\sigma_1 *} \; \otimes \; A^{\sigma_1} \right) \; \dots \; \left(\sum_{\sigma_1} A^{\sigma_N *} \; \otimes \; A^{\sigma_N} \right) \\ = & Tr \; E^N \end{aligned} \tag{5.16}$$

where $E = \sum_{\sigma} A^{\sigma *} \; \otimes \; A^{\sigma}$.

5.3.3 Cluster state

In 2009, Raussendorf, Browne, and Briegel [114] introduced the concept of cluster state. It is a highly entangled state of multiple qubits [103]. It refers to the unique ground state of the 3-body interactions as $\sum_i \sigma_i^z \sigma_{i+1}^x \sigma_{i+2}^z$. Consider a cluster state

of 2-qubits:

$$\begin{aligned}|\phi_2\rangle =&|+\rangle_1|+\rangle_2 \ = \frac{1}{2}\left(|0\rangle+|1\rangle\right)\left(|0\rangle+|1\rangle\right) \rightarrow \frac{1}{2}\left(|0\rangle|0\rangle+|0\rangle|1\rangle+|1\rangle|0\rangle-|1\rangle|1\rangle\right)\\ =&\frac{1}{2}\left(\left(|0\rangle+|1\rangle\right)|0\rangle\left(|0\rangle-|1\rangle\right)|1\rangle\right)\\ =&\frac{1}{\sqrt{2}}\left(\left(|+\rangle_1|0\rangle_2\right)+\left(|-\rangle_1|1\rangle_2\right)\right).\end{aligned} \tag{5.17}$$

It forms Bell state. Its matrix product state is represented as

$$A^0=|0\rangle|+\rangle=\begin{pmatrix}1 & 1\\ 0 & 0\end{pmatrix}, \quad A^1=|1\rangle|-\rangle=\begin{pmatrix}0 & 0\\ 1 & -1\end{pmatrix}. \tag{5.18}$$

5.3.4 W state

$$|W\rangle=\frac{1}{\sqrt{n}}\left(|100\ldots0\rangle+|010\ldots0\rangle+\cdots+\ |000\ldots1\rangle\right). \tag{5.19}$$

The W state is an entangled quantum state represents and refers to the quantum superposition of pure states with an equal coefficients [112]. It represents a specific type of multipartite entanglement in which exactly one of the qubits is in excited state $|1\rangle$, while all others are in ground state $|0\rangle$. The W state for 3-qubit is $\frac{1}{\sqrt{3}}\left(|100\rangle\ +|010\rangle\ +\ |001\rangle\right)$. It is represented by the matrices [115]:

$$\begin{aligned}A(1)^0&=\begin{pmatrix}0&0&0\\0&1&0\\0&0&1\end{pmatrix}, \quad A(2)^0=\begin{pmatrix}1&0&0\\0&0&0\\0&0&1\end{pmatrix}, \quad A(3)^0=\begin{pmatrix}1&0&0\\0&1&0\\0&0&0\end{pmatrix}\\ A(1)^1&=\begin{pmatrix}1&0&0\\0&0&0\\0&0&0\end{pmatrix}, \quad A(2)^1=\begin{pmatrix}0&0&0\\0&1&0\\0&0&0\end{pmatrix}, \quad A(3)^1=\begin{pmatrix}0&0&0\\0&0&0\\0&0&1\end{pmatrix}.\end{aligned} \tag{5.20}$$

Both $|GHZ\rangle$ and $|W\rangle$ represent two different kinds of tripartite entanglement. $|W\rangle$ is less entangled than the $|GHZ\rangle$ because it leaves bipartite entanglement on measure one of its sub-systems. But on the measurement of $|GHZ\rangle$, its collapse into a mixture or a pure state.

5.4 Constructing QFSM

In this section, we have designed QFSM of various MPS models and proved that it is equivalent to MPS representation of the ground state of quantum spin systems. Every QFSM indeed generates a SFSM with the same word probabilities. Let $M_Q = \left(Q,\ \langle\Psi|\ , X,\ Y,\ P^{(y)}:\ y\in Y,\ U(y)\right)$ be a QFSM and $M'_S=\left(S,\ X,\ Y,\ T^{(y)'}:\ y\in y\right)$ is a SFSM, where $|Q|=|S|$ and $T_{ij}^{(y)'}=|U(y)_{ij}|^2$ (proved in [106]) from the definition of equivalent SFSM. Therefore, M'_S produces the same probabilities for every word as M_Q.

Consider a one-dimensional matrix product state; our process is to construct a quantum finite-state machine as follows: We have an equivalent matrix product state models with unitary matrix or with unistochastic transition matrix T. Then, we construct a quantum finite-state machine with the same number of states. It has been proved that every deterministic quantum generator has an equivalent deterministic classical generator that produces the same stochastic process, if there are unitary evolution and projective measurement for quantum process [107, 116]. Amanda constructed the quantized versions of several well-known stochastic finite-state machines [106]. There are two ways to construct a quantum finite-state machine from a classical machine. We have unistochastic matrix for which there exists a unitary matrix such that $T_{ij} = |U_{ij}|^2$, and other using quantum analogy $U(y) = U \cdot P^{(y)}$ for the symbols transition matrix and projection operators. Following are the construction of QFSM of various MPS models:

5.4.1 GHZ state

In the construction of QFSM, we need to find a unitary matrix U for which $T_{ij} = |U_{ij}|^2$. From the definition of QFSM, recall the $U(y) = U \cdot P^{(y)}$, which is a quantum analogy of the symbol-labelled transition matrices. We consider the following quantum finite-state machine for the maximum entangled GHZ state of 3-qubit:

$$\begin{aligned} Q = \{A,\ B\}, \quad Y = \{0,\ 1\}, \quad U = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}, \\ P^{(0)} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, P^{(1)} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}. \end{aligned} \tag{5.21}$$

The transition matrices are:

$$U(0) = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad U(1) = \frac{1}{\sqrt{2}} \begin{pmatrix} -1 \\ 1 \end{pmatrix}. \tag{5.22}$$

Correspondingly, the stochastic finite-state machine of GHZ state is defined as:

$$S = \{A, B\}, \quad Y = \{0,\ 1\}, \quad T^{(0)} = \frac{1}{2} \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}, \quad T^{(1)} = \frac{1}{2} \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}. \tag{5.23}$$

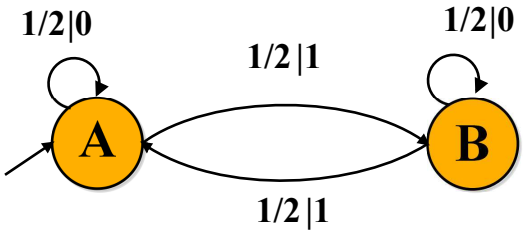


Figure 5.3: Stochastic finite-state machine of GHZ state

Table 5.1: Probability of words for a stochastic machine of GHZ state

Word w	Probability $Pr(w)$

0	1/2
1	1/2
00	1/4
01	1/4
10	1/4
11	1/4
000	1/8
001	1/8
010	1/8
011	1/8
100	1/8
101	1/8
110	1/8
111	1/8

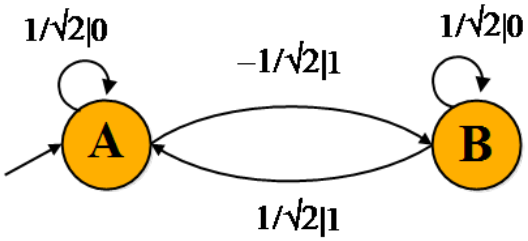


Figure 5.4: Quantum finite-state machine of GHZ state

Table 5.2: Probability of words for a quantum machine of GHZ state

Word w	**Probability $Pr(w)$**
0	1/2
1	1/2
00	1/4
01	1/4
10	1/4
11	1/4
000	1/8
001	1/8
010	1/8
011	1/8
100	1/8
101	1/8
110	1/8
111	1/8

It can be easily checked that GHZ state has a unistochastic matrix $T = T^{(0)} + T^{(1)}$. $T = T^{(0)} + T^{(1)}$. The matrix T satisfies $T_{ij} = |U_{ij} = |U_{ij}|^2$, where U is unitary matrix. Fig 5.3. shows the SFSM of GHZ state. The probability of words for the GHZ state with words up to length $L = 3$ can be found in Table 5.1. Thus, the

quantized version of the GHZ state is shown in Fig 5.4. It can be easily verified that the probabilities of words produced by the above QFSM and SFSM for GHZ state are same. Therefore, QFSM and SFSM for GHZ state are equivalent (Table 5.2).

5.4.2 AKLT state

The two-state stochastic finite-state machine of AKLT is given as

$$\begin{aligned} S = \{A,\ B\}, \quad Y = \{0,\ 1\}, \quad T^{(+)} = \begin{pmatrix} 0 & \sqrt{\frac{2}{3}} \\ 0 & 0 \end{pmatrix}, \\ T^{(0)} = \begin{pmatrix} -1/\sqrt{3} & 0 \\ 0 & 1/\sqrt{3} \end{pmatrix}, \quad T^{(-)} = \begin{pmatrix} 0 & 0 \\ -\sqrt{\frac{2}{3}} & 0 \end{pmatrix}. \end{aligned} \tag{5.24}$$

Above, transition matrices are not stochastic. We normalized the matrices to form stochastic matrices such that $T^{(y)} = \langle T^{(y)} | T^{(y)} \rangle$. Thus, the above three matrices of AKLT state are encoded into two matrices to reduce calculations. Correspondingly, the ratio of rank $\left(P^{(0)}\right)$: rank $\left(P^{(1)}\right)$ is $1:2$. Therefore, $|Q| \geq 3$ and it will have minimal three states. Firstly, we wish to design a SFSM for AKLT state that produces the same probabilities of words as a quantum version of the machine and transition matrix is unistochastic. In Fig. 5.5, we have designed a four-state stochastic machine for AKLT state with unistochastic matrix.

$$T^{(0)} = \begin{pmatrix} \frac{1}{3} & \frac{2}{3} \\ 0 & \frac{1}{3} \end{pmatrix}, \ T^{(1)} = \begin{pmatrix} 0 & 0 \\ \frac{2}{3} & 0 \end{pmatrix}, \quad T = \begin{pmatrix} \frac{1}{3} & \frac{2}{3} \\ \frac{2}{3} & \frac{1}{3} \end{pmatrix}.$$

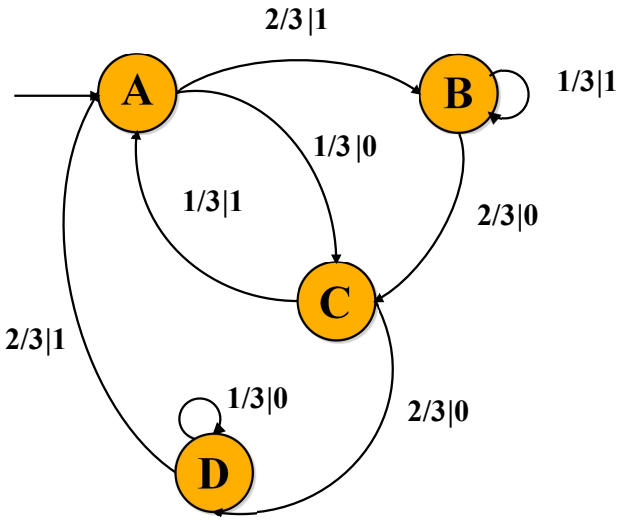


Figure 5.5: Four-state stochastic finite-state machine of AKLT state

We have split the states $A,\ B$ to form four-state SFSM for AKLT state given as

$$S = \{A,\ B,\ C,\ D\}, \quad Y = \{0,\ 1\},$$

$$T^{(1)} = \begin{pmatrix} 0 & \frac{2}{3} & 0 & 0 \\ 0 & \frac{1}{3} & 0 & 0 \\ \frac{1}{3} & 0 & 0 & 0 \\ \frac{2}{3} & 0 & 0 & 0 \end{pmatrix}, \quad T^{(0)} = \begin{pmatrix} 0 & 0 & \frac{1}{3} & 0 \\ 0 & 0 & \frac{2}{3} & 0 \\ 0 & 0 & 0 & \frac{2}{3} \\ 0 & 0 & 0 & \frac{1}{3} \end{pmatrix}$$

It has a unistochastic matrix $T = T^{(0)} + T^{(1)}$.

$$T = \begin{pmatrix} 0 & \frac{2}{3} & \frac{1}{3} & 0 \\ 0 & \frac{1}{3} & \frac{2}{3} & 0 \\ \frac{1}{3} & 0 & 0 & \frac{2}{3} \\ \frac{2}{3} & 0 & 0 & \frac{1}{3} \end{pmatrix}. \tag{5.25}$$

Correspondingly, matrix T (5.25) satisfies $T_{ij} = |U_{ij}|^2$, where U is unitary matrix. Thus, the quantized version of AKLT state is given by

$$Q = \{A,\ B,\ C,\ D\}, \quad Y = \{0,\ 1\}, \quad U = \begin{pmatrix} 0 & -\sqrt{\frac{2}{3}} & -\frac{1}{\sqrt{3}} & 0 \\ 0 & \frac{1}{\sqrt{3}} & -\sqrt{\frac{2}{3}} & 0 \\ -\frac{1}{\sqrt{3}} & 0 & 0 & -\sqrt{\frac{2}{3}} \\ -\sqrt{\frac{2}{3}} & 0 & 0 & \frac{1}{\sqrt{3}} \end{pmatrix}$$

$$P^{(0)} = |e_1\rangle\langle e_1| + |e_2\rangle\langle e_2|\langle e_3| + |e_4\rangle\langle e_4|. \tag{5.26}$$

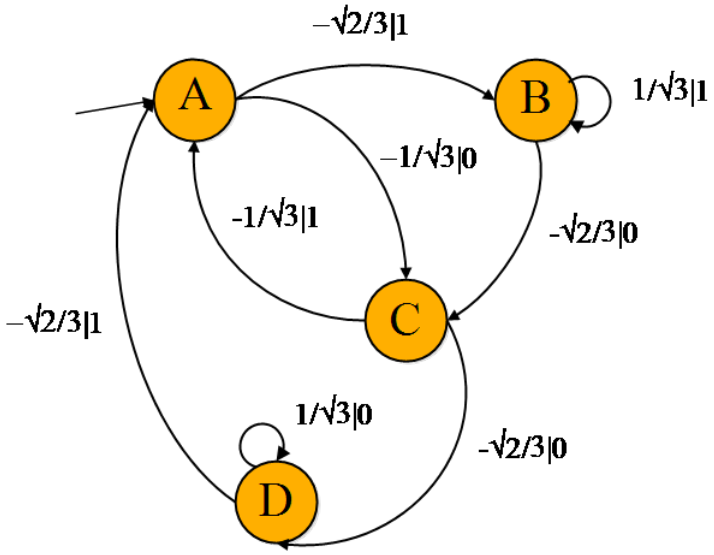


Figure 5.6: Four-state quantum finite-state machine of AKLT state

The probability of single letter $\Pr(0) = \Pr(1) = 1$, i.e., the ratio of rank $\left(P^{(0)}\right)$: rank $\left(P^{(1)}\right)$ is 1:1. The probability of QFSM for AKLT state with words of length $L = 2,\ 3$ is computed in Table 5.3.

Table 5.3: Probability of words for QFSM of AKLT state

Word w	**Probability** $Pr(w)$
00	1/4
01	1/4
10	1/4
11	1/4
000	1/12

001	1/6
010	1/12
011	1/6
100	1/6
101	1/12
110	1/6
111	1/12

5.4.3 Cluster state

Recently, cluster states have found widespread interest in quantum information theory. The reason behind this is measurement-based quantum computation, which is different from the circuit model in quantum computation. Generally, cluster states are also graph states. We have designed a 2-state quantum finite-state machine of cluster state of 3-qubits:

$$\begin{aligned} |\phi\rangle_{c3} =& |+\rangle_1|+\rangle_2|+\rangle_3 \ = \frac{1}{\sqrt{2}}\left((|+\rangle_1|0\rangle_2) + (|-\rangle_1|1\rangle_2)\right)(|0\rangle_3 + |1\rangle_3) \\ =& \frac{1}{\sqrt{2}}\left(|+\rangle_1|0\rangle_2|+\rangle_3) + (|-\rangle_1|1\rangle_2|-\rangle_3\right) = |GHZ_3\rangle. \end{aligned} \tag{5.27}$$

It shows that the cluster state $|\phi\rangle_{c3}$ of 3-qubits is equivalent to GHZ state. Therefore, its QFSM is similar to GHZ state of 3-qubits, which is shown in Fig. 5.4, and the probability of words is given in Table 5.2.

5.4.4 W state

We constructed the 4-state quantum finite-state machine of 4-qubit W state by using the projection matrices Eq (5.20) such that

$$Q = \{A,\ B,\ C,\ D\}, \quad Y = \{0,\ 1\}, \quad U = \frac{1}{2}\begin{pmatrix} 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 \\ -1 & -1 & 1 & 1 \\ -1 & 1 & -1 & 1 \end{pmatrix}$$

It is easier to construct models with bistochastic matrices than to construct it with a unistochastic matrix. But, it is not sure that if we can find a stochastic model with a bistochastic matrix that will be unistochastic matrix. The four basis states named $(\varphi_A,\ \varphi_B,\ \varphi_C,\ \varphi_D)$ spanning the Hilbert space $\mathcal{H}$ for W state are

$$|\varphi_A\rangle = |1000\rangle, \quad |\varphi_B\rangle = |0100\rangle, \quad |\varphi_C\rangle = |0010\rangle, \quad |\varphi_D\rangle = |0001\rangle.$$

$$P(1)^0 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad P(2)^0 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

$$P(3)^0 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad P(4)^0 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$

$$P(1)^1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad P(2)^1 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$

$$P(3)^1 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad P(4)^1 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

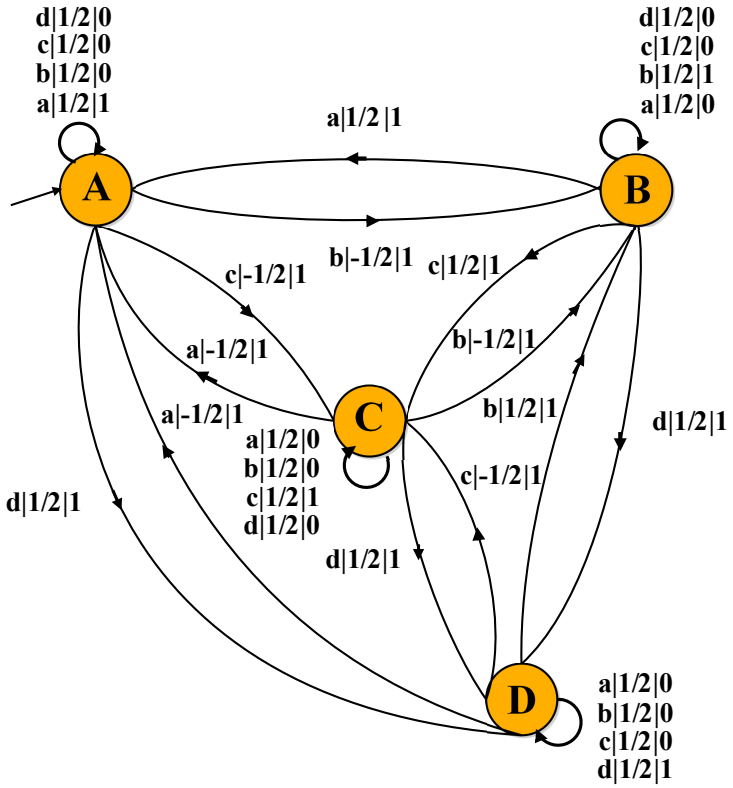


Figure 5.7: Quantum finite-state machine of W state

The quantized version of W state is shown in Fig 5.7. Consider $X = \{a, b, c, d\}$, the input sequence $(abcd)^+$ and the probability of different word sequences of length $L = 4$, i.e., $w_1w_2w_3w_4 \in Y^4$ is computed as $\Pr(w_L) = Tr(U^*(w_L)\rho U(w_L))$, where $\rho = \frac{1}{4} \cdot 1$.

$$UP(1)^0 = \frac{1}{2}\begin{pmatrix} 0 & -1 & -1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & -1 & 1 & 1 \\ 0 & 1 & -1 & 1 \end{pmatrix}, \quad UP(1)^1 = \frac{1}{2}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{pmatrix},$$

$$UP(2)^0 = \frac{1}{2}\begin{pmatrix} 1 & 0 & -1 & 1 \\ 1 & 0 & 1 & 1 \\ -1 & 0 & 1 & 1 \\ -1 & 0 & -1 & 1 \end{pmatrix}, \quad UP(2)^1 = \frac{1}{2}\begin{pmatrix} 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix},$$

$$UP(3)^0 = \frac{1}{2}\begin{pmatrix} 1 & -1 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ -1 & -1 & 0 & 1 \\ -1 & 1 & 0 & 1 \end{pmatrix}, \quad UP(3)^1 = \frac{1}{2}\begin{pmatrix} 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 \end{pmatrix},$$

$$UP(4)^0 = \frac{1}{2}\begin{pmatrix} 1 & -1 & -1 & 0 \\ 1 & 1 & 1 & 0 \\ -1 & -1 & 1 & 0 \\ -1 & 1 & -1 & 0 \end{pmatrix}, \quad UP(4)^1 = \frac{1}{2}\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Quantum information can classify entanglement by means of some mathematical or physical uniformity. Therefore, it helps in increasing the practical abilities of quantum information protocols. In quantum information theory, many multi-particle entangled states (GHZ and W state) and metrology can be represented by MPS. Cluster states are featuring in the context of measurement-based quantum computing. MPS are so powerful and efficient such that they are optimally suited for quantum state tomography in condensed matter physics.

5.5 Weightless Neural Networks

Weightless neural networks (WNNs) are the models of neural computation which have inputs and outputs in binary form. The weightless neuron is implemented by RAM, where the information is stored in a look-up table. Learning is achieved by simply adjusting the contents of the look-up table. It is quite simple, instead of adjusting the weights, which makes the learning process highly flexible and efficient.

5.5.1 RAM node

A RAM node with n inputs has 2^n memory locations addressed by the n-bit string $a = (a_1 a_2 ... a_n)$. A binary signal $x = (x_1 x_2 ... x_n)$ can access one of these locations on the input lines result in producing output $y = C[x]$. Thus, the Boolean function is executed by the neuron i.e., determined by its contents. Fig 5.8 shows the n-input RAM node.

The PLN is based on RAM node. A two-bit number is stored at memory location (0,1 or u) in PLN instead of one-bit number, interpreted as (00, 11 and 01 (or 10)) respectively, where u denotes the node flipping its output with an equal probability

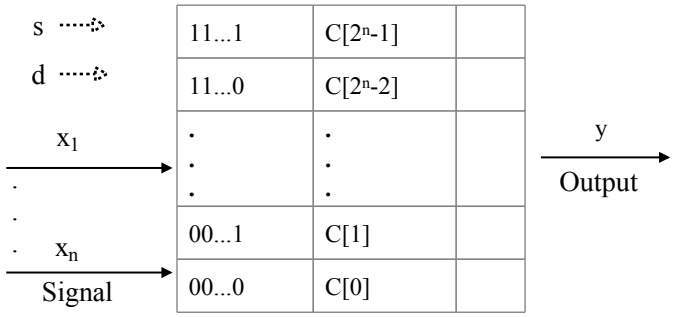


Figure 5.8: Classical RAM node

between 0 and 1. Therefore, PLN consists of RAM node expanded with probabilistic output generator and its output is given as:

$$y = \left\{ \begin{array}{ll} 0 & \text{if } C[x] = 0 \\ 1 & \text{if } C[x] = 1 \\ random(0,1) & \text{if } C[x] = u \end{array} \right\} \tag{5.28}$$

The multi-valued probabilistic logic node (MPLN) stores the probability of the node output 1. The main difference between PLN and MPLN is that it allows a wide range of probabilities to be stored at each memory location. A k-bit valued MPLN mode can store a value n in range of $\{0, ..., 2^k - 1\}$, which is represented as firing probability (p) such that $p = \frac{n}{2^k - 1}$.

5.5.2 Quantum weightless neural networks

Quantum weightless neural networks (QWNNs) have been proposed on several occasions. Some focused on quantum approach using laws of quantum physics, and others are based on quantum-inspired policies. In 2010, Silva et al. [117] proposed a quantum version of classical weightless neural networks named qRAM using mathematical quantization.

5.5.2.1 qRAM node-quantum neuron node

The key idea behind the quantization of RAM is to replace the sets with Hilbert spaces. Moreover, its elements are placed in one-to-one relationship with an orthonormal basis. Generally, the RAM node stores a unique bit at each addressable location. Silva et al. followed this approach by storing one qubit in qRAM. A qRAM node with n inputs has 2^n quantum memory locations. Initially, it has been investigated that qRAM node can simulate the behaviour of classical RAM node. Consider an example of single input and output:

$$A = \begin{bmatrix} I & 0 \\ 0 & X \end{bmatrix}, \begin{array}{l} A\left|0\right\rangle\left|y\right\rangle = \left|0\right\rangle I\left|y\right\rangle \\ A\left|1\right\rangle\left|y\right\rangle = \left|1\right\rangle X\left|y\right\rangle \end{array} \tag{5.29}$$

We store the qubits named selectors and apply the quantum operator (matrix A) to acquire stored qubit. Thus, matrix A "selects" which operator (I or X) is going to be applied on the first qubit in a computational basis. However, a qubit is not directly stored in the quantum register as in case of classical RAM because it can demolish information stored in qRAM memory during measurement. The first qubit

is called selector and second qubit parameter. The n-qubits input $|i\rangle$ qRAM node is represented by the operator N, as shown in Eq (5.30). A qRAM is a collection of an input addressable matrix memories. The input $|i\rangle$ "addresses" or "selects" the memory matrix A_i which returns its memory content. An example of the one qubit addressable qRAM is shown in Fig 5.9. The memory A_0 or A_1 is addressed whether the input qubit $|i\rangle$ is 0 or 1, respectively. The output register $|o\rangle$ depends on the value of the selector $|s_i\rangle$. The qRAM of n input qubits has a set of 2^n of that A's, consequently it has 2^n qubits selectors $|s\rangle$ and one qubit output $|o\rangle$ i.e., set to zero. A quantum circuit of qRAM node with single input is shown in Fig 5.9.

$$N = \sum_{i=0}^{2^n-1} |i\rangle_n \langle i|_n A_{s_i,o} \tag{5.30}$$

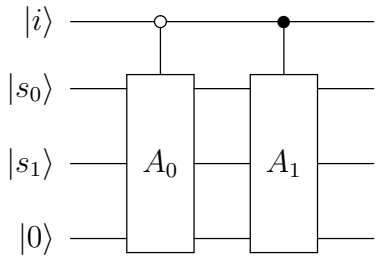


Figure 5.9: Quantum circuit of qRAM node with single input (o for 0's and bullet for 1's)

5.6 Quantum Dynamics

Over the last three decades, quantum control theory is a rapidly developing research area. It has got noteworthy success in various areas of chemistry, molecular physics, quantum optics, and quantum information. In quantum control theory, quantum systems can be controlled whose dynamics are led by principles of quantum mechanics. It has been investigated that it produces a non-linear behaviour on affecting its dynamics during measurement. Usually, the quantum dynamics of a quantum state $|\phi\rangle$ at time $t_1 = t_0 + 1$ is related as

$$|\phi_{t_1}\rangle = U\,|\phi_{t_0}\rangle \tag{5.31}$$

where U is any quantum operator.

5.6.1 Quantum output extractor model

Initially, Neto et al. [118] analyzed the quantum neuron dynamics using an output qubit extraction block. Further, it has been shown that controlled NOT operator produces entanglement between states, and its impact has been explored. Thus, entangled states avert the extraction of output qubit to be given back as an input. The amplitudes of output qubit in qRAM dynamics can be recovered using the mathematical algorithm. In order to find amplitudes of output qubit $|\phi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$, we need to take the square root of sum of probabilities of all quantum states ends with $|0\rangle$ in the even positions and $|1\rangle$ in the odd positions respectively, where

$$\alpha = \sqrt{\sum_{i:even} |\,\phi_i\,|^2},\ \beta = \sqrt{\sum_{i:odd} |\,\phi_i\,|^2} \tag{5.32}$$

Consider an example with single input $|i\rangle$ to rebuild the amplitudes (complex numbers) in qRAM node dynamics. We have an input $|i\rangle$, selectors $|s\rangle = |s_0 s_1\rangle$, and output $|o\rangle$. In Fig 5.10, the output qubit amplitude values are extracted through a mathematical algorithm. For example, if we have a state of 2 qubits $|\psi\rangle = \alpha_0 |00\rangle + \alpha_1 |01\rangle + \alpha_2 |10\rangle + \alpha_3 |11\rangle$, the quantum theory claims that the second qubit can take with probability $|\alpha_0|^2 + |\alpha_2|^2$ the value $|0\rangle$ and $|\alpha_1|^2 + |\alpha_3|^2$ the value $|1\rangle$. The last qubit amplitudes are obtained for $|0\rangle$ and $|1\rangle$ take the square root of the sum of the probability of all the n-qubit states ending with $|0\rangle$, $|q_1, ..., q_{n-2}, 0\rangle$ and all ending with $|q_1, ..., q_{n-2}, 1\rangle$, respectively. It is just a mathematical operation to obtain the amplitudes of the output qubit, which help us to analyze the system as a mathematical entity. However, it is not at all physical realisable operation since amplitudes cannot be obtained or evaluated in general quantum physical systems. The approximation of the amplitudes can be obtained by repetitive measurements according to quantum mechanics postulates. Note that the output qubit must have real amplitudes after the extraction step.

$$|i\rangle = \begin{bmatrix} i_0 \\ i_1 \end{bmatrix},\ |s_0\rangle = \begin{bmatrix} s_0^0 \\ s_0^1 \end{bmatrix},\ |s_1\rangle = \begin{bmatrix} s_1^0 \\ s_1^1 \end{bmatrix},\ |o\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \tag{5.33}$$

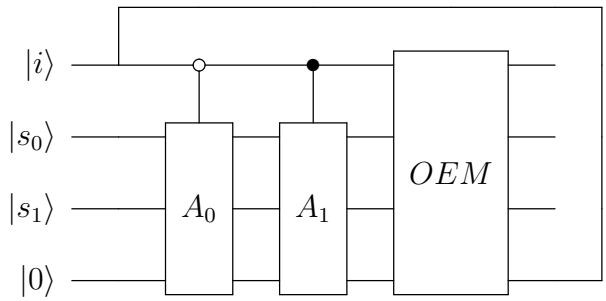


Figure 5.10: Extraction of output qubit and feedback to input register (iterations)

The quantum state $|\phi\rangle = |i\rangle\,|s\rangle\,|o\rangle$ is given as an entry to qRAM node such as

$$\begin{aligned}|\phi\rangle =& i_0 s_0^0 s_1^0 \alpha\,|0000\rangle + i_0 s_0^0 s_1^0 \beta\,|0001\rangle + i_0 s_0^0 s_1^1 \alpha\,|0010\rangle + i_0 s_0^0 s_1^1 \beta\,|0011\rangle + i_0 s_0^1 s_1^0 \alpha\,|0100\rangle + \\ & i_0 s_0^1 s_1^0 \beta\,|0101\rangle + i_0 s_0^1 s_1^1 \alpha\,|0110\rangle + i_0 s_0^1 s_1^1 \beta\,|0111\rangle + i_1 s_0^0 s_1^0 \alpha\,|1000\rangle + i_1 s_0^0 s_1^0 \beta\,|1001\rangle + \\ & i_1 s_0^0 s_1^1 \alpha\,|1010\rangle + i_1 s_0^0 s_1^1 \beta\,|1011\rangle + i_1 s_0^1 s_1^0 \alpha\,|1100\rangle + i_1 s_0^1 s_1^0 \beta\,|1101\rangle + i_1 s_0^1 s_1^1 \alpha\,|1110\rangle + \\ & i_1 s_0^1 s_1^1 \beta\,|1111\rangle\end{aligned} \tag{5.34}$$

On applying the qRAM node N to above quantum state such as $|\phi'\rangle = N\,|\phi\rangle$, we have

$$\begin{aligned}|\phi'\rangle =& i_0 s_0^0 s_1^0 \alpha\,|0000\rangle + i_0 s_0^0 s_1^0 \beta\,|0001\rangle + i_0 s_0^0 s_1^1 \alpha\,|0010\rangle + i_0 s_0^0 s_1^1 \beta\,|0011\rangle + i_0 s_0^1 s_1^0 \beta\,|0100\rangle + \\ & i_0 s_0^1 s_1^0 \alpha\,|0101\rangle + i_0 s_0^1 s_1^1 \beta\,|0110\rangle + i_0 s_0^1 s_1^1 \alpha\,|0111\rangle + i_1 s_0^0 s_1^0 \alpha\,|1000\rangle + i_1 s_0^0 s_1^0 \beta\,|1001\rangle + \\ & i_1 s_0^0 s_1^1 \beta\,|1010\rangle + i_1 s_0^0 s_1^1 \alpha\,|1011\rangle + i_1 s_0^1 s_1^0 \alpha\,|1100\rangle + i_1 s_0^1 s_1^0 \beta\,|1101\rangle + i_1 s_0^1 s_1^1 \beta\,|1110\rangle + \\ & i_1 s_0^1 s_1^1 \alpha\,|1111\rangle\end{aligned} \tag{5.35}$$

The output qubit amplitudes are recovered using Eq (5.30) as:

$$\begin{aligned} |\alpha|^2_{t+1} = {} & |i_0 s_0^0 s_1^0 \alpha|^2 + |i_0 s_0^0 s_1^1 \alpha|^2 + |i_0 s_0^1 s_1^0 \beta|^2 + |i_0 s_0^1 s_1^1 \beta|^2 + \\ & |i_1 s_0^0 s_1^0 \alpha|^2 + |i_1 s_0^0 s_1^1 \beta|^2 + |i_1 s_0^1 s_1^0 \alpha|^2 + |i_1 s_0^1 s_1^1 \beta|^2 \end{aligned} \tag{5.36}$$

$$\begin{aligned} |\beta|^2_{t+1} = {} & |i_0 s_0^0 s_1^0 \beta|^2 + |i_0 s_0^0 s_1^1 \beta|^2 + |i_0 s_0^1 s_1^0 \alpha|^2 + |i_0 s_0^1 s_1^1 \alpha|^2 + \\ & |i_1 s_0^0 s_1^0 \beta|^2 + |i_1 s_0^0 s_1^1 \alpha|^2 + |i_1 s_0^1 s_1^0 \beta|^2 + |i_1 s_0^1 s_1^1 \alpha|^2 \end{aligned} \tag{5.37}$$

In the end, we recuperate the output qubit to give back as input itself in the next iteration. The process is repeated many times depending upon the value set of iterations (orbit) in quantum dynamical studies.

$$|0\rangle_{t+1} = \alpha_{t+1}|0\rangle + \beta_{t+1}|1\rangle \tag{5.38}$$

According to the quantum mechanics theory, we can only get an approximate value of amplitudes by repeating measurements finitely. In Fig 5.11, we proposed a detailed qRAM node network using an output extraction model (OEM) with additional supplementary queue and central queue. In extraction model dynamics, the output qubit is extracted via OEM block and is fedback in the qRAM input register. The output qubit amplitude values are extracted as mentioned above and stored in a classical memory model queue, that is used to keep track of the amplitudes of the output qubit at time t for the next iteration. In this model, n-qubit input $|i\rangle$ is given with 2^n qubits selectors $|s\rangle = |s_0 s_1\rangle$ and one-qubit output $|o\rangle$. qRAM node network uses the recuperate procedure for amplitudes as presented above.
Here, we will give the input of each MPS to qRAM node (N) and extract the output qubit for next iteration. For example, an input $I = I_1 I_2 ... I_n$ is given into qRAM node network symbol by symbol. During the process, several transition matrices are applied after reading each symbol corresponding to node N. Repeat the computation process until the last symbol has been fed, and the central queue is not empty. Consider an example of 2-qubit GHZ state which is given as an input to qRAM node network with selectors $|s\rangle$ and output qubit $|o\rangle$ is kept as zero. In the first iteration of computation process, the node operators $N = N_1 ... N_m$ (transition matrix) are going to be applied on the first qubit whether it is zero or one. Thus, the first qubit selects the operator and stores the result in output qubit. During the whole computation process, we keep track of quantum states at any time t in a supplementary queue. Further, we transfer the quantum states from a supplementary queue to central called previous states at time $t-1$ for the next iteration. At the initial stage, the output probability of GHZ state is calculated as 0.5. In the next iteration, the qubits are built using the amplitude values encountered by the repetition of the measurement procedure using Eq (5.38). After extracting the output qubit, the probability of an output quantum state is coming out to be 1. There may occur positive or negative interference between quantum states.
After the last symbol has been fed, if the system is in the set of final states in a supplementary queue, then the probabilities associated with the present state can be summed. Therefore, the above qRAM node network can be applicable in pattern recognition, biological sequencing, meteorological forecasting quantum automata, and quantum language recognition, which is left for future work. Here, we focused on the entanglement dynamics of matrix product state using output extraction model, where the MPS is given as an input on extracting the output qubit for next iteration, and the results are plotted in next section.

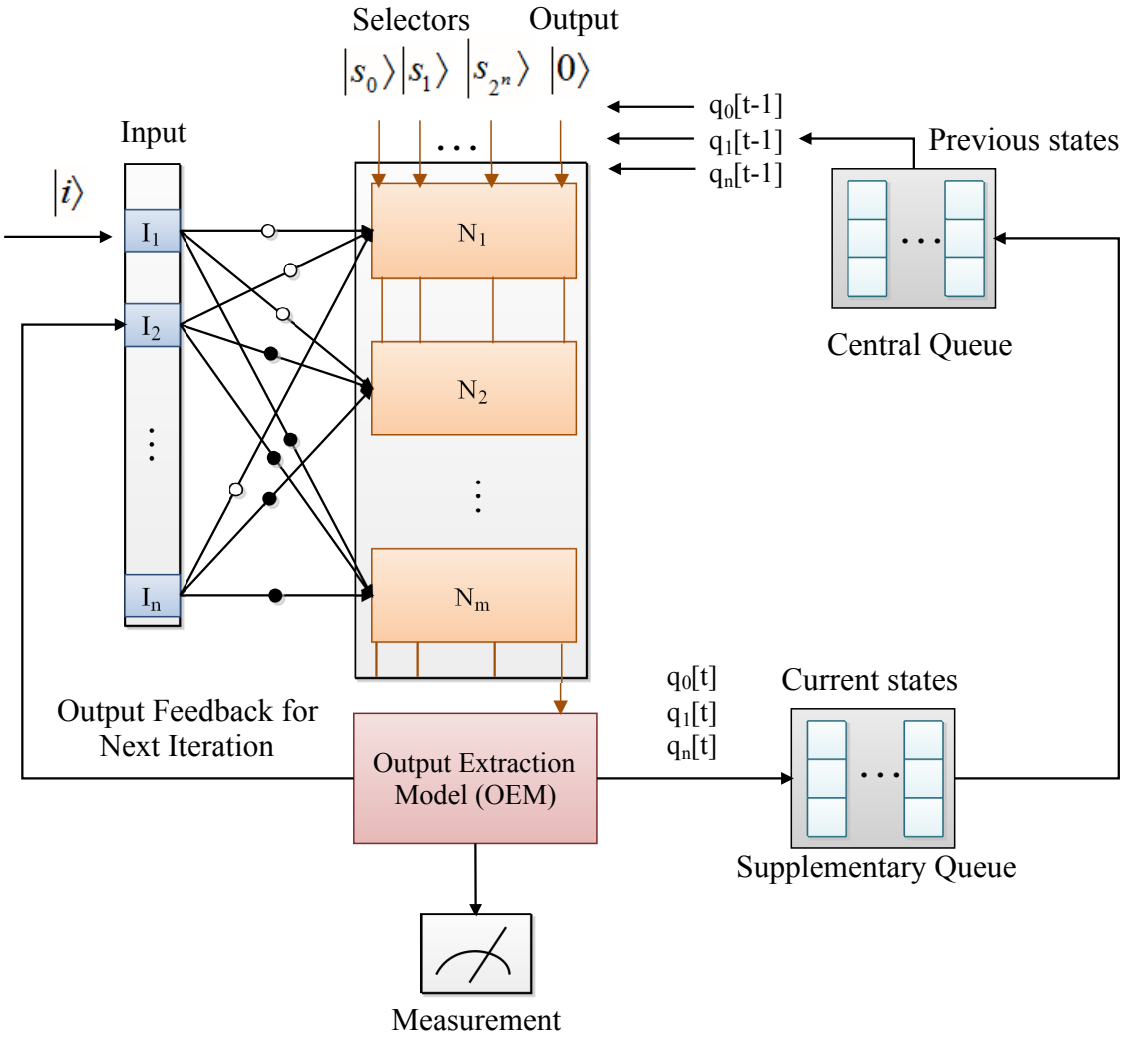


Figure 5.11: qRAM node network using an output extraction model

5.6.2 Experiments and discussion

Focusing on investigating the dynamics of matrix product state, we have fed the output qubit $|o\rangle$ back to the input by using OEM. We have given $|i, s_0, s_1, o\rangle$ to qRAM and extracted the output qubit using the recover procedure presented in above section. The whole process of output feedback is called iteration, and it means to repeat the process many times. In dynamics studies, the process is recapitulated as a function. We set the value for such iterations called orbit, where the previous output of function is used as an input to a function for next iteration. On plotting the orbit, it represents the dynamics shape. Thus, the output qubit amplitudes are plotted for each MPS during the dynamics.

The dynamics behaviour can be classified depending upon the plotting the orbit: it can be non-damped, under-damped, or over-damped behaviour in time. In order to determine the dynamics of any MPS at the current time, we have applied their respective unitary operator on quantum state at previous time. We have initialized the amplitudes of output qubits in the set of $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$ after normalized the states. Here, we started with the maximally entangled GHZ state. On plotting the GHZ output qubit amplitudes, it shows an undamped behaviour, i.e., permanent oscillation as shown in Fig 5.12. During the 100 iterations,

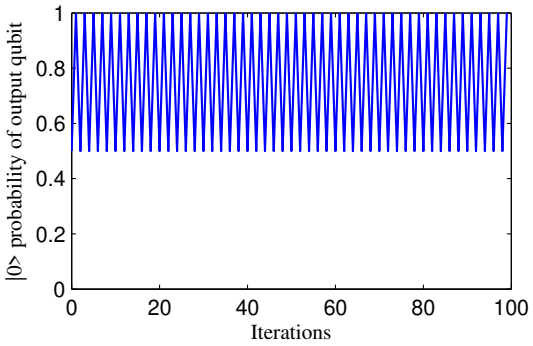


Figure 5.12: Dynamics of GHZ state, initial condition: $|i\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle, |s_0\rangle = |0\rangle, |s_1\rangle = |1\rangle, |o\rangle = |0\rangle$

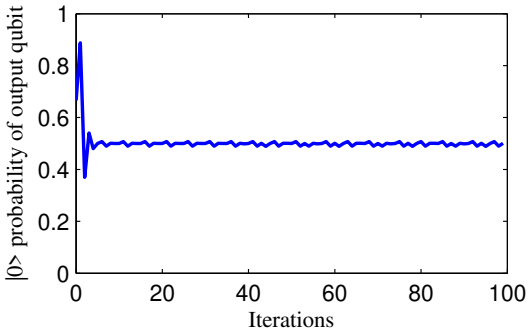


Figure 5.13: Dynamics of AKLT state, initial condition: $|i\rangle = \sqrt{\frac{2}{3}}|0\rangle + \frac{1}{\sqrt{3}}|1\rangle, |s_0\rangle = |0\rangle, |s_1\rangle = |1\rangle, |o\rangle = |0\rangle$

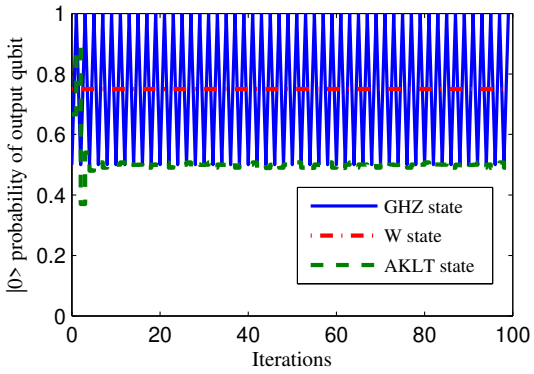


Figure 5.14: Quantum Dynamics of GHZ, AKLT and W state

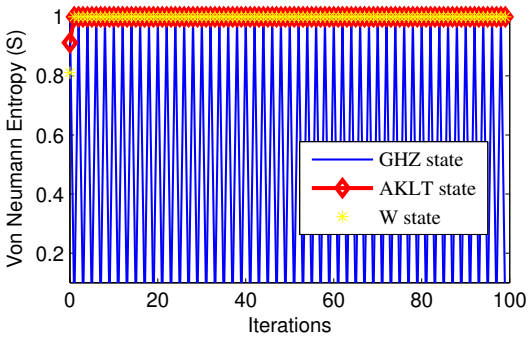


Figure 5.15: Von Neumann Entropy of MPS

it shows oscillating nature between 0.5 and 1 probability of output qubit on giving output back to the input. It is known that 3-qubit cluster state is equivalent to GHZ state. Thus, its dynamics are similar to GHZ state. Further, we plotted dynamics for AKLT state and W state using unitary operator. The dynamics of AKLT state starts with higher value and after it is decreased. There are oscillations but in a convergent series. But, it remains constant for W state. Thus, the dynamics of GHZ and AKLT state shows an undamped and under-damped behaviour in Fig 5.13 and 5.14, respectively.

To analyze the entanglement behaviour of MPS, we have checked the Von Neumann entropy of the output quantum state. We have analyzed the entanglement behaviour by taking into consideration the action of output extraction operator as an interaction with the environment. Fig 5.15 shows the measured entanglement behaviour of GHZ, AKLT, and W state. It shows an oscillating without damping for GHZ state. Thus, the dynamics of entropy are increasing and decreasing between the intervals. Consequently, it can be said that output extractor operator as restoration of entanglement because of reconstruction of qubits. Fig 5.15 shows critically damping for AKLT and W state, respectively. On analyzing, the entanglement increases in the beginning and converges to a fixed value.

Chapter 6

Quantum Omega Automata

Inspired by the results of finite automata working on infinite words, we studied the quantum ω-automata with Büchi, Muller, Rabin and Streett acceptance conditions. Quantum finite automata play a pivotal part in quantum information and computational theory. Investigation of the power of quantum finite automata over infinite words is a natural goal. In this chapter, we have investigated the classes of quantum ω-automata from two aspects: language recognition and their closure properties. It has been shown that quantum Muller automaton is more dominant than quantum Büchi automaton. Furthermore, we have demonstrated the languages recognized by one-way quantum finite automata with different quantum acceptance conditions. Finally, we have proved the closure properties of quantum ω-automata.

6.1 Introduction

Some forty years ago, automata on infinite words have gained a lot of importance and have a huge impact in the field of theoretical computer science for the verification and specification of reactive systems. In the sixties, Büchi [119] studied the automata on infinite words and showed a relationship between ω-regular languages and monadic second-order logic (MSOL) of infinite words. After a few years, Muller [120] introduced another definition of finite automata on infinite words. In the meantime, Landauer [8] articulated a concept of reversibility in quantum computing. McNaughton [121] generated the infinite sequences by finite automata and established the relationship with regular expressions, grammars, and logical systems. Further, it has been shown that Muller automaton and deterministic finite automaton (DFA) can recognize all ω-regular languages. Further, the research by Rabin [122] demonstrated the decision problem of Büchi for MSOL of infinite words to monadic second order of the infinite binary tree.
Since the last two decades, various quantum variants of each classical automata have been introduced. Some of the quantum variants are more powerful than classical automata. Recently, Wang et al. [123] introduced the notion of quantum Büchi automata with disturbing and non-disturbing acceptance conditions for ω-words. Further, Wang et al. studied the closure properties for QBA and described the relationship with classical Büchi automata based on pumping lemmas. In 2010, Bērzina [124] introduced the quantum variants of ω-automata for languages over infinite words. Further, she has examined the class of languages described Büchi measure-once quantum finite state automaton with bounded error and its closure

properties. We have demonstrated the languages accepted by quantum finite state automata using Muller, Rabin and Streett acceptance conditions and proved that quantum Muller acceptance is more powerful than quantum Büchi acceptance condition. Further, we have defined the closure properties of quantum ω-automata.

6.2 Classes of Quantum Omega-Automata

In this section, some notations and formal definitions of classes of quantum ω-automata: quantum ω-automaton, quantum Büchi automaton, quantum Muller automaton, quantum Rabin automaton and quantum Streett automaton are given.

Definition 6.2.1. [124,125] *A quantum ω-automaton (M_a) is defined as quintuple $(Q, \Sigma, \delta, q_0, Q_{acc})$, where*

- *Q is a set of states,*
- *Σ is an input alphabet,*
- *q_0 is a starting state,*
- *The transition function δ is defined by $Q \times \Sigma \times Q \to \mathbb{C}$, it satisfy the unitary condition:*

$$\sum_{p \in Q}^{\forall (q_1,\sigma),(q_2,\sigma) \in Q \times \Sigma} \overline{\delta(q_1,\sigma,p)}\delta(q_2,\sigma,p) = \left\{ \begin{array}{l} 1 \; q_1 = q_2 \\ 0 \; q_1 \neq q_2 \end{array} \right\} \tag{6.1}$$

- *Q_{acc} is the acceptance component.*

The transition function δ is characterized as V_σ unitary matrix of automaton M_a, where V_σ is a unitary evolution of state q after reading the symbol σ, which is defined as

$$V_\sigma(|q\rangle) = \sum_{q' \in Q}^{\sigma \in \Sigma} \delta(q,\sigma,q') \, |q'\rangle \tag{6.2}$$

On reading an input ω-word $\sigma = \sigma_1\sigma_2...\sigma_n \in \Sigma^\omega$, the computation process of M_a begins in the superposition $\rho_0 = |q_0\rangle \langle q_0|$ and the corresponding unitary transition is applied on reading the current symbol.

Definition 6.2.2. [7] *The density matrix is an alternate representation of a state in quantum mechanics. Such states are called the mixed states and can be represented as a summation of orthonormal bases $|\psi_i\rangle' s$, $\rho = \sum_i \rho_i |\psi_i\rangle \langle\psi_i|$, where ρ_i refers to the probability of the quantum system to be in state ψ_i, and ψ_i's are the diagonal basis for ρ. ρ_i's are called eigenvalues of the density matrix ρ. On Hilbert space, it must fulfil the trace condition, i.e. the sum of diagonal elements of the matrix must be 1. The state of automaton is represented as density matrix $\rho = |\psi\rangle \langle\psi|$. A density matrix $\rho(j) = |\psi_j\rangle \langle\psi_j|$, $\left(|\psi_j\rangle = \sum_{i=1}^{n} \alpha_i |q_i\rangle \right)$ is called accepting $Q_{acc} \subseteq Q$ with probability p if $\sum_{q_i \in Q_{acc}}^{n} |\alpha_i|^2 > p$, which is the acceptance probability of superposition.*

Let $M{=}(Q, \Sigma, \delta, q_0, Q_{acc})$ be a quantum ω-automaton. A run of M on a ω-word $\sigma = \sigma_1\sigma_2...\sigma_n \in \Sigma^\omega$ is an infinite sequence of density operators $\rho_\omega = \rho(0)\rho(1)...\rho(n)$, where $\rho(j) = |\psi_j\rangle \langle\psi_j|, \left(|\psi_j\rangle = \sum_{i=1}^{n} \alpha_i |q_i\rangle\right)$ and $|q_i\rangle$ is the set of states of automaton M and the following holds: $\rho(0) = \rho_0$ and $\rho(i) = U_{\sigma_i}\ \rho(i-1)U^*_{\sigma_i}$ for $i{>}0$. The acceptance conditions for a quantum case can be seen in a similar way as for classical ω-automata.

Definition 6.2.3. [124] *A Büchi acceptance condition for quantum case: Q_{acc} is a subset of Q. An infinite run $\rho_\omega = \rho(0)\rho(1)...\rho(n)$ is called Büchi accepting with probability p if run ρ_ω contains infinitely many Q_{acc} accepting density matrices. The definition of other classes of omega automata for a quantum case is the same as for Büchi automaton for quantum case. But, the definition of an acceptance condition is different.*

Definition 6.2.4. *A run of a quantum Muller automaton on a ω-word $\sigma = \sigma_1\sigma_2...\sigma_n \in \Sigma^\omega$ is said to be successful if Q_{acc} is a subset of 2^Q. An infinite run $\rho_\omega = \rho(0)\rho(1)...\rho(n)$ is called Muller accepting with probability p if run ρ_ω contains infinitely many Q_{acc} accepting density matrices.*

Definition 6.2.5. [22] *A Rabin acceptance condition for quantum case: Q_{acc} is a finite set of pairs (H_i, K_i), where H_i's and K_i's are subsets of Q. An infinite run $\rho_\omega = \rho(0)\rho(1)...\rho(n)$ is called Rabin accepting with probability p if run ρ_ω contains an only finite number of H_i accepting and infinite number of K_i accepting density matrices for some $i \in \{1, 2, ...s\}$.*

Definition 6.2.6. [22] *A Streett acceptance condition for quantum case: Q_{acc} is a finite set of pairs (H_i, K_i), where H_i's and K_i's are subsets of Q. An infinite run $\rho_\omega = \rho(0)\rho(1)...\rho(n)$ is called Streett accepting with probability p if run ρ_ω contains an only infinite number of H_i accepting or a finite number of K_i accepting density matrices for every $i \in \{1, 2, ...s\}$.*

Definition 6.2.7. [126] *MM-1QFA over infinite words is defined as a sextuple $(Q, \Sigma, \delta, q_0, Q_{acc}, Q_H)$, where*

- *Q is a finite set of states,*
- *Σ is an input alphabet,*
- *State transition δ is defined by $Q \times \Sigma \cup \{\#\} \times Q \to \mathbb{C}$, which represents the amplitudes flows from one state to another after reading the symbol from the input tape. It must satisfy the unitary condition.*
- *q_0 is a starting state,*
- *$Q_H \subseteq Q$ is a set of halting states,*
- *Q_{acc} is the acceptance component ($Q_{acc} \subseteq Q_H$).*

The computation process of MM-1QFA over infinite words consists of an input string $x{=}\#\sigma_1\sigma_2...\sigma_n \in \Sigma^\omega$. The tape head reads x from left-end marker # to end of string and transition function corresponding to each symbol is performed. It begins with the superposition ρ_0. After every transition, it measures its states with respect

to the subspace. Therefore, after every step of measurement, the superposition of states ends with measured subspace.
Due to the non-zero probability of halting of MM-1QFA, it is convenient to have a path of the aggregate rejecting and accepting probabilities. Hence, the state of automaton is denoted as $(|\phi\rangle, p_1, p_2, ..., p_H)$, where p_i is the aggregate probability of halting state $q_i \in Q_H$ and δ is defined as $(P_{non}|\phi^{'}\rangle, p_1 + \|P_1|\phi^{'}\rangle\|^2, p_2 + \|P_2|\phi^{'}\rangle\|^2, p_{|Q_H|} + \|P_{|Q_H|}|\phi^{'}\rangle\|^2)$, where $|\phi^{'}\rangle = V_\sigma|\phi\rangle$.

6.3 Language Recognition and Relationship

In this section, we have shown the languages recognized by quantum finite ω-automata with Muller acceptance condition and Rabin acceptance condition. Furthermore, we have investigated their relationship with quantum Büchi automaton based on language recognition power.

6.3.1 Quantum Muller Automaton

Theorem 6.3.1. *A language recognized by quantum Büchi automaton is also recognized by quantum Muller automaton.*

Proof. Consider a quantum Büchi automaton $M_{QBA} = (Q, \Sigma, \delta, q_0, Q_{acc})$. We can construct quantum Muller automaton $M_{QMA} = (Q, \Sigma, \delta^{'}, q_0, F_{acc})$

- $\delta^{'}(q, \sigma) = q^{'}\ if(q, \sigma, q^{'}) \in \delta$
- $\{F_{acc} \in 2^Q | Q_{acc} \bigcap F_{acc} \neq \phi\}$

Thus, M_{QMA} is well-defined. As, there can be at most one transition that exists in M_{QBA}. Therefore, it can be easily checked that quantum Muller automaton simulates the quantum Büchi automaton.

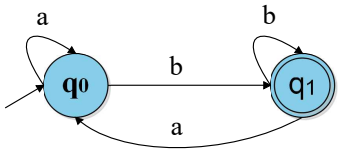


Figure 6.1: DMA for language L_1

Let us consider ω-language $L_1 = \{\alpha | \alpha\ has\ infinitely\ many\ b's\}$ in the given input alphabet $\Sigma = \{a, b\}$. It has proved that language L_1 cannot be recognized by QBA, but it can be recognized by deterministic Büchi automaton (DBA) [124]. Therefore, QBA with bounded error cannot recognize the whole class of languages recognized by DBA. We will prove that QMA recognizes a larger language class than QBA.
It is known that deterministic Muller automaton (DMA) can be designed for L_1 with acceptance state q_1 as shown in Fig 6.1. Now, consider the same language by quantum Muller acceptance condition. Let $M = (Q, \Sigma, \delta, q_0, Q_{acc})$ be a QMA for L_1, where Q=$\{q_0, q_1, q_2, q_3\}$, $\Sigma = \{a, b\}$, q_0 is an initial state, $Q_{acc} = \{\{q_1\}\}$ and transition function δ is defined by $Q \times \Sigma \times Q \to \mathbb{C}$, it satisfy the unitary condition:

$$\sum_{p\in Q}^{\forall(q_1,\sigma),(q_2,\sigma)\in Q\times\Sigma} \overline{\delta(q_1,\sigma,p)}\delta(q_2,\sigma,p) = \left\{ \begin{array}{ll} 1 & q_1 = q_2 \\ 0 & q_1 \neq q_2 \end{array} \right\} \tag{6.3}$$

The transition function δ is represented by set of unitary transition matrices $\{V_\sigma\}_{\sigma\in\Sigma}$, where V_σ is a unitary evolution of state q after reading the symbol σ, which is defined as

$$V_\sigma(|q\rangle) = \sum_{q'\in Q}^{\sigma\in\Sigma} \delta(q,\sigma,q')\,|q'\rangle \tag{6.4}$$

On reading an input ω-word $\sigma = \sigma_1\sigma_2...\sigma_n \in \Sigma^\omega$, the computation process of automaton begins in the superposition $\rho_0 = |q_0\rangle\langle q_0|$ and the corresponding unitary transition matrix is applied on reading the current symbol.

$$V_\# = I, V_a = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & 0 & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & 0 & 0 & -\frac{1}{\sqrt{2}} \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \end{bmatrix}, V_b = \begin{bmatrix} 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & 0 & 0 & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & 0 & 0 & -\frac{1}{\sqrt{2}} \end{bmatrix} \tag{6.5}$$

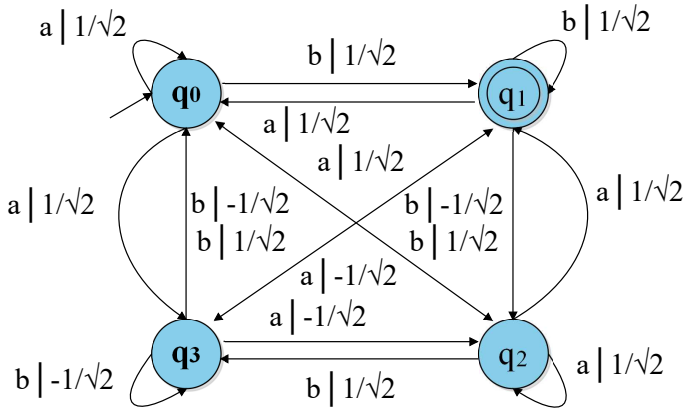


Figure 6.2: QMA for language L_1

The state transition diagram of QMA for L_1 is shown in Fig 6.2. It is noted that the transitions with amplitudes between states q_0 and q_1 resemble exactly as in the case of DMA. On reading the symbol $\sigma \in \Sigma^\omega$, its corresponding unitary matrix is applied to the current state. We have $Q_{acc} = \{\{q_1\}\}$, i.e. for each successful run state q_1 occurs infinitely often and q_0 finite often. During computation, the amplitudes cancel each other if they have opposite phases. Thus, if the infinite word belongs to L_1, then the input is said to be accepted with probability at least $1/2$, else rejected. Hence, the language consisting infinitely many b's over the input alphabet $\Sigma = \{a, b\}$ can be recognized by QMA, but cannot be recognized by QBA.
Consider the language L_2 i.e. compliment of L_1 (consisting of a finite number of b's). It is known that DBA cannot be designed for L_2. As a result, it cannot

be recognized by QBA. Although, it can be recognized by non-deterministic Büchi automata (NBA). It has been proved that the language L_2 can be recognized by quantum ω-automaton with Streett acceptance criteria. Hence, QSA is more dominant than QBA in terms of language recognition [124]. It is known that QSA is dual to QRA. Thus, the language can also be recognized by quantum ω-automaton with Rabin acceptance condition. □

6.3.2 Quantum Rabin Automaton

Theorem 6.3.2. *A quantum Rabin automaton contains the limit of group languages L_ω (a subset of regular languages), where a language L is recognized by a measure-once quantum finite automaton.*

Proof. Let L_ω be a language recognized by Rabin quantum finite state automaton $M_{QRA} = (Q, \Sigma, \delta, q_0, Q_{acc1})$. Consider a MO-1QFA $M = (Q, \Sigma, \delta, q_0, Q_{acc2})$, where $Q_{acc1} = Q_{acc2}$ which is a finite set of pairs $(H_1, K_1), (H_2, K_2), ..., (H_i, K_i)$ such that for some $i \in \{1, 2, ...s\}$.

$$Q_{acc1} = \left\{ \begin{array}{l} H_i = \{q_0, q_1, ...q_n | q_i \notin Q_{acc2}\} \\ K_i = \{q_0, q_1, ...q_n | q_i = Q_{acc2}\} \end{array} \right\} \tag{6.6}$$

Let us assume an infinite word $w \in L_\omega$ i.e. limit of a group language. It can be easily checked that on reading w, QRA results in many accepting superpositions. Therefore, the limit of a group language is recognized by QRA. □

6.4 MM-1QFA over Infinite Words

In this section, we have introduced a MM-1QFA with quantum Büchi, Muller and Rabin acceptance criteria.

6.4.1 MM-1QFA with QBA

Theorem 6.4.1. *A language consisting of infinite words starting with 'a' is recognized by a measure-many quantum finite automaton with Büchi acceptance condition. But, it cannot be recognized by a measure-once quantum finite automaton with Büchi acceptance condition, i.e. the language does not belong to limit languages.*

Proof. The proof has been shown in [124]. □

6.4.2 MM-1QFA with QMA

Definition 6.4.1. *A measure-many one-way quantum finite automaton with Muller acceptance condition follows that Q_{acc} is a subset of 2^Q, where Q_{acc} is a set of accepting states. The computation process of an input string is said to be accepted if the probability to be in states $q_i \in Q_{acc}$ is 1 after reading the whole input string.*

Example 6.4.1. *Let us consider a measure-many quantum finite automaton with Muller acceptance condition recognizing a language $L_3 = (a + b)^*(a + b)^\omega$.*

Let $M = (Q, \Sigma, \delta, q_0, Q_{acc}, Q_H)$ be a MM-1QFA with Muller acceptance condition, where Q=$\{q_0, q_1, q_2\}, \Sigma = \{a, b\}, Q_H = \{q_1, q_2\}, Q_{acc} = Q_H$ and transition matrices:

$$V_{\#} = I, V_a = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \end{bmatrix}, V_b = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \end{bmatrix} \tag{6.7}$$

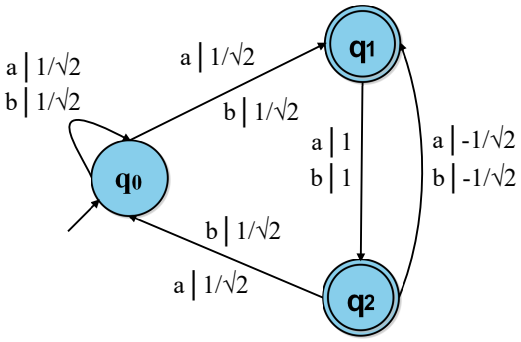


Figure 6.3: QMA for language L_3

If an infinite word belongs to language, then after every step of measurement, the superposition of states ends with measured subspace. Therefore, each time MM-1QFA with QMA visits an accepting state, it sends half of the non-halting probability to the halting state and continues the computation process with remaining states. The state transition diagram of QMA for L_3 is shown in Fig 6.3. Suppose a word is recognized by MM-1QFA with QMA, then accepting states visited infinitely and its probability to be in Q_{acc} is comes out to be $\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + ...$ equals to 1. Else, if the infinite word does not belong the language, then it has visited acceptance state finite number of times i.e. a finite number of i prefixes instead of infinite ω-word. In such case, the probability of acceptance comes out to be $\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + ... + \frac{1}{2^i}$ equals to be less than 1.

6.4.3 MM-1QFA with QRA

Example 6.4.2. *Let us consider a example with Rabin measure-many quantum finite automaton recognizing a language* $L_4 = (a(b + c) + (a + b)b)^*(a + b)^\omega$.

Consider a MM-1QFA with Rabin acceptance condition, where Q=$\{q_0, q_1, q_2\}, \Sigma = \{a, b, c\}, Q_H = \{q_1, q_2\}, Q_{acc} = \{q_2\}, V_c |q_2\rangle = |q_0\rangle$ and unitary transition matrices:

$$V_{\#} = I, V_a = \begin{bmatrix} 0 & -\frac{1}{2} & \frac{\sqrt{3}}{2} \\ 1 & 0 & 0 \\ 0 & -\frac{\sqrt{3}}{2} & -\frac{1}{2} \end{bmatrix}, V_b = \begin{bmatrix} 0 & 1 & 0 \\ \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \end{bmatrix} \tag{6.8}$$

Here, Q_{acc} defines a set of accepting states which describe the language having strings ending with infinite words of a or b. On following a computation process of

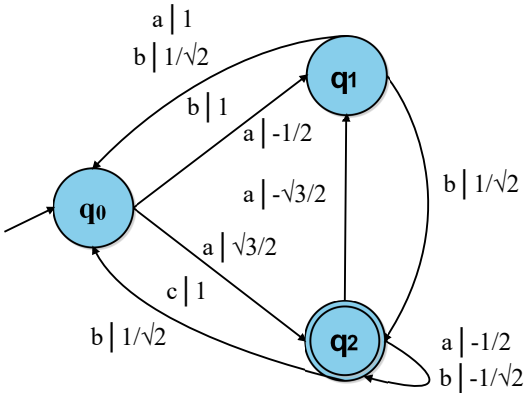


Figure 6.4: QRA for language L_4

MM-1QFA as above, we measured after reading each symbol. An infinite word ω belongs to the language $\{\omega \in L_4\}$ is said to be recognized by MM-1QFA with QRA acceptance criteria if $|q_2\rangle$ is visited infinite times. It is noted that after reading the a, it sends half of the non-halting probability to the halting state and the probability of acceptance comes out to be close to 1. It can be easily checked that if the infinite word does not belong to language, then its probability of acceptance is less than 1.

6.5 Closure Properties

In this section, we discuss the closure properties of ω-languages recognized by Quantum ω-automata. However, the class of ω-languages recognized by the quantum finite state automata with Büchi acceptance condition are closed under union, but not under complementation and intersection. It has been proved in [124]. Here, we will focus on the closure properties of other quantum ω-automata (QMA, QRA and QSA). Let $M_1 = (Q_1, \Sigma, \delta_1, q_{01}, Q_{acc1})$ and $M_2 = (Q_2, \Sigma, \delta_2, q_{02}, Q_{acc2})$ are two quantum Muller automata over the alphabet Σ and $L_{\omega_1}, L_{\omega_2}$ are the languages recognized by M_1 and M_2 respectively.

Theorem 6.5.1. *The class of languages recognized by quantum Muller automaton is closed under union.*

Proof. Let $M_3 = (Q_3, \Sigma, \delta_3, q_{03}, Q_{acc3})$ be a quantum Muller automaton (QMA) i.e. union of two QMAs over input alphabet Σ. The proof is exactly similar to as quantum Büchi automaton and set of accepting states is the union of quantum accepting states of both QMA's $Q_{acc3} = (Q_{acc1} \bigcup Q_{acc2})$. And, $Q_3 = Q_1 \times Q_2$, $\delta((q_1, q_2), \sigma) = (q_3, q_4)$ such that $\delta(q_1, \sigma) = q_3$, $\delta(q_2, \sigma) = q_4$ are transition functions (unitary matrix), $q_{03} = (q_{01}, q_{02})$ and quantum accepting state (Q_{acc3}) contains the set of pairs $\{(q_1, q_1^{'}), (q_2, q_2^{'}), ..., (q_n, q_n^{'})\}$, where the set $\{q_1, q_2, ..., q_n\} \in Q_{acc1}$ and $\{q_1^{'}, q_2^{'}, ..., q_n^{'}\} \in Q_{acc2}$. An infinite run of M_3 on σ is $\rho_\omega = \rho(0)\rho(1)...\rho(n)$ is said to be recognized by M_3 with probability p if run ρ_ω contains infinity many Q_{acc3} accepting density matrices such that Q_{acc3} is a subset of Q_3. □

Now to prove the closure properties of QRA, we assume that $M_1 = (Q_1, \Sigma, \delta_1, q_{01}, Q_{acc1})$ and $M_2 = (Q_2, \Sigma, \delta_2, q_{02}, Q_{acc2})$ are two quantum Rabin automata over the alphabet Σ and $L_{\omega_1}, L_{\omega_2}$ are the languages recognized by M_1 and M_2 respectively.

The class of languages recognized by QRA is closed under union, intersection, and complementation.

- $L_{\omega_1}(M_1) \bigcup L_{\omega_2}(M_2)$ is recognizable by QRA.
- $\Sigma^\omega / L_{\omega_1}(M_1)$ is recognizable by QRA.
- $L_{\omega_1}(M_1) \bigcap L_{\omega_2}(M_2)$ is recognizable by QRA.

Theorem 6.5.2. *If L_{ω_1} and L_{ω_2} are the ω-languages recognized by M_1 and M_2 QRA respectively, then $L_{\omega_1}(M_1) \bigcup L_{\omega_2}(M_2)$ is also recognized by a QRA.*

Proof. Let $M_3 = (Q_3, \Sigma, \delta_3, q_{03}, Q_{acc2})$ be a QRA i.e. union of two QRA's over input alphabet Σ, where $Q_3 = Q_1 \times Q_2$, $\delta((q_1, q_2), \sigma) = (q_3, q_4)$ such that $\delta(q_1, \sigma) = q_3$ and $\delta(q_2, \sigma) = q_4$ are transition functions (unitary matrix), $q_{03} = q_{01} \times q_{02}$.
To prove the theorem, we have to show that if a word σ is recognized by M_3 QRA $(M_1 \bigcup M_2)$, then σ is in M_1 or M_2 QRA. Let ω-word $x{=}\sigma_1\sigma_2...\sigma_n$. has an infinite run $\rho_\omega = \rho(0)\rho(1)...\rho(n)$ is recognized by M_1 with probability p if run ρ_ω contains an only finite number of E_i accepting and infinite number of F_i accepting density matrices belongs to Q_{acc1}. Therefore, Q_{acc3} contains all sets of pairs $(E_{11}E_{21}, F_{11}F_{21}), ..., (E_{1i}E_{2i}, F_{1i}F_{2i})$ named it as (H_i, K_i), where H_i is an element of $(E_{11}E_{21}, E_{1i}E_{2i})$ and K_i is an element of $(F_{11}F_{21}, F_{1i}F_{2i})$ are subsets of Q_{acc3}. An infinite run of M_3 on σ is $\rho_\omega = \rho(0)\rho(1)...\rho(n)$ is said to be recognized by M_3 with probability p if there is some $i{\in}\ \{0, 1, ..n\}$ for which run ρ_ω contains only a finite number of H_i accepting and an infinite number of K_i accepting density matrices. Thus, $\sigma \in L_{\omega_3}(M_3)$ i.e. $L_{\omega_1}(M_1) \subset L_{\omega_3}(M_3)$. It can be easily checked that $L_{\omega_2}(M_2) \subset L_{\omega_3}(M_3)$ i.e. $L_{\omega_1}(M_1) \bigcup L_{\omega_2}(M_2) \subset L_{\omega_3}(M_3)$. □

Theorem 6.5.3. *The class of languages recognized by QRA is closed under complementation.*

Proof. Let $\overline{M_1} = (Q_1, \Sigma, \delta_1, q_{01}, \overline{Q_{acc1}})$ be a compliment QRA of M_1, where $\overline{Q_{acc1}} \subseteq Q_1$ but $\overline{Q_{acc1}} \notin Q_{acc1}$. Let us assume there is a ω-word σ not recognized by M_1 QRA i.e. an infinite run ρ_ω of M_1 does not contain an accepting pair of density matrices belong to Q_{acc1}. But, it must belongs to $\overline{Q_{acc1}}$ and recognized by $\overline{M_1}$. Therefore, we can say that $L_{\omega_1}(\overline{M_1}) = \Sigma^\omega / L_{\omega_1}(M_1)$. □

Theorem 6.5.4. *The class of languages recognized by QRA is closed under intersection.*

Proof. As a consequence of Theorem 6.5.2 and 6.5.3 and De-Morgan's law, it can be proved that the intersection of ω-languages recognized by M_1 and M_2 QRA is

$$L_{\omega_1}(M_1) \bigcap L_{\omega_2}(M_2) = (\Sigma^\omega / (\Sigma^\omega / L_{\omega_1}(M_1)) \bigcup (\Sigma^\omega / L_{\omega_2}(M_2)))$$

The class of language recognized by QSA is closed under union, intersection, and complementation. Quantum Streett acceptance condition is a negation of quantum Rabin acceptance condition. Therefore, it can be easily checked the closure property of QSA. □

Chapter 7

Linear Temporal Logic and Quantum Finite Automata

Linear temporal logic is a widely used method for verification of model checking and expressing the system specifications. The relationship between the theory of automata and logic had a great influence in computer science. Investigation of the relationship between quantum finite automata and linear temporal logic is a natural goal. In this chapter, we present a construction of quantum finite-automata on finite words from linear-time temporal logic formulas. Further, the relation between quantum finite automata and linear temporal logic is explored in terms of language recognition and acceptance probability. We have shown that the class of languages accepted by quantum finite automata are definable in LTL, except for measure-once one-way quantum finite automata.

7.1 Introduction

In the sixties, Büchi [127] and Elgot [128] indicated that monadic second-order logic (MSOL) and classical finite automata are equivalent. Further, the research by Büchi [119], Mchnaughton [121] and Rabin [122] showed the equivalence between MSOL and finite automata on finite words and trees. In 1977, Pnueli [129] introduced one of most widely used temporal logic (linear temporal logic) for the system specification. Temporal logic can be classified into linear and branch time logic. Mandal et al. [130] investigated a ternary logic function in connection with projection operations. It has been shown that proposed method for logic synthesis of ternary quantum circuits has a better cost compared to existing methods. In linear time logic, formulas are read over linear sequences of actions corresponding to runs of the system, whereas in branch time logic, formulas are read over computational trees. Here, we concentrated on linear time.
Model checking is a widespread technique for verifying temporal properties of reactive programs. There are several ways to develop the theory of model checking, a particularly attractive one being through the construction of automata from temporal logic formulas. Linear temporal logic (LTL) offers two important advantages. First, decision procedures for temporal logic are of elementary complexity. Second, writing temporal logic formals is easy and simpler. Till now, the research has been focused on the relationship between the class of languages recognized by MO-1QFA; MM-1QFA; LQFA; and the class of languages defined by first-order (FO) logic and

modular logic [124].
The concept of linear temporal logic uses the quantum automata theory as a unifying paradigm for program synthesis, verification, and specification. The program synthesis and specification are the essential elements to describe the computations, which are treated as input strings belong to some input alphabet Σ. The automata-theoretic perspective considers the specifications as relationships between languages and the relationships between programs. The queries about specifications can be easily lessen to queries about quantum automata after translation. Hence, the queries about correctness of programs and satisfiability of specifications can be lessen to containment and nonemtpiness queries of automata. The logical explanation of the quantum computational models performance has an impact on quantum complexity. In this chapter, we have focused on linear temporal logic which is equivalent to first-order logic in expressive power. LTL has a variable free compact syntax, intuitive semantics and it can be transformed into automata more efficiently than FO formulas [131]. Further, we have characterized the class of languages accepted by MO-1QFA, MM-1QFA, and LQFA using LTL formulas.

7.2 Preliminaries

In this Section, some basic syntax and semantics of LTL formulas are given.

7.2.1 LTL and classical automata

Originally, LTL was introduced as a specification language for concurrent programs over infinite traces. We described characterizations over finite words of LTL languages. It is identified that LTL languages cannot count modulo n for any n>1. Lenore [132] has shown the direct translations between LTL formulae and star-free regular expressions. Wilke [133] proved that the LTL formula can be written for the counter-free deterministic finite automata. The complexity of model checking and satisfiability is essential when it comes to applications of LTL to express formal specification and system verification. Sistla and Clarke [134] have shown that model checking and satisfiability of LTL are PSPACE-complete.

7.2.1.1 Syntax and semantics

The linear temporal logic can be defined with the use of a finite set of atomic propositions (AP), basic temporal and logical operators. The syntax of LTL formula is given by the following abstract syntax:

$$\phi = a \mid \phi_1 \wedge \phi_2 \mid \neg\phi \mid O\phi \mid \phi_1 \cup \phi_2 \tag{7.1}$$

where $a \in$ AP and ϕ, ϕ_1, ϕ_2 are LTL formulas. There are basic logical operators: negation ($\neg$), conjunction ($\wedge$), disjunction ($\vee$), implication ($\rightarrow$) and equivalence ($\leftrightarrow$). Temporal Operators are next (O), until ($\cup$). Some common abbreviations are used such as $O\phi$ means ϕ has to hold at the next state, $\phi_1 \cup \phi_2$ says ϕ_1 has to hold at least until ϕ_2 holds, $\diamond\phi$ stands for true$\cup\phi$ and it means that ϕ eventually has to hold before the last instant, $\square\phi$ stands for $\neg\diamond\neg\phi$ and it means that ϕ always hold till the last instant [135, 136].

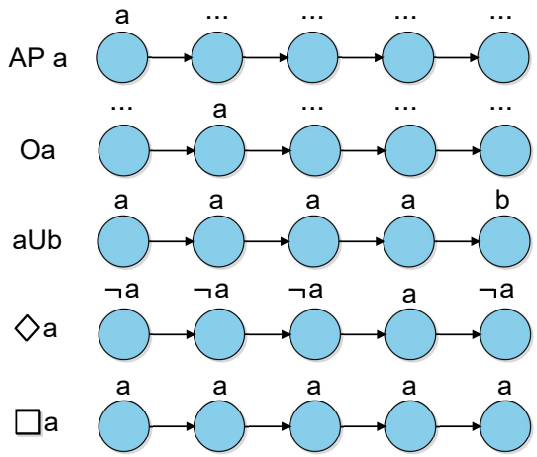


Figure 7.1: LTL formula evaluation

We interpret LTL formulas for finite strings s over finite traces. LTL formula is represented as finite string s over the alphabet of 2^{AP}. The length n of the string is denoted by $|s|$ and position $s(i, j)$ denotes the string s starts at positions i and terminating at positions j such that $0 \leq i \leq j \leq n$. Further, $s(i, *)$ represents the suffix $s(i, n)$. Consider LTL formula ϕ and a string s; we can say that LTL formula hold in s, denoted by $s \models$. Inductively, we can define

- $s \models a$ if $s(0) = a$.
- $s \models \neg\phi$ if not $s \models \phi$.
- $s \models X\phi$ if $|s| > 1$ and $s(1, *) = \phi$.
- $s \models \Diamond\phi$ if there is i with $0 < i < n$ such that $s(i, *) = \phi$.
- $s \models \phi_1 \cup \phi_2$ if there is i with $0 < i < n$ such that $s(j, *) = \phi_1$ for every $j \in \{1, 2, ...i - 1\}$ and $s(i, *) = \phi_2$.

Schützenberger [137] and McNaughton-Papert [138] proved the following equivalence results:

Theorem 7.2.1. *Let $L \subseteq \Sigma^+$ be a language, where Σ is an alphabet. Then, the following are correspondent:*

- L is definable in first-order,
- L is non-counting and regular,
- L can be defined by a star-free regular expression,
- The syntactic monoid of L is finite and aperiodic,
- It is recognized by a counter-free minimal DFA.

Theorem 7.2.2. *McNaughton [139] and Kamp [140] proved that the conditions "FO definable", "star-free", and "LTL-definable" for a language L are equivalent.*

7.3 MO-1QFA and linear temporal logic

MO-1QFA with bounded error can be designed for a subset of regular languages. Firstly, we study the linear temporal logic on finite trace and its connection with MO-1QFA.

Theorem 7.3.1. *If a language is accepted by MO-1QFA and it is defined by LTL formula, then it is regular and non-counting.*

Proof. If MO-1QFA recognizes a language L with bounded error, it can be recognized by one-way group finite state automata (1GFA). Let us assume that there is a language L which is even language and recognized by MO-1QFA. Correspondingly, minimal 1GFA has both accepting and rejecting states. As the automata accept counting language, it can go from accepting state to rejecting after reading a symbol. Now, we consider an even language $L_1 = a^{2n}$ which is recognized by MO-1QFA and 1GFA can be designed. For LTL formula, the property "p needs to be true on every even step", which is not possible. Hence, LTL cannot define properties which involve counting. It shows that LTL cannot express all regular languages. Therefore, there is a language that can be recognized by MO-1QFA and corresponding 1GFA can be designed but cannot be definable in LTL formula. □

7.4 MM-1QFA and linear temporal logic

In this section, the connection between MM-1QFA and LTL regarding accepting probabilities and language recognition has been studied. MM-1QFA can recognize all languages which are recognized by MO-1QFA, but such languages cannot be definable in LTL formula. Consider a language $L_1 = a^{2n}$ which consists words of multiple of 2, is recognized by MM-1QFA, but cannot be definable in LTL.

Theorem 7.4.1. *The language described by the LTL formula of the form* $(a \wedge O(\neg a \cup a))$ *can be accepted by MM-1QFA.*

Proof. The language explicated by the LTL formula in the given form is accepted by the MM-1QFA $(Q, \Sigma, \delta, q_0, Q_{acc}, Q_{rej})$, where $Q = \{q_0, q_1, q_2, q_a, q_r\}$, $\Sigma = \{a, b\}$, q_0 is an initial state, q_a is an accepting state and q_r is a rejecting state. The LTL formula evaluation and transition functions are defined as:

Table 7.1: Details of the transition functions

$V_{\#}\|q_0\rangle = \|q_0\rangle$	$V_a\|q_0\rangle = \|q_1\rangle$	$V_b\|q_0\rangle = \|q_r\rangle$	$V_{\$}\|q_0\rangle = \|q_r\rangle$
$V_{\#}\|q_1\rangle = \|q_1\rangle$	$V_a\|q_1\rangle = \|q_2\rangle$	$V_b\|q_1\rangle = \|q_1\rangle$	$V_{\$}\|q_1\rangle = \|q_r\rangle$
$V_{\#}\|q_2\rangle = \|q_2\rangle$	$V_a\|q_2\rangle = \|q_r\rangle$		$V_{\$}\|q_2\rangle = \|q_a\rangle$

We have defined all such values so that V_σ is unitary. Therefore, it can be easily checked that L_2 can be accepted with probability 1 by MM-1QFA and can be defined in LTL. There are certain languages accepted by MM-1QFA with probability less than 1, can we define LTL formula for such languages? The answer is yes, consider a language $L_2 = \{a^*b^*\}$ can be recognized by MM-1QFA with probability less than 1 and LTL formula is given below. □

Theorem 7.4.2. *The language described by the LTL formula of the form* $(a \vee (\neg b \cup \Box \neg a) \vee b)$ *can be accepted by MM-1QFA with a probability less than 1.*

Proof. The language explicated by the formula is $L_2 = a^*b^*$, which is recognized by MM-1QFA $(Q, \Sigma, \delta, q_0, Q_{acc}, Q_{rej})$, where $Q = \{q_0, q_1, q_a, q_r\}$, $\Sigma = \{a, b\}$, q_0 is an initial state, $Q_{acc} = \{q_a\}$ and $Q_{rej} = \{q_r\}$. The transition function δ is defined by

$$
\begin{aligned}
V_{\#} |q_0\rangle &= \frac{1}{\sqrt{2}} |q_0\rangle + \frac{1}{\sqrt{2}} |q_1\rangle , \\
V_a |q_0\rangle &= \frac{1}{\sqrt{2}} |q_0\rangle + \frac{1}{\sqrt{6}} |q_1\rangle + \frac{1}{\sqrt{3}} |q_a\rangle , \\
V_a |q_1\rangle &= -\frac{1}{\sqrt{2}} |q_0\rangle + \frac{1}{\sqrt{6}} |q_1\rangle + \frac{1}{\sqrt{3}} |q_a\rangle , \\
V_b |q_0\rangle &= |q_r\rangle , V_{\$} |q_0\rangle = |q_r\rangle , \\
V_b |q_1\rangle &= |q_1\rangle , V_{\$} |q_1\rangle = |q_a\rangle
\end{aligned}
$$

We can look at two cases. Consider a word $w \in L_2$, the computation starts with q_0 and proceed as follows:
On reading the left-end marker #, the transition function corresponding the input symbol is applied on q_0, resulting into a superposition of states with equal probability. On reading the symbol a, the unitary equation $V_a |q_0\rangle$ and $|q_1\rangle$ is applied and the superimposed state $\frac{1}{\sqrt{2}} |q_0\rangle + \frac{1}{\sqrt{2}} |q_1\rangle$ is changed into,

$$\frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}}\right) |q_0\rangle + \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{6}}\right) |q_1\rangle + \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{3}} + \frac{1}{\sqrt{3}}\right) |q_a\rangle ,$$

And computation continues to the next symbol due to the observation of non-halting state.

$$\frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{6}}\right) |q_1\rangle$$

On reading the second symbol a, the above state is changed into

$$\frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}+\frac{1}{\sqrt{6}}\right)\left(-\frac{1}{\sqrt{2}}\right) |q_0\rangle+\frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}+\frac{1}{\sqrt{6}}\right)\left(\frac{1}{\sqrt{6}}\right) |q_1\rangle+\frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}+\frac{1}{\sqrt{6}}\right)\left(\frac{1}{\sqrt{3}}\right) |q_a\rangle$$

Moreover, the probability of acceptance is $\left(\frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}+\frac{1}{\sqrt{6}}\right)\left(\frac{1}{\sqrt{3}}\right)\right)^2$. The process of computation proceeds with non-halting states:

$$\frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{6}}\right)\left(-\frac{1}{\sqrt{2}}\right) |q_0\rangle + \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{6}}\right)\left(\frac{1}{\sqrt{6}}\right) |q_1\rangle$$

On reading the symbol b, the above state changed into

$$\frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{6}}\right)\left(-\frac{1}{\sqrt{2}}\right) |q_r\rangle + \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{6}}\right)\left(\frac{1}{\sqrt{6}}\right) |q_1\rangle$$

An automaton rejects the input word with probability $\left(\frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}+\frac{1}{\sqrt{6}}\right)\left(-\frac{1}{\sqrt{2}}\right)\right)^2$ and with probability $\left(\frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}} + \frac{1}{\sqrt{6}}\right)\left(\frac{1}{\sqrt{6}}\right)\right)^2$, it remains in superposition q_1.

The total acceptance probability can be calculated as the summation of all accepting probabilities in the above steps. On reading \$, the automata change its states into q_a and recognize the input string with probability less than 1.
In the same way, we can show that if the input word $w \notin L_2$, then the MM-1QFA will reject it with probability less than 1. □

LTL is equivalent to FOL in expressive power, so strictly less expressive than regular expressions. The languages defined by LTL are a subset of regular languages; MM-1QFA can recognize languages definable in LTL formula. Till now, there are several extensions of LTL with finite automata, regular expressions and grammar operators have been proposed. Each of them has its drawbacks. To overcome such difficulties, Giacomo et al. [135] proposed a temporal logic by combining LTL and regular expressions, named as the linear dynamic logic on finite traces (LDL). It is known that MM-1QFA cannot be designed for $\{(a,b)^*b\}$, but the language can be defined by LTL formula as $(\Box\Diamond a)$.

7.5 Latvian quantum finite automata and linear temporal logic

It is a generalized version of MO-1QFA and recognizes a proper subset of MM-1QFA languages. In this section, we have determined the connection between the class of languages recognized by LQFA and LTL.

Theorem 7.5.1. *The language described by the LTL formula of form $(\Diamond a)$ can be accepted by LQFA.*

Proof. The LTL formula describes the language $L_3 = (a \mid b)^* a (a \mid b)^*$, which is recognized by LQFA. It is known that languages recognized by LQFA, also accepted by MM-1QFA.
LQFA $\{Q, \Sigma, \{A_\sigma\}, \{P_\sigma\}, q_0, Q_{acc}\}$ is designed for the language L_3, where $Q = \{q_0, q_1\}$, $\Sigma = \{a, b\}$, $Q_{acc} = \{q_1\}$, the projective measurement matrix is $P_\# = P_a = P_b = E_0 = E_1, E_i = \{span : |q_i\rangle\}$ and the transition functions are given as:

$$V_\# = V_\$ = V_b = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, V_a = \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$$

It can be easily checked that any input string $w \in L_3$, can be accepted with probability 1/2. Therefore, the class of languages accepted by LQFA with probability less than 1, are definable in LTL. □

LQFA recognized languages are closed under union, inverse homeomorphisms, word quotient, and complement. It is identified that the language $L = a\Sigma^* b\Sigma^*$ cannot be recognized by LQFA, but it can be recognized by MM-1QFA. It's LTL formula is defined as $(a \wedge \Diamond b)$.

Chapter 8

Modeling of RNA secondary structures using 2QFA

The representation of ribonucleic acid (RNA) and deoxyribonucleic acid (DNA) structures using the theory of automata had a great influence in computer science. Investigation of the relationship between QFA and RNA structures is a natural goal. Two-way quantum finite automata (2QFA) is more dominant than its classical model in language recognition. Motivated by the concept of quantum finite automata, we have modeled RNA secondary structure loops hairpin loop, internal loop, and double helix loop using two-way quantum finite automata. The major benefit of this approach is that these structure loops can be parsed in linear time. In contrast, two-way deterministic finite automata (2DFA) cannot represent RNA secondary structure loops and two-way probabilistic finite automata (2PFA) can parse these sequences in an exponential time. To the best of author's knowledge, this is the first attempt to represent RNA secondary structure loops using two-way quantum finite automata.

8.1 Introduction

Bioinformatics [141] is a field of applied science in which computational and mathematical models are applied to biology. Nowadays, a major concern of bioinformatics is the analysis and representation of deoxyribonucleic acid (DNA) and ribonucleic acid (RNA). The grammatical formalism of biological sequences, such as DNA, RNA, and proteins, can be used to solve bioinformatics problems efficiently, such as multiple alignment calculation, classification, prediction of biological sequences with their primary and secondary structures. The RNA structure folds around itself to do base pairing, which leads to the formation of various secondary structure motifs (loops). In RNA secondary structure formation, some of the base pairs contain exact complement base pairs, whereas other base pairs will not perform pairing because of the lack of exact complement pairs, thereby tending to form loops. Loops can be of various types, such as the hairpin loop, bulge loop, internal loop, double helix, and pseudoknot, depending on their shape. Moreover, RNA and DNA are made by monomers known as nucleotides. Each nucleotide consists of a pentose carbon sugar, a phosphate group, and a nitrogenous base. If the sugar is ribose, then the polymer is RNA. If the sugar is deoxyribose, then the polymer is DNA. Besides, RNA is a single-stranded structure, whereas DNA is double-stranded helical structure.

RNA structure is classified into primary, secondary, and tertiary structures. RNA is single-stranded structure i.e. the sequence of the bases (a, c, g, u) are in linear order in its primary structure form [142], where purines are classified into adenine (a) and guanine (g), while pyrimidine are classified into cytosine (c) and uracil (u). At least three bases are required to form loop [143]. Watson-Crick pairing mostly occurs whereas Wobble–pair rarely occurs in RNA secondary structure. The tertiary structure involves the orientation of these secondary structure motifs for each other. It forms certain structures such as pseudoknot which have many functional features. RNA tends to form single-stranded 3D structures due to the presence of an extra 2'-hydroxyl group present in the ribose part of RNA .

The field of quantum computation and information processing have subsequently made a significant impact on the academic and research community alike. Recently, various applications of QFA models to interactive proof system (Nishimura et al., 2009, Zheng et al., 2015) [61,62], debate system (Yakaryilmaz et al., 2016) [63]; temporal logical theory (Berzina, 2010) [124]; automata in molecular biology by quantum mechanics (Khrennikov and Yurova, 2017) [144] have been investigated. Furthermore, various generalizations of quantum computational models such as quantum computation over infinite words (Giannakis et al. 2015) [145]; QFA models on promise problems (Zheng et at. 2017) [57], multihead one-way quantum finite automata (1QFA(k)) (Ganguly et al. 2016) [83]; Two-tape QFA models (Ganguly et al. 2016) [84] has been introduced. Recently, (Pan et al. 2017) [146] quantified the multiqubit states in Grover's searching algorithm by using the geometric measure of entanglement (GME). (Li et al. 2017) [147] studied the phase estimation by using distributed semi-computing model. They have proposed a semi-distributed algorithm for phase estimation which has achieved exponential acceleration and performs better than the classical algorithm.

In classical automata theory, Freivalds [148] shown that a non-regular language $L = \{a^m b^m \mid m \geq 1\}$ can be recognized by 2PFA with arbitrarily small error. Later, Dwork and Stockmeyer [149] proved that 2PFA takes an exponential time to recognize the same language. It has been shown that non-regular language can be recognized by 2PFA with error probability below $\frac{1}{2}$, then there exists a constant $b > 0$, such that for the infinite number of input n, the expected runtime of 2PFA must exceed 2^{n^b}, where n is the length of an input. It is known that 2DFA can recognize only regular languages [150, 151].

The research has consistently grown in the field of quantum finite automata theory. Kondacs and Watrous [14] shown that a non-regular language $L = \{a^m b^m \mid m \geq 1\}$ can be recognized by 2QFA with one-sided error in linear time. In this chapter, we have shown that it is possible to define non-context free languages by using 2QFA. This chapter is concerned with representing the secondary structure loops of RNA using 2QFA. But, it cannot be represented by 2DFA, and two-way probabilistic finite automata can parse these sequences in exponential time. Thus, it follows that 2QFA is more powerful than its classical variants in terms of language recognition.

8.1.1 Prior work

Various representations of DNA and RNA sequences using formal grammar and automata have been found in literature. For example, Kalra and Kumar [152] represented DNA and RNA biological sequences, such as inverted repeat, pseudoknot

and tandem repeat using state grammar and deep pushdown automata. Sung [153] represented the secondary structure loops of RNA, such as the hairpin loop, internal loop, bulge loop, and double helix, using context-sensitive grammar. Various forms of context-free grammar are also used to represent RNA sequences [154,155].
Besides, cross-interaction grammar was used by Rivas and Eddy [156] to represent the secondary structure loops of RNA, including pseudoknots. Searls [157] used indexed grammar to represent DNA and RNA sequences, such as tandem repeat, inverted repeat, and pseudoknot. Searls [158] also represented DNA sequences using string variable grammar. Mizoguchi et al. [159] used stochastic multiple context-free grammar to represent various classes of pseudoknots. Parallel communicating grammar systems were used by Cai et al. [160] to represent the pseudoknot structure of RNA.
Various researchers have represented the structures of RNA and DNA using concept of grammar and automata theory. Kuppusamy et al. [161] introduced the concept of matrix-insertion deletion grammar and represented the commonly found structures of DNA and RNA, which occur at intramolecular level such as pseudoknot, hairpin, stem loop, cloverleaf, dumbbell, and attenuator. Kuppusamy et al. [162] also represented the DNA and RNA biomolecules structures which occur at an intermolecular level such as nick language, double-strand language and linear hybridization (ligated) languages using matrix insertion-deletion grammar. Further, Kuppusamy and Mahendran [163] extended their work of bio-molecular representation using matrix-insertion deletion grammar and represented the RNA and DNA bio-molecular structures of the intermolecular level, intramolecular level and RNA secondary level using matrix-insertion deletion grammar.
Anderson et al. [164,165] represented the RNA structure using stochastic context-free grammar. Rothemund [166] and Cavaliere et al. [167] represented the action of a restriction enzyme in DNA using a Turing machine and pushdown automata, respectively. Furthermore, Krasinski et al. [168] extended the Cavaliere et al. [167] approach and represented the action of a restriction enzyme in DNA using circular pushdown automata in which stack and input tape are on the same circular strand. Recently, Khrennikov and Yurova [169] have proposed a model of protein behavior by using theory of automata and also explored the similarities between the modeling of the behavior of proteins and quantum systems.

8.2 Modeling of RNA secondary structure loops using 2QFA

In this section, we model the RNA secondary loops such as hairpin loop, internal loop and double helix using two-way quantum finite automata (2QFA).

8.2.1 Hairpin loop

A hairpin loop is formed when RNA stands folds to pair with another section of the same strand and ends to form the unpaired loop [170]. Fig 8.1 describes the structure of hairpin loop. The structure of hairpin loop is as follows i.e. sequence 1 followed by hairpin loop followed by paired sequence 1 [171].

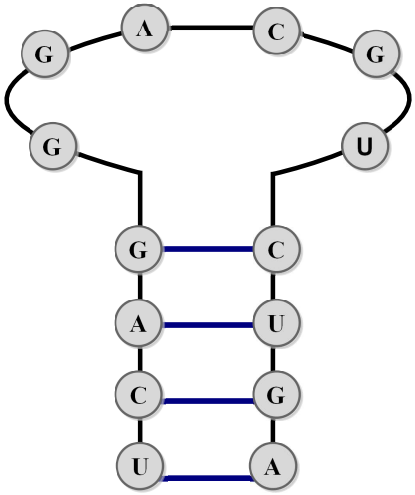


Figure 8.1: Representation of Hairpin loop structure

The quantum finite automaton for language $L_1 = \{u_1^p v^m u_2^p \mid p > 1, m \geq 3\}$ is as follows:

Table 8.1: Details of transition function and tape head function

$V_{\#}\|q_0\rangle = \|q_0\rangle$	$V_v\|q_0\rangle = \|q_1\rangle$	$V_{u_1}\|q_1\rangle = \|q_2\rangle$
$V_{\#}\|q_1\rangle = \|q_5\rangle$	$V_v\|q_2\rangle = \|q_2\rangle$	$V_{u_1}\|q_2\rangle = \|q_5\rangle$
$V_{\$}\|q_2\rangle = \|q_5\rangle$	$V_v\|q_3\rangle = \|q_4\rangle$	$V_{u_1}\|q_4\rangle = \|q_5\rangle$
$V_{u_1}\|q_0\rangle = \|q_0\rangle$	$V_v\|q_4\rangle = \|q_5\rangle$	$V_{u_2}\|q_0\rangle = \|q_5\rangle$
		$V_{u_2}\|q_4\rangle = \|q_4\rangle$
$V_{\$}\|q_4\rangle = \frac{1}{\sqrt{n}}\sum_{i=1}^{n}\|p_{i,0}\rangle$		
$V_{u_2}\|p_{i,0}\rangle = \|p_{i,n-i+1}\rangle$ for $1 \leq i \leq n$		
$V_{u_2}\|p_{i,j}\rangle = \|p_{i,j-1}\rangle$ for $1 \leq i \leq n, 1 \leq j \leq n-i+1$		
$V_v\|p_{i,0}\rangle = \|s_{i,0}\rangle, V_b\|s_{i,0}\rangle = \|r_{i,0}\rangle$, If $v \neq 3$		
$V_v\|r_{i,0}\rangle = \|t_{i,0}\rangle$ for $1 \leq i \leq N$, $V_{u_1}\|s_{i,0}\rangle = \|K_{i,0}\rangle$		
$V_{u_1}\|w_{i,0}\rangle = \|w_{i,i}\rangle$ for $1 \leq i \leq n$, $V_{u_1}\|K_{i,0}\rangle = \|Y_{i,i}\rangle$		
$V_{u_1}\|w_{i,j}\rangle = \|w_{i,j-1}\rangle$ for $1 \leq i \leq n, 1 \leq j \leq n-i+1$		
$V_{u_1}\|Y_{i,j}\rangle = \|Y_{i,j-1}\rangle$ for $1 \leq i \leq n, 1 \leq j \leq n-i+1$		
$V_{\#}\|w_{i,0}\rangle = \frac{1}{\sqrt{n}}\sum_{l=1}^{n} e^{\frac{2\pi\iota}{n}il}\|f_l\rangle$ for $1 \leq i \leq n$		
$V_{\#}\|Y_{i,0}\rangle = \frac{1}{\sqrt{n}}\sum_{l=1}^{n} e^{\frac{2\pi\iota}{n}il}\|R_l\rangle$ for $1 \leq i \leq n$		
Head functions		
$D(q_0) = \rightarrow$, $D(q_1) = \leftarrow$, $D(q_2) = \rightarrow$,		
$D(q_3) = \leftarrow$, $D(q_4) = \rightarrow$, $D(q_5) = \uparrow$		

Theorem 8.2.1. *For a language $L_1 = \{u_1^p v^m u_2^p \mid p > 1, m \geq 3\}$ and for an arbitrary fixed positive integer $n \geq 2$ such that $n \in \mathbb{N}$, there exists a 2QFA such that for $x \in L_1$, it accepts x with probability 1 and rejects $x \notin L_1$ with probability at least $1 - \frac{1}{n}$.*

Proof. We construct 2QFA model for the language:

$$L_1 = \{u_1^p v^m u_2^p \mid p > 1, m \geq 3\}$$

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5\} \cup \{p_{i,j}, w_{i,j} \mid 1 \leq i \leq n, 1 \leq j \leq max(i, n-i+1)\} \cup \{p_{i,0}, w_{i,0}, r_{i,0}, s_{i,0}, t_{i,0}, K_{i,0}, Y_{i,0} \mid 1 \leq i \leq n\} \cup \{f_i, R_i \mid 1 \leq i \leq n\}, \Sigma \in \{u_1, v, u_2\}, Q_{acc} = \{F_n\}, Q_{rej} = \{q_5\} \cup \{f_i \mid 1 \leq i < n\} \cup \{R_i \mid 1 \leq i \leq n\}$. The state transition functions and head functions are given in Table 8.1 using a definition 2.3.12. Each V_σ take the values as shown in Table 8.1 and also for each σ, the vectors formed V_σ are unitary on Hilbert space $L_2(Q)$ and $D : Q \to \{\leftarrow, \uparrow, \rightarrow\}$ represents head functions.
The process of designing 2QFA for language consists of three phases. In the first phase, if the input is not of the form $u_1^+ v^+ u_2^+$, where $^+$ represents the Kleene plus, then the computation will be rejected. Therefore, the first phase traverses the input string (i.e. scanning through each symbol of input string). Fig 8.2 shows the detailed state transition diagram of L_1. At the start of the second phase; the state q_4 reads the right-end marker \$. Furthermore, the computation is split in to the n different paths (state transformations occur in parallel). Each path possesses equal amplitude $\frac{1}{\sqrt{n}}$. Along the n different paths, each path moves deterministically to the left-end marker #. On reading symbol u_2, along the i^{th} path, tape W head remains stationary for $n-i+1$ times. On reading symbol u_1, head remains stationary for i times. On reading symbol v, it must satisfy the condition ($m \geq 3$); otherwise, it leads to rejecting the input string at the end. If the number of u_1's and u_2's are equal in an input string, then different paths will reach the left-end marker # at the same time. Finally, in the third phase on reading #, each computation path comes into n-way Quantum Fourier transform (QFT). It produces either one accepting state or rejecting state. Finally, 2QFA accepts the string $u_1^2 v^3 u_2^2$ with probability 1. If the input string is not in the desired form, then all computation paths reaches the left-end marker at a different time and there is no cancellation of rejecting states. Thus, for all n computation paths, the conditional probability tha observation results in acceptance is $\frac{1}{n}$, such that some of the paths come to halt. Therefore, the input string is said to be accepted with probability $\frac{1}{n}$. Correspondingly, the input string is said to be rejected with probability $1 - \frac{1}{n}$. □

8.2.2 Internal loop

Internal loops are formed when there is no matching pair on both sides of the double-stranded RNA i.e. formation of the hairpin loop on both sides of double-stranded RNA [172]. Fig 8.3 describes the structure of an internal loop. The structure of an internal loop is as follows i.e. Sequence 1 followed by hairpin loop followed by sequence 2, sequence, paired sequence 2, hairpin loop and paired sequence 1 [171]. The language generated by an internal loop is

$$L_2 = \{u_1^r u_2^m u_3^q v^z w_1^q w_2^m w_3^r \mid r, q > 1, m \geq 3, z \leq 2\}$$

Here, $(u_1, u_2, u_3, v, w_1, w_2, w_3) \in \{g, c, a, u\}$ The quantum finite automaton for language L_2 is as follows:

Theorem 8.2.2. *For a language* $L_2 = \{u_1^r u_2^m u_3^q v^z w_1^q \ w_2^m w_3^r \mid r, q > 1, m \geq 3, z \leq 2$ *and* $r, q, m \in \mathbb{N}_{>0}, z \in N_{\geq 0}\}$ *and for arbitrary computational paths n, M,* $P \in \mathbb{N}$,

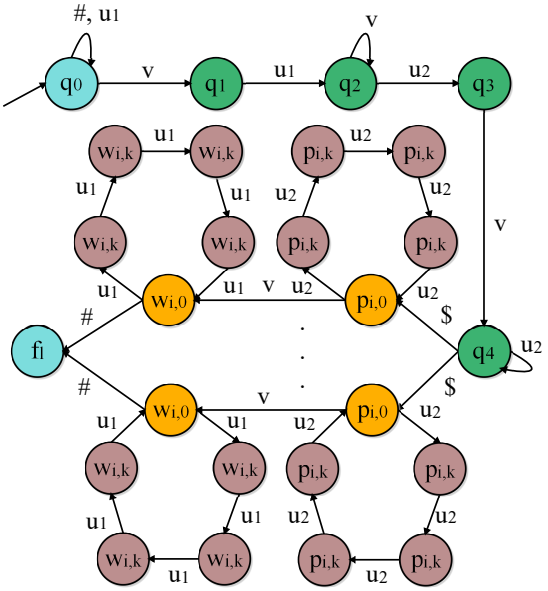


Figure 8.2: State diagram of Hairpin loop for $L_1 = u_1^p v^m u_2^p$

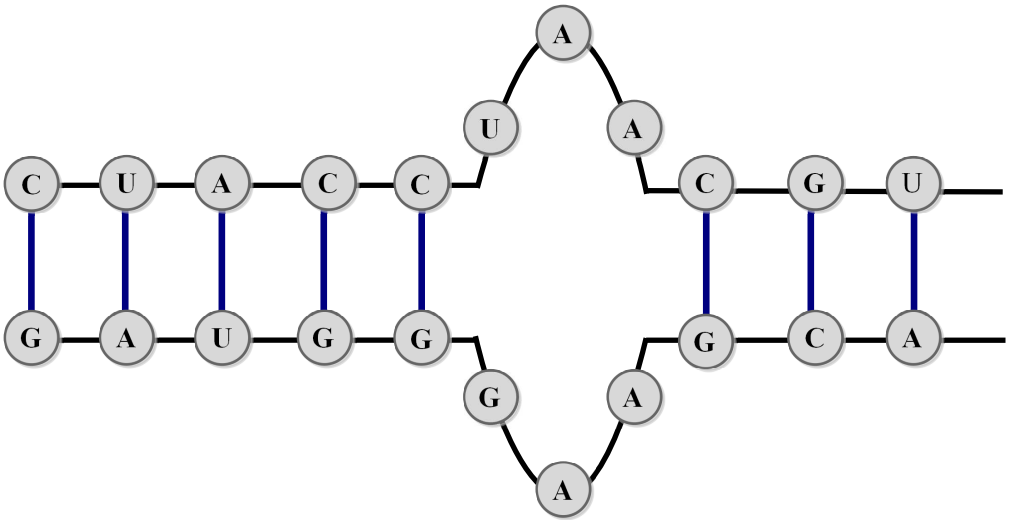


Figure 8.3: Illustration of Internal loop structure

there exists a 2QFA such that for $x \in L_2$*, it accepts* x *with bounded error* ϵ *and rejects* $x \notin L_2$ *with probability at least* $1 - \dfrac{1}{nMP}$.

Proof. We construct 2QFA model for $L_2 = \{u_1^r u_2^m u_3^q \; v^z w_1^q w_2^m w_3^r \mid r, q > 1, m \geq 3, z \leq 2\}$.
Let $M_{2QFA} = (Q, \Sigma, \delta, q_0, Q_{acc}, Q_{rej})$ be a 2QFA $Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9, q_{10}, q_{11}, q_{12}, q_{13}, q_{14}\} \cup \{p_{i,0}, r_{i,0}, u_{i,0}, s_{i,0}, t_{s,0}, v_{s,0} \mid 1 \leq i \leq n, 1 \leq s \leq n\} \cup \{q_{ij,0}, r_{ij,0}, g_{ij,0}, u_{ij,0}, v_{ij,0} \mid 1 \leq i \leq N, 1 \leq j \leq M\} \cup \{q_{ijl,0}, v_{ijl,0}, g_{ijl,0} \mid 1 \leq l \leq n, 1 \leq j \leq M, 1 \leq l \leq P\} \cup \{s_{i,k}, p_{i,k}, v_{ij,k}, g_{ijl,k} \mid 1 \leq i \leq n, 1 \leq j \leq M, 1 \leq l \leq max(j, M-j+1)\}, \Sigma \in \{u_1, u_2, u_3, v, w_1, w_2, w_3\}, Q_{acc} = \{y_n\}$ and $Q_{rej} = \{q_{14}\}$. The state transition functions and head functions are given in Table 2 using equation (10). All vectors formed V_σ are unitary on Hilbert space $L_2(Q)$ and D represents head functions.
The process of designing a 2QFA for language L_2 consists of seven phases namely

traverse the input (i.e. scanning through the each symbol of input string), split (state transformations occur in parallel) of computation in to n different paths for having superposition of states on reading the right-marker \$ (i.e. quantum system is in more than one state at a time), then n different paths split in to M different paths while reading leftmost w_3, further again M different paths split into P different paths with equal probability on reading leftmost w_2, perform P-way QFT at the end of u_3, then perform M-way QFT at the end of u_2 and similarly perform n-way QFT to yield the result at the end.

Table 8.2: Details of transition function and tape head function

$V_{\#}\|q_0\rangle = \|q_0\rangle$	$V_{u_3}\|q_2\rangle = \|q_3\rangle$	$V_{w_2}\|q_9\rangle = \|q_{10}\rangle$
$V_{\$}\|q_0\rangle = \|q_{14}\rangle$	$V_{u_3}\|q_4\rangle = \|q_4\rangle$	$V_{w_2}\|q_{11}\rangle = \|q_{11}\rangle$
$V_{\$}\|q_{11}\rangle = \|q_{14}\rangle$	$V_{u_3}\|q_5\rangle = \|q_6\rangle$	$V_{\$}\|q_{12}\rangle = \|q_{13}\rangle$
$V_{u_1}\|q_0\rangle = \|q_0\rangle$	$V_{u_3}\|q_8\rangle = \|q_9\rangle$	$V_{w_2}\|q_0\rangle = \|q_{14}\rangle$
$V_{u_1}\|q_1\rangle = \|q_2\rangle$	$V_{u_3}\|q_0\rangle = \|q_{14}\rangle$	$V_{w_2}\|q_2\rangle = \|q_{14}\rangle$
$V_{u_2}\|q_0\rangle = \|q_1\rangle$	$V_{w_1}\|q_2\rangle = \|q_{14}\rangle$	$V_{w_2}\|q_4\rangle = \|q_{14}\rangle$
$V_{u_2}\|q_1\rangle = \|q_2\rangle$	$V_{w_1}\|q_4\rangle = \|q_8\rangle$	$V_{w_3}\|q_{11}\rangle = \|q_{12}\rangle$
$V_{u_2}\|q_3\rangle = \|q_4\rangle$	$V_{w_1}\|q_6\rangle = \|q_7\rangle$	$V_{w_3}\|q_{13}\rangle = \|q_{13}\rangle$
$V_{u_2}\|q_0\rangle = \|q_1\rangle$	$V_{w_1}\|q_9\rangle = \|q_9\rangle$	$V_{w_3}\|q_0\rangle = \|q_{14}\rangle$
$V_v\|q_4\rangle = \|q_5\rangle$	$V_{w_1}\|q_{10}\rangle = \|q_{11}\rangle$	$V_{w_3}\|q_2\rangle = \|q_{14}\rangle$
$V_v\|q_6\rangle = \|q_6\rangle$	$V_{w_1}\|q_0\rangle = \|q_{14}\rangle$	$V_{w_3}\|q_4\rangle = \|q_{14}\rangle$
$V_v\|q_2\rangle = \|q_{14}\rangle$	$V_{u_2}\|q_2\rangle = \|q_2\rangle$	$V_{w_3}\|q_9\rangle = \|q_{11}\rangle$
Head functions		
$D(q_0) = \rightarrow$, $D(q_1) = \rightarrow$, $D(q_2) = \rightarrow$, $D(q_3) = \leftarrow$		
$D(q_4) = \rightarrow$, $D(q_5) = \leftarrow$, $D(q_6) = \rightarrow$, $D(q_7) = \leftarrow$		
$D(q_8) = \leftarrow$, $D(q_9) = \rightarrow$, $D(q_{10}) = \leftarrow$, $D(q_{11}) = \rightarrow$		
$D(q_{12}) = \leftarrow$, $D(q_{13}) = \rightarrow$, $D(q_{14}) = \uparrow$		

In the first phase, if the input string is not of the form $u_1^+u_2^+u_3^+v^+w_1^+w_2^+w_3^+$ or $u_1^+u_2^+u_3^+w_1^+w_2^+w_3^+$, where $^+$ represents the Kleene plus, then the computation will be rejected. In Table 8.2, q_{13} reads the right-end marker \$ and the computation splits in to n different paths with equal probability $\frac{1}{\sqrt{n}}$. Along all n paths, it checks whether the number of w_3's is greater than 1 or not. Further, we can define the remaining transition functions such that V_σ must satisfy the unitary property. Fig 8.4 shows the detailed state transition diagram of L_2. In the third phase, upon reading the leftmost w_2, n paths splits in to M different paths with equal probability $\frac{1}{\sqrt{M}}$ and checks whether the number of w_2's satisfy the condition or not. Furthermore, in the fourth phase, M paths split into P different paths with equal probability $\frac{1}{\sqrt{P}}$ and checks whether the number of w_1's is greater than 1 or not. In fifth phase, all P different paths read v, u_3 or both. On reading the v, it must satisfy the condition ($z \leq 2$) and all paths move towards left. On reading the u_3's, each computation path enters P-way QFT. In the sixth, each computation path enters M-way QFT upon reading the left-most u_2. In the last phase, on reading the left-marker #, each computation path enters N-way QFT to yield the result. It results in a single accepting state and other rejecting states. If all n^{th} paths read the left-end marker # at the same time, then the input is accepted with probability 1. □

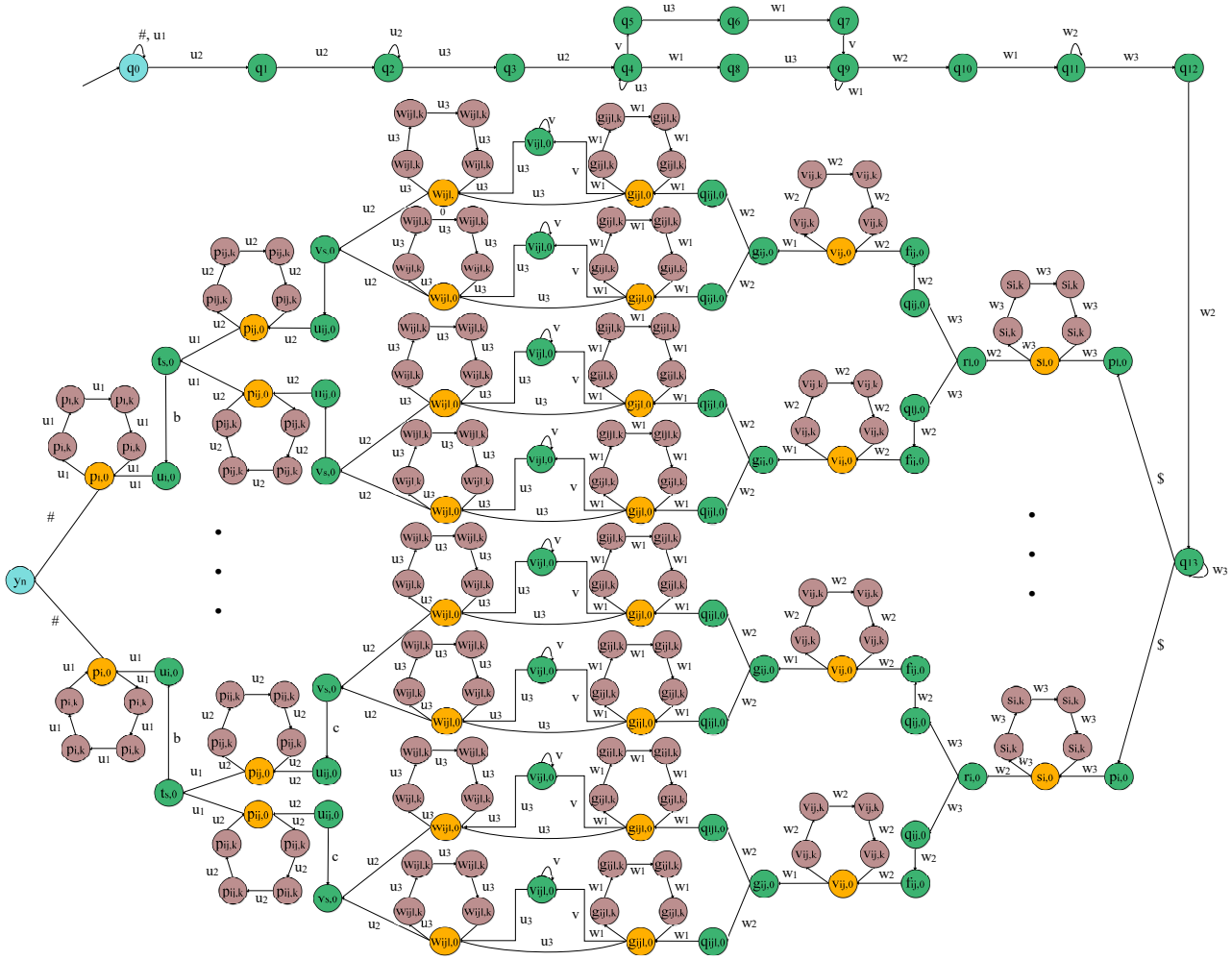


Figure 8.4: State transition diagram of Internal loop for $L_2 = u_1^r u_2^m u_3^q v^z w_1^q w_2^m w_3^r$

8.2.3 Double helix

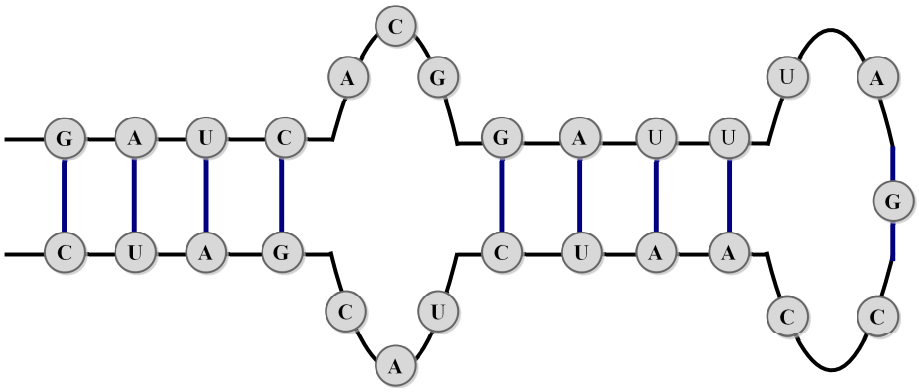


Figure 8.5: Illustration of Double helix loop structure

Double helix loops are formed when there is no matching pair on both sides of the double-stranded RNA and also ends in a hairpin loop [171]. If in the language $\{L_2 = u_1^r u_2^m u_3^q v^z w_1^q w_2^m w_3^r \mid r, q > 1, m \geq 3 \mid z \leq 2\}$ of internal loop, value of z is

($z \geq 3$), it leads to the formation of a double helix.
Fig 8.5 describes the structure of double helix loop. The language generated by grammar is of double helix loop i.e. Sequence 1 followed by loop segment 1 followed by sequence 2, followed by hairpin loop, followed by paired sequence 2 followed by loop segment 2 followed by paired sequence 1 [5]. The language generated by double helix is $L_3 = \{u_1^r u_2^m u_3^q v^z w_1^q \ w_2^m w_3^r \mid r, q > 1, m, z \geq 3\}$. Here, $(u_1, u_2, u_3, v, w_1, w_2, w_3) \in \{g, c, a, u\}$. The quantum finite automaton for language L_3 is as follows:

Theorem 8.2.3. *For a language $L_3 = \{u_1^r u_2^m u_3^q v^z w_1^q \ w_2^m w_3^r \mid r, q > 1, m, z \geq 3$ and $r, q, m, z \in \mathbb{N}_{>0}\}$ and for arbitrary computational paths $n, M, P \in \mathbb{N}$, there exists a 2QFA such that for $x \in L_3$, it accepts x with bounded error ϵ and otherwise rejects it ($x \notin L_3$).*

Proof. It functions similar to above languages instead its process of designing consists of seven phases namely traverse the input, split the computation into n different paths for having superposition of states on reading the right-marker \$ and process w_3's, then n paths split in to M different paths while reading leftmost w_3, further again M different paths split into P different paths with equal probability on reading leftmost w_2, perform P-way QFT at the end of u_3, then perform M-way QFT at the end of u_2 and similarly perform n-way QFT to yield the result at the end. If all n^{th} paths read the left-end marker # at the same time, then the input is accepted with probability 1. Therefore, it can be easily checked that input string $x \in L_3$, can be recognized by 2QFA with probability 1. □

Chapter 9

Quantized Quadratic Sieve Algorithm

In recent years, quantum computation is receiving much attention for its capability to solve difficult problems efficiently contrast to classical computers. Specifically, some well-known public-key cryptosystems depend on the difficulty of factoring large numbers, which takes a very long time. It is expected that the emergence of a quantum computer have the potential to break such cryptosystems by 2020 due to the discovery of powerful quantum algorithms (Shor's factoring, Grover's searching algorithm and many more). In this chapter, we have designed a quantum variant of classical factorization algorithm named "Quadratic Sieve". We have constructed the simulation framework of quantized quadratic sieve algorithm using high-level programming language Mathematica. Further, we have examined its simulation results on a classical computer to get a feel of the quantum system and proved that it is more efficient than its classical variants from computational complexity point of view.

9.1 Introduction

In the early 18th century, integer factorization was identified as fundamental problem and most important research area in computational number theory [173]. After the advent of digital computers, it has been applied in various applications related computing, cryptography and information security. Particularly, it is extremely related with the field of cryptography as finding factors of large integers is difficult for computers [174]. Over the last three decades, public key cryptosystems (Diffie-Hellman key exchange, the RSA cryptosystem, digital signature algorithm (DSA), and Elliptic curve cryptosystems) have become a crucial component of cyber security [175, 176]. Cotfas et al. [177] demonstrated the quantizations of non-equivalent coherent states and explored the localization features based on real Hermite polynomials. In this regard, security depends on the difficulty of a definite number of theoretic problems (integer factorization or the discrete log problem).
Since its beginning, Mathematicians have been trying to find factors of composite numbers in faster and efficient ways [178, 179]. Many algorithms have been devised for determining the prime factors of a given integer such as Trial division, Fermat's method, Elliptic curve method, Pollard's Rho method and fastest algorithms: Number sieve, Quadratic sieve and many more. These algorithms are differ in complexity

and superiority. Each method has become a stepping stones for the next method. Therefore, it is a very challenging task to design an efficient and effective algorithm for hard computational problem in complexity theory.
In 1981, Pomerance [180] introduced fastest factorization algorithm named Quadratic sieve. It is an improvement over Kraitchik's and Dixon's factorization method. In early 1990's, Quadratic sieve method was the most effective and efficient general purpose algorithm. It is the best choice method to find factors of numbers under 100 digits after general number sieve method [181]. The running time of number field and quadratic sieve depends on the size of a number (n) to factorize [182]. The fastest General number field sieve algorithm and second fastest quadratic sieve algorithm works in super-polynomial, but sub-exponential time using a classical computer. But, Shor's algorithm shown a great improvement over these algorithms, which can find factors in polynomial time using a quantum computer. The efficiency of quantum algorithms is based on the "Quantum Fourier transform" i.e. quantum variant of classical discrete Fourier transform, which can be constructed in polynomial time.
In this chapter, we have designed a quantum variant of quadratic sieve method by using quantum parallelism and entanglement. The most significant property *entanglement* separates the classical world from the quantum world. It is one of the most central topics in quantum information theory. Quantum entanglement is purely quantum mechanical correlation between two parts of the quantum system [105]. Further, we have examined its simulation on Mathematica tool and compared its complexity measure with its own classical variant and competitive Shor's factorization method.

9.2 Preliminaries and Definitions

Before we can discuss the classical quadratic sieve and design its quantum variant, some preliminaries and definitions are given in this section.

- *Residue class* [183]: Let $m \in N$. It is defined as the set of integers that are congruent to some integer a modulo m. For any $a \in$ Z, $[a]$ denotes the equivalence class to which a belongs, such that $[a]= \{a \in Z: n \equiv a \ mod \ m\}$.

- *Quadratic residue* [184]: A integer a is said to be quadratic residue *mod* m if for coprime integers m, a with $m >0$, the congruence has the solution such that $x^2 \equiv a \ mod \ m$. If it does not have a solution, then a is said to be a quadratic non-residue.

- *Legendre symbol* [184]: Let p be an odd prime and a an integer. Suppose that gcd(a, p)=1. Then, the Legendre symbol is defined as

$$\left(\frac{a}{p}\right) = \left\{ \begin{array}{ll} 0 & \text{if } a \equiv mod \ p \\ 1 & \text{if } a \text{ is quadratic residue } mod \ p \\ -1 & \text{if } a \text{ is quadratic non-residue } mod \ p \end{array} \right\}$$

  In quadratic sieve, we have small prime factors. Therefore, an integer $n \in N$ is called as B-smooth for $B \in R^+$, if it has no prime factors greater than B.

- *Exponent vector* [185]: Let p_i denotes the ith prime, such that

$$m = \prod_i p_i^{v_i}$$

The exponent vector is $v(m) = (v_1, v_2, ..., v_i)$. Each entry in *v(m)* represents the exponent on the i^{th} prime, where the integer 2 is the first prime, 3 the second and so forth.

- *Partial measurement* [186]: In formal way, it is defined as a state vector that is projected onto subspace spanned by the single qubit or quantum registers with probability equal to the square of the projection. In other words, a state vector projected on to the orthogonal subspace spanned by quantum registers with remaining probability. It does not covers the whole system, it just look at the part of the system. Consider a two quantum registers of size n and m qubits.

$$|\phi\rangle = \sum_{i=0}^{2^n-1} \sum_{j=0}^{2^m-1} c_{i,j} |i,j\rangle$$

The probability pr_j to measure j in the second register is $pr_j = \sum_{i=0}^{2^n-1} c_{i,j}^* c_{i,j}$. And, the new state after the measurement is given as

$$|\phi^{'}\rangle = \frac{1}{\sqrt{pr_j}} \sum_{i=0}^{2^n-1} c_{i,j} |i,j\rangle$$

Thus, the new state is given as (normalized) projection onto the respective subspace.

9.3 Quadratic Sieve Algorithm

Quadratic sieve is most extensively used and fastest algorithm to find factors of number less than 130 decimal digits. Since the discovery of quadratic sieve algorithm, it was used to factored 100 to 120 digits long number. Recently, in 1994, it has factored 129 decimal-digit RSA challenge number [187]. It is based on the classical congruences of squares method i.e. finding the squares whose difference is 0 modulo n, where $n \in N$ to be factored. Suppose, a integer n is an odd composite integer for which we need to find factors. First step in this algorithm is to form factor base. It is a set of primes S such that each element of S is less than smoothness bound B. Generally, selection of B is very crucial part of this algorithm to form factor base. Therefore, it helps in avoiding the unnecessary computations throughout the sieve process.
Sieve of Eratosthenes is factoring algorithm to generate a list of prime numbers less than smoothness bound B. The detailed algorithm of Sieve of Eratosthenes can be easily found in [188, 189]. Its efficiency is $O(\sqrt{n})$. It is found to be useful if someone needs lowest 10,000 primes for some computation. It is used to generate prime numbers in bulk. In case, if we have selected very small value of B, then there are very limited smooth number to find factors. If it is selected very large, then we need to get more numbers for having linear set. Therefore, the probability that a given

integer n is B-smooth is calculated as $p_r^{-p_r}$, where $p_r = \frac{ln(n)}{ln(B)}$. Though, its proof is given in [190]. Now, we have all the primes labeled as $p_2,\ p_3, ...$ Next step is to calculate Legendre symbol for each prime such that quadratic residue $\left(\frac{n}{p_i}\right) = 1$, where $i \in$ *2, 3, ...*, for each prime p_i.
Second step is to sieve through the polynomial values generated through the sequence $(x^2 - n)$, where $x = \lceil\sqrt{n},\ \lceil\sqrt{n}\rceil + 1, ...$ to have B-smooth values. We can define the sieve interval using optimum value of M such that $[\sqrt{n} + M,\ \sqrt{n} - M]$, where $M = e^{(ln\ n\ ln(ln\ (n)))^{3\sqrt{2}/4}}$. Now, we sieve through the each prime number in factor base such that $(x^2 - n) \equiv\ 0\ mod\ p$. It means prime p surely divides polynomial. Note that, it also sieve through the powers of prime in factor base unless it does not greater than B. It transform the division process into multiplication. At the end, it forms a list of numbers which indeed a factor by using primes. Now, we have prime factors for each sequence and it can be written as $m = \prod_{i=1}^{k} p_i^{e_i}$.
Third step is to transform the prime factors for $(k+1)B$-smooth values in to corresponding exponent vector, such that for

$$\tilde{v}(x^2 - n)\ =\ (e_1,\ e_2,\ ...,\ e_k)$$

Further, we reduce the above vector $\vec{v}$ to modulo 2. Here, comes the role of linear algebra to generate row vectors whose sum is equal to zero vector by using Gaussian Elimination.

$$\tilde{v}(x_1) + \tilde{v}(x_2) + ... + \tilde{v}(x_k) = \tilde{0} \tag{9.1}$$

Finally, we are left with $x = x_1\ .\ x_2\ ...\ x_k\ (mod\ n)$ and other variable y^2 (i.e. product of $x_i^2 - n$) is calculated as:

$$y = \sqrt{(x_1^2 - n)\ .\ (x_2^2 - n)\ ...\ (x_k^2 - n)}\ (mod\ n) \tag{9.2}$$

Now, we have desired identity such that $x^2\ \equiv\ y^2\ (mod\ n)$. And, calculate the greatest common divisors (gcd) of integer n to have factors such that $f_1 = gcd(x - y, n)$ and $f_2 = gcd(x + y, n)$. Although, detailed explanation of quadratic sieve algorithm and its working with an example can be found in [185, 191, 192].

9.4 Quantum Quadratic Sieve Algorithm

In this section, we provides a quantum variant of above classical quadratic sieve algorithm. Give an integer n, the algorithm will find the factors of n such that $x^2 \equiv y^2 mod\ n$. Assume that system has two quantum registers Register-1 and Register-2. The algorithm consists of three steps (steps 1 through 3) with step 1.1 and 3 requiring the use of classical computer and the remaining all other steps are executed on quantum computer. In step 1.1, we have used classical sieve of Eratosthenes algorithm to form prime state, which runs in an exponential time. Although, there exists an polynomial algorithm to form prime state by using Grover's algorithm, whose oracle is a quantum variant of Miller-Rabin primality test, which can be use in future [193]. Here, we begin with briefly describing all the steps of an algorithm.

Step 1 [Initialize] The Register-1 and Register-2 are initialized to zero. Therefore, the state of the registers becomes:

$$|\psi_0\rangle = |0\rangle_1 |0\rangle_2$$

Step 1.1 [Determine the primes] The prime numbers in the range (2,B) by using prime counting function on Register-1 are listed. Thus, the total state of the system becomes:

$$|\psi_p\rangle = \frac{1}{\sqrt{\pi(B)}} \sum_{p \in\ prime \leq B} |p\rangle_1 |0\rangle_2$$

Thus, we load the first register with an equally weighted superposition of all primes in range less than equal to B. And, Register-2 with zeros.

Step 1.2 [Prepare factor base] Compute the Legendre symbol on all primes below B. Thus, apply the unitary transformation $U_p : n^{\frac{p-1}{2}}\ mod\ p$ to each prime in the Register-1 and store the result in Register-2. The state of the system is changes as:

$$|\psi_1\rangle = \frac{1}{\sqrt{\pi(B)}} \sum_{p \in\ prime \leq B} |p\rangle_1 |n^{\frac{p-1}{2}}\ mod\ p\rangle_2$$

Now, the states of both registers are entangled with each other.

Step 1.3 [Perform measurement] Partial measurement is performed on Register-2 for which the calculated Legendre symbol $\left(\frac{n}{p}\right) = 1$. It will result in a probability distribution over outcomes of the Register-2: $P_{2,1} = Pr[outcome = 1] = \sum_{p \in\ prime} |\alpha_{p,l}|^2$ where, $\alpha_{p,l} = 1/\sqrt{\pi(B)}$ is the amplitude of prime associated with Legendre symbol 1 in Register-2.
The new state of the system becomes:

$$|\psi_2\rangle = \frac{1}{\sqrt{P_{2,1}}} \sum_{p \in\ prime} |p\rangle_1 |1\rangle_2$$

In order to form the factor base, we measure the second register for each 1. The state of second register will collapse, but the first register stays in superposition.

Step 2 [Initialize] Registers subscript 3,4 and 5 are initialized to zeros in order to sieve the sequence (x^2-n) for the interval $[\sqrt{n}+M,\ \sqrt{n}-M]$ by using optimum value of M. For simplification, it can be written as [a,b].

$$|\psi_s\rangle = |0\rangle_3 |0\rangle_4 |0\rangle_5$$

[Prepare information for quantum registers] Then, apply the Quantum Fourier transform (QFT) to Register-3 and state becomes:

$$|\psi_{s1}\rangle = \frac{1}{\sqrt{b-a}} \sum_{x=a}^{b} |x\rangle_3 |0\rangle_4 |0\rangle_5$$

.

Step 2.1 [Generate sequence] Next step is to apply the unitary transformation on the generated sequence in above step of Register-3 and store its result in Register-4. The state of the system becomes:

$$|\psi_3\rangle = \frac{1}{\sqrt{b-a}} \sum_{x=a}^{b} |x\rangle_3 |x^2-n\rangle_4 |0\rangle_5$$

The state $|\psi_3\rangle$ shows more than superpositions of three registers. Now, the states of these registers are entangled with each other.

Step 2.3 [Tensor product of sequence and factor base] Take the tensor product of states $|\psi_2\rangle$ and $|\psi_3\rangle$. Thus, the new state of the system becomes:

$$|\psi_{23}\rangle = |\psi_2\rangle \bigotimes |\psi_3\rangle$$

i.e.

$$|\psi_{23}\rangle = \frac{1}{\sqrt{P_{2,1}}} \frac{1}{\sqrt{b-a}} \sum_{x=a}^{b} |x\rangle_3 \sum_{p \in prime} |x^2-n\rangle_4 |0\rangle_5 |p\rangle_1 |1\rangle_2$$

Apply, QFT on the Register-4 and state becomes:

$$|\psi_{24}\rangle = \frac{1}{\sqrt{P_{2,1}}} \frac{1}{(b-a)} \sum_{y=a}^{b} \sum_{p \in prime} |1\rangle_2 \sum_{x=a}^{b} |x\rangle_3 |y^2-n\rangle_4 |0\rangle_5 |p\rangle_1$$

Step 2.4 [Factorize the sequence] To compute factors, divide the sequence stored in Register-4 using each prime of factor base (Register-1) iteratively such that

$$|\psi_{25}\rangle = \frac{1}{\sqrt{P_{2,1}}} \frac{1}{(b-a)} \sum_{y=a}^{b} \sum_{p \in prime} |1\rangle_2 \sum_{x=a,\ y \in integer}^{b} |x\rangle_3 |y^2-n/p\rangle_4 |0\rangle_5 |p\rangle_1$$

Here, we use the Register-4 as an input and output as well. Until, its output is an integer, we keep on increment the Register-5. Hence, there is no loss of information in Register- 4. If needed, we can retrieve it using the Register-3.

Step 2.5 [Observe the state of the quantum computer] Perform partial measurement on Register-4 of above state that equals 1 corresponds to smooth number. The probability of observing the state $|1, x, 1, e, p\rangle_{2,3,4,5,1}$ is

$$P_{4,1} = Pr[outcome = 1] = \sum_{x=a}^{b} |\beta_{x,1}|^2$$

where, $\beta_{x,l} = 1/(b-a)$ is the amplitude of each sequence associated with 1 in Register-4. Note, the comma here denotes that the registers are entangled with each other and e denotes the exponent associated with each prime for sequence. Thus, the resultant state becomes:

$$|\phi\rangle = \frac{1}{\sqrt{P_{2,1}}} \frac{1}{\sqrt{P_{4,1}}} \sum_{x=a}^{b} \sum_{p \in prime} |1\rangle_2 |x\rangle_3 |1\rangle_4 |e \ mod \ 2\rangle_5 |p\rangle_1$$

And also, perform the modulo 2 on exponents stored in Register-5.

Step 3 [Matrix processing of state vectors] Represent the quantum Register-5 in form of matrix M corresponds to its prime value for sequence in above state. Then, perform modulo 2 on matrix. Now, we have to look for the vector $(\overrightarrow{v})$ by using Gaussian elimination such that $M.\overrightarrow{v} = \overrightarrow{0}$. Further, we take the corresponding values of x from Register-3 using vector and calculate the other variable $y \ mod \ n$ (as given in Section 3). At the end, compute the greatest common divisors (gcd) to get factors of an integer n: $f_1 = gcd(x - y, \ n)$ and $f_2 = gcd(x + y, \ n)$.

9.5 Simulation and Results

In this section, we demonstrate a simulation of quantum quadratic sieve algorithm on a classical computer using computational language Mathematica. Its helps us to differentiate the result between quantum computer and classical computer. We have used Mathematica packages "Quantum Computing" (binary qubits, tensor products and tensor powers) and "Quantum Notation" (kets, bras and other quantum objects in Dirac notation).Therefore, high-level programming language Mathematica consists quantum operators and quantum states which gives us the concept for simulation of quantum algorithm on a quantum computer. Simulation of some quantum algorithms are performed in [194, 195].
The fastest classical number field sieve factoring algorithm takes $O(exp(cn^{1/3}(log\ n)^{2/3}))$ operations for some constant c, i.e. it is exponential in $n^{1/3}$, where n is the number to factored [196]. Other classical quadratic sieve algorithm also takes exponential operations to compute factors i.e. $O(exp((1+\epsilon)\sqrt{logn\ loglogn}))$, where $\epsilon > 0$ [197]. Shor's quantum algorithm takes asymptotically $O(n^2 logn\ loglogn)$, only a polynomial number of operations on quantum computer along with polynomial

time on classical computer [198]. On comparing these algorithms complexity with our proposed quantum quadratic sieve algorithm takes $O(n^2 logy\ loglogn)$ polynomial time on quantum computer, where y is the number of primes in factor base, classically takes $O(nloglogn)$ to form prime factor base using classical sieve of Eratosthenes and $O(y(w + ylog(y)\ log(logy)))$ for Gaussian elimination, where w is the number of operations needed to multiply with the vector to form null space. In short, if we use trial division method to find smooth numbers in sequence then it will be time consuming process. Thus, by using quantum parallelism, quantum quadratic sieve algorithm shortens the time drastically. Note, its computational complexity can be more improved if we use quantized sieve of Eratosthenes or existing polynomial quantum Grover's algorithm with Miller-Rabin primality test and recently introduced quantum Gauss Jordan elimination algorithm which has computation time of order $O(2^{N/2})$, where $N \geq 1$ for $N \times N$ size matrices [199]. Thus, it would be complete quantum quadratic sieve algorithm for factorization taking polynomial time on quantum computer.

However, the proposed quadratic sieve algorithm uses small primes factor base to compute factors efficiently over the sequence, which is computed polynomially over sieving interval. Although, its succession rate depends upon the smoothness bound B, if it is chosen randomly close to its optimal value, then it would have enough values to form congruent squares and possible to find factors of a large integer in less computational time as compared to its classical variant and number field sieve algorithm. On the other hand, quantum Shor's algorithm is not a factoring algorithm, but rather an algorithm for finding the order of element x modulo n, which results in a successful factorization of integer n [174]. Therefore, it fails half of the time values of small positive integer r, such that $x^r \equiv 1 (mod\ n)$.

In simulation, we have used three gates namely: Hadamard, controlled phase-shift and swap gate to compute QFT. Hadamard gate acts on a single qubit. $|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), |1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Controlled phase-shift gate modifies the phase of the qubit and its probability of measuring the qubit remains unchanged. Therefore, it maps the $|1\rangle$ to $e^{i\phi}|1\rangle$, where ϕ is phase shift and basis state $|0\rangle$ remains as it is. Swap gate exchanges the two qubits i.e. $|10\rangle \rightarrow |01\rangle$.

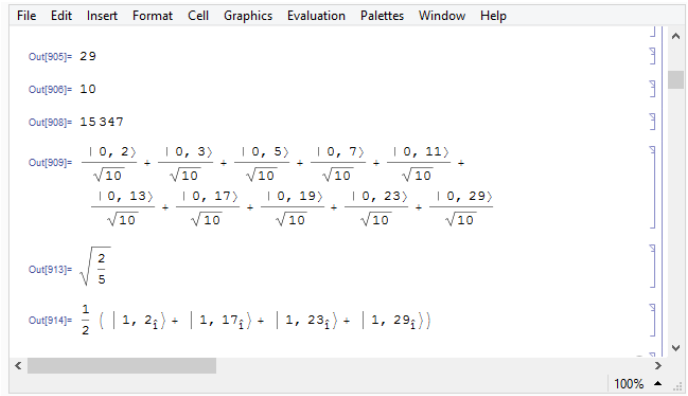


Figure 9.1: State after performing partial measurement on calculating Legendre symbol

$$|\psi_p\rangle = Expand[\frac{1}{\sqrt{Z}} \sum_{c=1}^{S} |Sequence[Part[S, c]], 0]\rangle \tag{9.3}$$

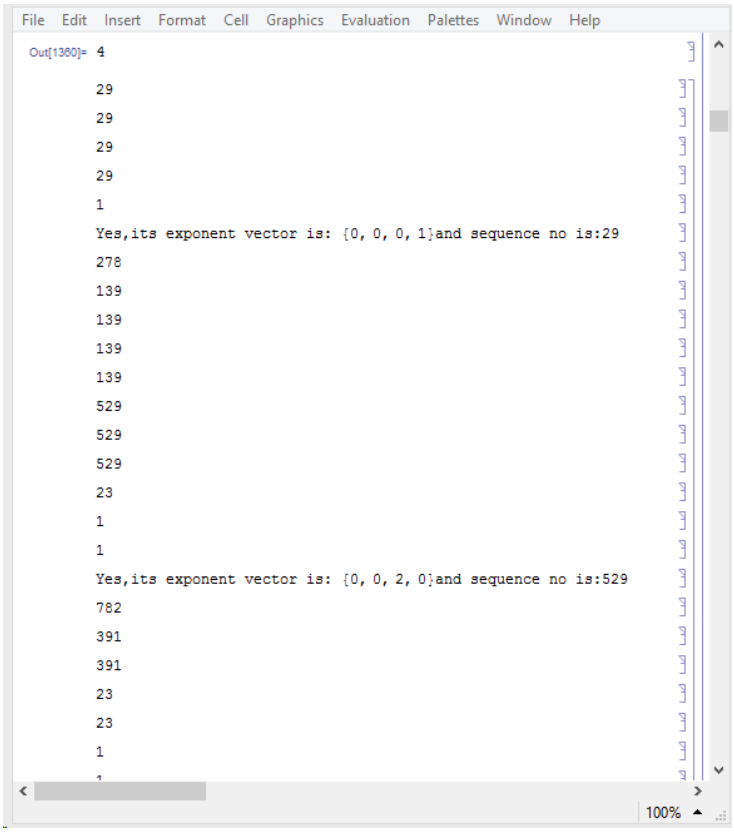


Figure 9.2: Generating exponent vectors

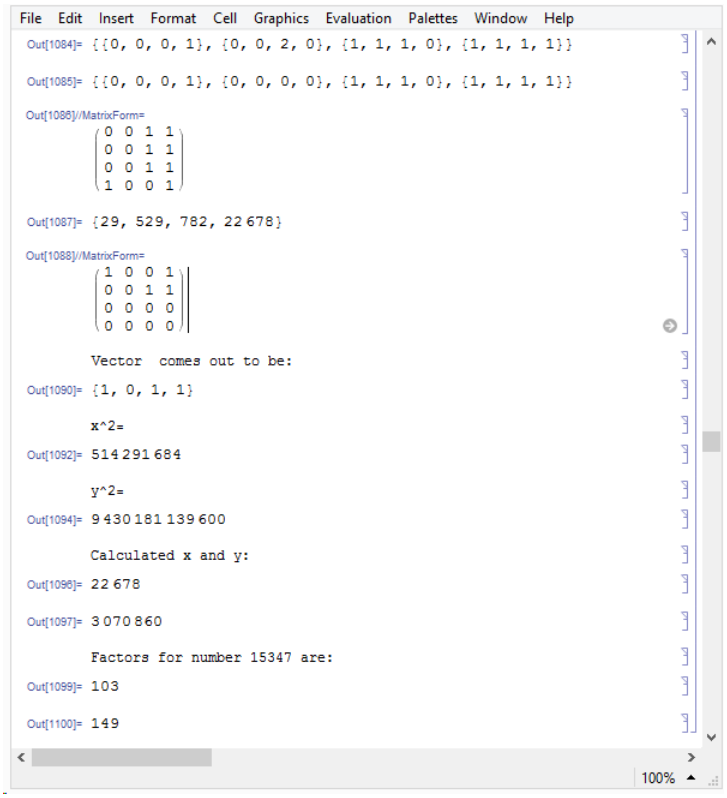


Figure 9.3: Simulation results for integer n=15347

In step 1.1, the state ψ_p is formed by using a function for classical sieve of Eratos-

thenes which returns an array (S) of primes on giving B as an input. The Expand command expands out a list of primes in Register-1 and zeros in Register-2. Figure 9.1 lists the all primes with equal probability less than equal to smoothness bound-B. Also, it shows the probability of measuring a Register-2 in Legendre symbol equal to 1 and outputs the a new quantum state after performing the partial measurement. Note, the comma in between the registers shows entanglement. Figure 9.2 shows a screen-shot of an exponent vectors generated for each sequence dividing with prime factor base.

In the last step of algorithm, classical Gaussian elimination is performed on matrix consisting exponent vectors. We have used the QubitToDec command over the quantum states for having decimal values in order to perform arithmetic operations and to calculate greatest common divisors. Finally, we have the factors of an integer n=15347 i.e. 103 and 149 as shown in Figure 9.3.

Chapter 10

McEliece Cryptosystem based On Extended Golay Code

With increasing advancements in technology, it is expected that the emergence of a quantum computer will potentially break many of the public-key cryptosystems currently in use. It will negotiate the confidentiality and integrity of communications. In this regard, we have privacy protectors (i.e. Post-Quantum Cryptography), which resists attacks by quantum computers, deals with cryptosystems that run on conventional computers and are secure against attacks by quantum computers. The practice of code-based cryptography is a trade-off between security and efficiency. In this chapter, we have explored the most successful McEliece cryptosystem, based on extended Golay code [24, 12, 8]. We have examined the implications of using an extended Golay code in place of usual Goppa code in McEliece cryptosystem. Further, we have implemented a McEliece cryptosystem based on extended Golay code using MATLAB. The extended Golay code has lots of practical applications. The main advantage of using extended Golay code is that it has codeword of length 24, a minimum Hamming distance of 8 allows us to detect 7-bit errors while correcting for 3 or fewer errors simultaneously and can be transmitted at high data rate.

10.1 Introduction

Shor's algorithm is well-known in the field of cryptography given its potential application in cracking various cryptosystems, such as RSA algorithm and elliptic curve cryptography [176]. These all public key cryptosystems can be attacked in polynomial time using Shor's algorithm. Table 1 represents the present status of several cryptosystems [175].
Since many years, the concept of error correcting codes have gained significant interest among researchers in field of cryptography. In particular, McEliece proposed a public-key cryptographic system based on binary Goppa codes, which is proved to be very secure against attacks by quantum computers. This cryptosystem applies a generator matrix as the private key and a linear transformation of it as the public key. The security lies in difficulty of decoding a large linear code with no visible structure, that is an NP complete problem. Several variants have been introduced to replace these binary Goppa codes. The original McEliece cryptosystem remains unbroken, as no algorithm able to realize a total break in an acceptable time has been presented till now. Despite the advances in the field, however, the factors

Table 10.1: Impact of Quantum computing on cryptographic algorithms

Cryptosystem	Broken by Quantum algorithms?
Diffie-Hellman key-exchange [200]	Broken
RSA public key encryption [201]	Broken
Algebraically Homomorphic [202]	Broken
Buchmann-Williams key-exchange [203]	Broken
Elliptic curve cryptography [204]	Broken
NTRU public key encryption [205]	Not broken yet
McEliece public key encryption [206]	Not broken yet
Lattice-based public key encryption [207]	Not broken yet

needed for its violation remain very high and difficult to trace in real. Moreover, the original McEliece cryptosystem is two or three orders of magnitude faster than RSA, the later being, likely, the most widely used public key encryption algorithm currently. Motivated by the above mentioned facts, a variant of McEliece cryptosystem is proposed based on Golay codes.
Post-Quantum Cryptography offers secure alternatives. The goal of post-quantum cryptography is to develop cryptographic systems that are secure against both quantum and classical computers, and compatible with existing communications protocols and networks. Apart from RSA, digital signature algorithm (DSA), and Elliptic curve cryptosystems (ECDSA), there are other important classes of cryptographic systems which include Code-based, Lattice-based, Hash-based, Multivariate-quadratic-equations and Secret-key cryptosystem.
Code-based cryptography [208] generally refers to cryptosystems in which the algorithmic primitive uses an error correcting code C. It may comprise of appending an error to a word of C or in calculating a syndrome related to a parity check matrix of C. There are several codes for which efficient decoders are known.
In 1949, Golay [209] discovered Golay codes. A binary Golay code is a linear error-correcting code used in digital communication. Golay codes are perfect codes in which the Hamming spheres surrounding the codewords fill the Hamming space without overlap. These spheres have a radius e, which can correct e errors and their codewords separated from each other by a distance $d=e+1$. Perfect codes possess complete bounded-distance decoders and satisfy the Hamming bound with equality. If Golay codes are augmented with bit interleaving technique, it enables us to correct burst errors [210].

10.2 Preliminaries

In this section, some preliminaries and basic notations are given, which will be used throughout the chapter.

- *Linear code*: Linear code C [211] of length n and dimension k over a field F is a k-dimensional subspace of the vector space F_q^n with q elements, a set of n-dimensional vectors can be referred to as a $[n, k]$ code and elements of bits such that F=GF(2)=$\{0,1\}$. If the minimum Hamming distance of the code is

d, then the code is called a $[n,\ k,\ d]$ code.

- *Hamming distance*: A Hamming distance [210] $d_H(x, y)$ is the number of positions in which two codewords *(x, y)* differ. Let C be a $[n,\ k]$ linear code over F_q^n and $x = (x_1, x_2, ..., x_n), y = (y_1, y_2, ..., y_n)$ are two code words.

$$d_H(x, y) =\mid i : x_i \neq y_i, 1 \leq i \leq n \mid \tag{10.1}$$

- *Hamming weight*: A Hamming weight [210] $wt_H(x)$ is defined as the number of non-zero positions in the codeword x. Let C be a $[n,\ k]$ linear code over F_q^n and $x = (x_1, x_2, ..., x_n)$ is a code word, such that

$$wt_H(x) =\mid i : x_i \neq 0, 1 \leq i \leq n \mid \tag{10.2}$$

- *Generator matrix*: A generator matrix [210] for C is a $k \times n$ matrix G having the vectors of $V = (v_1, v_2, ..., v_k)$ as rows, which forms a basis of C such that

$$C = \{mG : m \in F_q^n\},\ G = \begin{bmatrix} v_1 \\ v_2 \\ ... \\ v_k \end{bmatrix} \tag{10.3}$$

  The matrix G generates the code as a linear map: for each message $m \in F_q^n$, we obtain the corresponding code word mG.

- *Dual code*: Let C be a $[n,\ k]$ linear code over F_q^n. The dual code [26] of C is the set, such that $C^{\perp} = \{x \in F_q^n : x.y = 0, \forall y \in C\}$.

- *Parity matrix*: A $(n-k) \times n$ generator matrix H is called parity-check matrix [210] for codeword C, which is described by

$$C = \{m \in F_q^n : mH^T = 0\}, \tag{10.4}$$

10.3 Prior work

Originally, Golay codes [209] were invented in the early 1950's, and have experienced incredible responses in the last few years. In 1978, McEliece [206] proposed an asymmetric encryption cryptosystem based on Goppa codes, which remains unbroken, even after 15 years of adaptation of its proposal security parameters [212]. Niederreiter [213] proposed a knapsack-type cryptosystem based on Reed-Solomon codes. Sidelnikov and Shestakov [211] attacked the Niederreiter cryptosystem and proved that it is insecure using Reed-Solomon codes as well as Goppa codes.
Sidelnikov [214] proposed a public-key cryptosystem based on binary Reed-Muller codes. It offered a high security with transmission rate close to 1, and complexity of encryption and decryption process is low. Minder and Shokrollahi [215] attacked the Sidelnikov public-key cryptosystem which generates a private key from a known public key. It has been shown that running time of the attack is subexponential using low weight finding algorithms.
Janwa and Moreno [216] proposed a McEliece public key cryptosystems based on Algebraic-Geometric Codes (AGC). It shows the various aspects of McEliece cryptosystem, based on the larger class of q-ary algebraic-geometric Goppa codes and

listed some open problems for future improvements. Faure and Minder [217] presented an algorithm based on algebraic geometry codes to recover the structure of algebraic geometry codes defined over a hyperelliptic code. In 2014, Couvreur et al. [218] constructed a polynomial time algorithm attack against public key cryptosystems based on algebraic-geometric codes.
In 2000, Monico et al. [219] showed an efficient way of using low-density parity check codes in McEliece cryptosystem. In 2007, Baldi et al. [220] introduced a new variant of McEliece cryptosystem, based on quasi-cyclic low-density parity check (QCLDPC) codes. Furthermore, they examined the relevant attacks against LDPC and QCLDPC. Londahl and Johansson [221] constructed a new version of McEliece cryptosystem based on convolutional codes. Landais and Tillich [222] implemented an attack against McEliece cryptosystem based on convolutional codes. Various researchers proposed modified McEliece cryptosystems by replacing Goppa codes and using different error-correcting codes, e.g. algebraic geometric codes (AGC), low-density parity check codes (LDPCC) or convolutional codes. However, all of these schemes have proven to be insecure, making Goppa codes a standard solution.

10.4 McEliece Cryptosystem

McEliece cryptosystem is based on linear error-correcting code for creating public and private key. Binary Goppa code [206] is used as the error-correcting code in McEliece cryptosystem. The secret key can be drawn from the various alternate codes. Several versions of McEliece cryptosystem were proposed using various secret codes such as Reed-Solomon codes, concatenated codes and Goppa codes. Interested researchers can study the original McEliece cryptosystem algorithm described in [206].

10.5 Golay Codes

Golay codes can be classified into binary and ternary Golay codes. Furthermore, binary Golay codes are divided into extended (G_{24}) and perfect (G_{23}) binary Golay codes [209, 210]. The extended binary Golay code G_{24} is a [24, 12, 8] code, which encodes 12 bits of data into a word of 24-bit length in such a way that any 3-bit errors can be corrected or any 7-bit errors can be detected.
The perfect binary Golay code G_{23} is a [23, 12, 7] code that is having a code word of length 23. It can be obtained from the extended binary Golay code by deleting one coordinate position. It is useful in the applications where a parity bit is added to each word for producing a half-rate code [223]. It is constructed by a factorization $x^{23}-1$ over field F_2^m such that: $x^{23}-1 = (x-1)(x^{11}+x^9+x^7+x^6+x^5+x+1)(x^{11}+x^{10}+x^6+x^5+x^4+x^2+1)$, $g_1(x) = (x^{11}+x^9+x^7+x^6+x^5+x+1)$ and $g_2(x) = (x^{11}+x^{10}+x^6+x^5+x^4+x^2+1)$ are irreducible polynomials of degree (m=11). These polynomials are reverse of each other and can generate the same cycle code words. Therefore, the generator matrix 12×23 of perfect binary Golay code is $G_{23} = [I_{12}, A_{23}]$, where I_{12} is 12×12 the identity matrix. Matrix A_{23} is as follow:

$$A_{23} = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

10.5.1 Binary extended Golay codes

In 1977, extended Golay codes G_{24} [209] were used for error control on the Voyager 1 and 2 spacecraft launched towards Jupiter and Saturn. The perfect binary Golay code results into 3-byte extended Golay code by adding a parity bit. Some special properties of extended Golay Codes are:

- G_{24} is a self-dual code with a generator matrix $G = [I_{12} \mid A]$.
- Parity check matrix for G_{24} is $H = [A \mid I_{12}]$ [224].
- Another generator and parity check matrix for G_{24} are $G' = [A \mid I_{12}]$ and $H = G^t[\frac{A}{I_{12}}]$ respectively [225].
- The weight of every code word in G_{24} is a multiple of 4 and distance is 8.

The extended Golay code generated by the 12×24 matrix $G = [I_{12} \mid A]$, where I_{12} is 12×12 the identity matrix and matrix A is as shown below.

$$A = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

10.6 McEliece Cryptosystem using extended Golay code

McEliece cryptosystem based on extended Golay code works similarly as McEliece cryptosystem, but it generates the secret matrix G with a different way, and different decoding procedure will be used for the decoding process. Golay code matrix A is having a cyclic structure, in which the second row is obtained by moving the first component to the last position. Similarly, each row of the matrix A can be obtained by a right shift of the previous row, except last one row. The matrix A is being a part of both the generator and the parity check matrices of extended Golay code; its decoding procedure is very simple. The main idea is to replace the Goppa code used in McEliece by an extended Golay code that can be efficiently decoded. Fig 10.1 shows the working of McEliece cryptosystem based on extended Golay code

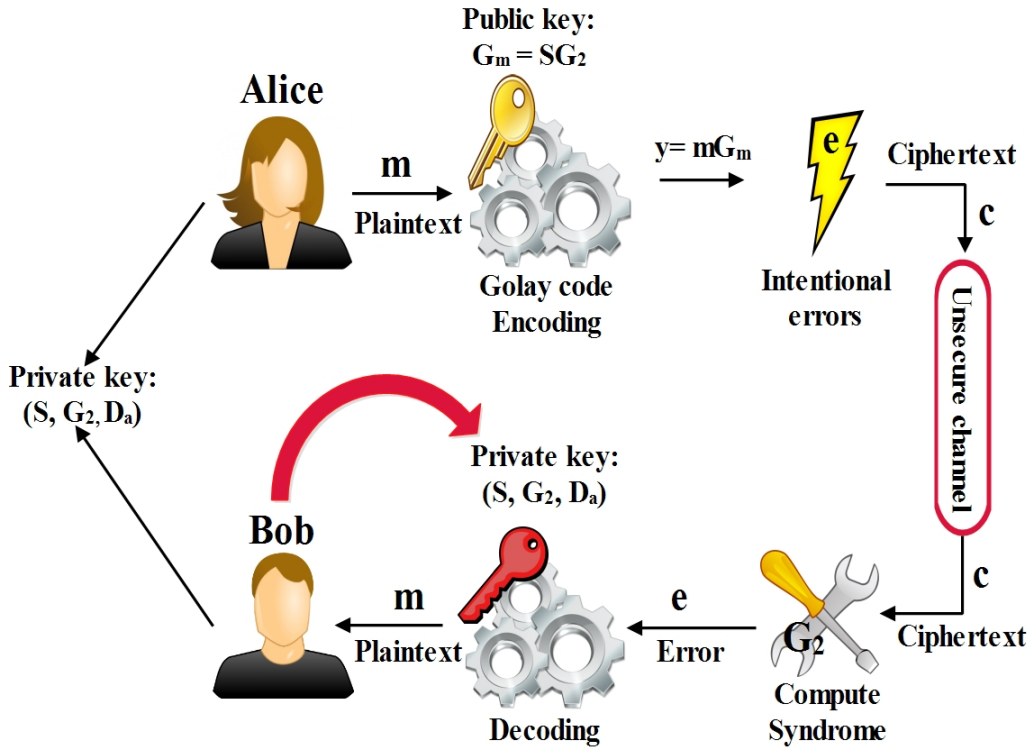


Figure 10.1: McEliece Cryptosystem using extended Golay code

10.6.1 Key generation

McEliece cryptosystem based on extended Golay code, G_{24} encode 12-bits of data in 24-bit length of the word. Random permutation matrix (P) acts on generator matrix (G). Then, reorder the computed matrix and named it as G_2. Compute $G_m \leftarrow SG_2$ by the random invertible matrix (S) and makes the public key (G_m, t) and secret key (S, G_2, D_a) correcting any 3-bit of errors. Key generation is described in Algorithm 1. The detailed algorithm of McEliece cryptosystem based on extended Golay code is given below.

Algorithm 1: Key generation
System parameters: Let F be a family of t-error correcting ($t \in N$) q-ary linear [n, k, d] codes, where $t \ll n$.
Input: G_{24} [24, 12, 8] is an extended Golay which encodes (k=12) bits of data in a word of (n=24) bit length and any (t=3) bit errors can be corrected.
Output matrices:

- Generate generator matrix G: $k \times n$ generator matrix for code C capable of correcting e errors over F of dimension k. $G \leftarrow [I_{12} \mid A]$, where I_{12} is 12×12 the identity matrix.
- Generate permutation matrix P: $n \times n$ is a random permutation matrix, having exactly 1 in every row and column; with all other entries is zero.
- Compute $k \times n$ matrix $G_1 = GP$, arrange G_1 in systematic format of generator matrix and named it as G_2.
- Generate a non-singular invertible matrix $S \in F_2^{k \times k}$.
- Compute $k \times n$ matrix $G_m \leftarrow SG_2$.
- Return public key: (G_m, t), private key: (S, G_2, D_a), where D_a is an efficient decoding algorithm.

10.6.2 Encoding

In encoding, the plaintext is a random non-zero binary vector of length k, i.e. ($m \in F_2^k$). A ciphertext ($c \in F_2^n$) is the code word of the code with generator matrix G_m and we choose random error vector ($e \in F_2^n$) exactly of weight t. The encoding process is defined in the Algorithm 2.

Algorithm 2: Encoding **Input:** Public key ($G_m \in F_2^{k \times n}$), message ($m \in F_2^k$), error vector ($e \in F_2^n$). **Output:** Ciphertext ($c \in F_2^n$)
Compute $y \leftarrow mG_m$ Add error vector $c \leftarrow y + e$ Return c

10.6.3 Decoding

The decoding process is defined in the Algorithm 3. It uses the decoding procedure of extended Golay code, whereas original McEliece cryptosystem uses Patterson's algorithm for the decoding process.
Here, we call a subroutine $D_a(c)$, which computes an error vector described in the algorithm 4. Therefore, on reading input a ciphertext ($c \in F_2^n$), it generates an output as the original message ($m \in F_2^k$). In step 1, it computes a syndrome using private key G_2 checks whether the weight of syndrome s_1 is less than or equal to 3. If yes, then it returns an error vector e=[s_1, 000000000000]. Otherwise, it checks the

Algorithm 3: Decoding **Input:** Ciphertext ($c \in F_2^n$), Private key: (S, G_2, D_a) **Output:** Original message ($m \in F_2^k$)
Compute the encoded message $y_1 \leftarrow c + e$, where e is calculated by calling subroutine $D_a(c, G_2)$. $y_1 \leftarrow mSG_2 + e$, compute message mS by row reducing $[G_2^t \mid (mSG_2)^t]$. Multiply mS by S^{-1}. Return m

weight of $(s_1 + A_i)$ is less than or equal to 2, then the error vector is e=$[s_1 + A_i, j_i]$. If it does not satisfy the first condition, then further it computes the second syndrome s_2 and checks whether the weight of syndrome s_2 is less than or equal to 3. If yes, then it returns an error vector e=$[000000000000, s_2]$. Otherwise, it checks the weight of $(s_2 + A_i)$ is less than or equal to 2, then the error vector is e=$[j_i, s_2 + A_i]$. In any case, if both the conditions do not satisfy and the error pattern e is not yet determined, then it requests retransmission. Finally, mS is found by row reducing form and the original message is computed by multiplying mS by S^{-1}.

Algorithm 4: $D_A(c, G_2)$ **Input:** Ciphertext ($c \in F_2^n$), generator matrix (private): (G_2) **Output:** Error vector ($e \in F_2^n$)

Compute the first syndrome: $s_1 \leftarrow cG_2$
If $wt(s_1) \leq 3$, then
Return $e \leftarrow [s_1, 000000000000]$
Else If $wt(s_1 + A_i) \leq 2$, then
Return $e \leftarrow [s_1 + A_i, j_i]$, where j_i the word of length 12 with 1 in the i^{th} position and 0 elsewhere in I_{12} identity matrix.
Else
Compute the second syndrome: $s_2 \leftarrow s_1 A$
If $wt(s_2) \leq 3$, then
Return $e \leftarrow [000000000000, s_2]$
Else If $wt(s_2 + A_i) \leq 2$, then
Return $e \leftarrow [j_i, s_2 + A_i]$
Else If the error pattern e is not yet determined, then request retransmission.

10.6.4 Security

The security of the proposed McEliece cryptosystem depends on the difficulty level to decode y into message m. The attacker will have a tough time trying to separate G_2 from G_m because he/she does not know P and inverse of a matrix S, which are not publicly available. Therefore, an attacker cannot find an error because it's hard to recover the specific structure of the matrix G_2. Maximum-likelihood decoding can be used to recover error but making tables for big codes ($2^{n-k} = 2^{24-12} = 4096$) coset leader is a time-consuming and inefficient. It also needs more storage space and decoding time can be quite long also. Therefore, we rely on syndrome decoding of extended Golay code.

10.7 Implementation of McEliece Cryptosystem using Extended Golay Code

We have used a personal computer to implement McEliece cryptosystem based on extended Golay code with the following specification: CPU Intel Core i3-3217U 1.80 GHz, RAM 2.00 GB, OS Windows 8 Enterprise 32 bit and MATLAB 7.11.0 (R2010b).
We have used 12×24 generator matrix $G=[I_{12} \mid A]$ to generate extended Golay code G_{24}, where I_{12} is 12×12 the identity matrix. Fig 10.2 shows the matrix A, which is obtained by adding a parity bit at the end of each codeword of perfect Golay code G_{23}. We have used the random permutation matrix 24×24 to compute $G_1 \leftarrow GP$ as shown in Fig 10.3.

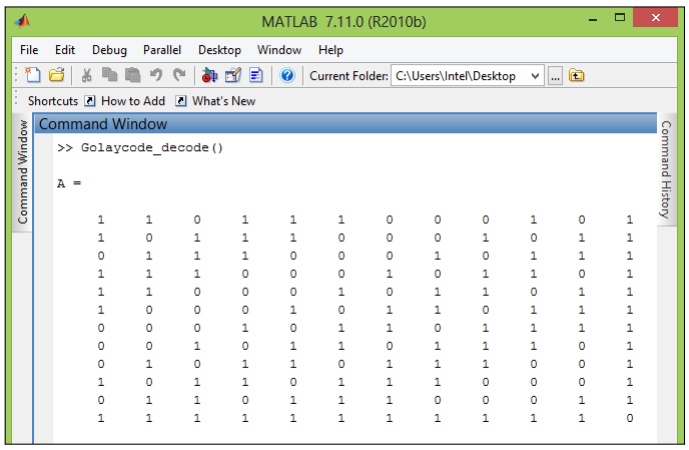


Figure 10.2: Generator polynomial matrix A of G_{24}

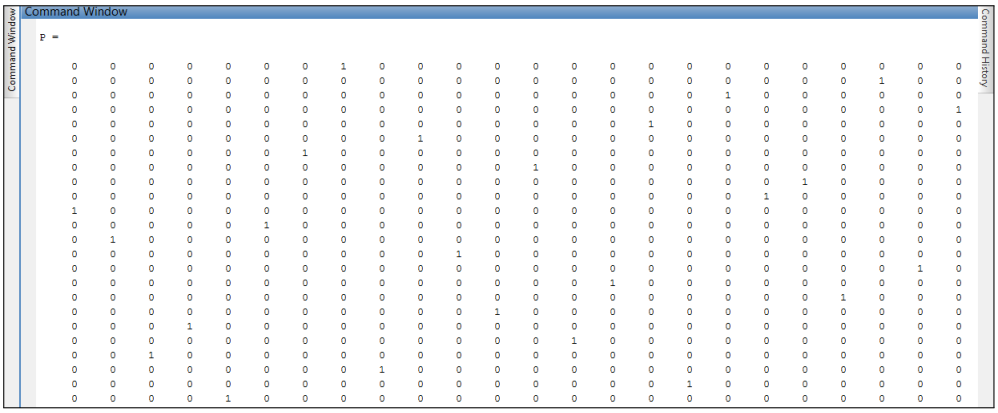


Figure 10.3: Random permutation matrix P

We have used the random function to generate 12×12 a random invertible matrix S of binary numbers. The matrix G_1 is reordered and renamed as G_2, then we computed $G_m = SG_2$, where G_m is the encoding matrix. Fig 10.4 represents the random invertible matrix S. Furthermore, G_m encoding matrix results in public key: (G_m, t), the private key consists of a random matrix S, systematic generator matrix G_2 and D_a efficient decoding algorithm such that (S, G_2, D_a). We have used random plaintext m of length 12 and random error vector e of length is 24 having weight

($wt \leq 3$). Then, we compute codeword by $y = mG_m$ and encode it by computing ciphertext such that $c = y + e$. Fig 10.5 shows the computed G_m matrix codeword, random error, and ciphertext.

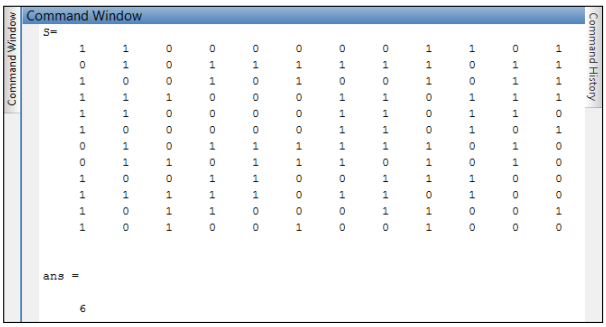


Figure 10.4: Random invertible matrix S

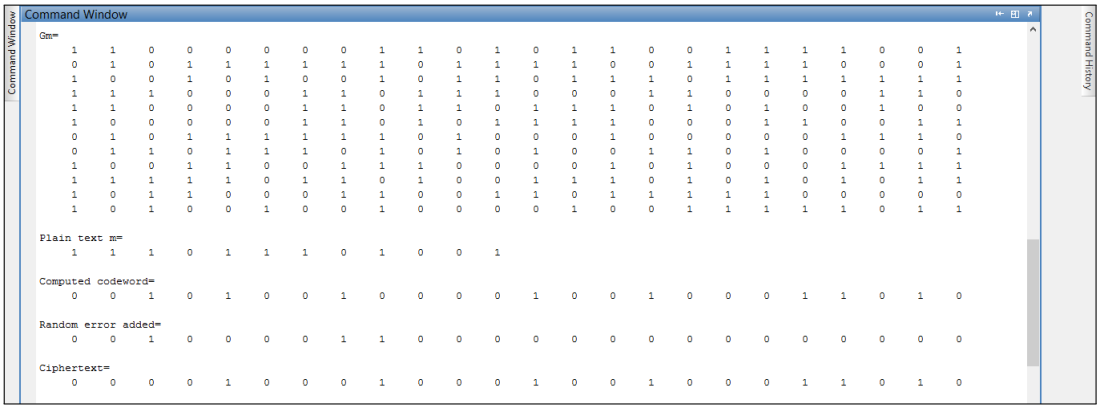


Figure 10.5: Generate ciphertext by adding intended error

During decoding, we call a subroutine as described in Algorithm 4 for computing an error e by using private key G_2. Further, we recovered the actual codeword such that $y = c + e$. Fig 10.6 shows the calculated syndrome for error detection in the ciphertext.

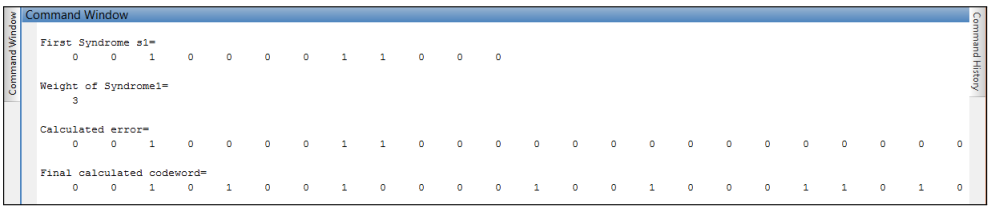


Figure 10.6: Error detection by calculating syndrome

Compute the error and actual codeword; we recover the plaintext by multiplying it with the inverse of S. Fig 10.7 shows the actual message sent over the channel.
We have examined the McEliece cryptosystem using extended Golay code. The developed system is effective and secure until S is chosen sparse random matrix. It corrects up to three-bit errors per codeword. Sparse matrices make it efficient and it allows a significant compression. Moreover, we have implemented the McEliece cryptosystem using extended Golay code and designed a finite state machine for

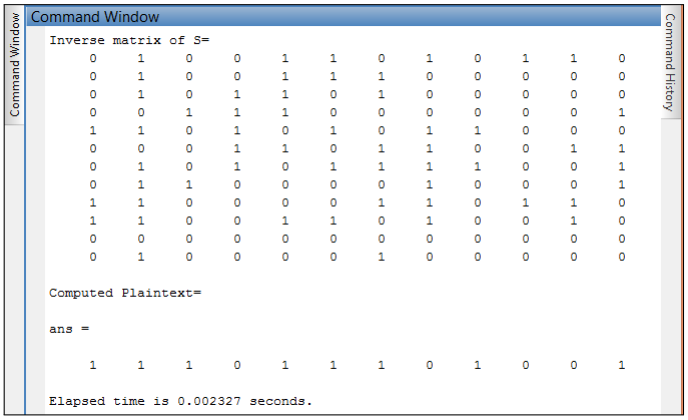


Figure 10.7: Decoding of Ciphertext

its decoding component. In future, we will design McEliece cryptosystem using extended Golay code associated with bit interleaving technique to correct bursts of errors per codeword.

Chapter 11

Conclusion

A quantum finite automaton is a theoretical model with finite memory which lay down the vision of quantum processor. There are various quantum automata models which are more powerful than classical ones. In this Thesis, we have comprehensively reviewed and analyzed various aspects of quantum finite automata based on past research literature. It helps to understand theoretically the fundamentals of various quantum computing models. We have outlined their definitions, behavior, closure properties, language recognition power, comparison, inclusion relationships, equivalence criteria, minimization and simulation results. We have subsequently summarized the literature published to date in the form of a systematic evolution of QFA models. Further, various quantum finite automata models are compared on the basis of language recognition. This Thesis makes a positive contribution to the growing body of quantum computing literature by exploring the computational power of various QFA models. Moreover, various issues present in the research are addressed and some outstanding research questions still unresolved in various QFA models are identified. Furthermore, we have introduced quantum variants of classical computational models and compared with their counterparts. We have obtained the following:

- A variant of two-way quantum finite automata named two-way multihead quantum finite automaton is proposed.

- It has been shown that the class of languages recognized by k-head two-way quantum finite automata is a subset of the class of languages recognized by $(k+1)$-head two-way quantum finite automata.

- It has been investigated that quantum variant of two-way deterministic multihead finite automata takes less number of heads to recognize a language containing of all words whose length is a prime number.

- A quantum analogue of classical queue automata is introduced by using the definition of the quantum Turing machine and quantum finite-state automata.

- We have also introduced a generalization of real-time deterministic queue automata, the real-time quantum queue automata which work in real-time i.e. the input head can move towards the right direction only and takes exactly one step per input symbol.

- It has been proved that real-time quantum queue automaton is more superior than its real-time classical variants by using quantum transitions.
- There exist languages that can be recognized by real-time quantum queue automata and cannot be recognized by real-time deterministic (reversible) queue automata and real-time non-deterministic queue automata.

The further research in the connection between quantum finite state machines and matrix product state in tensor network theory. We have constructed quantum finite state machines of GHZ, AKLT, cluster and W state using unitary criteria and their probability distributions are investigated respectively. Further, the proposed unitary criteria is used to investigate the dynamics of matrix product state with quantum weightless neural networks, where the output qubit is extracted and fed back (iterated) to input and Von Neumann entropy to measure the possible entanglement of output quantum state. Inspired by the results of finite automata working on infinite words, we studied the quantum ω-automata with Büchi, Muller, Rabin and Streett acceptance condition. The class of languages recognized by quantum ω-automata are investigated from two aspects: the language recognition and their closure properties. It has been proved that quantum Muller automaton is more dominant than quantum Büchi automaton. Furthermore, the languages recognized by one-way quantum finite automata with different quantum acceptance conditions are shown and their closure properties are proved. The notations quantum finite automata from linear temporal logic point of view is studied. We have shown that the class of languages accepted by quantum finite automata are definable in linear temporal logic formulas, except for measure-once one-way quantum finite automata.

The third part of the Thesis is devoted to applying the assets of quantum computing in different areas. Firstly, RNA secondary structure loops such as, hairpin loop, internal loop and double helix loop modeled using two-way quantum finite automata in linear time. In contrast, two-way deterministic finite automata (2DFA) cannot represent RNA secondary structure loops and two-way probabilistic finite automata (2PFA) can parse these sequences in exponential time. Further, we have designed a quantum variant of classical factorization algorithm named "Quadratic Sieve" using the quantum principle of superposition and entanglement. Its has been proved that it is more efficient than its classical variants from computational complexity point of view. Finally, the most successful McEliece cryptosystem is explored based on extended Golay code [24, 12, 8] and examined the implications of using an extended Golay code in place of usual Goppa code. Although, we have not compared the newly McEliece cryptosystem with the existing ones, which is left for future work. We have not designed the quantum finite state machines of other members of tensor network theory, so it is another potential research area. In future, the proposed quantum quadratic sieve algorithm can be simulated on a quantum computer and compared with its counterparts.

Bibliography

[1] R. P. Feynman, "Simulating physics with computers," *International Journal of Theoretical Physics*, vol. 21, no. 6, pp. 467–488, 1982.

[2] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *FOCS '94: Proceedings of the 35th Annual Symposium on Foundations of Computer Science*. IEEE, Santa Fe, New Mexico, USA, 1994, pp. 124–134.

[3] D. Wineland, C. Monroe, D. Meekhof, B. King, D. Leibfried, W. Itano, J. Bergquist, D. Berkeland, J. Bollinger, and J. Miller, "Quantum state manipulation of trapped atomic ions," in *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 454, no. 1969. The Royal Society, 1998, pp. 411–429.

[4] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *STOC '96: Proceedings of the 28th Annual ACM Symposium on Theory of computing*. ACM, Pennsylvania, 1996, pp. 212–219.

[5] P. Kwiat, J. Mitchell, P. Schwindt, and A. White, "Grover's search algorithm: an optical approach," *Journal of Modern Optics*, vol. 47, no. 2-3, pp. 257–266, 2000.

[6] A. Younes, "Strength and weakness in Grover's quantum search algorithm," *arXiv preprint:0811.4481*, 2008.

[7] J. Wang, *Handbook of Finite State Based Models and Applications*. CRC Press, 2012.

[8] R. Landauer, "Irreversibility and heat generation in the computing process," *IBM Journal of Research and Development*, vol. 5, no. 3, pp. 183–191, 1961.

[9] D. Deutsch, "Quantum theory, the Church-Turing principle and the universal quantum computer," in *Proceedings of Royal Society London*, vol. 400, no. 1818. The Royal Society, 1985, pp. 97–117.

[10] A. Ambainis, A. Kikusts, and M. Valdats, "On the class of languages recognizable by 1-way quantum finite automata," in *STACS '01: Proceedings of the 18th Annual Symposium on Theoretical Aspects of Computer Science*. Springer, Dresden, Germany, 2001, pp. 75–86.

[11] K. H. M. Nakanishi, T. Indoh, K. Hamaguchi, and T. Kashiwabara, "On the power of quantum pushdown automata with a classical stack and 1.5-way quantum finite automata," *Nara Institute of Science and Technology, Technical report*, 2001.

[12] A. C. C. Yao, "Quantum circuit complexity," in *FOCS '93: Proceedings of the 34th Annual Symposium on Foundations of Computer Science*. IEEE, Palo Alto, California, 1993, pp. 352–361.

[13] C. Moore and J. P. Crutchfield, "Quantum automata and quantum grammars," *Theoretical Computer Science*, vol. 237, no. 1-2, pp. 275–306, 2000.

[14] A. Kondacs and J. Watrous, "On the power of quantum finite state automata," in *FOCS '97: Proceedings of the 38th Annual Symposium on Foundations of Computer Science*. IEEE, Florida, USA, 1997, pp. 66–75.

[15] J. Gruska, *Quantum computing*. McGraw-Hill London, 1999, vol. 2005.

[16] A. Ambainis and A. Yakaryilmaz, "Automata and quantum computing," *arXiv preprint:1507.01988*, 2015.

[17] D. Qiu and L. Li, "An overview of quantum computation models: quantum automata," *Frontiers of Computer Science in China*, vol. 2, no. 2, pp. 193–207, 2008.

[18] C. Mereghetti, B. Palano, S. Cialdi, V. Vento, M. G. Paris, and S. Olivares, "Photonic realization of a quantum finite automaton," *Physical Review Research*, vol. 2, no. 1, p. 013089, 2020.

[19] Y. Tian, T. Feng, M. Luo, S. Zheng, and X. Zhou, "Experimental demonstration of quantum finite automaton," *npj Quantum Information*, vol. 5, no. 1, pp. 1–5, 2019.

[20] M. A. Nielsen and I. Chuang, *Quantum computation and quantum information*. Cambridge University Press, New York, 2010.

[21] J. E. Hopcroft, *Introduction to automata theory, languages, and computation*, 3rd ed. Pearson Education, India, 2013.

[22] I. Dzelme-Bērziņa, "Quantum finite automata and logic," Ph.D. dissertation, University of Latvia, Riga, 2010.

[23] P. Wittek, *Quantum machine learning: what quantum computing means to data mining*. Academic Press, 2014.

[24] D. C. Marinescu, *Classical and quantum information*. Academic Press, 2011.

[25] J. Bub, "Quantum information and computation," *arXiv preprint quant-ph/0512125*, 2005.

[26] A. Ambainis, M. Beaudry, M. Golovkins, A. Kikusts, M. Mercer, and D. Thérien, "Algebraic results on quantum automata," *Theory of Computing Systems*, vol. 39, no. 1, pp. 165–188, 2006.

[27] A. Bertoni, C. Mereghetti, and B. Palano, "Quantum computing: 1-way quantum automata," in *DLT '03: Proceedings of the International Conference on Developments in Language Theory*. Springer, Szeged, Hungary, 2003, pp. 1–20.

[28] K. Paschen, "Quantum finite automata using ancilla qubits," *University of Karlsruhe, Technical report*, 2000.

[29] A. Ambainis, A. Nayak, A. Ta-Shma, and U. Vazirani, "Dense quantum coding and a lower bound for 1-way quantum automata," in *STOC '99, Proceedings of the 31st Annual ACM Symposium on Theory of Computing*. ACM, Georgia, USA, 1999, pp. 376–383.

[30] A. Nayak, "Optimal lower bounds for quantum automata and random access codes," in *FOCS '99: Proceedings of the 40th Annual Symposium on Foundations of Computer Science*. IEEE, New York, 1999, pp. 369–376.

[31] M. Hirvensalo, "Quantum automata with open time evolution," in *Nature-Inspired Computing Design, Development, and Applications*. IGI Global, 2012, pp. 74–89.

[32] L. Li, D. Qiu, X. Zou, L. Li, L. Wu, and P. Mateus, "Characterizations of one-way general quantum finite automata," *Theoretical Computer Science*, vol. 419, pp. 73–91, 2012.

[33] A. Ambainis and J. Watrous, "Two-way finite automata with quantum and classical states," *Theoretical Computer Science*, vol. 287, no. 1, pp. 299–311, 2002.

[34] D. Qiu, P. Mateus, and A. Sernadas, "One-way quantum finite automata together with classical states," *arXiv preprint:0909.1428*, pp. 3006–3017, 2009.

[35] A. Brodsky and N. Pippenger, "Characterizations of 1-way quantum finite automata," *SIAM Journal on Computing*, vol. 31, no. 5, pp. 1456–1478, 2002.

[36] A. Yakaryilmaz and A. C. Say, "Languages recognized with unbounded error by quantum finite automata," in *International Computer Science Symposium in Russia*. Springer, 2009, pp. 356–367.

[37] M. Amano and K. Iwama, "Undecidability on quantum finite automata," in *STOC '99: Proceedings of the 31st Annual ACM Symposium on Theory of Computing*. ACM, Georgia, USA, 1999, pp. 368–375.

[38] A. Ambainis and R. Freivalds, "1-way quantum finite automata: strengths, weaknesses and generalizations," in *FOCS '98: Proceedings of the 39th Annual Symposium on Foundations of Computer Science*. IEEE, Palo Alto, California, 1998, pp. 332–341.

[39] A. Kikusts, "A small 1-way quantum finite automaton," *arXiv preprint: quant-ph/9810065*, 1998.

[40] M. P. Ciamarra, "Quantum reversibility and a new model of quantum automaton," in *FCT '01: Proceedings of the 13th International Symposium on Fundamentals of Computation Theory*. Springer, Riga, Latvia, 2001, pp. 376–379.

[41] A. Díaz-Caro and A. Yakaryılmaz, "Affine computation and affine automaton," in *International Computer Science Symposium in Russia*. Springer, 2016, pp. 146–160.

[42] A. Yakaryilmaz and A. Say, "Language recognition by generalized quantum finite automata with unbounded error (abstract & poster)," *arXiv preprint: 0901.2703*, 2009.

[43] A. Yakaryılmaz, "Superiority of one-way and realtime quantum machines," *RAIRO-Theoretical Informatics and Applications*, vol. 46, no. 4, pp. 615–641, 2012.

[44] M. Nakanishi and A. Yakaryılmaz, "Classical and quantum counter automata on promise problems," in *CIAA '15: Proceedings of the 20th International Conference on Implementation and Application of Automata*. Springer, Umea, Sweden, 2015, pp. 224–237.

[45] A. Yakaryılmaz and A. C. Say, "Efficient probability amplification in two-way quantum finite automata," *Theoretical Computer Science*, vol. 410, no. 20, pp. 1932–1941, 2009.

[46] D. Qiu, "Some observations on two-way finite automata with quantum and classical states," in *ICIC '08: Proceedings of the 4th International Conference on Intelligent Computing*. Springer, Shanghai, China, 2008, pp. 1–8.

[47] S. Zheng, D. Qiu, and L. Li, "Some languages recognized by two-way finite automata with quantum and classical states," *International Journal of Foundations of Computer Science*, vol. 23, no. 05, pp. 1117–1129, 2012.

[48] A. Ambainis and N. Nahimovs, "Improved constructions of quantum automata," *Theoretical Computer Science*, vol. 410, no. 20, pp. 1916–1922, 2009.

[49] A. Ambainis and A. Yakaryılmaz, "Superiority of exact quantum automata for promise problems," *Information Processing Letters*, vol. 112, no. 7, pp. 289–291, 2012.

[50] M. P. Bianchi, C. Mereghetti, and B. Palano, "Complexity of promise problems on classical and quantum automata," in *Computing with New Resources*. Springer, 2014, pp. 161–175.

[51] J. Rashid and A. Yakaryılmaz, "Implications of quantum automata for contextuality," in *CIAA '14: Proceedings of the 19th International Conference on Implementation and Application of Automata*. Springer, Giessen, Germany, 2014, pp. 318–331.

[52] A. Yakaryilmaz and A. C. Say, "Succinctness of two-way probabilistic and quantum finite automata," *Discrete Mathematics and Theoretical Computer Science*, vol. 12, no. 4, pp. 19–40, 2010.

[53] S. Zheng, D. Qiu, J. Gruska, L. Li, and P. Mateus, "State succinctness of two-way finite automata with quantum and classical states," *Theoretical Computer Science*, vol. 499, pp. 98–112, 2013.

[54] J. Gruska, L. Li, and S. Zheng, "Recognizability versus solvability of promise problems in finite classical and quantum automata framework," *Computing Research Repository, abs/1411.3870 v2*, 2014.

[55] S. Zheng and D. Qiu, "From quantum query complexity to state complexity," in *Computing with New Resources*. Springer, 2014, pp. 231–245.

[56] S. Zheng, D. Qiu, and J. Gruska, "Time-space complexity advantages for quantum computing," in *International Conference on Theory and Practice of Natural Computing*. Springer, Prague, Czech Republic, 2017, pp. 305–317.

[57] S. Zheng, L. Li, D. Qiu, and J. Gruska, "Promise problems solved by quantum and classical finite automata," *Theoretical Computer Science*, vol. 666, pp. 48–64, 2017.

[58] S. Zheng, D. Qiu, L. Li, and J. Gruska, "One-way finite automata with quantum and classical states," in *Languages Alive, Lecture Notes in Computer Science*. Springer, 2012, vol. 7300, pp. 273–290.

[59] M. P. Bianchi, C. Mereghetti, and B. Palano, "On the power of one-way automata with quantum and classical states," in *CIAA '14: Proceedings of the 19th International Conference on Implementation and Application of Automata*. Springer, Giessen, Germany, 2014, pp. 84–97.

[60] A. Gainutdinova and A. Yakaryılmaz, "Unary probabilistic and quantum automata on promise problems," *Quantum Information Processing*, vol. 17, no. 2, p. 28, 2018.

[61] H. Nishimura and T. Yamakami, "An application of quantum finite automata to interactive proof systems," *Journal of Computer and System Sciences*, vol. 75, no. 4, pp. 255–269, 2009.

[62] S. Zheng, D. Qiu, and J. Gruska, "Power of the interactive proof systems with verifiers modeled by semi-quantum two-way finite automata," *Information and Computation*, vol. 241, pp. 197–214, 2015.

[63] A. Yakaryilmaz, A. C. Say, and H. G. Demirci, "Debates with small transparent quantum verifiers," *International Journal of Foundations of Computer Science*, vol. 27, no. 02, pp. 283–300, 2016.

[64] T. Yamakami, "One-way reversible and quantum finite automata with advice," *Information and Computation*, vol. 239, pp. 122–148, 2014.

[65] T. Yamakami, "Constant-space quantum interactive proofs against multiple provers," *Information Processing Letters*, vol. 114, no. 11, pp. 611–619, 2014.

[66] O. Scegulnaja-Dubrovska, L. Lāce, and R. Freivalds, "Postselection finite quantum automata," in *UC '10: Proceedings of the 9th International Conference on Unconventional Computation*. Springer, Tokyo, Japan, 2010, pp. 115–126.

[67] A. Yakaryilmaz and A. Say, "Probabilistic and quantum finite automata with postselection," *arXiv preprint:1102.0666*, 2011.

[68] M. Macko, "On closure properties of quantum finite automata," Ph.D. dissertation, Comenius University, Bratislava, 2006.

[69] A. Yakaryılmaz and A. C. Say, "Unbounded-error quantum computation with small space bounds," *Information and Computation*, vol. 209, no. 6, pp. 873–892, 2011.

[70] J. Watrous, "On the complexity of simulating space-bounded quantum computations," *Computational Complexity*, vol. 12, no. 1-2, pp. 48–84, 2003.

[71] L. Li and D. Qiu, "Determining the equivalence for one-way quantum finite automata," *Theoretical Computer Science*, vol. 403, no. 1, pp. 42–51, 2008.

[72] S. Zheng, L. Li, and D. Qiu, "Two-tape finite automata with quantum and classical states," *International Journal of Theoretical Physics*, vol. 50, no. 4, pp. 1262–1281, 2011.

[73] A. Ambainis, R. Bonner, R. Freivalds, and A. Ķikusts, "Probabilities to accept languages by quantum finite automata," in *COCOON '99: Proceedings of the International Computing and Combinatorics Conference*. Springer, Tokyo, Japan, 1999, pp. 174–183.

[74] S. Zheng, J. Gruska, and D. Qiu, "On the state complexity of semi-quantum finite automata," *RAIRO-Theoretical Informatics and Applications*, vol. 48, no. 2, pp. 187–207, 2014.

[75] M. O. Rabin and D. Scott, "Finite automata and their decision problems," *IBM Journal of Research and Development*, vol. 3, no. 2, pp. 114–125, 1959.

[76] A. L. Rosenberg, "On multi-head finite automata," *IBM Journal of Research and Development*, vol. 10, no. 5, pp. 388–394, 1966.

[77] M. Kutrib and A. Malcher, "One-way reversible multi-head finite automata," in *RC '12: Proceedings of the 4th International Workshop on Reversible Computation*. Springer, Copenhagen, Denmark, 2012, pp. 14–28.

[78] O. H. Ibarra, "On two-way multihead automata," *Journal of Computer and System Sciences*, vol. 7, no. 1, pp. 28–36, 1973.

[79] K. Morita, "Two-way reversible multi-head finite automata," *Fundamenta Informaticae*, vol. 110, no. 1-4, pp. 241–254, 2011.

[80] K. Morita, "A deterministic two-way multi-head finite automaton can be converted into a reversible one with the same number of heads," in *RC '12: Proceedings of the 4th International Workshop on Reversible Computation*. Springer, Copenhagen, Denmark, 2012, pp. 29–43.

[81] P. Duris and Z. Galil, "Fooling a two way automation or one pushdown store is better than one counter for two way machines," *Theoretical Computer Science*, vol. 21, no. 1, pp. 39–53, 1982.

[82] F. G. Jeronimo and A. V. Moura, "On the quantum-classical separation of marking and multi-head automata," *Universidade Estadual De Campinas, Technical report*, 2014.

[83] D. Ganguly, K. Chatterjee, and K. S. Ray, "1-way multihead quantum finite state automata," *Applied Mathematics*, vol. 7, no. 09, p. 1005, 2016.

[84] D. Ganguly and K. S. Ray, "2-tape 1-way quantum finite state automata," *arXiv preprint:1607.00811*, 2016.

[85] "Two-way deterministic finite automata, howpublished = https://en.wikipedia.org/wiki/two-way_deterministic_finite_automaton, note = Accessed: 2017-09-02."

[86] A. Okhotin and M. Kunc, "Reversible two-way finite automata over a unary alphabet," *Turku Centre for Computer Science, Technical Report*, no. 1024, 2011.

[87] M. Golovkins, "Quantum pushdown automata," in *SOFSEM: Proceedings of the 27th International Conference on Current Trends in Theory and Practice of Computer Science*. Springer, Milovy, Czech Republic, 2000, pp. 336–346.

[88] S. Gudder, "Quantum automata: An overview," *International Journal of Theoretical Physics*, vol. 38, no. 9, pp. 2261–2282, 1999.

[89] D. Qiu, "Quantum pushdown automata," *International Journal of Theoretical Physics*, vol. 41, no. 9, pp. 1627–1639, 2002.

[90] A. Cem Say and A. Yakaryilmaz, "Quantum counter automata," *International Journal of Foundations of Computer Science*, vol. 23, no. 05, pp. 1099–1116, 2012.

[91] E. L. Post, "Finite combinatory processes—formulation 1," *The Journal of Symbolic Logic*, vol. 1, no. 3, pp. 103–105, 1936.

[92] A. Pettorossi, *Elements of computability, decidability, and complexity*, 4th ed. Aracne editrice Srl, 2014.

[93] D. I. Cohen, *Introduction to computer theory*. Wiley New York, 1991, vol. 2.

[94] Z. Manna, *Mathematical Theory of Computation*. McGraw Hill, New York, 1974, vol. 41.

[95] F.-J. Brandenburg, "Multiple equality sets and post machines," *Journal of Computer and System Sciences*, vol. 21, no. 3, pp. 292–316, 1980.

[96] A. Cherubini, C. Citrini, S. C. Reghizzi, and D. Mandrioli, "Qrt fifo automata, breadth-first grammars and their relations," *Theoretical Computer Science*, vol. 85, no. 1, pp. 171–203, 1991.

[97] S. Jakobi, K. Meckel, C. Mereghetti, and B. Palano, "Queue automata of constant length," in *International Workshop on Descriptional Complexity of Formal Systems*. Springer, London, 2013, pp. 124–135.

[98] M. Kutrib, A. Malcher, and M. Wendlandt, "Reversible queue automata," *Fundamenta Informaticae*, vol. 148, no. 3-4, pp. 341–368, 2016.

[99] M. Kutrib, A. Malcher and M. Wendlandt, "Queue automata: Foundations and developments," in *Reversibility and Universality*. Springer, 2018, pp. 385–431.

[100] O. Scegulnaja, "Quantum Real-Time Turing Machine," in *FCT '01: Proceedings of the 13th International Symposium on Fundamentals of Computation Theory*. Springer, Riga, Latvia, 2001, pp. 412–415.

[101] S. Y. Yan, *Quantum attacks on public-key cryptosystems*. Springer, 2013.

[102] R. Devaraj, A. Sarkar, and S. Biswas, "Real-time scheduling of non-preemptive sporadic tasks on uniprocessor systems using Supervisory Control of timed DES," in *American Control Conference (ACC), 2017*. IEEE, 2017, pp. 3212–3217.

[103] R. Orús, "A practical introduction to tensor networks: Matrix product states and projected entangled pair states," *Annals of Physics*, vol. 349, pp. 117–158, 2014.

[104] S. R. White, "Density-matrix algorithms for quantum renormalization groups," *Physical Review B*, vol. 48, no. 14, p. 10345, 1993.

[105] R. Orús, "Symmetries, fermions, entanglement, and holography," *arXiv preprint:1407.6552*, 2014.

[106] A. Young, "Quantum finite state machines," *University of California, Davis*, 2014.

[107] K. Wiesner and J. P. Crutchfield, "Computation in finitary stochastic and quantum processes," *Physica D: Nonlinear Phenomena*, vol. 237, no. 9, pp. 1173–1195, 2008.

[108] D. Petz, "Entropy, Von Neumann and the Von Neumann entropy," in *John Von Neumann and the foundations of quantum physics*. Springer, 2001, pp. 83–96.

[109] W. R. de Oliveira, A. J. Silva, T. B. Ludermir, A. Leonel, W. R. Galindo, and J. C. Pereira, "Quantum logical neural networks," in *SBRN '08, Proceedings of the 10th Brazilian Symposium on Neural Networks*. IEEE, 2008, pp. 147–152.

[110] J. D. Biamonte, S. R. Clark, and D. Jaksch, "Categorical tensor network states," *AIP Advances*, vol. 1, no. 4, p. 042172, 2011.

[111] D. M. Greenberger, "GHZ (Greenberger—Horne—Zeilinger) Theorem and GHZ States," in *Compendium of quantum physics*. Springer, 2009, pp. 258–263.

[112] G. Uchida, "Geometry of GHZ type quantum states," Ph.D. dissertation, Uniwien, 2013.

[113] I. Affleck, T. Kennedy, E. H. Lieb, and H. Tasaki, "Rigorous results on valence-bond ground states in antiferromagnets," *Physical review letters*, vol. 59, no. 7, p. 799, 1987.

[114] R. Raussendorf, "Measurement-based quantum computation with cluster states," *International Journal of Quantum Information*, vol. 7, no. 06, pp. 1053–1203, 2009.

[115] "Matrix Product Formalism," http://www2.mpq.mpg.de/Theorygroup/CIRAC/wiki/images/9/9f/Eckholt_Diplom.pdf/, [Online; accessed 15-march-2018].

[116] J. P. Crutchfield and K. Wiesner, "Intrinsic quantum computation," *Physics Letters A*, vol. 372, no. 4, pp. 375–380, 2008.

[117] A. Silva, W. de Oliveira, and T. Ludermir, "A weightless neural node based on a probabilistic quantum memory," in *SBRN '10: Proceedings of the 11th Brazilian Symposium on Neural Networks*. IEEE, 2010, pp. 259–264.

[118] F. M. de Paula Neto, T. B. Ludermir, W. R. de Oliveira, and A. J. da Silva, "Fitting parameters on quantum weightless neuron dynamics," in *BRACIS '15: Proceedings of the Brazilian Conference on Intelligent Systems*. IEEE, Natal, Brazil, 2015, pp. 169–174.

[119] J. R. Büchi, "On a decision method in restricted second order arithmetic," in *The Collected Works of J. Richard Büchi*. Springer, 1990, pp. 425–435.

[120] D. E. Muller, "Infinite sequences and finite machines," in *SWCT '63: Proceedings of the 4th Annual Symposium on Switching Circuit Theory and Logical Design*. IEEE, Chicago, 1963, pp. 3–16.

[121] R. McNaughton, "Testing and generating infinite sequences by a finite automaton," *Information and control*, vol. 9, no. 5, pp. 521–530, 1966.

[122] M. O. Rabin, "Decidability of second-order theories and automata on infinite trees," *Transactions of the American Mathematical Society*, vol. 141, pp. 1–35, 1969.

[123] Q. Wang and M. Ying, "Quantum Büchi Automata," *arXiv preprint:1804.08982*, 2018.

[124] I. Dzelme-Bērziņa, "Quantum finite automata and logic," Ph.D. dissertation, University of Latvia, Riga, 2010.

[125] K. Giannakis, C. Papalitsas, and T. Andronikos, "Quantum automata for infinite periodic words," in *IISA '15: Proceedings of the 6th International Conference on Information, Intelligence, Systems and Applications*. IEEE, Corfu, Greece, 2015, pp. 1–6.

[126] I. Dzelme-Bērziņa, "Quantum finite state automata over infinite words," in *UC '10: Proceedings of the 9th International Conference on Unconventional Computation*. Springer, Tokyo, Japan, 2010, pp. 188–188.

[127] J. R. Büchi, "Weak second-order arithmetic and finite automata," *Mathematical Logic Quarterly*, vol. 6, no. 1-6, pp. 66–92, 1960.

[128] C. C. Elgot, "Decision problems of finite automata design and related arithmetics," *Transactions of the American Mathematical Society*, vol. 98, no. 1, pp. 21–51, 1961.

[129] A. Pnueli, "The temporal logic of programs," in *FOCS '77: Proceedings of the 18th Annual Symposium on Foundations of Computer Science*. IEEE, USA, 1977, pp. 46–57.

[130] S. B. Mandal, A. Chakrabarti, and S. Sur-Kolay, "Synthesis techniques for ternary quantum logic," in *ISMVL '11: Proceedings of the 41st IEEE International Symposium on Multiple-Valued Logic*. IEEE, Tuusula, Finland, 2011, pp. 218–223.

[131] M. Zimmermann, "Optimal bounds in parametric LTL games," *Theoretical Computer Science*, vol. 493, pp. 30–45, 2013.

[132] L. Zuck, "Past temporal logic," Ph.D. dissertation, The Weizmann Institute of Science, 1986.

[133] T. Wilke, "Classifying discrete temporal properties," in *STACS '99: Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science*. Springer, Trier, Germany, 1999, pp. 32–46.

[134] A. P. Sistla and E. M. Clarke, "The complexity of propositional linear temporal logics," *Journal of the ACM (JACM)*, vol. 32, no. 3, pp. 733–749, 1985.

[135] G. De Giacomo and M. Y. Vardi, "Linear temporal logic and linear dynamic logic on finite traces," in *IJCAI '13: Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, vol. 13. Beijing, China, 2013, pp. 854–860.

[136] M. Leucker and C. Sánchez, "Regular linear temporal logic," in *ICTAC '07: Proceedings of the 4th International Colloquium on Theoretical Aspects of Computing*. Springer, Macau, China, 2007, pp. 291–305.

[137] M. P. Schützenberger, "On finite monoids having only trivial subgroups," *Information and control*, vol. 8, no. 2, pp. 190–194, 1965.

[138] R. McNaughton and S. A. Papert, *Counter-Free Automata (MIT research monograph no. 65)*. The MIT Press, 1971.

[139] R. McNaughton, "The loop complexity of pure-group events," *Information and Control*, vol. 11, no. 1-2, pp. 167–176, 1967.

[140] J. A. W. Kamp, "Tense logic and the theory of linear order," Ph.D. dissertation, University of California, Los Angeles, 1968.

[141] R. García, "Prediction of rna pseudoknotted secondary structure using stochastic context free grammars (scfg)," *CLEI Electronic Journal*, vol. 9, no. 2, pp. 1–12, 2006.

[142] M. J. Sippl, *Biological sequence analysis: Probabilistic models of proteins and nucleic acids*. Protein Science, Cambridge University Press, 1999, vol. 8, no. 3.

[143] C. Chen, D. A. Ridzon, A. J. Broomer, Z. Zhou, D. H. Lee, J. T. Nguyen, M. Barbisin, N. L. Xu, V. R. Mahuvakar, M. R. Andersen, K. Q. Lao, K. J. Livak, and K. J. Guegler, "Real-time quantification of microrna by stem–loop rt–pcr," *Nucleic Acids Research*, vol. 33, no. 20, pp. e179–e179, 2005.

[144] A. Khrennikov and E. Yurova, "Automaton model of protein: dynamics of conformational and functional states," *Progress in biophysics and molecular biology*, vol. 130, pp. 2–14, 2017.

[145] K. Giannakis, C. Papalitsas, and T. Andronikos, "Quantum automata for infinite periodic words," in *IISA '15: Proceedings of the 6th International Conference on Information, Intelligence, Systems and Applications*. IEEE, Corfu, Greece, 2015, pp. 1–6.

[146] M. Pan, D. Qiu, and S. Zheng, "Global multipartite entanglement dynamics in Grover's search algorithm," *Quantum Information Processing*, vol. 16, no. 9, p. 211, 2017.

[147] K. Li, D. Qiu, L. Li, S. Zheng, and Z. Rong, "Application of distributed semi-quantum computing model in phase estimation," *Information Processing Letters*, vol. 120, pp. 23–29, 2017.

[148] R. Freivalds, "Probabilistic two-way machines," in *MFCS '81: Proceedings of the 10th International Symposium on Mathematical Foundations of Computer Science*. Springer, Czech Republic, 1981, pp. 33–45.

[149] C. Dwork and L. Stockmeyer, "A time complexity gap for two-way probabilistic finite-state automata," *SIAM Journal on Computing*, vol. 19, no. 6, pp. 1011–1023, 1990.

[150] J. C. Shepherdson, "The reduction of two-way automata to one-way automata," *IBM Journal of Research and Development*, vol. 3, no. 2, pp. 198–200, 1959.

[151] M. O. Rabin and D. Scott, "Finite automata and their decision problems," *IBM Journal of Research and Development*, vol. 3, no. 2, pp. 114–125, 1959.

[152] N. Kalra and A. Kumar, "State grammar and deep pushdown automata for biological sequences of nucleic acids," *Current Bioinformatics*, vol. 11, no. 4, pp. 470–479, 2016.

[153] K. Y. Sung, "The Use of Context-Sensitive Grammar for Modeling RNA Pseudoknots," in *BIOCOMP '06: Proceedings of the International Conference on Bioinformatics and Computational Biology*. CSREA Press, Las Vegas, USA, 2006, pp. 338–344.

[154] Y. Sakakibara, M. Brown, R. Hughey, and I. S. Mian, "The application of stochastic context-free grammars to folding, aligning and modeling homologous RNA sequences," University of California, Santa Cruz, CA, USA, Technical report, 1993.

[155] S. Kobayashi and T. Yokomori, "Modeling RNA secondary structures using tree grammars," *Genome Informatics*, vol. 5, pp. 29–38, 1994.

[156] E. Rivas and S. R. Eddy, "The language of RNA: a formal grammar that includes pseudoknots," *Bioinformatics*, vol. 16, no. 4, pp. 334–340, 2000.

[157] D. B. Searls, "The computational linguistics of biological sequences," *Artificial intelligence and molecular biology*, vol. 2, pp. 47–120, 1993.

[158] D. B. Searls, "String variable grammar: A logic grammar formalism for the biological language of DNA," *The Journal of Logic Programming*, vol. 24, no. 1-2, pp. 73–102, 1995.

[159] N. Mizoguchi, Y. Kato, and H. Seki, "A grammar-based approach to RNA pseudoknotted structure prediction for aligned sequences," in *ICCABS '11: Proceedings of the 1st International Conference on Computational Advances in Bio and Medical Sciences*. IEEE, Orlando, USA, 2011, pp. 135–140.

[160] L. Cai, R. L. Malmberg, and Y. Wu, "Stochastic modeling of RNA pseudoknotted structures: a grammatical approach," *Bioinformatics*, vol. 19, no. 1, pp. i66–i73, 2003.

[161] L. Kuppusamy, A. Mahendran, and S. N. Krishna, "Matrix insertion-deletion systems for bio-molecular structures," in *ICDCIT '11: Proceedings of the International Conference on Distributed Computing and Internet Technology*. Springer, Bhubaneswar, India, 2011, pp. 301–312.

[162] L. Kuppusamy, A. Mahendran, and É. V. de La Clergerie, *Modelling Intermolecular Structures and Defining Ambiguity in Gene Sequences using Matrix Insertion-Deletion Systems.* Biology, Computation and Linguistics, IOS Press, 2011, vol. 228.

[163] L. Kuppusamy and A. Mahendran, "Modelling DNA and RNA secondary structures using matrix insertion–deletion systems," *International Journal of Applied Mathematics and Computer Science*, vol. 26, no. 1, pp. 245–258, 2016.

[164] J. W. Anderson, P. Tataru, J. Staines, J. Hein, and R. Lyngsø, "Evolving stochastic context-free grammars for RNA secondary structure prediction," *BMC bioinformatics*, vol. 13, no. 1, p. 78, 2012.

[165] M. Knudsen, "Stochastic context-free grammars and RNA secondary structure prediction," Ph.D. dissertation, Bioinformatic Research Center, 2005.

[166] P. W. K. Rothemund, "A DNA and restriction enzyme implementation of Turing machines," *DNA based computers, DIMACS series in Discrete Mathematics and Theoretical Computer Science*, vol. 27, pp. 75–119, 1995.

[167] M. Cavaliere, N. Jonoska, S. Yogev, R. Piran, E. Keinan, and N. C. Seeman, "Biomolecular implementation of computing devices with unbounded memory," in *DNA '10: Proceedings of the 10th International Workshop on DNA-Based Computers.* Springer, Milan, Italy, 2004, pp. 35–49.

[168] T. Krasinski, S. Sakowski, and T. Poplawski, "Autonomous push-down automaton built on DNA," *Informatica, An International Journal of Computing and Informatics*, vol. 36, no. 3, pp. 263–276, 2011.

[169] A. Khrennikov and E. Yurova, "Automaton model of protein: dynamics of conformational and functional states," *Progress in biophysics and molecular biology*, 2017.

[170] J. C. Martin, *Introduction to Languages and the Theory of Computation.* McGraw-Hill New York, USA, 1991, vol. 4.

[171] K.-Y. Sung, "The Use of Context-Sensitive Grammar For Modeling RNA Pseudoknots," in *BIOCOMP '06: Proceedings of the International Conference on Bioinformatics and Computational Biology.* CSREA Press, Las Vegas, USA, 2006, pp. 338–344.

[172] T. J. Macke, D. J. Ecker, R. R. Gutell, D. Gautheret, D. A. Case, and R. Sampath, "RNAMotif, an RNA secondary structure definition and search algorithm," *Nucleic acids research*, vol. 29, no. 22, pp. 4724–4735, 2001.

[173] N. R. Council, *Mathematical challenges from theoretical/computational chemistry.* National Academies Press, 1995.

[174] S. Y. Yan, *Number theory for computing.* Springer Science & Business Media, 2002.

[175] D. J. Bernstein, "Introduction to post-quantum cryptography," in *Post-quantum cryptography.* Springer, 2009, pp. 1–14.

[176] L. Chen, L. Chen, S. Jordan, Y.-K. Liu, D. Moody, R. Peralta, R. Perlner, and D. Smith-Tone, *Report on post-quantum cryptography.* US Department of Commerce, National Institute of Standards and Technology, 2016.

[177] N. Cotfas, J. P. Gazeau, and K. Górska, "Complex and real Hermite polynomials and related quantizations," *Journal of Physics A: Mathematical and Theoretical*, vol. 43, no. 30, p. 305304, 2010.

[178] M. Cozzens and S. J. Miller, *The mathematics of encryption: an elementary introduction*. American Mathematical Society, 2013, vol. 29.

[179] D. M. Davis, *The nature and power of mathematics*. Courier Corporation, 2004.

[180] C. Pomerance, "The quadratic sieve factoring algorithm," in *Advances in Cryptology, EUROCRYPT, Lecture Notes in Computer Science*, vol. 209. Springer, 1984, pp. 169–182.

[181] C. Pomerance, "Smooth numbers and the quadratic sieve," in *Algorithmic Number Theory: Lattices, Number Fields, Curves and Cryptography: Proceedings of the Mathematical Sciences Research Institute (MSRI)*, vol. 44, 2008.

[182] C. Pomerance, "A tale of two sieves," *Biscuits of Number Theory*, vol. 85, p. 175, 2008.

[183] P. L. Montgomery, "Modular multiplication without trial division," *Mathematics of computation*, vol. 44, no. 170, pp. 519–521, 1985.

[184] S. Wright, *Quadratic residues and non-residues: selected topics*. Lecture Notes in Mathematics, Springer, 2016, vol. 2171.

[185] S. L. Garrett, *On the quadratic sieve*. Master Thesis, The University of North Carolina, Greensboro, 2008.

[186] R. Downey, *Turing's Legacy: Developments from Turing's ideas in logic*. Cambridge University Press, 2014, vol. 42.

[187] B. Kaliski Jr, "RSA factoring challenge," in *Encyclopedia of Cryptography and Security*. Springer, 2011, pp. 1064–1065.

[188] M. E. O'Neill, "The genuine sieve of Eratosthenes," *Journal of Functional Programming*, vol. 19, no. 1, pp. 95–106, 2009.

[189] C. Hoare, "Proof of a structured program:The sieve of Eratosthenes'," *The Computer Journal*, vol. 15, no. 4, pp. 321–325, 1972.

[190] R. Crandall and C. B. Pomerance, *Prime numbers: a computational perspective*. Springer Science & Business Media, 2006, vol. 182.

[191] C. Pomerance, "The quadratic sieve factoring algorithm," in *Advances in Cryptology: In Proceedings of EUROCRYPT '84. A Workshop on the Theory and Application of Cryptographic Techniques*. Springer, Paris, France, 1984, pp. 169–182.

[192] C. Pomerance, J. W. Smith, and R. Tuler, "A pipeline architecture for factoring large integers with the quadratic sieve algorithm," *SIAM Journal on Computing*, vol. 17, no. 2, pp. 387–403, 1988.

[193] J. I. Latorre and G. Sierra, "Quantum computation of prime number functions," *arXiv preprint:1302.6245*, 2013.

[194] B. Juliá-Díaz, J. M. Burdis, and F. Tabakin, "Qdensity—a mathematica quantum computer simulation," *Computer Physics Communications*, vol. 174, no. 11, pp. 914–934, 2006.

[195] T. Altenkirch and J. Grattage, "A functional quantum programming language," in *LICS '05: Proceedings of the 20th Annual IEEE Symposium on Logic in Computer Science*. IEEE, Chicago, IL, USA, 2005, pp. 249–258.

[196] A. K. Lenstra, H. W. Lenstra Jr, M. S. Manasse, and J. M. Pollard, "The number field sieve," in *STOC '90: Proceedings of the 22nd Annual ACM Symposium on Theory of computing*. ACM, Baltimore, MD, USA, 1990, pp. 564–572.

[197] K. Bimpikis and R. Jaiswal, "Modern factoring algorithms," *University of California, San Diego*, 2005.

[198] A. Ekert and R. Jozsa, "Quantum computation and Shor's factoring algorithm," *Reviews of Modern Physics*, vol. 68, no. 3, p. 733, 1996.

[199] D. N. Diep and D. H. Giang, "Quantum Gauss Jordan Elimination," *arXiv preprint quant-ph/0511062*, 2005.

[200] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[201] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[202] R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On data banks and privacy homomorphisms," *Foundations of secure computation*, vol. 4, no. 11, pp. 169–180, 1978.

[203] J. Buchmann and H. C. Williams, "A key-exchange system based on imaginary quadratic fields," *Journal of Cryptology*, vol. 1, no. 2, pp. 107–118, 1988.

[204] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of computation*, vol. 48, no. 177, pp. 203–209, 1987.

[205] J. Hoffstein, J. Pipher, and J. H. Silverman, "NTRU: A ring-based public key cryptosystem," in *ANTS '98: Proceedings of the 3rd International Algorithmic Number Theory Symposium*. Springer, Portland, Oregon, USA, 1998, pp. 267–288.

[206] R. J. Mceliece, "A public-key cryptosystem based on algebraic," *Coding Thv*, vol. 4244, pp. 114–116, 1978.

[207] J.-Y. Cai and T. W. Cusick, "A lattice-based public-key cryptosystem," in *SAC '98: Proceedings of the Annual International Workshop on Selected Areas in Cryptography*. Springer, Ontario, Canada, 1998, pp. 219–233.

[208] R. Overbeck and N. Sendrier, "Code-based cryptography," in *Post-quantum cryptography*. Springer, 2009, pp. 95–145.

[209] "Coding and Cryptography, Coding theory, notes," http://www.maths.uq.edu.au/courses/MATH3302/2010/files/codingnotes.pdf./, [Online; accessed 15-march-2018].

[210] C. Löndahl, "Some notes on code-based cryptography," Ph.D. dissertation, Lund University, 2015.

[211] V. M. Sidelnikov and S. O. Shestakov, "On insecurity of cryptosystems based on generalized Reed-Solomon codes," *Discrete Mathematics and Applications*, vol. 2, no. 4, pp. 439–444, 1992.

[212] N. Sendrier, "On the concatenated structure of a linear code," *Applicable Algebra in Engineering, Communication and Computing*, vol. 9, no. 3, pp. 221–242, 1998.

[213] H. Niederreiter, "Knapsack-type cryptosystems and algebraic coding theory," *Prob. Control and Inf. Theory*, vol. 15, no. 2, pp. 159–166, 1986.

[214] V. M. Sidelnikov, "A public-key cryptosystem based on binary Reed-Muller codes," *Discrete Mathematics and Applications*, vol. 4, no. 3, pp. 191–208, 1994.

[215] L. Minder and A. Shokrollahi, "Cryptanalysis of the sidelnikov cryptosystem," in *Advances in Cryptology: EUROCRYPT '07: Proceedings of the 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques.* Springer, Barcelona, Spain, 2007, pp. 347–360.

[216] H. Janwa and O. Moreno, "McEliece public key cryptosystems using algebraic-geometric codes," *Designs, Codes and Cryptography*, vol. 8, no. 3, pp. 293–307, 1996.

[217] C. Faure and L. Minder, "Cryptanalysis of the McEliece cryptosystem over hyperelliptic codes," in *Proceedings of the 11th international workshop on Algebraic and Combinatorial Coding Theory, ACCT*, vol. 2008, 2008, pp. 99–107.

[218] A. Couvreur, I. Márquez-Corbella, and R. Pellikaan, "A polynomial time attack against algebraic geometry code based public key cryptosystems," in *ISIT '04: Proceedings of the IEEE International Symposium on Information Theory.* IEEE, Illinois, USA, 2014, pp. 1446–1450.

[219] C. Monico, J. Rosenthal, and A. Shokrollahi, "Using low density parity check codes in the McEliece cryptosystem," in *ISIT: Proceedings of the IEEE International Symposium on Information Theory.* IEEE, Sorrento, Italy, 2000, p. 215.

[220] M. Baldi, F. Chiaraluce, R. Garello, and F. Mininni, "Quasi-cyclic low-density parity-check codes in the McEliece cryptosystem," in *ICC'07: Proceedings of the IEEE International Conference on Communications.* IEEE, Glasgow, Scotland, 2007, pp. 951–956.

[221] C. Löndahl and T. Johansson, "A new version of McEliece PKC based on convolutional codes," in *ICICS '12: Proceedings of the 14th International Conference on Information and Communications Security.* Springer, Hong Kong, 2012, pp. 461–470.

[222] G. Landais and J.-P. Tillich, "An efficient attack of a McEliece cryptosystem variant based on convolutional codes," in *PQCrypto '13: Proceedings of the 5th International Workshop on Post-Quantum Cryptography.* Springer, Limoges, France, 2013, pp. 102–117.

[223] J. I. Hall, *Notes on coding theory.* Michigan State University, USA, 2003.

[224] E. Berlekamp, "Decoding the Golay code," *Deep Space Network Progress Report*, vol. 11, pp. 81–85, 1972.

[225] D. Hankerson, G. Hoffman, D. A. Leonard, C. C. Lindner, K. T. Phelps, C. A. Rodger, and J. R. Wall, *Coding theory and cryptography: the essentials.* CRC Press, 2000.