UNIVERSITY OF LATVIA

OKSANA ŠČEGUĻNAJA-DUBROVSKA

MODELS OF QUANTUM COMPUTATION

PHD THESIS

Nozare: datorzinātnes

Apakšnozare: datorzinātņu matemātiskie pamati

Promocijas darba vadītājs:

prof. , Dr.habil.math. RŪSIŅŠ MĀRTIŅŠ FREIVALDS

Rīga – 2010

1 ANOTĀCIJA

Kvantu modelis ar pēcatlasi tiek definēts Scott Aaronson darbā. Beigu stāvokļu kopa, kas parasti sastāv no akceptējošiem un noraidošiem stāvokļiem, tiek papildināta ar parametru, kas norāda, vai dotais beigu stāvoklis ietilpst atlases kopā. Mērījumi tiek veikti tikai atlases kopas beigu stāvokļos. Tiek ieviests papildus stāvoklis q_+, un ja visu pēcatlases stāvokļu amplitūdas ir 0, tad q_+ amplitūda saņem vērtību 1.

Pēcatlase ļauj pētīt ne tikai kvantu, bet arī tradicionālo algoritmu īpašības.

Pētījuma mērķis ir salīdzināt varbūtisko un kvantu galīgo pēcatlases automātu klases un aprakstīt valodu klases, ko atpazīst kvantu galīgs automāts ar pēcatlasi.

Pētījuma procesā iegūti šādi rezultāti:

- Definēts kvantu galīgā automāta ar pēcatlasi jēdziens;
- Aprakstīta valoda PALINDROMES, ko atpazīst galīgs kvantu automāts ar pēcatlasi ar mērījumu katrā solī un galīgs kvantu automāts ar pēcatlasi ar mērījumu beigās;
- Aprakstīta valoda, kuru nevar atpazīt galīgs kvantu automāts ar pēcatlasi ar mērījumu katrā solī un galīgs kvantu automāts ar pēcatlasi ar mērījumu beigās: $L = \{w \mid w \in \{0,1\}^*$ and there exist x, y, u, z such that $w = x1y = u1z$ and $|x| = |z|\}$

Viens no promocijas darba uzdevumiem ir aplūkot kvantu vaicājošos algoritmus Bula funkciju rēķināšanai. Darba sākumā tiek pierādīti kvantu algoritmu apakšējie novērtējumi dažādām funkcijām, kas apraksta grafu problēmas. Ir izveidoti efektīvi kvantu vaicājošie algoritmi. Šajā sadaļā iegūti rezultāti sekojošām funkcijām:

- 3-sum problēma,
- Hamiltona ceļš,
- Hamiltona aplis,
- Ceļojošais pārdevējs.

Vēl promocijas darbā tiek apskatīta reāla laika Tjūringa mašīnas kvantu analoģija. Tiek parādīts, ka eksistē valoda, kuru pazīst reāla laika kvantu Tjūringa mašīna un nepazīst reāla laika determinēta Tjūringa mašīna.

2 ABSTRACT

Postselection quantum model is defined by Scott Aaronson. A new parameter is added to a halting set of states, that consists of accepting and rejecting states, which defines if the state is in postselection set. Only states in postselection set are measured. New state q_+ is added and if all postselection states amplitudes are equal to 0, then q_+ amplitude is set to 1.

Postelection appears to be very useful to study not only quantum, but also traditional algorithms .

Paper goal is to compare probabilistic and quantum finite automata with postselection and define language class, that can be recognized by quantum finite automata with postselection.

The following results are obtained:

- The notion of quantum finite automata with postselection is given;
- Language PALINDROMES is defined, that can be recognized by MO- and MM- quantum finite automata with postselection;
- Language is defined, that cannot be recognized by MO- and MM- quantum finite automata with postselection: $L = \{w \mid w \in \{0,1\}^*$ and there exist x, y, u, z such that $w = x1y = u1z$ and $|x| = |z|\}$

One of the research object of this work is find quantum query algorithms to compute Boolean functions. At first we prove higher lower bounds of quantum query algorithms for some of graph problems. Effective quantum query algorithms are created with complexity lower than deterministic one. Results for the following functions are obtained:

- 3-sum problem,
- Hamiltonian path,
- Hamiltonian circuit,
- Travelling salesman.

Another aim of this paper is to introduce a quantum counterpart for real – time Turing machine. The recognition of a special kind of language, that can’t be recognized by a deterministic real – time Turing machine, is shown.

CONTENTS

Contents

3 Introduction

The quantum computation theory started to evolve recently, and it has become a fact that the quantum mechanism gives us a certain kind of power, which cannot be achieved by the deterministic or probabilistic approach.

The possibility of performing computations in a quantum system was first explicitly pointed out independently by Paul Benioff [1] and Richard Feynman [2] in 1982. While Benioff work was more concerned with the trend of the minimization that exists in the area of the computing devices, such that their size can eventually approach the border where quantum effects can be experienced, Feynman motivation was quite different and more interesting for us.

He supposed that it might require exponential time to simulate quantum mechanical processes on classical computers. This served as a basis to the opinion that quantum computers might have the advantages versus the classical ones.

The first impressive result obtained in 1994, that made quantum computing theory the area of great interest, was Peter Shor quantum algorithm for factorization of integers in polynomial time [3] while all known classical algorithms are exponential.

Any quantum computation is performed by means of unitary operators. One of the simplest properties of these operators shows that such a computation is reversible. The result always determines the input uniquely.

While researching quantum computing many counterparts of the basic conceptions of the classical computation theory (such as Turing machine, finite automata etc.) have been defined and researched (see [4] page 150).

First part of this work presents postselection quantum finite automata. Postselection for quantum computing devices was introduced by S.Aaronson [5] as an excitingly efficient tool to solve long standing problems of computational complexity related to classical computing devices only. This was a surprising usage of notions of quantum computation.

We introduce MO- and MM - postselection quantum finite automata notion and prove that PALINDROMES can be recognized by MM-quantum finite automata with postselection. This result distinguish quantum automata with postselection from probabilistic automata with non-isolated cut-point 0 because probabilistic finite automata with non-isolated cut-point 0 can recognize only regular languages but PALINDROMES is not a regular language.

Proofs of nonrecognizability of languages by postselection quantum automata are much more difficult than similar proofs for probabilistic finite automata with non-isolated cut-points. This is because the latter proofs are always based on property Turakainen [6] named linearity of probabilistic finite automata. However, the work of quantum automata (including MM-quantum finite automata with postselection) involves measurement which makes the automata nonlinear. "Naive" techniques are capable to prove nonrecognizability of languages by postselection quantum automata only in cases when the languages are far from recognizable by these automata, e.g. when they demand exponential time on Turing machines. We have introduced a novel technique for nonrecognizability proofs. This technique allowed us to prove nonrecognizability of a context-free language by MM-quantum finite automata with postselection.

Second part of the paper presents quantum query algorithms to compute Boolean functions and their lower bounds. We consider graph problems, such as 3-sum, Hamiltonian path, Hamiltonian circuit, Travelling salesman. We know that sometimes it is possible to create quantum algorithms, that are more efficient than their classical (deterministic and probabilistic) couterparts. For example, Grover [7] discovered a remarkable *quantum* algorithm that, making queries in superposition, can be used to compute OR function with small error probability using only $O(\sqrt{N})$ queries, while both deterministic and probabilistic algorithms require $O(N)$ queries. On the other hand, quantum algorithms are in a sense more restricted. For instance, only unitary transformations are allowed for state transitions. Hence rather often a problem arises whether or not the needed quantum automaton exists. In such a situation lower bounds of complexity are considered. It is proved in [8] that Grover database search algorithm is the best possible. We use a result by A.Ambainis [9] to prove lower complexity bounds for quantum query algorithms. Currently, this is the most powerful method to prove lower bounds of complexity for quantum query algorithms. In some cases there still remains a gap between the upper and the lower bounds of the complexity. In these cases it is proved additionally that Ambainis' method cannot provide a better lower bound for this problem.

The third part of this paper introduces a quantum counterpart for real – time Turing machine of the classical computation theory, namely real – time quantum Turing machine, and compares its capabilities with deterministic case.

It was in 1985, when D. Deutsch introduced the notion of quantum Turing machine [10] and proved that quantum Turing machines compute the same recursive functions as classical deterministic Turing machines do. But is it possible to find quantum Turing machine advantages over deterministic that have some limitations, such as, for example, time, tape space or the quantity of the head turns?

Such subclasses, that are weaker that general Turing machine, are defined and studied for deterministic Turing machines. They give us a computational model that is more realistic.

The limitation is, that if the input word consists of *n* symbols, than the computation is to be performed for *n* steps, one step for one input symbol. Such computation is called *real – time computation*.

There are some results about real – time computation in deterministic case [11,12,13], mainly about real – time computation possibility.

For quantum counterpart, we show, that there is a language that is accepted by a quantum real – time Turing machine, but can’t be accepted by a deterministic real – time Turing machine.

4 Preliminaries

The model of the quantum computing will be described here to introduce the notation used further. To get more information on the specific topic please refer to Josef Gruska [4].

The indivisible unit of classical information is the bit that can take any of two values: *true* or *false*. The probabilistic counterpart of the classical bit can be *true* with the probability α and *false* with probability β, where $\alpha + \beta = 1$. The corresponding unit of quantum information is the quantum bit or *qubit*. For a qubit the possibility to be *true* or *false* is stated as $|\alpha|^2 + |\beta|^2 = 1$, where α and β are the arbitrary complex numbers. If we observe qubit, we get *true* with probability $|\alpha|^2$ and *false* with probability $|\beta|^2$. However, if we modify a quantum system without observing it (this will be explained further), the set of transformations that one can perform is larger than in the probabilistic case.

We consider quantum systems with m basis states $|q_1>$, $| q_2>, \ldots, | q_m>$. Let ψ be a linear combination of them with complex coefficients

$$\psi = \alpha_1|q_1> + \alpha_2|q_2> + \ldots + \alpha_m|q_m>.$$

The norm of ψ is

$$\|\psi\| = \sqrt{|\alpha_1|^2 + |\alpha_2|^2 + \ldots + |\alpha_n|^2}$$

The state of quantum system can be any ψ with $\|\psi\| = 1$. ψ is called a *superposition* of $|q_1>, | q_2>, \ldots, | q_m>$. $\alpha_1, \alpha_2, \ldots, \alpha_m$ are called *amplitudes* of $|q_1>, | q_2>, \ldots, | q_m>$.

There are two types of transformations that can be performed on a quantum system. The first types are *unitary transformations*. A unitary transformation is a linear transformation U that preserves norm (any ψ with $\|\psi\| = 1$ is mapped to ψ' with $\|\psi'\| = 1$).

Second, there are *measurements*. The simplest measurement is observing $\psi = \alpha_1|q_1> + \alpha_2|q_2> + \ldots + \alpha_m|q_m>$ in the basis $|q_1>, |q_2>, \ldots, |q_m>$. It gives $|q_i>$ with probability α_i^2. After the measurement, the state of the system changes to $|q_i>$ and repeating the measurement gives the same state $|q_i>$.

5 Postselection finite quantum automata

Scott Aaronson [5] introduced an interesting notion of postselection for quantum computing devices. It is clear from the very beginning that they can never be implemented because they contradict laws of Quantum Mechanics.

However, this notion appears to be extremely useful to prove properties of existing types of algorithms and machines.

The definition of postselection by S.Aaronson [5] cannot be used for finite automata directly because his construction needs unlimited ammount of memory.

Definition 1.

A postselection quantum finite automaton is a quantum finite automaton (MO- or MM-quantum automaton) with a set of states called postselection set of states *and a special state* q_+*. At the very end of the work of the automaton when the end-marker is already read but before the measurement of accepting and rejecting states the amplitudes of all the states outside the postselection set are mechanically made to equal zero.*

If at least one of the postselection states is not equal zero, then the amplitudes of all the postselection statesare normalized, i.e. multiplied to a positive real number such that in the result of this normalization the total of squares of the modulos of the amplitudes of the postselection states equals 1. If at the moment of postselection all the amplitudes of the postselection states are equal to zero, then these amplitudes stay equal to zero but the state q_+ *gets amplitude 1. This way, at the result of the postselection the total of squares of the modulos of the amplitudes of all the states equals 1.*

Postselection is the power of discarding all runs of a computation in which a given event does not occur. To illustrate, suppose we are given a Boolean formula in a large number of variables, and we wish to find a setting of the variables that makes the formula true. Provided such a setting exists, this problem is easy to solve using postselection: we simply set the variables randomly, then postselect on the formula being true.

We study the power of postselection in a quantum computing context. S. Aaronson [5] defines a new complexity class called *PostBQP* (postselected bounded-error quantum polynomial-time), which consists of all problems solvable by a quantum computer in polynomial time, given the ability to postselect on a measurement yielding a specific outcome. The main result is that *PostBQP* equals the well-known classical complexity class *PP* (probabilistic polynomial-time). Here *PP* is the class of problems for which there exists a probabilistic polynomial-time Turing machine that accepts with probability greater than 1/2 if and only if the answer is *yes*. For example, given a Boolean formula, a *PP* machine can decide whether the majority of settings to the variables make the formula true. Indeed, this problem turns out to be *PP* -complete (that is, among the hardest problems in *PP*).

S. Aaronson himself describes his aim as follows:

"The motivation for the *PostBQP=PP* result comes from two quite different sources. The original motivation was to analyse the computational power of *fantasy* versions of quantum mechanics, and thereby gain insight into why quantum mechanics is the way it is. In particular, 4 will show that if we changed the measurement probability rule from $|\psi|^2$ to $|\psi|^p$ for some $p \neq 2$, or allowed linear but non-unitary evolution, then we could simulate postselection, and thereby solve *PP*-complete problems in polynomial time. If we consider such an ability extravagant, then we might take these results as helping to explain why quantum mechanics is unitary, and why the measurement rule is $|\psi|^2$.

A related motivation comes from an idea that might be called *anthropic computing* - arranging things so that we are more likely to exist if a computer produces a desired output than if it does not. As a simple example, under the many-worlds interpretation of quantum mechanics, we might kill ourselves in all universes where a computer fails! My result implies that, using this *technique*, we could solve not only *NP*-complete problems efficiently, but *PP*-complete problems as well.

However, the *PostBQP=PP* result also has a more unexpected implication. One reason to study quantum computing is to gain a new, more general perspective on classical computer science. By analogy, many famous results in computer science involve only deterministic computation, yet it is hard to imagine how anyone could have proved these results had researchers not long ago *taken aboard* the notion of randomness.

Likewise, taking quantum mechanics aboard has already led to some new results about classical computation [18,**Error! Reference source not found.**,**Error! Reference source not found.**].

What this paper will show is that, even when classical results are already known, quantum computing can sometimes provide new and simpler proofs for them."

5.1 Definitions

A quantum finite automaton (QFA) is a theoretical model for a quantum computer with a finite memory. If we compare them with their classical (non-quantum) counterparts, QFAs have both strengths and weaknesses. The strength of QFAs is shown by the fact that quantum automata can be exponentially more space efficient than deterministic or probabilistic automata [14]. The weakness of QFAs is caused by the fact that any quantum process has to be reversible (unitary). This makes quantum automata unable to recognize some regular languages.

We start by reviewing the concept of probabilistic finite state transducer.For a finite set *X* we denote by *X** the set of all finite strings formed from *X*, the empty string is denoted ε.

Definition 2.

A probabilistic finite state transducer (pfst) is a tuple

$$T = (Q, \Sigma_1, \Sigma_2, V, f, q_0, Q_{acc}, Q_{rej})$$

where Q is a finite set of states, Σ_1, Σ_2 is the input/ output alphabet, $q_0 \in Q$ is the initial state, and $Q_{acc}, Q_{rej} \in Q$ are (disjoint) sets of accepting and rejecting states, respectively. (The other states, forming set Q, are called non--halting). The transition function $V : \Sigma_1 \times Q \to Q$ is such that for all $a \in \Sigma_1$ the matrix $(V_a)_{qp}$ is stochastic, and $f_a : Q \to \Sigma_2^*$ is the output function. If all matrix entries are either 0 or 1 the machine is called a deterministic finite state transducer (dfst).

The meaning of this definition is that, being in state q, and reading input symbol a, the transducer prints $f_a(q)$ on the output tape, and changes to state p with probability $(V_a)_{qp}$, moving input and output head to the right. After each such step,

if the machine is found in a halting state, the computation stops, accepting or rejecting the input.

Definition 3.

Let $R \subset \Sigma_1^* \times \Sigma_2^*$.

For $\alpha > 1/2$ *we say that* T *computes the relation* R *with probability* α *if for all* v, *whenever* $(v,w) \in R$, *then* $\mathrm{T(w \mid v)} \geq \alpha$, *and whenever* $(v,w) \notin R$, *then* $\mathrm{T(w \mid v)} \leq 1-\alpha$.

For $0 < \alpha < 1$ *we say that* T *computes the relation* R *with isolated cutpoint* α *if there exists* $\varepsilon > 0$ *such that for all* v, *whenever* $(v,w) \in R$, *then* $\mathrm{T(w \mid v)} \geq \alpha + \varepsilon$, *but whenever* $(v,w) \notin R$, *then* $\mathrm{T(w \mid v)} \leq \alpha - \varepsilon$.

The following definition is modelled after the ones for pfst for quantum finite state automata [15]:

Definition 4.

A quantum finite state transducer (qfst) is a tuple

$$T = (Q, \Sigma_1, \Sigma_2, V, f, q_0, Q_{acc}, Q_{rej}),$$

where Q *is a finite set of states,* Σ_1, Σ_2 *is the input/output alphabet,* $q_0 \in Q$ *is the initial state, and* $Q_{acc}, Q_{rej} \in Q$ *are (disjoint) sets of accepting and rejecting states, respectively. The transition function* $V : \Sigma_1 \times Q \rightarrow Q$ *is such that for all* $a \in \Sigma_1$ *the matrix* $(V_a)_{qp}$ *is unitary, and* $f_a : Q \rightarrow \Sigma_2^*$ *is the output function.*

Probabilistic and quantum finite automata are special cases of the transducers where the result can be only 0 or 1. Nothing needs to be added for the definition of probabilistic automata. However, the case of quantum automata is much more complicated.

5.2 Specifics of quantum finite automata

Quantum finite automata (QFA) were introduced independently by Moore and Crutchfield [16] and Kondacs and Watrous [15]. They differ in a seemingly small detail. The first definition allows the measurement only at the very end of the computation process. Hence the computation is performed on the quantum information only. The second definition allows the measurement at every step of the computation. In the process of the measurement the quantum information (or rather, a part of it) is transformed into the classical information. The classical information is not processed in the subsequent steps of the computation. However, we add the classical probabilities obtained during these many measurements. We will see below that this leads to unusual properties of the quantum automata and the languages recognized by these automata.

To distinguish these quantum automata, we call them, correspondingly, MO-QFA (measure-once) and MM-QFA (measure-many).

Definition 5.

An MM-QFA is a tuple

$$M = (Q, \Sigma, V, q_0, Q_{acc}, Q_{rej})$$

where Q is a finite set of states, Σ is an input alphabet, V is a transition function, $q_0 \in Q$ is a starting state, and $Q_{acc}, Q_{rej} \in Q$ are sets of accepting and rejecting states ($Q_{acc} \cap Q_{rej} = 0$). The states in Q_{acc} and Q_{rej} are called halting states and the states in $Q_{non} = Q - (Q_{acc} \cup Q_{rej})$ are called non halting states. κ and \$ are symbols that do not belong to Σ. We use κ and \$ as the left and the right endmarker, respectively. The working alphabet of M is $\Gamma = \Sigma \cup \{\kappa; \$\}$.

The state of M can be any superposition of states in Q (i. e., any linear combination of them with complex coefficients). We use $|q\rangle$ to denote the superposition consisting of state q only. $l_2(Q)$ denotes the linear space consisting of all superpositions, with l_2-distance on this linear space.

The transition function V is a mapping from $\Gamma \times l_2(Q)$ to $l_2(Q)$ such that, for every $a \in \Gamma$, the function $V_a : l_2(Q) \to l_2(Q)$ defined by $V_a(x) = V(a, x)$ is a unitary transformation (a linear transformation on $l_2(Q)$ that preserves l_2 norm).

The computation of a MM-QFA starts in the superposition $|q_0\rangle$. Then transformations corresponding to the left endmarker κ, the letters of the input word x and the right endmarker \$ are applied. The transformation corresponding to $a \in \Gamma$ consists of two steps.

1. First, V_a is applied. The new superposition ψ' is $V_a(\psi)$ where ψ is the superposition before this step.

2. Then, ψ' is observed with respect to $E_{acc}, E_{rej}, E_{non}$ where

$E_{acc} = span\{|q\rangle : q \in Q_{acc}\}$,

$E_{rej} = span\{|q\rangle : q \in Q_{rej}\}$,

$E_{non} = span\{|q\rangle : q \in Q_{non}\}$.

It means that if the system's state before the measurement was

$$\psi' = \sum_{q_i \in Q_{acc}} \alpha_i |q_i\rangle + \sum_{q_j \in Q_{rej}} \beta_j |q_j\rangle + \sum_{q_k \in Q_{non}} \gamma_k |q_k\rangle$$

then the measurement accepts ψ' with probability $\Sigma\alpha_i^2$, rejects with probability $\Sigma\beta_j^2$ and continues the computation (applies transformations corresponding to next letters) with probability $\Sigma\gamma_k^2$ with the system having state $\psi = \sum \gamma_k |q_k\rangle$.

We regard these two transformations as reading a letter a. We use V'_a to denote the transformation consisting of V_a followed by projection to E_{non}. This is the transformation mapping ψ to the non-halting part of $V_a(\psi)$. We use V'_w to denote the product of transformations $V'_w = V'_{a_n} V'_{a_{n-1}} ... V'_{a_2} V'_{a_1}$, where a_i is the i-th letter of the word w. We also use ψ_y to denote the non-halting part of QFA's state after reading the left endmarker κ and the word $y \in \Sigma *$. From the notation it follows that $\psi_w = V'_{\kappa w}(|q_0\rangle)$.

We will say that an automaton recognizes a language L with probability p ($p > 1/2$) if it accepts any word $x \in L$ with probability $> p$ and accepts any word $x \notin L$ with probability $\leq p$.

The MO-QFA differ from MM-QFA only in the additional requirement demanding that non-zero amplitudes can be obtained by the accepting and rejecting states no earlier than on reading the end-marker of the input word.

A probability distribution $\{(p_i, \phi_i) \mid 1 \leq i \leq k\}$ on pure states $\{\phi_i\}_{i=1}$ with probabilities $0 \leq p_i \leq 1 \left(\Sigma_{i=1}^{k}(p_i) = 1\right)$, is called a <u>*mixed state*</u> or <u>*mixture*</u>.

Definition 6.
A quantum finite automaton with mixed states is a tuple

$$(Q, \Sigma, \phi_{init}, \{T_\delta\}, Q_a, Q_r, Q_{non}),$$

where Q is finite a set of states, Σ is an input alphabet, ϕ_{init} is an initial mixed state, $\{T_\delta\}$ is a set of quantum transformations, which consists of defined sequence of measurements and unitary transformations, $Q_a \subseteq Q$, $Q_r \subseteq Q$ and $Q_{non} \subseteq Q$ are sets of accepting, rejecting and non-halting states.

Comment 1.
For quantum finite automata the term rejection is misleading. One can imagine that if an input word is accepted with a probability p then this word is rejected with probability $1-p$. Instead the reader should imagine that the only possible result of our automata is acception. The counterpart of our notion in recursive function theory is recursive enumerability but not recursivity. For probabilistic automata all the results by M.O.Rabin [11] are valid for both possible definitions but for quantum automata the difference is striking.

Sometimes even MO-QFA can be size-efficient compared with the classical FA.

Theorem 1.
[14]

1. For every prime p the language $L_p =\{$ the length of the input word is a multiple of p $\}$ can be recognized by a MO-QFA with no more than $const \log p$ states.

2. For every p a deterministic FA recognizing L_p needs at least p states.

3. For every p a probabilistic FA with a bounded error recognizing L_p needs at least p states.

The first results on MM-quantum finite automata were obtained by Kondacs and Watrous [15]. They showed that the class of languages recognized by QFAs is a proper subset of regular languages.

Theorem 2. [15]
1. All languages recognized by 1-way MM-QFAs are regular.

2. There is a regular language that cannot be recognized by a 1-way MM-QFA with probability $\frac{1}{2}+\varepsilon$ for any $\varepsilon > 0$.

Brodsky and Pippenger [17] generalized the second part of Theorem 2 by showing that any language satisfying a certain property is not recognizable by an MM-QFA.

Theorem 3. [17]
Let L be a language and M be its minimal automaton (the smallest DFA recognizing L). Assume that there is a word x such that M contains states q_1, q_2 satisfying:

1. $q_1 \neq q_2$,

2. If M starts in the state q_1 and reads x, it passes to q_2,

3. If M starts in the state q_2 and reads x, it passes to q_2, and

4. There is a word y such that if M starts in q_2 and reads y, it passes to q_1,

then L cannot be recognized by any 1-way quantum finite automaton (Fig.1).

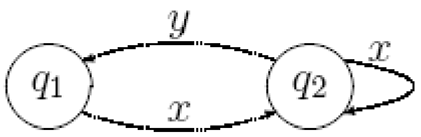

Fig. 1.

Theorem 4.
[18] The class of languages recognizable by a MM-QFA is not closed under union.

Corollary 1.
[18] The class of languages recognizable by a MM-QFA is not closed under any binary boolean operation where both arguments are significant. Another direction of research is studying the accepting probabilities of QFAs.

Theorem 5.

[14] The language $a_* b_*$ is recognizable by an MM-QFA with probability 0.68, but not

with probability 7/9 + ε for any ε > 0.

This shows that the classes of languages recognizable with different probabilities are different. Next results in this direction were obtained by [19] where the probabilities with which the languages $a_1^* ... a_n^*$ can be recognized are studied.

There is also a lot of results about the number of states needed for QFA to recognize different languages. In some cases, it can be exponentially less than for deterministic or even for probabilistic automata [14]. In other cases, it can be exponentially bigger than for deterministic automata [20].

Summarizing these results we can see that in spite of seeming naturality of the notion of MM-quantum finite automata with isolated cut-point this class of recognizable languages has rather specifical properties.

On the other hand, there have been many results on probabilistic and quantum algorithms working with non-isolated cut-point and on relations between recognition of languages with isolated and non-isolated cut-point[21]. However, it needs to be added that most of these papers when describing quantum automata restrict themselves to MO-quantum automata. MM-quantum automata are the most popular ones among the papers studying recognition with isolated cut-point, and MO-quantum automata are the most popular ones among the papers studying recognition with non-isolated cut-point.

5.3 Co-PALINDROMES can be recognized by postselection finite quantum automata

There exist nonregular languages recognizable by probabilistic finite automata with non-isolated cut-point (they are called *stochastic* laguages) and languages recognizable by quantum finite automata with non-isolated cut-point. Since MO-quantum finite automata differ from MM-quantum finite automata, it is possible that these classes are different as well. However, most natural problems on these automata are still open. We concentrate here on a very special subclass of these languages, namely, on classes of languages recognizable with cut-point 0.

In the case of probabilistic recognition this is not an interesting notion because in this case the input word is declared *accepted* if the probability of acception exceeds 0, and it is declared *rejected* if the probability of acception equals 0. It is obvious that such automata are equivalent to nondeterministic automata but nondeterministic finite automata recognize the same regular languages as deterministic automata do.

The case of quantum finite automata is different. We consider the language PALINDROMES, i.e. the language

$$PALINDROMES = \{x \mid x \in \{0,1\}^* \text{ and } x = x^{rev}\}$$

The first unexpected result on languages recognizable by MM-quantum postselection finite automata with probability 1 was the following theorem.

Theorem 6.
The complement of PALINDROMES is recognizable by 1-way MM-quantum finite automata with non-isolated cut-point 0.

Sketch of proof. We denote a real number

0.000 ... 0x(1)x(2)x(3) ... x(n)

n zeros

by

0.0{n}0x(1)x(2)x(3) ... x(n).

The main idea is to have at every moment of the processing the input word *0{n}x(1)x(2)x(3) ... x(n)* 2 special states of the automaton (say, q_2 and q_3) the amplitudes of which are, respectively, 0.0{n}0x(1)x(2)x(3) ... x(n) and
0.0{n}x(n) ... x(3)x(2)x(1).
We have

$$0.0\{n+1\}x(1)x(2)x(3)...x(n+1) =$$
$$= (0.0\{n+1\}x(1)x(2)x(3)...x(n)) \times \frac{1}{2} + \varepsilon_{n+1}$$

where

$$\varepsilon_{n+1} = \begin{cases} \left(\frac{1}{2}\right)^{2n+2} & ,if \quad x(n+1) = 1 \\ 0 & ,if \quad x(n+1) = 0 \end{cases}$$

We have also

$$0.0\{n+1\}x(n+1) ... x(3)x(2)x(1)=$$
$$= (0.0\{n+1\}x(n+1)...x(3)x(2)x(1)) \times \frac{1}{4} + \delta_{n+1}$$

Where

$$\delta_{n+1} = \begin{cases} \left(\frac{1}{2}\right)^{n+2} & ,if \quad x(n+1) = 1 \\ 0 & ,if \quad x(n+1) = 0 \end{cases}$$

Two states (q_4 and q_5) are used to have amplitudes $\left(\frac{1}{2}\right)^{2n}$ and $\left(\frac{1}{2}\right)^{n+1}$, respectively, in order to produce the current ε_n and δ_n. It is not possible to half unlimitedly the amplitudes in a quantum automaton but we have an MM-quantum automaton, and we use the Hadamard operation

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

instead, and we follow Hadamard operation by measuring part of the amplitude to REJECT.

We consider below the part of the states (q_3, q_6, q_5, q_7, q_8), among which the first one is q_3 and the third one is q_5. The rest of them are auxiliary states used to ensure that during the processing input symbol $x(n)$ the amplitudes are changed from $z, o, \frac{1}{2^n}, 0, 0, \ldots$ to $z, o, \frac{1}{2^{n+1}}, 0, 0, \ldots$ where z= *0.0{n}x(n) ... x(3)x(2)x(1).*

If *x(n)=1* then we use the following operation:

$$\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & 0 & 0 \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} & 0 \end{pmatrix} =$$

$$\begin{pmatrix} \frac{1}{2} & 0 & \frac{1}{2} & 0 & \frac{1}{\sqrt{2}} \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & -\frac{1}{\sqrt{2}} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} & 0 \end{pmatrix}$$

If x(n)=0 then we use the following operation:

$$\begin{pmatrix} \frac{1}{2} & -\frac{\sqrt{3}}{2} & 0 & 0 & 0 \\ \frac{\sqrt{3}}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & -\frac{\sqrt{3}}{2} & 0 \\ 0 & 0 & \frac{\sqrt{3}}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Now we consider the part of the states

$(q_2, q_9, q_4, q_{10}, q_{11}, q_{12}, q_{13}, q_{14}, q_{15}, q_{16}, q_{17}, q_{18})$, among which the first one is q_2 and the third one is q_4. The rest of them are auxiliary states used to ensure that during the processing input symbol x(n) the amplitudes arechanged from $z, o, \frac{1}{2^n}, 0, 0, \ldots$ to $z, o, \frac{1}{2^{n+1}}, 0, 0, \ldots$ where *z= 0.0{n}x(n) ... x(3)x(2)x(1).*

If x(n)=1 then we use the following operation

$$= \begin{pmatrix} \frac{1}{2} & -\frac{\sqrt{3}}{4} & 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 \\ \frac{\sqrt{3}}{4} & \frac{1}{2} & 0 & 0 & 0 & 0 & \frac{\sqrt{3}}{4} & \frac{\sqrt{3}}{4} & \frac{\sqrt{3}}{4} & 0 & 0 & 0 \\ \frac{1}{4} & 0 & 0 & \frac{1}{4} & \frac{1}{2} & 0 & -\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ -\frac{1}{4} & 0 & 0 & \frac{1}{4} & \frac{1}{2} & 0 & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & 0 & 0 & \frac{1}{4} & -\frac{1}{2} & 0 & -\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ -\frac{1}{4} & 0 & 0 & \frac{1}{4} & -\frac{1}{2} & \frac{1}{2} & \frac{1}{4} & -\frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \\ 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \\ 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

being the product of

$$\begin{pmatrix} \frac{1}{2} & -\frac{\sqrt{3}}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{\sqrt{3}}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

and

$$\begin{pmatrix} \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \\ 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \\ 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

If x(n)=0 then we use the following operation

$$\begin{pmatrix} \frac{1}{4} & -\frac{\sqrt{15}}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{\sqrt{15}}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{4} & -\frac{\sqrt{15}}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{\sqrt{15}}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

When all the input word is read, the operation corresponding to the end-marker confronts the states q_2 and q_3 with the Hadamard operation

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

and the resulting amplitude of q_3 is sent by measuring to ACCEPT. If the amplitudes for q_2 and q_3 have been equal before this operation, the word is rejected; otherwise it is accepted.

We are interested in recognition of PALINDROMES but Theorem 6. considers only the complement of this language.

It is not at all true that recognizability of a language implies the recognizability of the complement as well. It is so for deterministic finite automata and even for nondetermninistic finite automata. However, for nondetermninistic automata the size of the recognizing automaton may differ even exponentially. For probabilistic and

quantum finite automata with isolated cut-point it is so but in the case of non-isolated cut-point this has been an open problem for a long time.
We study the case of non-isolated cut-point 0 here. There is no problem for probabilistic automata because in this case probabilistic automata are equivalent to nondeterministic automata and they recognize only regular languages but regular languages are closed to complementation. We prove below that PALINDROMES can be recognized by MM-quantum finite automata with non-isolated cut-point 0.

Theorem 7.

Co-PALINDROMES can be recognized by an MM-quantum postselection finite automaton with probability 1.

Proof. The MM-quantum finite automaton recognizing Co-PALINDROMES
after the final application of Hadamard operation

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

measures the resulting amplitude of q_3.

If the amplitudes for q_2 and q_3 have been equal before this operation, the word is rejected; otherwise it is accepted.
The MM-quantum postselection finite automaton makes postselection after the Hadamard operation but before the final measuring.
The posselection set of states consists of one state only, namely, the state q_3. If the amplitudes for q_2 and q_3 have not been equal before the Hadamard operation, the postselection normalizes the amplitude to 1 or to -1. If If the amplitudes for q_2 and q_3 have been equal before the Hadamard operation, the postselection does not change the amplitude 0.

Theorem 8.

If a language L can be recognized by an MM-quantum postselection finite automaton with probability 1 then the complement of the language L can also be recognized by an MM-quantum postselection finite automaton with probability 1.

Proof. Obvious.

Corollary

PALINDROMES can be recognized by an MM-quantum postselection finite automaton with probability 1.

5.4 A context-free language that cannot be recognized by postselection finite quantum automata

We consider the following language.

$L = \{w \mid w \in \{0,1\}^*$ and there exist x, y, u, z such that $w = x1y = u1z$ and $|x| = |z|\}$

To paraphrase, L is the set of all strings for which there is a number d such that the d th symbol from the left and the d th symbol from the right, are 1.

We start with two geometrical lemmas.

Lemma 1.

The two following assertions are equivalent:

1) Given a r-dimensional real Euclidean space, there exist N distinct $(r-1)$-dimensional hyperplanes dividing the space into 2^N non-empty regions;

2) $N \le r$.

For instance, one line divides a plane into 2 regions, two lines can divide the plane into $2^2 = 4$ regions, but no 3 lines can exist dividing the plane into $2^3 = 8$ regions.

Proof. $\Leftarrow$ Immediate.

$\Rightarrow$ Assume that $N > r$. We denote the N hyperplanes by

$a_{11}x_1 + a_{12}x_2 + ... + a_{1N}x_N = d_1$

$a_{21}x_1 + a_{22}x_2 + ... + a_{2N}x_N = d_2$

...

$a_{N1}x_1 + a_{N2}x_2 + ... + a_{NN}x_N = d_N$

$N > r$ implies that the rank r_M of the matrix of this system of equalities is at most $r < N$. Hence in the system of vectors

$(a_{11}, a_{12}, ..., a_{1N})$

$(a_{21}, a_{22}, ..., a_{2N})$

...

$(a_{N1}, a_{N2}, ..., a_{NN})$

all the vectors can be represented as linear combinations of r_M of them. (With no restriction of generality we can assume that these are the first r_M vectors in this system.) Then the system of linear equations

$a_{11}x_1 + a_{12}x_2 + ... + a_{1N}x_N = d_1$

$a_{21}x_1 + a_{22}x_2 + ... + a_{2N}x_N = d_2$

...

$a_{r_M 1}x_1 + a_{r_M 2}x_2 + ... + a_{r_M N}x_N = d_{r_M}$

has exactly one solution $(c_1, c_2, ..., c_N)$.

we can change variables $x_1, x_2, ..., x_N$ into $x\ , x_2, ..., x_N$ by a linear transformation in such a way that the system of the first r_M equations becomes a subsystem

$y_1 = 0$

$y_2 = 0$

...

$y_{r_M} = 0$

The point $(c_1, c_2, ..., c_N)$ becomes the point $(0,0,...,0)$. The hyperplanes defined by the abovementioned r_M linear equations indeed divide the N-dimensional space into 2^{r_M} distinct nonempty regions. However, for every other hyperplanes

$a_{j1}y_1 + a_{j2}y_2 + ... + a_{jN}y_N = c_j$

we can assert that there is at least one of the 2^{r_M} abovementioned nonempty regions which is not divided by this hyperplane. For instance, if the point $B_j = (e_1, e_2, ..., e_N)$ is the point of this hyperplane with the minimum distance form the point $(0,0,...,0)$ then the hyperplane does not divide the region containing the point $(-e_1, -e_2, ..., -e_N)$.

Hence the hyperplanes cannot divide the space into 2^N distinct nonempty regions.

Contradiction.

Lemma 2.

The two following assertions are equivalent:

1) Given a r-dimensional real Euclidean space, there exist N distinct $(r-1)$-dimensional hyperplanes $H_1, H_2, ..., H_N$ and 2^N points such that for arbitrary word $w \in \{0,1\}*$ if $w = w_1 w_2 ... w_N$ then the corresponding point belongs to the hyperplane H_v iff $w_v = 1$

2) $N \leq r$.

Proof. $\Leftarrow$ Immediate.

$\Rightarrow$ Assume that $N > r$.

Consider a set J of N points of a real r-dimensional space corresponding to words $w_1, w_2, ..., w_N$ containing exactly 1 symbol 1 and $N-1$ symbols 0. The assertion 1) of our Lemma implies that every point from the set J belongs to exactly $N-1$ hyperplanes H_v, and each hyperplane H_v contains all points of the set J but exactly one. An elementary theorem of linear algebra asserts that $r+1$ elements of an r-dimensional space cannot be linearly independent. Hence at least one point of the set J can be represented as a linear combination of the remaining $N-1$ points. However, there is a hyperplane $N-1$ not containing this point but containing all the other points. On the other hand, any linear combination of points in a hyperplane also belongs to this hyperplane.

Contradiction.

Now we proceed to prove

Theorem 9.

The language L cannot be recognized by an MM-quantum postselection finite automaton with probability 1.

Proof. Assume from the contrary that L can be recognized. In the first part of the proof we consider the case when the quantum automaton is an MO-quantum postselection finite automaton with probability 1. It accepts all the words in L with probability 1, and rejects all the words not in L with probability 1.

We denote the number of states of this automaton by k.

We take a natural number $N > 4k$ and for all 2^N binary words x we consider the distribution of amplitudes

$\left(p_1(x), p_2(x), ... p_k(x)\right)$

to be in the corresponding states after input of the word x. This distribution of amplitudes can be represented as a point in a k-dimensional unit cube over the field of complex numbers.

We denote by $y_1, y_2, ..., y_N$ binary words consisting of many zeros and only one symbol 1, namely, only the symbol number i in the word y_i equals 1. We denote by $Z(x)$ the set of all the natural numbers i such that the i-th symbol of the word x (counted from the right to the left) equals 1. It is easy to see that

$xy_i \in L \Leftrightarrow i \in Z(x)$

Hence there are exactly 2^N words x nonequivalent with respect to the set $y_1, y_2, ..., y_N$. We denote the set of these words by T_N.

For every $i, j, t \in \{1,2,...,k\}$ we consider the amplitude $s_t^j(y_i)$ describing the transition from the state j to the state t during the processing of y_i.

Since our automaton works with probability 1, after the postselection and measurement of all accepting statesany word xy_i is accepted either with the probability 1 or with the probability 0.

With no restriction on generality we can assume that the accepting states are $1,2,...,u$. Then the probability to accept xy_i equals

$$P(x,i) = \left|p_1(x)s_1^1(y_i) + p_2(x)s_1^2(y_i) + ... + p_k(x)s_1^k(y_i)\right|^2 +$$

$$+\left|p_1(x)s_2^1(y_i) + p_2(x)s_2^2(y_i) + ... + p_k(x)s_2^k(y_i)\right|^2 +$$

$$+ ... +$$

$$+\left|p_1(x)s_u^1(y_i) + p_2(x)s_u^2(y_i) + ... + p_k(x)s_u^k(y_i)\right|^2,$$

where $|\alpha|^2$ is square of modulo of the complex number α.

Now we consider two distinct binary words x_1 and x_2 of the length N. Since they are distinct, there is at least one value of i such that $Z(x_1)$ differs from $Z(x_2)$ in the number i. Hence either $P(x_1,i)=1$ and $P(x_2,i)=0$, or $P(x_1,i)=0$ and \$ $P(x_2,i)=1$.

For every $v \in \{1,2,...,u\}$ we denote the real part and the imaginary part of the complex number $p_1(x)s_v^1(y_i) + p_2(x)s_v^2(y_i) + ... + p_k(x)s_v^k(x)$ by $\mathrm{Re}(x,i,v)$ and $\mathrm{Im}(x,i,v)$.

Equality $P(x,i)=0$ implies $\mathrm{Re}(x,i,v)=0$ and $\mathrm{Im}(x,i,v)=0$ for all $1 \le v \le u$.

Inequality $P(x,i) \neq 0$ implies either $\mathrm{Re}(x,i,v) \neq 0$ or $\mathrm{Im}(x,i,v) \neq 0$ (or both of them) for every $1 \leq v \leq u$.

We considered above

$(p_1(x), p_2(x), \dots p_k(x))$

as a point in a complex-valued unit cube. In order to use our Lemma2 we wish to transform this complex-valued unit cube into an auxiliary real-valued space at the expense of slight loosing the connection with our quantum finite automaton.

For every $x \in T_N$ (independently of all other values of x) we consider a k-dimensional space S_x and for the fixed x we replace the point $(p_1(x), p_2(x), \dots p_k(x))$ by a real point $(r_1(x), r_2(x), \dots r_k(x))$ where

$$r_m(x) = \lambda_1 . \mathrm{Re}(p_1(x)) + \lambda_2 . \mathrm{Re}(p_2(x)) + \dots + \lambda_k . \mathrm{Re}(p_k(x)) +$$
$$\mu_1 . \mathrm{Im}(p_1(x)) + \mu_2 . \mathrm{Im}(p_2(x)) + \dots + \mu_k . \mathrm{Im}(p_k(x))$$

and

$\lambda_1, \lambda_2, \dots, \lambda_k, \mu_1, \mu_2, \dots, \mu_k$

are specially chosen (irrational) numbers.

Notice that

$$p_1(x) s_1^1(y_i) + p_2(x) s_1^2(y_i) + \dots + p_k(x) s_1^k(x) = 0$$

is equivalent to

$$\mathrm{Re}(p_1(x)) \mathrm{Re}(s_1^1(y_i)) - \mathrm{Im}(p_1(x)) \mathrm{Im}(s_1^1(y_i)) +$$
$$+ \mathrm{Re}(p_2(x)) \mathrm{Re}(s_1^2(y_i)) - \mathrm{Im}(p_2(x)) \mathrm{Im}(s_1^2(y_i)) +$$
$$+ \dots +$$
$$+ \mathrm{Re}(p_k(x)) \mathrm{Re}(s_1^k(y_i)) - \mathrm{Im}(p_k(x)) \mathrm{Im}(s_1^k(y_i)) = 0$$

and

$$\mathrm{Re}(p_1(x)) IM(s_1^1(y_i)) + \mathrm{Im}(p_1(x)) \mathrm{Re}(s_1^1(y_i)) +$$
$$+ \mathrm{Re}(p_2(x)) \mathrm{Im}(s_1^2(y_i)) + \mathrm{Im}(p_2(x)) \mathrm{Re}(s_1^2(y_i)) +$$
$$+ \dots +$$
$$+ \mathrm{Re}(p_k(x)) \mathrm{Im}(s_1^k(y_i)) + \mathrm{Im}(p_k(x)) \mathrm{Re}(s_1^k(y_i)) = 0$$

We have N possible choices for i and 2^N choices for x. For each pair (x,i) we consider the linear combinations

$\beta_i.[\ \operatorname{Re}(p_1(x))\operatorname{Re}(s_1^1(y_i)) - \operatorname{Im}(p_1(x))\operatorname{Im}(s_1^1(y_i)) +$

$+\operatorname{Re}(p_2(x))\operatorname{Re}(s_1^2(y_i)) - \operatorname{Im}(p_2(x))\operatorname{Im}(s_1^2(y_i)) +$

$+ \dots +$

$+\operatorname{Re}(p_k(x))\operatorname{Re}(s_1^k(y_i)) - \operatorname{Im}(p_k(x))\operatorname{Im}(s_1^k(y_i))\ ] +$

$+\gamma_i.[\ \operatorname{Re}(p_1(x))IM(s_1^1(y_i)) + \operatorname{Im}(p_1(x))\operatorname{Re}(s_1^1(y_i)) +$

$+\operatorname{Re}(p_2(x))\operatorname{Im}(s_1^2(y_i)) + \operatorname{Im}(p_2(x))\operatorname{Re}(s_1^2(y_i)) +$

$+ \dots +$

$+\operatorname{Re}(p_k(x))\operatorname{Im}(s_1^k(y_i)) + \operatorname{Im}(p_k(x))\operatorname{Re}(s_1^k(y_i))\ ].$

In this paragraph we define the values β_i and γ_i depending on x but the dependence is such that we have a continuum choices for β_i and γ_i. For a finite number of pairs (x,i) (namely, for those pairs where $P(x,i)=1$) we demand that these linear combinations are not equal zero. Now we choose one value for every β_i and γ_i such that all the needed inequalities hold. Now our values of β_i and γ_i no more depend on x.

Notice that for those pairs where $P(x,i)=0$ the equality holds

$\beta_i.[\ \operatorname{Re}(p_1(x))\operatorname{Re}(s_1^1(y_i)) - \operatorname{Im}(p_1(x))\operatorname{Im}(s_1^1(y_i)) +$

$+\operatorname{Re}(p_2(x))\operatorname{Re}(s_1^2(y_i)) - \operatorname{Im}(p_2(x))\operatorname{Im}(s_1^2(y_i)) +$

$+ \dots +$

$+\operatorname{Re}(p_k(x))\operatorname{Re}(s_1^k(y_i)) - \operatorname{Im}(p_k(x))\operatorname{Im}(s_1^k(y_i))\ ] +$

+Re(p_k(x))Re(s^k_1(y_i))-Im(p_k(x))Im(s^k_1(y_i))]+

$+\gamma_i.[\ \operatorname{Re}(p_1(x))\operatorname{Im}(s_1^1(y_i)) + \operatorname{Im}(p_1(x))\operatorname{Re}(s_1^1(y_i)) +$

$+\operatorname{Re}(p_2(x))\operatorname{Im}(s_1^2(y_i)) + \operatorname{Im}(p_2(x))\operatorname{Re}(s_1^2(y_i)) +$

$+ \dots +$

$+\operatorname{Re}(p_k(x))\operatorname{Im}(s_1^k(y_i)) + \operatorname{Im}(p_k(x))\operatorname{Re}(s_1^k(y_i))\ ] = 0\,.$

Every such equality may be re-written as

$\operatorname{Re}(p_1(x)).[\beta_1 \operatorname{Re}(s_1^1(y_i)) + \gamma_1 \operatorname{Im}(s_1^1(y_i))] +$

$+\operatorname{Re}(p_2(x)).[\beta_2 \operatorname{Re}(s_1^2(y_i)) + \gamma_2 \operatorname{Im}(s_1^2(y_i))] +$

$$
\begin{aligned}
&+ \ldots + \\
&+ \mathrm{Re}(p_k(x)).\left[\beta_k \mathrm{Re}\left(s_1^k(y_i)\right) + \gamma_k \mathrm{Im}\left(s_1^k(y_i)\right)\right] + \\
&+ \mathrm{Im}(p_1(x)).\left[-\beta_1 \mathrm{Im}\left(s_1^1(y_i)\right) + \gamma_1 \mathrm{Re}\left(s_1^1(y_i)\right)\right] + \\
&+ \mathrm{Im}(p_2(x)).\left[-\beta_2 \mathrm{Im}\left(s_1^2(y_i)\right) + \gamma_2 \mathrm{Re}\left(s_1^2(y_i)\right)\right] + \\
&+ \ldots + \\
&+ \mathrm{Im}(p_k(x)).\left[-\beta_k \mathrm{Im}\left(s_1^k(y_i)\right) + \gamma_k \mathrm{Re}\left(s_1^k(y_i)\right)\right] = 0
\end{aligned}
$$

This equality can be interpreted as a $(2k-1)$-dimensional hyperplane in a real $2k$-dimensional space

$$\left(\mathrm{Re}(p_1(x)), \mathrm{Re}(p_2(x)), \ldots, \mathrm{Re}(p_k(x)), \mathrm{Im}(p_1(x)), \mathrm{Im}(p_2(x)), \ldots, \mathrm{Im}(p_k(x))\right)$$

and the coefficients of the hyperplane do not depend on x.

We have 2^N points in this space corresponding to pairs (x,i). If $P(x,i)=0$ then the point is on the corresponding hyperplane. If $P(x,i)=1$ then the point is not on the corresponding hyperplane. Now for each of the 2^N points separately and independently one from another we construct new points $R_1, R_2, \ldots, R_k$, $I_1, I_2, \ldots, I_k$ such that all 2^N new points are outside these N hyperplanes. If $P(x,i)=1$ then we have an inequality of type "$\neq 0$". Hence we alway have N hyperplanes and 2^N points in a $2k$-dimensional space such that they all are in distinct regions divided by these hyperplanes. Since we assumed from the contrary that $N > 2k$ this is a contradiction with Lemma2 proving the first case of our theorem (for MO-quantum automata).

In the second part of the proof we consider the case when the quantum automaton is an MM-quantum postselection finite automaton with probability 1.

The proof differs from the proof in the first part only by additional step which precedes the old proof.

We replace the MM-quantum automaton by an automaton which is neither quantum nor probabilistic. The states of the old automaton function as before but there are two new states: one to accumulate acceptation probability, the other one to accumulate rejection probability before the end of the processing. This computation cannot be done by matrix multiplication but this does not influence the remaining part of the proof.

5.5 Probabilistic VS quantum finite automata with postselection

In this section some results about quantum finite automata with postselection are shown, that are obtained by Abuzer Yakary_lmaz and A.C. Cem Say. [22].

It turns out that QFAs with Aaronson postselection have the same computational power as the recently introduced one-way QFAs with restart [23], and that QFAs with postselection are strictly more powerful than classical probabilistic finite automata (PFAs) with postselection.

We call the class of languages recognized by 1PostPFAs with bounded error PostS (post-stochastic languages).

Bounded error language recognition of 1PostQFAs is similar to those of 1PostPFAs. We call the class of languages recognized by 1PostQFAs with bounded error PostQ (post-quantum languages).

Abuzer Yakarylmaz and A.C. Cem Say call QFA with postselection that is defined in **Definition 1.** Latvian 1PostQFAs or Latvian quantum finite automata with postselection. The bounded-error classes corresponding to the 1LPostPFA and 1LPostQFA models are called LPostS and LPostQ, respectively.

Here are some results obtained by Abuzer Yakarylmaz and A.C. Cem Say:

Theorem 10.
$NQL \cup coNQL \subseteq LPostQ$,

where NQL is a class of languages recognized by nondeterministic finite automata.

Theorem 11.
$LPostQ \subseteq UQL$,

where class of unbound-error quantum languages $UQL = QL \cup coQL$, *where QL(coQL) are class of languages recognized by 1QFAs with cutpoint (nonstrict cutpoint) called quantum languages (co-quantum languages).*

Here is a summary of results:

$$\begin{tikzcd}
 & \subsetneq \mathrm{QL} \\
\mathrm{PostS} = \mathrm{LPostS} \subsetneq \mathrm{PostQ} \arrow[ur] \arrow[dr] & \\
 & \subseteq \mathrm{LPostQ} \subseteq \mathrm{UQL}
\end{tikzcd}$$

6 Quantum algorithms and lower bounds

6.1 Graph problems

Boolean decision trees model is the simplest model to compute Boolean functions. In this model the primitive operation made by an algorithm is evaluating an input Boolean variable. The cost of a (deterministic) algorithm is the number of variables it evaluates on a worst-case input. It is easy to find the deterministic complexity of all explicit Boolean functions (for most functions it is equal to the number of variables).

The <u>*black-box*</u> model of computation arises when one is given a black-box containing an N-tuple of Boolean variables $X=(x_0,x_1,...,x_{N-1}.)$. The box is equipped to output x_i on input i. We wish to determine some property of X, accessing the x_i only through the black-box. Such a black-box access is called a <u>*query*</u>. A property of X is any Boolean function that depends on X, i.e. a property is function $f:\{0,1\}^N \rightarrow \{0,1\}$. We want to compute such properties using as few queries as possible.

Consider, for example, the case where the goal is to determine whether or not X contains at least one 1, so we want to compute the property $\mathrm{OR}(X)= x_0 \vee ... \vee x_{N-1}$. It is well known that the number of queries required to compute OR by any *classical* (deterministic or probabilistic) algorithm is O(N).
Grover [7] discovered a remarkable *quantum* algorithm that, making queries in superposition, can be used to compute OR with small error probability using only $O(\sqrt{N})$ queries.
On the other hand, quantum algorithms are in a sense more restricted. For instance, only unitary transformations are allowed for state transitions. Hence rather often a problem arises whether or not the needed quantum automaton exists. In such a situation lower bounds of complexity are considered. It is proved in [8] that Grover database search algorithm is the best possible. It is proved in [8] that no quantum query algorithm exists for PARITY with $\Omega(N)$ queries, etc.

We use a result by A.Ambainis [9] to prove lower complexity bounds for quantum query algorithms. Currently, this is the most powerful method to prove lower bounds of complexity for quantum query algorithms. In some cases there still remains a gap between the upper and the lower bounds of the complexity. In these cases we prove additionally that

Ambainis' method cannot provide a better lower bound for this problem.

6.1.1 Query model

In the query model, the input $x_1,\ldots x_N$ is contained in a black box and can be accessed by queries to the black box. In each query, we give i to the black box and the black box outputs x_i. The goal is to solve the problem with the minimum number of queries. The classical version of this model is known as *decision trees* [24].

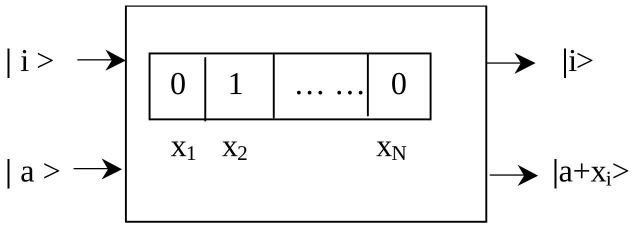

Fig. 2. ***Quantum black box.***

There are two ways how to define the query box in the quantum model. The first is the extension of the classical query (Fig. 2). It has two inputs: i, consisting of [logN] bits and b consisting of 1 bit. If the input to the query box is a basic state $|i\rangle|b\rangle$, the output is $|i\rangle|b\oplus x_i\rangle$. If the input is a superposition $\sum_{i,b} a_{i,b}|i\rangle|b\rangle$, the output is $\sum_{i,b}a_{i,b}|i\rangle|b\oplus x_i\rangle$. Notice that this definition applies both to case when x_i are binary and to the case when they are k-valued. In the k-valued case, we just make b to consist of $\lceil \log_2 k \rceil$ bits and take $b\oplus x_i$ to be bitwise XOR of b and x_i.

In the second form of quantum query (which only applies to problems with {0,1}-valued x_i), the black box has just one input i. If the input is a state $\sum_i a_i|i\rangle$, the output is $\sum_i (-1)^{x_i} a_i |i\rangle$. While this form is less intuitive, it is very convenient for the use in quantum algorithms, including Grover's search algorithm [7]. A query of second type can be simulated by a query of first type [7].

A quantum query algorithm with T queries is just a sequence of unitary transformations

$$U_0 \rightarrow O \rightarrow U_1 \rightarrow O \rightarrow \ldots \rightarrow U_{T-1} \rightarrow O \rightarrow U_T$$

on some finite- dimensional space C^k. U_0, U_1, …, U_T can be any unitary transformations that do not depend on the bits $x_1, \ldots, x_N$ inside the black box. O are query transformations that consist of applying the query box to the first logN+1 bits of the state. That is, we represent basic states of C^k as $|i,b,z\rangle$. Then, O maps $|i,b,z\rangle$ to $|i,b\oplus x_i,z\rangle$. We use O_x to denote the query transformation corresponding to an input $x=(x_1,\ldots x_N)$.

The computation starts with state $|0\rangle$. Then, we apply U_0, O_x ,…, O_x, U_T and measure the final state. The result of the computation is the rightmost bit of the state obtained by the measurement (or several bits if we are considering a problem where the answer has more than 2 values).

The quantum algorithm computes a function $f(x_1,\ldots,x_N)$ if, for every $x=(x_1,\ldots,x_N)$ for which f is defined, the probability that the rightmost bit of $U_T\, O_x\, U_{T-1}\ldots\, O_x\, U_0|0\rangle$ equals $f(x_1,\ldots,x_N)$ is at least $1\text{-}\varepsilon < ½$.

The query complexity of f is the smallest number of queries used by a quantum algorithm that computes f. We denote it $Q(f)$.

Our proofs use the following results by A.Ambainis.

Theorem 12 [9].

Let $A \subset \{0,1\}^n$, $B \subset \{0,1\}^n$ be such that $f(A)=1$, $f(B)=0$ and for every $x=(x_1..x_n)\in A$, there are at least m values $i \in \{1,\ldots,n\}$such that $(x_1,\ldots,x_{i-1},1\text{-}x_i,x_{i+1},\ldots,x_n) \in B$, for every $x=(x_1..x_n)\in B$, there are at least m' values $i \in \{1,\ldots,n\}$such that

$(x_1,\ldots,x_{i-1},1\text{-}x_i,x_{i+1},\ldots,x_n) \in A$.

Then $Q(f)= \Omega(\sqrt{mm'})$.

Theorem 13 [9].

Let $f(x_1,x_2, \ldots ,x_n)$ be a function of n {0, 1} - valued variables and X, Y be two sets of inputs such that $f(x)\neq f(y)$ if $x \in X$ and $y \in Y$. Let $R \subset X^* Y$ be such that

For every $x \in X$ there exist at least m different $y \in Y$ such that $(x, y) \in R$,

For every $y \in Y$ there exist at least m' different $x \in X$ such that $(x, y) \in R$,

For every $x \in X$ and $i \in \{1, ..., n\}$ there are at most l_i different $y \in Y$ such that $(x, y) \in R$ and $x_i \neq y_i$,

For every $y \in Y$ and $i \in \{1, ..., n\}$ there are at most l_i' different $x \in X$ such that $(x, y) \in R$ and $x_i \neq y_i$,

Then, any quantum algorithm computing f uses

$\Omega(\sqrt{\frac{mm'}{\max(l_i * l'_i)}})$ *queries.*

Definition. *For any Boolean function $f:\{0,1\}^N \to \{0,1\}$ and any $x=(x_1..x_n)$, $ND(f,x)$ is the number of queries needed by nondeterministic algorithms on the values $x=(x_1..x_n)$.*

Definition. *For any Boolean function $f:\{0,1\}^N \to \{0,1\}$:*
$ND_0(f)= \max_{f(x)=0} ND(f,x)$ and $ND_1(f)= \max_{f(x)=1} ND(f,x)$.

Theorem 14 [25].
Whatever the sets A and B, Theorem 1 cannot prove a better lower bound for the query complexity Q(f) than $\sqrt{ND_0(f) \cdot ND_1(f)}$.

6.1.2 Graph problems

We consider the following graph problems:

Problem 1. Hamiltonian circuit

INSTANCE: Graph G=(V,E).

QUESTION: Does G contain Hamiltonian circuit?

Problem 2. Directed Hamiltonian circuit

INSTANCE: Directed graph G=(V,A).

QUESTION: Does G contain directed Hamiltonian circuit?

Problem 3. Hamiltonian path

INSTANCE: Graph G=(V,E).

QUESTION: Does G contain Hamiltonian circuit?

Problem 4. Travelling salesman

INSTANCE: Set C of m cities, distance $d(c_i, c_j) \in Z^+$ for each pair of cities, $c_i\ c_j \in C$, positive integer B.

QUESTION: Is there a tour of C having length B or less, i.e. a permutation $< c_{\pi(1)}, c_{\pi(2)}, ..., c_{\pi(m)} >$ of C such that

$$\left(\sum_{i=1}^{m} d(c_{\pi(i)}, c_{\pi(i+1)}) \right) + d(c_{\pi(m)}, c_{\pi(1)}) \leq B \, ?$$

Lemma 3.
If a graph G=(V,E), |V|=5n, satisfes the following requirements:

there are n mutually not connected (red) vertices,

there are 2n green vertices not connected with red ones, green vertices are grouped in pairs and each pair is connected by edge,

subgraph induced by the rest 2n vertices (black) is a complete graph and all black vertices are connected to all red and green vertices,

then ***Hamiltonian circuit*** *problem is solvable.*

Proof: We denote black vertices m_1 to m_{2n}. Red vertices are denoted k_1 to k_n, pairs of green with k_{n+1} to k_{2n}. Sequence $m_1\ k_1\ ..\ m_n\ k_n\ m_{n+1}\ k_{n+1}\ ..\ m_{2n}\ k_{2n}\ m_1$ (i.e. black, red, ..black, red, black, green, green, …,black, green, green, black) satisfies **Hamiltonian circuit problem.**

□

Lemma 4.
If graph G=(V,E), |V|=5n, satisfies the following requirements:

there are n+2 mutually not connected (red) vertices,

there are 2n-2 green vertices not connected with red ones, green vertices are grouped in pairs and each pair is connected by edge,

subgraph induced by the rest 2n vertices (black) is a complete graph and all black vertices are connected to all red and green vertices,

then ***Hamiltonian circuit*** *problem is not solvable.*

Proof: The red vertices and the pairs of green vertices are mutually not connected. The only way to get from one red vertice to another (or from one green pair to another) is through some black vertice.

There are 2n black in the graph, but n+2 red vertices, and n-1 green pair makes altogether 2n+1. So at least one of the black vertices will be used twice, which is not allowed in Hamiltonian circuit.

□

Theorem 15.
Hamiltonian circuit problem *requires $\Omega(n^{1.5})$ quantum queries.*

Proof: We construct the sets *A* and *B* for the usage of Theorem 12 [9].

The set *A* consists of all graphs *G* satisfying the requirements of Lemma 1. The value of the function corresponding to the **Hamiltonian circuit** problem is 1. (This follows from Lemma 3.) *B* consists of all graphs *G* satisfying the requirements of Lemma 4. The value of the function corresponding the **Hamiltonian circuit** problem is 0. (This follows from Lemma 4.)

From each graph $G \in A$, we can obtain $G' \in B$ by disconnecting any one of the edges, which connect the green vertices. Hence m=n=O(n). From each graph $G \in B$, we can obtain $G' \in A$ by connecting any two red vertices. Hence m'=(n+2)(n+1)=O(n^2).

By Theorem 12 [9]., the quantum query complexity is $\Omega\sqrt{n \cdot n^2}$ =$\Omega(n^{1.5})$.

□

The same idea proves Theorem 16.

Theorem 16.
Directed Hamiltonian circuit *requires $\Omega(n^{1.5})$ quantum queries.*

□

Lemma 5.
If graph G=(V,E), |V|=5n, satisfies the requirements of Lemma 4., then **Hamiltonian path** problem is solvable.

Proof: We denote black vertices m_1 to m_{2n}. Red vertices are denoted k_1 to k_{n+2}, pairs of green with k_{n+3} to k_{2n+1}. Sequence k_1 m_1 .. k_{n+2} m_{n+2} k_{n+3} m_{n+3}.. m_{2n} k_{2n+1} (i.e. *red, black, .. red, black, green, green, black, …,black, green, green*) satisfies **Hamiltonian path problem.**

□

Lemma 6.
If graph G=(V,E), |V|=5n, satisfies the following requirements:
there are n+4 mutually not connected (red) vertices,
there are 2n-4 green vertices not connected with red ones, green vertices are grouped in pairs and each pair is connected by edge,
subgraph induced by the rest 2n vertices (black) is a complete graph and all black vertices are connected to all red and green vertices,
then ***Hamiltonian path*** *problem is not solvable.*

Proof: The proof is analogical to that of Lemma 4.

□

Theorem 17.
Hamiltonian path *requires $\Omega(n^{1.5})$ quantum queries.*

Proof: We construct the sets A and B for the usage of Theorem 12 [9].
The set A consists of all graphs G satisfying the requirements of Lemma 4. The value of the function corresponding to the **Hamiltonian path** problem is 1. (This follows from Lemma 3.) B consists of all graphs G satisfying the requirements of Lemma 4. The value of the function corresponding the **Hamiltonian path** problem is 0. (This follows from Lemma 4.)

From each graph $G \in A$, we can obtain $G' \in B$ by disconnecting any one of the edges, which connect the green vertices. Hence m=n-1=O(n). From each graph $G \in B$, we can obtain $G' \in A$ by connecting any two red vertices. Hence $m'=(n+4)(n+3)=O(n^2)$.
By Theorem 12 [9]., the quantum query complexity is $\Omega\sqrt{n \cdot n^2} = \Omega(n^{1.5})$.

□

Theorem 18.
Travelling salesman *requires $\Omega(n^{1.5})$ quantum queries.*

Travelling salesman problem can be easily reduced to **Hamiltonian circuit** problem, by taking all the distances equal to 1 and *B* equal to number of cities.

□

Theorem 19.
The lower bound for ***Hamiltonian circuit*** *cannot be improved by Ambainis' method.*

Proof: We use Theorem 14 [25]. Let the Boolean function *f* describe *Hamiltonian circuit*.
$ND_1(f) = O(n)$, because it suffices to guess the sequence of vertices and ask the edge for every pair of subsequent vertices.
$ND_0(f) = O(n^2)$, because it suffices to check that a graph satisfies conditions of Lemma 4.

Hence $\sqrt{ND_1(f) \cdot ND_0(f)} = O(n^{1.5})$.

□

6.2 3-Sum problem

We observe the following problem, called 3-Sum problem:
Definition 1. *Given the set S of N numbers, detect whether there are three numbers $a \in S$; $b \in S$; $c \in S$, such that $a + b + c = 0$.*

Alternative model, often called 3-sum', is:
Definition 2. *Given the 3 sets A, B and C each of N numbers, detect whether there are three numbers $a \in A$; $b \in B$; $c \in C$, such that $a + b = c$.*
There is a big cluster of problems in computational geometry that are called 3-Sum Hard. Gajentaan and Overmas [26] described them as problems that can be reduced to the 3-Sum problem. The example is, for instance, a GeomBase problem: given points

on three equally spaced horizontal lines, are there points, one from each line, that are collinear.

In classical computation the best currently known algorithm for any 3-Sum Hard problem takes $O(N^2)$ time, while the best lower bound for the time complexity is Ω (*NlogN*), which is very low and unreachable. It is believed that 3-Sum lower bound is the same as upper bound, $\Omega(N^2)$, so 3-Sum hard problems in classical computation sometimes call - N^2 *hard*. For some of 3-Sum hard problems $\Omega(N^2)$ lower bound has been proved.

6.3 Quantum algorithm for the 3-Sum problem

This section show algorithm and lower bound, that is presented in [27].
In this algorithm we make use of quantum amplitude amplification method, which generalizes Grover quantum search. Here is an essence of amplitude amplification:

Theorem 20.
There exists the quantum algorithm QSearch with the following property. Let A be any quantum algorithm that uses no measurements, and let $\chi : Z \to \{0,1\}$ *be any boolean function. Let a denote the initial success probability of A of finding a solution (i.e. the probability of outputting z such that* $\chi(z)$ *= 1). Algorithm QSearch finds a solution using an expected number of* $O(1/\sqrt{a})$ *applications of A and* A^{-1} *if a > 0, and otherwise runs forever.*
The algorithm QSearch does not need to know the value of a in advance, but if a is known, it can find a solution in worst-case $O(1/\sqrt{a})$ applications.

Theorem 21.
There exists a quantum algorithm that solves 3-Sum problem in O(NlogN).
Proof. The algorithm works as follows:
1. Sort set C classically, that takes O(NlogN) time.
2. Construct an algorithm that can solve the problem with small probability.
The algorithm takes an input of two random elements, one from set A and the other from set B and outputs whether these two elements are summing up to some element from C:
(a) Compute a+b, a $\in$ A; b $\in$ B.

(b) Check whether a+b can be found in C. Needs $O(logN)$ time, because C is sorted.

3. Construct quantum superposition over all the $|a\rangle|b\rangle$ and use amplitude amplification on that superposition with the algorithm just described as a kernel. Amplitude amplification method uses Grover algorithm idea to speed up computation. The maximum speedup it allows to get is quadratic. In our case, classically we must repeat algorithm kernel steps $O(N^2)$ times. Amplitude amplification method allows us to get the same result, using only O(N) steps.

So the total time the algorithm uses is O(NlogN).

There are several approaches for estimating lower bounds for quantum algorithms.

Theorem 22. 3-Sum problem has quantum lower bound $\Omega\left(\sqrt{N}\right)$

Proof. The proof is based on A. Ambainis adversary method of proving quantum lower bounds. We use his **Theorem 13.**

That means, that we need to find all the variants how the input x can be modified.

For our case, we'll take X to contain only one element: A consisting of all zeros; B - all 1s, C - all 2s. It is an input, on which our function returns 0. Let Y contain all inputs made of X, with one element in any of sets A, B and C changed so, that the function returns 1 (e.g., any element of A changed to 1; any element of B - to 2; any element of C - to 1). Let $R \subset X \times Y$ consist of such $X \times Y$ pairs where y differs from x in exactly one position. According to the theorem, m = 3N, because for every x $\in$ X there are exactly 3N different y $\in$ Y , which differs from x in exactly one position. m' = 1, because for every y $\in$ Y there is only one x $\in$ X, which differs from y in exactly one position. l = l' = 1 that follows from our definition of R.

Using this formula we get a lower bound $\Omega\left(\sqrt{3N}\right)$.

Unfortunately, this method gives almost trivial result in this case. The better idea was to try the same method as in classical case.

6.4 Improved lower bound for 3-Sum problem

Classical lower bound Ω(*NlogN*) follows from the technique of Dobkin and Lipton [28] in the linear decision tree model. They observed that the set of inputs following a fixed computational path through a linear decision tree is connected. Since the set of nondegenerate inputs has $n^{\Omega(n)}$ connected components, any linear decision tree must have $n^{\Omega(n)}$ leaves and therefore must have depth Ω (*NlogN*). As quantum algorithm cannot give any speedup on a linear decision tree, we must conclude, that quantum lower bound for 3-Sum is Ω (*NlogN*).

6.5 3-sum generalization

The most natural generalization of 3-Sum problem is its r-Sum:

Definition 7. *Given a set S of N numbers, detect whether there are r numbers in S which sum to zero.*

Alternative definition called r-Sum’ is the following:

Definition 8. *Given r sets $S_{1}, ..., S_{r}$ of N numbers, detect whether there are r numbers one from each set that sum to zero.*

Similarly we can define the class of r-Sum Hard problems.

In deterministic case these problems have a lower bound of $\Omega(N \log N)$ and best known deterministic algorithm can solve the r-sum problem in $\mathrm{O}(N^{r+1/2})$ when r is odd and $\mathrm{O}(N^{r/2}\log(r))$ when r is even.

Theorem 23.

Quantum algorithm can solve r-Sum problem in $\mathrm{O}(N^{\lceil r/3 \rceil}\log N)$ *time.*

Proof. We will further divide sets $S_{1}, ..., S_{r}$ in two more groups of sets: First group will contain x sets of data - let’s call them $C_1,..., C_x$ and the second will contain remaining (r-x) sets of data - let’s call them $Q_1,..., Q_{r-x}$.

The algorithm itself consists of two parts - classical and the quantum one.

First, we execute the classical part of the algorithm and then execute the quantum part of it.

Classical part of algorithm works as follows: we take sets $C_1,..., C_x$ and perform following operations on them:

1. Make all the possible groups of elements picking one from each of sets $C_1,..., C_x$ and sum them up. (Call this new set CSum)
2. Sort the summary set Csum

After classical part of algorithm finishes its work we will get the sorted set of all the possible element combinations in sets $C_1,..., C_x$. This will take $\mathrm{O}(N^{x}\log N)$ time.

Then starts the quantum part of the algorithm, that uses sets $Q_1,\ldots, Q_{r-x}$ as well as the set CSum. The quantum part of r-sum algorithm is in fact generalized version of quantum 3-sum algorithm:

1. Construct the probabilistic algorithm that can find the solution of the problem with a small probability. The algorithm will take two steps: This algorithm will randomly take one element from each of sets $Q_1,\ldots, Q_{r-x}$ and sum those elements. Then it will take the sum obtained and search for it in CSum sorted set.
The first algorithm step can be accomplished in constant time that depends only on r. The second algorithm part takes $O(\log N)$ time to search element in sorted database. So the total running time of the algorithm is $O(\log N)$.
However this algorithm will find the solution with probability only $\frac{1}{N^{r-x}}$.
2. To boost the probability of success we use amplitude amplification technique.
We prepare the starting quantum superposition $\sum |Q_1\rangle |Q_2\rangle .. |Q_{r-x}\rangle$ and call the amplitude amplification with algorithm from the step 1 embedded in it.
Then, after $O\left(N^{(r-x)/2}\right)$ steps the algorithm will give an answer with high probability of success. So the running time of quantum part of algorithm is $O\left(N^{(r-x)/2} \log N\right)$.
The total running time of this algorithm is: $O\left(\max\left(N^x \log N, N^{(r-x)/2} \log N\right)\right)$,
so we should minimize this function. This is the case when x = (r-x)/2. So we get the value of x = 1/3r and total algorithm running time $O\left(N^{\lceil r/3 \rceil} \log N\right)$.

Theorem 24. *r-Sum problem has quantum lower bound* $\Omega\left(\sqrt{N}\right)$.

Proof. The proof is the same as for 3-Sum problem.

7 Real-time quantum Turing machine

Here we observe another model of computation called real-time Turing machine

7.1 *Definitions*

Here we'll discuss some definitions that compare real – time Turing machine in deterministic and quantum case. The first definition was taken from Rabin [11] and the second one is investigated by the author and is also based on Rabin and a common pattern of quantum basic constructions (see more in Gruska [4]).

Definition 9.
Real - time deterministic Turing machine (TM) is a set $M = <\Sigma, \Sigma_w, Q, q_0, q_f, I>$, where:

Σ- finite alphabet (of input symbols), including symbols # and $;

Σ_w – finite alphabet (work tape symbols), including symbol λ;

Q – set of states;

q_0 – initial state;

q_f - final state;

$\{\leftarrow,\rightarrow,\downarrow\}$ – movements of the head (left, right, stop);

I – set of instructions. Instruction is a row $\Sigma^*\Sigma_w{}^*Q \rightarrow Q^*\Sigma_w{}^*\{\leftarrow,\rightarrow,\downarrow\}$.

First three symbols of the instruction are called the left side, and the last four are the right side. It may be one and the only one instruction with the same left side in I for every $\Sigma^*\Sigma_w{}^*(Q\backslash\{q_0\})$.

Such a machine has one endless input tape and one endless work tape with one head moving on each tape. At the beginning machine is in the state q_0, the input tape head is on the first symbol of the word from the left. Work tape is empty and TM reads the first symbol of the word from the input tape. As a second step, it reads the second symbol, then the third etc. After the last symbol of the word the machine reads symbols $.

Let it be the instruction of I: $xyq_k \rightarrow q_jz\rightarrow$, where current state is $q_k \in Q$, the machine is reading symbol $x \in \Sigma$ and the work tape head is observing symbol $y \in \Sigma_w$. Then the machine moves to the state q_j, replacing the symbol y with z, and moves to the right.

Real – time TM every moment reads a new symbol. The moment, when the symbol $ has been read, the work is finished.

Real - time TM accepts the word, if the work is finished, the working tape contains one symbol "1", the rest of the tape is filled with "λ" and the head observes the symbol "1". It rejects the word, if the work is finished, the working tape contains one symbol "0", the rest of the tape is filled with "λ" and the head observes the symbol "0". Real - time TM recognizes the language L in time t(x), if for every word $x \in L$ exists a set of instructions from I, that needs not more than t(x) steps to obtain the result 1, and there isn't any word $x \notin L$ that a set of instructions from I for x leads to the result 1.

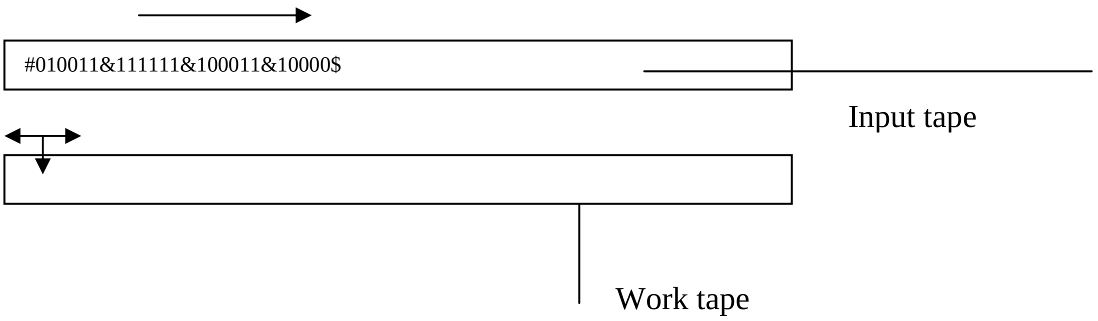

Fig. 3

The definition of real – time quantum Turing machine is a compilation of the quantum Turing machine definition [10] and the real – time Turing machine [11].

Definition 10

<u>Real - time quantum Turing machine</u> (QTM) is a set $M = <\Sigma, \Sigma_w, Q, q_0, q_f, \delta>$, where:

Σ - finite alphabet (of input symbols), including symbols # and $;

Σ_w – finite alphabet (work tape symbols), including symbol λ;

Q – set of states;

q_0 – initial state;

q_f - final state;

transition amplitude mapping $\delta : Q*\Sigma*\Sigma_w*\Sigma_w*Q*\{\leftarrow,\rightarrow,\downarrow\} \rightarrow C_{[0,1]}$ is required to be such, that quantum evolution of M is unitary. That means that quantum evolutions of M can be defined as unitary matrices U, where UU = I and U* is a conjugate* *transpose of U, i.e. the transposition of U and conjugation of its elements, and I is the unit matrix.*

To be quantum Turing machine, it has to meet so-called well-formedness conditions (see more about unitarity conditions in [4]):

Local probability condition.

$$\sum_{\sigma,q,d} \left|\delta(q_1,\sigma_1,q,\sigma,d)\right|^2 = 1$$

Separability condition 1: For any two different pairs q_1, σ_1 un q_2, σ_2

$$\sum_{\sigma,q,d} \delta^*(q_1,\sigma_1,q,\sigma,d)\delta(q_2,\sigma_2,q,\sigma,d) = 0$$

Separability condition 2: For any two different pairs q_1, σ_1, d_1 un q_2, σ_2, d_2

$$\sum_{\sigma,q} \delta^*(q\ ,\sigma\ ,q_1,\sigma_1,d_1)\delta(q\ ,\sigma\ ,q_2,\sigma_2,d_2) = 0$$

Separability condition 3: For any q_1, σ_1, σ_2 un q_2, σ_3, σ_4 un d_1 <> d_2

$$\sum_{q} \delta^*(q_1,\sigma_1,q\ ,\sigma_2,d_1)\delta(q_2,\sigma_3,q,\sigma_4,d_2) = 0$$

Here σ, q, d are the symbol, that is printed out on the work tape, new state of the QTM and the direction of the work tape head movement (←,→,↓).

After each step the measurement is performed. If the result of the measurement is neither positive nor negative, then the measurement doesn't change the state of the machine and the process goes on. As a result of the measurement, the word can be accepted or rejected.

Such a machine has one endless input tape and one endless work tape with one head moving on each tape. At the beginning machine is in the state q_0, work tape is empty, and it reads the first symbol of the word from the input tape. As a second step, it reads the second symbol, then the third etc. After the last symbol of the word the machine reads symbols $.

Let it be δ: $q_k xyz q_j \rightarrow\rightarrow 1/\sqrt{2}$, where current state is $q_k \in Q$, the machine is reading symbol $x \in \Sigma$ and the head is observing symbol $y \in \Sigma_w$. Then the machine moves to the state q_j, replacing the symbol y with z, and moves to the right. Real – time QTM every moment reads a new symbol. The moment, when the symbol $ has been read, the work is finished and the measurement is made to decide, whether the word belongs to the language or not. Real - time QTM recognizes the language L with amplitude Δ ($\Delta > ½$), if M working on any word x with amplitude not less than Δ accepts x, if $x \in L$, and rejects x, if $x \notin L$.

7.2 Deterministic versus Quantum real – time TM

We are going to show that real – time QTM with certain limitations can be more powerful than its classical counterpart. To prove that, we need to show a language, that can be recognized by the first one, but can' t be recognized by the last one. That gives us the following theorem.

Theorem 25.
The language L = {$x\&y\&x^{rev}\&y^{rev}$}, where x, y = {0,1} can be recognized by a real – time QTM.*

Proof. The idea is that the only way to compare both x and x^{rev} and y and y^{rev} is to divide the computational process into branches, that work simultaneously.
The first step is to split the process into three states (see Matrix 1), one of which is rejecting state q_{rr}, and q_1 and q_2 compare x and x^{rev} and y and y^{rev} respectively.
Than the word belongs to the language L, if both the branches q_1 and q_2 say "yes", and doesn't, if any of the branches says "no".
q_0 – initial state;
{q_a, q_{aa}} – a set of accepting states;
{q_r, q_{rr}} – a set of rejecting states.

$$\begin{pmatrix} \#\varepsilon & q_0 & q_1 & q_2 & q_3 & q_4 & q_5 & q_6 & q_7 & q_8 & q_a & q_{aa} & q_r & q_{rr} \\ q_0 & 0 & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{3}} \end{pmatrix}$$

Matrix 1
All the other transitions of this matrix are arbitrary such that the matrix is unitary (see more in [4]).

Then the first branch moves like that:
q_1 – reads x and writes it down to the work tape, moving to the right;
q_3 – waits while y is read;
q_5 – reads x^{rev} and goes to the left the work tape, comparing x and x^{rev};
q_7 – waits while y^{rev} is read.

The second branch:

q_2 – waits while x is read;

q_4 – reads y and writes it down to the work tape, moving to the right;

q_6 – waits while x^{rev} is read;

q_8 – reads y^{rev} and goes to the left the work tape, comparing y and y^{rev}.

If the branch finds the difference between x and x^{rev} or y and y^{rev}, then it goes to one of the rejecting states. Otherwise after the symbol \$ is read it goes to one of the accepting states.

Here are all the matrices that describe the evolution of the QTM. Each matrix is defined by two symbols in the upper left corner. Each matrix describes the transitions, where the first symbol is that one that is read from the input tape, and the second one is read from the work tape. Symbol “ε” means, that it can be any symbol.

$$\left(\begin{array}{cccccccccccccc}
10 & q_0 & q_1 & q_2 & q_3 & q_4 & q_5 & q_6 & q_7 & q_8 & q_a & q_{aa} & q_r & q_{rr} \\
q_0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
q_1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
q_2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
q_3 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
q_4 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
q_5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
q_6 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
q_7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
q_8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
q_a & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
q_{aa} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
q_r & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
q_{rr} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0
\end{array}\right)$$

Matrix 2

If going backward the work tape (states q_5 and q_8) the symbol read from the work tape doesn’t match the symbol read from the input tape, QTM goes to the rejecting states (see Matrix 2). Otherwise QTM doesn’t change its state.

$$\left(\begin{array}{cccccccccccccc}
01 & q_0 & q_1 & q_2 & q_3 & q_4 & q_5 & q_6 & q_7 & q_8 & q_a & q_{aa} & q_r & q_{rr} \\
q_0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
q_1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
q_2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
q_3 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
q_4 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
q_5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
q_6 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
q_7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
q_8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
q_a & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
q_{aa} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
q_r & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
q_{rr} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0
\end{array}\right)$$

Matrix 3

If going backward the work tape (states q_5 and q_8) the symbol read from the work tape doesn't match the symbol read from the input tape, QTM goes to the rejecting states (see Matrix 3). Otherwise QTM doesn't change its state.

$$\left(\begin{array}{cccccccccccccc}
\&\,\varepsilon & q_0 & q_1 & q_2 & q_3 & q_4 & q_5 & q_6 & q_7 & q_8 & q_a & q_{aa} & q_r & q_{rr} \\
q_0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
q_1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
q_2 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
q_3 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
q_4 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
q_5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
q_6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
q_7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
q_8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
q_a & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
q_{aa} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
q_r & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
q_{rr} & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}\right)$$

Matrix 4

When the symbol ‘&’ is read from the input tape, the QTM changes its state (see Matrix 4). That means, that the whole word has been read, and QTM is ready to accept the next word. The branches switch from waiting to moving states and vice versa.

$$\left(\begin{array}{cccccccccccccc}
\&\,\varepsilon & q_0 & q_1 & q_2 & q_3 & q_4 & q_5 & q_6 & q_7 & q_8 & q_a & q_{aa} & q_r & q_{rr} \\
q_0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
q_1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
q_2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
q_3 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
q_4 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
q_5 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
q_6 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
q_7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
q_8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
q_a & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
q_{aa} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
q_r & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
q_{rr} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{array}\right)$$

Matrix 5

When the symbol ‘$’ is read from the input tape, the QTM finishes its work (see Matrix 5). Then the working states (q_7 and q_8) go to the accepting states. For all other states it’s impossible to read symbol ‘$’, so their state changes are arbitrary to meet unitarity criteria.

All the matrices, that don’t change the state of the QTM, look like Matrix 6 and are not shown separately. That is, going backward the work tape, the symbol read from the input tape matches the symbol read from the work tape.

$$\begin{pmatrix} 11 & q_0 & q_1 & q_2 & q_3 & q_4 & q_5 & q_6 & q_7 & q_8 & q_a & q_{aa} & q_r & q_{rr} \\ q_0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ q_1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ q_2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ q_3 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ q_4 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ q_5 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ q_6 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ q_7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ q_8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ q_a & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ q_{aa} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ q_r & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ q_{rr} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Matrix 6

So if the word is from L, then both of the branches say “yes” with probability 2/3;
If the word is not from L, then it is rejected by at least one branch, and in total the word is rejected with probability >= 2/3.

Theorem 26.

Language L = {x&y&x^{rev}&y^{rev}}, where x, y = {0,1}, can’t be recognized by a real – time deterministic Turing machine*

Proof. Our proof is based on Rabin [11] and standard proofs in [29], [30], [31], and uses the idea of bottleneck, though which information can’t flow fast enough.
We’ll observe machine M, which recognizes L in real – time. Number of states is m and number of letters in work alphabet is n.
If w – input sequence, then <u>*work zone*</u> t(w) is the set of the cells of the tape, which have been visited by the machine when input w was read.
If x – sequence of cells or symbols, then l(x) - <u>*length of x*</u> (number of elements).
If x – input sequence, then <u>*code of x*</u> is a sequence of symbols in the work zone t(x), state of M and its position on the tape after x had been read.

Lemma 7.

There is a constant c > 0 such that for every x = {0,1} and every integer i > 0 exists y = {0,1}* such that l(y) = i and ci ≤ l(t(x&y)).*

Proof. There is 2^i sequences y = {0,1}* such that l(y) = i. If $y_1 \neq y_2$ then x&y_1 and x&y_2 must have different codes, otherwise both x&y_1&x^{rev}&y_1^{rev} and x&y_2&x^{rev}&y_1^{rev} will be accepted by M.

Let l(t(x&y)) ≤ k for all y, l(y) = i. Then there are no more than n^k * k * m different codes for input x&y. It means, $2^i \leq n^k$ * k * m. If i is large, then k is large too, so we can assume km ≤ n^k (we assume 2 ≤ n). So, $2^i \leq n^{2k}$, and

$$\frac{1}{2}\frac{\ln 2}{\ln n} i \leq k .$$

We can take $c_1 = \frac{1}{2}\frac{\ln 2}{\ln n}$. This c_1 will work for any i greater than some i_0; for c small enough lemma will work for every i.

Lemma 8.

There is an integer d>0 (which depends only on M) such that for every x and every integer i ≥ l(x) there is a sequence y, l(y) = i, such that (a) ci ≤ l(t(x&y)) and (b) no more than 1/5 of the work zone t(x&y) cells are visited by M more than d times.

Proof. We'll find the sequence y, l(y) = i, where (a) is true. Let d_1 be a number such that more than 1/5 of t(x&y) cells are visited by M more than d_1 times. Then number of M steps is greater than d_1*(1/5)*l(t(x&y)) ≥ (1/5)* $d_1$ci. But M is real – time, so the number of M steps is exactly l(x) + l(y) + 1 ≤ 2i. Then (1/5)* $d_1$ci ≤ 2i and d_1 ≤ 10/c. So, d = [10/c +1] meets (b).

Let x, y = {0,1}*. Cell B ∈ t(x&y) is called <u>*bottleneck cell*</u> of t(x&y), if (1) during input x&y M passes through B no more then d times (where d is from Lemma 2), (2) B is outside work zone t(x), (3) the length of the t(x&y) segment, which is divided by B and doesn't contain t(x), is greater than l(x)+1.

Lemma 9.

For every x there is y such that zone t(x&y) has a bottleneck cell.

Proof. Let i be a natural number, such that $5l(x) + 5 < ci$ and $l(x) \leq i$. By Lemma 2, there is a sequence y such that $ci \leq l(t(x\&y))$ and no more than 1/5 of work zone t(x&y) cells are visited by M more than d times. We get $l(t(x)) \leq l(x) + 1 < ci/5 \leq l(t(x\&y))/5$. Dividing t(x&y) into five equal parts, we can see, that either on the left, or on the right end of t(x&y) there is a segment of length $(2/5)*l(t(x\&y))$, which doesn't contain t(x) cells. In this segment consider 1/5 part of t(x&y), that doesn't contain the end of the zone. As less than 1/5 of t(x&y) cells are visited more than d times, there is a cell B, that is situated in our selected segment, that is visited by M no more than d times. And finally, between B and the end of t(x&y) there are at least $(1/5)*l(t(x\&y)) \geq ci/5 > l(x) + 1$ cells. Thus, B is a bottleneck cell.

Let x, y be such, that zone t(x&y) has a bottleneck cell. Let's assume, that B is to the right of the zone t(x). When input word x&y is being read, the machine comes to the last cell E of the zone t(x&y) for the first time. Let $w \in Z$ be the first segment of sequence y, such that after input x&w machine M comes to E for the first time. Thus t(x&y) and t(x&w) have the same last cell E and B is the bottleneck cell for t(x&w) as well.

Let the first cell to the right from B be R. *M passage through B* will be either movement from B to R, or movement from R to B. *Machine state after passage* is state, to which the machine comes after coming to R (in the first case) or to B (the second case). If the machine begins and finishes the elementary operation in B, we will not consider this as a passage through B.

Let $p_1, p_2, \ldots, p_r$ be sequential passages through B. p_1 means going from B to R, p_2 – from R to B and so on. Let M state during passage p_i be s_i, $1 \leq I \leq r$. The *scheme* in B is a set $(e, s_1, s_2, \ldots, s_r)$ where e = 1, if B is to the right of t(x), or e = -1, if B is to the left of t(x), and $s_1, s_2, \ldots, s_r$ are defined above.

The number of passages through B r is no more than d. Than there are no more than $N = 2m + 2m^2 + \ldots + 2m^d$ different schemes in B, where m is number of states of M.

Let g be a number such that $N < 2^g$. For every x, l(x) = g let y be a sequence such that t(x&y) has a bottleneck cell B_x, and let w be a segment of the input sequence y, after which the cell E is visited for the first time. There must be two different sequences x_1, x_2, $l(x_1) = l(x_2) = g$, such that bottleneck cells B_{x1} and B_{x2} have the same schemes.

Let $x_1\&w_1 = x_1\&\varepsilon_1 \dots \varepsilon_{n1} \dots \varepsilon_{n2} \dots \varepsilon_{nr} \dots \varepsilon_{nr+1}$,

$x_{2\&}w_2 = x_2\&\delta_1 \dots \delta_{m1} \dots \delta_{m2} \dots \delta_{mr} \dots \delta_{mr+1}$,

where $\varepsilon, \delta \in \{0, 1\}$, ε_{n1} – input symbol during first M passage through B_{x1}, ε_{n2} – input symbol during second M passage through B_{x1}, and so on. The same is for δ_{m1}, δ_{m2}, … for the sequence $x_2\&w_2$. After input ε_{nr+1} or δ_{mr+1} M visits cell E_{x1} or E_{x2} respectively. In sequence $x_1\&w_1$ replace for each odd $1 \le I \le r - 2$ segment $\varepsilon_{n_i+1} + 1 ... \varepsilon_{n_{i+1}-1}$ with sequence $\delta_{m_i+1} ... \delta_{m_{i+1}-1}$. Then replace $\varepsilon_{n_r+1} ... \varepsilon_{n_{r+1}}$ with $\delta_{m_r+1} ... \delta_{m_{r+1}}$. We'll call this new sequence $x_1\&w'_1$. We can notice, that all the alterations were made in segment w_1. We get, that $x_1\&w_1$ and $x_2\&w_2$ have the same schemes in B_{x1} and B_{x2}, and our alterations were made only in segments between passages through B_{x1} (when M was to the right of $B_{x1)}$ or after the last passage through B_{x1}. So $x_1\&w'_1$ has the same scheme $(1, s_1, s_2, \dots, s_r)$ and for every input symbol ε_{nj}, where j is odd and $2 \le I \le r + 1$, tape segment to the right of B_{x1} looks the same as the $t(x_2\&w_2)$ tape segment to the right of B_{x2} with input symbol δ_{mj} , and M states are the same on the corresponding inputs. Zones $t(x_1\&w'_1)$ and $t(x_2\&w_2)$ have bottleneck cells B_{x1} and B_{x2} with the following properties:

Zone $t(x_i)$ is to the right from B_{xi}, i=1,2;

The segments of zones $t(x_1\&w'_1)$ and $t(x_2\&w_2)$ which are situated to the right of B_{x1} and B_{x2} has the length greater than $l(x_1) = l(x_2) = g$.

So, after input $x_1\&w'_1$ and $x_2\&w_2$ M is in the last cell E_{x1} and E_{x2}, tape segments between B_{x1} and E_{x1} and between B_{x2} and E_{x2} look the same, states of M in both cases are the same.

Assume, that after both $x_1\&w'_1$ and $x_2\&w_2$ there is input $\&x_{1rev}$, and after that $\&w'_{1rev}$ and $\&w_{2rev}$. As $x_1 \neq x_2$, then after $x_2\&w'_1\&x_{1rev}$ M must not accept the word, no matter what input will follow. But $l(\&x_{1rev}) = g+1$ is less, than the distance between E_{xi} and B_{xi}, I=1,2. As M is real – time and performs no more than 1 movement for every input symbol, after input $\&x_{1rev}$ it will remain to the right of B_{xi}. So, M in both cases starts working in the same state with the same tape. And after input $x_1\&w'_1\&x_{1rev}$ and $x_2\&w_2\&x_{1rev}$ machine also in both cases will be in the same state, and after reading &w'$_{1rev}$ and &w$_{2rev}$ both of the sequences will be either accepted or rejected. It's a contradiction.

8 Conclusion

Main purpose of this paper was to study different quantum computational models. This work was based on the following publications:

1. Oksana Scegulnaja:
   Quantum Real-Time Turing Machine.
   Lecture Notes in Computer Science 2138 Springer 2001, 412-415

2. Aija Bērziņa, Andrej Dubrovsky, Rūsiņš Freivalds, Lelde Lāce, Oksana Scegulnaja: Quantum Query Complexity for Some Graph Problems. Lecture Notes in Computer Science 2932 Springer 2004, 140-150

3. Andrej Dubrovsky and Oksana Scegulnaja **.**
   Quantum algorithms and lower bounds for 3-Sum problem.
   ERATO Conference on Quantum Information Science, 2003,
   September 4-6, 2003, Kyoto, Japan.

4. Andrej Dubrovsky, Oksana Scegulnaja-Dubrovska
   Improved Quantum Lower Bounds for 3-Sum Problem.
   Proceedings of Baltic DB&IS 2004, vol. 2, Riga, Latvia, pp.40-45.

5. Rūsiņš Freivalds, Lelde Lāce, Oksana Scegulnaja-Dubrovska
   “Two lower bounds for quantum query complexity” The 8th
   International conference on Quantum Communication, Measurement and Computing (QCMC), 28th November - 3rd December, 2006
   Tsukuba, Japan.

6. Andrej Dubrovsky and Oksana Schegulnaja-Dubrovska
   Hopcroft's problem in quantum setup
   The Ninth Workshop on Quantum Information Processing Paris, January 16-20, 2006

7. Lelde Lāce, Oksana Ščeguļnaja-Dubrovska. Nondeteministic and postselection quantum query algorithms Proceedings of the 2008 International Conference on Foundations of Computer Science (FCS'2008), p. 102-105.

8. Lelde Lāce, Oksana Ščeguļnaja-Dubrovska, Rūsiņš Freivalds.
   Languages Recognizable by Quantum Finite Automata with cut-point 0.
   SOFSEM 2009, January 24-30, 2009, Špindlerův Mlýn, Czech Republic.

9. Oksana Scegulnaja-Dubrovska, Lelda Lāce and Rūsiņš Freivalds
   Postselection finite quantum automata
   9th international conference on Unconventional computation, Tokyo, Japan, June 21-25, 2010.

10. Oksana Scegulnaja-Dubrovska, and Rūsiņš Freivalds

A context-free language not recognizable by postselection finite quantum automata.

MFCS 2010, August 22, 2010. Brno, Czech Republic.

The next promising task will be study quantum models with postselection and its properties.

9 References

1. Benioff, P.: Quantum mechanical Hamiltonian models of Turing machines.
   J. Stat. Phys. Vol. 29 (1982) 515-546.
2. Feynman, R.: Simulating physics with computers.
   Int. J. Of Theor. Phys. Vol. 21 No. 6/7 (1982) 467-488.
3. Shor, P.: Algorithms for quantum computation: discrete logarithms and factoring.
   In Proc. 35th FOCS (1994) 124 – 134.
4. Gruska, J.: Quantum Computing, McGraw Hill (1999).
5. Scott Aaronson.
   Quantum Computing, Postselection, and Probabilistic Polynomial-Time.
   Proceedings of the Royal Society A, 461(2063):3473-3482, 2005.
   (See also quant-ph/0412187).
6. Paavo Turakainen. On stochastic languages.
   Information and Control, v. 12, No. 4, pp. 304--313, 1968.
7. L. Grover. *A* fast quantum mechanical algorithm for database search.
   Proceedings of the 28th ACM symposium on Theory of Computing, pp 212-219, 1996.
8. Charles H. Bennett, Ethan Bernstein, Gilles Brassard, Umesh V. Vazirani.
   Strengths and Weaknesses of Quantum Computing.
   SIAM Journal on Computing, v. 26, 1997, pp. 1510 –1523.
9. A.Ambainis. *Quantum lower bounds by quantum arguments*.
   Journal of Computer and System Sciences, 64:750-767, 2002.
10. David Deutsch.
    Quantum theory, the Church – Turing principle and the universal quantum computer.
    Proc. Royal Society London, A400, 1989. P. 96-117.
11. Rabin, M.
    Real time computation,
    Israel J. Of Math., 1, #4 (1963), 203 – 211.
12. Hartmanis J., Stearns R. E.
    On the computational complexity of algorithms,
    Trans. Amer. Math. Soc., 117, #5 (1965).
13. Jamada H.

Real – time computation and recursive functions not real – time computable, IRE Trans. El. Computers, EC-11 (1962), 753 – 760.

14. Andris Ambainis and Rūsiņš Freivalds.
1-way quantum finite automata: strengths, weaknesses and generalizations. Proc. FOCS'98, p. 332-- 341.
(See also quant-ph/9802062).
15. Attila Kondacs and John Watrous.
On the power of quantum finite state automata. In Proc. FOCS'97, p. 66--75.
16. C. Moore, J. Crutchfield. Quantum automata and quantum grammars.
Theoretical Computer Science}, 237:275--306, 2000.
Also quant-ph/9707031.
17. Alex Brodsky, Nicholas Pippenger.
Characterizations of 1-way quantum finite automata.
SIAM Journal on Computing, 31(5):1456–1478, 2002.
(See also quant-ph/9903014).
18. Māris Valdats.
The class of languages recognizable by 1-way quantum finite automata is not closed under union.
Proc. Int. Workshop ”Quantum Computation and Learning, Sundbyholm Slott, Sweden, 52–64, 2000.
19. Andris Ambainis, Arnolds Ķikusts.
Exact results for accepting probabilities of quantum automata.
Theoretical Computer Science, 295 (1-3): 3–25, 2003.
20. Andris Ambainis, Ashwin Nayak, Amnon Ta-Shma, Umesh Vazirani.
Quantum dense coding and quantum finite automata.
Journal of ACM, 49:496–511, 2002. Earlier version: STOC’99 and quant-ph/9804043.
21. Farid M. Ablayev, Rūsiņš Freivalds.
Why Sometimes Probabilistic Algorithms Can Be More Effective.
Lecture Notes in Computer Science, 233:1–14, 1986.
22. Abuzer Yakarylmaz and A.C. Cem Say
Probabilistic and quantum finite automata with postselection
23. Abuzer Yakarylmaz and A. C. Cem Say.
Succinctness of two-way probabilistic and quantum finite automata.

Discrete Mathematics and Theoretical Computer Science.
(Also available at arXiv:0903.0050).

24. H. Buhrman and R. de Wolf.
Complexity Measures and Decision Tree Complexity : A Survey.
Theoretical Computer Science, v. 288(1): 21-43 (2002)
25. A.Ambainis. Personal communication. 2003.
26. Gajentaan, A.; Overmars, M. H.:
On a class of $o(n_2)$ problems in computational geometry.
Computational Geometry: Theory and Applications, 5(3):165-185,1995.
27. Scegulnaja, O, Dubrovsky, A.
Quantum algorithms and lower bounds for 3-Sum problem.
In proceedings of EQIS’2003, pages 131-132, 2003.
28. Dobkin, D.P., Lipton, R.J.
On the complexity of computations under varying sets of primitives.
J. Comput. Syst. Sci.,18:86-91, 1979.
29. Hennie, F.
On – line Turing machine computation.
IEEE Trans. Electr. Comp., EC-15:35--44, 1966
30. Hartmanis, J.
Computational complexity of one – tape Turing machine computations.
J. Assoc. Comput. Mach. 15 (1968), 325—339.
31. Freivalds, R.
Complexity of palindromes recognition by Turing machines with an input.
“Algebra i Logika”, 1965, v.4, No. 1, p. 47-58 (in Russian)