University of Latvia

Ilze Dzelme-Bērziņa

Quantum Finite Automata and Logic

Doctoral Thesis

Area: Computer Science

Sub-Area: Mathematical Foundations of Computer Science

Scientific Advisor:

Dr. habil. math., Prof.

Rūsiņš Freivalds

Riga 2010

Abstract

The connection between the classical computation and mathematical logic has had a great impact in the computer science which is the main reason for the interest in the connection between the quantum computation and mathematical logic.

The thesis studies a connection between quantum finite state automata and logic. The main research area is a quantum finite state automaton and its different notations (measure-once quantum finite state automaton, measure-many quantum finite state automaton, and Latvian quantum finite state automaton), more precisely, the languages accepted by the various models of the quantum finite state automaton and its connection to languages described by the different kinds of logic (first order, modular etc.). Additionally, a quantum finite state automaton over infinite words is introduced.

The first part of the thesis is devoted to the connection between such quantum finite state automata as measure-once quantum finite state automata, measure-many quantum finite state automata, and Latvian quantum finite state automata and first order logic, modular logic, and generalized quantifiers - Lindström quantifier and group quantifier. For measure-once quantum finite state automata, we have characterized the language class accepted by measure-once quantum finite state automata in terms of logic using generalized quantifiers - Lindström quantifier and group quantifier, studied the relationship between the language class accepted by measure-once quantum finite state automata and the language class described by first order logic and modular logic. For measure-many quantum finite state automata, the connection between language classes accepted by quantum finite state automata and first order logic and modular logics has been studied, as well as the connection between acceptance probability of quantum finite state automata and logic. We also examined the language class accepted by Latvian quantum finite state automata in terms of logic.

The second part is devoted to the quantum finite state automata over infinite words. We extend the notation of quantum finite automata for infinite words. The class of languages accepted by Büchi quantum finite state automata has been studied and we examine the closure properties of Büchi quantum finite state automata.

Anotācija

Matemātiskās loģikas un klasiskās skaitļošanas saistībai ir bijusi liela nozīme datorzinātnes attīstībā. Tas ir galvenais iemesls, kas raisījis interesi pētīt kvantu skaitļošanas un loģikas saistību.

Promocijas darbs aplūko saistību starp galīgiem kvantu automātiem un loģiku. Pamatā pētījumi balstās uz galīgu kvantu automātu un tā dažādiem veidiem (galīgu kvantu automātu ar mērījumu beigās, galīgu kvantu automātu ar mērījumu katrā solī, galīgo "latviešu" kvantu automātu), precīzāk, valodām, ko akceptē dažādie kvantu automātu modeļi, un to saistību ar valodām, ko apraksta dažādie loģikas veidi (pirmās kārtas loģika, modulārā loģika u.c.). Darbā ir arī aplūkoti galīgi kvantu automāti, kas akceptē bezgalīgus vārdus.

Promocijas darba pirmā daļa ir veltīta galīga kvantu automāta ar mērījumu beigās, galīga kvantu automāta ar mērījumu katrā solī un galīgā "latviešu" kvantu automāta saistībai ar pirmās kārtas loģiku, modulāro loģiku un loģiku, kas izmanto vispārinātus kvantorus - Lindstroma kvantoru un grupas kvantoru. Galīgiem kvantu automātiem ar mērījumu beigās ir aprakstīta valodu klase, ko tie atpazīst, izmantojot vispārinātus kvantorus - Lindstroma kvantoru un grupas kvantoru, kā arī apskatīta galīga kvantu automāta ar mērījumu beigās saistība ar pirmās kārtas loģiku un modulāro loģiku. Galīgiem kvantu automātiem ar mērījumu katrā solī ir apskatīta to saistība ar pirmās kārtas loģiku un modulāro loģiku ne tikai no valodas atpazīšanas viedokļa, bet arī no galīga kvantu automāta ar mērījumu katrā solī akceptēšanas varbūtības viedokļa. Darbā aplūkota arī galīgā "latviešu" kvantu automāta saistība ar pirmās kārtas loģiku un modulāro loģiku, un aprakstīta valodu klase, izmantojot grupas kvantoru.

Otrā darba daļa ir veltīta kvantu automātiem bezgalīgiem vārdiem. Autors paplašina kvantu galīgā automāta definīciju bezgalīgiem vārdiem. Darbā aplūko valodu klasi, ko atpazīst Büchi galīgs kvantu automāts, kā arī Büchi galīga kvantu automāta slēgtību pret apvienojumu, šķēlumu un papildinājumu.

Preface

This thesis assembles the research performed by the author and reflected in the following publications:

1. Ilze Dzelme-Bērziņa. **Quantum Finite State Automata over Infinite Words.** *UC2010: 9th International Conference on Unconventional Computation*, Lecture Notes in Computer Science, vol 6079, pp. 188 - 188, 2010.

2. Ilze Dzelme-Bērziņa. **Mathematical logic and quantum finite state automata.** *Theoretical Computer Science vol 410(20): Quantum and Probabilistic Automata* , pp. 1952-1959, 2009.

3. Ilze Dzelme-Bērziņa. **First Order Logic and Acceptance Probability of Quantum Finite State Automata.** *DLT 2007 - 11th International Conference on Developments in Language Theory, Satellite Workshop on Probabilistic and Quantum Automata. Proceedings.*, TUCS General Publication No 45, pp. 29-36, 2007.

4. Ilze Dzelme. **Quantum Finite Automata and Logics.** *SOFSEM 2006: Theory and Practice of Computer Science, 32nd Conference on Current Trends in Theory and Practice of Computer Science*, Lecture Notes in Computer Science, vol 3831, pp. 246-253, 2006.

5. Ilze Dzelme-Bērziņa. **Modular Logic and Quantum Finite State Automata.** *Proceedings of the Asian Conference on Quantum Information Science 2006*, pp. 145-146, 2006.

6. Ilze Dzelme-Bērziņa. **Formulas of first order logic and quantum finite automata.** *Proceedings of 8th International conference on Quantum Communication, Measurement and Computing*, pp. 93-96, NICT Press, Japan, 2006.

7. Laura Mancinska, Maris Ozols, Ilze Dzelme-Berzina, Rubens Agadzanjans, Ansis Rosmanis. **Principles of Optimal Probabilistic Decision Tree Construction.**

International Conference on Foundations of Computer Science, CSREA Press, pp. 116–122, 2006.

8. Ansis Rosmanis, Ilze Dzelme-Berzina. **Mixed States in Quantum Cryptography.** International Conference on Foundations of Computer Science, CSREA Press, pp. 214–218, 2006.

The results of the thesis were presented at the following international conferences and workshops:

1. 13th Workshop on Quantum Information Processing QIP 2010, 15.-22.01.2010, Zürich, Switzerland. Poster **Latvian Quantum Finite State Automata and Logic**.

2. 11th International Conference on Developments in Language Theory. Satellite Workshop on Probabilistic and Quantum Automata (DLT 07), 3.-6.07.2007, Turku, Finland. Presentation **First Order Logic and Acceptance Probability of Quantum Finite State Automata**.

3. 8th International Conference on Quantum Communication, Measurement and Computing (QCMC 06), 28.11.-3.12.2006, Tsukuba, Japan. Poster **Formulas of first order logic and quantum finite automata**.

4. Asian Conference on Quantum Information Science 2006 (AQIS 06), 01.-04.09.2006, Beijing, China. Poster **Modular Logic and Quantum Finite State Automata**.

5. Theory and Practice of Computer Science, 32nd Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2006), 21.-27.01.2006, Merin, the Czech Republic. Presentation **Quantum Finite Automata and Logics**.

Many people have contributed to the successful completion of the thesis. My gratitude to my supervisor, professor Rūsiņš Freivalds, who encouraged to start the research and whose support helped me to complete the work. Many thanks to Andris Ambainis for his support and helpfulness.

Finally, my gratitude to the colleagues, friends, and relatives, especially to my daughter, husband, parents, and sister whose support has encouraged me and without whom the thesis would not be finished.

Contents

Chapter 1

Introduction

The rapid development of the quantum computation and the huge impact of mathematical logic in the classical computation were the main reasons to study the relationship between quantum finite state automata and mathematical logic.

The connection between logic and the classical automata theory started with the work of Büchi [17] and Elgot [22]. They showed how a logical monadic second-order formula can effectively be transformed into a finite state automaton accepting the language defined by the formula and vice versus - how a finite state automaton can be transformed to a logical monadic second order formula which specifies the language accepted by the automaton. The logical description of the computation models' behaviour also influenced complexity theory. In 1974, Fagin [23] gave a characterization of non-deterministic polynomial time (NP) as the set of properties expressible in the second order existential logic.

The above results inspired us to study the connection between the quantum automata theory and logic. The goal of the research was to describe language classes recognized by different quantum automaton models using logical formulas, to find properties of quantum automata that can be connected to logics. We have achieved the following:

- characterized the language class accepted by measure-once quantum finite state automata with bounded error in the terms of logic;
- proved that intersection of the language class accepted by measure-once quantum finite automata with bounded error and languages defined by $FO[<]$ contains only trivial languages, i.e., an empty language and Σ^*;
- proved that languages described by modular logic using only modular quantifiers are recognized by measure-once quantum finite state automata;

- studied the connection between languages accepted by measure-many quantum finite state automata and first order logic, as well as, the connection between acceptance probability of measure-many quantum finite state automata and first order logic was examined;
- studied the connection between acceptance probability of measure-many quantum finite state automata and modular logic using first order quantifiers;
- studied the connection between Latvian quantum finite state automata and logic.

The theory of classical automata over infinite strings has been applied in the various research areas, including the verification of reactive systems, reasoning about infinite games, and decision problems for certain logics. Our research of the connection between finite state automata and logic lead us to automata over infinite words. We devoted the second part of the thesis to quantum finite state automata over infinite words. We have achieved the following:

- extended definitions of group automata and measure-once quantum finite state automata to infinite words;
- defined Büchi, Streett, and Rabin acceptance conditions for quantum case;
- proved that our Büchi quantum finite state automata over infinite words with bounded error recognize the limit of the group languages;
- proved that our Büchi quantum finite state automata is closed under union;
- showed that our Büchi quantum finite state automata is not closed under intersection and complementation;
- defined a measure-many quantum finite state automaton over infinite input.

In the Chapter 1, the background to the quantum computation and to the connection between mathematical logic and classical computation is presented. We give a brief history of development of the quantum computation, quantum Turing machine, and quantum automata. The section "Logics and classical computation" is devoted to the main results in the connection between classical computation and logic. And an overview of automata over infinite words is presented in the section "Automata over infinite words".

The Chapter 2 contains main notations and definitions used in the thesis - probabilistic systems, brief introduction to quantum mechanics, notations of quantum finite state

automata used in the thesis, a brief overview to mathematical logic, algebra, and classical computation, as well as, introduction to automata over infinite words.

The Chapter 3 is devoted to connection between measure-once quantum finite state automata and logic - first order logic, modular logic, and logic using generalized quantifiers - Lindström quantifier and group quantifier. The connection between measure-many quantum finite state automata and first order logic and modular logic is studied in the Chapter 4. In the Chapter 5, we study Latvian quantum finite state automata and its connection to logic.

The Chapter 6 is devoted to the quantum finite state automata over infinite words, where we give a definition of quantum finite state automata over infinite words and obtained results.

1.1 Quantum Computation

Over the past half century, the power of computers has doubled every year and a half. This phenomena is known as "Moore's law", named after Gordon Moore, who had stated the exponential advance in the 1960's [39]. If the Moore's law is to be sustained then we must learn to build a computer based on the quantum physics (quantum computers represent the ultimate level of miniaturization) and study quantum computation.

Quantum computation investigates the computation power and other properties of the computers based on quantum mechanic principles. The concept of quantum computing dates back to the early 80's, to the speech of Nobel Prize winner Richard P. Feynman and accompanying paper "Simulating physics with computers" [24]. His paper was the first work that explicitly discussed the construction of machine operating according to the laws of quantum physics. Feynman discussed the idea of a universal simulator - a machine using quantum effects to explore other quantum effects and run simulations.

In 1982 [13], Benioff described a quantum mechanical computation model. However, his model did not use any quantum mechanical effects, it was a hybrid Turing machine storing qubits on the tape instead of classical bits and measuring each qubit from the tape at each step. A year later in [3], David Albert described a self measuring quantum automaton that performed tasks which no classical computer could simulate. However, the machine was largely unspecified. By instructing the automaton to measure itself, it can obtain 'subjective' information that is absolutely inaccessible by measurement from the outside. Finally in 1985 [20], Deutsch introduced a fully quantum computational model and gave the description of a *universal quantum computer*. Later [14], Bernstein and Vazirani introduced the construction of a *universal quantum Turing machine* capable

of simulating any other quantum Turing machine with polynomial efficiency.

After the Deutsch paper [20] in the following years, there was a small interest in quantum computation. However during these years, one of the first examples of a quantum algorithm (The Deutsch - Jozsa algorithm [21]) that is more efficient than any possible classical algorithm has been presented, and Dan Simon [50] invented an oracle problem for which a quantum computer would be exponentially faster than a conventional computer. The main ideas of the algorithm were later developed in Shor's factoring algorithm [49].

Interest in the quantum computation increased dramatically, when in 1994 Peter Shor discovered efficient quantum algorithms for the problems of integer factorization and discrete logarithms [49]. The importance of the Shor's algorithm for finding factors is in fact that reliability of the widely used public-key cryptography schema RSA is based on the assumption that factoring large numbers is computationally infeasible. Shor demonstrated that it is not the case if we could build a quantum computer. Shor's results are the powerful evidence, that quantum computers are more powerful than Turing machines, even probabilistic Turing machines. Further evidence of the quantum automata's power came in 1996 when Grover showed, that the problem of conducting a search through some unstructured search space could be sped up by a quantum computer [26]. Although Grover's algorithm did not had as powerful speed-up as Shor's algorithm, the widespread applications of search-based methodologies has excited considerable interest in the algorithm of Grover.

However, the theory of quantum computers is far more developed than the practice: a large scale quantum computer has not been built yet. In 1998, the first working 2-qubit Nuclear magnetic resonance (NMR) quantum computer was demonstrated by Jonathan A. Jones and Michele Mosca at Oxford University. In the same year, the first working 3-qubit quantum computer has been developed and the first execution of Grover's algorithm on an NMR computer has been performed. In 2006, the theorists and experimentalists at the Institute for Quantum Computing and Perimeter Institute for Theoretical Physics in Waterloo, along with MIT, Cambridge, have presented an operational control method in quantum information processing extending up to 12 qubits.

The more developed field of quantum theory is quantum cryptography. Although large scale quantum computers are not built, the quantum devices for cryptography have been provided. Quantum cryptography was first proposed by Stephen Wiesner, who, in the early 1970s, introduced the concept of quantum conjugate coding. His paper "Conjugate Coding" was rejected by IEEE Information Theory, but it was eventually published in 1983 in SIGACT News (15:1 pp. 78-88, 1983). In the early 1980s, Charles H. Bennett

and Gilles Brassard proposed a method for secure communication based on Wiesner's "conjugate observables". In 1990, independently and initially unaware of the earlier work, Artur Ekert developed a different approach to quantum cryptography based on peculiar quantum correlations known as quantum entanglement. Quantum cryptography is also used in practice.

- Quantum encryption technology provided by the Swiss company Id Quantique was used in the Swiss canton of Geneva to transmit ballot results to the capitol in the national election in 2007.
- In 2004, the world's first bank transfer using quantum cryptography was carried out in Austria. An important cheque, which needed absolute security, was transmitted from the Mayor of the city to an Austrian bank.
- The world's first computer network SECOQC (Secure Communication Based on Quantum Cryptography) protected by quantum cryptography was implemented in October 2008, at a scientific conference in Vienna. The network used 200 km of standard fibre optic cable to interconnect six locations across Vienna and the town of St Poelten located 69 km to the west.

For more information about quantum computation please refer to [28], [27], and [41], but now we are going to consider the main models of quantum computation - a quantum Turing machine and a quantum automaton. In comparison with quantum Turing machine, a quantum automaton has finite memory and the computation steps does not exceed the length of the input.

1.1.1 Quantum Turing Machine

The beginnings of the modern computer science goes back to a remarkable paper [54] of Alan Turing, known as father of modern computer science, written in 1936. He developed an abstract notation of computation now known as *Turing machine*. Turing showed that there is a *Universal Turing machine* that can be used to simulate any other Turing machine. The Turing machine later evolved into the modern computer.

Quantum computation also has its Turing machine - quantum Turing machine. Quantum Turing machine is an abstract machine which is used to model the effect of the quantum computation. Any quantum algorithm can be expressed using a particular quantum Turing machine. We may say that quantum Turing machines have the same relation to the quantum computation as Turing machines have in the classical computation. As already mentioned, the first quantum computational model (quantum Turing machine)

was proposed by Deutsch [20]. Afterwards [14], Bernstein and Vazirani introduced the construction of a *universal quantum Turing machine* capable of simulating any other quantum Turing machine with polynomial efficiency.

However, quantum Turing machines are not always used for the quantum computation analysis, in quantum information theory a quantum circuit is more commonly used model. It has been proved that quantum Turing machines and quantum circuits are computationally equivalent. [19]

Iriyama, Ohya, and Volovich have developed a model of a Linear Quantum Turing Machine [31]. This is a generalization of a classical quantum Turing machine that has mixed states and that allows irreversible transition functions. The quantum Turing machine with postselection was defined by Scott Aaronson, who showed that the class of polynomial time on such a machine is equal to the classical complexity class PP (the class of decision problems solvable by a probabilistic Turing machine in polynomial time) [2]. Another model of quantum Turing machine is the classically-controlled quantum Turing machine - a Turing machine with a quantum tape for acting on quantum data, and a classical transition function for a formalized classical control was introduced by Perdrix and Jorrand in [43], where they showed that any classical Turing machine can be simulated by a classically-controlled quantum Turing machine without loss of efficiency.

1.1.2 Quantum Automata

A natural model of classical computing with finite memory is a finite state automaton, likewise a quantum finite state automaton is a natural model of quantum computation. Quantum finite state automata refer to the quantum computers in a similar way as finite state automata are related to Turing machines. An automaton reads input symbols from the given input and performs a transition function defined for the input symbol, after the input is read, the automaton accepts or rejects an input word. A quantum automaton can reject or accept a word with a probability between zero and one.

Different notations of quantum finite state automata are used. The most simple and one of the most popular notation of quantum finite state automata is a definition of a quantum finite state automaton introduced by Moore and Crutchfield [38] known as measure-once quantum finite state automaton (MO-QFA). The measure-once quantum finite state automata performs unitary transformation for each input symbol and makes the only measurement when the whole word is read obtaining the result whether the input is accepted or rejected. MO-QFA is pure state model of quantum finite state automata. Brodsky and Pippenger have proved [16], that MO-QFA with bounded error recognize the same language class as group automata [52]. In the same paper [16], Brodsky and

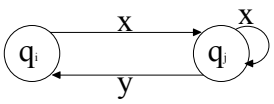

Figure 1.1 The forbidden construction of [16].

Pippenger showed that measure-once quantum finite state automaton can be simulated by a probabilistic finite state automaton and described an algorithm that determines if two measure-once quantum finite state automata are equivalent. However, if we consider the measure-once quantum finite state automata with unbounded error it can recognize non-regular languages, for example, $L_1 = \{\omega | \omega \in \{0,1\}^*, |\omega|_0 = |\omega|_1\}$.

Another widely used notation of quantum finite state automata is a measure-many quantum finite state automaton (MM-QFA) introduced by Kondacs and Watrous [32]. MO-QFA and MM-QFA have seemingly small difference, the definition of measure-once quantum finite state automata allows the measurement only at the end of the computation, but the definition of measure-many quantum finite state automata allows the measurement at every step of the computation (the measurement provides a probabilistic decision on every input symbol by projecting a state on the subspace of accepting states, the subspace of rejecting states, and the subspace of the automaton's non-halting states). The computation of MM-QFA halts when an accepting state or a rejecting state is reached. While a measure-many quantum finite state automaton is in a non-halting state, the computation continues. Although MM-QFA is more powerful than MO-QFA, measure-many quantum finite state automata with bounded error recognise only the subset of the regular languages. The several attempts have been made to define the language class of measure-many quantum finite state automata. Brodsky and Pippenger [16] introduced the first forbidden construction of MM-QFA (see the figure 1.1) - if a minimal deterministic finite state automaton of a language contains such construction then the language cannot be recognized by a measure-many quantum finite state automaton. Later Ambainis, Ķikusts and Valdats [7] have shown another forbidden construction (the figure 1.2) for measure-many quantum finite state automata, but it is still an open question how to characterize the language class accepted by measure-many quantum finite state automata.

In the thesis, we also consider a notation of Latvian quantum finite automata which was introduced in [1]. A Latvian quantum finite state automaton uses mixed states, at every step of the computation an automaton performs a unitary transformation and a projection defined for each input symbol. It has been provided an algebraic characterization of the languages recognized by Latvian quantum finite state automata. Enhanced quantum finite state automata is the measure-many version of Latvian quantum

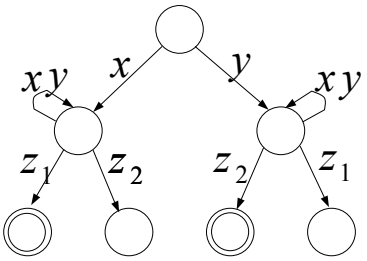

Figure 1.2 The forbidden construction of [7].

finite state automata, it was defined in [8], where it was shown that there are languages for which enhanced quantum finite automata take exponentially more states than those of the corresponding classical automata.

There are also other notations of quantum finite state automata as:

- one-way quantum finite automata with control language [15] - the accepting behaviour is controlled by the result of the projective measurement performed at each step in the computation, it was proved that one-way quantum finite automata with control language with bounded error recognize exactly regular languages [36];
- one-way quantum finite automata together with classical states with bounded error accepting all regular languages [46];
- ancilla quantum finite state automata, where an ancilla quantum part is imported, and then the internal control states and the states of the ancilla part together evolve by a unitary transformation [42];
- measure-once one-way general quantum finite state automata and measure-many one-way general quantum finite state automata [33] instead of a unitary transformation a trace-preserving quantum operation is used.

1.2 Logics and Classical Computation

The connection between automata theory and logic dates back to the early sixties to the work of Büchi [17] and Elgot [22], who showed that the finite automata and monadic second order logic (interpreted over finite words) have the same expressive power, and that the transformation from logical monadic second order formulas to finite state automata and vice versus are effective. Later, the equivalence between finite state automata and monadic second order logic over infinite words and trees were shown in the works of

Büchi [18], McNaughton [35], and Rabin [47]. The reduction of formulas to finite state automata was the key to prove decidability results in mathematical theories.

The next important step in the connection between automata theory and logic was Pnueli's work [45]. It was proposed to use temporal logic for reasoning about continuously operating concurrent programs. In the eighties, temporal logics and fixed-point logics took the role of specification languages and more efficient transformations from logic formulas to automata were found. This led to powerful algorithms and software systems for the verification of finite state programs ("model-checking"). The research of the equivalence between automata theory and logic formalism also influenced language theory itself. For example, the logical approach helped generalizing the domain of words to more general structures like trees and partial orders.

The logical description of the computation models' behaviour also influenced complexity theory. In 1974, Fagin [23] gave a characterization of non-deterministic polynomial time as the set of properties expressible in the second order existential logic. Later, Immerman [29] and Vardi [55] characterized polynomial time as the set of properties expressible in the first order inductive definition, which is defined by adding the least point operator to the first order logic. In the similar way, polynomial space has also been characterized [30] as second order logic with transitive closure. These results led to the development of a new field - Description complexity - a sub field of computational complexity theory and mathematical logic, which seeks to characterize complexity classes by the type of logic needed to express the language in them.

A merge of techniques and results from automata theory, logic, and complexity was achieved in circuit complexity theory, which studies the computational power of boolean circuits, regarding restrictions in their size, depth, and types of allowed gates. Natural families of circuits can be described by generalized models of finite state automata as well as by appropriate systems of the first order logic.

1.3 Automata over Infinite Words

The study of finite state automata working on infinite words was initiated by Büchi [17]. Büchi discovered connection between formulas of the monadic second order logic of infinite sequences (S1S) and ω-regular languages, the class of languages over infinite words accepted by finite state automata. The complexity of theory of automata over infinite words was evident from the inital work of Büchi, where he showed that non-deterministic automata over infinite words are strictly more powerful than deterministic automata over infinite words.

Few years later after Büchi paper, Muller proposed an alternative definition of finite automata on infinite words [40]. McNaughton proved that with Muller's definition, deterministic automata recognizes all ω-regular languages [35]. Later, Rabin extended decidability result of Büchi for S1S to the monadic second order of the infinite binary tree (S2S) [47]. Rabin's theorem can be used to settle a number of decision problems in logic.

A theory of automata over infinite words has started from these studies. It can be applied in the various research areas, including the verification of reactive systems, reasoning about infinite games, and decision problems for certain logics.

Recently probabilistic variants of finite state automata over infinite words have been introduced and studied in [12], [10], and [11].

Chapter 2

Preliminaries

The chapter provides the main notations, definitions, and results of the quantum computation and connection between logic and classical computation which are going to be helpful for the rest of the thesis. Additionally, we give definitions of the classical finite state automata over infinite words.

The most of the definitions we refer to [28], [27], and [41] for Quantum Computation. For the main results and definitions of the connection between logic and automata and automata over infinite words, we refer to [25] and [53].

2.1 Probabilistic Systems

A system admitting probabilistic nature means that we do not know for certain the state of the system, but we know that the system is in the states $x_1, ..., x_n$ with probabilities $p_1, ..., p_n$ that sum up to 1.

Definition 2.1.1. *Notation*

$$p_1[x_1] + p_2[x_2] + ... + p_n[x_n] \tag{2.1}$$

where $p_i \geq 0$ and $p_1 + p_2 + ... + p_n = 1$ stands for a probability distribution, meaning that the system is in state x_i with probability p_i.

We also call distribution 2.1 a *mixed state*. States x_i are called *pure states*.

Example 2.1.1. *Tossing a fair coin will give head h or tail t with a probability of $\frac{1}{2}$. The notation $\frac{1}{2}h + \frac{1}{2}t$ is the mixed state of a fair coin tossing.*

Instead of dealing with only one possible "reality" of how the process might evolve under time, in probabilistic system, there is some indeterminacy in its future evolution

described by the probability distributions. There are many possibilities the process might go to, but some paths are more probable and others are less.

The time evolution of a probabilistic system develops each state x_i into distribution

$$x_i \mapsto p_{1i}[x_1] + p_{2i}[x_2] + ... + p_{ni}[x_n], \tag{2.2}$$

such that $p_{1i} + p_{2i} + ... + p_{ni} = 1$ for each i. In the notation 2.2, p_{ji} is the probability that the system x_i into x_j. Thus a distribution

$$p_1[x_1] + p_2[x_2] + ... + p_n[x_n] \tag{2.3}$$

evolves into

$$\begin{aligned} p_1(p_{11}[x_1] + ... + p_{n1}[x_n]) + ... + p_n(p_{1n}[x_1] + p_{2i}[x_2] + ... + p_{nn}[x_n]) = \\ = (p_{11}p_1 + ... + p_{1n}p_n)[x_1] + ... + (p_{n1}p_1 + ... + p_{nn}p_n)[x_n] = \\ = p'_1[x_1] + ... + p'_n[x_n], \end{aligned}$$

where $p'_i = p_{1i}p_1 + ... + p_{1i}p_n$. Therefore the probabilities p_i and p'_i are related by

$$\begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{n1} & p_{n2} & \cdots & p_{nn} \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{pmatrix} = \begin{pmatrix} p'_1 \\ p'_2 \\ \vdots \\ p'_n \end{pmatrix}. \tag{2.4}$$

The matrix in the equation 2.4 is a stochastic matrix, also called Markov matrix, it has non-negative entries and $p_{1i} + p_{2i} + ... + p_{ni} = 1$ for each i, which guaranties that $p'_1 + p'_2 + ... + p'_n = p_1 + p_2 + ... + p_n$.

Definition 2.1.2. *A real (n x n) matrix $A = [a_{ij}]$ is called a Markov matrix or stochastic matrix if*

- $a_{ij} > 0$ *for* $1 \leq i, j \leq n$
- $\sum_{i=0}^{n} a_{ij} = 1$ *for* $1 \leq j \leq n$.

A probabilistic system with a time evaluation described above is called a Markov chain.

2.2 Quantum Mechanics

Here, we give a brief introduction to quantum mechanics. The term "quantum mechanics" is used for the mathematical structure describing "quantum physics". Quantum mechanics was born in the beginnings of the twentieth century, when experiments on atoms and radiations could not be fully explained by classical physics even by using the Markov chain described in the previous section. At first, we introduce the formalism of quantum mechanics in a basic form on state vectors and we assume that the quantum system is a finite - dimensional.

2.2.1 Superposition

The quantum mechanical description of a physical system looks similar to description of a probabilistic system in 2.1. However, they differ essentially. In quantum mechanics, a state of an n-level system is described as a unit-length vector in an n-dimensional complex vector space H_n called Hilbert space (see Subsection 2.2.2). H_n is called *the state space* of the system. To define a quantum state we choose an orthonormal basis $\{|x_1\rangle, |x_2\rangle, ..., |x_n\rangle\}$ [1] for the Hilbert space H_n.

Definition 2.2.1. *A pure quantum state $|\phi\rangle$ is a superposition of a classical states, written*

$$|\phi\rangle = \alpha_1|x_1\rangle + \alpha_2|x_2\rangle + ... + \alpha_n|x_n\rangle, \tag{2.5}$$

where α_i are complex numbers called amplitudes (with respect to the chosen basis) and $|\alpha_1|^2 + |\alpha_3|^2 + ... + |\alpha_n|^2 = 1$, where $|\alpha_j|^2$ is the squared norm of the corresponding amplitude α_j ($|ai + b| = \sqrt{a^2 + b^2}$).

The quantum state 2.5 can be also seen as the n-dimensional column vector

$$\begin{pmatrix} \alpha_1 \\ \alpha_2 \\ ... \\ \alpha_n \end{pmatrix}. \tag{2.6}$$

Example 2.2.1. *A two-level quantum system can be used to represent a bit. Such a system is called a quantum bit or qubit. The system has an orthonormal basis $\{|0\rangle, |1\rangle\}$. A general state of the system is represented by $\alpha_0|0\rangle + \alpha_1|1\rangle$, where $|\alpha_1|^2 + |\alpha_2|^2 = 1$. The probability to see the system to have the property 0 is $|\alpha_0|^2$, and 1 - $|\alpha_1|^2$.*

[1] Notation like '$|x\rangle$' is standard notation for states in quantum mechanics and is called "ket" notation or Dirac notation.

2.2.2 Hilbert Space

Hilbert space is a mathematical framework suitable for describing the concepts and principles of quantum system. In this subsection, we will define Hilbert space.

Definition 2.2.2. *An inner-product space H is a complex vector space, equipped with an inner product $\langle\cdot|\cdot\rangle : H \times H \to C$ satisfying the following axioms for any vectors $\phi, \psi, \phi_1, \phi_2 \in H$, and any $c_1, c_2 \in C$.*

1. $\langle\phi|\psi\rangle = \langle\psi|\phi\rangle^{*}$[2],
2. $\langle\psi|\psi\rangle \geq 0$ *and* $\langle\psi|\psi\rangle = 0$ *if and only if* $\psi = 0$,
3. $\langle\psi|c_1\phi_1 + c_2\phi_2\rangle = c_1\langle\psi|\phi_1\rangle + c_2\langle\psi|\phi_2\rangle$.

The inner product introduces on H the norm or length $\|\psi\| = \sqrt{\langle\psi|\psi\rangle}$ and the metric (Euclidean distance) $dist(\phi, \psi) = \|\phi - \psi\|$.

Definition 2.2.3. *An inner-product vector space H is called complete if for each vector sequence ψ_i such that $\lim_{m,n\to\infty} \|\psi_m - \psi_n\| = 0$, there exists a vector $\psi \in H$ such that $\lim_{n\to\infty} \|\psi_n - \psi\| = 0$. A complete inner-product space is called a Hilbert space.*

An n-dimensional complex vector Hilbert space is denoted by H_n or C^n.

A vector ψ of an n-dimensional Hilbert space is denoted by $|\psi\rangle$, and is referred as a ket-vector, and it can be seen as an n-dimensional column vector 2.6. The $\langle\psi|$ is referred to as bra-vector and can be seen as an n-dimensional row vector

$$\begin{pmatrix} \alpha_1^* & \alpha_2^* & \dots & \alpha_n^* \end{pmatrix}, \tag{2.7}$$

where α_i are complex numbers. The transformation $|\psi\rangle \leftrightarrow \langle\psi|$ corresponds to transposition and conjunction.

The inner product $\langle\psi|\phi\rangle$ is then "row vector $\times$ column vector" product, which produces a complex number as output. The outer product $|\psi\rangle\langle\phi|$ is an $n \times n$ matrix - "column vector $\times$ row vector" product.

Example 2.2.2. *Let us consider a two states of the H_2: $|\psi_1\rangle = \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix}$ and $|\psi_2\rangle = \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix}$. The inner product of ψ_1 and ψ_2 is*

$$\langle\psi_1|\psi_2\rangle = \begin{pmatrix} \alpha_0^* & \alpha_1^* \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} = \alpha_0^*\beta_0 + \alpha_1^*\beta_1$$

[2]Notation z^* denotes the complex conjugate of the complex number z: $(a + bi)^* = a - bi$.

and the outer product is

$$|\psi_1\rangle\langle\psi_2| = \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} \begin{pmatrix} \beta_0^* & \beta_1^* \end{pmatrix} = \begin{pmatrix} \alpha_0\beta_0^* & \alpha_0\beta_1^* \\ \alpha_1\beta_0^* & \alpha_1\beta_1^* \end{pmatrix}.$$

2.2.3 Observables and Measurement

In order to extract quantum information from a quantum system, we have to observe the system to perform a measurement.

At first, we consider a measurement in the computational basis. Suppose, that a quantum system is in a state

$$|\phi\rangle = \alpha_1|x_1\rangle + \alpha_2|x_2\rangle + ... + \alpha_n|x_n\rangle.$$

We cannot "see" a superposition itself, but only classical states. The basis state $|x_j\rangle$ is *observed* with probability of $|\alpha_j|^2$, which is the squared norm of the corresponding amplitude α_j ($|ai + b| = \sqrt{a^2 + b^2}$). Observing $|\phi\rangle$ collapses the quantum state $|\phi\rangle$ to the classic state $|x_j\rangle$ and all the "information" that might have been contained in the α_j is gone.

We may generalize the measurement as follows:

Definition 2.2.4. *An observable $E = \{E_1, E_2, ..., E_m\}$ ($m \leq n$) is a collection of mutually orthogonal subspaces of the Hilbert space H_n such that*

$$H_n = E_1 \oplus E_2 \oplus ... \oplus E_m.$$

We equip the subspaces E_i with distinct real number "labels" $\theta_1, \theta_2, ..., \theta_m$. Each vector $|x\rangle \in H_n$ can be decomposed in a unique way as $|x\rangle = |x_1\rangle + |x_2\rangle + ... + |x_m\rangle$ such that $|x_i\rangle \in E_i$. Instead of observing subspaces E_i, we can talk about observing the labels θ_i: by observing E, value θ_i will be seen with a probability of $\|x_i\|^2$.

Example 2.2.3. *The observable E can be defined as $E = \{E_1, E_2, ..., E_n\}$, where each E_i is the one-dimensional subspace spanned to x_i. Now, we equip the subspaces E_i with the label i. If the subspace is in the state 2.5, the value i is observed with a probability of $\|\alpha_i x_i\| = |\alpha_i|^2$.*

Another way of viewing the observables is using projectors. The measurement is described by projectors $P_1, P_2, ..., P_m$ ($m \leq n$), which sum to identity. The projectors are orthogonal ($P_iP_j = 0$, if $i \neq j$). The projector P_j projects on the subspace E_j of the Hilbert space H_n, and every vector $|x\rangle \in H_n$ can be decomposed in a unique way as

$|x\rangle = |x_1\rangle + |x_2\rangle + ... + |x_m\rangle$ such that $|x_j\rangle = P_j|x\rangle \in E_j$. We will get outcome θ_j with probability $\|x_j\|^2 = Tr(P_j|x\rangle\langle x|)$ and the state will collapse to the new state

$$|x'\rangle = \frac{|x_j\rangle}{\|x_j\|} = \frac{P_j|x\rangle}{\|P_j|x\rangle\|}.$$

Example 2.2.4. *If we consider the previous example using projectors, then $n = m$ and $P_i = |x_i\rangle\langle x_i|$. Applying our measurement to ψ we will get output i with probability $\|P_i|\psi\rangle\|^2 = \|\alpha_i|x_i\rangle\|^2 = |\alpha_i|^2$ and the state collapses to $\frac{\alpha_i|x_i\rangle}{\|\alpha_i|x_i\rangle\|} = \frac{\alpha_i}{|\alpha_i|}|x_i\rangle$.*

2.2.4 Unitary Evolution

We have measured a quantum state, but what about the time evaluation of the quantum system? For quantum systems, the time evaluation of probabilistic systems via Markov matrices is replaced by matrices with complex number entries that preserve the constraint $\sum_{i=1}^{n} |\alpha_i|^2 = 1$. Thus, the quantum system state

$$|\phi\rangle = \alpha_1|x_1\rangle + \alpha_2|x_2\rangle + ... + \alpha_n|x_n\rangle$$

evolves into the state

$$|\phi'\rangle = \alpha'_1|x_1\rangle + \alpha'_2|x_2\rangle + ... + \alpha'_n|x_n\rangle,$$

where amplitudes $\alpha_1, \alpha_2, ..., \alpha_n$ and $\alpha'_1, \alpha'_2, ..., \alpha'_n$ are related by

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

and $\sum_{i=1}^{n} |\alpha_i|^2 = \sum_{i=1}^{n} |\alpha'_i|^2 = 1$. Thus, the quantum systems time evaluation should be unitary transformation.

Definition 2.2.5. *A complex matrix U is called unitary if $UU^\dagger = U^\dagger U = I_n$, where I_n is identity matrix in n dimensions and $U^\dagger$ is conjugate transpose of U.*

As unitary transformation always has inverse, it follows that time evaluation (non-measurement) must be reversible. The measurement is not reversible.

Example 2.2.5. *Let us consider a quantum coin flipping. We have a quantum system with*

two basis states $|0\rangle$ *and* $|1\rangle$ *and time evolution*

$$\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}.$$

Notice that if we start either in state $|0\rangle$ *or* $|1\rangle$ *after the first toss we can observe both options with probability* $\frac{1}{2}$*, but if we do not observe the system with the second toss we return to the starting state.*

Definition 2.2.6. *A Hermitian matrix or self-adjoint matrix is a square matrix with complex entries which is equal to its own conjugate transpose (*$A = A^{\dagger}$*).*

2.2.5 Mixed States and Density Matrices

Pure states are fundamental objects for quantum mechanics in the sense that evolution of any closed system can be seen as a unitary evolution of pure states. However, to deal with opened and composed quantum systems the concept of mixed state is important.

Definition 2.2.7. *A probability distribution* $\{(p_i, \phi_i)|1 \leq i \leq n\}$ *on pure states* $\{\phi_i\}_{i=1}^n$*, with probabilities* $0 < p_i \leq 1$*, where* $\sum_{i=1}^n p_i = 1$*, is called a mixed state or probabilistic mixture, and denoted by* $[\psi\rangle = \{(p_i, \phi_i)|1 \leq i \leq n\}$ *or*

$$[\psi\rangle = p_1\phi_1 \oplus ... \oplus p_n\phi_n = \bigoplus_{i=1}^{n} p_i\phi_i.$$

The result of the measurement of a pure state $\psi = \sum_{i=1}^n \alpha_i|\phi_i\rangle$ with respect to the observable given by an orthonormal basis $\{\phi\}_{i=1}^n$ can be consider as the mixed state

$$[\psi\rangle = \bigoplus_{i=1}^{n} |\alpha_i|^2\phi_i.$$

Definition 2.2.8. *To each mixed state* $[\psi\rangle = \bigoplus_{i=1}^n p_i\phi_i$ *corresponds a density matrix or density operator*

$$\rho_{[\psi\rangle} = \sum_{i=1}^{n} p_i|\phi_i\rangle\langle\phi_i|.$$

However, the density operator of a mixed state does not capture all the information about a mixed state. Different mixed states can have the same density operator. For

example, if

$$[\psi_1\rangle = \frac{1}{2}|0\rangle \oplus \frac{1}{2}|1\rangle \text{ and } [\psi_2\rangle = \frac{1}{2}(|0\rangle + |1\rangle) \oplus \frac{1}{2}(|0\rangle - |1\rangle)$$

the corresponding density matrix for both mixed states is

$$\rho_{\psi_1} = \rho_{\psi_2} = \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{pmatrix}.$$

Now, we list some important properties of density matrices $\rho = |x\rangle\langle x|$:

- matrices ρ have a unit trace [3].
- ρ is Hermitian.
- Eigenvalues of ρ are non-negative.
- $\rho = \rho^2$ if and only if ρ is a density matrix of a pure state. If ρ is a density matrix of mixed state then $\rho^2 < \rho$ and $Tr(\rho^2) < 1$.
- If ρ is a density matrix of a pure state then it has one eigenvalue equal to 1 and all other eigenvalues equal to 0.

Evaluation of density operator

For a pure state $|\psi\rangle$, the evaluation can be described with unitary transformation as $U|\psi\rangle$, where U is a unitary matrix. For the mixed state, if the system was in the state $|\psi_i\rangle$ with probability p_i then after the evaluation it will be in the state $U|\psi_i\rangle$ with probability p_i. Thus, the evaluation of the density operator can be described by the equation

$$\rho = \sum_i p_i|\psi_i\rangle\langle\psi_i| \xrightarrow{U} \sum_i p_i U|\psi_i\rangle\langle\psi_i|U^\dagger = U\rho U^\dagger. \tag{2.9}$$

Measurement

Suppose we perform measurements with projectors $P_1, P_2, ..., P_m$. If the initial state is ψ_i, then the probability of obtaining the result j is

$$p(j|i) = Tr(P_j|\psi_i\rangle\langle\psi_i|). \tag{2.10}$$

[3] Trace of the matrix ρ denoted by $Tr(\rho)$ is the sum of the diagonal elements.

Using the law of total probability the probability of getting result j is

$$p(j) = \sum_i p(j|i)p_i = \sum_i Tr(P_j|\psi_i\rangle\langle\psi_i|) = Tr(P_j\rho) \tag{2.11}$$

Using the techniques of the probability theory we can obtain that the density operator after obtaining measurement j is

$$\rho_j = \frac{P_j\rho P_j}{Tr(P_j\rho)}. \tag{2.12}$$

2.3 Quantum Finite State Automata

A classical finite state automaton servers as a basic model of a classical finite size machine. Similarly, a quantum finite state automaton can be seen as a basic quantum model of finite state quantum machines.

2.3.1 Models of Classical Finite State Automata

Different models of finite state automata have been developed and investigated in the classical computation. We consider the very basic notation of finite state automata - deterministic finite state automata, non-deterministic finite state automata, and probabilistic finite state automata.

Definition 2.3.1. *A deterministic finite state automaton (DFA) is a tuple* $A = (Q, \Sigma, \delta, q_0, Q_a)$ *where:*

- Q *is a finite set of states.*
- Σ *is a finite set of input symbols. It is said to be alphabet of the automaton.*
- $\delta : Q \times \Sigma \to Q$ *is a transition function.*
- $q_0 \in Q$ *is an initial state (the state of the automaton when no input has been processed).*
- $Q_a \subseteq Q$ *is a set of accepting states.*

An automaton reads a finite string of input letters $a_1a_2...a_n$ ($a_i \in \Sigma$), which is called *an input word*. Set of all words is denoted by Σ^*. Sometimes a special character is used to denote the end of the word. The automaton starts the computation at the initial state q_0 and performs the transition function corresponding to the current state and the current input symbol.

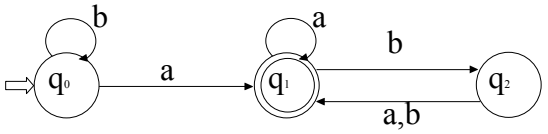

Figure 2.1 The deterministic finite state automaton A_1.

Definition 2.3.2. *A run of the automaton $A = (Q, \Sigma, \delta, q_0, Q_a)$ on the input word $a_1a_2...a_n \in \Sigma^*$ is a sequence of states $q_0q_1...q_n$, where $q_i \in Q$ and $q_i = \delta(q_{i-1}, a_i)$. ($q_n$ is the final state of the run.)*

Definition 2.3.3. *We say, the automaton $A = (Q, \Sigma, \delta, q_0, Q_a)$ accepts or recognizes an input word $a_1a_2...a_n$ if $q_n \in Q_a$.*

Definition 2.3.4. *We say, the automaton $A = (Q, \Sigma, \delta, q_0, Q_a)$ recognizes the language $L(A)$, if $L(A)$ contains all the words recognized by the automaton A.*

Example 2.3.1. *Let us consider the automaton $A_1 = (Q, \Sigma, \delta, q_0, Q_a)$, where*

- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{a, b\}$
- δ *can be described by*

	a	*b*
q_0	q_1	q_0
q_1	q_1	q_2
q_2	q_1	q_1

- q_0 *is the initial state*
- $Q_a = \{q_1\}$

The automaton can also be characterized using the state diagram (see the figure 2.1), where it has three states labelled q_0, q_1, and q_2. The initial state is indicated by the arrow pointing to it from nowhere, for the current example - it is the state q_0. The accepting states are indicated by double circle (in the example - $\{q_1\}$). The arrow from the state q_i to the state q_j with label a_k denotes transition $q_j = \delta(q_i, a_k)$.

In our example, the automaton A_1 recognizes language $L_2 = \{$ w | w has at least 1 a and even number of b's following the last a$\}$.

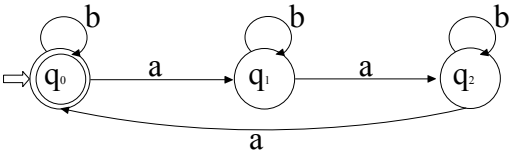

Figure 2.2 The group finite state automaton recognizing the language a^{3n}.

Besides deterministic finite state automaton there are also other variations in different components of automata (input, transition, acceptance). For example, an automata can accept finite or infinite words (the infinite words will be consider later in the section 2.6), or tree structures.

Let us consider three other variations of finite state automata - group finite state automata, non-deterministic finite state automata, and probabilistic finite state automata.

Definition 2.3.5. *A group finite state automaton (GFA) is a deterministic finite automaton* $A_{QFA} = (Q, \Sigma, \delta, q_0, Q_a)$ *with the restriction that for every state* $q \in Q$ *and every input symbol* $\sigma \in \Sigma$ *there exists exactly one state* $q' \in Q$ *such that* $\delta(q', \sigma) = q$.

In other words, δ is a complete one-to-one function and the automaton derived from A_{GFA} by reversing all transitions is deterministic. The automaton in the figure 2.1 is not a group automaton as there are three states q_0, q_1, and q_2 from which we can reach the state q_1 with input letter a.

In the figure 2.2, a group finite state automata recognizing the language $\{a^{3n} | n \geqslant 0\}$ is displayed.

Definition 2.3.6. *A non-deterministic finite state automaton (NFA) is a tuple* $A_{NFA} = (Q, \Sigma, \delta, q_0, Q_a)$ *where:*

- Q *is a finite set of states.*
- Σ *is a finite set of input symbols. It is said to be alphabet of the automaton.*
- $\delta : Q \times (\Sigma \cup \{\epsilon\}) \to P(Q)$ *is a transition function, where* $P(Q)$ *denotes the power set* [4] *of* Q.
- $q_0 \in Q$ *is an initial state (the state of the automaton when no input has been processed).*
- $Q_a \subseteq Q$ *is a set of accepting states.*

[4] $P(Q) \stackrel{def}{=} \{X : X \subseteq Q\}$ is the power set of Q.

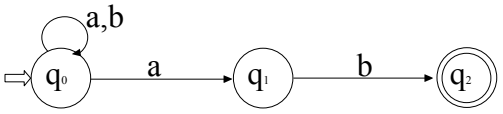

Figure 2.3 The non-deterministic automaton.

We may think of a non-deterministic automaton as a kind of parallel computing, where all possible transitions from a current state are followed simultaneously. If at least one of these is accepting, then the entire computation is accepted.

Definition 2.3.7. *A non-deterministic finite state automaton* $A_{NFA} = (Q, \Sigma, \delta, q_0, Q_a)$ *accepts the word* $w = w_1 w_2 ....w_n$ *(*$w_i \in \Sigma$*) if there exists a run* $q_{i_1} q_{i_2} .. q_{i_k}$ *(*$q_{i_j} \in Q$*) and* $y_1 y_2 ... y_k$ *(*$y_i \in \Sigma \cup \{\epsilon\}$*) such that:*

- $w = y_1 y_2 ... y_k$,
- $q_{i_1} = q_0$,
- $q_{i_k} \in Q_a$,
- *for every* j *(*$1 \leqslant j \leqslant k$*):* $q_{i_j} \in \delta(q_{i_{j-1}}, y_j)$.

Example 2.3.2. *Let us consider a non-deterministic finite state automaton of the figure 2.3. It recognizes words ending with the string* ab*. The accepting state can be reached only if the last two symbols are* ab*.*

For finite input, the language class accepted by non-deterministic finite state automata is equal to the language class accepted by the deterministic finite state automata.

Definition 2.3.8. *A probabilistic finite state automaton (PFA) is a tuple* $A_{PFA} = (Q, \Sigma, \delta, q_0, Q_a)$ *where:*

- Q *is a finite set of states.*
- Σ *is a finite set of input symbols. It is said to be an alphabet of the automaton.*
- $\delta : Q \times \{\Sigma \cup \{\#\} \cup \{\$\}\} \times Q \to \mathbb{R}_{[0,1]}$ *is a transition function, where for each state* $q \in Q$ $\sum_{q' \in Q} \delta(q, a, q') = 1$*, and* $\#$ *is the left end-marker denoting the start of the input word and* $\$$ *is the right end-marker denoting the end of the input word.*
- $q_0 \in Q$ *is an initial state (the state of the automaton when no input has been processed).*

- $Q_a \subseteq Q$ *is a set of accepting states.*

The transition function for each input symbol $\sigma \in \Sigma$ can be described as a Markov matrix $A_\sigma = [a_{ij}]$, where $a_{ij} = \delta(q_i, \sigma, q_j)$ - the probability of going from the q_i to the state q_j by reading a symbol σ. The computation of a probabilistic finite state automaton starts in the initial state q_0 and the transition corresponding to the current input symbol is performed. The computation process of a probabilistic finite state automaton can be viewed as Markov chain. The computation on an input word $w = \#w_1w_2...w_n\$$ is described by

$$A_\$ A_{w_n} A_{w_{n-1}} ... A_2 A_1 A_\# \begin{pmatrix} 1 \\ 0 \\ ... \\ 0 \end{pmatrix}, \tag{2.13}$$

where $\begin{pmatrix} 1 \\ 0 \\ ... \\ 0 \end{pmatrix}$ corresponds to probability distribution $1[q_0]$.

Definition 2.3.9. *We say that a probabilistic automaton $A_{PFA} = (Q, \Sigma, \delta, q_0, Q_a)$ accepts or recognizes a word $w = \#w_1w_2...w_3\$$ with probability p if $\sum_{q_j \in Q_a} p_j \geqslant p$, where*

$$\begin{pmatrix} p_0 \\ p_1 \\ ... \\ p_{n-1} \end{pmatrix} = A_\$ A_{w_n} A_{w_{n-1}} ... A_2 A_1 A_\# \begin{pmatrix} 1 \\ 0 \\ ... \\ 0 \end{pmatrix} \tag{2.14}$$

Similarly, we can define rejection with probability p. In this case, we consider the probability $\sum_{q_j \notin Q_a} p_j \geqslant p$.

Definition 2.3.10. *A probabilistic finite state automaton A_{PFA} is said to accept a language L with probability p if it accepts any $x \in L$ with probability at least p and rejects any $x \notin L$ with probability at least p.*

Definition 2.3.11. *A language L is said to be accepted by a probabilistic finite state automaton with bounded error if there is $\epsilon > 0$ such that any $x \in L$ is accepted with probability at least $p + \epsilon$ and rejects any $x \notin L$ is rejected with probability at least $p + \epsilon$.*

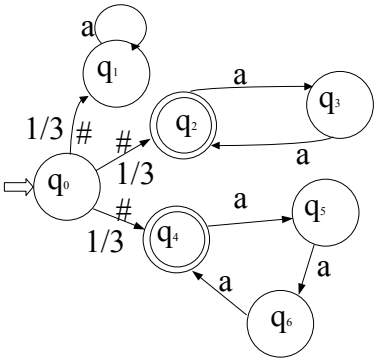

Figure 2.4 The probabilistic finite state automaton.

Example 2.3.3. *Let us consider a probabilistic finite state automaton of the figure 2.4. It recognizes language containing words of the length* $6n$ *with probability* $\frac{2}{3}$*. If an input word is of the length* $6n$ *it is accepted with probability* $\frac{2}{3}$*. If an input word is of the length* $3n$*, but not* $2n$ *it is rejected with probability* $\frac{2}{3}$*, the automaton rejects the word with the same probability if it is of the length* $2n$ *but not* $3n$*. If the word is neither with length* $2n$ *nor with length* $3n$ *it is rejected with probability* 1*.*

2.3.2 Measure-Once Quantum Finite State Automata

Definition 2.3.12. *A measure-once quantum finite state automaton (MO-QFA) is defined by a tuple as follows [38]*

$$A_{MO-QFA} = (Q; \Sigma; \delta; q_0; Q_{acc}; Q_{rej})$$

where

- *Q is a finite set of states,*
- Σ *is an input alphabet and* Γ=$\Sigma \cup \{\$\}$ *is working alphabet of* A_{MO-QFA}*, and* $\$$ *is the right end-marker, denoting the end of a word,*
- $q_0 \in Q$ *is a initial state,*
- $Q_{acc} \subseteq Q$ *and* $Q_{rej} \subseteq Q$ *are sets of accepting and rejecting states* $(Q_{acc} \cap Q_{rej} = \emptyset)$*,*
- δ *is the transition function* $\delta : Q \times \Gamma \times Q \to C_{[0,1]}$*, which represents the amplitudes that follows from the state* q *to the* q' *after reading symbol* σ*.*

For all states $q_1, q_2, q' \in Q$ *and symbols* $\sigma \in \Gamma$*, the function* δ *must be unitary, thus the function satisfies the condition*

$$\sum_{q'} \overline{\delta(q_1,\sigma,q')}\delta(q_2,\sigma,q') = \begin{cases} 1\ (q_1 = q_2) \\ 0\ (q_1 \neq q_2) \end{cases} \quad (2.15)$$

And it is assumed that an input word ends with the right end-marker.

The linear superposition of the automaton's A_{MO-QFA} states is represented by an n-dimensional complex unit vector, where $n = |Q|$. The vector is denoted by $|\phi\rangle = \sum_{i=1}^{n} \alpha_i |q_i\rangle$, where $\{|q_i\rangle\}$ is the set of orthonormal basis vectors corresponding to the states of the automaton A_{MO-QFA}.

The transition function δ is represented by a set of unitary matrices $\{V_\sigma\}_{\sigma\in\Gamma}$, where V_σ is the unitary transition of the automaton A_{MO-QFA} after reading the symbol σ and is defined by $V_\sigma(|q\rangle) = \sum_{q'\in Q} \delta(q,\sigma,q') |q'\rangle$.

A computation of A_{MO-QFA} on input $\sigma_1\sigma_2...\sigma_n\$$ proceeds as follows. It starts in the superposition $|q_0\rangle$. Then a transformation corresponding to the first input symbol is performed, followed by the transformations corresponding to the proceeding symbols of the input word and the right end-marker \$. After reading the right end-marker \$, the final superposition is observed with respect to E_{acc} and E_{rej} where $E_{acc} = span\{|q\rangle : q \in Q_{acc}\}$ and $E_{rej} = span\{|q\rangle : q \in Q_{rej}\}$. It means if the final superposition is $|\psi\rangle = \sum_{q_i\in Q_{acc}} \alpha_i |q_i\rangle + \sum_{q_j\in Q_{rej}} \beta_j |q_j\rangle$ then the measure-once quantum finite state automaton A_{MO-QFA} accepts the input word with probability $\sum \alpha_i^2$ and rejects $\sum \beta_j^2$.

Example 2.3.4. *Let us consider a measure-once quantum finite state automaton* $A_{L_3} = (\{q_0, q_1\}, \{a\}, \{U_a, U_\$\}, q_0, \{q_0\}, \{q_1\})$ *accepting language* $L_3 = \{a^{3n} | n \in Z\}$.

$$U_a = \begin{pmatrix} cos\frac{2\pi}{3} & sin\frac{2\pi}{3} \\ -sin\frac{2\pi}{3} & cos\frac{2\pi}{3} \end{pmatrix}, U_\$ = I_2.$$

The automaton starts the computation in the superposition $|q_0\rangle$*, after the first input symbol it changes the superposition to* $cos\frac{2\pi}{3}|q_0\rangle - sin\frac{2\pi}{3}|q_1\rangle$*, after the next symbol the automaton is in the superposition* $cos\frac{4\pi}{3}|q_0\rangle - sin\frac{4\pi}{3}|q_1\rangle$*. Reading the next input symbol, the automaton returns to the superposition* $|q_0\rangle$*. The automaton accepts a word belonging to language* L_3 *with probability 1, and rejects a word if it does not belong to* L_3 *with probability* $\frac{3}{4}$.

Theorem 2.1. *[16] A language* L *is recognized by a measure-once quantum finite state automaton if and only if it is recognized by a group finite state automaton.*

2.3.3 Measure-Many Quantum Finite State Automata

Definition 2.3.13. *A measure-many quantum finite state automaton (MM-QFA) is defined by a 6-tuple as follows [32]*

$$A_{MM-QFA} = (Q; \Sigma; \delta; q_0; Q_{acc}; Q_{rej})$$

where

- *Q is a finite set of states,*
- *Σ is an input alphabet and Γ=$\Sigma \cup \{ \sharp; \$ \}$ is working alphabet of A_{MM-QFA}, where $\sharp$ and $\$$ ($\notin \Sigma$) are the left and the right end-markers, denoting the start and the end of an input,*
- *δ is the transition function $\delta : Q \times \Gamma \times Q \to C_{[0,1]}$, which represents the amplitudes that flows from the state q to the state q' after reading symbol σ,*
- *$q_0 \in Q$ is the initial state,*
- *$Q_{acc} \subseteq Q$ and $Q_{rej} \subseteq Q$ are sets of accepting and rejecting states ($Q_{acc} \cap Q_{rej} = \emptyset$).*

The states in Q_{acc} and Q_{rej} are halting states and the states in $Q_{non} = Q \setminus (Q_{acc} \cup Q_{rej})$ are non-halting states.

For all states $q_1, q_2, q' \in Q$ and symbols $\sigma \in \Gamma$, the function δ must be unitary (the equation 2.15 should be satisfied). And it is assumed that an input word starts with the left end-marker and ends with the right end-marker.

The linear superposition of the automaton's A_{MM-QFA} states is also represented by an n-dimensional complex unit vector, where $n = |Q|$. The vector is denoted by $|\phi\rangle = \sum_{i=1}^{n} \alpha_i |q_i\rangle$, where $\{|q_i\rangle\}$ is the set orthonormal basis vectors corresponding to the states of the automaton A_{MM-QFA} and the transition function δ is represented by a set of unitary matrices $\{V_\sigma\}_{\sigma \in \Gamma}$, where V_σ is the unitary transition of the automaton A_{MM-QFA} after reading the symbol σ and is defined by $V_\sigma(|\ q\rangle) = \sum_{q' \in Q} \delta(q, \sigma, q') \mid q'\rangle$.

A computation of the automaton A_{MM-QFA} on the input word $\sharp\sigma_1\sigma_2...\sigma_n\$$ proceeds as follows:

- it starts in the superposition $|q_0\rangle$;
- a unitary transition corresponding to the current input symbol is performed;

- after every transition, the automaton A measures its state with respect to the observable $E_{acc} \oplus E_{rej} \oplus E_{non}$ where $E_{acc} = span\{|q\rangle : q \in Q_{acc}\}$, $E_{rej} = span\{|q\rangle : q \in Q_{rej}\}$ and $E_{non} = span\{|q\rangle : q \in Q_{non}\}$. If the observed state of the automaton A_{MM-QFA} is in E_{acc} subspace, then it accepts the input; if the observed state of A_{MM-QFA} is in E_{rej} subspace, then it rejects the input, otherwise the computation continues.

After every measurement, the superposition collapses to the measured subspace. A measurement is represented by a diagonal zero-one projection matrices P_{acc}, P_{rej} and P_{non} which project the vector onto E_{acc}, E_{rej} and E_{non}.

Since the automaton A_{MM-QFA} can have a non-zero probability of halting, it is useful to keep a track of the cumulative accepting and rejecting probabilities. Therefore, the state of the automaton A_{MM-QFA} is represented by a triple (ϕ, p_{acc}, p_{rej}), where p_{acc} and p_{rej} are the cumulative probabilities of accepting and rejecting. The transition of A_{MM-QFA} on reading the symbol σ is denoted by $(P_{non}|\phi'\rangle, p_{acc} + \|P_{acc}\phi'\|^2, p_{rej} + \|P_{rej}\phi'\|^2)$, where $\phi' = V_\sigma\phi$.

Example 2.3.5. *As the example we consider a measure-many quantum finite state automaton recognizing the language $L_{a^*b^*} = \{a^*b^*\}$ in the alphabet $\{a, b\}$ [5].*

$$A_{a^*b^*} = (\{q_0, q_1, q_{acc}, q_{rej}\}, \{a, b\}, \delta, \{q_{acc}\}, \{q_{rej}\}, \{q_0, q_1\}),$$

where δ :

$$\begin{gathered} V_\sharp|q_0\rangle = \sqrt{(1-p)}|q_0\rangle + \sqrt{p}|q_1\rangle, \\ V_a(|q_0\rangle) = (1-p)|q_0\rangle + \sqrt{p(1-p)}|q_1\rangle + \sqrt{p}|q_{rej}\rangle, \\ V_a(|q_1\rangle) = \sqrt{p(1-p)}|q_0\rangle + p|q_1\rangle - \sqrt{1-p}|q_{rej}\rangle, \\ V_b(|q_0\rangle) = |q_{rej}\rangle, V_b(|q_1\rangle) = |q_1\rangle, \\ V_\$(|q_0\rangle) = |q_{rej}\rangle, V_\$(|q_1\rangle) = |q_{acc}\rangle, \end{gathered}$$

where p is the root of $p^3 + p = 1$.

The automaton starts the computation in the superposition $|q_0\rangle$, the automaton changes the superposition to $\sqrt{1-p}|q_0\rangle + \sqrt{p}|q_1\rangle$. Now we consider several input cases:

- *The input is a^*. While the current input symbol is a, the automaton does not change its superposition $\sqrt{1-p}|q_0\rangle + \sqrt{p}|q_1\rangle$ and when the automaton reads the right end-marker, it accepts the word with probability p.*

- *The input is a^*bb^*. While the current input symbol is a, the automaton does not change its superposition $\sqrt{1-p}|q_0\rangle + \sqrt{p}|q_1\rangle$. However, when the automaton reads the input symbol b, it rejects the input word with probability $1-p$ and with*

probability p is in the superposition $|q_1\rangle$. While the automaton reads the input symbol b, it stays in the superposition $|q_1\rangle$ with probability p. After the automaton reads the right end-marker, the automaton changes its state to $|q_{acc}\rangle$ and accepts the word with probability p.

- *The input is not in $L_{a^*b^*}$, it means that after a^*bb^* we have at least one a. After reading a^*bb^* when the automaton is in $|q_1\rangle$ with probability p. When the automaton receives an a, then the rejecting probability becomes $(1-p)+p(1-p) = 1-p^2$ and the automaton is in the superposition $\sqrt{1-p}|q_0\rangle + \sqrt{p}|q_1\rangle$ with probability p^2, by receiving next b or the right end-marker, the rejecting probability becomes $1-p^2+p^2(1-p) = 1-p^3 = p^3+p-p^3 = p$.*

A measure-once quantum finite state automaton $A_1 = (Q_1; \Sigma; \delta_1; q_1; Q_{acc_1}; Q_{rej_1})$ with n states can be easily simulated by a measure-many quantum finite state automaton $A_2 = (Q_2; \Sigma; \delta_2; q_1; Q_{acc_2}; Q_{rej_2})$, where $Q_2 = \{q_1, ..., q_{2n}\}$, states $q_1, ..., q_n$ are non-halting states, the state q_{n+i} is accepting if the state q_i of the automaton A_1 is accepting, otherwise $q_{n+i} \in Q_{rej_2}$, the transition function δ_2 is defined as follows:

- $V_{2_\#}(|q\rangle) = |q\rangle$ for $q \in \{q_1, ..., q_{2n}\}$
- $V_{2_\sigma}(|q\rangle) = V_{1_\sigma}(|q\rangle)$ for $q \in \{q_1, ..., q_n\}$ and $\sigma \in \Sigma$
- $V_{2_\sigma}(|q\rangle) = |q\rangle$ for $q \in \{q_{n+1}, ..., q_{2n}\}$ and $\sigma \in \Sigma$
- $V_{2_\$}(|q\rangle) = \sum_{q_{n+j} \in \{q_{n+1}, .., q_{2n}\}} \delta(q, \$, q_{j+n}) \, |q_{j+n}\rangle$ if
  $V_{1_\$}(|q\rangle) = \sum_{q_j \in Q_1} \delta(q, \$, q_j) \, |q_j\rangle$ for $q \in \{q_1, ..., q_n\}$.

2.3.4 Latvian Quantum Finite State Automata

Definition 2.3.14. *A Latvian quantum finite state automaton (LQFA) is defined by a tuple as follows [1]*

$$A_{LQFA} = (Q; \Sigma; \{A_\sigma\}; \{P_\sigma\}; q_0; Q_{acc})$$

where

- *Q is a finite set of states,*

- Σ *is an input alphabet and* $\Gamma = \Sigma \cup \{ \sharp; \$ \}$ *is working alphabet of* A_{LQFA}*, where* $\sharp$ *and* $\$$ $(\notin \Sigma)$ *are the left and the right end-markers,*
- $\{A_\sigma\}$ *are unitary matrices defined for each working symbol,*
- $\{P_\sigma\}$ *are measurements for each working symbol, each* P_σ *is defined as a set* $\{E_1, ..., E_j\}$ *of orthogonal subspaces,*
- $q_0 \in Q$ *is the initial state,*
- $Q_{acc} \subseteq Q$ *and* $Q_{rej} = Q \setminus Q_{acc}$ *are sets of accepting and rejecting states.*

A Latvian quantum finite state automaton A_{LQFA} starts a computation in the state $|q_0\rangle$. By reading each input letter σ, the automaton transforms the current state by a unitary matrix A_σ and performs a measurement P_σ. At the end of the computation automaton A_{LQFA} accepts or rejects the input according to the output of $P_\$$ measurement, where $P_\$ = E_{acc} \otimes E_{rej}$ ($E_{acc} = span\{|q\rangle | q \in Q_{acc}\}$ and $E_{rej} = span\{|q\rangle | q \in Q_{rej}\}$). As a Latvian quantum finite state automaton uses mixed states, it is convenient to use a density matrix to identify the state of the automaton.

Example 2.3.6. *A Latvian quantum finite state automaton for language* $\Sigma^* a \Sigma^*$ *in the alphabet* $\Sigma = \{a, b\}$ *is* $A_{LQFA_1} = (Q, \Sigma, \{A_\sigma\}, \{P_\sigma\}, q_0, Q_{acc})$ *:* $Q = \{q_0, q_1, q_2, q_3\}$, $\Sigma = \{a, b\}$, $Q_{acc} = \{q_1, q_2, q_3\}$, $P_\sharp = P_a = P_b = E_0 \otimes E_1 \otimes E_2 \otimes E_3$ *(*$E_i = span\{|q_i\rangle\}$*),*

$A_\sharp = A_\$ = A_b = I_4$, $A_a = \begin{pmatrix} -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \end{pmatrix}$

The computation of the automaton A_{LQFA_1} *on the input* $bbaab$ *is following:*

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \rightarrow_\sharp \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \rightarrow_b \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \rightarrow_b \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \rightarrow_a$$

$$\begin{pmatrix} \frac{1}{4} & 0 & 0 & 0 \\ 0 & \frac{1}{4} & 0 & 0 \\ 0 & 0 & \frac{1}{4} & 0 \\ 0 & 0 & 0 & \frac{1}{4} \end{pmatrix} \rightarrow_a \begin{pmatrix} \frac{1}{4} & 0 & 0 & 0 \\ 0 & \frac{1}{4} & 0 & 0 \\ 0 & 0 & \frac{1}{4} & 0 \\ 0 & 0 & 0 & \frac{1}{4} \end{pmatrix} \rightarrow_b \begin{pmatrix} \frac{1}{4} & 0 & 0 & 0 \\ 0 & \frac{1}{4} & 0 & 0 \\ 0 & 0 & \frac{1}{4} & 0 \\ 0 & 0 & 0 & \frac{1}{4} \end{pmatrix}.$$

The automaton A_{LQFA_1} *correctly rejects the input with probability 1 and correctly accepts the input with probability* $\frac{3}{4}$.

Theorem 2.2. *[1] Latvian quantum finite state automata recognize exactly those languages whose syntactic monoid is in BG.*

2.4 Monoids

In the thesis, we also refer to an algebraic structure with a single associative binary operation and an identity element called a monoid. The section contains the definitions which will be later used in the thesis.

Definition 2.4.1. *A monoid is a tuple $\mathcal{M} = (M, \bullet_M, 1_M)$ consisting of a non-empty set M, an associative binary operation $\bullet_M$ and a unit element $1_M \in M$, i.e., the following has to be satisfied:*

- *closure:* $\forall x, y : x \bullet_M y \in M$
- *associativity:* $\forall x, y, z \in M : (x \bullet_M y) \bullet_M z = x \bullet_M (y \bullet_M z)$
- *unit element:* $\forall x : 1_M \bullet_M x = x = x \bullet_M 1_M$.

Definition 2.4.2. *A monoid $\mathcal{M} = (M, \bullet_M, 1_M)$ is finite if and only if M is finite.*

Definition 2.4.3. *Let $\mathcal{M}$, $\mathcal{N}$ be monoids. The mapping $\phi : M \to N$ is a homomorphism if and only if:*

- $\phi(x \bullet_M y) = \phi(x) \bullet_N \phi(y)$,
- $\phi(1_M) = 1_N$.

Example 2.4.1. *For an alphabet Σ, the set of Σ^* with concatenation as binary operation and empty word ϵ as unit element is a monoid.*

Definition 2.4.4. *Let $\mathcal{M}$ be a monoid, Σ an alphabet, and $\phi : \Sigma^* \to M$ a homomorphism. Every subset N of M defines a subset of Σ^*:*

$$\phi^{-1}(N) = \{w \in \Sigma^* | \phi(w) \in N\}.$$

Definition 2.4.5. *The monoid $\mathcal{M}$ accepts the language $L \subseteq \Sigma^*$ if and only if there is $N \subseteq M$ and a homomorphism $\phi : \Sigma^* \to M$ such that $L = \phi^{-1}(N)$.*

Example 2.4.2. *The monoid accepting the language of the deterministic finite state automata in the figure 2.1 is $M = \{\delta_\epsilon, \delta_a, \delta_b, \delta_{ab}\}$ with binary operation:*

	δ_ϵ	δ_a	δ_b	δ_{ab}
δ_ϵ	δ_ϵ	δ_a	δ_b	δ_{ab}
δ_a	δ_a	δ_a	δ_{ab}	δ_{ab}
δ_b	δ_b	δ_a	δ_ϵ	δ_{ab}
δ_{ab}	δ_{ab}	δ_a	δ_a	δ_{ab}

and $N = \{\delta_a\}$.

Definition 2.4.6. *For a regular language* L*, the transition monoid of the minimal automaton is called the syntactic monoid of* L.

Theorem 2.3. *Let* $L \subseteq \Sigma^*$*. Then the following is equivalent:*

1. *L is accepted by a finite monoid.*
2. *L is regular.*

Definition 2.4.7. *The finite monoid* $\mathcal{M}$ *is called aperiodic if and only if it is a monoid and there exists a non-negative integer* n*, such that* $m^{n+1} = m^n$ *for all* $m \in M$.

Example 2.4.3. *Let us look at the monoid accepting the language containing a substring* aba *-* $M = \{\delta_\epsilon, \delta_a, \delta_b, \delta_{ab}, \delta_{ba}, \delta_{bb}, \delta_{aba}, \delta_{abb}, \delta_{bab}, \delta_{bba}, \delta_{babb}, \delta_{bbab}\}$*, where binary operation is defined as follows:*

	δ_ϵ	δ_a	δ_b
δ_ϵ	δ_ϵ	δ_a	δ_b
δ_a	δ_a	δ_a	δ_{ab}
δ_b	δ_b	δ_{ba}	δ_{bb}
δ_{ab}	δ_{ab}	δ_{aba}	δ_{abb}
δ_{ba}	δ_{ba}	δ_{ba}	δ_{bab}
δ_{bb}	δ_{bb}	δ_{bba}	δ_{bb}
δ_{aba}	δ_{aba}	δ_{aba}	δ_{aba}
δ_{abb}	δ_{abb}	δ_a	δ_{abb}
δ_{bab}	δ_{bab}	δ_{aba}	δ_{babb}
δ_{bba}	δ_{bba}	δ_{bba}	δ_{bbab}
δ_{babb}	δ_{babb}	δ_{ba}	δ_{babb}
δ_{bbab}	δ_{bbab}	δ_{aba}	δ_{bb}

where $N = \{\delta_{aba}\}$*. The monoid is aperiodic as for all* $\delta \in M$ $\delta^m = \delta^{m+1}$*, where* $m \geqslant 2$.

Definition 2.4.8. *A group is a monoid* $\mathcal{G} = (G, \bullet_G, 1_G)$ *with an inverse element:* $\forall a \in G$ *there exists an element* $b \in G$ *such that* $a \bullet_G b = b \bullet_G a = 1_G$*. The inverse element of an element* a *is denoted by* a^{-1}.

The monoid of the example 2.4.2 is not a group. However, let us consider the monoid accepting the language of the group finite state automata in the figure 2.2.

Example 2.4.4. *$M = \{\delta_\epsilon = \delta_b, \delta_a, \delta_{aa}\}$ with binary operation:*

	δ_b	δ_a	δ_{aa}
δ_b	δ_b	δ_a	δ_{aa}
δ_a	δ_a	δ_{aa}	δ_b
δ_{aa}	δ_{aa}	δ_b	δ_a

and $N = \{\delta_b\}$. It can be seen that it forms the group, the inverse element of δ_b is δ_b, δ_a - δ_{aa}, and δ_{aa} - δ_a.

Definition 2.4.9. *We say that some subset H of G is a subgroup of $\mathcal{G} = (G, \bullet_G, 1_G)$ if H also forms a group under the operation $\bullet_G$.*

If we look at the group of the example 2.4.4, then we can see that $H = \{\delta_b\}$ forms a subgroup of M. However, let us consider an example of group which contains a bigger subgroup.

Example 2.4.5. *We construct a group recognizing the language described by a regular expression $b^*ab(a^* \vee b(ab^*a)^*b)^*$. The group consists of $G = \{\delta_\epsilon, \delta_a, \delta_b, \delta_{ab}, \delta_{ba}, \delta_{aba}\}$, $N = \{\delta_{ab}, \delta_{aba}\}$ and binary operation:*

	δ_ϵ	δ_a	δ_b	δ_{ab}	δ_{ba}	δ_{aba}
δ_ϵ	δ_ϵ	δ_a	δ_b	δ_{ab}	δ_{ba}	δ_{aba}
δ_a	δ_a	δ_ϵ	δ_{ab}	δ_b	δ_{aba}	δ_{ba}
δ_b	δ_b	δ_{ba}	δ_ϵ	δ_{aba}	δ_a	δ_{ab}
δ_{ab}	δ_{ab}	δ_{aba}	δ_a	δ_{ba}	δ_ϵ	δ_b
δ_{ba}	δ_{ba}	δ_b	δ_{aba}	δ_ϵ	δ_{ab}	δ_a
δ_{aba}	δ_{aba}	δ_{ab}	δ_{ba}	δ_a	δ_b	δ_ϵ

This group has several subgroups $\{\delta_\epsilon, \delta_a\}$, $\{\delta_\epsilon, \delta_b\}$, $\{\delta_\epsilon, \delta_{aba}\}$, $\{\delta_\epsilon, \delta_{ab}, \delta_{ba}\}$.

Definition 2.4.10. *A subgroup N of a group G is called a normal subgroup if it is invariant under conjugation, that is, for each element $n \in N$ and each $g \in G$, the element $g \bullet_G n \bullet_G g^{-1}$ is still in N. We write $N \triangleleft G \iff \forall n \in N, \forall g \in G, g \bullet_G n \bullet_G g^{-1} \in N$.*

Now, let us consider the subgroups of the previous example, if they are normal subgroups:

- $\{\delta_\epsilon, \delta_a\}$ is not a normal subgroup as $\delta_b\delta_a\delta_b = \delta_{aba}$, which is not in the subgroup $\{\delta_\epsilon, \delta_a\}$.

- $\{\delta_\epsilon, \delta_b\}$ is not a normal subgroup as $\delta_a\delta_b\delta_a = \delta_{aba}$, which is not in the subgroup $\{\delta_\epsilon, \delta_b\}$.
- $\{\delta_\epsilon, \delta_{aba}\}$ is not a normal subgroup as $\delta_a\delta_{aba}\delta_a = \delta_b$, which is not in the subgroup $\{\delta_\epsilon, \delta_{aba}\}$.
- $\{\delta_\epsilon, \delta_{ab}, \delta_{ba}\}$ is a normal subgroup, as $\delta_a\delta_{ab}\delta_a = \delta_{ba}$, $\delta_b\delta_{ab}\delta_b = \delta_{ba}$, $\delta_{aba}\delta_{ab}\delta_{aba} = \delta_{ba}$, $\delta_a\delta_{ba}\delta_a = \delta_{ab}$, $\delta_b\delta_{ba}\delta_b = \delta_{ab}$, and $\delta_{aba}\delta_{ba}\delta_{aba} = \delta_{ab}$.

Definition 2.4.11. *A subnormal series of a group G is a sequence of subgroups, each a normal subgroup of the next one.*

Definition 2.4.12. *A group is said to be an abelian group A if for all $g, h \in A$ $g \bullet h = h \bullet g$.*

The group M in the example 2.4.4 is an abelian group.

Definition 2.4.13. *If G is a group, H is a subgroup of G, and g is an element of G, then a left coset of H in G is $gH = \{gh | h \in H\}$, and a right coset of H in G is defined as $Hg = \{hg | h \in H\}$.*

Example 2.4.6. *Let us consider the group of the example 2.4.5 and the subgroup $H = \{\delta_\epsilon, \delta_{ab}, \delta_{ba}\}$. The left cosets of H in G are:*

- $\delta_\epsilon H = H$
- $\delta_a H = \{\delta_a, \delta_b, \delta_{aba}\}$
- $\delta_b H = \{\delta_a, \delta_b, \delta_{aba}\}$
- $\delta_{ab} H = H$
- $\delta_{ba} H = H$
- $\delta_{aba} H = \{\delta_a, \delta_b, \delta_{aba}\}$

Definition 2.4.14. *If N is a normal subgroup of G then the quotient group or factor group G/N is defined to be the set of cosets*

$$G/N = \{gN | g \in G\}$$

together with the binary operation given by $gN \bullet hN = g \bullet hN$.

The quotient or factor group of the example 2.4.6 is $\{\{\delta_a, \delta_b, \delta_{aba}\}, \{\delta_\epsilon, \delta_{ab}, \delta_{ba}\}\}$.

Definition 2.4.15. *A group G is called solvable if it has a subnormal series whose quotient groups are all abelian, that is, if there are subgroups $\{1_G\} = G_0 \triangleleft G_1 \triangleleft G_2 \triangleleft ... \triangleleft G_k = G$ such that G_{j-1} is normal in G_j, and G_j/G_{j-1} is an abelian group, for $j = 1...k$.*

The group G of the example 2.4.5 is also solvable group as it has a subnormal series $\{\delta_\epsilon\} \triangleleft \{\delta_\epsilon, \delta_{ab}, \delta_{ba}\} \triangleleft G$.

Example 2.4.7. *As another example of a group, let us consider a symmetric group. A symmetric group on set X is the group consisting of all bijections of the set (all one-to-one and onto functions from the set to itself) with function composition as the group operation.*

The elements of the symmetric group on a set are the permutations of that set. The permutations can be described in a number of ways. Two most common ways of describing permutations are the one-line notation and the cycle decomposition. For instance, the two-line notation for a permutation on the set $\{1, 2, 3, 4, 5\}$:

$$\begin{pmatrix} 1 & 4 & 2 & 3 & 5 \\ 3 & 5 & 2 & 1 & 4 \end{pmatrix}.$$

The two-line notation for a permutation is not unique, because the order of elements in the first row is arbitrary. If we fix the order of elements in the first row of the permutation, then we can omit the first row of the permutation and simply write the second row. This is termed the one-line notation for permutations. For instance, for the set 1,2,3,4,5, we can fix the first row to have $1, 2, 3, 4, 5$ in that order. In this case, the two-line notation is:

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 2 & 1 & 5 & 4 \end{pmatrix},$$

thus we get the one-line notation:

$$\begin{pmatrix} 3 & 2 & 1 & 5 & 4 \end{pmatrix}.$$

Any permutation can be expressed uniquely as a product of disjoint cycles (a cycle is a permutation of the elements of some set X which maps the elements of some subset S to each other in a cyclic fashion, while fixing (i.e., mapping to themselves) all other elements.). A cycle $(a_1, a_2, ..., a_n)$ being in a permutation means that under the permutation, a_i is mapped to a_{i+1}, and a_n is mapped to a_1. Two cycles are disjoint if they do not have any element in common. In our case, cycles are $(1, 3)(4, 5)$.

The symmetric group S_5 is the group of all permutations on a set of five elements. It

has 120 elements and the generating set of S_5 is $(1,2)(1,2,3,4,5)$ using which we can get every element in S_5. It is known that S_5 forms a non-solvable group.

Definition 2.4.16. *An element σ of the monoid M is called idempotent if $\sigma = \sigma \bullet_m \sigma$.*

The set of idempotents of a monoid M is devoted by $E(M)$. In the example 2.4.2, $E(M) = \{\delta_\epsilon, \delta_a, \delta_{ab}\}$, but in example 2.4.4, $E(M) = \{\delta_b\}$.

Definition 2.4.17. *A monoid M is block group if and only if for any e and f in $E(M)$ $eM = fM$ or $Me = Mf$ implies $e = f$.*

The monoid of the example 2.4.4 is a block group. For languages recognized by block groups we have the following theorem:

Theorem 2.4. *[44] For any language L, its syntactic monoid $M(L)$ is a block group if and only if L is a Boolean combination of languages of the form $L_0a_1L_1a_2...a_nL_n$, where $a_i \in \Sigma$ and L_i is the language recognized by a finite group.*

2.5 Logic

The section contains the notations and results of logic the connection classical computation.

Definition 2.5.1. *A signature or vocabulary is defined by a triple $\delta = (F, R, ar)$, where F and R are disjoint sets not containing any other logic symbols, F is function symbols and R is relation symbols, and $ar : F \cup R \to N_0$, which assigns non-negative integer called arity to every function and relation symbol. A function or relation symbol is called n-ary if it has an arity n. 0-ary function is called a constant.*

In other words, a signature is a set of non-logical constants together with additional information identifying each symbol as either a constant symbol, or a function symbol (for example: 0, +) of a specific arity n or a relation symbol (<, $\in$) of a specific arity.

Definition 2.5.2. *A structure is defined by a triple $\mathcal{A} = (A, \delta, I)$, consisting of a domain A (an arbitrary set), a signature δ, and an interpretation function I that indicates how the signature should be interpreted on the domain.*

Example 2.5.1. *Let us consider a structure $\mathcal{A}_1 = (Z, \delta_+, I_Z)$, where Z are a set of integers, δ_+ consists of a binary function symbol $+$, an unary function symbol $-$ and constant symbols 0, 1, and interpretation:*

- $I_+(+) : Z \times Z \to Z$ - *addition of integers,*
- $I_+(-) : Z \to Z$ - *function that transforms each integer* x *to* $-x$,
- $I_+(0) \in Z$ *is the number* 0,
- $I_+(1) \in Z$ *is the number* 1.

2.5.1 First Order Languages

Definition 2.5.3. *The* $word\ model$ *for the word* $\omega = a_1a_2...a_n$ *be a word over the alphabet* Σ *is represented by the relational structure*

$$\underline{\omega} = (dom(\omega), <, (Q_a)_{a\in\Sigma})$$

where $dom(\omega) = \{1, 2, ..., n\}$ *is the set of the letters "positions" of* ω *(the "domain" of* ω*),* $<$ *is the order relation on* $dom(\omega)$*, and* $Q_a = \{i \in dom(\omega) \mid a_i = a\}$ *("position carries letter a").*

We consider word models over the finite alphabet Σ. The corresponding *first order language* $FO[<]$ has variables $x, y, ...$ ranging over positions in the word models, and is built from atomic formulas of the form

$$x = y, Q_a(x), x < y$$

by means of the connectivities $\neg, \vee, \wedge, \to, \leftrightarrow$ and quantifiers $\exists$ and $\forall$. We may also use the successor relation $S(x, y)$, which can be expressed as first order formula $x < y \wedge \neg\exists z(x < z \wedge z < y)$. The notation $\varphi(x_1, x_2, ..., x_n)$ indicates that in the formula φ at most the variables $x_1, x_2, ..., x_n$ are free, i.e. they are not in the scope of a quantifier. A $sentence$ is a formula with no free variables. If $p_1, p_2, ..., p_n$ are positions from $dom(\omega)$ then $(\underline{\omega}, p_1, p_2, ..., p_n) \models \varphi(x_1, x_2, ..., x_n)$ means that φ is satisfied in the word model $\underline{\omega}$ when $p_1, p_2, ..., p_n$ serve as an interpretation of $x_1, x_2, ..., x_n$. The language defined by the sentence φ is $L(\phi) = \{\omega \in A^* \mid \underline{\omega} \models \varphi\}$. Languages defined by such sentences are the first order $FO[<]$ languages. For example, the sentence $\forall x(Q_a(x))$ over the alphabet $A = \{a, b\}$ defines the language containing all words having only letters a. This language is a $FO[<]$ language.

The classical equivalence result of the first order logic is result by Schützenberg [48]:

Theorem 2.5. *For a language* $L \in A^*$ *the following are equivalent*

1. L *is star-free (the smallest class that satisfies the following: all finite languages over* A *belong to star-free languages, if languages* L_1*,* L_2 *are star-free then so are* $L_1 \cdot L_2$*,* $L_1 \cup L_2$*,* $L_1 \cap L_2$ *and* $\bar{L}_1 = A^* \setminus L$*).*

2. *L is recognizable by a finite aperiodic monoid.*

3. *L is defined by a first order formula.*

2.5.2 Modular Logic

We will consider a new quantifier called modular quantifier [51].

Definition 2.5.4. *A modular quantifier is $\exists^{m,n} x\varphi(x)$ that means $\varphi(x)$ is true for a number of x equal to $n \bmod m$.*

Definition 2.5.5. *Let us denote by $MOD[<]$ the class of languages defined by first order atomic formulas ($x = y$, $Q_a(x)$, $x < y$) by means of the connectivities $\neg, \vee, \wedge, \rightarrow, \leftrightarrow$ and the modular quantifier.*

An example of $MOD[<]$ is the language containing all words that has even number of symbol a, the corresponding formula for this language is $\exists^{2,0} x Q_a(x)$.

Theorem 2.6. *[51] Let $L \subseteq A^*$ be a regular language, then $L \in MOD[<]$ if and only if the syntactic monoid of language L is a solvable group.*

2.5.3 Generalized Quantifier

Logical framework can also be extended with generalized quantifiers; they have been introduced by Mostowski [9]. One of such quantifiers is Lindström quantifier.

Definition 2.5.6. *Consider a language L over the alphabet $\Sigma = (a_1, a_2, ..., a_s)$. Let $\bar{x}$ be a k-tuple of variables (each ranging from 1 to the input length n). In the following, we assume the lexical ordering on $\{1, 2, ..., n\}^k$, and we write $X_1, X_2, ..., X_{n^k}$ for this sequence of the potential values taken on by $\bar{x}$. Let $\phi_1(\bar{x}), \phi_2(\bar{x}), ..., \phi_{s-1}(\bar{x})$ be s-1 Γ-formulas for some alphabet Γ. The*

$$Q_L \bar{x}[\phi_1(\bar{x}), \phi_2(\bar{x}), ..., \phi_{s-1}(\bar{x})]$$

holds on string $\omega = \omega_1\omega_2...\omega_n$, if and only if the word of length n^k whose i-th letter ($1 \leq i \leq n^k$) is

$$\begin{cases} a_1 & if\ \omega \models \phi_1(X_i), \\ a_2 & if \omega \models \neg\phi_1(X_i) \wedge \phi_2(X_i), \\ a_3 & if \omega \models \neg\phi_1(X_i) \wedge \neg\phi_2(X_i) \wedge \phi_3(X_i), \\ ... & \\ a_s & if \omega \models \neg\phi_1(X_i) \wedge \neg\phi_2(X_i) \wedge ... \wedge \neg\phi_{s-1}(X_i), \end{cases}$$

belongs to L.

Example 2.5.2. *Consider alphabet $\Sigma = \{0,1\}$ and a language L which is defined by regular expression$(0,1)^*0(0,1)^*$, then formula $Q_L(\phi(x))$ is equal to the classical first order existential quantifier applied to some quantifier-free formula ϕ with free variable x, i.e. $\exists x\phi(x)$. It is easy to see, that the formula will be true if there is at least one position of x for which $\phi(x)$ will be true.*

Let us consider another type of generalized quantifier - a group quantifier [37].

Definition 2.5.7. *Fix a finite group G and a mapping from $\{0,1\}^k$ onto g for a fixed integer k. Let $\langle\psi_1(x),...,\psi_k(x)\rangle$ be a vector of the first order formulas with a single common free variable x. For each x, let $g(x)$ be the element of G denoted by the vector of truth values of $\langle\psi_1(x),...,\psi_k(x)\rangle$. For an element $g \in G$ and an input of length n, we define $(\Gamma^{G,g}x)\langle\psi_1(x),...,\psi_k(x)\rangle$ to be true if and only if the element of G obtained by multiplying $g(1)g(2)...g(n)$ is g.*

Example 2.5.3. *We can define a modular quantifier $\exists^{(m,l)}\psi(x)$ in terms of group quantifier. We have a cyclic group (a group that can be generated by a single element) $G = \{1_G, \sigma_1, \sigma_2, ..., \sigma_m\}$, then we define mapping 0 onto 1_G and 1 onto σ_1. $(\Gamma^{G,\sigma_l})\langle\psi(x)\rangle$ is equal to $\exists^{(m,l)}\psi(x)$, as $(\Gamma^{G,\sigma_l})\langle\psi(x)\rangle$ is true if and only if $g(1)g(2)...g(n) = \sigma_l$ ($g(i) = 1_G$ or $g(i) = \sigma_1$). $g(1)g(2)...g(n) = \sigma_l$ is satisfied if and only if $l \mod m$ number of elements $g(i)$ are σ_1 so $\psi(x)$ must be true $l \mod m$ times.*

2.6 Finite State Automata over Infinite Words

At first, we give the main notations for infinite words.

Definition 2.6.1. *Σ^ω is the set of all infinite words over alphabet Σ. For infinite word α we write $\alpha = \alpha(0)\alpha(1)\alpha(2)...$ with $\alpha(i) \in \Sigma$. A set of infinite words over a given alphabet is called an ω-language.*

The number of occurrences of the symbol a in an infinite word (ω-word) α is denoted by $|\alpha|_a$.

Definition 2.6.2. *For a given ω-word $\alpha \in \Sigma^\omega$, let $Occ(\alpha) = \{a \in \Sigma | \exists i \alpha(i) = a\}$ be the finite set of letters occurring in α and $Inf(\alpha) = \{a \in \Sigma | \exists^\omega j \alpha(j) = a\}$ where $\exists^\omega$ denotes the quantifier "there exists infinite many".*

Definition 2.6.3. ***Limit languages*** *Let $U \subseteq \Sigma^*$ be a language of finite strings. The limit U is defined as $lim(U) = \{\alpha \in \Sigma^\omega | \exists^\omega n \in N_0 : \alpha[0..n] \in U\}$.*

A word belongs to $lim(U)$ if and only if it has infinitely many prefixes in U. For example, $U = \{aba^*\}$, then $lim(U) = \{aba^\omega\}$.

2.6.1 Classical Automata over Infinite Words

In this section, the basic concepts of finite state automata over infinite words or ω-automata are presented (for more details refer to [25], [53]).

Definition 2.6.4. *An **finite state automata over infinite words** or **ω-automaton** is a quintuple $A = (Q, \Sigma, \delta, q_0, Acc)$, where*

- *Q is a finite set of states,*
- *Σ is a finite alphabet,*
- *$\delta : Q \times \Sigma \to 2^Q$ is the state transition function,*
- *$q_0 \in Q$ is the initial state,*
- *Acc is the acceptance component.*

In a deterministic ω-automaton, a transaction function $\delta : Q \times \Sigma \to Q$ is used.

The acceptance component can be given in different ways, and it will be explained below.

Definition 2.6.5. *Let $A = (Q, \Sigma, \delta, q_0, Acc)$ be an ω-automaton. A **run** of A on an ω-word $\alpha = \alpha_1\alpha_2... \in \Sigma^\omega$ is an infinite state sequence $\rho = \rho(0)\rho(1)\rho(2)... \in Q^\omega$, such that the following holds:*

1. *$\rho(0) = q_0$*
2. *$\rho(i) \in \delta(\rho(i-1), \alpha_i)$ for $i > 0$ if A is non-deterministic and $\rho(i) = \delta(\rho(i-1), \alpha_i)$ for $i > 0$ if A is deterministic.*

The acceptance conditions Acc defines which of the infinite runs are accepting. In the thesis, we consider the three types of acceptance conditions.

Definition 2.6.6. *A **Büchi acceptance condition** Acc is a subset F of Q, where the elements of F are called accepting states. An infinite run $\rho = \rho(0)\rho(1)...$ is called Büchi accepting if $Inf(\rho) \cap F \neq \emptyset$, it means that run ρ visits states of F infinitely often.*

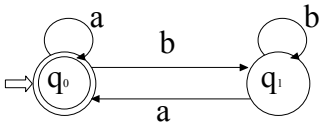

Figure 2.5 The deterministic Büchi automaton.

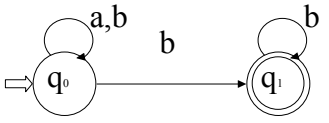

Figure 2.6 The non-deterministic Büchi automaton.

Definition 2.6.7. *A **Streett acceptance condition** Acc is a finite set of pairs (H_i, K_i) where H_i and K_i are subsets of Q ($Acc = \{(H_1, K_1), ..., (H_s, K_s)\}$). An infinite run $\rho = \rho(0)\rho(1)...$ is called Streett accepting if for each $i \in \{1, 2, ..., s\}$ $Inf(\rho) \cap H_i \neq \emptyset$ or $Inf(\rho) \cap K_i = \emptyset$.*

Definition 2.6.8. *A **Rabin acceptance condition** Acc is a finite set of pairs (H_i, K_i) where H_i and K_i are subsets of Q ($Acc = \{(H_1, K_1), ..., (H_s, K_s)\}$). An infinite run $\rho = \rho(0)\rho(1)...$ is called Rabin accepting if there is some $i \in \{1, 2, ..., s\}$ for which $Inf(\rho) \cap H_i = \emptyset$ and $Inf(\rho) \cap K_i \neq \emptyset$.*

The Streett condition is dual to the Rabin condition. It is therefore sometimes called complemented pair condition. Büchi acceptance condition can be considered as the special case of Streett $\{(F, Q)\}$ and Rabin acceptance $\{(\emptyset, F)\}$ conditions.

ω-automata with Büchi acceptance condition are called Büchi automata, with Streett acceptance condition - Streett automata, and with Rabin acceptance condition - Rabin automata.

The accepted language of the ω-automaton A ($L(A)$) is defined as a set of infinite words $\alpha \in \Sigma$ which have accepting run in A.

Example 2.6.1. *Let the language $L \subseteq \Sigma^\omega$ consist of all infinite words such that there are infinite many occurrences of the symbol a. The Büchi automaton recognizing the language is displayed in the figure 2.5 (the acceptance condition is $\{q_0\}$). The state q_0 can be reached only with the symbol a, so the good state will occur infinity often only if the symbol a will have infinitely many occurrences. However, if we consider the complement of the language L - containing all infinite words such that there are only finite many occurrences of the symbol a. It cannot be recognized by deterministic Büchi automaton, but it can be done by non-deterministic Büchi automaton (the figure 2.6). Using the transitions of the*

automaton in the figure 2.5 we can built the Rabin and Streett automata recognizing the complement of L. The acceptance condition for Streett automaton would be $(\emptyset, \{q_0\})$, to accept the word the condition $Inf(\rho) \cap \{q_0\} = \emptyset$ should be satisfied, that means that there is only finitely many symbols a. The Rabin acceptance condition would be $(\{q_0\}, \{q_1\})$.

It is known that the classes of languages accepted by non-deterministic Büchi automata (NBA), deterministic Streett automata (DSA), non-deterministic Streett automata (NSA), deterministic Rabin automata (DRA), non-deterministic Rabin automata (NRA) are the same. These are ω-regular languages and can be represented by ω-regular expressions (finite sums of expressions in the form $\alpha\beta^\omega$, where α and β are regular expressions over finite words, and language defined by β is non-empty and does not contain empty string). Deterministic Büchi automata (DBA) are less powerful, the class of DBA is a proper subclass of the ω-regular languages, it recognizes limit languages.

Chapter 3

Measure-Once Quantum Finite State Automata and Logic

In this chapter, we study the connection between language class recognized by measure-once quantum finite state automata and languages defined by first order logic, modular logic, and logic using generalized quantifiers - Lindström quantifier and group quantifier.

3.1 Measure-Once Quantum Finite State Automata and First Order Logic

The most popular notations of quantum finite state automata with bounded error recognize only regular languages but not all regular languages. The logical description of these language classes should be weaker than monadic second order logic described by Büchi, which follows from the theorem of Büchi - languages defined in monadic second order are regular languages. The first intention was to study “natural" subclasses of monadic second order logic. The most ”natural" subclass of monadic second-order logic is $FO[<]$.

Theorem 3.1. *If a language in alphabet Σ can be recognized by a measure-once quantum finite state automaton and it is $FO[<]$ definable, then it is trivial, i.e. an empty language or Σ^*.*

Proof. Let us suppose that there exists a language L which is not trivial, it can be recognized by MO-QFA, and it is first order definable. If the language L is recognized by a MO-QFA, it can also be recognized by a group finite state automaton (GFA). As L is not trivial, the corresponding minimal GFA has both accepting and rejecting states. Let

us look at the accepting states q_a. The state q_a can be reached by a word ω, and there exists a symbol σ, such that after reading the word σ^k ($k > 1$) the GFA returns in the state q_a. (As the automaton accepts non-trivial language, it means it has both accepting and rejecting states, that means we have a symbol σ which goes from accepting state to rejecting. As the automaton is a group automaton, after reading finite number of symbols σ the automaton returns to the accepting state.)

Now, we consider the syntactic monoid of the group automaton, which recognizes the language. As the automaton accepts words ω and $\omega\delta^k$, it means that the monoid has subgroup $\{1_M, \delta_\sigma, \delta_{\sigma^{k-1}}\}$, but from this follows that this monoid is not aperiodic, so the language cannot be first-order definable. □

3.2 Measure-Once Quantum Finite State Automata and Modular Logic

The next "natural" logic which describes a sub-class of regular languages is modular logic. If we consider languages in a single letter alphabet, it is easy to see that all such languages accepted by measure-once quantum finite state automaton can be defined by modular logic.

Lemma 3.1. *Languages in a single letter alphabet recognized by measure-once quantum finite state automata are definable by modular logic.*

Proof. If the language L is recognized by a MO-QFA, it can also be recognized by a group finite state automaton. Let us consider the group automaton recognizing the language $A_{QFA} = (Q, \Sigma, \delta, q_0, Q_{acc})$ (assume that $Q = \{q_0, q_1, ..., q_n\}$ and $\delta(q_i, a) = q_{i+1}$, it can be easily assumed as the function should be reversible). Now, we can construct the formula recognizing the language. The words accepted by the accepting state q_i can be described as $\exists^{(n+1,i)}(Q_a(x))$, and the language is the union of all accepted states - $\vee_{q_i \in Q_{acc}} \exists^{(n+1,i)}(Q_a(x))$. □

Lemma 3.2. *There exists a language that can be recognized by a measure-once quantum finite automaton, but cannot be defined by modular logic.*

Proof. To prove the lemma we will use the theorem of [51] - a regular language L is in $MOD[<]$ if and only if the syntactic monoid of language L is a solvable group. It is known that the symmetric group S_5 forms a non-solvable group. We will construct an automaton whose syntactic monoid forms the symmetric group S_5. In the figure 3.1, we show a group automaton whose syntactic monoid forms the symmetric group S_5. The

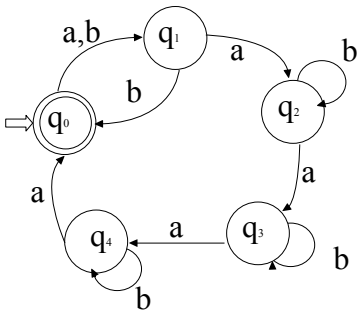

Figure 3.1 The group automaton whose syntactic monoid forms the symmetric group S_5.

input symbol a works as generation set $(1, 2, 3, 4, 5)$ and the symbol b as generation set $(1, 2)$. We can easily transform it to the measure-once quantum finite state automaton $A = (Q, \Sigma, \delta, q_0, Q_a, Q_r)$, where $Q = \{q_0, q_1, q_2, q_3, q_4\}$, $\Sigma = \{a, b, \}$, $Q_a = \{q_0\}$, $Q_r = \{q_1, q_2, q_3, q_4\}$, for letter a corresponds transformation with matrix

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}, \text{ and } \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$

for letter b. □

Theorem 3.2. *Every regular language L in $MOD[<]$ can be recognized by a measure-once quantum finite state automaton.*

Proof. Let us consider language $L \in MOD[<]$, its syntactic monoid is a solvable group. Let $\psi : \Sigma^* \to M$ be a homomorphism and $L = \psi^{-1}(N)$ for $N \subseteq M$, where M is the syntactic monoid of the language L. Now we show that the deterministic automaton $A(M, \Sigma, 1_M, \delta, N)$ with transition function $\delta(m, \sigma) = m \bullet_m \psi(\sigma)$ accepts the language L.

For every input word $w \in \Sigma^*$ ($w = w_1 w_2 ... w_k$) and every state $m \in M$ we have $\delta(m, w) = m \bullet_M \psi(w)$, as $\delta(m, w_1) = m \bullet_M \psi(w_1)$ by the definition of transition function and if $\delta(m, w_1..w_{k-1}) = m \bullet_M \psi(w_1...w_{k-1})$ then $\delta(m, w_1..w_k) = \delta(\delta(m, w_1...w_{k-1}), w_k) = \delta(m, w_1...w_{k-1}) \bullet_M \psi(w_k) = m \bullet_M \psi(w_1...w_{k-1}) \bullet_M \psi(w_k) = m \bullet_M \psi(w_1...w_k)$. Consequently, $\delta(1_M, w) = 1_M \bullet_M \psi(w) = \psi(w)$, it means $w \in L = \psi^{-1}(N)$ if and only if $\psi(w) \in N$ (by definition), if and only if $\delta(1_M, w) \in N$, if and only if w is accepted by the automaton.

As monoid M is a solvable group the automaton A will be a group automaton (the equation $\delta(m_1, \sigma) = \delta(m_2, \sigma)$ holds only if and only if $m_1 = m_2$, as $m_1 \bullet_M \psi(\sigma) \bullet \psi(\sigma)^{-}1 = m_2 \bullet_M \psi(\sigma) \bullet \psi(\sigma)^{-}1$). And the language will also be recognized by a measure-once quantum finite state automaton. □

However, if we extend our modular logic with additional successor function $S(x)$ which denotes the next positioned after x, and $MOD[S, <]$ denotes the language class defined by modular logic with additional successor function, then we get a language class which is not fully recognized by a measure-once quantum finite state automata.

Theorem 3.3. *Languages defined by the modular formula of the form* $\exists^{(n,m)} x(Q_a(x) \wedge Q_b(S(x)))$ *cannot be recognized by a measure-once quantum finite state automaton nor with a measure-many quantum finite state automaton.*

Proof. The minimal deterministic finite state automaton recognizing the language is displayed in the figure 3.2. The minimal deterministic finite state automaton is not a group finite automaton, therefore the language cannot be recognized by a measure-once quantum finite state automaton. As the minimal deterministic finites state automaton contains forbidden constructions [6] nor the language can be recognized by a measure-many quantum finite state quantum automaton. □

3.3 Measure-Once Quantum Finite State Automata and Generalized Quantifiers

The whole class of MO-QFA recognizable languages could not be described by using these “natural" subclasses of monadic second order logic, so less standard logic should be considered, one of extensions could be use of generalized quantifiers. Using Lindström quantifier, the following theorem has been proved:

Theorem 3.4. *A language can be recognized by a measure-once quantum finite automaton if and only if this language can be described by Lindström quantifier formula corresponding to the group languages using atomic formulas* $Q_a(x)$.

Proof. As group languages are those languages that are recognized by a group finite state automaton, these languages are also recognized by a measure-once quantum finite state automaton. So for given MO-QFA, that recognizes a language L in alphabet $\Sigma = \{a_1, ..., a_k\}$ the corresponding formula with Lindström quantifier the Lindström quantifier is over the language L in the same alphabet and the formula is $Q_L x(Q_{a_1(x)}, Q_{a_2(x)}, .., Q_{a_{k-1}(x)})$.

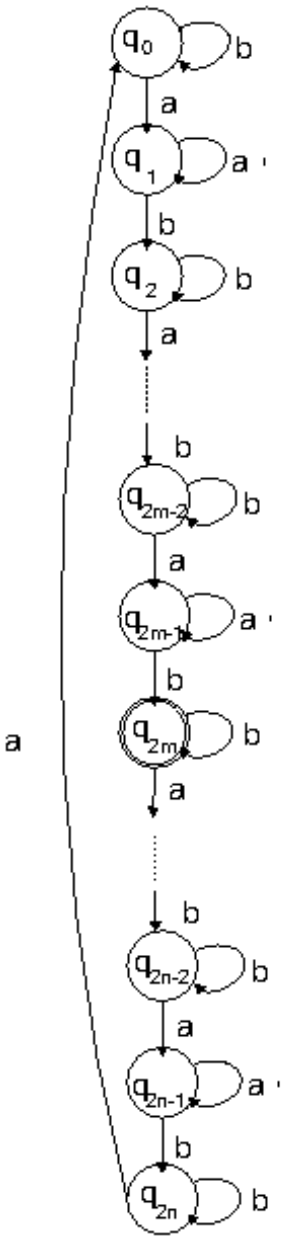

Figure 3.2 The deterministic finite state automaton recognizing language defined by $\exists^{(n,m)}x(Q_a(x) \wedge Q_b(S(x)))$.

For a given formula $Q_{LG}x(Q_{a_1}, Q_{a_2}, .., Q_{a_{s-1}})$ in alphabet $\Sigma_f = \{b_1, ..., b_k\}$ and $a_i \in \Sigma_f$ over a group language LG in alphabet $\Sigma_L = \{\delta_1, \delta_2, ..., \delta_s\}$ consider the language, that it defines. And look at mapping from Σ_f to Σ_{LG}. If the letter b_i is in the position x, then there are three possibilities:

1. Lindström quantifier has exactly one Q_{b_i} then for the letter b_i the corresponding letter is δ_j, where j is occurrence of Q_{b_i}.

2. Lindström quantifier contains more than one Q_{b_i} then for the letter b_i the corresponding letter is δ_j, where j is first occurrence of Q_{b_i}.

3. Lindström quantifier has none Q_{b_i} then for the letter b_i the corresponding letter is δ_s.

The transformation of MO-QFA that recognizes the given language for a letter b_i corresponds to transformation of MO-QFA that recognizes language LG for b_i mapping. □

Theorem 3.5. *A language accepted by a measure-once quantum finite state automaton if and only if it can be described the formula using only group quantifier.*

Proof. If a language L is accepted by a measure-once finite state automaton, let us consider its syntactic monoid $M(L)$. The syntactic monoid $M(L)$ forms a group so we can define the language as $\vee_{g\in\psi(N)}(\Gamma^{M(L),g}x)_{a_1}(x), Q_{a_2}(x), ..., Q_{a_n}(x)\rangle$, where $|i\rangle$ is mapped to σ_{a_i} ($(100...0)$ is mapped to σ_{a_1}, $(010...0)$ is mapped to σ_{a_2} and so on).

A measure-once quantum finite state automaton accepts the language if the transition monoid of the language forms a group. It has already been proved that the formula using only group quantifiers describes a language recognized by a group [37].

□

Chapter 4

Measure-Many Quantum Finite State Automata and Logic

In the chapter, the connection between measure-many quantum finite state automata with bounded error and first order logic and modular logic has been studied with respect to language recognition and acceptance probability of measure-many quantum finite state automata.

4.1 Measure-Many Quantum Finite State Automata and First Order Logic

In this section, we look at the connection between first order logic and measure-many quantum finite state automata. It was shown in the previous chapter that the languages described by a first order formula cannot be recognized by measure-once quantum finite state automata and vice versus except trivial languages. As measure-many quantum finite state automata recognize all languages that are accepted by measure-once quantum finite automata, it is clear there are such languages which are recognized by measure-many quantum finite state automata but not describable by first order logics. For example, the language that contains all words with length multiple of 3, is recognized by a measure-many quantum finite state automaton, but cannot be described by the first order formula.

Do we have a language that can be defined by first order logic, but cannot be recognized by a measure-many quantum finite state automaton? Yes, there are such languages. It is known [32] that the language $\{\{a,b\}^* b\}$ cannot be recognized by a measure-many quantum finite state automaton, but the language can be defined by first order formula $\forall x(last(x) \rightarrow Q_b(x))$. There are also languages which can be described

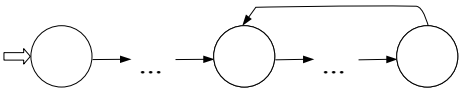

Figure 4.1 A DFA for a language in the single letter alphabet.

by the first order formula and can be recognized by measure-many quantum finite state automata, for example, the language containing all words starting with a symbol a. Our aim is to show constructions of first order formulas for whom there exists a measure-many quantum finite state automaton recognizing the language defined by the formula or there is no such measure-many quantum finite state automaton recognizing the language.

Theorem 4.1. *The languages in the single letter alphabet defined by a first order formula can be recognized by measure-many quantum finite state automata, but not vice versus.*

Proof. The deterministic finite state automaton for a regular language in the single letter alphabet is displayed in the figure 4.1.

We can construct the measure-many quantum finite state automaton for the language. The regular language in a single letter alphabet contains the words of the form - $a^{k'}a^{nm'}$, where k' ($k' \geqslant 0$) is the length of the constant part and m' ($m' \geqslant 0$) is the length of the cycle. Let us assume, that a deterministic finite state automaton recognizing a language in the single letter alphabet has k states for the constant part and m states for the cycle. The measure-many quantum finite state automaton has $k+m+1$ non-halting states ($k+1$ states are used for the first k letters and m are used for the cycle) and $k+m+3$ halting states. The transition function is defined as follows

$$V_{\#}(|q_0\rangle) = \tfrac{2}{3}|q_0\rangle + \tfrac{1}{3}\left|q_{k+1+m-k(mod\ m)}\right\rangle\ (k \neq m),$$
$$V_{\#}(|q_0\rangle) = \tfrac{2}{3}|q_0\rangle + \tfrac{1}{3}|q_{k+1}\rangle\ (k = m),$$
$$V_a(|q_i\rangle) = |q_{i+1}\rangle, \text{ where } i \in \{0, 1, ..., k-1, k+1, k+m-1\},$$
$$V_a(|q_k\rangle) = \tfrac{1}{2}|q_{2k+2m+2}\rangle + \tfrac{1}{2}|q_{2k+2m+3}\rangle,$$
$$V_a(|q_{k+m}\rangle) = |q_{k+1}\rangle,\ V_{\$}(|q_i\rangle) = |q_{i+k+m+1}\rangle\ (i \in \{0, 1, ..., k+m\}).$$

$|q_{2k+2m+2}\rangle$ is an accepting state, $|q_{2k+2m+3}\rangle$ is rejecting state, if $|q_i\rangle$ $(i < k+1)$ is an accepting state in the deterministic finite state automaton, then $|q_{i+k+m+1}\rangle$ is an accepting state in the measure-many quantum finite automaton and vice versus. If $|q_i\rangle$ $(i > k)$ is an

accepting state in the deterministic finite state automaton, then $|q_{i+k+m+2}\rangle$ is an accepting state in the measure-many quantum finite automaton and vice versus.

Consider a word $\omega \in L$ if it is accepted by the measure-many quantum finite state automaton. We can look at two cases -

1. the length of ω is less or equal with k, then ω will be accepted with probability at least $\frac{2}{3}$. It means that after reading ω the measure-many quantum finite state automaton will be in the state $q_{|\omega|-1}$ with probability $\frac{2}{3}$ and after reading the right end-marker the automaton passes from the state $q_{|\omega|-1}$ to the accepting state with probability $\frac{2}{3}$.

2. the length of ω is greater then k, then ω will also be accepted with probability at least $\frac{2}{3}$, after reading the first k letters the measure-many quantum finite state automaton is in the state q_{k+1} with probability $\frac{1}{3}$ and in the state q_k with probability $\frac{2}{3}$, after reading the next symbol the automaton proceeds in the state q_{k+2} with probability $\frac{1}{3}$ and it is in the accepting state $|q_{2k+2m+2}\rangle$ with probability $\frac{1}{3}$. After reading the rest of symbols a, the automaton is in the state $q_{k+1+(|\omega|-k)mod\ m}$ with probability $\frac{1}{3}$ and it is in the accepting state $|q_{2k+2m+2}\rangle$ with probability $\frac{1}{3}$. When the automaton reads the right end-marker, the word is accepted with probability at least $\frac{2}{3}$.

In the same way we can show, that if the input word $\omega \notin L$, then the measure-many quantum finite state automaton will reject it with probability at least $\frac{2}{3}$. □

As languages defined by the first order logic are subclass of the regular languages, measure-many quantum finite state automata can recognize languages defined by the first order logic. All regular languages in the single letter alphabet are not first order definable, and it means that all languages in the single letter alphabet recognized by measure-many quantum finite state automata cannot be defined by the first order logic. It can be proved by the following lemma.

Lemma 4.1. *The language in a single letter alphabet is defined by the first order logic if and only if its minimal deterministic finite state automaton contains a cycle with the length 1.*

Proof. It is easy to construct a first order formula defining the language recognized by the deterministic finite state automaton containing a cycle with the length 1. We will enumerate the states of the deterministic finite state automaton so that q_0 is the initial state, from the state q_i the automaton passes to the state q_{i+1}. For each accepting state the formula is made in the following way -

- if the state q_i is not the last state ($i \neq n$, $n+1$ - the number of the states), then the words accepted by the state q_i can be defined by first order formula $\exists x_1 x_2 ... x_i (first(x_1) \wedge S(x_1, x_2) \wedge ... \wedge S(x_{i-1}, x_i) \wedge last(x_i))$
- if the state q_n is the last state then the words accepted by the state q_n can be defined by first order formula $\exists x_1 x_2 ... x_n (first(x_1) \wedge S(x_1, x_2) \wedge ... \wedge S(x_{n-1}, x_n))$

We enumerate all the formulas ϕ of the accepting states from 1 to j, then the first order formula $\phi_1 \vee \phi_2 ... \vee \phi_j$ defines the language accepted by the deterministic finite state automaton.

Let us assume, that it is possible to construct the first order formula also for the unary deterministic finite state automaton containing a cycle with the length greater then 1. We will use the same notation as in the previous proof - the regular language in a single letter alphabet is given as $a^k a^{nm}$, where k is the length of the constant part and m is the length of the cycle. The monoid M recognizing the language is $M = \{1_M, \delta_a, \delta_{a^2}, ..., \delta_{a^k}, \delta_{a^{k+1}}, ..., \delta_{a^{k+m-1}}\}$ and it is easy to see that for the monoid M there is no such n for which $a^n = a^{n+1}$. Thus the monoid is not aperiodic and the assumption is wrong. It is not possible to construct the first order formula for the unary deterministic finite state automaton containing a cycle with the length greater then 1. □

Now, let us examine the forms of the first order logic using a larger alphabet, which are and which are not recognized by a measure-many quantum finite state automaton.

Lemma 4.2. *The languages defined by first order logic in the form* $\forall x(Q_\sigma(x))$ $(\sigma \in \Sigma)$ *can be recognized by a measure-many quantum finite state automaton.*

Proof. The measure-many quantum finite state automaton recognizing the language defined by formula in the form $\forall x(Q_\sigma(x))$ $(\sigma \in \Sigma)$ consists of three states - two halting $\{q_1, q_2\}$, where q_1 is an accepting state and q_2 - a rejecting state and one non-halting state q_0 which is also the initial state. The transition function for the left end-marker and σ is defined by unit matrix, the transition function for a letter γ ($\gamma \in \Sigma$ and $\gamma \neq \sigma$) is defined by the matrix $\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$ and the transition for the right end marker is defined by the matrix $\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$. □

Lemma 4.3. *The languages defined by a first order formula in the form*

$\exists x_1 x_2 ... x_n (first(x_1) \wedge \phi_1 \wedge S(x_1, x_2) \wedge \phi_2 \wedge S(x_2, x_3) \wedge ... \wedge S(x_{n-1}, x_n) \wedge \phi_n)$ $(\sigma \in \Sigma)$

where ϕ_i is in the form

$$Q_{\sigma_1}(x_i) \vee Q_{\sigma_2}(x_i) \vee ... \vee Q_{\sigma_s}(x_i)$$

$(\sigma_j \in \Sigma, i \in \{1, , 2..., n-1\})$ and ϕ_n is in the form $Q_{\sigma_1}(x_n) \vee Q_{\sigma_2}(x_n) \vee ... \vee Q_{\sigma_s}(n_i)\,[\wedge last(x_n])$ $(\sigma_j \in \Sigma)$ can be recognized by a measure-many quantum finite state automaton.

Proof. The language defined by the first order formula in the given form is accepted by the measure-many quantum finite state automaton

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

where $Q = \{q_0, q_1, ..., q_{2n+1}, q_{2n+2}\}$, $Q_{acc} = \{q_{2n+2}\}$ if the formula ϕ_n contains $last(x_n)$, $Q_{acc} = \{q_n, q_{2n+2}\}$ otherwise, $Q_{rej} = \{q_{n+1}, ..., q_{2n+1}\}$, the transition for the left end-marker is defined by a unit matrix, for the letter σ_i it is defined by $V_{\sigma_i}(|q_{j-1}\rangle) = |q_j\rangle$, if the formula ϕ_j contains $Q_{\sigma_i}(x_j)$ and $j < n+1$, $V_{\sigma_i}(|q_{j-1}\rangle) = |q_{n+j}\rangle$, if the formula ϕ_j does not contain $Q_{\sigma_i}(x_j)$. If ϕ_n contains $last(x_n)$ then $V_\sigma(|q_n\rangle) = |q_{2n+2}\rangle$. It is easy to see that the automaton accepts the language described by the formula. □

Theorem 4.2. *The languages defined by the first order formula in the form $\phi_1 \vee \phi_2 \vee ... \vee \phi_n$ where ϕ is in the form of the first order formula in the lemma 4.3.*

Proof. The language L defined by the formula is the union of the languages L_i defined by the formulas ϕ_i. The non-deterministic finite state automaton recognizing the language L is displayed in the figure 4.2, where L_{ϕ_i} is language defined by the first order formula ϕ_i and in the figure is used to identify the deterministic finite state automaton recognizing language L_{ϕ_i}.

It is easy to proof that a state q_i of the deterministic automaton recognizing the language is reachable from different state q_j and q_l by a letter a if $q_i = q_j$ or $q_i = q_l$ and for all input symbols $\sigma \in \Sigma$ $\delta(q_i, \sigma) = q_i$. □

Theorem 4.3. *The language defined by the first order formula in the form*

$$\exists x_1 x_2 ... x_n (Q_{\sigma_1} \wedge S(x_1, x_2) \wedge Q_{\sigma_2} \wedge S(x_2, x_3) \wedge ... \wedge S(x_{n-1}, x_n) \wedge Q_{\sigma_n})$$

where $\sigma_j \in \Sigma$ and $n > 2$ cannot be recognized by a measure-many quantum finite state automaton.

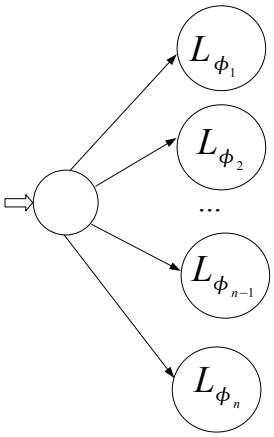

Figure 4.2 A non-deterministic automaton of $\phi_1 \vee \phi_2 \vee ... \vee \phi_n$.

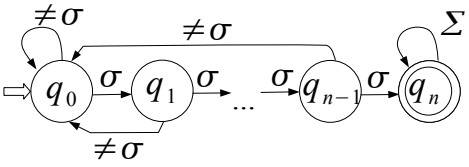

Figure 4.3 A deterministic automaton of $\exists x_1 x_2 ... x_n (Q_{\sigma_1} \wedge S(x_1, x_2) \wedge Q_{\sigma_2} \wedge S(x_2, x_3) \wedge ... \wedge S(x_{n-1}, x_n) \wedge Q_{\sigma_n})$.

Proof. At first we assume that $\sigma_i = \sigma_j$ for all i, j. The deterministic finite state automaton is shown in the figure 4.3. As the deterministic finite state automaton contains the "forbidden" construction [16], the language defined by the formula in the form $\exists x_1 x_2 ... x_n (Q_\sigma \wedge S(x_1, x_2) \wedge Q_\sigma \wedge S(x_2, x_3) \wedge ... \wedge S(x_{n-1}, x_n) \wedge Q_\sigma)$ is not recognized by a measure-many quantum finite state automaton.

Now, we will look at the case when there exist σ_i such that $\sigma_1 \neq \sigma_i$. We enumerate the states of the minimal deterministic finite state automaton accepting the language so that the automaton passes from the state q_i to the state q_j with symbol σ_j. Assume that the first symbol of the formula different from σ_1 is σ_i ($i \neq n$). It means that reading the symbol σ_1 the automaton passes from the state q_{i-1} to q_{i-1} and from q_{i-2} to q_{i-1}. At the same time, the automaton passes from the state q_i to q_1 reading symbol σ_1 (as $\sigma_1 \neq \sigma_i$). Thus we get that the automaton contains the "forbidden" construction [16]. If the first symbol of the formula different from σ_1 is σ_n, then with σ_n the automaton goes from q_0 to q_0, from q_1 to q_0 (as $n > 2$), and with the symbol σ_1 from q_0 to q_1. Thus we again get that the automaton contains the "forbidden" construction [16]. □

However, if we replace the operator $S(x, y)$ by $x < y$ in the theorem 4.3 we get a formula which describes a language accepted by measure-many quantum finite state

automata.

Lemma 4.4. *The language defined by the first order formula in the form*

$$\exists x_1 x_2 ... x_n (Q_{\sigma_1}(x_1) \wedge x_1 < x_2 \wedge Q_{\sigma_2}(x_2) \wedge x_2 < x_3 \wedge ... \wedge x_{n-1} < x_n \wedge Q_{\sigma_n}(x_n))$$

where $\sigma_j \in \Sigma$ is recognized by a measure-many quantum finite state automaton.

Proof. The language defined by the formula is $\Sigma^* \sigma_1 \Sigma^* \sigma_2 \Sigma^* ... \Sigma^* \sigma_n \Sigma^*$, it is known that it is accepted by Latvian quantum finite state automaton, and the languages accepted by Latvian quantum finite state automata, are also accepted by measure-many quantum finite state automaton. □

Additionally, we study the connection between accepting probabilities of quantum finite state automata and $FO[<]$ considering accepting probability of automata. From the fact that intersection of languages recognized by measure-once quantum finite state automata and languages defined by $FO[<]$ contains only trivial languages follows that there are languages recognized by measure-many quantum finite state automata which cannot be defined by $FO[<]$. However, there are $FO[<]$ languages recognized by measure-many quantum finite state automata with probability 1.

Let us look at the language class CL_1, which contains all languages defined by the following rules:

1. $a_i \in \Sigma$, $\{a_1 a_2 ... a_k\} \in CL_1$, $k \in N$
2. $a_i \in \Sigma$, $\{a_1 a_2 ... a_k \Sigma^*\} \in CL_1$, $k \in N$
3. if $L_i \in CL_1$ and $L_j \in CL_1$, then $\overline{L_i} \in CL_1$, $L_i \cup L_j \in CL_1$ and $L_i \cap L_j \in CL_1$.

These languages can be defined by the $FO[<]$ formula. $FO[<]$ formula describing the language can be constructed as follows:

1. $\exists x_1, x_2, ..., x_k (first(x_1) \wedge Q_{a_1}(x_1) \wedge S(x_1, x_2) \wedge Q_{a_2}(x_2) \wedge S(x_2, x_3) \wedge ... \wedge Q_{a_{k-1}}(x_{k-1}) \wedge S(x_{k-1}, x_k) \wedge Q_{a_k}(x_k) \{\wedge last(x_k)\}^*)$
2. if ϕ_i defines language $L_i \in CL_1$ and ϕ_j recognizes the language $L_j \in CL_1$, then $\neg \phi_i$ recognizes $\overline{L_i}$, $\phi_i \vee \phi_j$ - $L_i \cup L_j \in CL_1$, and $\phi_i \wedge \phi_j$ - $L_i \cap L_j \in CL_1$.

Lemma 4.5. *The languages in CL_1 can be recognized by a measure-many quantum finite state automaton with probability 1.*

Proof. Lets look at the language $L_i \in CL_1$ to construct the measure-many quantum finite state automaton for the language L_i, we need to represent the language L_i in the tree view. The representation tree is constructed in the following way:

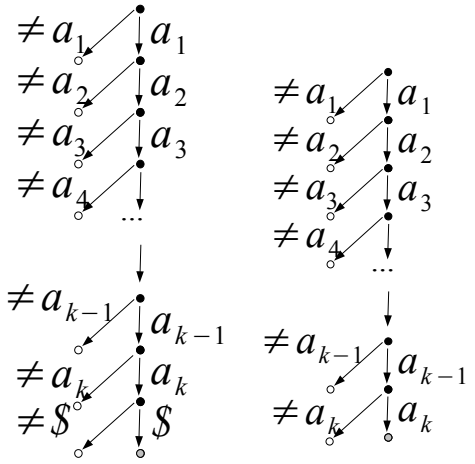

Figure 4.4 A representation tree of $\{a_1a_2...a_k\}$ (on the left) and $\{a_1a_2...a_k\Sigma^*\}$ (on the right).

- *The representation tree for the language* $\{a_1a_2...a_k\}$. The representation tree has $k+2$ levels. The edges of the tree are labelled with the letters from the alphabet $\Sigma \cup \$$. The nodes of the representation tree are coloured in three colours - white, grey, and black. White and grey nodes are leaves. Each black node has $|\Sigma|+1$ children, one outgoing edge for each letter. The parent node of the tree is black, and it is the first level of the tree. A node of level $1 < j \leq k+1$ is coloured black if the ingoing edge is labelled with a_{j-1} and white otherwise. A node of level $k+2$ is coloured grey if the ingoing edge is labelled with $\$$ and white otherwise.

- *The representation tree for the language* $\{a_1a_2...a_k\Sigma^*\}$. The representation tree has $k+1$ levels. The edges of the tree are labelled with the letters from the alphabet $\Sigma \cup \$$. The nodes of the representation tree are painted in three colours - white, grey, and black. White and grey nodes are leaves. Each black node has $|\Sigma|+1$ children, one outgoing edge for each letter. The parent node of the tree is black, and it is the first level of the tree. A node of level $1 < j \leq k$ is coloured black if the ingoing edge is labelled with a_{j-1} and white otherwise. A node of level $k+1$ is coloured grey if the ingoing edge is labelled with a_k and white otherwise.

- For $\overline{L_i}$ the L_i tree grey nodes are coloured white and the white nodes in grey.

- The tree of $L_i \cup L_j$ is union of the trees for L_i and L_j. If the edge with label a from the level k to $k+1$ in one of the trees goes to white leaf, then the sub-tree of the other tree is chosen in the final tree, if the leaf in the one of the trees is grey, then the final tree will have this edge.

- The tree of $L_i \cap L_j$ is intersection of the trees for L_i and L_j. If the edge with label a from the level k to $k+1$ in one of the trees goes to white leaf, then the final tree will have this edge, if the leaf in the one of the trees is grey, then the sub-tree of the other tree is chosen in the final tree.

The measure-many quantum finite state automaton accepting the language L_i with probability 1 is the automaton $A = (Q; \Sigma; \delta; q_0; Q_a; Q_r)$, where $Q = \{q_0, q_1, ..., q_n\}$, where n is the count of nodes in the representation tree of L_i. Q_a contains the states which correspond to nodes coloured in grey, Q_r contains the states which correspond to nodes coloured white. The transition function are defined by the representation tree. The edge (i, j, a_l) defines the transition $|q_i\rangle = |q_j\rangle$ for letter a_l. As each node has exactly one ingoing edge, the transition function of the automaton is unitary. □

From the above lemma follows:

Theorem 4.4. *$FO[<]$ contains languages which can be recognized by a measure-many quantum finite state automaton with probability 1.*

Lets look at the languages that can be recognized by MM-QFA with accepting probability 1 and which cannot be recognized by measure-once quantum finite state automata. Currently we have shown a specific class of languages which can be recognized by MM-QFA with probability 1 and which are first order definable. Naturally, questions arise:

- Are there other $FO[<]$ languages which can be recognized by MM-QFA with probability 1, but are not in the language class L_1?

  The answer to which is yes, for example, a language ab^*a can be recognized by MM-QFA with probability 1 and it is $FO[<]$ definable.

- Is it possible to give a characteristics of the languages which can be recognized by measure-many quantum finite state automata with probability 1, but which cannot be recognized by measure-once quantum finite state automata and are not $FO[<]$ definable?

  It is clear that such languages exist, for example, $a^{2k}b$ where $k \in Z$.

- What about recognition probability less than 1 - can we present $FO[<]$ languages which cannot be recognized by a measure-many quantum finite state automaton with probability 1?

$FO[<]$ contains such languages which can only be recognized by a measure-many quantum finite automaton with probability less than 1. One of examples is the language a^*b^*.

However, can we present a probability p, such that every measure-many quantum finite state automaton accepts language in $FO[<]$ with probability at least p.

Theorem 4.5. *There is no such probability p greater than $\frac{1}{2}$ for which the following holds:*

- *any measure-many quantum finite automaton accepts $FO[<]$ languages (recognized by MM-QFA) at least with probability p.*

Proof. Lets assume that it is possible to provide such probability p. We can express p as $\frac{1}{2} + l$. It is possible to find such n so that $l > \frac{3}{\sqrt{n-1}}$. At the same time it is known [4], that language L_n (L_n is defined as $a_1^*a_2^*a_3^*...a_n^*$) cannot be recognized with probability greater then $\frac{1}{2} + \frac{3}{\sqrt{n-1}}$. At the same time, the language L_n can be defined by a first order formula

$$\forall x_1, x_2, ..., x_n(\wedge_{i=1}^{n-1}(Q_{a_i}(x_i) \rightarrow \forall y(S(x_i, y) \rightarrow (Q_{a_i}(y) \vee Q_{a_{i+1}}(x_n))) \\ \wedge Q_{a_n}(x_n) \rightarrow \forall y(S(x_n, y) \rightarrow Q_{a_n}(y)))$$

We got a contradiction which means that such p does not exist. □

4.2 Measure-many quantum finite state automata and modular logic

In this section, we consider the connection between measure-many quantum finite automata and the modular logic. From the results of the previous chapter, we can get the following:

Theorem 4.6. *Languages in the language class $MOD[<]$ can be recognized by a measure-many quantum finite state automaton with probability 1.*

Now we extend the language class $MOD[<]$ by $FO[<]$ formulas, denoted by $MOD+FO$.

Lemma 4.6. *Languages in the language class $MOD + FO$ defined by the formula of the form $\exists^{(n,m)}x(Q_a(x) \wedge \forall y(S(x, y) \wedge Q_b(y)))$ cannot be recognized by a measure-many quantum finite state automata.*

Proof. The minimal deterministic finite state automaton (DFA) recognizing the language is displayed in the Figure 3.2. The automaton is minimal DFA as we need to remember n fragments of ab, that means we have $2n$ states. As the minimal DFA contains forbidden constructions [6] (an automaton has the following transitions $q_i \xrightarrow{x} q_j$, $q_j \xrightarrow{x} q_j$, and $q_j \xrightarrow{y} q_i$) the language cannot be recognized by a measure-many quantum finite state automaton. □

However, we have obtained also positive results for languages belonging to the language class $MOD[<] + FO$, but not to $MOD[<]$.

Lemma 4.7. *The language defined by the formula* $\exists x(Q_b(x) \wedge \exists^{2,1} y((x < y) \wedge Q_a(y)))$ *is accepted by a measure-many quantum finite state automaton.*

Proof. The measure-many quantum finite state automaton accepting the language is following $A = (Q; \Sigma; \delta; q_0; Q_{acc}; Q_{rej})$, where $Q = \{q_0, q_1, q_2, q_{acc_1}, q_{acc_2}, q_{rej_1}, q_{rej_2}\}$, $Q_acc = \{q_{acc_1}, q_{acc_2}\}$, $Q_rej = \{q_{rej_1}, q_{rej_2}\}$, $\Sigma = \{a, b\}$, and δ is defined as:

$$\mathrm{V}_{\#}(|q_i\rangle) = |q_i\rangle \text{for all } q_i \in Q$$
$$\mathrm{V}_a(|q_0\rangle) = |q_0\rangle, V_a(|q_1\rangle) = |q_2\rangle, V_a(|q_2\rangle) = |q_1\rangle$$
$$\mathrm{V}_b(|q_0\rangle) = \tfrac{2}{3}|q_0\rangle + \tfrac{\sqrt{2}}{3}|q_1\rangle + \tfrac{1}{\sqrt{3}}|q_{acc_1}\rangle$$
$$\mathrm{V}_b(|q_1\rangle) = \tfrac{\sqrt{2}}{3}|q_0\rangle + \tfrac{1}{3}|q_1\rangle - \tfrac{\sqrt{2}}{\sqrt{3}}|q_{acc_1}\rangle$$
$$\mathrm{V}_b(|q_2\rangle) = |q_{acc_2}\rangle$$
$$\mathrm{V}_\$(|q_0\rangle) = |q_{rej_1}\rangle, V_\$(|q_1\rangle) = |q_{rej_2}\rangle, V_\$(|q_2\rangle) = |q_{acc_2}\rangle$$

Now, let us look at input words, if an input is:

- a^*. The left end-marker does not changes the initial superposition $|q_0\rangle$. By reading a^* automaton stays in the superposition $|q_0\rangle$, when the right end-marker is read, the word is rejected with a probability 1.

- a^*bb^*. After reading a^*, the automaton is in the superposition $|q_0\rangle$, when the first b is read, the automaton changes its superposition to $\frac{\sqrt{2}}{\sqrt{3}}|q_0\rangle + \frac{1}{\sqrt{3}}|q_1\rangle$ with probability $\frac{2}{3}$ and accepts the input with probability $\frac{1}{3}$. While the automaton reads b the superposition is not changed, it stays $\frac{\sqrt{2}}{\sqrt{3}}|q_0\rangle + \frac{1}{\sqrt{3}}|q_1\rangle$ with probability $\frac{2}{3}$, when the left end-marker is read the input is rejected with the probability $\frac{2}{3}$.

- $a^b(b^*a^{2n})^*$. After reading a^*b the automaton is in the superposition $\frac{\sqrt{2}}{\sqrt{3}}|q_0\rangle + \frac{1}{\sqrt{3}}|q_1\rangle$ with probability $\frac{2}{3}$ and acceptance probability is $\frac{1}{3}$, while reading b the superposition does not change. When an input symbol a is read the superposition is changed to $\frac{\sqrt{2}}{\sqrt{3}}|q_0\rangle + \frac{1}{\sqrt{3}}|q_2\rangle$, and the second a changes the superposition back to $\frac{\sqrt{2}}{\sqrt{3}}|q_0\rangle + \frac{1}{\sqrt{3}}|q_1\rangle$. As in the previous case, the input is rejected with $\frac{2}{3}$.

- $a^b(b^*a^{2n})^*a$. As we have already identified - after reading $a^b(b^*a^{2n})^*$, the automaton is in the superposition $\frac{\sqrt{2}}{\sqrt{3}}|q_0\rangle + \frac{1}{\sqrt{3}}|q_1\rangle$ with probability $\frac{2}{3}$ and acceptance probability is $\frac{1}{3}$, by reading a the superposition is changed to $\frac{\sqrt{2}}{\sqrt{3}}|q_0\rangle + \frac{1}{\sqrt{3}}|q_2\rangle$, and after reading the left end-marker the input is accepted by $\frac{5}{9}$.

- $a^b(b^*a^{2n})^*aba, b^*$. After reading $a^b(b^*a^{2n})^*a$ is in the superposition $\frac{\sqrt{2}}{\sqrt{3}}|q_0\rangle + \frac{1}{\sqrt{3}}|q_2\rangle$ and accepts the input with $\frac{1}{3}$, when the b is read the word is accepted with probability $\frac{5}{9}$, that means input will be accepted with probability at least $\frac{2}{3}$.

□

Chapter 5

Latvian Quantum Finite State Automata and Logic

The chapter contains the results obtained in the connection between Latvian quantum finite state automata and logic. The language recognition power of Latvian quantum finite state automata has been compared with the languages recognized by first order languages and languages recognized by modular logic.

5.1 Latvian Quantum Finite State Automata and First Order Logic

Measure-once quantum finite state automata are special case of Latvian quantum finite state automata and at the same time Latvian quantum finite state automata recognizes a strict subclass of the languages accepted by measure-many quantum finite state automata [1]. In the previous chapters, we illustrated a relation between languages accepted by measure-once quantum finite state automata, measure-many quantum finite state automata and first order definable languages. The aim of this section is to illustrate where the language class recognized by Latvian quantum finite automata is located.

Lemma 5.1. *Languages defined by the first order formula in the form*

$$\exists x_1, x_2, ..., x_k (Q_{a_1}(x_1) \wedge Q_{a_2}(x_2) \wedge ... \wedge Q_{a_3}(x_3) \wedge (x_1 < x_2 < .... < x_3))$$

where $a_i \in \Sigma$ can be recognized by a Latvian quantum finite state automaton.

Proof. The formula of the lemma describes the language $\Sigma^* a_1 \Sigma^* a_2 \Sigma^* ... \Sigma^* a_n \Sigma^*$, which is known to be recognized by Latvian quantum finite state automata [1] Theorem 5. □

Theorem 5.1. *There exists languages which belongs to the intersection of $FO[<]$ and languages recognized by measure-many quantum finite state automata, but cannot be recognized by a Latvian quantum finite state automaton.*

Proof. To prove the theorem we will prove the following lemma:

Lemma 5.2. *Languages described by the first-order formulas of the form*

$$\exists x_1, x_2, ..., x_n(first(x_1) \wedge Q_{a_1}(x_1) \wedge Q_{a_2}(x_2) \wedge ... \wedge Q_{a_n}(x_n) \wedge (x_1 < x_2 < ... < x_n)) \tag{5.1}$$

can be recognized by a measure-many quantum finite state automaton, but it cannot be recognized by a Latvian quantum finite state automaton.

The language described by the first order formula 5.1 can be described as

$$a_1\Sigma^* a_2\Sigma^* a_3...a_n\Sigma^*.$$

As

$$a_1\Sigma^* a_2\Sigma^* a_3...a_n\Sigma^* \cup \neg\Sigma^* a_2\Sigma^* a_3...a_n\Sigma^* = a_1\Sigma^*$$

and the language $\Sigma^* a_2\Sigma^* a_3...a_n\Sigma^*$ is accepted by Latvian quantum finite state automata and languages accepted by Latvian quantum finite state automata are closed under union and complement, but the language $a_1\Sigma^*$ is not recognized by LQFA, the language $a_1\Sigma^* a_2\Sigma^* a_3...a_n\Sigma^*$ is not accepted by Latvian quantum finite state automaton. However, it can be accepted by measure-many quantum automaton. It can be easily constructed from the automaton recognizing the language $\Sigma^* a_1\Sigma^* a_2...a_n\Sigma^*$, where we change the transition of the initial state and all symbols except a_1, so that the initial state is changed to a rejecting state.

In fact, the languages described by the first order formulas of the form $\forall x(first(x) \wedge \psi(x))$ and $\forall x(last(x) \wedge \psi(x))$, where $\psi(x)$ is a first order formula containing $Q_a(x)$ ($a \in \Sigma$) cannot be accepted by Latvian quantum finite state automata. □

5.2 Latvian Quantum Finite State Automata and Modular Logic

Similarly to the previous section in this section, we want to locate languages recognized by Latvian quantum finite state automata in connection to the relation between measure-once quantum finite state automata, measure-many quantum finite state automata and languages defined by modular logic.

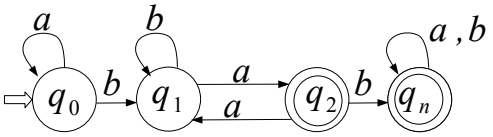

Figure 5.1 The deterministic finite state automaton recognizing the language defined by the formula $\exists x(Q_b(x) \wedge \exists^{2,1} y((x < y) \wedge Q_a(y)))$.

As measure-once quantum finite state automata are special case of Latvian quantum finite state automata we can conclude the following:

Lemma 5.3. *Languages in $MOD[<]$ are recognized by Latvian quantum finite state automata.*

Lemma 5.4. *Languages in the language class $MOD[S, <]$ defined by the formula of the form $\exists^{(n,m)} x(Q_a(x) \wedge Q_b(S(x)))$ cannot be recognized by Latvian quantum finite state automata.*

Lemma 5.5. *The language defined by the formula $\exists x(Q_b(x) \wedge \exists^{2,1} y((x < y) \wedge Q_a(y)))$ is accepted by a Latvian quantum finite state automaton.*

Proof. In the figure 5.1 is displayed a deterministic finite state automaton recognizing the language defined by the formula. Now, we examine the monoid M accepting the language. $M = \{1_M, \delta_a, \delta_b, \delta_{ab}, \delta_{ba}, \delta_{aba}, \delta_{bab}\}$ and the binary operation is defined in the following table:

	1_M	δ_a	δ_b	δ_{ab}	δ_{ba}	δ_{aba}	δ_{bab}
1_M	1_M	δ_a	δ_b	δ_{ab}	δ_{ba}	δ_{aba}	δ_{bab}
δ_a	δ_a	1_M	δ_{ab}	δ_b	δ_{aba}	δ_{ba}	δ_{bab}
δ_b	δ_b	δ_{ba}	δ_b	δ_{bab}	δ_{ba}	δ_{bab}	δ_{bab}
δ_{ab}	δ_{ab}	δ_{aba}	δ_{ab}	δ_{bab}	δ_{aba}	δ_{bab}	δ_{bab}
δ_{ba}	δ_{ba}	δ_b	δ_{bab}	δ_{bab}	δ_{bab}	δ_{ba}	δ_{bab}
δ_{aba}	δ_{aba}	δ_{aba}	δ_{bab}	δ_{ab}	δ_{bab}	δ_{aba}	δ_{bab}
δ_{bab}	δ_{bab}	δ_{bab}	δ_{bab}	δ_{bab}	δ_{bab}	δ_{bab}	δ_{bab}

and $N = \{\delta_{ba}, \delta_{aba}, \delta_{bab}\}$. $E(M) = \{e, b, aba, bab\}$ and the monoid M is a block group, it means that the language is recognized by Latvian quantum finite state automaton. □

We have shown that Latvian quantum finite state automaton also recognizes languages in $MOD+FO$, but not in $MOD[<]$. However, not all languages recognized by measure-many quantum finite state automata in $MOD + FO$ is also recognized by a Latvian

quantum finite state automaton. For example, language described by $\forall x(first(x) \rightarrow (Q_b(x) \wedge \exists^{(2,1)}(Q_a(y) \wedge x < y)))$ can be recognized by a measure-many quantum finite state automaton, but cannot be recognized by a Latvian quantum finite state automaton.

5.3 Latvian Quantum Finite State Automata and Generalized Quantifiers

The language class accepted by Latvian quantum finite state automata can be similarly characterized as languages recognized by measure-once quantum finite state automaton using group quantifier.

Theorem 5.2. *A language L is recognized by Latvian quantum finite state automaton if and only if it is described by the formula built from the formulas of the form $\exists x_1, x_2, ..., x_n(\psi_0(x_1) \wedge Q_{a_1}(x_1) \wedge \psi_1(x_1, x_2) \wedge Q_{a_2}(x_2)...Q_{a_n}(x_n) \wedge \psi_n(x_n))$ by means of connectivities $\vee, \wedge, \neg$, where $\psi_i(x_i, x_{i+1})$ is a formula containing only group quantifiers and denoting "the input fragment starting with x_i+1 and ending with $x_{i+1}-1$ satisfies ψ_i" ($1 \leqslant i \leqslant n-1$), $\psi_0(x_1)$ is a formula containing only group quantifiers and denotes "the initial fragment before x_1 (ending at $x-1$) satisfies ψ_0, and $\psi_n(x_n)$ is a formula containing only group quantifiers and denotes "the final input fragment starting with $x_n + 1$ satisfies ψ_n".*

Proof. We have already stated that Latvian quantum finite state automaton recognizes Boolean combination of the languages in the form $L_0a_1L_1...L_{n-1}a_nL_n$, where L_i is a group language. The connectivities of the formula is equivalent to the Boolean combinations of the languages, so its enough to consider languages in the form $L_0a_1L_1...L_{n-1}a_nL_n$ and formulas of the form $\exists x_1, x_2, ..., x_n(\psi_0(x_1) \wedge Q_{a_1}(x_1) \wedge \psi_1(x_1, x_2) \wedge Q_{a_2}(x_2)...Q_{a_n}(x_n) \wedge \psi_n(x_n))$.

From the theorem 3.5, we can conclude that $\psi(x_i, x_{i+1})$ describes that the fragment between $x_i + 1$ and $x_{i+1} - 1$ is a group language, $\psi(x_1)$ describes that the initial fragment before x_1 is a group language, and $\psi(x_n)$ describes that the fragment after x_n forms a group language. It means that the formula $\exists x_1, x_2, ..., x_n(\psi_0(x_1) \wedge Q_{a_1}(x_1) \wedge \psi_1(x_1, x_2) \wedge Q_{a_2}(x_2)...Q_{a_n}(x_n) \wedge \psi_n(x_n))$ describes the languages in the form $L_0a_1L_1...L_{n-1}a_nL_n$, where L_i is a group language. Thus the language can be recognized by Latvian quantum finite state automata.

The language L_i is a group language, we can conclude that it can be described by a formula $\psi_i()$ using only group quantifiers. Thus $L_0a_1L_1...L_{n-1}a_nL_n$ can be described as $\exists x_1, x_2, ..., x_n(\psi_0(x_1) \wedge Q_{a_1}(x_1) \wedge \psi_1(x_1, x_2) \wedge Q_{a_2}(x_2)...Q_{a_n}(x_n) \wedge \psi_n(x_n))$, where

$\psi_i(x_i, x_{i+1})$ is a formula containing only group quantifiers and denoting "the input fragment starting with $x_i + 1$ and ending with $x_{i+1} - 1$ satisfies ψ_i" ($1 \leqslant i \leqslant n - 1$), $\psi_0(x_1)$ is a formula containing only group quantifiers and denotes "the initial fragment before x_1 (ending at $x - 1$) satisfies ψ_0, and $\psi_n(x_n)$ is a formula containing only group quantifiers and denotes "the final input fragment starting with $x_n + 1$ satisfies ψ_n". $\square$

Chapter 6

Quantum Automata over Infinite Words

This chapter is devoted to the study of the quantum finite state automaton over infinite words. We adapt the definition of the automata for infinite input and study languages accepted by a quantum Büchi automaton.

6.1 Definition of Quantum Finite State Automata over Infinite Words

In this section, we will adapt definition of quantum finite state automata [38] for infinite words.

Definition 6.1.1. *A **quantum finite state over infinite words or quantum ω-automaton** is a quintuple $A = (Q, \Sigma, \delta, q_0, Acc)$*

- *Q is a finite set of states,*
- *Σ is a finite input alphabet,*
- *δ is the transition function $\delta : Q \times \Sigma \times Q \to C_{[0,1]}$, which represents the amplitudes that flows from the state q to the state q' after reading symbol σ,*
- *$q_0 \in Q$ is the initial state,*
- *Acc is the acceptance component.*

For all states $q_1, q_2, q' \in Q$ and symbols $\sigma \in \Sigma$, the function δ must be unitary, thus the function satisfies the condition

$$\sum_{q'} \overline{\delta(q_1, \sigma, q')}\delta(q_2, \sigma, q') = \begin{cases} 1 \ (q_1 = q_2) \\ 0 \ (q_1 \neq q_2) \end{cases} .$$

The linear superposition of the automaton's A states is represented by an n-dimensional complex unit vector, where $n = |Q|$. The vector is denoted by $|\phi\rangle = \sum_{i=1}^{n} \alpha_i |q_i\rangle$, where $\{|q_i\rangle\}$ is the set orthonormal basis vectors corresponding to the states of the automaton A. We will use the density matrix to represent the state of the automaton $\rho = |\psi\rangle\langle\psi|$.

The transition function δ is represented by a set of unitary matrices $\{V_\sigma\}_{\sigma\in\Sigma}$, where V_σ is the unitary transition of the automaton A after reading the symbol σ and is defined by $V_\sigma(|\ q\rangle) = \sum_{q'\in Q} \delta(q,\sigma,q')\ |\ q'\rangle$.

A computation of the automaton A on an input word $\alpha = a_1a_2a_3.... \in \Sigma^\omega$ proceeds as follows:

- It starts computation in the superposition $\rho_0 =|\ q_0\rangle\langle q_0\ |$.
- An unitary transition corresponding to the current input letter is performed.

Definition 6.1.2. *Let $A = (Q, \Sigma, \delta, q_0, Acc)$ be a quantum ω automaton. A **run** of A on an ω-word $\alpha = a_1a_2... \in \Sigma^\omega$ is an infinite sequence of density operators $\rho_\omega = \rho(0)\rho(1)\rho(2)...$, where $\rho(j) = |\psi_j\rangle\langle\psi_j|$ ($|\psi_j\rangle = \sum_{i=1}^{n} \alpha_i |q_i\rangle$ and $\{|q_i\rangle\}$ is the set of the automaton's A states), such that the following holds:*

1. $\rho(0) = \rho_0$
2. $\rho(i) = U_{a_i}\rho(i-1)U_{a_i}^\dagger$ *for* $i > 0$.

The acceptance conditions can be viewed in the similar way as for classical ω-automata. We examine quantum ω-automata with Büchi, Streett and Rabin acceptance condition and we use the abbreviations: QBA for Büchi quantum automata, QSA for Streett quantum automata, and QRA for Rabin quantum automata.

Definition 6.1.3. *A density matrix $\rho = |\psi\rangle\langle\psi|$ ($|\psi\rangle = \sum_{i=1}^{n} \alpha_i |q_i\rangle\ \langle q_i|$) is called F **accepting** ($F \subseteq Q$) with probability p if*

$$\sum_{q_i\in F} |\alpha_i|^2 > p,$$

which is acceptance probability of the superposition.

Definition 6.1.4. *A **Büchi acceptance condition for quantum case** Acc is a subset F of Q, where the elements of F are called accepting states. An infinite run $\rho_\omega = \rho(0)\rho(1)\rho(2)...$ is called Büchi accepting with probability p if run ρ_ω contains infinitely many F accepting density matrices.*

Definition 6.1.5. *A **Streett acceptance condition** Acc is a finite set of pairs (H_i, K_i) where H_i and K_i are subsets of Q ($Acc = (H_1, K_1), ..., (H_s, K_s)$. An infinite run $rho_\omega = \rho(0)\rho(1)\rho(2)...$ is called Streett accepting with probability p if for each $i \in \{1, 2, ..., s\}$ ψ_ρ contains infinite number of H_i accepting density matrices or ψ_ω contains only finite number of K_i accepting density matrices.*

Definition 6.1.6. *A **Rabin acceptance condition** Acc is a finite set of pairs (H_i, K_i) where H_i and K_i are subsets of Q ($Acc = (H_1, K_1), ..., (H_s, K_s)$. An infinite run $\rho_\omega = \rho(0)\rho(1)\rho(2)...$ is called Rabin accepting with probability p if there is some $i \in \{1, 2, ..., s\}$ for which ρ_ω contains only finite number of H_i accepting and K_i accepting density matrices.*

The language accepted by a quantum ω-automata A with the alphabet Σ and cut-point λ, denoted $L_\lambda(A)$, is defined as set of infinite words $\sigma \in \Sigma$ that has accepting run with probability $p > \lambda$ in A. A quantum ω-automaton accepts language with bounded error if there exists an $\epsilon > 0$ such that for all accepting runs probability is greater than $\lambda + \epsilon$.

Example 6.1.1. *Let us consider QBA A recognizing the ω-language $L_1 = (L_1')^\omega$, where $L_1' = \{a^{3n} | n \leq 0\}$ in the alphabet $\{a, b\}$. The automaton has:*

- $Q = \{q_0, q_1\}$,
- *V_b is identity matrix, $V_a = \begin{pmatrix} cos\frac{2\pi}{3} & sin\frac{2\pi}{3} \\ -sin\frac{2\pi}{3} & cos\frac{2\pi}{3} \end{pmatrix}$,*
- $Acc = \{q_0\}$.

The automaton can be in three different states $\mid \psi_0\rangle = \mid q_0\rangle$, $\mid \psi_1\rangle = cos\frac{2\pi}{3} \mid q_0\rangle - sin\frac{2\pi}{3} \mid q_1\rangle$, and $\mid \psi_2\rangle = cos\frac{4\pi}{3} \mid q_0\rangle - sin\frac{4\pi}{3} \mid q_1\rangle$, from which the accepting density matrix is for state $\mid q_0\rangle$. By reading an a automaton changes the superposition, reading input symbol b the super position stays the same, to have an infinitely many superpositions $\mid q_0\rangle$ the input should be in the language L_1. The automaton recognizes the language with probability equal to 1. We assume that the cut point of the language is $\frac{1}{2}$.

Example 6.1.2. *Let us consider another example with Streett acceptance condition recognizing the language containing infinite words with finite number of b.*

The automaton has:

- $Q = \{q_0, q_1\}$,
- *V_a is identity matrix, $V_b = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$,*

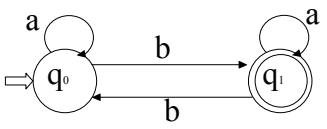

Figure 6.1 The group Büchi automaton recognizing the language consisting of the words which have only finite number of b.

- $Acc = \{\emptyset, |q_1\rangle\langle q_1|\}$.

The run of automaton can have in two different superpositions $\mid q_0\rangle$ *and* $\mid q_1\rangle$*. As the first condition is always false for each run, we have to consider the second condition, that means, that there is only finite number of* q_1 *accepting superpositions, in other words, we can have only finite number of* b*s. The automaton recognizes the language with probability equal to 1.*

6.1.1 Group Automata over Infinite Words

As the language class recognized by measure-once quantum finite state automata over finite words with bounded error is exactly the class of languages accepted by group finite automaton, we are also considering group automata over infinite words.

Definition 6.1.7. *A group automata over infinite words is a deterministic* ω *- automata with the restriction that for every state* $q \in Q$ *and every input symbol* $\sigma \in \Sigma$ *there exists exactly one state* $q' \in Q$ *such that* $\delta(q', \sigma) = q$*.*

We consider group ω-automata with Büchi (GBA), Strettt (GSA) and Rabin acceptance condition (GRA).

Example 6.1.3. *The figure 6.1 contains* GSA *for the language consisting of the words which have only finite number of* b *with the same acceptance condition as in quantum case.*

6.2 Quantum Finite State Automata over Infinite Words with Büchi Acceptance Condition

In this section, we consider languages recognized by quantum finite state ω - automata with Büchi acceptance condition with bounded error.

Lemma 6.1. *A language $L_\omega \subseteq \Sigma^\omega$ is recognized by a Büchi quantum finite state automaton if and only if L_ω is a limit language of $L \subseteq \Sigma^*$ and L is recognized by a measure-once quantum finite state automaton.*

Proof. Let L be a language recognizable by a measure-once quantum finite state automaton $A_{MO-QFA_1} = (Q_1, \Sigma, \delta_1, q_0, Q_{acc_1})$ with acceptance probability p. A word is accepted by A_{MO-QFA_1} if after reading the last symbol it is in accepting superposition ($\sum_{q_i \in Q_{acc_1}} |\alpha_i|^2 \geq p$).

Let us consider a Büchi quantum finite state automaton $A_{QBA_1} = (Q_1, \Sigma, \delta_1, q_0, Acc_1)$, where $Acc_1 = Q_{acc_1}$. Now, we look at the infinite word $\alpha \in lim(L)$, it means that α has infinitely many prefixes in L. It is easy to see that if A_{QBA_1} reads α it has infinitely many Q_{acc_p}-accepting superpositions. A_{QBA_1} recognizes $lim(L)$.

Let L_ω be a language recognized by a Büchi quantum finite state automaton $A_{QBA_2} = (Q_2, \Sigma, \delta_2, q_0, Acc_2)$. We will also study a measure-once quantum finite state automaton $A_{MO-QFA_2} = (Q_2, \Sigma, \delta_2, q_0, Q_{acc_2})$, where $Q_{acc_2} = Acc_2$. Now if an infinite input $\alpha \in L_\omega$ is considered, then it is easy to see if it is recognized by A_{QBA_2}, then infinitely many prefixes are recognized by A_{MO-QFA_2}. It means that $L_\omega = lim(L)$ and L is recognized by A_{MO-QFA_2}. □

Similarly we can prove the following lemma:

Lemma 6.2. *A language $L_\omega \subseteq \Sigma^\omega$ is recognized by a Büchi group finite state automaton if and only if L_ω is a limit language of a group language (language recognized by a group finite state automaton).*

As measure-once quantum finite state automata recognize exactly group languages, from the previous results we get the following:

Theorem 6.1. *A language L_ω can be recognized by a QBA with bounded error if and only if it can be accepted by GBA.*

Lemma 6.3. *QBA with bounded error can't recognize whole language class accepted by DBA.*

Proof. Let us consider ω-language $L_a = \{\alpha | \alpha \; has \; infinitely \; many \; a\}$ in the alphabet $\Sigma = \{a, b\}$. First assume that QBA with bounded error can recognize the language L_a. In this case, we can construct the GBA for the language. Now, let us consider a word $w \in \Sigma^*$ by which we reach an accepting state, it is clear that there exists a word b^k ($k > 0$) such that wb^k also will be in the accepting state, it means that the word $w(b^k)^\omega$ will be accepted, but it does not contain infinitely many a. So the language L_a cannot be accepted by QBA with bounded error, but L_a is recognized by a DBA (the figure 6.2).

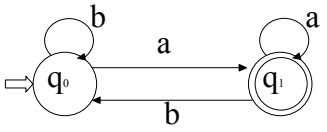

Figure 6.2 The deterministic Büchi automaton recognizing the language L_a.

□

From the above mentioned results we get the following:

Theorem 6.2. *QBA with bounded error accepts the proper subset of DBA which is equal to the languages accepted by GBA.*

It is known that deterministic Rabin automata and deterministic Streett automata can recognize larger language class than deterministic Büchi automata. What about quantum case?

Theorem 6.3. *The language class accepted by QBA with bounded error is the proper subset of the languages accepted by QSA.*

Proof. Let us consider language containing only finitely many input symbols b. It is known that this language cannot be recognized by DBA, so it cannot also be recognized by QBA with bounded error. However, it can be recognized by QSA, see Example 6.1.2. At the same time, as we have already mentioned Büchi finite state automata can be seen as special case of Streett finite state automata. □

6.3 Closure Properties of Quantum Finite State Automata over Infinite Words with Büchi Acceptance Condition

In this section, we consider a basic closure properties of the language accepted by Büchi quantum finite state automata with bounded error. We will study the standard set operations - union, intersection, and complementation. It turns out that the class of languages accepted by a Büchi quantum finite state automata with bounded error is closed under a union, but not under intersection and complementation.

Theorem 6.4. *If ω-languages L_{ω_1} and L_{ω_2} are recognized by some QBA then $L_{\omega_1} \cup L_{\omega_2}$ is also recognized by a QBA.*

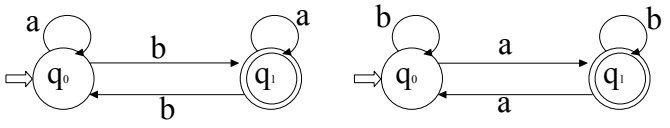

Figure 6.3 The deterministic Büchi automata recognizing the language L_a and L_b.

Proof. Let us consider ω-languages L_{ω_1} and L_{ω_2} recognized by Büchi quantum finite state automata. From the results of the previous section, we get that $L_{\omega_1} = lim(L_1)$ and $L_{\omega_2} = lim(L_2)$ and the languages L_1 and L_2 are recognized by group finite state automata. It is known that group finite state automata are closed under a union, that means we have a GFA $A_{L_1 \cup L_2} = (Q, \Sigma, \delta_1, q_{init}, Q_{acc})$ which recognizes $L_1 \cup L_2$, from the lemma 6.2 we get that $A_{lim(L_1 \cup L_2)} = (Q, \Sigma, \delta_1, q_{init}, Acc)$, where $Acc = Q_{acc}$ is Büchi acceptance condition, recognizes the limit language of $L_1 \cup L_2$. Let us prove that $A_{lim(L_1 \cup L_2)}$ recognizes a union of L_{ω_1} and L_{ω_2}. It is obvious that $L_{\omega_1} = lim(L_1) \subseteq lim(L_1 \cup L_2)$.

Now to prove the theorem we have to show that if a word α is accepted by $A_{lim(L_1 \cup L_2)}$, then α is in $L_{\omega_1} = lim(L_1)$ or in $L_{\omega_2} = lim(L_2)$. Let us examine word α. As it is accepted by GBA, we have a state $q_i \in Acc$ which occurs infinity often in α. If we consider a group automaton $A_{L_1 \cup L_2}$ for finite input on the prefixes of α which reaches the accepting state q_i, each of these prefixes belongs to L_1 or L_2, as there are infinitely many such accepted prefixes, then there will be infinitely many prefixes belonging to L_1 or L_2, from it follows that α belongs to L_1 or L_2. □

Theorem 6.5. *The class of languages recognized by QBA is not closed under intersection.*

Proof. Let us assume the contrary - it is closed under intersection, it means the class of languages accepted by GBA is also closed under intersection. Let us consider two GBA given in the figure 6.3 ($Acc = \{q_1\}$). One of GBA accepts the language $L_a = lim(w|odd\ number\ of\ 'a')$ other accepts the language $L_b = lim\{w|odd\ number\ of\ 'b')$. Assume there is a GBA recognizing $L_a \cap L_b$.

Now let us examine a Büchi group finite state automaton $A_{GBA} = (Q, \Sigma, \delta_1, q_0, Acc)$ recognizing the language $L_a \cap L_b$. Let us look at the set of states Q_a containing the states of A_{GBA} reachable from the initial state q_0 reading only the symbol a. As A_{GBA} is group finite state ω-automaton, there is a word a^k by reading which the automaton returns to the initial state. If $q_i \in Q_a$, then $q_i \notin Acc$ otherwise the word a^ω will be accepted, but it does not belong in L_b. Similarly for the symbol b, Q_b denotes the set of states containing the states of A_{GBA} reachable from the initial state q_0 reading only the symbol b. As A_{GBA} is group finite state ω-automaton, there is a word b^l by reading which the automaton returns

to the initial state. If $q_i \in Q_b$, then $q_i \notin Acc$ otherwise the word b^ω will be accepted, but it does not belong in L_1. However, the automaton A_{GBA} should accept the word $(a^k b^l)^\omega$, but it does not. Our assumption was wrong there is no GBA accepting the intersection of L_a and L_b. It means - the class of languages accepted by Büchi quantum finite state automata with bounded error is not closed under intersection, as Büchi quantum finite state automata recognize exactly the same language class as Büchi group finite state automata. □

Theorem 6.6. *The class of languages recognized by QBA is not closed under complementation.*

Proof. In the proof, we consider the same language L_a as in the proof of the above theorem. Assume there is a Büchi quantum finite state automaton accepting the complement of L_a, then there is also a Büchi group finite state automaton $A_{\overline{L_a}} = (Q, \Sigma, \delta_1, q_0, Acc)$ accepting the complement of the language L_a. Since $aab^\omega \in \overline{L_1}$ there is an accepting state $q_{i_1} \in Acc$ which is reachable by reading a word aab^{k_1} ($k_1 > 0$) from the initial state q_0. Since $aab^{k_1}aab^\omega \in \overline{L_1}$ there is accepting state $q_{i_2} \in Acc$ which is reachable by reading a word aab^{k_2} ($k_2 > 0$) from the state q_{i_1}. Using this argument, we obtain infinitely many $k_1, k_2, ...k_3 > 0$, such that $aab_1^k aab_2^k aab_3^k aab_4^k ...$ is accepted by the automaton. However, $aab_1^k aab_2^k aab_3^k aab_4^k ... \in L_a$. Thus, the class of languages accepted by Büchi quantum finite state automata is not closed under complement. □

6.4 Measure-Many Quantum Finite State Automata over Infinite Words

In this section, we extend a definition of a measure-many quantum finite state automaton and define a Büchi acceptance criteria for this kind of automata.

Definition 6.4.1. *A **measure-many quantum finite state over infinite words** is a quintuple* $A = (Q, \Sigma, \delta, q_0, Q_H, Acc)$

- Q *is a finite set of states,*
- Σ *is a finite input alphabet,*
- δ *is the transition function* $\delta : Q \times \Sigma \cup \{\sharp\} \times Q \to C_{[0,1]}$*, which represents the amplitudes that flows from the state* q *to the state* q' *after reading symbol* σ*, and* $\sharp$ *is the left end-marker, which denotes the start of the word,*
- $q_0 \in Q$ *is the initial state,*

- $Q_H \subseteq Q$ *are halting states of the automaton,*
- Acc *is the acceptance component.*

For all states $q_1, q_2, q' \in Q$ *and symbols* $\sigma \in \Sigma$*, the function* δ *must be unitary, thus the function satisfies the condition*

$$\sum_{q'} \overline{\delta(q_1, \sigma, q')}\delta(q_2, \sigma, q') = \begin{cases} 1 \ (q_1 = q_2) \\ 0 \ (q_1 \neq q_2) \end{cases} .$$

The linear superposition of the automaton's states is also represented by an n-dimensional complex unit vector, where $n = |Q|$ and the transition function δ is represented by a set of unitary matrices $\{V_\sigma\}_{\sigma\in\Gamma}$, where V_σ is the unitary transition of the automaton A_{MM-QFA} after reading the symbol σ and is defined by $V_\sigma(|\ q\rangle) = \sum_{q'\in Q} \delta(q, \sigma, q') \mid q'\rangle$.

A computation of the automaton on the input word $\sharp\sigma_1\sigma_2... \in \Sigma^\omega$ proceeds as follows:

- it starts in the superposition ρ_0;
- a unitary transition corresponding to the current input symbol is performed;
- after every transition, the automaton A measures its state with respect to the observable $\bigoplus_{i=1}^{|Q_h|} E_i \oplus E_{non}$ where $E_i = span\{|q_i\rangle\}$ for each $q_i \in Q_H$ and $E_{non} = span\{|q\rangle : q \notin Q_H\}$. If the observed state of the automaton is in E_i subspace, then the input halts, otherwise the computation continues.

After every measurement, the superposition collapses to the measured subspace. We keep track of halting probabilities for each halting state, therefore, the state of the automaton is represented by a tuple $(\phi, p_1, p_2, ..., p_{|Q_H|})$, where p_i is the cumulative probabilities of halting to the state $q_i \in Q_H$. The transition of the automaton on reading the symbol σ is denoted by $(P_{non}\,|\phi'\rangle\,, p_1 + \|P_1\phi'\|^2\,, p_2 + \|P_2\phi'\|^2\,, ..., p_{|Q_H|} + \left\|P_{|Q_H|}\phi'\right\|^2)$, where $\phi' = V_\sigma\phi$ and P_i is a diagonal zero-one projection matrix projecting onto E_i subspace.

Definition 6.4.2. *A Büchi acceptance condition for a measure-many quantum finite state automaton* Acc *is a subset* $F \subseteq Q_H$*, where the elements of* F *are called accepting states. The computation of an automaton on the input* $\sharp\sigma_1\sigma_2... \in \Sigma^\omega$ *is called accepting, if after reading an input the probability to be in states* $q_i \in F$ *is 1.*

Example 6.4.1. *Let us consider an example of a Büchi measure-many quantum finite state automaton recognizing the limit of the language containing the words of odd number of* a*. The automaton has:*

- *a set of states* $\{q_0, q_1, q_2\}$,
- *the halting set of states is* $Q_H = \{q_2\}$,
- *an input alphabet is* $\{a, b\}$,
- *the accepting component* $Acc = Q_H$,
- $V_\sharp = V_b = I_3$, $V_a = \begin{pmatrix} 0 & 1 & 0 \\ \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \end{pmatrix}$

If a word belongs to the language, then the state q_1 *has been visited infinite often it means that the probability to be in* Acc *is* $\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + ... = 1$, *if the word does not belong to the language then there is only finite number* k *of prefixes having odd number of* a. *In this case the probability to be in* Acc *is* $\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + ... + \frac{1}{2^k} = 1 - \frac{1}{2^k} < 1$.

Theorem 6.7. *The language class recognized by quantum Büchi automata is proper subset of the language class accepted by measure-many quantum finite state automata with Büchi acceptance criteria.*

Proof. We transform a group Büchi automaton to equivalent Büchi measure-many quantum finite state automaton. The group Büchi automaton $A_{GBA} = (Q, \Sigma, \delta, q_1, Acc)$ accepts language L, the equivalent Büchi measure-many quantum finite state automaton has:

- a set of states $Q = \{q_1, ..., q_{|Q|+|Acc|}\}$,
- an input alphabet Σ,
- q_0 as initial state,
- $Acc' = Q_H = \{q_{|Q|+1}, ..., q_{|Q|+|Acc|}\}$,
- a transition function is defined as follows:
  - $V_\sigma(|q_i\rangle) = \frac{1}{\sqrt{2}}|q_j\rangle + \frac{1}{\sqrt{2}}|q'_j\rangle$ ($q'_j \in Q_H$) if $\delta(q_i, \sigma) = q_j$ and $q_j \in Acc$,
  - $V_\sigma(|q_i\rangle) = |q_j\rangle$ if $\delta(q_i, \sigma) = q_j$ and $q_j \notin Acc$.

Each time when the group Büchi automaton visits an accepting run, Büchi measure-many quantum finite state automaton sends a half of the non-halting probability to the halting state. If a word is accepted by group Büchi automaton, then accepting states have been visited infinite often it means that the probability to be in Acc is $\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + ... = 1$, if

an input word is not recognized by the group Büchi automaton, then there is only finite number k of prefixes having odd number of a. In this case the probability to be in Acc is $\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + ... + \frac{1}{2^k} = 1 - \frac{1}{2^k} < 1$.

At the same time, the infinite language containing words starting with symbol a can be accepted by Büchi measure-many quantum finite state automaton, which has:

- a set of states $Q = \{q_1, q_2, q_3\}$,
- an input alphabet Σ,
- q_0 as initial state,
- $Q_H = \{q_2, q_3\}$,
- $Acc = \{q_2\}$
- a transition function for symbol a is defined as $V_a(|q_1\rangle) = |q_2\rangle$ and for the rest of symbols $V_{\neq a}(|q_1\rangle) = |q_3\rangle$.

This language cannot recognized by a quantum Büchi automaton as it does not belong to the limit languages of measure-once quantum finite state automata. □

Chapter 7

Conclusion

In the thesis, we investigated the connection between quantum finite state automata and logic. We investigated three different notations of quantum finite state automata - measure-once quantum finite state automata [38], measure-many quantum finite state automata [32], and Latvian quantum finite state automaton [1] and its connection to first order logic, modular logic, and logic using generalized quantifiers - Lindström quantifier and group quantifier. We have obtained the following:

- characterized the language class accepted by measure-once quantum finite state automata with bounded error in the terms of logic;
- proved that intersection of the language class accepted by measure-once quantum finite automata with bounded error and languages defined by $FO[<]$ contains only trivial languages, i.e., an empty language or Σ^*;
- proved that languages described by $MOD[<]$ are recognized by measure-once, measure-many and Latvian quantum finite state automata;
- studied the connection between languages accepted by measure-many quantum finite state automata and first order logic, as well as, the connection between acceptance probability of measure-many quantum finite state automata and first order logic was examined;
- studied the connection between acceptance probability of measure-many quantum finite state automata and modular logic using the first order quantifiers;
- studied the connection between Latvian quantum finite state automata and logic.

The further research in the connection between quantum finite state automata and logic requires to study new kinds of logic, the use of generalized quantifiers could help

to characterize the languages accepted by quantum finite state automata. The study of other notations of quantum finite state automata from logic point of view could help to get relationship between these notations of quantum finite state automata.

The second part of the thesis was devoted to the quantum finite state automata over infinite words. We studied the class of languages accepted by Büchi measure-once quantum finite state automata with bounded error and proved its closure under union, as well as, showed that the class of languages accepted by Büchi measure-once quantum finite state automata is not closed under two other standard set operations - intersection and complementation. Similarly to classical case also in the quantum case, the Rabin and Streett acceptance conditions are more powerful than Büchi acceptance condition. We have also defined a Büchi measure-many quantum finite state automaton, which is more powerful then quantum Büchi automaton. However, the started research is a small step in this research area. We examined the simplest model of the quantum finite state automata, there are other models such as already mentioned Latvian quantum finite state automata and others. We have not studied quantum finite state automata using other acceptance condition just showed that Rabin and Streett conditions are more powerful, so it is another potential research area, as well as, study of acceptance probabilities and other properties of automata.

Bibliography

[1] M. Golovkins A. Kikusts M. Mercer A. Ambainis, M. Beaudry and D. Therien. Algebraic results on quantum automata. In *STACS 2004*, volume 2996/2004 of *Lecture Notes in Computer Science*, pages 93–104. Springer, 2004. 1.1.2, 2.3.14, 2.2, 5.1, 5.1, 7

[2] Scott Aaronson. Quantum computing, postselection, and probabilistic polynomial-time. *Electronic Colloquium on Computational Complexity (ECCC)*, (003), 2005. 1.1.1

[3] David Z. Albert. On quantum-mechanical automata. *Physics Letters A*, 98:249–252, 1983. 1.1

[4] Andris Ambainis, Richard F. Bonner, Rusins Freivalds, and Arnolds Kikusts. Probabilities to accept languages by quantum finite automata. In *COCOON*, pages 174–183, 1999. 4.1

[5] Andris Ambainis and Rusins Freivalds. 1-way quantum finite automata: strengths, weaknesses and generalizations. In *FOCS '98: Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, page 332, Washington, DC, USA, 1998. IEEE Computer Society. 2.3.5

[6] Andris Ambainis and Arnolds Kikusts. Exact results for accepting probabilities of quantum automata. *Theor. Comput. Sci.*, 295:3–25, 2003. 3.2, 4.2

[7] Andris Ambainis, Arnolds Kikusts, and Maris Valdats. On the class of languages recognizable by 1-way quantum finite automata. In *STACS 2001, 18th Annual Symposium on Theoretical Aspects of Computer Science*, volume 2010 of *Lecture Notes in Computer Science*, pages 75–86. Springer, 2001. 1.1.2, 1.2

[8] Andris Ambainis, Ashwin Nayak, Ammon Ta-Shma, and Umesh Vazirani. Dense quantum coding and a lower bound for 1-way quantum automata. In *STOC '99:*

Proceedings of the thirty-first annual ACM symposium on Theory of computing, pages 376–383. ACM, 1999. 1.1.2

[9] A.Mostowski. On a generalization of quantifiers. *Fundamenta Mathenaticae*, 44:12–36, 1957. 2.5.3

[10] Christel Baier, Nathalie Bertrand, and Marcus Größer. On decision problems for probabilistic büchi automata. In *FoSSaCS*, volume 4962 of *Lecture Notes in Computer Science*, pages 287–301. Springer, 2008. 1.3

[11] Christel Baier, Nathalie Bertrand, and Marcus Größer. Probabilistic acceptors for languages over infinite words. In *SOFSEM*, volume 5404 of *Lecture Notes in Computer Science*, pages 19–33. Springer, 2009. 1.3

[12] Christel Baier and Marcus Größer. Recognizing omega-regular languages with probabilistic automata. In *LICS*, pages 137–146. IEEE Computer Society, 2005. 1.3

[13] Paul A. Benioff. Quantum mechanical hamiltonian models of discrete processes that erase their own histories: Application to turing machines. *International Journal of Theoretical Physics*, 21(3/4):177–202, 1982. 1.1

[14] Ethan Bernstein and Umesh Vazirani. Quantum complexity theory. *SIAM Journal on Computing*, 26:11–20, 1997. 1.1, 1.1.1

[15] Alberto Bertoni, Carlo Mereghetti, and Beatrice Palano. Quantum computing: 1-way quantum automata. In *Developments in Language Theory*, pages 1–20, 2003. 1.1.2

[16] Alex Brodsky and Nicholas Pippenger. Characterizations of 1-way quantum finite automata. *SICOMP: SIAM Journal on Computing*, 31, 2002. 1.1.2, 1.1, 1.1.2, 2.1, 4.1

[17] Richard J. Büchi. Weak second-order arithmetic and finite automata. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 6(1-6):66–92, 1960. 1, 1.2, 1.3

[18] Büchi, Richard J. On a Decision Method in Restricted Second Order Arithmetic. In *Proceedings of the 1960 International Congress of Logic, Methodology and Philosophy of Science*, pages 1–12. Stanford University Press, 1962. 1.2

[19] A. Chi-Chih Yao. Quantum circuit complexity. In *SFCS '93: Proceedings of the 1993 IEEE 34th Annual Foundations of Computer Science*, pages 352–361. IEEE Computer Society, 1993. 1.1.1

[20] David Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer. *Royal Society of London Proceedings Series A*, 400:97–117, 1985. 1.1, 1.1.1

[21] David Deutsch and Richard Jozsa. Rapid solutions of problems by quantum computation. *Proceedings of the Royal Society of London*, A439:553–55, 1992. 1.1

[22] C.C. Elgot. Decision problems of finite automata design and related arithmetic. *Transactions of the American Mathematical Society*, 98:21–52, 1961. 1, 1.2

[23] Ronald Fagin. Generalized first-order spectra and polynomial-time recognizable sets. In *Complexity and Computation*, volume 7, pages 43–73. SIAM-AMS Proceedings, 1974. 1, 1.2

[24] Richard P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6/7):467–488, 1982. 1.1

[25] Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research*, volume 2500 of *Lecture Notes in Computer Science*. Springer, 2002. 2, 2.6.1

[26] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of The Twenty-Eighth Annual ACM Symposium On The Theory Of Computing (STOC '96)*, pages 212–219. ACM Press, 1996. 1.1

[27] Jozef Gruska. *Quantum computing*. McGraw-Hill, 1999. 1.1, 2

[28] Mika Hirvensalo. *Quantum Computing*. Springer, 2004. 1.1, 2

[29] Neil Immerman. Relational queries computable in polynomial time. *Information and Control*, 68(1–3):86–104, 1986. 1.2

[30] Neil Immerman. Languages that capture complexity classes. *SIAM Journal on Computing*, 16(4):760–778, 1987. 1.2

[31] Satoshi Iriyama, Masanori Ohya, and Igor Volovich. Generalized quantum turing machine and its application to the sat chaos algorithm, 2004. 1.1.1

[32] Attila Kondacs and John Watrous. On the power of quantum finite state automata. In *Proceedings of the 38th IEEE Conference on Foundations of Computer Science*, pages 66–75, 1997. 1.1.2, 2.3.13, 4.1, 7

[33] Lvzhou Li, Daowen Qiu, Xiangfu Zou, Lvjun Li, and Lihua Wu. Characterizations of one-way general quantum finite automata, 2009. 1.1.2

[34] Seth Lloyd. *Programming the Universe*. Alfred A. Knopf, 2006.

[35] Robert McNaughton. Testing and generating infinite sequences by a finite automaton. *Information and Control*, 9:521–530, 1966. 1.2, 1.3

[36] Carlo Mereghetti and Beatrice Palano. Quantum finite automata with control language. *ITA*, 40(2):315–332, 2006. 1.1.2

[37] David A. Mix Barrington, Neil Immerman, and Howard Straubing. On uniformity within nc1. *Journal of Computer and System Sciences*, 41(3):274–306, 1990. 2.5.3, 3.3

[38] Cristopher Moore and James P. Crutchfield. Quantum automata and quantum grammars. *Theoretical Computer Science*, 237:275–306, 2000. 1.1.2, 2.3.12, 6.1, 7

[39] Gordon E. Moore. Cramming more components onto integrated circuits. *Electronics*, 38, 1965. 1.1

[40] D.E. Muller. Infinite sequences an finite machines. In *4th IEEE Symp. on Switching Circuit Theory and Logical Design*, pages 3–16, 1963. 1.3

[41] Michael A. Nielsen and Isaac L. Chuang. *Quantum computation and quantum information*. Cambridge University Press, 2000. 1.1, 2

[42] K. Paschen. Quantum finite automata using ancilla qubits. 1.1.2

[43] Simon Perdrix and Philippe Jorrand. Classically-controlled quantum computation. *Mathematical Structures in Computer Science*, 16:601, 2006. 1.1.1

[44] Jean-Eric Pin. Bg = pg: A success story, 1995. 2.4

[45] Amir Pnueli. The temporal logic of programs. In *Proceedings of the 18th IEEE Symposium on the Foundations of Computer Science (FOCS-77)*, pages 46–57. IEEE Computer Society Press, 1977. 1.2

[46] Daowen Qiu, Paulo Mateus, Xiangfu Zou, and Amilcar Sernadas. One-way quantum finite automata together with classical states: Equivalence and minimization, 2009. 1.1.2

[47] Michael O. Rabin. Decidability of second-order theories and automata on infinite trees. *Transactions of the American Mathematical Society*, 141:1–35, 1969. 1.2, 1.3

[48] Marcel Paul Schützenberger. On finite monoids having only trivial subgroups. *Information and Control*, 8(2):190–194, 1965. 2.5.1

[49] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997. 1.1

[50] Daniel R. Simon. On the power of quantum cryptography. In *FOCS*, pages 116–123, 1994. 1.1

[51] Howard Straubing, Denis Thérien, and Wolfgang Thomas. Regular languages defined with generalized quantifiers. In *ICALP*, volume 317 of *Lecture Notes in Computer Science*, pages 561–575. Springer, 1988. 2.5.2, 2.6, 3.2

[52] Gabriel Thierrin. Permutation automata. *Mathematical Systems Theory*, 2(1):83–90, 1968. 1.1.2

[53] Wolfgang Thomas. Languages, automata, and logic. In *Handbook of Formal Languages*, pages 389–455. Springer, 1996. 2, 2.6.1

[54] Alan M. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proc. London Math. Soc.*, 2(42):230–265, 1936. 1.1.1

[55] Moshe Y. Vardi. The complexity of relational query languages. In *Proceedings of the 14th ACM Symposium on Theory of Computing (STOC)*, pages 137–146. ACM Press, 1982. 1.2