University of Latvia
Institute of Mathematics and Computer Science

Vasilijs Kravcevs

Quantum algorithm complexity.

Doctoral Thesis

Area: Computer Science
Sub-Area: Mathematical Foundations of Computer Science

Scientific Advisor:
Dr. habil. math., Prof.
Rūsiņš Freivalds

Riga 2008

Abstract

The speedups of quantum algorithms over classical algorithms have been a main reason for the current interest on quantum computing. The most famous quantum algorithms are: Grover's search algorithm and Shor's factoring algorithm. Grover's algorithm solves an arbitrary search problem with N possibilities in time $\emptyset(\sqrt{N})$. It is obvious that classical algorithm needs $\Omega(N)$ time to solve the problem. So we are very interested in finding all such problems where quantum algorithms can give a better speedup over classical algorithms. The central question of the quantum computing is: how powerful the quantum algorithms are, how big speedup is possible? We give some answers to this question in our thesis.

We are considering two main problems in this thesis. Firstly we show the problem of finding cycle in the graph. We provide the quantum algorithm solving this problem better than any classical analogue and prove that this algorithm is optimal.

In the second part of our work we complete the research of probabilistic reversible automata by investigating properties of probabilistic reversible Decide and Halt automata (DH-PRA). We show general class of languages not recognizable by DH-PRA, that is very similar to the class of languages not recognizable by measure-many quantum automata (MM-QFA). We also show that the class of languages recognized by DH-PRA is not closed under union. That allows us to speculate that that class of languages recognizable by DH-PRA is likely to include the one recognizable by MM-QFA or these classes are equal. We are still working on this problem.

Anotācija

Kvantu algoritmu veiktspējas pārākums par klasiskajiem algoritmiem ir galvenais iemesls pašreizējai lielai ieinteresētībai par kvantu skaitļošanu. Vieni no slavenākiem kvantu algoritmiem ir Grovera meklēšanas algoritms un Šora skaitļu faktorizācijas algoritms. Piemēram, Grovera algoritms atrisina elementa meklēšanas problēmu N elementu sarakstā ar laiku $\emptyset(\sqrt{N})$, kaut klasiskam algoritmam ir nepieciešams $\Omega(N)$ laiks, lai atrisinātu šo problēmu. Tātad mēs esam ļoti ieienteresēti atrast šādas problēmas, kurām kvantu algoritms var būt parāks par klasisko analogu. Kvantu skaitļošanas centrālais jautājums ir, cik spēcīgi ir kvantu algoritmi? Cik liels veiktspējas uzlabojums ir iespējams? Šajā darbā mēs sniedzam dažus jautājumus uz šīm jautājumiem.

Mēs aplūkojam divas lielas problēmas šajā darbā. Pirmkārt, mēs aplūkojam cikla eksistences grafā problēmu. Mēs piedāvājam kvantu vaicājošo algoritmu, kas atrisina šo problēmu efektīvāk par jebkuru klasisko analogu, un pierādām, ka mūsu piedāvātais algoritms ir optimāls.

Otrkārt mēs turpinām pētījumus par varbūtiskiem apgriežamiem automātiem, izpētot apstādināmo varbūtisko apgriežamo automātu (DH-PRA) īpašības. Mēs parādām vispārējo valodu klasi, ko nevar pazīt ar DH-PRA, un kas ir ļoti līdzīga valodu klasei, ko nevar pazīt ar daudz-mērījumu kvantu automātiem (MM-QFA). Mēs arī parādām, ka valodu klase, kuru pazīst DH-PRA nav slēgta pret apvienojuma operāciju. Tas ļauj mums izteikt minējumu, ka valodu klase, ko pazīst DH-PRA automāti iekļauj valodu klasi, ko pazīst MM-QFA vai nu šīs klases ir ekvivalentas. Mēs turpinām strādāt pie šī minējuma pierādīšanas.

Preface

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

1. M. Golovkins, M. Kravtsev, and V. Kravcevs. On a Class of Languages Recognizable by Probabilistic Reversible Decide-and-Halt Automata. *DLT 2007 - 11th International Conference on Developments in Language Theory, Satellite Workshop on Probabilistic and Quantum Automata. Proceedings.*, TUCS General Publication No 45, pp. 37-54, 2007.

2. V. Kravcevs, M. Kravtsev, and M. Golovkins. Closure Properties of Probabilistic Reversible DH Automata. *QCMC 2006 - 8th International Conference on Quantum Communication, Measurement and Computing. Proceedings.*, pp. 117-120, NICT Press, Japan, 2006.

3. M.Golovkins, M. Kravcevs, V. Kravcevs. On the Class of Languages Recognizable by Probabilistic Reversible Decide-and-Halt Automata. Extended Abstract. *5th int. ERATO Conference on Quantum Information Systems. Proceedings, ERATO project*, pp. 131-132, 2005.

4. V. Kravcevs. Quantum query algorithm complexity for graph circuit problem *6th International Baltic Conference on Databases and Information Systems. Proceedings.*, Latvian University Press, 673:66-72, 2004.

5. V. Kravcevs. Boolean function computation by probabilistic and quantum decision trees. *3rd International Workshop on Quantum Computation and Learning. Proceedings, Malardaren University Press, pp. 32-41,2002.*

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

1. 11th International Conference on Developments in Language Theory. Turku, Finland, 2007, July 3 - 6, Presentation ”On the class of languages recognizable by probabilistic decide-and-halt automata”.

2. 8th International Conference on Quantum Communication, Measurement and Computing. Tsukuba, Japan, 2006, November 28 - December 3, Poster ”Closure properties of the Probabilistic Reversible Decide-and-Halt automata”.

3. 5th ERATO Conference on Quantum Information Systems. Tokyo, 2005, August 24-31, Poster ”On DH-Probabilistic reversible automata”.

4. 6th International Baltic Conference on DataBases and Information Systems, Riga, 2004, July 6-9, Presentation ”Quantum query complexity for graph circuit problem”.

5. 7th workshop on Quantum Information Processing, (QIP’2004), Waterloo University, Canada , 2004. January 14-19, Poster ”Some Boolean function computation by probabilistic and quantum decision trees”.

6. Quantum Computation and Learning. 3rd International Workshop. Riga, Latvia, May 25-26, 2002. Presentation ”Boolean function computation by probabilistic and quantum decision trees”.

I thank my co-authors Marats Golovkins, Maksims Kravcevs, who significantly contributed to this research.

I gratefully thank my supervisor Prof. Rūsiņš Freivalds, whose support and ideas were one of the key factors for me to pursue and complete the research.

Contents

Chapter 1

Introduction

The speedups of quantum algorithms over classical algorithms have been a main reason for the current interest on quantum computing. The most famous quantum algorithms are: Grover's search algorithm and Shor's factoring algorithm. Grover's algorithm solves an arbitrary search problem with N possibilities in time $\O(\sqrt{N})$. It is obvious that classical algorithm needs $\Omega(N)$ time to solve the problem. So we are very interested in finding all such problems where quantum algorithms can give a better speedup over classical algorithms. The central question of the quantum computing is: how powerful the quantum algorithms are, how big speedup is possible? We give some answers to this question in our thesis.

We are considering two main problems in this thesis. Firstly we show the problem of finding cycle in the graph. We provide the quantum algorithm solving this problem better than any classical analogue and prove that this algorithm is optimal.

In the second part of our work we complete the research started by M. Golovkins and M. Kravtsev [GK 02] by investigating properties of probabilistic reversible Decide and Halt automata. We have almost completed the proof that probabilistic reversible automata is at least as powerful as quantum automata, but the problem is still open.

In Chapter 1 of this thesis we give some background of quantum computation, including quantum Turing machine, quantum automata and quantum query complexity.

In Chapter 2 of the thesis we state some common notations and definitions used in the rest part of the thesis. In Section 2.1 we recall several notions of linear algebra used in quantum computations. In Sections 2.2 and 2.3 we discuss notions applicable to quantum automata and decision tree models. We also recall several notions from the probability theory in Section 2.4

In Chapter 3 we consider all known quantum query complexity lower bound techniques, paying more attention on Adversary methods. We use this technique to prove lower bounds of graph cycle problem in Section 3.4, showing that our quantum algorithm, that solves the problem, is optimal.

In Chapter 4 we explore properties of the probabilistic reversible Decide and Halt automata. We show a general class of regular languages, not recognizable by DH-PRA. This class is identical to a class not recognizable by MM-QFA [AKV 01] (and similar to the class of languages, not recognizable by C-PRA [GK 02]). We also prove that the class of languages recognizable by DH-PRA is not closed under union. The one open problem still remains: we show the class of languages for which we can not prove whether it is recognizable by DH-PRA or not. So we still unable to prove or disprove that class of languages recognizable by DH-PRA is likely to include the one recognizable by MM-QFA or these classes are equal.

For those interested in quantum computation in general we refer to the monographs of J. Gruska [Gr 99] and M.Nielsen and I.Chuang [NC 00] for the complete overview on the subject.

1.1 Quantum Turing machine

As in the classical theory of computation there are a lot of models of quantum computation. We can consider such models as: quantum Turing machine, quantum circuits, quantum finite automata etc. We refer to the [BV 97] for the description of quantum Turing Machine. Deutch [De 85] introduced the notion of quantum Turing machine in 1985. He gave a precise model of quantum computation and proved that quantum Turing machines compute exactly the same recursive functions as classical deterministic Turing machines do. A quantum Turing Machine as a quantum physical analogue of a probabilistic Turing Machine it has an infinite tape and a transition function, and the actions of the machine are local and completely specified by this transition function. Unlike probabilistic Turing Machines, quantum Turing Machines allow branching with complex "probability amplitudes", but impose the further requirement that the machine's evolution be timereversible. Yao [Y 93] extended the Deutch's results by proving that quantum circuits are polynomially equivalent to quantum Turing machines. Bernstein and Vazirani [BV 97] introduced an efficient universal quantum Turing machine and defined **BQP** - the class of decision problems (languages) that can be solved in polynomial time by quantum Turing machines with error probability bounded by 1/3. **BQP** is the quantum analogy of the **BPP** - the class of decision problems (languages) that can be solved in polynomial time by probabilistic Turing machines with error probability bounded by 1/3 (for all inputs). As in the case with **BPP**, the error probability of **BQP** machines can be made exponentially small [BBBV 97]. In [BV 97] Bernstein and Vazirani proved that $\mathbf{BPP} \subseteq \mathbf{BQP} \subseteq \mathbf{PSPACE}$ [1]. Simon [Si 94] gave the evidence that $\mathbf{BQP} \neq \mathbf{BPP}$, proving the existence of an oracle, relative to which **BQP** cannot even be simulated by probabilistic machines allowed to run for $2^{n/2}$ steps. Finally Bennett, Bernstein, Brassard and Vazirani in [BBBV 97] gave evidence that quantum Turing Machines can not solve every problem in NP in polynomial time. They showed that relative to an oracle chosen uniformly at random, with probability 1, the class NP cannot be solved on a quantum Turing Machine in time $o(2^{n/2})$.

1.2 Quantum automata

1.2.1 Quantum finite automata

Opposite to Turing machine Quantum finite automata represent the cases when specific restrictions are inevitable regarding space consumption and time usage. We can say that Quantum one way finite automaton is the most restricted model of quantum computation, where computation is performed with finite memory and the number of computation steps does not exceed the length of input. The advantage of such model is that such device is relatively simple to be built because the quantum device is controlled by the classical part. It sequentially reads letters of the word from the input and applies certain quantum operations dependant on the letter read on the quantum system. In other words the computation of a word can be represented as consecutive application of the unitary transformations according to the letters of the word on the state of the underlying finite dimensional quantum system.

Several models of quantum finite automata were presented in different papers. Major differences between models are what are the allowed measurements and definition of acceptance. We refer to the type of language recognition with halting before reading the whole input as “decide and halt” and to the respective automata as “decide and halt” automata (DH-automata). We refer to the type of language recognition when no halting states exist and decision on acceptance is taken by the state of automaton after reading the whole word as “classical” acceptance. Such acceptance model is usually used in deterministic or probabilistic automata. We briefly describe the models

[1] thus they established that it will not be possible to conclusively prove that $\mathbf{BPP} \neq \mathbf{BQP}$ without resolving the major open problem $\mathbf{P} \overset{?}{=} \mathbf{PSPACE}$.

connected with topic of research in Probabilistic Reversible Automata in this section.

Moore and Crutchfield in [MC 00] introduced "Measure Once" (MO-QFA) - the most straightforward model of the Quantum finite automata. It makes the only measurement when the whole word is processed obtaining classical result - whether word is accepted or rejected. According to this MO-QFA is pure state model with classical acceptance, as no measurement are allowed at all. Brodsky and Pippenger in [BP 99] proved, that MO-QFA recognize the same language class as permutation automata ([T 68]).

Kondacs and Watrous in [KW 97] proposed another definition of the QFA. Their "Measure-many" quantum finite automata (MM-QFA) differs from the MO-QFA in such way that it allows a special measurement to be performed after each unitary transformation when reading each letter. The measurement provides the probabilistic decision on every letter of the input by projecting the state of the automaton to one of three subspaces: one that corresponds to accepting states of automata, one to the rejecting states and one to the non-halting states. The computation halts when accepting or rejecting state is reached and continues otherwise. Thus MM-QFA has more power in comparison with MO-QFA, but still this model recognizes only the subset of the regular languages ([KW 97]). Moreover the class of languages recognized by MM-QFA is not defined completely. Brodsky and Pippenger in [BP 99] introduced the first "forbidden construction" [2] for languages recognizable by MM-QFA. Later Ambainis, Ķikusts and Valdats in [AKV 01] have shown another "forbidden constructions" for MM-QFA, but still it is open problem whether any language that does not have "forbidden construction" is recognizable by MM-QFA. According to our classification MM-QFA is pure state model with decide and halt acceptance.

Nayak in [N 99] generalized MM-QFA allowing any orthogonal measurement to be performed after reading each letter. Processing of each input letter in this model means applying of finite sequence of unitary transformations and orthogonal measurements, followed by the measurement on fixed subspaces formed by accepting, rejecting and non-halting states as in the MM-QFA case. The automata of that model can recognize only the subset of regular languages anyway. According to our classification Nayak's model is mixed state model with decide and halt acceptance.

In the [ABGKMT 06] Nayak's enhanced QFA model with classical accep-

[2] If minimal deterministic automaton for the language contains such construction then the language can not be recognized by MM-QFA

tance is considered. It is called "latvian QFA" in the paper. This model is similar counterpart for Nayak's model, as MO-QFA for MM-QFA. There are no measurements according to the MM-QFA rules in this model, but any orthogonal measurement as a valid intermediate computational step is allowed in this model. The class of languages recognizable by this model is found. It is expressed in algebras terms but can be expressed in terms of "forbidden constructions" as well.

1.2.2 Probabilistic Reversible Automata

Ambainis and Freivalds in [AF 98] raised the question what kind of probabilistic automata can be considered as a special case of QFA. To answer this question and to study relationship between QFA and probabilistic finite automata (PFA) M.Kravcevs and M.Golovkins in [GK 02] have introduced the model of probabilistic reversible automata (PRA). They noticed that if we consider Nayak's model of QFA with measurement allowed only to subspaces restricted to be one dimensional, then such automaton will be probabilistic automaton which transition matrix being double stochastic[3]. The property of transition matrix to be doubly stochastic is used to define Probabilistic Reversible automata. Note that we can not obtain this model as subclass of quantum automata because not every double stochastic matrix has an unitary prototype.

One way C-PRA

One way C-PRA automata were introduced by M. Golovkins and M. Kravcevs in the [GK 02]. They showed general class of regular languages, not recognizable by C-PRA. For example, such languages as (a,b)*a and a(a,b)* are in this class. This class has strong similarities with the class of languages, not recognizable by DH-QFA [AKV 01]. M. Golovkins and M. Kravcevs expressed this class as a set of forbidden constructions for minimal deterministic automata. There are 2 forbidden constructions first one is exactly the one as for DH-QFA [BP 99], the second one includes the one considered in [AKV 01]. In [GK 02] it was also proved that the class of languages recognized by C-PRA is closed under boolean operations, inverse homomorphisms and word quotient. But also it is not closed under homomorphisms.
Later, in [ABGKMT 06] M. Mercer, D. Thrien e.t.c. have found the class of languages recognizable by C-PRA. This class coincides with the class of the enhanced QFA with classical acceptance. This class expressed in terms

[3]double stochastic mean the sum of elements in every column and row of transition matrix equals to 1

of forbidden constructions for the minimal deterministic automata coincides with the forbidden constructions found in [GK 02].

DH-PRA automata

Properties of DH-PRA model are studied in [GKK 05] and [GKK 07]. Obviously C-PRA recognize proper subset of languages recognizable by DH-PRA, for example C-PRA can not recognize a(a,b)* that can be recognized by DH-PRA.

We prove forbidden constructions for DH-PRA. These constructions are very similar to the constructions for DH-QFA considered in [AKV 01]. In fact it is possible to prove that for a language a minimal automaton contains one of the forbidden constructions for DH-PRA iff it contains one of the forbidden constructions for DH-QFA

Thus we are not able to show the exact class of languages recognizable by DH-PRA. The unknown gap is left for languages which minimal automaton contains forbidden construction for C-PRA but not for DH-PRA.

We show that the class of languages recognized by DH-PRA is not closed under union. This proof uses the same languages and resembles the proof for DH-QFA in [AKV 01].

That leads to a conjecture that class of languages recognizable by DH-PRA is likely to include the one recognizable by DH-QFA or these classes are equal. Still we are unable to prove that.

1.3 Background on quantum query complexity

1.3.1 Background on black-box model

The black-box 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 query. A property of X is any Boolean function that depends on X, i.e. a property is a 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 OR(X) $= x_0 \vee x_1 \vee ... \vee x_{N-1}$. It is well known that the number of queries required to compute OR by any classical (deterministic or probabilistic) algorithm is $\Theta(N)$. Grover [GR 96] 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. This number of queries was shown to be asymptotically optimal [BBBV 97],[BBHT 98], [ZA 97]. Many other quantum algorithms can be naturally expressed in the black-box model, such as an algorithm due to Simon [Si 94], in which one is given a function $f(X) : \{0,1\}^n \rightarrow \{0,1\}^n$, which technically can also be viewed as a black-box X = $(x_0, ..., x_{N-1})$ with N = n^{2n}. The black-box X satisfies a particular promise, and the goal is to determine whether or not X satisfies some other property (the details of the promise and properties are explained in [Si 94]). Simon's quantum algorithm is proven to yield an exponential speed-up over classical algorithms. It makes $(\log N)^{O(1)}$ queries, whereas every classical randomized algorithm for the same function must make $N^{\Omega(1)}$ queries. The promise means that the function $f : 0, 1^N \rightarrow 0, 1$ is partial; it is not defined on all $X \in 0, 1^N$. (In the previous example of OR, the function is total; however, the quantum speed-up is only quadratic.) Some other quantum algorithms that are naturally expressed in the black-box model are described in [DJ 92, BL 95, BBHT 98, BH 97, ME 98, BHT 98, M 98]. Of course, upper bounds in the black-box model immediately yield upper bounds for the circuit description model, in which the function X is succinctly described as a $(\log N)^{O(1)}$ -sized circuit computing x_i from i. On the other hand, lower bounds in the black-box model do not imply lower bounds in the circuit model, though they can provide useful guidance, indicating what certain algorithmic approaches are capable of accomplishing. It is noteworthy that, at present, there is no known algorithm for computing OR (i.e. satisfiability) in the circuit model that is significantly more efficient than using the circuit solely to make queries (though, proving that no better algorithm exists is likely to be difficult, as it would imply P≠NP). It should also be noted that the black-box complexity of a function only considers the number of queries; it does not capture the complexity of the auxiliary computational steps that have to be performed in addition to the queries. In cases such as OR, PARITY, MAJORITY, this auxiliary work is not significantly larger than the number of queries; however, in some cases it may be much larger. For example, consider the case of factoring N-bit integers. The best known algorithms for this involves $\Theta(N)$ queries to determine the integer, followed by $2^{N\Omega(1)}$ operations in the classical case, but only $N^2(\log N)^{O(1)}$ operations in the quantum case [Sh 97]. Thus, the number of queries is apparently not of primary importance in the case of factoring.

We use black-box model proving upper and lower bounds for graph cycle problem.

1.3.2 Background on lower bound methods

Usually we are interested in bound-error computation, where the output is correct with probability at least $\frac{2}{3}$ for all inputs. We use $Q_2(f)$ to denote minimal number of queries for computing f with bound-error. Two main lower bound techniques for proving $Q_2(f)$ are the polynomial method [BBCMW 01] and hybrid/adversary method (sometimes called Ambainis's method) [AM 02],[LM 04],[SS 04].

1.3.3 Lower bounds by polynomials

The polynomial method is used both for proving lower bounds in classical and quantum complexity. Beals, Buhrman and others [BBCMW 01] proved that number of queries $Q_E(f)$ needed to compute a Boolean function f by an exact quantum algorithm is at least $\frac{deg(f)}{2}$, where $deg(f)$ is the degree of multilinear polynomial representing function f. The number of queries $Q_2(f)$ needed to compute a Boolean function f by a two-sided quantum algorithm is at least $\frac{\widetilde{deg(f)}}{2}$, where $\widetilde{deg(f)}$ is the smallest degree of multilinear polynomial approximating f.This reduces proving lower bounds on quantum algorithms to prove lower bounds on degree of polynomials. This is a well-studied mathematical problem with methods from approximation theory [C 66] available. Quantum lower bounds shown by polynomials method include a $Q_2(f) = \Omega(\sqrt[6]{D(f)})$ relation for any total Boolean function f [BBCMW 01], lower bounds on finding mean and median [NW 99], collisions and element distinctness [Ku 03], [AS 04]. Polynomials method is also a key part of $\Omega(\sqrt{N})$ lower bound on set disjointness which resolved a longstanding open problem in quantum communication complexity [RA 03].

1.3.4 Adversary methods

The quantum adversary method runs a quantum algorithm on different inputs from some set. If every input in this set can be changed in many different ways so that the value of the function changes, many queries are needed. The original version of the quantum adversary method, let us call it unweighted, was invented by Ambainis [AM 02]. It was successfully used to obtain the following tight lower bounds: $\Omega(\sqrt{n})$ for Grover search [GR 96], $\Omega(\sqrt{n})$ for two-level And-Or trees (see [HMW 03] for a matching upper bound), and $\Omega(\sqrt{n})$ for inverting a permutation. The method starts with choosing a set of pairs of inputs on which f takes different values. Then the lower bound is determined by some combinatorial properties of the graph of all pairs chosen.

Some functions, such as sorting or ordered search, could not be satisfactorily lower-bounded by the unweighted adversary method. Hoyer, Neerbek, and Shi used a novel argument [HNS 02] to obtain tight bounds for these problems. They weighted the input pairs and obtained the lower bound by evaluating the spectral norm of the Hilbert matrix. Barnum, Saks, and Szegedy proposed a general method [BSS 03] that gives necessary and sufficient conditions for the existence of a quantum query algorithm. They also described a special case, the so-called spectral method, which gives a lower bound in terms of spectral norms of an adversary matrix. Ambainis also published a weighted version of his adversary method [AM 04]. He showed that it is stronger than the unweighted method, and successfully applied it to get a lower bound for several iterated functions. This method is slightly harder to apply, because it requires one to design a so-called weight scheme, which can be seen as a quantum counterpart of the classical hard distribution on the inputs. Zhang observed that Ambainis had generalized his oldest method [AM 02] in two independent ways, so he unified them, and published a strong weighted adversary method [ZH 04]. Finally, Laplante and Magniez used Kolmogorov complexity in an unusual way and described a Kolmogorov complexity method [LM 04].

It was assumed that all adversary methods are related to each other. At first, Laplante and Magniez showed that the Kolmogorov complexity method is at least as strong as all the following methods: the Ambainis unweighted and weighted methods, the strong weighted method and the spectral method. Later Spalek and Szegedy proved that all methods are equivalent.

We use the results obtained by A. Ambainis [AM 02, AM 04] to prove lower complexity bounds for quantum query algorithm that solves graph cycle problem.

Chapter 2

Preliminaries

In this chapter we consider notions, definitions and well-known or elementary facts, referenced directly or indirectly further in the thesis. We refer to [G 02] for the most of definitions.

2.1 Unitary and Stochastic Operations

In this section, we recall well known definitions and theorems from linear algebra. We also consider elementary properties of Doubly Stochastic Matrixes. Some of the theorems are supplied with elementary proofs for the sake of completeness.

Unitary Matrices

As noted in the next sections infinite unitary matrices with finite number of nonzero elements in each row and column describe the work of quantum automata. Further lemmas state some properties of such matrices.

Definition 2.1. *A complex matrix U is called* unitary*, if $UU^* = U^*U = I$.*

Lemma 2.2. *If matrices A and B are unitary, then their direct product is a unitary matrix.*

If U is a finite matrix, then $UU^* = I$ iff $U^*U = I$. However this is not true for infinite matrices:

Example 2.3.

$$U = \begin{pmatrix} \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 & \ldots \\ \frac{1}{\sqrt{2}} & 0 & 0 & 0 & 0 & \ldots \\ 0 & 1 & 0 & 0 & 0 & \ldots \\ 0 & 0 & 1 & 0 & 0 & \ldots \\ 0 & 0 & 0 & 1 & 0 & \ldots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

Here $U^*U = I$ but $UU^* \neq I$.

Lemma 2.4. *If infinite matrices A, B, C have finite number of nonzero elements in each row and column, then their multiplication is associative: $(AB)C = A(BC)$.*

Proof. The element of matrix $(AB)C$ in i-th row and j-th column is $k_{ij} = \sum\limits_{s=1}^{\infty}\sum\limits_{r=1}^{\infty} a_{ir}b_{rs}c_{sj}$. The element of matrix $A(BC)$ in the same row and column is $l_{ij} = \sum\limits_{r=1}^{\infty}\sum\limits_{s=1}^{\infty} a_{ir}b_{rs}c_{sj}$. As in the each row and column of matrices A, B, C there is a finite number of nonzero elements, it is also finite in the given series. Therefore the elements of the series can be rearranged, and $k_{ij} = l_{ij}$. □

Lemma 2.5. *If $U^*U = I$ have finite number of nonzero elements in each row and column, then the norm of any row in the matrix U does not exceed 1.*

Proof. Let us consider the matrix $S = UU^*$. The element of this matrix $s_{ij} = \langle r_j | r_i \rangle$, where r_i is i-th row of the matrix U. Let us consider the matrix $T = S^2$. The diagonal element of this matrix is

$$t_{ii} = \sum_{k=1}^{\infty} s_{ik}s_{ki} = \sum_{k=1}^{\infty} \langle r_k | r_i \rangle \langle r_i | r_k \rangle = \sum_{k=1}^{\infty} |\langle r_k | r_i \rangle|^2.$$

On the other hand, taking into account Lemma 2.4, we get that

$$T = S^2 = (UU^*)(UU^*) = U(U^*U)U^* = UU^* = S.$$

Therefore $t_{ii} = s_{ii} = \langle r_i | r_i \rangle$. It means that

$$\sum_{k=1}^{\infty} |\langle r_k | r_i \rangle|^2 = \langle r_i | r_i \rangle. \tag{2.1}$$

This implies that every element of series (2.1) does not exceed $\langle r_i | r_i \rangle$. Hence $|\langle r_i | r_i \rangle|^2 = \langle r_i | r_i \rangle^2 \leq \langle r_i | r_i \rangle$. The last inequality implies that $0 \leq \langle r_i | r_i \rangle \leq 1$. Therefore $|r_i| \leq 1$. □

Lemma 2.6. *Let us assume that $U^*U = I$. Then the rows of the matrix U are orthogonal iff every row of the matrix has norm* 0 *or* 1*.*

Proof. Let us assume that the rows of the matrix U are orthogonal. Let us consider equation (2.1) from the proof of Lemma 2.5, i.e., $\sum\limits_{k=1}^{\infty} |\langle r_k|r_i\rangle|^2 = \langle r_i|r_i\rangle$. As the rows of the matrix U are orthogonal, $\sum\limits_{k=1}^{\infty} |\langle r_k|r_i\rangle|^2 = |\langle r_i|r_i\rangle|^2$. Hence $\langle r_i|r_i\rangle^2 = \langle r_i|r_i\rangle$, i.e., $\langle r_i|r_i\rangle = 0$ or $\langle r_i|r_i\rangle = 1$. Therefore $|r_i| = 0$ or $|r_i| = 1$.

Let as assume that every row of the matrix has norm 0 or 1. Then $\langle r_i|r_i\rangle^2 = \langle r_i|r_i\rangle$ and in compliance with the equation (2.1), $\sum\limits_{k \in n^+ \setminus \{i\}} |\langle r_k|r_i\rangle|^2 = 0$. This implies that $\forall k \neq i \ |\langle r_k|r_i\rangle| = 0$. Hence the rows of the matrix are orthogonal. □

Lemma 2.7. *The matrix U is unitary iff $U^*U = I$ and its rows have norm 1.*

Proof. Let us assume that the matrix U is unitary. Then in compliance with Definition 2.1, $U^*U = I$ and $UU^* = I$, i.e, the rows of the matrix are orthonormal.

Let us assume that $U^*U = I$ and the rows of the matrix are normalized. Then in compliance with Lemma 2.6 the rows of the matrix are orthogonal. Hence $UU^* = I$ and the matrix is unitary. □

Doubly Stochastic Matrices

These matrixes stand for transition matrixes for Probabilistic Reversible Automata considered in this thesis

Definition 2.8. *A real $(n \times n)$ matrix S, $s_{i,j} \geq 0$, is called* stochastic*, if* $\forall j \ \sum\limits_{i=1}^{n} s_{i,j} = 1$.

Definition 2.9. *A stochastic $n \times n$ matrix D is called* doubly stochastic*, if* $\forall i \ \sum\limits_{j=1}^{n} d_{i,j} = 1$.

Lemma 2.10. *If matrices A and B are doubly stochastic, then their direct product is a doubly stochastic matrix.*

Lemma 2.11. *If A is a doubly stochastic matrix and X - a vector with components $x_i \geq 0$, then* $\max(X) \geq \max(AX)$ *and* $\min(X) \leq \min(AX)$*.*

Proof. Let us consider $X = \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix}$ and $A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}$, where A is doubly stochastic. Let us suppose that $x_j = \max(X)$. For any i, $1 \le i \le n$,

$$x_j = a_{i1}x_j + a_{i2}x_j + \ldots + a_{in}x_j \ge a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n.$$

Therefore x_j is greater or equal than any component of AX. The second inequality is proved in the same way. □

Definition 2.12. *We say that a doubly stochastic matrix S is* unitary stochastic ([MO 79])*, if exists a unitary matrix U such that $\forall i,j\ |u_{i,j}|^2 = s_{i,j}$.*

Remark 2.13. Not every doubly stochastic matrix is unitary stochastic. Such matrix is, for example, $\begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & \frac{1}{2} & \frac{1}{2} \end{pmatrix}$.

2.2 Automata

In this section, we define notions applicable to arbitrary type of automata we will use through out the thesis.

Abstract Automaton

Consider an abstract automaton $A = (Q, \Sigma_1, \ldots, \Sigma_m, q_0, \delta)$, where Q is a finite set of states, Σ_k is an alphabet of the k-th tape, q_0 is the initial state and δ is a transition function.

Each tape is potentially infinite on both directions. The cells of each tape are indexed by numbers in $\mathbb{Z}$. Each cell of the k-th tape stores a symbol in Σ_k or white space, denoted λ. A cell the k-th tape head is above is called *the k-th current cell.* The transition function determines possible transitions of the automaton depending on its current configuration.

Definition 2.14. *A* configuration *of an abstract automaton is $c = (q_i, n_1, \sigma_1, \tau_1, \ldots, n_m, \sigma_m, \tau_m)$, where the automaton is in a state $q_i \in Q$ and $\sigma_k\tau_k \in \Sigma_k^*$ is a finite word on the k-th input tape. The k-th current cell is indexed by n_k and it contains the last symbol of the word σ_k, if $\sigma_k \ne \epsilon$ and λ, otherwise. All cells before or after $\sigma_k\tau_k$ are blank (contain λ).*

The automaton operates in discreet time moments $(t_0, \dots, t_r, \dots)$. If the automaton cannot change contents of a particular tape, it is called *input tape*. Let us assume that the automaton has p input tapes, and renumber the tapes, so that first come input tapes. At the time moment t_0, the automaton is in configuration $(q_0, 0, \epsilon, \tau_1, \dots, 0, \epsilon, \tau_p, 0, \epsilon, \epsilon, \dots, 0, \epsilon, \epsilon)$, where $\tau_1, \dots, \tau_p$ are *input words*. We refer to the input word tuple as *input*. At each time moment, the automaton performs a single transition, called *step*. At each step, depending on its current state and symbols in current cells, the automaton may change its current state, change the contents of current cells, and afterwards, move each tape head one cell forward or backward.

Formally, the transition function δ defines a binary relation ρ from the set $Q \times \Sigma_1 \times \dots \times \Sigma_m$ to the set $Q \times \Sigma_{p+1} \times \dots \times \Sigma_m \times \{\leftarrow, \downarrow, \rightarrow\}^m$. $(q_1, s_1, \dots, s_m)\rho(q_2, s'_{p+1}, \dots, s'_m, d_1, \dots, d_m)$, $d_i \in \{\leftarrow, \downarrow, \rightarrow\}$, means that for the automaton being in the state q_1 and having symbols $s_1, \dots, s_m$ in current cells, the following transition is possible: the automaton goes to the state q_2, writes $s'_{p+1}, \dots, s'_m$ into the current cells of the tapes $p+1, \dots, m$ and moves tape heads according to the directions d_i. If this relation is a function, we speak about *deterministic automata*, other considered possibilities are *probabilistic automata* and *quantum automata*. Probabilistic automata perform transitions with certain probabilities, whereas quantum automata - with certain *amplitudes*.

For technical reasons, we may introduce two categories of white spaces for input tapes, called *end-markers*; one is used before input word and denoted as $\looparrowright$, and the other after input word and denoted as $\looparrowleft$. So every input word is enclosed into end-marker symbols $\looparrowright$ and $\looparrowleft$[1]. Therefore we introduce a *working alphabet* of the k-th input tape as $\Gamma_k = \Sigma_k \cup \{\looparrowright, \looparrowleft\}$. We define the *length of input* as the length of the longest word in the input word tuple (including one end-marker to the left of the word and one to the right of the word).

By $\mathbf{C}$ we denote the set of all configurations of an automaton. This set is countably infinite.

Remark 2.15. It is possible to reach only a finite number of other configurations from a given configuration in one step, all the same, within one step the given configuration is reachable only from a finite number of different configurations.

[1]To get rid of infinite input tapes we may also assume that input tapes are circular and the length of every input tape is $l = \max\limits_{0<k\le p}\{|\tau_k|\} + 2$, so that the next cell after the cell indexed by $l-1$ is the cell indexed by 0. The cells indexed by 0 store $\looparrowright$ and the rest blank cells store $\looparrowleft$.

An abstract automaton introduced above is actually a description of an m-tape Turing machine. To define other types of automata, we apply specific restrictions to this general model. We say that an automaton is *1-way*, if at each step, it must move each input tape head one cell forward. We say that an automaton is *1.5-way*, if at each step, it may not move input tape heads backward. Otherwise, an automaton is called *2-way*. We refer to an automaton as a *finite automaton*, if all of its tapes are input tapes.

To halt computation of the automaton, we may consider at least two options. According to the first option, a subset of $\mathbf{C}$ is introduced and configurations in the subset are marked as *halting* configurations. We monitor the computation of the automaton and stop the computation as soon as the automaton enters a halting configuration. According to the second option, we determine the number of steps of computation in advance, and run the automaton the specified number of steps. In particular, when the number of steps is equal to the length of input, we get *real-time* automata.

Word Acceptance

We study automata in terms of formal languages they recognize. At least two definitions exist, how to interpret word acceptance, and hence, language recognition, for automata.

Definition 2.16. "Decide and halt" acceptance. *Consider an automaton with the set of configurations partitioned into non-halting configurations and halting configurations, where halting configurations are further classified as accepting configurations and rejecting configurations. We say that an automaton accepts (rejects) an input in a decide-and-halt manner, if the following conditions hold:*

- *the computation is halted as soon as the automaton enters a halting configuration;*
- *if the automaton enters an accepting configuration, the input is accepted;*
- *if the automaton enters a rejecting configuration, the input is rejected.*

We refer to the decide-and-halt automata as DH-automata further in the thesis. In case of real-time automata, we may use the following definition.

Definition 2.17. Classical acceptance. *Consider an automaton with the set of configurations partitioned into accepting configurations and rejecting configurations. We say that an automaton accepts (rejects) an input classically, if the following conditions hold:*

- *the computation is halted as soon as the number of computation steps is equal to the length of input;*
- *if the automaton has entered an accepting configuration when halted, the input is accepted;*
- *if the automaton has entered a rejecting configuration when halted, the input is rejected.*

We refer to the classical acceptance automata as classical automata or C-automata further in the thesis.

The both definitions generally are not equivalent.

Language Recognition

Having defined word acceptance, we define language recognition in an equivalent way as in [R 63].

By $p_{x,A}$ we denote the probability that an input x is accepted by an automaton A.

Furthermore, we denote $P_L = \{p_{x,A} \mid x \in L\}$, $\overline{P_L} = \{p_{x,A} \mid x \notin L\}$, $p_1 = \sup \overline{P_L}$, $p_2 = \inf P_L$.

Definition 2.18. *We say that an automaton A recognizes a language L with interval (p_1, p_2), if $p_1 \leq p_2$ and $P_L \cap \overline{P_L} = \emptyset$.*

Definition 2.19. *We say that an automaton A recognizes a language L with bounded error and interval (p_1, p_2), if $p_1 < p_2$.*

We consider only bounded error language recognition in this thesis.

Definition 2.20. *An automaton recognizes a language with probability p if the automaton recognizes the language with interval $(1-p, p)$.*

Definition 2.21. *We say that a language is recognized by some class of automata with probability $1-\varepsilon$, if for every $\varepsilon > 0$ there exists an automaton in the class which recognizes the language with interval $(\varepsilon_1, 1-\varepsilon_2)$, where $\varepsilon_1, \varepsilon_2 \leq \varepsilon$.*

Quantum Automata

In case of a quantum automaton, the transition function is

$$\delta : (Q \times \Sigma_1 \times \ldots \times \Sigma_m) \times (Q \times \Sigma_{p+1} \times \ldots \times \Sigma_m \times \{\leftarrow, \downarrow, \rightarrow\}^m) \longrightarrow \mathbb{C}_{[0,1]}.$$

On each computation step, the quantum automaton is in quantum superposition of configurations[2] $|\psi\rangle = \sum_{c \in \mathbf{C}} \alpha_c |c\rangle$, where $\sum_{c \in \mathbf{C}} |\alpha_c|^2 = 1$ and $\alpha_c \in \mathbb{C}$ is the amplitude of a configuration $|c\rangle$. Every configuration $|c\rangle \in \mathbf{C}$ is a basis vector in the Hilbert space H, determined by $l_2(\mathbf{C})$. Every quantum automaton defines a linear operator (evolution) over this Hilbert space. Due to the laws of quantum mechanics, this operator must be unitary. Although evolution operator matrix is infinite, by Remark 2.15 it has a finite number of nonzero elements in each row and column, therefore it is possible to derive necessary and sufficient conditions, i.e., *well-formedness conditions* to check unitarity for each particular automata type.

General measurements. After each step, a *measurement* is applied to the current quantum superposition of configurations. A measurement is defined as follows. We introduce a set partition of $\mathbf{C}$ as $\{\mathbf{C_1}, \mathbf{C_2}, \ldots, \mathbf{C_z}\}$. So $\bigcup_{0<i\leq z} \mathbf{C_i} = \mathbf{C}$ and if $i \neq j$ then $\mathbf{C_i} \cap \mathbf{C_j} = \emptyset$. $E_1, E_2, \ldots, E_z$ are subspaces of H spanned by $\mathbf{C_1}, \mathbf{C_2}, \ldots, \mathbf{C_z}$, respectively. We use the observable $\mathcal{O}_1$ that corresponds to the orthogonal decomposition $H = E_1 \oplus E_2 \oplus \ldots \oplus E_z$. If the quantum superposition before the observation is $\sum_{c \in \mathbf{C}} \alpha_c |c\rangle$, with probability $p_i = \sum_{c \in \mathbf{C_i}} |\alpha_c|^2$ the outcome of the observation is $|\psi_i\rangle = \frac{1}{\sqrt{p_i}} \sum_{c \in \mathbf{C_i}} \alpha_c |c\rangle$. Hence the total outcome of the observation is a *mixed state* $\sum_{i=1}^{z} p_i |\psi_i\rangle\langle\psi_i|$.

If $z = 1$, we get quantum automata with pure states, otherwise we generally have quantum automata with mixed states. We get other marginal case, when $\mathbf{C}$ is set partitioned into infinitely many subsets, with a single configuration in each subset[3]. In that case, the resulting quantum automaton is a special kind of a probabilistic automaton. See the next subsection for further details.

Word acceptance measurements. Another type of measurement is applied to the quantum automaton to facilitate language recognition.

Decide-and-halt acceptance. We have to monitor when the quantum automaton enters a halting configuration. Hence we perform the following measurement after each step. We partition $\mathbf{C}$ as $\mathbf{C_a}$, $\mathbf{C_r}$ and $\mathbf{C_{non}}$, i.e., accepting, rejecting and non-halting configurations. E_a, E_r and E_{non} are subspaces of H spanned by $\mathbf{C_a}$, $\mathbf{C_r}$, and $\mathbf{C_{non}}$, respectively. We use the observable $\mathcal{O}_2$ that corresponds to the orthogonal decomposition $H = E_a \oplus E_r \oplus E_{non}$. The

[2]More precisely, the automaton with certain probabilities is one of several possible quantum superpositions, or in a mixed state.

[3]By Remark 2.15, on each computation step the number of configurations in a quantum superposition is finite, so on each step it is possible to make the corresponding measurement actually using some finite partition of $\mathbf{C}$.

outcome of each observation is either "accept" or "reject" or "continue". If the quantum superposition before the observation is $\sum\limits_{c\in\mathbf{C}} \alpha_c|c\rangle$, with probability $p_a = \sum\limits_{c\in\mathbf{C_a}} |\alpha_c|^2$ the input is accepted, with probability $p_r = \sum\limits_{c\in\mathbf{C_r}} |\alpha_c|^2$ the input is rejected, and with probability $p_{non} = \sum\limits_{c\in\mathbf{C_{non}}} |\alpha_c|^2$ the automaton is in the quantum superposition of non-halting states $|\psi\rangle = \frac{1}{\sqrt{p_{non}}} \sum\limits_{c\in\mathbf{C_{non}}} \alpha_c|c\rangle$.

Classical acceptance. After the computation is halted, we have to determine, whether the automaton has entered accepting or rejecting configuration. We partition $\mathbf{C}$ as $\mathbf{C_{acc}}$ and $\mathbf{C_{rej}}$, i.e., accepting and rejecting configurations. E_{acc}, E_{rej} are subspaces of H spanned by $\mathbf{C_{acc}}$ and $\mathbf{C_{rej}}$, respectively. We use the observable $\mathcal{O}_3$ that corresponds to the orthogonal decomposition $H = E_{acc} \oplus E_{rej}$. The outcome of the observation is either "accept" or "reject". If the quantum superposition before the observation is $\sum\limits_{c\in\mathbf{C}} \alpha_c|c\rangle$, with probability $p_{acc} = \sum\limits_{c\in\mathbf{C_{acc}}} |\alpha_c|^2$ the input is accepted and with probability $p_{rej} = \sum\limits_{c\in\mathbf{C_{rej}}} |\alpha_c|^2$ the input is rejected.

In case both general measurement and word acceptance measurement have to be performed in a single step, it is easy to see that the order of measurements is irrelevant, actually both measurements may be combined into a single measurement after each step.

Putting things together, each computation step consists of two parts. At first the unitary evolution operator is applied to the current quantum superposition and then the appropriate measurements are applied, using observables as defined above.

Probabilistic Reversible Automata

Let us consider A. Nayak's model of quantum automata with mixed states, [N 99]. A variety of this model for arbitrary type of automata was considered in the previous subsection. (The difference is that Nayak's model allows a fixed sequence of unitary transformations and subsequent measurements after each step.) As noted there, if a result of every observation is a single configuration, not a superposition of configurations, we actually get a probabilistic automaton. However, the following property applies to such probabilistic automata - their evolution matrices are *doubly* stochastic.

The transition function is

$$\delta : (Q \times \Sigma_1 \times \ldots \times \Sigma_m) \times (Q \times \Sigma_{p+1} \times \ldots \times \Sigma_m \times \{\leftarrow, \downarrow, \rightarrow\}^m) \longrightarrow \mathbb{R}_{[0,1]}.$$

After its every step, the probabilistic automaton is in some probability distribution $p_0c_0 + p_1c_1 + \ldots + p_zc_z$, where $p_0 + p_1 + \ldots + p_z = 1$. Such probability distribution is called a superposition of configurations.

A linear closure of $\mathbf{C}$ forms a linear space, where every configuration can be viewed as a basis vector. This basis is called a *canonical basis*. Every probabilistic automaton defines a linear operator (evolution) over this linear space. The corresponding evolution matrix must be doubly stochastic. So we give the following definition for probabilistic reversible automata:

Definition 2.22. *A probabilistic automaton is called* reversible *if its evolution is described by a doubly stochastic matrix, using canonical basis.*

If the evolution of a probabilistic reversible automaton is described by *unitary stochastic* matrix (see Definition 2.12), the automaton can be viewed as a special case of a quantum automaton with mixed states.

It is necessary to note that in [AF 98], A. Ambainis and R. Freivalds proposed a more restricted notion of probabilistic reversibility. For example, they remarked that for the language $L = \{a^{2n+3} | n \in \mathbb{N}\}$, not recognizable by a 1-way deterministic reversible finite automata, there exists a 1-way probabilistic reversible finite automaton which recognizes the language. Consequently, this restricted notion was used in [YKTI 00]. That model is actually a restricted special case of probabilistic reversible DH-automata, as defined in the thesis.

Deterministic Reversible Automata

Deterministic reversible automata can be viewed both as a special case of quantum automata or as a special case of probabilistic reversible automata. The transition function is

$$\delta : (Q \times \Sigma_1 \times \ldots \times \Sigma_m) \times (Q \times \Sigma_{p+1} \times \ldots \times \Sigma_m \times \{\leftarrow, \downarrow, \rightarrow\}^m) \longrightarrow \{0, 1\}.$$

Automata Notations

We regard quantum automata, probabilistic reversible automata and deterministic reversible automata as reversible automata. Refering to different types of automata, we shall use the following notation:

$$[\text{C}|\text{DH-}]\langle\text{automata type}\rangle[\text{-P}|\text{M}].$$

C refers to “classical”, whereas DH refers to “decide-and-halt”. Notations P and M are used in the case of quantum automata. P denotes an automaton with pure states, whereas M - an automaton with mixed states.

For example, C-QFA-M are quantum finite automata with mixed states, using classical definition of language recognition.

2.3 Boolean function query complexity

In this section, we recall well known definitions for black box computational model in deterministic and quantum cases. A good survey of this model is given in [BW 02].

Consider a Boolean function $f : 0, 1^n \to 0, 1$. We are equipped with an oracle that provide us information about variables of function's input vector $X = (x_0, x_1, ...x_{n-1})$. At one moment of time we can receive from the oracle value of the one x_i from X. Such call to oracle is called a query. We are interested to compute function f making as few number of query as possible. The classical algorithm that computes f using oracle queries is called a decision tree, because it can be represented as a binary tree where each vertex represent call for the oracle.

2.3.1 Deterministic decision tree complexity

A deterministic decision tree is a rooted ordered binary tree T. Each internal node of T is labeled with a variable x_i, and each leaf is labeled with a value 0 or 1. Given an input $X \in \{0, 1\}^N$, the tree is evaluated as follows. Start at the root, if this is a leaf, then stop. Otherwise, query the variable x_i that labels the root. If $x_i = 0$ then recursively evaluate the left subtree, if x_i=1 then recursively evaluate the right subtree. The output of the tree is the value (0 or 1) of the leaf that is reached eventually. Note that the input X deterministically determines the leaf, thus the output, that the procedure ends up in.

Definition 2.23. *The decision tree computes Boolean function $f : 0, 1^n \to 0, 1$ if for each input tuple $X = (x_0, x_1, ..., x_{n-1})$, computation goes to accepting leaf if $f(X) = 1$ and the computation ends in the rejecting leaf, if $f(X) = 0$.*

Definition 2.24. *Decision tree computes Boolean function $f(X)$ with complexity k if k is the number of oracle queries in worth case (k is the depth of decision tree).*

Definition 2.25. *The decision tree complexity $D(F)$ of the Boolean function $f(X)$ is complexity of the best decision tree that computes $f(X)$.*

2.3.2 Randomized decision tree

As in many other models of computation we can add the power of randomization to the decision trees. There are two ways to view a randomized decision tree. Firstly we can add (possibly biased) coin flips as internal nodes to the tree. That is, the tree may contain internal nodes labelled by a bias $p \in [0,1]$, and when the evaluation procedure reaches such a node, it will flip a coin with bias p and will go to the left child on outcome 'heads' and to the right child on 'tails'. Now an input X no longer determines with certainty which leaf of the tree will be reached, but instead induces a probability distribution over the set of all leaves. Thus the tree outputs 0 or 1 with certain probability. The complexity of the tree is the number of queries on the worst-case input and the worst-case outcome of the coin flips. A second way to define a randomized decision tree is as a probability distribution μ over deterministic decision trees. The tree is evaluated by choosing a deterministic decision tree according to μ, which is then evaluated as before. The complexity of the randomized tree in this second definition is the depth of the deepest T that has $\mu(T) \geq 0$. Obviously these two definitions are equivalent.

Definition 2.26. *A randomized decision tree computes f with bounded-error if its output equals $f(x)$ with probability at least $2/3$ for all $X \in \{0,1\}^N$. $R_2(f)$ denotes the complexity of the optimal randomized decision tree that computes f with bounded error.*

2.3.3 Quantum decision tree

The quantum decision tree (or quantum circuit) is a quantum analogue of the classical decision tree where each oracle query and other operations are made in quantum superposition. Formally it can be described in terms of quantum circuits.

Consider Boolean function $f : 0,1^n \rightarrow 0,1$. If we have a quantum circuit U that works on N-bit input data, using M extra bits. We can rewrite this circuit as:

$$U|x\rangle|0^M\rangle = \sum_{y,b} \alpha_{y,b}|y\rangle|b\rangle$$

where x is N-bit string $0,1^N$ and b is arbitrary bit: b$\in 0,1$.

Definition 2.27. *The quantum circuit U exactly computes Boolean function $f(X)$,iff*

$$\sum_{y} \sum_{b=f(X)} |\alpha_{y,b}|^2 = 1$$

(It means that outcome bit of the measurement is equal with $f(X)$)

Definition 2.28. *The quantum circuit U computes Boolean function $f(X)$ with bound error δ,iff*

$$\sum_{y} \sum_{b=f(X)} |\alpha_{y,b}|^2 = 1 - \delta$$

(It means that outcome bit of the measurement is equal with $f(X)$ with probability at least $1-\delta$). If for each X we define unitary transformation O_X such that: $O_X|i\rangle|b\rangle$=$|i\rangle|b \oplus x_i\rangle$. Then we can imagine this unitary transformation as oracle, because it returns the bits of input data. (Sometimes it is better to use such transformation O_x: $O_x|i,b\rangle$=$(-1)^{bx}|i,b\rangle$ both transformations are equivalent, as each can be simulated by another).

Then quantum decision tree with T queries can be represented as quantum circuit:

$$U = U_T O_X U_{T-1} O_X ... O_X U_1 O_X U_0,$$

where U_i is arbitrary unitary transformations and O_X is unitary transformation that depends on input data X. The computation begins in the state $|0\rangle$ then we sequentially apply transformations: U_0,O_X...U_T and finally make a measurement of the final state. The outcome of the algorithm is the rightmost bit of the measurement. If the result of the algorithm is equal with $f(X)$ for every X then quantum decision tree exactly computes $f(X)$. If for each X the outcome of the algorithm equals with $f(X)$ with probability at least $frac23$ then we say that algorithm computes $f(X)$ with bound-error.

The complexity of the quantum algorithm that computes $f(X)$ is equal with the $\max_X(|O_X|)$. The quantum query complexity for Boolean function $f(X)$ is the complexity of the best quantum algorithm that computes $f(X)$. We denote $Q_E(f)$ for the query complexity of quantum algorithms that computes $f(X)$ exactly, and $Q_2(f)$ for the query complexity for quantum algorithms that computes $f(X)$ with bounded-error. Usually we are interested in bound-error computation, where the output is correct with probability at least $\frac{2}{3}$ for all inputs. Note that we just count the number of queries, not the complexity of the U_i.

Unlike the classical deterministic or randomized decision trees, the quantum algorithms are not really trees anymore (the names 'quantum query algorithm' or 'quantum black-box algorithm' are also in use). Nevertheless we prefer the term 'quantum decision tree', because such quantum algorithms generalizes classical trees in the sense that they can simulate them, as sketched below. Consider a T-query deterministic decision tree. It first determines which variable it will query initially. Then it determines the next query depending upon its history, and so on for T. Eventually it outputs an output-bit depending on its total history. The basis state of the corresponding quantum algorithm have the form $|i, b, h, a\rangle$ where i, b is the query-part,

h ranges over all possible histories of the classical computation (this history includes all previous queries and their answers), and a is the rightmost qubit, which will eventually contain the output. Let U_0 maps the initial state $\left|\overrightarrow{0}, 0, \overrightarrow{0}, 0\right\rangle$ to $\left|i, 0, \overrightarrow{0}, 0\right\rangle$, where x_i is the first variable that the classical tree would query. Now the classical algorithm applies O, which turns the state into $\left|i, x_i, \overrightarrow{0}, 0\right\rangle$. Then the algorithm applies a transformation U_1 that maps $\left|i, x_i, \overrightarrow{0}, 0\right\rangle$ to $|j, 0, h, 0\rangle$, where h is the new history (which includes i and x_i) and x_j is the variable that classical tree would query given the outcome of the previous query. Then the quantum algorithm applies O for the second time, it applies the transformation U_2 that updates the workspace and determines the next query e.t.c. Finally, after T queries the quantum tree sets the answer bit to 0 or 1 depending on its total history. All operations U_i performed here are injective mappings from basis states to basis states, which are unitary transformations. Thus a T-query deterministic decision tree can be simulated by an exact T-query quantum algorithm. Similarly (basically because a superposition can 'simulate' a probability distribution) a T-query randomized decision tree can be simulated by a T-query quantum decision tree with the same error probability. Accordingly, we have $Q_2(f) \leq R_2(f) \leq D(f) \leq n$ and $Q_2(f) \leq Q_E(f) \leq D(f) \leq n$ for all f.

2.4 Markov Chains

We recall several definitions from the theory of finite Markov chains ([KS 76], etc.) used in this thesis when describing behavior of PRA.
A Markov chain with n states can be determined by an $n \times n$ stochastic matrix A, i.e., matrix, where the sum of elements of every column in the matrix is 1. If $A_{i,j} = p > 0$, it means that a state q_i is accessible from a state q_j with a positive probability p in one step.

2.4.1 Classification of states

Definition 2.29. *A state q_j is accessible from q_i (denoted $q_i \rightarrow q_j$) if there is a positive probability to get from q_i to q_j in 1 or more steps.*

Note that some authors consider zero steps are valid for this definition that means $q_i \rightarrow q_i$ for any i.

Definition 2.30. *States q_i and q_j communicate (denoted $q_i \leftrightarrow q_j$) if $q_i \rightarrow q_j$ and $q_j \rightarrow q_i$.*

For accessibility or communication in one step we will put the corresponding matrix above the symbol. Example: $q_i \xrightarrow{A} q_j$ means there is a positive probability to get from q_i to q_j in 1 step. Or the same $A_{j,i}$¿0. When working with automata we will use sometimes input words instead of their matrixes.

Definition 2.31. *A state q is called recurrent if $\forall i \ q \to q_i \Rightarrow q_i \to q$. Otherwise the state is called transient.*

There several different definitions for transient states proven to be equivalent to the above, important for us is

Definition 2.32. *A state q_i is called transient iff $\sum\limits_{n\to\infty} (A^n)_{i,i} < \infty$*

Definition 2.33. *A state q is called absorbing if there is a zero probability of exiting from this state.*

Definition 2.34. *A Markov chain without transient states is called irreducible if for all q_i, q_j $q_i \leftrightarrow q_j$. Otherwise the chain without transient states is called reducible.*

Definition 2.35. *The period of an recurrent state $q_i \in Q$ of a Markov chain with a matrix A is defined as $d(q_i) = \gcd\{n > 0 \mid (A^n)_{i,i} > 0\}$.*

Definition 2.36. *An recurrent state q_i is called aperiodic if $d(q_i) = 1$. Otherwise the recurrent state is called periodic.*

Definition 2.37. *A Markov chain without transient states is called aperiodic if all its states are aperiodic. Otherwise the chain without transient states is called periodic.*

Definition 2.38. *Markov chain is called absorbing iff that contains at least one absorbing state, and for any non-absorbing state q_i there is an absorbing state that is accessible from q_i. Thus the states of absorbing Markov Chain can be numbered so that transition matrix A has a form*

$$\begin{matrix} A & O \\ B & I \end{matrix}.$$

where I - unit matrix, O - all zero matrix

Definition 2.39. *A probability distribution X of a Markov chain with a matrix A is called stationary, if $AX = X$.*

2.4.2 Behaviour of Markov chains

We recall the following theorem from the theory of finite Markov chains about stationary distribution:

Theorem 2.40. *If a Markov chain with a matrix A is irreducible and aperiodic, then*
a) it has a unique stationary distribution Z;
b) $\lim_{n\to\infty} A^n = (Z, \dots, Z)$;
c) $\forall X \lim_{n\to\infty} A^n X = Z$.

We recall the following fact regarding transient states of a Markov Chain

Theorem 2.41. *Given Markov chain with matrix A and transient state q_i, for matrix A^n when $n \to \infty$ $a^n_{ij} \to 0$ for any j*

Proof. from Definition 2.32 □

2.4.3 Doubly Stochastic Markov Chains

The notion again is introduced according to the needs of PRA.

Definition 2.42. *A Markov chain is called doubly stochastic, if its transition matrix is a doubly stochastic matrix.*

Corollary 2.43. *If a doubly stochastic Markov chain with an $m \times m$ matrix A is irreducible and aperiodic,*

a) $\lim_{n\to\infty} A^n = \begin{pmatrix} \frac{1}{m} & \dots & \frac{1}{m} \\ \dots & \dots & \dots \\ \frac{1}{m} & \dots & \frac{1}{m} \end{pmatrix}$*;*

b) $\forall X \lim_{n\to\infty} A^n X = \begin{pmatrix} \frac{1}{m} \\ \dots \\ \frac{1}{m} \end{pmatrix}$.

Proof. By Theorem 2.40. □

Lemma 2.44. *If M is a doubly stochastic Markov chain with a matrix A, then $\forall q\ q \to q$.*

Proof. Assume existence of q_0 such that q_0 is not accesible from itself. Let $Q_{q_0} = \{q_i \mid q_0 \to q_i\} = \{q_1, \dots, q_k\}$. Q_{q_0} is not empty set. Consider those rows and columns of A, which are indexed by states in Q_{q_0}. These rows and columns form a submatrix A'. Each column j of A' must include all non-zero elements of the corresponding column of A as those states are accesible from the state q_j, hence also from q_0 and are in Q_{q_0}. Therefore $\forall j,\ 1 \leq j \leq k$, $\sum\limits_{i=1}^{k} A'_{i,j} = 1$ and $\sum\limits_{1\leq i,j\leq k} A'_{i,j} = k$. On the other hand, since $q_0 \notin Q_{q_0}$, a row of A' indexed by a state accesible in one step from q_0 does not include all nonzero elements. Since A is doubly stochastic, $\exists i, 1 \leq i \leq k,\ \sum\limits_{j=1}^{k} A'_{i,j} < 1$ and $\sum\limits_{1\leq i,j\leq k} A'_{i,j} < k$. This is a contradiction. □

Corollary 2.45. *Suppose A is a doubly stochastic matrix. Then exists $k > 0$, such that $\forall i\ (A^k)_{i,i} > 0$.*

Proof. Consider an $m \times m$ doubly stochastic matrix A. By Lemma 2.44, $\forall i$ $\exists n_i > 0\ (A^{n_i})_{i,i} > 0$. Take $n = \prod\limits_{s=1}^{m} n_s$. For every i, $(A^n)_{i,i} > 0$. □

Lemma 2.46. *If M is a doubly stochastic Markov chain with a matrix A, then $\forall q_a, q_b\ A_{b,a} > 0 \Rightarrow q_b \rightarrow q_a$.*

Proof. $A_{b,a} > 0$ means that q_b is accesible from q_a in one step. We have to prove, that $q_b \rightarrow q_a$. Assume from the contrary, that q_a is not accesible from q_b. Let $Q_{q_b} = \{q_i \mid q_b \rightarrow q_i\} = \{q_1, q_2, \ldots, q_k\}$. By Lemma 2.44, $q_b \in Q_{q_b}$. As in proof of Lemma 2.44, consider a matrix A', which is a submatrix of A and whose rows and columns are indexed by states in Q_{q_b}. Each column j has to include all nonzero elements of the corresponding column of A. Therefore $\forall j,\ 1 \leq j \leq k,\ \sum\limits_{i=1}^{k} A'_{i,j} = 1$ and $\sum\limits_{1\leq i,j\leq k} A'_{i,j} = k$. On the other hand, $A_{b,a} > 0$ and $q_a \notin Q_{q_b}$, therefore a row of A' indexed by q_b does not include all nonzero elements. Since A is doubly stochastic, $\sum\limits_{j=1}^{k} A'_{b,j} < 1$ and $\sum\limits_{1\leq i,j\leq k} A'_{i,j} < k$. This is a contradiction. □

Corollary 2.47. *If M is a doubly stochastic Markov chain and $q_a \rightarrow q_b$, then $q_a \leftrightarrow q_b$.*

Proof. If $q_a \rightarrow q_b$ then exists a sequence $q_{i_1}, q_{i_2}, \ldots, q_{i_k}$, such that $A_{i_1,a} > 0,\ A_{i_2,i_1} > 0, \ldots, A_{i_k,i_{k-1}} > 0,\ A_{b,i_k} > 0$. By Lemma 2.46, we get $q_b \rightarrow q_{i_k}$, $q_{i_k} \rightarrow q_{i_{k-1}}$, …, $q_{i_2} \rightarrow q_{i_1}$, $q_{i_1} \rightarrow q_a$. Therefore $q_b \rightarrow q_a$. □

Lemma 2.48. *Suppose A is a doubly stochastic matrix and $k > 0$, such that $\forall i\ (A^k)_{i,i} > 0$. Then exist $m > 0$ such that for all pairs $q_i\ q_j$ where $q_i \rightarrow q_j$ for A^k, $q_i \rightarrow q_j$ in one step for $(A^k * m)$*

Proof. Assume $q_i \rightarrow q_j$ in x steps, as according to Lemma 2.45 $q_i \rightarrow q_i$ in one step, $q_i \rightarrow q_j$ in x+1 step as well. For any pair of states q_i q_j where $q_i \rightarrow q_j$ for A^k, q_j is accessible in less then n number of rows in A steps. Thus $m = n$ gives the necessary constant. □

Corollary 2.49. *Communication is a class property for states of doubly stochastic Markov chains.* [4]

Proof. .

- reflexive - $\forall i q_i \leftrightarrow q_i$ by Lemma 2.44
- symmetric - If $q_i \leftrightarrow q_j$ then $q_j \leftrightarrow q_i$
- transitive - If $q_i \leftrightarrow q_j$ and $q_j \leftrightarrow q_k$ then $q_i \leftrightarrow q_k$

□

Corollary 2.50. *Accessibility is a class property for states of doubly stochastic Markov chains.*

Proof. .

- reflexive - $\forall i q_i \rightarrow q_i$ by Lemma 2.44
- symmetric - If $q_i \rightarrow q_j$ then $q_j \rightarrow q_i$ by Corollary 2.47
- transitive - If $q_i \rightarrow q_j$ and $q_j \rightarrow q_k$ then $q_i \rightarrow q_k$

□

Therefore, for doubly stochastic Markov chains communication and accessibility divides the state space into mutually disjoint exclusive classes.

[4]if we would use definition of accessibility where a state is always accessible from itself then this corollary would hold for any finite Markov chain

Chapter 3

Quantum query algorithm complexity

3.1 Quantum lower bounds by polynomials

In this section we briefly describe polynomial method of quantum query complexity. We do not pay much attention on it, because we use adversary lower bound technique to prove quantum query complexity of our problem. We refer to [BBCMW 01, AM 04] in this section.

Consider finite field Z_2 For each Boolean function $f(X) : 0, 1^N \to 0, 1$ we can find such $p_f(x_0, x_1, ..., x_{N-1}) \in Z_2[x_0, x_1, ..., x_{N-1}]$ such that $f(a_0, a_1, ..., a_{N-1})$ $= p_f(a_0, a_1, ..., aN - 1)$ for all $(a_0, a_1, ..., a_{N-1}) \in Z_2^N$. That's true because every Boolean function can be written using only 2 binary operations: AND and NOT. AND operation can be represented as polynomial $p_{AND}(x_1, x_2) = x_1 x_2$, and NOT operation cab be represented as polynomial $p_{NOT}(x) = 1 - x$ (remember that $x^k = x$ for every $x \in Z_2$). We are interested to find such polynomials in the field of real numbers.

Definition 3.1. *We say the polynomial $p \in R[x_0, x_1, ...xN - 1]$* ***represents*** *Boolean function $f(x_0, x_1, ...x_{N-1})$ iff $f(x_0, x_1, ...x_{N-1}) = p(x_0, x_1, ...x_{N-1})$ for all $(x_0, x_1, ...x_{N-1}) \in 0, 1^N$*

We use $deg(f)$ to denote the degree of a minimum-degree p that represents f.

Definition 3.2. *We say the polynomial $p \in R[x_0, x_1, ...xN - 1]$* ***approximates*** *Boolean function $f(x_0, x_1, ...x_{N-1})$ with bounded-error δ ($\delta \leq \frac{1}{3}$), iff $|p(x_0, x_1, ...x_{N-1}) - f(x_0, x_1, ...x_{N-1})| \leq \delta$ for all $(x_0, x_1, ...x_{N-1}) \in 0, 1^N$*

We use $\widetilde{deg}(f)$ to denote the degree of a minimum-degree p that approximates f. For example, $x_0 x_1 ... x_{N-1}$ is a multilinear polynomial of degree N that represents the AND-function. Similarly, $1-(1-x_0)(1-x_1)...(1-x_{N-1})$ represents OR. The polynomial $\frac{1}{3}x_0+\frac{1}{3}x_1$ approximates but does not represent AND on 2 variables.

The next two lemmas link quantum networks with polynomials. The idea is to obtain the lower bound of approximation polynomial's degree. Then we can obtain lower bounds for Boolean function.

Lemma 3.3. *Consider quantum network that makes T oracle queries. Let $x_0, x_1, ..., x_{N-1}$ is input bit vector. For each k: $0 \geq k \leq 2^M - 1$ exists polynomials p_k, $p_k \in C[x_0, x_1, ..., xN-1]$ each of degree at most T, such that final state of the network is the superposition*

$$\sum_k p_k(x_0, x_1, ..., x_{N-1}) |k\rangle$$

where M is the maximum number of qubits used in computational process.

Proof. Every quantum network U can be represented as sequence of unitary transformations: $U{=}U_T O_X U_{T-1} O_x ... U_1 O_x U_0$ where U_k is arbitrary unitary transformation and O_X is unitary transformation representing oracle query that maps state $|i, b, z\rangle$ to the state $|i, b \oplus x_i, z\rangle$ Let quantum network is in state $|\phi_i\rangle$ before ith query. Then $|\phi_{i+1}\rangle{=}U_i O_X |\phi_i\rangle$ The amplitudes in $|\phi_0\rangle$ depend on the initial state and on U_0 but not on X, so they are polynomials of X of degree 0. A query maps basis state $|i, b, z_i\rangle$ to $|i, b \oplus x_i, z_i\rangle$. Hence if the amplitude of $|i, 0, z_i\rangle$ in $|\phi_0\rangle$ is α and the amplitude of $|i, 1, z_i\rangle$ in $|\phi_0\rangle$ is β, then the amplitude of $|i, 0, z_i\rangle$ after the query becomes $(1-x_i)\alpha+x_i\beta$ and the amplitude of $|i, 1, z_i\rangle$ becomes $x_i\alpha+(1-x_i)\beta$, which are polynomials of degree 1. (In general, if the amplitudes before a query are polynomials of degree $\leq j$, then the amplitudes after the query will be polynomials of degree $\leq j+1$.) Between the first and the second query lies the unitary transformation U_1. However, the amplitudes after applying U_1 are just linear combinations of the amplitudes before applying U_1, so the amplitudes in $|\phi_1\rangle$ are polynomials of degree at most 1. Continuing in this manner, the amplitudes of the final states are found to be polynomials of degree at most T. □

If the amplitude of state $|k\rangle$ is α_k then the probability of getting this state $|k\rangle$ after the measurement is $|\alpha_k|^2$ By lemma 3.3 α_k are polynomials in field $C[x_0, x_1, ..., x_{N-1}]$ each of degree at most T. Then we can rewrite α_k: $\alpha_k{=}\beta_k(x_0, x_1, ..., x_{N-1}){+}i\gamma_k(x_0, x_1, ..., x_{N-1})$ where β_k,γ_k *in* $R[x_0, x_1, ..., x_{N-1}]$ each of degree at most T. As $|\alpha_k|^2 = |\beta_k|^2 + |\gamma_k|^2$ we obtain the next lemma:

Lemma 3.4. *Let N be a quantum network that makes T oracle queries. Then exists polynomials $q_k(x_0, x_1, ...x_{N-1}) \in R[x_0, x_1, ..., x_{N-1}]$ each of degree at* *most 2T such that probability of getting state $|k\rangle$ after the measurement is equal $q_k(x_0, x_1, ...x_{N-1})$*

The lemmas 3.3 and 3.4 imply the lower bounds of Boolean functions.

Theorem 3.5. *For Boolean function f : $Q_2(f) \geq \frac{\widetilde{deg(f)}}{2}$*

Proof. Consider a quantum network that computes Boolean function f with bounded-error *delta*. Let T be a number of oracle queries made. Then the final state of the network Ψ can be rewritten as follows:

$$\Psi = \sum_{k=0}^{2^{M-1}-1} \alpha_k |k\rangle|1\rangle + \sum_{k=0}^{2^{M-1}-1} \beta_k |k\rangle|0\rangle \,,$$

where the rightmost bit is the result of the computation. The probability of getting 1 after the measurement is equal $\sum_{k=0}^{2^{M-1}-1} |\alpha_k|^2$. The lemma 3.4 implies that probability of getting 1 is a multilinear real polynomial p with degree at most 2T. As quantum network computes f with bounded-error δ then such statements hold:

- if $f(x_0, x_1, ..., x_{N-1}) = 1$ then $p(x_0, x_1, ..., x_{N-1}) \geq 1 - \delta$, and then

$$|f(x_0, x_1, ..., x_{N-1}) - p(x_0, x_1, ..., x_{N-1})| \leq \delta$$

- if $f(x_0, x_1, ..., x_{N-1}) = 0$ then $p(x_0, x_1, ..., x_{N-1}) \leq \delta$, and then

$$|f(x_0, x_1, ..., x_{N-1}) - p(x_0, x_1, ..., x_{N-1})| \leq \delta$$

Then $p(x_0, x_1, ..., x_{N-1})$ approximates f with bounded-error δ. As p degree is at most 2T, then T $\geq \frac{\widetilde{deg(f)}}{2}$ □

Theorem 3.6. *For Boolean function f : $Q_E(f) \geq \frac{deg(f)}{2}$*

Proof. The proof is analogical to the proof of theorem 3.5. The only difference is that computation is exact, then $p(x_0, x_1, ..., x_{N-1})$ represents f. As p degree is at most 2T, then T $\geq \frac{deg(f)}{2}$ □

For example, for Boolean function $AND(x_0, x_1, ..., x_{N-1})$ we can find multilinear real polynomial $p(x_0, ...x_{N-1}) = x_0 x_1 ... x_{N-1}$ that represents AND (indeed if all input bits are 1 then p(X) = 1, otherwise p(X) = 0). This implies that $Q_E(f) \geq \frac{N}{2}$

The main advantage of the polynomials method is that this reduces proving lower bounds on quantum algorithms to proving lower bounds on degree of polynomials. This is a well-studied mathematical problem with methods from approximation theory [C 66] available. Quantum lower bounds shown by polynomials method include a $Q_2(f) = \Omega(\sqrt[6]{D(f)})$ relation for any total Boolean function f [BBCMW 01], lower bounds on finding mean and median [NW 99], collisions and element distinctness [AM 99, Ku 03]. Polynomials method is also a key part of recent $\Omega(\sqrt{N})$ lower bound on set disjointness which resolved a longstanding open problem in quantum communication complexity [RA 03].

Given the usefulness of polynomials method, it is an important question how tight is the polynomials lower bound. [BBCMW 01, BW 02] proved that, for all total Boolean functions, $Q_2(f) = O(deg^6(f)$ and $Q_E(f) = O(deg^4(f))$. The second result was improved to $Q_E(f) = O(deg^3(f))$ [MI 04]. Thus, the bound is tight up to polynomial factor. Even stronger result would be $Q_E(f) = O(deg(f))$ or $Q_2(f) = O(\widetilde{deg}(f))$. Then, determining the quantum complexity would be equivalent to determining the degree of a function as a polynomial. It has been an open problem to prove or disprove either of these two equalities [BBCMW 01, BW 02]. A. Ambainis In [AM 04] showed the first provable gap between polynomial degree and quantum complexity: $deg(f) = 2^d$ and $Q_2(f) = \Omega(2.5^d)$. Since $deg(f) \geq \widetilde{deg}(f)$ and $Q_E(f) \geq Q_2(f)$, this implies a separation both between $Q_E(f)$ and $deg(f)$ and between $Q_2(f)$ and $\widetilde{deg}(f)$. To prove the lower bound, Ambainis used the quantum adversary method of [AM 02]. This method is of high importance for us, because we use it to prove lower bounds for graph circuit problem.

3.2 Adversary methods

The quantum adversary method runs a quantum algorithm on different inputs from some set. If every input in this set can be changed in many different ways so that the value of the function changes, many queries are needed. The original version of the quantum adversary method, let us call it unweighted, was invented by Ambainis [AM 02]. It was successfully used to obtain the following tight lower bounds: $\Omega(\sqrt{n})$ for Grover search [GR 96], $\Omega(\sqrt{n})$ for two-level And-Or trees (see [HMW 03] for a matching upper bound), and $\Omega(\sqrt{n})$ for inverting a permutation. The method starts with choosing a set of pairs of inputs on which f takes different values. Then the lower bound is determined by some combinatorial properties of the graph of all pairs chosen.

Some functions, such as sorting or ordered search, could not be satisfactorily lower-bounded by the unweighted adversary method. Hoyer, Neerbek, and Shi used a novel argument [HNS 02] to obtain tight bounds for these problems. They weighted the input pairs and obtained the lower bound by evaluating the spectral norm of the Hilbert matrix. Barnum, Saks, and Szegedy proposed a general method [BSS 03] that gives necessary and sufficient conditions for the existence of a quantum query algorithm. They also described a special case, the so-called spectral method, which gives a lower bound in terms of spectral norms of an adversary matrix. Ambainis also published a weighted version of his adversary method [AM 04]. He showed that it is stronger than the unweighted method and successfully applied it to get a lower bound for several iterated functions. This method is slightly harder to apply, because it requires one to design a so-called weight scheme, which can be seen as a quantum counterpart of the classical hard distribution on the inputs. Zhang observed that Ambainis had generalized his oldest method [AM 02] in two independent ways, so he unified them, and published a strong weighted adversary method [ZH 04]. Finally, Laplante and Magniez used Kolmogorov complexity in an unusual way and described a Kolmogorov complexity method [LM 04].

It was assumed that all adversary methods are related to each other. At first, Laplante and Magniez showed that the Kolmogorov complexity method is at least as strong as all the following methods: the Ambainis unweighted and weighted method, the strong weighted method, and the spectral method. Later Spalek and Szegedy proved that all methods are equivalent.

In addition it was known limitations for lower bounds obtained by the adversary methods. Firstly Szegedy in [SZ 03] proved that the weighted adversary method is limited by $\min(\sqrt{C_0 n}, \sqrt{C_1 n})$, where C_0 is the zero-certificate complexity of Boolean function f and C_1 is the one-certificate complexity of f. Laplante and Magniez proved the same limitation for the Kolmogorov complexity method [LM 04]. Finally Zhang in [ZH 04] and independently Spalek and Szegedy [SZ 03] improved this bound to $\sqrt{C_0 C_1}$ for total f.

In this section we briefly describe all these methods, concentrating on Ambainis's method, because we use it to prove lower bounds for graph circuit problem.

3.2.1 Spectral adversary method

Theorem 3.7. *Let $f(X) : \{0,1\}^N \to \{0,1\}$ be a partial boolean function, D_i,F be $|N| \times |N|$ zero-one valued matrices that satisfy $D_i[x,y] = 1$ iff $x_i \neq y_i$ for $i \in [N]$, and $F[x,y] = 1$ iff $f(x) \neq f(y)$. Let Γ denote an $|N| \times |N|$ nonnegative symmetric matrix such that $\Gamma \circ F = \Gamma$. Then*

$$Q_2(f) = \Omega(\max_{\Gamma} \frac{\lambda(\Gamma)}{\max_{i\in[N]} \lambda(\Gamma \circ D_i)}).$$

3.2.2 Kolmogorov complexity

Deep knowledge of Kolmogorov complexity is not necessary to understand the results in this section. We just introduce lower bound technique based on this method. The results on the relation between various classical forms of the quantum adversary method and the Kolmogorov complexity method are taken from Laplante and Magniez [LM 04].

Definition 3.8. *A set is called **prefix-free** if none of its members is a prefix of another member.*

Definition 3.9. *For a fixed universal Turing machine M and a prefix-free set S, the **prefix-free Kolmogorov complexity** of x given y, denoted by $K(x|y)$, is the length of the shortest program from S that prints x if it gets y on the input. Formally,*

$$K(x|y) = \min\{|P| : P \in S, M(P, y) = x\}.$$

Laplante and Magniez gave lower bounds for quantum query complexity, using Kolmogorov complexity.

Theorem 3.10. *Let $\sigma \in \{0,1\}^*$ denote a finite string. Then*

$$Q_2(f) = \Omega(\min_{\sigma} \max_{x,y,f(x)\neq f(y)} \frac{1}{\sum_{i:x_i\neq y_i} \sqrt{2^{-K(i|x,\sigma)-K(i|y,\sigma)}}}).$$

3.2.3 Weighted adversary

The previously known version of quantum adversary method gives a weaker lower bound of $Q_2(f) = \Omega(2.1213...^d)$. While this already gives some gap between polynomial degree and quantum complexity, we can achieve a larger gap by using a new, more general version of the method. The new component is that we carry out this argument in a very general way. We assign individual weights to every pair of inputs and distribute each weight 2 among the two inputs in an arbitrary way. This allows us to obtain better bounds than with the previous versions of the quantum adversary method. We apply the new lower bound theorem to three functions for which deterministic query complexity is significantly higher than polynomial degree. The result is that, for all of those functions, quantum query complexity is higher than polynomial degree. The biggest gap is polynomial degree 2^d = M and query complexity $\Omega(2.5^d) = \Omega(M^{1.321...})$. Spalek and Szegedy [SS 04] have recently shown that Ambainis method is equivalent to two other methods, the spectral method of [BSS 03] that was known prior to Ambainis's work [AM 04] and the Kolmogorov complexity method of [LM 04] that appeared after the conference version of Ambainis's paper was published. Although all three methods are equivalent, they have different intuition. It appears to us that Ambainis's method is the easiest to use for results in this paper. Let start with the earliest Ambainis's adversary method [AM 02].

Theorem 3.11. *Let $f : 0,1^N \to 0,1$, $A \subset 0,1^N$, $B \subset 0,1^N$, be such that $f(A) = 0$, $f(B) = 1$ and*

- *for every $x \in A$, there are at least m inputs $y \in B$ such that y differs from x in one bit*
- *for every $y \in B$, there are at least m' inputs $x \in A$ such that x differs from y in one bit.*

Then, $Q_2(f) = \Omega(\sqrt{mm'})$

Later Ambainis gave another generalized version of this theorem.

Theorem 3.12. *Let $f : 0,1^N \to 0,1$, $A \subset 0,1^N$, $B \subset 0,1^N$, $R \subset A \times B$ be such that $f(A) = 0$, $f(B) = 1$ and*

- *for every $x \in A$, there are at least m inputs $y \in B$ such that $(x,y) \in R$*
- *for every $y \in B$, there are at least m' inputs $x \in A$ such that $(x,y) \in R$*
- *for every $x = (x_1...x_N) \in A$ and every $i \in [N]$ there are at most l inputs $y \in B$ such that $(x,y) \in R$ and $x_i \neq y_i$*
- *for every $y = (y_1...y_N) \in B$ and every $i \in [N]$, there are at most l' inputs $x \in A$ such that $(x,y) \in R$ and $x_i \neq y_i$.*

Then, $Q_2(f) = \Omega(\sqrt{\frac{mm'}{ll'}})$

The main idea of these theorems is if we can split input data into 2 sets X, Y such that f(X) =0 and f(Y) = 1 and there are lots of possibilities how to obtain X from Y, changing some bits, than lower bounds had to be big enough.

Finally Ambainis provided generalization of this theorem.

Theorem 3.13. *Let $f : 0,1^N \to 0,1$, $A \subset 0,1^N$, $B \subset 0,1^N$, $R \subset A \times B$ be such that $f(A) = 0$, $f(B) = 1$ and*

- *for every $x \in A$, there are at least m inputs $y \in B$ such that $(x,y) \in R$*
- *for every $y \in B$, there are at least m' inputs $x \in A$ such that $(x,y) \in R$*

- *let* $l_{x,i} = |y : (x,y) \in R, x_i \neq y_i|$, $l_{y,i} = |x : (x,y) \in R, x_i \neq y_i|$,

$$l_{max} = \max_{x,y,i:(x,y)\in R, i\in[N], x_i\neq y_i} l_{x,i} l_{y,i}$$

Then, $Q_2(f) = \Omega(\sqrt{\frac{mm'}{l_{max}}})$

In [AM 04] Ambainis gave another way to generalize theorem 3.12. He used weighted scheme to prove better lower bounds.

Definition 3.14. *Let* $f : 0,1^N \to 0,1$*,* $A \subset f^{-1}(0)$*,* $B \subset f^{-1}(1)$ *and* $R \subset A\times B$*. A weight scheme for A,B,R consists of numbers* $w(x,y) > 0$*,* $u(x,y,i) > 0$*,* $v(y,x,i) > 0$ *for all* $(x,y) \in R$ *and* $i \in [N]$ *satisfying* $x_i \neq y_i$*, we have*

$$u(x,y,i)v(y,x,i) \geq w^2(x,y).$$

Definition 3.15. *The weight of input vector is* $w_x = \sum_{y:(x,y)\in R} w(x,y)$*, if* $x \in A$ *and* $w_y = \sum_{x:(x,y)\in R} w(x,y)$ *if* $y \in B$*.*

Definition 3.16. *Let* $i \in [N]$*. The load of variable* x_i *is*

$$u_{x,i} = \sum_{y:(x,y)\in R, x_i\neq y_i} u(x,y,i)$$

and the load of variable y_i *is*

$$v_{y,i} = \sum_{x:(x,y)\in R, x_i\neq y_i} v(x,y,i)$$

.

Theorem 3.17. *Let* $f : 0,1^N \to 0,1$ *where* $X \subset f^{-1}(0)$*,* $Y \subset f^{-1}(1)$ *and* $R \subset X \times Y$ *. Let w, u, v be a weight scheme for X, Y, R. Then*

$$Q_2(f) = \Omega(\sqrt{\min_{x\in X, 0\leq i\leq N-1} \frac{w_x}{u_{x,i}} \min_{y\in Y, 0\leq j\leq N-1} \frac{w_y}{u_{y,j}}})$$

It's easy to see that theorem 3.17 generalizes theorem 3.13. Indeed, for each A, B, R from 3.13 we can construct such weight scheme: $w(x,y) = 1$, $u(x,y,i) = \frac{\sqrt{l_{max}}}{l_{x,i}}$, $v(x,y,i) = \frac{\sqrt{l_{max}}}{l_{y,i}}$ then:

- $u(x,y,i)v(x,y,i) = \frac{l_{max}}{l_{x,i}l_{y,i}} \geq 1 \Rightarrow u(x,y,i)v(x,y,i) \geq w^2(x,y)$

- $u_{x,i} = \sum_{y:(x,y)\in R, x_i \neq y_i} u(x,y,i) = l_{x,i} u(x,y,i)$
- $v_{y,i} = \sum_{x:(x,y)\in R, x_i \neq y_i} v(x,y,i) = l_{y,i} v(x,y,i)$

Then $\sqrt{\min_{x\in X, 0\leq i\leq N-1} \frac{w_x}{u_{x,i}} \min_{y\in Y, 0\leq j\leq N-1} \frac{w_y}{u_{y,j}}} = \sqrt{\min_{x\in X, 0\leq i\leq N-1} \frac{w_x}{l_{max}} \min_{y\in Y, 0\leq j\leq N-1} \frac{w_y}{l_{max}}}$ $= \sqrt{\frac{mm'}{l_{max}}}$, thus theorem 3.17 gives at least the same lower bounds as 3.13.

Later Zhang in [ZH 04] gave improved version of Ambainis's theorem.

Theorem 3.18. *Let $f : 0,1^N \to 0,1$, and X, Y be two sets of inputs such that $f(x) \neq f(y)$ if $x \in X$ and $y \in Y$. Let $R \subset X \times Y$. Let w, u, v be a weight scheme for X, Y,R. Then*

$$Q_2(f) = \Omega(\sqrt{\min_{(x,y)\in R, i\in[N], x_i\neq y_i} \frac{w_x w_y}{u_{x,i} v_{y,i}}})$$

It's obvious that theorem 3.18 gives better lower bounds that theorem 3.17. Indeed, for each, X, Y, R, w, u, v holds:

$$\min_{x,y,i,j:x\in X, y\in Y, i,j\in[N]} \frac{w_x w_y}{u_{x,i} v_{y,j}} \leq \min_{x,y,i:(x,y)\in R, i\in[N], x_i\neq y_i} \frac{w_x w_y}{u_{x,i} v_{y,i}}.$$

3.2.4 Limitations of adversary methods

Let us denote by $Alb_1(f)$, $Alb_2(f)$ and $Alb_3(f)$ the best lower bound for function f achieved by Theorem 3.13, 3.17 and 3.18, respectively (it means $Alb_1(f) = \max_{X,Y,R}(\sqrt{\frac{mm'}{l_{max}}})$ e.t.c.). Note that in the four *Alb*s, there are many parameters (X, Y,R, u, v,w) to be set. By setting these parameters in an appropriate way, one can get lower bounds of quantum query complexity for many problems. Since theorem 3.12 generalizes theorem 3.13 and theorem 3.18 generalizes 3.12 then $Alb_1 \leq Alb_2 \leq Alb_3$ generalizes. But Alb_3 also has a limitation that was shown by Zhang [ZH 04]. Let start with definition.

Definition 3.19. *Let $f : \{0,1\}^N \to \{0,1\}$ be a Boolean function and $x \in \{0,1\}^N$ be some input, a certificate set CS_x of f on x is a set of indices such that $f(x) = f(y)$ whenever $y_i = x_i$ for all $i \in CS_x$. The certificate complexity $C(f,x)$ of f on x is the size of a smallest certificate set of f on x. The 0-certificate complexity of f is $C_0(f) = \max_{x:f(x)=0} C(f,x)$. The 1-certificate complexity of f is $C_1(f) = \max_{x:f(x)=1} C(f,x)$. The certificate complexity of f is $C(f) = \max\{C_0(f), C_1(f)\}$. We further denote $C_{\_}(f) = \min\{C_0(f), C_1(f)\}$.*

For example, if f is the OR-function, then the certificate complexity on X = (1, 0, 0, . . . , 0) is 1, because the assignment $x_0 = 1$ already forces the OR to 1. The same holds for the other X for which $f(X) = 1$, so $C_1(f) = 1$. On the other hand, the certificate complexity on X = (0, 0, . . . , 0) is N, so $C_0(f) = N$ and $C(f) = N$ and $C_-(f) = 1$.

Theorem 3.20. *[ZH 04] $Alb_3(f) \leq \sqrt{N \cdot C_-(f)}$ for any total N-ary Boolean function f.*

Theorem 3.21. *[ZH 04] $Alb_3(f) \leq \sqrt{C_0(f) \cdot C_1(f)}$ for any total N-ary Boolean function f.*

Later Spalek and Szegedy in [SS 04] showing that all adversary methods are equivalent, have also proved that all adversary methods have such limitation as in theorem 3.21.

Theorem 3.22. *Let $f : \{0,1\}^N \to \{0,1\}$ be a Boolean function. Let $Q_2(f)$ denote the bounded-error quantum query complexity of f. Then $\frac{Q_2(f)}{1-2\sqrt{e(1-e)}} \geq SA(f) = WA(f) = \Omega(KA(f))$, where SA, WA and KA are lower bounds given by the following methods:*

- *$SA(f)$- Spectral adversary lower bound obtained by theorem 3.7*

$$SA(f) = \max_{\Gamma} \frac{\lambda(\Gamma)}{\max_{i\in[N]} \lambda(\Gamma \circ D_i)}$$

- *$WA(f)$ - weighted adversary lower bound obtained by theorem 3.17 and 3.18*

$$WA(f) = \sqrt{\min_{(x,y)\in R, i\in[N], x_i\neq y_i} \frac{w_x w_y}{u_{x,i} v_{y,i}}}$$

- *$KA(f)$- Kolmogorov lower bound obtained by theorem 3.10*

$$KA(f) = \min_{\sigma} \max_{x,y,f(x)\neq f(y)} \frac{1}{\sum_{i:x_i\neq y_i} \sqrt{2^{-K(i|x,\sigma)-K(i|y,\sigma)}}}$$

3.2.5 Separation between the polynomial and adversary method

The polynomial method and the adversary method are generally incomparable. There are examples when the polynomial method gives better bounds and vice versa. The polynomial method has been successfully applied to obtain tight lower bounds for the following problems: $\Omega(n^{1/3})$ for the collision problem and $\Omega(n^{2/3})$ for the element distinctness problem [1] (see [4] for a matching upper bound). The quantum adversary method is incapable of proving such lower bounds due to the small certificate complexity of the function. Furthermore, the polynomial method often gives tight lower bounds for the exact and zero-error quantum complexity, such as n for the Or function [8]. The adversary method completely fails in this setting and the only lower bound it can offer is the bounded-error lower bound. On the other hand, Ambainis exhibited some iterated functions [3] for which the adversary method gives better lower bounds than the polynomial method. The largest established gap between the two methods is $n^{1.321}$. Furthermore, it is unknown how to apply the polynomial method to obtain several lower bounds that are very simple to prove by the adversary method. A famous example is the two-level And-Or tree. The adversary method gives a tight lower bound $\Omega(\sqrt{n})$ [2], whereas the best bound obtained by the polynomial method is $\Omega(n^{1/3})$ and it follows [5] from the element distinctness lower bound [1]. There are functions for which none of the methods is known to give a tight bound. A long-standing open problem is the binary And-Or tree. The best known quantum algorithm is just an implementation of the classical zero-error algorithm by Snir [25] running in expected time $\O(n^{0.753})$, which is optimal for both zero-error [23] and bounded-error [24] algorithms. As we have shown in previosu section the adversary lower bounds are limited by $\sqrt{C_0 C_1} = \sqrt{n}$. The best known lower bound obtained by the polynomial method is also $\Omega(\sqrt{n})$ and it follows from embedding the parity function. It could be that the polynomial method can prove a stronger lower bound. Two other examples are triangle finding and verification of matrix products. For triangle finding, the best upper bound is $\O(n^{1.3})$ [20] and the best lower bound is $\Omega(n)$. For verification of matrix products, the best upper bound is $\O(n^{5/3})$[10] and the best lower bound is $\Omega(n^{3/2})$. Again, the adversary method cannot give better bounds, but the polynomial method might. It is an interesting open problem to find a lower bound that cannot be proved by the adversary or polynomial method.

3.3 Relation between classical and quantum complexity

In In this section we will show the relation between the classical complexities $D(f)$ and $R(f)$ and the quantum complexities. By a well-known result, the best randomized decision tree can be at most polynomially more efficient than the best deterministic decision tree: $D(f) \in O(R(f)^3)$ [23, Theorem 4]. Beals, Buhrmans e.t.c using polynomial method proved that also the quantum complexity can be at most polynomially better than the best deterministic tree: $D(f) \in O(Q_2(f)^6)$. In other words, if we can compute some function quantumly with bounded-error using T queries, we can compute it classically error-free with $O(T^6)$ queries. We will briefly describe the idea of their proof in this section. To start we define *blocksensitivity* complexity.

Let f be a Boolean function and $X = (x_0, ..., x_{N-1})$ an input to f. For a set $S \subset 0, ..., N-1$, $X^{(S)}$ denotes the input obtained from X by flipping all variables x_i, $i \in S$. f is *sensitive* to S on input X if $f(X) \neq f(X^{(S)})$.

Definition 3.23. *The* ***block sensitivity of f on input X*** *is the maximal number t such that there exist t pairwise disjoint sets S_0, S_1, . . ., S_{t-1} such that, for all $i \in \{0, ..., t-1\}$, f is sensitive to S_i on X. We denote it by $bs_X(f)$.*

Definition 3.24. *The* ***block sensitivity of f****, $bs(f)$ is just the maximum of $bs_X(f)$ over all inputs X.*

For example, $bs(OR) = N$, because for input $X = (0, 0, ..., 0)$ we can take $S_i = i$, then flipping the $i-th$ bit in X changes the value of OR from 0 to 1.

Nissan in [NS 91] proved the relationship between certificate complexity and block-sensitivity complexity.

Lemma 3.25. *[NS 91] $C_1(f) \leq C(f) \leq bs(f)^2$.*

Beals, Buhrman e.t.c in [BBCMW 01] using polynomial method proved quantum lower bounds in the terms of block sensitivity.

Lemma 3.26. *[BBCMW 01] If f is a Boolean function, then $Q_E(f) \geq \sqrt{\frac{bs(f)}{8}}$ and $Q_2(f) \geq \sqrt{\frac{bs(f)}{16}}$.*

Lemma 3.27. *[BBCMW 01] $D(f) \leq C_1(f)bs(f)$.*

Lemmas 3.25 and 3.27 imply that $D(f) \leq bs(f)^3$. Combining this with lemma 3.26 Beals e.t.c. in [BBCMW 01] obtained this result.

Theorem 3.28. *[BBCMW 01] If f is a Boolean function then $D(f) \leq 4096Q_2(f)^6$*

They also give a limitation on exact quantum query computation.

Theorem 3.29. *[BBCMW 01] If f is a Boolean function then $D(f) \leq 32Q_E(f)^4$*

Later this result was improved by Midrijanis [MI 04]

Theorem 3.30. *[MI 04] If f is a Boolean function then $D(f) \in \O(Q_E(f)^3)$*

It was an open problem whether the polynomial method is tight or not? It means whether $Q_E(f) = O(deg(f))$ or $Q_2(f) = O(\widetilde{deg(f)})$? Ambainis in [AM 04] showed the gap between polynomial degree and quantum complexity. He constructed the function f for which: $deg(f) = 2^d$ and $Q_2(f) = (2.5^d)$. Since $deg(f) \geq \widetilde{deg(f)}$ and $Q_E(f) \geq Q_2(f)$, this implies a separation both between $Q_E(f)$ and $deg(f)$ and between $Q_2(f)$ and $\widetilde{deg(f)}$

3.3.1 Query complexity upper bounds

There is no major technique to obtain quantum query complexity upper bounds for any problem as it is in the case of lower bounds. We just need to provide correct quantum algorithm that solves the problem. We also need to prove query complexity lower bounds for the problem to show that algorithm solving this problem is optimal. In this section we briefly describe the amplitude amplification method [BHMT 02] for getting query complexity upper bounds, because we will use it in our proof for graph circuit problem. The method is a generalization of Grover's search algorithm [GR 96].

Consider a problem that is characterized by Boolean function $\chi(x, y)$ in the sense that y is a good solution to instance x if and only if $\chi(x, y) = 1$ (there could be more than one good solution to a given instance x). If we have a probabilistic algorithm P that outputs a guess $P(x)$ on input x, we can call P and χ repeatedly until the solution to instance x is found. If $\chi(x, P(x)) = 1$ with probability $p_x > 0$ then we expect to repeat this process $1/p_x$ times on the average. Consider now the case when we have a quantum algorithm A instead of the probabilistic algorithm. Assume A makes no measurements: instead of the classical answer it produces quantum superposition $|\Psi_x\rangle$ when run on x. Let a_x denote the probability that $|\Psi_x\rangle$, if measured, is a good solution. If we repeat the process of running A on x, measuring the output, and applying χ to check the validity of the result, we shall expect to repeat $1/a_x$ times on the average before a solution is found. This is no better than the classical probabilistic paradigm.

Brassard e.t.c in [BHMT 02] described a more efficient approach to the problem which they called amplitude amplification. The probabilistic paradigm increases the probability of success roughly by a constant. By contrast, amplitude amplification increases the amplitude of success roughly by a constant on each iteration. Because amplitudes correspond to square roots of probabilities, it suffices to repeat the amplitude amplification process approximately

$1/\sqrt{a_x}$ times to achieve success with overwhelming probability. Formally, in [BHMT 02] it was proved such theorem.

Theorem 3.31. *[BHMT 02] Let A be a quantum algorithm with one-sided error and success probability at least $\epsilon > 0$. Then, there is a quantum algorithm B that solves the same problem with success probability $2/3$ by invoking A $O(\frac{1}{\epsilon})$ times.*

The Grover's search algorithm and amplitude amplification method can simplify constructing of quantum algorithms. It is well-known fact that any classical (either deterministic or probabilistic) computation can be simulated on a quantum computer (see []). More precisely,

- In the circuit model, a classical circuit with N gates can be simulated by a quantum circuit with $O(N)$ gates.
- In the query model (when only the number of queries is counted), a classical computation with N queries can be simulated by a quantum computation with N queries.

This greatly simplifies descriptions of quantum algorithms. Instead of describing a quantum algorithm, we can describe a classical algorithm that succeeds with some small probability ϵ. Then, we can transform the classical algorithm to a quantum algorithm and apply the amplitude amplification to the quantum algorithm. The result is a quantum algorithm with the running time or the number of queries that is $O(\frac{1}{\epsilon})$ times the one for the classical algorithm with which we started. A similar reasoning can be applied, if instead of a purely classical algorithm, we started with a classical algorithm that involves quantum subroutines. Such algorithms can also be transformed into quantum algorithms with the same complexity. For example we can provide very simple upper bound for graph triangle problem.

Finding triangles problem. [SZ 03] We have n^2 variables describing adjacency matrix of a graph. We would like to know if the graph contains a triangle. The very simple quantum algorithm can be written as follows:

- choose any 3 vertices of the graph and check whether they form a triangle;
- amplify the first step $\sqrt{n^3}$ times.

If graph contains a triangle then the probability of getting correct answer by choosing randomly any 3 graph vertices $p_\epsilon \geq \frac{1}{n} \cdot \frac{1}{n} \cdot \frac{1}{n}$. Then by theorem 3.31 we can repeat this step $O(\sqrt{\frac{1}{\frac{1}{n^3}}}) = O(\sqrt{n^3})$ times getting probability

$\geq 2/3$. This algorithm gives us quantum query complexity upper bounds: $Q_2(f) = O(\sqrt{n^3})$ (as we make constant number of oracle query in the first step of algorithm). In [SZ 03] it was proved better upper bound: $O(n^{1.3})$. It's easy to see that adversary methods can not give us a better lower bound then $\sqrt{n^2}$, because certificate complexity $C_0 = O(n^2)$ (if there is now triangle in the graph we need to check all edges), but $C_1 = 3$ (if there is a triangle, we just need to check the three edges, that form a triangle). It is an open problem to improve lower bounds for this problem, reducing the gap between upper and lower bound.

Similarly we can obtain upper bounds for graph rectangle problem.

Finding rectangle problem. We have n^2 variables describing adjacency matrix of a graph. We would like to know if the graph contains a rectangle. The quantum algorithm can be written as follows:

- choose any 2 vertices i, j of the graph;
- with 2 Grover's searches find vertices k, l that are connected with both i and j;
- amplify the first 2 steps $\sqrt{n^2} = n$ times.

If graph contains a rectangle then the probability of getting correct answer by choosing randomly any 2 graph vertices $p_\epsilon \geq \frac{1}{n} \cdot \frac{1}{n}$. Then by theorem 3.31 we can repeat this step $O(\sqrt{\frac{1}{\frac{1}{n^2}}}) = O(\sqrt{n^2}) = O(n)$ times getting probability $\geq 2/3$. This algorithm gives us quantum query complexity upper bounds: $Q_2(f) = O(\sqrt{n^3})$ (as we make 2 Grovers searches on n elements in the second step of the algorithm). It's easy to see that adversary methods can not give us a better lower bound then $\sqrt{n^2}$, because (as for triangle problem) certificate complexity $C_0 = O(n^2)$ (if there is now rectangle in the graph we need to check all edges), but $C_1 = 4$ (if there is a rectangle, we just need to check the 4 edges, that form a triangle).

We can not use similar proofs to obtain upper bounds for the problem whether graph contains cycle of length k for some fixed k. So it is still open problem. In the next section we consider the graph cycle problem, providing an algorithm solving this problem. We also prove that these algorithm is optimal.

3.4 Graph cycle problem. Algorithm. Lower bounds.

Let G be an undirected graph with N vertices. We are interested to know whether graph contains any cycle? In this section we provide a quantum algorithm solving this problem and then show that this algorithm is optimal using adversary lower bound technique.

Graph cycle problem.

INSTANCE: Undirected graph $G = (V, E)$, $|E| = n$, represented by adjacency matrix $A = (a_{ij})$, where

$$a_{ij} = \begin{cases} 1, \text{if there is edge between i-th and j-th vertices} \\ 0, \text{otherwise} \end{cases}$$

QUESTION: Is there exists any circuit in this graph G?

In other words we have boolean function $f_G : \{0,1\}^N \to \{0,1\}$ (where $N = n^2$) that outputs 1 if there is any cycle in the graph G representing by its adjacency matrix.

It is easy to see that query complexity of classical algorithm is $O(n^2)$ (indeed, in the worst case we need to query (n^2-n) elements of the adjacency matrix if there is no triangle in the graph). We will prove the complexity of quantum algorithm that is better than in classical case. The quantum algorithm that solves the problem uses such lemma.

Lemma 3.32. *If the number of edges in the undirected graph with n vertices is greater or equal with n then the graph contains cycle.*

Proof. Lets the graph is connected, thats it, from every vertex there exists path to another vertex. Assume from the contrary, that there is no cycle in the graph containing $\geq n$ vertices. As we can reach any vertex from another then there is spanning tree T containing all vertices of the graph (see fig. 3.1). There are (n-1) edges in the spanning tree, because each edge can be inside the tree only once. As the number of edges in the graph is $\geq n$, then exists at least 2 vertices a_i un a_j connected with edge e, that does not belong to tree T. It means that exist at least 2 different ways how to reach vertex a_j from the vertex a_i: by the edge e or by the path inside the tree T (see pict. 3.1). Then there is a cycle in the graph, containing vertices a_i and a_j. If the graph is not connected, then it can be split into m connected subgraphs each containing v_1, v_2, ..., v_m vertices. Then exists subgraph in which the number of edges e_i is greater or equal with the number of vertices. (Otherwise, $\sum_{i \leq m} s_i < \sum_{i \leq m} v_i = n$). Then this subgraph contains a cycle. □

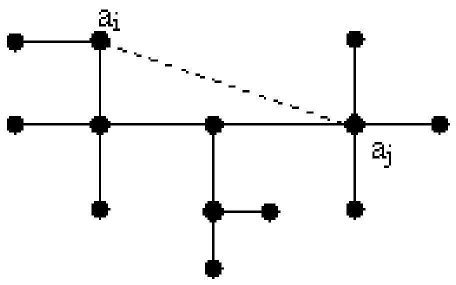

Figure 3.1: Spanning Tree. Additional edge $a_i a_j$ makes cycle

We provide quantum algorithm solving the problem 3.4 with bounded-error making $O(\sqrt{n^3})$ oracle queries.

Algorithm (FCYCLE(G,(a_{ij})).

1. Fix some $k = 1$, that denotes current row of the adjacency matrix we are working on. Fix $l = 1$, the starting column in the k-th row from which we shall begin quantum search for 1 in this row. Fix $M = 0$ - the number of founded edges.

2. Take the k-th row of the adjacency matrix. Run Grover's search algorithm to find such $j > l$ that $a_{kj} = 1$. If Grover's search finds such j then go to step 3. Otherwise, increase k by 1 (thats it, take next row of the adjacency matrix). If $k > n$ then go to step 4. Otherwise, set l to k, because the graph is undirected and we do not need to search in the first k elements of the k-th row in graph's adjacency matrix. Repeat the 2-nd step.

3. Remember the pair (k, j) and increase M by 1. If $M \geq n$, then output 1 and finish the computation. If $m < n$, tad set l to j and go to step 2 (search another 1 in the k-th row).

4. Run classical deterministic algorithm finding cycle on all pairs (k, j) (founded edges) remembered in the previous steps. The output of our algorithm equals with the output of this deterministic algorithm.

Note that algorithm not only detects whether graph contains a cycle or not, but it also provides a set of edges containing cycle, if there is cycle in the graph.

Correctness of the algorithm

We sequently find edges of the graph in the steps 2 and 3 of the Algorithm. We stop either we have found n-th edge or we have gone through all n rows of the adjacency matrix. If we have found n-th edge, then by lemma 3.32 graph contains cycle and we output 1. If we have found all edges and their count is less than n then we output correct answer just running deterministic algorithm on all founded edges.

Query complexity of the algorithm

We are interested in how many oracle questions about (a_{ij}) elements does the algorithm make. We are going to prove that $Q_2(f_g) = O(\sqrt{n^3})$. The 2-nd step of the Algorithm (where Grover's search is run) is computed $\leq 2n$ times, because each founded edge corresponds to one Grover's search, and one Grover's search is computed for each row, when there is no more 1-s(edges) in the row. In each Grover's search we make $O(\sqrt{n})$ queries, because input size is $\leq n$ for each Grover's search. Then the overall query complexity of the algorithm is $O(n\sqrt{n})$.

As we can see the quantum query algorithm complexity for graph cycle problem is $O(\sqrt{n^3})$, because we provide algorithm that solves the problem with such number of queries. To prove that this algorithm is optimal we need to prove that complexity lower bound is also $O(\sqrt{n^3})$. We can prove weaker lower bound, using theorem 3.11.

Theorem 3.33. *Quantum query algorithm, that solves graph cycle problem, needs to make $\Omega(n)$ queries.*

Proof. We will use theorem 3.11 to prove lower bounds. Lets make set X and Y in such way: the set X contains all spanning trees of the graph (see fig. 3.2). Then $f(X) = 0$. The set Y consists of all graphs that are made from spanning tree by adding one additional edge (see fig. 3.3). By lemma 3.32 $f(Y) = 1$.

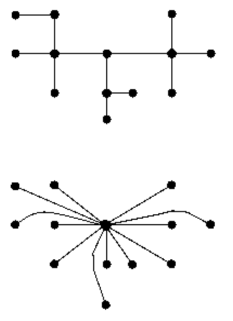
Figure 3.2: Instances of the set X

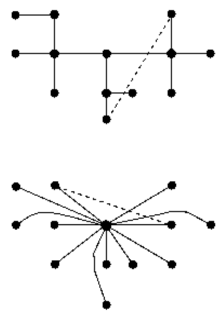
Figure 3.3: Instances of the set Y

For each $x \in X$ there are $m = C_n^2 = \frac{n(n-1)}{2} = O(n^2)$ different $y \in Y$ such that y differs from x in one bit. Indeed, we can add additional edge to the spanning tree by choosing any two vertices of the graph. For each $y \in Y$ there are O(1) different $x \in X$ such that x differs from y in one bit, because we obtain x from y by removing one of the edges in the cycle of the graph y. Note that there are such y from whom we can obtain x in $O(n)$ ways(if the length of the cycle is close to n), but for the graph with a short cycle, we do not have much possibilities to remove an edge. Then by theorem 3.11 quantum algorithm solving the problem makes $\Omega(\sqrt{n^2})$ queries to the oracle. □

As we can see the unweighted adversary method is not very useful for proving lower bounds for the graph cycle problem. We shall use weighted adversary method to improve lower bound.

Theorem 3.34. *Quantum query algorithm, that solves graph cycle problem, needs to make $\Omega(\sqrt{n^3})$ queries.*

Proof. We will construct sets X and Y in such way to apply theorem 3.17. Let the set X consists of all such graphs, that are obtained from the graphs with the cycle of length n by removing one edge from it (see fig. 3.4). The set Y consists of all such graphs, that are obtained from the graph with 2 cycles, each of length between $n/3$ and $2n/3$ by removing one random edge from one of the cycles (see fig. 3.5). The relation $R = X \times Y$ consists of such graph pairs (x, y) that only difference between x and y is: there are for 4 vertices a, b, c, d such that edges (a, b) and (c, d) are in the graph x, but not in y, and vice a versa: edges (a, c) and (b, d) are in the graph y, but not in x (see fig. 3.34). The weight scheme for X, Y, R is given as follows:

$$w(x, y) = 1, u(x, y, (i, j)) = \sqrt{n}, v(x, y, (i, j)) = \frac{1}{\sqrt{n}},$$

for all $(x, y) \in R$, $(i, j) \in [n^2]$ and $x_{ij} \neq y_{ij}$ ($x_{i,j}$ denotes an element of the graph's adjacency matrix). Its easy to see that weight scheme is correct, because $\sqrt{n}\frac{1}{\sqrt{(n)}} \geq w(x, y) = 1$

$w_x = \sum\limits_{y:(x,y)\in R} w(x, y) = O(n^2)$, because there are $O(n^2)$ such $y \in Y$, that $(x, y) \in R$. Indeed, we have $(n - 2)$ possibilities to choose edge (a, b) and then $n/3$ possibilities to choose the second edge (c, d). $w_y = \sum\limits_{x:(x,y)\in R} w(x, y) =$ $O(n^2)$, because there $O(n^2)$ such x, that $(x, y) \in R$. We need to remove two edges, each edge from the each of the two cycles in the graph y, then we have $O(n^2)$ possibilities to choose these edges, because the length of each cycle is greater or equal than $n/3$.

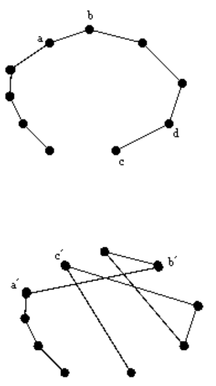

Figure 3.4: Instances of the set X

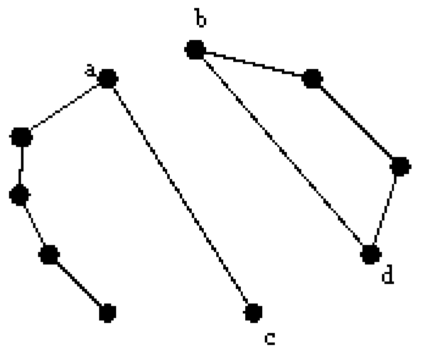

Figure 3.5: Instances of the set Y

$$u_{x,(i,j)} = \sum_{y:(x,y)\in R, x_{(i,j)}\neq y_{(i,j)}} u(x,y,(i,j)) =$$

$$= \sum_{y:(x,y)\in R, x_{(i,j)}=0\&y_{(i,j)}=1} u(x,y,(i,j)) + \sum_{y:(x,y)\in R, x_{(i,j)}=1\&y_{(i,j)}=0} u(x,y,(i,j))$$

$$\sum_{y:(x,y)\in R, x_{(i,j)}=0\&y_{(i,j)}=1} u(x,y,(i,j)) = 4\sqrt{n}$$

, because if there is no edge between i-th and j-th vertices in the graph x, but this edge is in the graph y, then these are vertices a, c (or vertices b and d). Then we have only 4 possibilities to choose neighbors of these vertices (the vertex b is the neighbor of the vertex a and vertex d is the neighbor of the vertex c).

$$\sum_{y:(x,y)\in R, x_{(i,j)}=1\&y_{(i,j)}=0} u(x,y,(i,j)) = O(n\sqrt{n})$$

, because the edge (i,j) is the edge (a,b) (or (c,d) edge) and we have $O(n)$ possibilities to remove from the graph the edge (c,d) (or the edge (a,b) respectively), because the length of the cycle is at least $n/3$. Similarly:

$$\sum_{x:(x,y)\in R, y_{(i,j)}=0\&x_{(i,j)}=1} v(x,y,(i,j)) = 4/\sqrt{n}$$

, and

$$\sum_{x:(x,y)\in R, y_{(i,j)}=1\&x_{(i,j)}=0} v(x,y,(i,j)) = O(n/\sqrt{n}).$$

Then the lower bound for the graph cycle problem is:

$$\sqrt{\min_{(x,y)\in R, i\in[n^2], x_{(i,j)}\neq y_{(i,j)}} \frac{w_x w_y}{u_{x,(i,j)} v_{y,(i,j)}}} =$$

$$\sqrt{\min(\min_{(x,y)\in R, i\in[n^2], x_{(i,j)}=0 \& y_{(i,j)}=0} \frac{w_x w_y}{u_{x,(i,j)} v_{y,(i,j)}}, \min_{(x,y)\in R, i\in[n^2], x_{(i,j)}=1 \& y_{(i,j)}=1} \frac{w_x w_y}{u_{x,(i,j)} v_{y,(i,j)}})} =$$

$$\sqrt{\min(\tfrac{n^2 n^2}{4\sqrt{n}(n/\sqrt{n})}, \tfrac{n^2 n^2}{n\sqrt{n}(4/\sqrt{n})})} =$$

$$\sqrt{\min(n^3, n^3)} = \sqrt{(n^3)}$$

. Then, by theorem 3.17 $Q_2(f) = \Omega(\sqrt{n^3})$. □

Chapter 4

Probabilistic reversible automata

Probabilistic reversible automata (PRA) were introduced by M. Golovkins and M. Kravtsev in [GK 02] as probabilistic automata which transformation is determined by doubly stochastic operators. In case of one-way automata that means that for each letter of working alphabet the transition matrix is doubly stochastic. This model is a probabilistic counterpart for Nayaks model of enhanced quantum automata. M.Golovkins and M.Kravtsev in [GK 02] and later M. Beadry e.t.c in [ABGKMT 06] proved the class of languages recognizable by PRA. In this thesis we address another type of PRA, i.e., the DH-PRA merely defined in [GK 02], that behave more like measure-many quantum automata [KW 97], as they halt when entering accepting or rejecting states. We show a general class of regular languages, not recognizable by DH-PRA. This class is identical to a class not recognizable by MM-QFA [AKV 01] (and similar to the class of languages, not recognizable by C-PRA [GK 02]. We also prove that the class of languages recognizable by DH-PRA is not closed under union. The one open problem still remains: we show the class of languages for which we can not prove whether it is recognizable by DH-PRA or not. So we still unable to prove or disprove that class of languages recognizable by DH-PRA is likely to include the one recognizable by MM-QFA or these classes are equal.

4.1 1-way Probabilistic Reversible Automata

In this section we give formal definition of 1-way PRA

Definition 4.1. *1-way probabilistic reversible C-automaton (C-PRA)* $A = (Q, \Sigma, q_0, Q_F, \delta)$ *is specified by a finite set of states* Q*, a finite input* *alphabet Σ, an initial state $q_0 \in Q$, a set of accepting states $Q_F \subseteq Q$, and a transition function*

$$\delta : Q \times \Gamma \times Q \longrightarrow \mathbb{R}_{[0,1]},$$

where $\Gamma = \Sigma \cup \{\#, \$\}$ is the input tape alphabet of A and $\#$, $\$$ are end-markers not in Σ. Furthermore, transition function satisfies the following requirements:

$$\forall (q_1, \sigma_1) \in Q \times \Gamma \ \sum_{q \in Q} \delta(q_1, \sigma_1, q) = 1 \tag{4.1}$$

$$\forall (q_1, \sigma_1) \in Q \times \Gamma \ \sum_{q \in Q} \delta(q, \sigma_1, q_1) = 1 \tag{4.2}$$

For every input symbol $\sigma \in \Gamma$, the transition function may be determined by a $|Q| \times |Q|$ matrix V_σ, where $(V_\sigma)_{i,j} = \delta(q_j, \sigma, q_i)$.

Lemma 4.2. *All matrices V_σ are doubly stochastic iff conditions (4.1) and (4.2) of Definition 4.1 hold.*

Proof. Trivial. □

A linear operator U_A corresponds to the automaton A. Formal definition of this operator follows:

Definition 4.3. *Given a configuration $c = \langle \nu_i q_j \sigma \nu_k \rangle$,*

$$U_A c \overset{\text{def}}{=} \sum_{q \in Q} \delta(q_j, \sigma, q) \langle \nu_i \sigma q \nu_k \rangle.$$

Given a superposition of configurations $\psi = \sum_{c \in C} p_c c$,

$$U_A \psi \overset{\text{def}}{=} \sum_{c \in C} p_c U_A c.$$

Using canonical basis, U_A is described by an infinite matrix M_A.

To comply with Definition 2.22, we have to state the following:

Lemma 4.4. *Matrix M_A is doubly stochastic iff conditions (4.1) and (4.2) of Definition 4.1 hold.*

Proof. Condition (4.1) takes place if and only if the sum of elements in every column in M_A equal to 1. Condition (4.2) takes place if and only if the sum of elements in every row in M_A equal to 1. □

This completes our formal definition of 1-way PRA.

4.2 1-way Probabilistic Reversible C Automata

In this section we give formal definition of C-PRA automata. And briefly describe the results from [GK 02] and [ABGKMT 06] because they are of high importance for us. We will further refer to them proving our theorems for the DH-PRA automata.

4.2.1 Definition

For a definition 2.22 we define word acceptance as specified in Definition 2.17. The set of rejecting states is $Q \setminus Q_F$. We define language recognition as in Definition 2.20. That completes formal definition of 1-way C-PRA automata.

4.2.2 Closure properties

Lemma 4.5. *If a language is recognized by a C-PRA A with interval* (p_1, p_2), *exists a C-PRA which recognizes the language with probability* p, *where*

$$p = \begin{cases} \frac{p_2}{p_1+p_2}, & \textit{if } p_1 + p_2 \geq 1 \\ \frac{1-p_1}{2-p_1-p_2}, & \textit{if } p_1 + p_2 < 1. \end{cases}$$

Theorem 4.6. *If a language is recognized by a C-PRA, it is recognized by C-PRA with probability* $1 - \varepsilon$.

Lemma 4.7. *If a language* L_1 *is recognizable with probability greater than* $\frac{2}{3}$ *and a language* L_2 *is recognizable with probability greater than* $\frac{2}{3}$ *then languages* $L_1 \cap L_2$ *and* $L_1 \cup L_2$ *are recognizable with probability greater than* $\frac{1}{2}$.

Theorem 4.8. *The class of languages recognized by C-PRA is closed under intersection, union and complement.*

Theorem 4.9. *The class of languages recognized by C-PRA is closed under inverse homomorphisms.*

Corollary 4.10. *The class of languages recognized by C-PRA is closed under word quotient.*

Separate question regarding necessity of end-markers is considered in [GK 02]. It has been proven that C-PRA automata without end-markers recognize the same class of languages as C-PRA automata with both end-markers. Thus if C-PRA without end-markers are considered, closure under word quotient remains true.

Theorem 4.11. *For every natural positive n, a language $L_n = a_1^* a_2^* \dots a_n^*$ is recognizable by some C-PRA with alphabet $\{a_1, a_2, \dots, a_n\}$.*

Corollary 4.12. *Quantum finite automata with mixed states (model of Nayak, [N 99]) recognize $L_n = a_1^* a_2^* \dots a_n^*$ with probability $1 - \varepsilon$.*

Proof. This comes from the fact, that matrices $V_{a_1}, V_{a_2}, \dots, V_{a_n}$ from the proof of Theorem 4.11 all have unitary prototypes (see Definition 2.12). □

4.2.3 Languages not recognizable by C-PRA

In this section we will show the results obtained from [GK 02], that regular languages which minimal deterministic automaton contain certain forbidden constructions can not be recognizable by 1-way C-PRA. We start by definition of these "forbidden" constructions.

Definition 4.13. *We say that a regular language is of Type 0 (Figure 4.1) if the following is true for the minimal deterministic automaton recognizing this language: Exist three states q, q_1, q_2, exist words x, y such that*

1) $q_1 \neq q_2$;
2) $qx = q_1, \; qy = q_2$;
3) $q_1 x = q_1, \; q_2 y = q_2$;
4) $\forall t \in (x, y)^* \; \exists t_1 \in (x, y)^* \; q_1 t t_1 = q_1$;
5) $\forall t \in (x, y)^* \; \exists t_2 \in (x, y)^* \; q_2 t t_2 = q_2$.

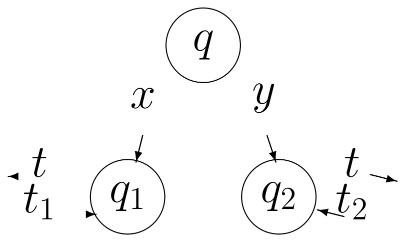

Figure 4.1: Type 0 construction

Definition 4.14. *We say that a regular language is of Type* 1 *(Figure 4.2) if the following is true for the minimal deterministic automaton recognizing this language: Exist two states q_1, q_2, exist words x, y such that*
1) $q_1 \neq q_2$;
2) $q_1x = q_2,\ q_2x = q_2$;
3) $q_2y = q_1$.

Definition 4.15. *We say that a regular language is of Type* 2 *(Figure 4.3) if the following is true for the minimal deterministic automaton recognizing this language: Exist three states q, q_1, q_2, exist words x, y such that*
1) $q_1 \neq q_2$;
2) $qx = q_1,\ qy = q_2$;
3) $q_1x = q_1,\ q_1y = q_1$;
4) $q_2x = q_2,\ q_2y = q_2$.

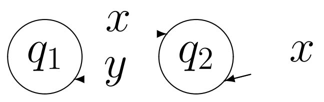

Figure 4.2: Type 1 construction

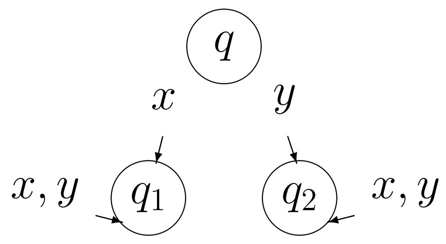

Figure 4.3: Type 2 construction

Type 1 languages are exactly those languages that violate the partial order condition of [BP 99].

The following two theorems illustrate the relationship between Type 0, Type 1 and Type 2 languages.

Theorem 4.16. *A regular language is of Type* 0 *iff it is of Type* 1 *or Type* 2.

Theorem 4.17. *A regular language L is of Type* 1 *iff L^R is of Type* 2.

Remark 4.18. Both C-DRA and C-QFA-P (see Section 5) recognize exactly the regular languages for which the corresponding minimal deterministic finite automata do not contain the following construction ([HS 66, BP 99]), denoted henceforth as Type A construction (Figure 4.4): Exist two states q_1, q_2, exist words x, y such that
1) $q_1 \neq q_2$;
2) $q_1x = q_2,\ q_2x = q_2$.
Similarly as in Theorem 4.17, it is possible to demonstrate that a regular language L is of Type A if and only if the language L^R is of Type A.

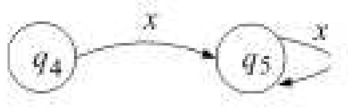

Figure 4.4: Type A construction

M.Golovkins and M. Kravtsev in [GK 02] proved that every language of Type 0 is not recognizable by any C-PRA. We give the complete proof from their work [GK 02], because this proof is very helpful for understanding our results for DH-PRA.

Definition 4.19. *By $q \xrightarrow{S} q'$, $S \subset \Sigma^*$, we denote that there is a positive probability to get to a state q' by reading a single word $\xi \in S$, starting in a state q.*

Lemma 4.20. *If a regular language is of Type 2, it is not recognizable by any C-PRA.*

Proof. Assume from the contrary, that A is a C-PRA automaton which recognizes a language $L \subset \Sigma^*$ of Type 2.

Since L is of Type 2, it is recognized by a minimal deterministic automaton D with particular three states q, q_1, q_2 such that $q_1 \neq q_2$, $qx = q_1$, $qy = q_2$, $q_1x = q_1$, $q_1y = q_1$, $q_2x = q_2$, $q_2y = q_2$, where $x, y \in \Sigma^*$. Furthermore, exists $\omega \in \Sigma^*$ such that $q_0\omega = q$, where q_0 is an initial state of D, and exists a word $z \in \Sigma^*$, such that $q_1z = q_{acc}$ if and only if $q_2z = q_{rej}$, where q_{acc} is an accepting state and q_{rej} is a rejecting state of D. Without loss of generality we assume that $q_1z = q_{acc}$ and $q_2z = q_{rej}$.

The transition function of the automaton A is determined by doubly stochastic matrices $V_{\sigma_1}, \dots, V_{\sigma_n}$. The words from the construction of Type 2 are $x = \sigma_{i_1} \dots \sigma_{i_k}$ and $y = \sigma_{j_1} \dots \sigma_{j_s}$. The transitions induced by words x and y are determined by doubly stochastic matrices $X = V_{\sigma_{i_k}} \dots V_{\sigma_{i_1}}$ and $Y = V_{\sigma_{j_s}} \dots V_{\sigma_{j_1}}$. Similarly, the transitions induced by words ω and z are determined by doubly stochastic matrices W and Z. By Corollary 2.45, exists $K > 0$, such that

$$\forall i \ (X^K)_{i,i} > 0 \text{ and } (Y^K)_{i,i} > 0. \tag{4.3}$$

Consider a relation between the states of the automaton defined as $R = \{(q_i, q_j) \mid q_i \xrightarrow{(x^K, y^K)^*} q_j\}$. By (4.3), this relation is reflexive.

Suppose exists a word $\xi = \xi_1\xi_2 \dots \xi_k$, $\xi_s \in \{x^K, y^K\}$, such that $q \xrightarrow{\xi} q'$. This means that $q \xrightarrow{\xi_1} q_{i_1}$, $q_{i_1} \xrightarrow{\xi_2} q_{i_2}, \dots, q_{i_{k-1}} \xrightarrow{\xi_k} q'$. By Corollary 2.47, since both X^K and Y^K are doubly stochastic, $\exists \xi'_k \dots \xi'_1$, $\xi'_s \in \{(x^K)^*, (y^K)^*\}$, such that $q' \xrightarrow{\xi'_k} q_{i_{k-1}}, \dots, q_{i_2} \xrightarrow{\xi'_2} q_{i_1}$, $q_{i_1} \xrightarrow{\xi'_1} q$, therefore $q' \xrightarrow{\xi'} q$, where $\xi' \in (x^K, y^K)^*$. So the relation R is symmetric.

Surely R is transitive. Therefore all states of A may be partitioned into equivalence classes $[q_0], [q_{i_1}], \dots, [q_{i_n}]$. Let us renumber the states of A in such a way, that states from one equivalence class have consecutive numbers. First come the states in $[q_0]$, then in $[q_{i_1}]$, etc.

Consider the word $x^K y^K$. The transition induced by this word is determined by a doubly stochastic matrix $C = Y^K X^K$. We prove the following proposition. States q_a and q_b are in one equivalence class if and only if $q_a \to q_b$ with matrix C. Suppose $q_a \to q_b$. Then $(q_a, q_b) \in R$, and q_a, q_b are in one equivalence class. Suppose q_a, q_b are in one equivalence class. Then

$$q_a \xrightarrow{\xi_1} q_{i_1}, q_{i_1} \xrightarrow{\xi_2} q_{i_2}, \dots, q_{i_{k-1}} \xrightarrow{\xi_k} q_b, \text{ where } \xi_s \in \{x^K, y^K\}. \tag{4.4}$$

By (4.3), $q_i \xrightarrow{x^K} q_i$ and $q_j \xrightarrow{y^K} q_j$. Therefore, if $q_i \xrightarrow{x^K} q_j$, then $q_i \xrightarrow{x^K y^K} q_j$, and again, if $q_i \xrightarrow{y^K} q_j$, then $q_i \xrightarrow{x^K y^K} q_j$. That transforms (4.4) to

$$q_a \xrightarrow{(x^K y^K)^t} q_b, \text{ where } t > 0. \tag{4.5}$$

We have proved the proposition.

By the proved proposition, due to the renumbering of states, matrix C is a block diagonal matrix, where each block corresponds to an equivalence class of the relation R. Let us identify these blocks as $C_0, C_1, \dots, C_n$. By (4.3), a Markov chain with matrix C is aperiodic. Therefore each block C_r corresponds to an aperiodic irreducible doubly stochastic Markov chain with states $[q_{i_r}]$. By Corollary 2.43, $\lim\limits_{m\to\infty} C^m = J$, J is a block diagonal matrix, where for each $(p \times p)$ block C_r $(C_r)_{i,j} = \frac{1}{p}$. Relation $q_i \xrightarrow{(y^K)^*} q_j$ is a subrelation of R, therefore Y^K is a block diagonal matrix with the same block ordering and sizes as C and J. (This does not eliminate possibility that some block of Y^K is constituted of smaller blocks, however.) Therefore $JY^K = J$, and $\lim\limits_{m\to\infty} Z(Y^K X^K)^m W = \lim\limits_{m\to\infty} Z(Y^K X^K)^m Y^K W = ZJW$. So

$$\forall \varepsilon > 0\ \exists m\ \left\| \Big(Z(Y^K X^K)^m W - Z(Y^K X^K)^m Y^K W \Big) Q_0 \right\| < \varepsilon. \tag{4.6}$$

However, by construction of Type 2, $\forall k\ \forall m\ \omega(x^k y^k)^m z \in L$ and $\omega y^k (x^k y^k)^m z \notin L$. This requires existence of $\varepsilon > 0$, such that

$$\forall m\ \left\| \Big(Z(Y^K X^K)^m W - Z(Y^K X^K)^m Y^K W \Big) Q_0 \right\| > \varepsilon. \tag{4.7}$$

This is a contradiction. $\square$

Lemma 4.21. *If a regular language is of Type* 1*, it is not recognizable by any C-PRA.*

Proof. Proof is nearly identical to that of Lemma 4.20.

Consider a C-PRA which recognizes the language L of Type 1. We prove that for words x, y exists constant K, such that for every ε exists m, such that for two words $\xi_1 = \omega(x^K(xy)^K)^m z$ and $\xi_2 = \omega(x^K(xy)^K)^m x^K z$, $|p_{\xi_1} - p_{\xi_2}| < \varepsilon$.

We denote The transition function of the automaton A is determined by doubly stochastic matrices $V_{\sigma_1}, \dots, V_{\sigma_n}$. The words from the construction of Type 2 are $x = \sigma_{i_1} \dots \sigma_{i_k}$ and $y = \sigma_{j_1} \dots \sigma_{j_s}$. The transitions induced by words x and y are determined by doubly stochastic matrices $X = V_{\sigma_{i_k}} \dots V_{\sigma_{i_1}}$ and $Y = V_{\sigma_{j_s}} \dots V_{\sigma_{j_1}}$. Similarly, the transitions induced by words ω and z are determined by doubly stochastic matrices W and Z.

By Corollary 2.45 we can select such K that $X^K_{ii} > 0$ and $(YX)^K_{ii} > 0$ for every i. Matrix $C = (YX)^K X^K$ corresponds to reading of $x^K(xy)^K$. We consider a relation between the states of the automaton defined as $R = \{(q_i, q_j) \mid q_i \xrightarrow{(x^K(xy)^K)^*} q_j\}$. This relation by Corollary 2.50 divides states into equivalence classes.

$q \xrightarrow{(x^K)^*} q'$ is subrelation of R. To show that rewrite $q \xrightarrow{(x^K)^*} q'$ as sequence $q \xrightarrow{x^K} q_{i_1}, \dots, q_{i_{k-2}} \xrightarrow{x^K} q_{i_{k-1}}, q_{i_{k-1}} \xrightarrow{x^K} q'$. As K selected so that $q_j \xrightarrow{(xy)^K} q_j$ for any j then we can substitute x^K with $x^K(xy)^K$ at each step $q \xrightarrow{x^K(xy)^K} q_{i_1}, \dots, q_{i_{k-2}} \xrightarrow{x^K(xy)^K} q_{i_{k-1}}, q_{i_{k-1}} \xrightarrow{x^K(xy)^K} q'$, getting $q \xrightarrow{(x^K(xy)^K)^*} q'$. Thus q_i and q_j are in one equivalence class in respect to R.

Due to the renumbering of states, matrix C is a block diagonal matrix, where each block corresponds to an equivalence class of the relation R. Let us identify these blocks as $C_0, C_1, \dots, C_n$. By (4.3), a Markov chain with matrix C is aperiodic. Therefore each block C_r corresponds to an aperiodic irreducible doubly stochastic Markov chain with states $[q_{i_r}]$. By Corollary 2.43, $\lim\limits_{m\to\infty} C^m = J$, J is a block diagonal matrix, where for each $(p \times p)$ block C_r $(C_r)_{i,j} = \frac{1}{p}$. As relation $q_i \xrightarrow{(x^K)^*} q_j$ is a subrelation of R, therefore X^K is a block diagonal matrix with the same block ordering and sizes as C and J. (This does not eliminate possibility that some block of X^K is constituted of smaller blocks, however.) Therefore $JX^K = J$, and $\lim\limits_{m\to\infty} ZX^K((YX)^K X^K)^m W = \lim\limits_{m\to\infty} Z((YX)^K X^K)^m W = ZJW$.

So

$$\forall \varepsilon > 0 \, \exists m \, \left\| \Big(Z(X^K((YX)^K X^K)^m W - Z((YX)^K X^K)^m W \Big) Q_0 \right\| < \varepsilon. \quad (4.8)$$

However, by construction of Type 1, we can select z such that $\omega(x^k(xy)^k)^m x^K z \in L$ and $\omega(x^k(xy)^k))^m z \notin L$. This requires existence of $\varepsilon > 0$, such that

$$\forall m \left\| \Big(ZX^K((YX)^K X^K)^m W - Z((YX)^K X^K)^m W \Big) Q_0 \right\| > \varepsilon. \tag{4.9}$$

This is a contradiction.

□

Theorem 4.22. *If a regular language is of Type* 0*, it is not recognizable by any C-PRA.*

Proof. By Lemmas 4.16, 4.20, 4.21. □

It can be easily noticed, that the Type 0 construction is a generalization of construction proposed by [AKV 01]. (Constructions of [BP 99] and [AKV 01] characterize languages, not recognized by measure-many quantum finite automata of [KW 97].)

Corollary 4.23. *Languages (a,b)*a and a(a,b)* are not recognized by C-PRA.*

Proof. Both languages are of Type 0. □

Corollary 4.24. *Class of languages recognizable by C-PRA is not closed under homomorphisms.*

Finally in [ABGKMT 06] it was proved the general class of languages recognizable by C-PRA automata.

Theorem 4.25. *The classes of languages recognizable by C-PRA and C-QFA-M are equal and coincide with the all regular languages but languages which minimal deterministic automaton contains forbidden constructions of Type* 0*.*

4.3 1-way Probabilistic Reversible DH Automata

4.3.1 Definition

Taken the definition 2.22 of Probabilistic Reversible automata we define word acceptance as specified in Definition 2.16. The set of accepting states is Q_F and set of rejecting states is $Q \setminus Q_F$, these states are halting. We define language recognition as in Definition 2.20. That completes formal definition of 1-way DH-PRA automata.

We can formulate Decide and Halt acceptance for classical automaton in an equivalent form of classical acceptance. Instead of halting once reaching the halting state we can consider that automaton continues to read input till the end of but remain in the same halting state.

We can see that transition matrixes V_{σ_k} are not doubly stochastic anymore, but for some V_{σ_k} we can enumerate states of DH-PRA in such way that :

1. q_1 ... q_k are states, from which halting states are not accessible
2. q_{k+1} ... q_{n-l} are non-halting states from which halting states are accessible,
3. q_{n-l+1} ... q_n are halting states

Then the structure of transition matrix V_{σ_k} will look so: $\begin{pmatrix} {\scriptstyle k\{DST} & O & O \\ O & a_{ij} & O \\ {\scriptstyle l\{O} & a_{ij} & I \end{pmatrix}$.

where

- DST - doubly stochastic matrix,
- I - unit matrix,
- $\forall k+1 \leq j \leq n-l : \sum\limits_{i=1}^{n} \alpha_{ij} = 1$ (it's still stochastic)
- $\forall k+1 \leq i \leq n-l : \sum\limits_{j=1}^{n} \alpha_{ij} <= 1$ (as originated from double stochastic matrix where sum in each row is one)

According to definitions 2.32 states q_{k+1} ... q_{n-l} are transient and states q_1 ... q_k and q_{n-l+1} ... q_n are recurrent, with q_{n-l+1} ... q_n being absorbing for Markov chain induced by transitions V_{σ_k}. Note that for different letters of the alphabet σ_k the numbering of non halting states will be different.

Let us call matrix of such type DH-stochastic matrix.

Lemma 4.26. *Let A a $k \times m$ matrix such that $m < k$, where the sum of elements in any column is one, and the sum of elements in any row is less or equal than one. Then exists a $k \times (k-m)$ matrix B, such that $(A\ B)$ is doubly stochastic.*

Proof. Let s_i the sum of elements of the i-th row of A. Let $B = \begin{pmatrix} \frac{1-s_1}{k-m} & \dots & \frac{1-s_1}{k-m} \\ \dots & \dots & \dots \\ \frac{1-s_k}{k-m} & \dots & \frac{1-s_k}{k-m} \end{pmatrix}$.

Now the sum of elements in any column of B is $\frac{k-\sum\limits_{i=1}^{k} s_i}{k-m} = 1$. Hence $(A\ B)$ is stochastic. The sum of the i-th row of $(A\ B)$ is $s_i + (k-m)\frac{1-s_i}{k-m} = 1$. Hence $(A\ B)$ is doubly stochastic. □ □

Theorem 4.27. *A stochastic matrix S is DH-stochastic, iff exists a permutation P such that $PSP^T = \begin{pmatrix} D & 0 & 0 \\ 0 & A & 0 \\ 0 & B & I \end{pmatrix}$, where D is a $k \times k$ doubly stochastic matrix with $k \geq 0$, A - $l \times l$ matrix with $l \geq 0$, I - $m \times m$ unit matrix with $m > 0$, and the sum of elements in any row in the matrices A and B is less or equal than one.*

Proof. To prove the theorem, it is sufficient to show that any matrix of the form $\begin{pmatrix} D & 0 & 0 \\ 0 & A & 0 \\ 0 & B & I \end{pmatrix}$, specified in the theorem, can be obtained from a doubly stochastic matrix. This indeed holds, since by Lemma 4.26, the matrix $\begin{pmatrix} D & 0 \\ 0 & A \\ 0 & B \end{pmatrix}$ can be complemented with new columns to obtain a doubly stochastic matrix. □ □

Certainly transformation that corresponds to the reading of a sequence of letters also is described by a DH-stochastic matrix.

Lemma 4.28. *For any $\sigma_s, \sigma_t \in \Sigma$: $V_{\sigma_s} \cdot V_{\sigma_t}$ - is also DH-stochastic matrix.*

Proof. Follows from the matrix manipulation. To show that for states from which halting states are not accessible the matrix is double stochastic observe that no sum in the row can exceed 1 still and also can not be less then 1 as otherwise summing by rows and columns would give different results. □

Note that the transient states in $V_{\sigma_s} \cdot V_{\sigma_t}$ may be different from the transient states in V_{σ_s} and in V_{σ_t}.

4.3.2 On Class of languages recognizable by 1-way DH-PRA

To prove forbidden constructions for DH-PRA we need to consider the behavior of the Markov chain induced by transition V_{σ_k} in long run.

Before looking at forbidden constructions for DH-PRA we need to examine how accessibility property stands for Markov chains with DH-stochastic matrixes and also how it changes if we consider several letters.

Obviously for recurrent states of DH-stochastic Markov chain it is still true that accessibility is class property by Lemma 2.50, that will be used without any further references in the proof of the main theorem of this section.

Theorem 4.29. *Given Markov chains with DH-stochastic matrixes A and B, there exists K such that for Markov chain with matrix $A^K B^K$ the following holds:*
A. any transient state with respect to A or B is transient for $A^K B^K$;
B. for any two recurrent states q_1 and q_2 that are in the same equivalence class regarding accessibility in A, either both are transient for $A^K B^K$ or both are recurrent and in the same equivalence class for $A^K B^K$;
C. for any two recurrent states q_1 and q_2 that are in the same equivalence class regarding accessibility in B, either both are transient for $A^K B^K$ or both are recurrent and in the same equivalence class for $A^K B^K$.

Proof. We take K to be

- $K > n$, where n is a number of states;
- K is a multiple of $K_1 * n$ where $(A^{K_1})_{i,i} > 0$ for all recurrent states of A;
- K is a multiple of $K_2 * n$ where $(B^{K_2})_{i,i} > 0$ for all recurrent states in B.

Recurrent states in A and B in general could be different. We can select such K_1 and K_2 by Corollary 2.45. We selected K to satisfy the conditions of Lemma 2.48 for both A and B.

For any transient state q of any DH-stochastic matrix D of size n there is some absorbing state q' such that $q \xrightarrow{D^K} q'$ (will be accessible by D^K in 1 step). For any absorbing state q' it holds $q' \xrightarrow{D} q'$. For any states q_1, q_2 and q_3 and DH-stochastic matrixes C and D, $q_1 \xrightarrow{C} q_2, q_2 \xrightarrow{D} q_3 \Longrightarrow q_1 \xrightarrow{DC} q_3$.

A. If a state q is transient for B, an absorbing state q' exists such that $q \xrightarrow{B^K} q'$, so $q' \xrightarrow{A^K} q'$ implies $q \xrightarrow{A^K B^K} q'$.

Assume a state q is transient for A but recurrent for B. As $q \xrightarrow{B^K} q$ and some absorbing state q' exists such that $q \xrightarrow{A^K} q'$, then again $q \xrightarrow{A^K B^K} q'$.

B. Let q_1 be transient for $A^K B^K$, we need to prove that q_2 is transient for $A^K B^K$:

a) if q_2 is transient for B^K then there is an absorbing state q' such that $q_2 \xrightarrow{B^K} q'$ and thus q_2 is transient for $A^K B^K$ since $q' \xrightarrow{A^K} q'$;
b) if q_2 is recurrent for B^K, then $q_2 \xrightarrow{B^K} q_2$ and as $q_2 \xrightarrow{A^K} q_1$, then $q_2 \xrightarrow{A^K B^K} q_1$. Hence q_2 is transient for $A^K B^K$.

Let q_1 and q_2 be recurrent for $A^K B^K$:
a) Assume q_1 is recurrent for B^K, as $q_1 \xrightarrow{B^K} q_1$ and $q_1 \xrightarrow{A^K} q_2$ we get $q_1 \xrightarrow{A^K B^K} q_2$, thus they are in the same equivalence class.
b) Assume q_1 is transient for B^K, then there is some absorbing state q' such that $q_1 \xrightarrow{B^K} q'$ and since $q' \xrightarrow{A^K} q'$, q_1 is transient for $A^K B^K$, leading to contradiction.

C. The case is not identical with B as matrices multiplication is not commutative.

Let q_1 be transient for $A^K B^K$, we need to prove that q_2 is transient for $A^K B^K$.
a) Assume q_1 is transient for A^K, then there is some absorbing state q' such that $q_1 \xrightarrow{A^K} q'$. So $q_2 \xrightarrow{B^K} q_1$ implies $q_2 \xrightarrow{A^K B^K} q'$.
b) Assume q_1 is recurrent for A^K, $q_1 \xrightarrow{A^K} q_1$. As $q_2 \xrightarrow{B^K} q_1$ it follows that $q_2 \xrightarrow{A^K B^K} q_1$ and therefore q_2 is transient for $A^K B^K$.

Let q_1 and q_2 be recurrent for $A^K B^K$.
a) Assume q_2 is recurrent for A^K. Since $q_1 \xrightarrow{B^K} q_2$ and $q_2 \xrightarrow{A^K} q_2$, we get $q_1 \xrightarrow{A^K B^K} q_2$, therefore they are in the same equivalence class for $A^K B^K$.
b) Assume q_2 is transient for A^K. Then exists an absorbing state q' such that $q_2 \xrightarrow{A^K} q'$, so $q_1 \xrightarrow{B^K} q_2$ implies $q_1 \xrightarrow{A^K B^K} q'$ and thus q_1 is transient for $A^K B^K$ leading to contradiction. □

The subsequent lemma and corollary illustrate Theorem 4.29 in terms of transition matrices:

Lemma 4.30. *Given a DH-stochastic matrix A, there exists K such that* $\lim_{n\to\infty} A^{Kn} = \begin{pmatrix} D' & 0 & 0 \\ 0 & 0 & 0 \\ 0 & a_{ij} & I \end{pmatrix}$, *where D' is a block diagonal $k_r \times k_r$ matrix such that each block is a doubly stochastic matrix with every element equal to $\frac{1}{k_i}$, where k_i is the size of the block.*

Proof. Take K such that $A^K_{i,i} > 0$ for all recurrent states (possible by Lemma 2.44). Rows filled entirely by zeros correspond to transient states. As non-halting recurrent states in A^K form a doubly stochastic matrix then they can be split into equivalence classes with respect to communication property

(Lemma 2.50) and each block diagonal submatrix corresponds to states in one equivalence class. Values in these submatrices are determined by Corollary 2.43. □ □

Corollary 4.31. *Given DH-stochastic matrices A and B, there exists K such that*

A. $\lim_{n\to\infty}(A^K B^K)^n = J = \begin{pmatrix} D & 0 & 0 \\ 0 & 0 & 0 \\ 0 & R & I \end{pmatrix}$*, where D is a block diagonal $k_s \times k_s$, $s \le r$, matrix such that each block is a doubly stochastic matrix with every element equal to $\frac{1}{k_i}$, where k_i is the size of the block.*

B. $A^K J = J$.

C. $JA^K = \begin{pmatrix} D & 0 & 0 \\ 0 & 0 & 0 \\ 0 & R' & I \end{pmatrix}$

Proof. Follows directly from Theorem 4.29 and Lemma 4.30 if we observe the structure of the matrix for A and B. Difference between items B and C is that if we apply A^K first we can get some different probability distribution for halting states, however if we apply A^K after J then it is the same as to apply J. □ □

Definition 4.32. *By $q \xrightarrow{S} q'$, $S \subset \Sigma^*$, we denote that there is a positive probability to get to a state q' by reading a single word $\xi \in S$, starting in a state q.*

Now we are ready to show the class of languages recognized by DH-PRA.

It is easy to see that the class of languages recognized by C-PRA is a proper subclass of languages recognized by DH-PRA. Indeed, by [FGK 04], we may assume that C-PRA does not use any end-markers. Given a C-PRA $\mathcal{A}$, for any final state q_i add an accepting halting state q_i^h, and for any non-final state q_j, add a rejecting halting state q_j^h. Add the final end-marker \$ with transitions $q_i\$ = q_i^h$, $q_i^h\$ = q_i$, $q_j\$ = q_j^h$, $q_j^h\$ = q_j$. $V_\$$ is doubly stochastic. The addition of end-marker ensures that any input word accepted (rejected) by $\mathcal{A}$ is also accepted (rejected) by the DH-PRA with the same probability.

There exist languages recognized by DH-PRA, and not recognized by C-PRA.

Example 4.33. The language a(a,b)* known not to be recognizable by C-PRA is recognizable by DH-PRA.

In this section we will prove that regular languages which minimal deterministic automaton contain certain forbidden constructions can not be

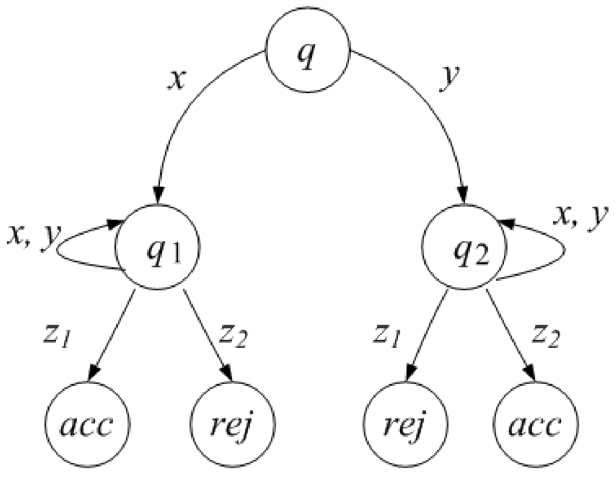

Figure 4.5: Type 3 construction

recognizable by 1-way DH-PRA. We start by definition of these "forbidden" constructions, that are quite similar to ones defined for C-PRA.

First class of languages to be considered will be Type 1 described in C-PRA section (see Figure 4.2).

Second class will be a modification of Type 2.

Definition 4.34. *We say that a regular language is of Type* 3 *(Figure 4.5) if a regular language is of Type* 2 *and additional conditions hold for states* q_1, q_2.
There exist 2 words z_1 *and* z_2 *such that*

1. *reading of* z_1 *when being in* q_1 *leads to accepting state and reading* z_1 *when being in* q_2 *leads to not accepting state;*
2. *reading of* z_2 *when being in* q_2 *leads to accepting state and reading* z_2 *when being in* q_1 *leads to not accepting state.*

Theorem 4.35. *If a regular language is of Type* 3 *then it is not recognizable by any DH-PRA.*

Proof. Assume from the contrary, that A is a DH-PRA automaton which recognizes a language $L \subset \Sigma^*$ of Type 3.

Since L is of Type 3, it is recognized by a minimal deterministic automaton $\mathcal{D}$ with particular three states q, q_1, q_2 such that $q_1 \neq q_2$, $qx = q_1$, $qy = q_2$, $q_1x = q_1$, $q_1y = q_1$, $q_2x = q_2$, $q_2y = q_2$, where $x, y \in \Sigma^*$. Furthermore, there exists $\omega \in \Sigma^*$ such that $q_0\omega = q$, where q_0 is an initial state of $\mathcal{D}$, and there exist words $z_1 \in \Sigma^*$, $z_2 \in \Sigma^*$, such that $q_1z_1 = q^1_{acc}$ and $q_1z_2 = q^1_{rej}$,

$q_2z_1 = q^2_{rej}$ and $q_2z_2 = q^2_{acc}$, where q^1_{acc}, q^2_{acc} are final states and q^1_{rej}, q^2_{rej} are non-final states of $\mathcal{D}$.

The transition function of the automaton A is determined by DH-stochastic matrices $V_{\sigma_1}, \dots, V_{\sigma_n}$. The words from the construction of Type 3 are $x = \sigma_{i_1} \dots \sigma_{i_k}$ and $y = \sigma_{j_1} \dots \sigma_{j_s}$. The transitions induced by words x and y are determined by DH-stochastic matrices $X = V_{\sigma_{i_k}} \dots V_{\sigma_{i_1}}$ and $Y = V_{\sigma_{j_s}} \dots V_{\sigma_{j_1}}$. Similarly, the transitions induced by words ω, z_1 and z_2 are determined by DH-stochastic matrices W, Z_1 and Z_2.

Let us select 2 words x_1 and x_2 of the form $x_1 = \omega y^K (x^K y^K)^m$ and $x_2 = \omega (x^K y^K)^m$.

We select K as in Theorem 4.29. Using related Corollary 4.31 we get that $\lim\limits_{m\to\infty} (Y^K X^K)^m$ converges to some matrix J of the form described in Corollary 4.31 and that J differs from JY^K only in transitions from transient to absorbing states for $(Y^K X^K)^m$.

That means that after reading $x_1 = \omega y^K (x^K y^K)^m$ and $x_2 = \omega (x^K y^K)^m$ from the initial state we will get arbitrary close probability distributions for non-halting states, but possibly different probability distributions for the halting states. Let ρ_{a1} be accepting probability and ρ_{r1} rejecting probability after reading x_1. (So at that moment the automaton remains in a non-halting state with probability $1 - \rho_{a1} - \rho_{r1}$). Let ρ_{a2} be accepting probability and ρ_{r2} rejecting probability after reading x_2. Consider reading z_1 after x_1 that needs to be rejected and x_2 that needs to be accepted. As distributions on non-halting states before reading z_1 are arbitrary close, and reading any word cannot significantly increase their difference, at that moment the word $x_1 z_1 \notin L$ is accepted with probability $\rho_{a1} + c_1$, and the word $x_2 z_1 \in L$ is accepted with probability $\rho_{a2} + c_2$, where c_1 and c_2 are arbitrary close nonnegative values. Due to the assumption that L is recognized with bounded error, we get $\rho_{a1} < \rho_{a2}$. On the other hand, consider reading z_2 after x_1 that needs to be accepted and x_2 that needs to be rejected. In a similar fashion, we get that $\rho_{a2} < \rho_{a1}$. This is a contradiction. □

Theorem 4.36. *If a regular language is of Type* 1*, it is not recognizable by any DH-PRA.*

Proof. Assume from the contrary, that A is a DH-PRA automaton which recognizes a language $L \subset \Sigma^*$ of Type 1.

Since L is of Type 1, it is recognized by a deterministic automaton $\mathcal{D}$ which has two states q_1, q_2 such that $q_1 \neq q_2$, $q_1 x = q_2$, $q_2 y = q_1$, $q_2 x = q_2$ where $x, y \in \Sigma^*$. Furthermore, exists $\omega \in \Sigma^*$ such that $q_0 \omega = q_1$, where q_0 is an initial state of $\mathcal{D}$, and exists a word $z \in \Sigma^*$, such that $q_1 z = q_{acc}$ if and only if $q_2 z = q_{rej}$, where q_{acc} is an accepting state and q_{rej} is a rejecting state of $\mathcal{D}$.

The transition function of the automaton A is determined by DH-stochastic matrices $V_{\sigma_1}, \ldots, V_{\sigma_n}$. The words $x = \sigma_{i_1} \ldots \sigma_{i_k}$ and $y = \sigma_{j_1} \ldots \sigma_{j_s}$. The transitions induced by words x and y are determined by DH-stochastic matrices $X = V_{\sigma_{i_k}} \ldots V_{\sigma_{i_1}}$ and $Y = V_{\sigma_{j_s}} \ldots V_{\sigma_{j_1}}$. Similarly, the transitions induced by word ω is determined by DH-stochastic matrix W.

Let us select two words x_1 and x_2 of the form $x_1 = \omega(x^K(xy)^K)^m$ and $x_2 = \omega(x^K(xy)^K)^m x^K$.

We will show that for any ε we can select K and m such that $|p_{x_1} - p_{x_2}| < \varepsilon$. Then as $x_1 z \in L$ and $x_2 z \notin L$ we get a contradiction.

We select K as in Theorem 4.29. Using related Corollary 4.31 we get that $\lim_{m\to\infty} ((YX)^K X^K)^m$ converges to some matrix J of the form described in Corollary 4.31, for which the equality $X^K J = J$ stands.

That means that after reading $x_1 = \omega(x^K(xy)^K)^m$ and $x_2 = \omega(x^K(xy)^K)^m x^K$ we will get arbitrary close probability distributions that gives us required contradiction. Or formally,

$\lim_{m\to\infty} ZX^K((YX)^K X^K)^m W = \lim_{m\to\infty} Z((YX)^K X^K)^m W = ZJW$. So

$$\forall \varepsilon > 0 \; \exists m \; \left\| \Big(Z(X^K((YX)^K X^K)^m W - Z((YX)^K X^K)^m W \Big) Q_0 \right\| < \varepsilon. \tag{4.10}$$

As we can select z such that $\omega(x^k(xy)^k)^m x^K z \in L$ and $\omega(x^k(xy)^k))^m z \notin L$, that requires existence of $\varepsilon > 0$, such that

$$\forall m \; \left\| \Big(ZX^K((YX)^K X^K)^m W - Z((YX)^K X^K)^m W \Big) Q_0 \right\| > \varepsilon. \tag{4.11}$$

□

4.3.3 Closure properties of the DH-PRA automata

In this section we prove that the class of languages recognizable by DH-PRA automata is not closed by the union. In [AKV 01] there is proposed a language which is union of languages recognizable by QFA-DH which is not recognisable by QFA-DH, we basically follow their proof. Although forbidden construction for QFA-DH considered in [AKV 01] is different from considered above we find also Type 3 forbidden construction in this language.

Theorem 4.37. *There are two languages L_2 and L_3 which are recognizable by DH-PRA, but the union of them $L_1 = L_2 \cup L_3$ is not recognizable by DH-PRA.*

Proof. Let L_1 be the language consisting of all words that start with any number of letters a and after first letter b (if there is one) there is an odd number of letters a. Its minimal automaton G_1 is shown in Fig. 4.6.

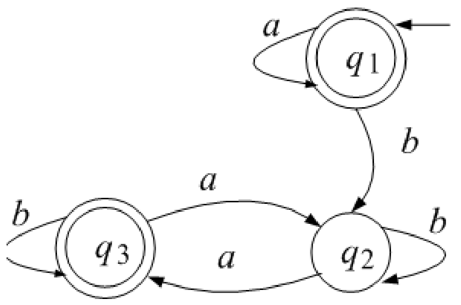

Figure 4.6: Minimal Automaton of L_1

This language satisfies the conditions of Theorem 4.35 (g_1, g_2 and q_3 of Theorem 4.35 are just g_1, g_2 and q_3 of G_1. x, y, z_1, z_2 are b, aba, ab and b.) Hence it cannot be recognized by a DH-PRA. Consider two other languages L_2 and L_3 defined as follows. L_2 consists of all words which start with an even number of letters a and after firs letter b (if there is one) there is an odd number of letters a. L_3 consists of all wards which start with an odd number of letters a and after firs letter b (if there is one) there is an odd number of letters a. It is easy to see that $L_1 = L_2 \cup L_3$. The minimal automatons G_2 and G_3 are shown on Fig. 4.7 and Fig. 4.8.

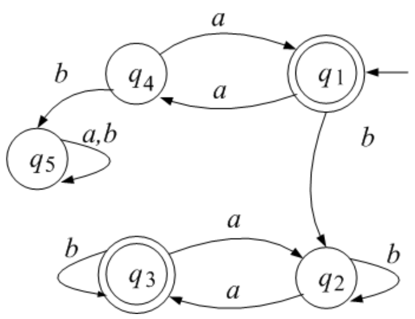

Figure 4.7: Minimal Automaton of L_2 "even"

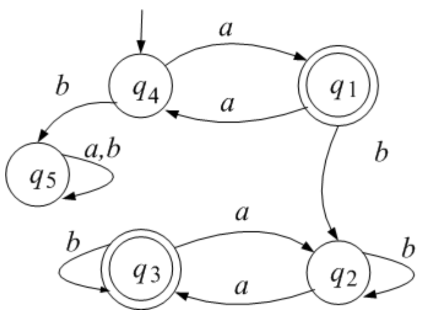

Figure 4.8: Minimal Automaton of L_3 "odd"

We construct two DH-PRA automata K_2 and K_3 which recognize languages L_2 and L_3. The automaton K_2 is obtained from the automaton G_2 (see Fig. 4.7) in a simple way: just splitting the automaton into two disconnected sub-automata, by replacing transition $q_1 \xrightarrow{b} q_2$ with the transition $q_1 \xrightarrow{b} q_{acc}$ and the transition $q_1 \xrightarrow{b} q_{rej}$, each with the probability 1/2. The initial state with the probability 2/3 is q_1 and with the probability 1/3 is q_2. This transformation of the automaton G_2 can be written as follows:

- the states q_1,q_2,q_3,q_4 are non-halting states;

- the state q_5 is a rejecting state;
- the transition $q_1 \xrightarrow{b} q_2$ is replaced with two transitions: $q_1 \xrightarrow{b} q_{acc}$ and $q_1 \xrightarrow{b} q_{rej}$, each with probability 1/2;
- reading right end-marker from the states q_2 and q_4 leads to two additional rejecting states, and from the states q_1 and q_3 leads to two additional accepting states;
- create a new initial state q_0. Reading left end-marker in the q_0 leads to the q_1 with probability 2/3 and to the q_2 with probability 1/3.

Formally we define the automaton K_2 to comply with the DH-PRA definition. The automaton consists of 12 states: g_1, g_2, g_3, g_4, g_5, g_6, g_7, g_8, g_9, g_{10}, g_{11} and g_{12}, where $Q_{non} = \{g_1, g_2, g_3, g_4, q_{12}\}$, $Q_{rej} = \{g_5, g_6, g_7, g_8\}$ and $Q_{acc} = \{g_9, g_{10}, g_{11}\}$. The starting state of K_2 is q_{12}. The transition matrixes V_k, V_a, V_b and $V_\$$ are defined as follows:

$$V_k = \begin{pmatrix} \frac{1}{3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{2}{3} \\ 0 & \frac{2}{3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{3} \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ \frac{2}{3} & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

$$V_a = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

$$V_b = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

$$V_\$ = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

1. After reading the left endmarker k K_2 with probability $\frac{2}{3}$ is in the state q_1 and with probability $\frac{1}{3}$ is in the state q_2. G_2 is in the starting state q_1.

2. After reading even number of letters a K_2 with probability $\frac{2}{3}$ is in the state q_1 and with probability $\frac{1}{3}$ is in the state q_2.

3. After reading odd number of letters a K_2 with probability $\frac{2}{3}$ is in the state q_4 and with probability $\frac{1}{3}$ is in the state q_3.

4. If after reading an odd number of the letter a K_2 receives the letter b or right endmarker then it rejects input with probability at least $\frac{2}{3}$ (from the state q_4 by reading b or right endmarker K_2 goes to rejecting state)

5. If after reading even number of letters a K_2 receives right endmarker then it accepts the input with probability $\frac{2}{3}$

6. If after reading even number of letters a K_2 receives letter b then with probability $\frac{1}{3}$ K_2 passes to accepting state, with probability $\frac{1}{3}$ K_2 passes to rejecting state, and probability $\frac{1}{3}$ K_2 passes to the non-final state q_2

7. By reading the letter a automaton K_2 passes from q_2 to q_3 or back. By reading the letter b automaton K_2 passes from q_2 to q_2 and from q_3 to q_3, so receiving right endmarker in the state q_3 the input is accepted with total probability $\frac{2}{3}$ and receiving right endmarker in the state q_2 the input is rejected with total probability $\frac{2}{3}$.

   This shows that K_2 accepts the language L_2 with probability $\frac{2}{3}$. Similarly we construct K_3 that accepts L_3 with probability $\frac{2}{3}$.

   Thus we have shown that there are two languages L_2 and L_3 which are recognizable by DH-PRA with probability $\frac{2}{3}$, but the union of them $L_1 = L_2 \cup L_3$ is not recognizable by DH-PRA.

$\square$

Chapter 5

Conclusion

In this thesis we investigated some problems concerning possible speedup of quantum algorithms over classical ones. We considered the two main problems in this thesis. In the first part of the thesis we considered quantum query complexity of Boolean functions and gave an overview of the lower bound techniques. We have introduced the problem of finding cycle in the graph. We provided the quantum algorithm solving this problem better than any classical analogue. The overall query complexity of the algorithm is $O(n\sqrt{n})$ while it is $O(n^2)$ in classical case (where n is a size of adjacency matrix representing the graph). We also showed that the algorithm is optimal, using the adversary lower bound technique ([AM 02]).

In the second part of our work we completed the research started by M. Golovkins and M. Kravtsev [GK 02] by investigating properties of probabilistic reversible Decide and Halt automata. The DH-PRA behave more like measure-many quantum automata [KW 97], as they halt when entering accepting or rejecting states. In the thesis we showed a general class of regular languages, not recognizable by DH-PRA. This class is identical to a class not recognizable by MM-QFA [AKV 01] (and similar to the class of languages, not recognizable by C-PRA [GK 02]. We also proved that the class of languages recognizable by DH-PRA is not closed under union. The one open problem still remains: we showed the class of languages for which we can not prove whether it is recognizable by DH-PRA or not. So we still unable to prove or disprove that class of languages recognizable by DH-PRA is likely to include the one recognizable by MM-QFA or these classes are equal.

We propose the following classification for one-way reversible finite automata:

	C-Automata	DH-Automata
Deterministic Automata	Permutation Automata [HS 66, T 68] (C-DRA)	Reversible Finite Automata [AF 98] (DH-DRA)
Quantum Automata with Pure States	Measure-Once Quantum Finite Automata [MC 00] (C-QFA-P)	Measure-Many Quantum Finite Automata [KW 97] (DH-QFA-P)
Probabilistic Automata	Probabilistic Reversible C-Automata (C-PRA)	Probabilistic Reversible DH-Automata (DH-PRA)
Quantum Finite Automata with Mixed States	"Latvian" QFA [ABGKMT 06] (C-QFA-M)	Enhanced Quantum Finite Automata [N 99] (DH-QFA-M)

Language class problems have been solved for C automata and DH-DRA, for the remaining types they are still open. Every type of DH-automata may simulate the corresponding type of C-automata.

The following relation among language classes also presents interest, question marks denoting conjectures:

$$\text{C-DRA} = \text{C-QFA-P} \subset \text{C-PRA} = \text{C-QFA-M}$$
$$\text{DH-DRA} \subset \text{DH-QFA-P} \overset{?}{\subset} \text{DH-PRA} \overset{?}{\subset} \text{DH-QFA-M}$$

Note that language classes recognized by C-automata are closed under boolean operations, while DH-automata are not.

Bibliography

[ABFK 99] A. Ambainis, R. Bonner, R. Freivalds, A. Ķikusts. Probabilities to Accept Languages by Quantum Finite Automata. *COCOON 1999, Lecture Notes in Computer Science*, 1999, Vol. 1627, pp. 174-183.
`http://arxiv.org/abs/quant-ph/9904066`

[ABGKMT 06] A. Ambainis, M. Beaudry, M. Golovkins, A. Kikusts, M. Mercer, D. Thrien. Algebraic results on quantum automata. Theory of Computing Systems, 39(2006), pages 165-188. Earlier version in STACS'04 `http://www.math.uwaterloo.ca/%7Eambainis/ps/lqfa.ps`

[AF 98] A. Ambainis, R. Freivalds. 1-Way Quantum Finite Automata: Strengths, Weaknesses and Generalizations. *Proc. 39th FOCS*, 1998, pp. 332-341.
`http://arxiv.org/abs/quant-ph/9802062`

[AI 99] M. Amano, K. Iwama. Undecidability on Quantum Finite Automata. *31st STOC*, pp. 368-375, 1999.

[AKN 97] D. Aharonov, A. Kitaev and N. Nisan. Quantum circuits with mixed states. *30th Annual ACM Symposium on Theory of Computation*, 1997, pp. 2030.

[AKV 01] A. Ambainis, A. Ķikusts, M. Valdats. On the Class of Languages Recognizable by 1-Way Quantum Finite Automata. *STACS 2001, Lecture Notes in Computer Science*, 2001, Vol. 2010, pp. 75-86.
`http://arxiv.org/abs/quant-ph/0009004`

[AM 99] A. Ambainis. A better lower bound for quantum algorithms searching an ordered list. *Proceedings of FOCS99*, pp. 352-357. Also quant-ph/9902053.

[AM 02] A. Ambainis. Quantum lower bounds by quantum arguments. *Journal of Computer and System Sciences*, 64:750-767, 2002. Earlier versions at STOC00 and quant-ph/0002066.

[AM 04] A. Ambainis. Polynomial degree vs. quantum query complexity. *quant-ph/0305028v4.*

[AW 99] A. Ambainis, J. Watrous. Two-Way Finite Automata with Quantum and Classical States.
`http://arxiv.org/abs/cs.CC/9911009`

[AS 04] S. Aaronson, Y. Shi. Quantum lower bounds for the collision and the element distinctness problems. *Journal of ACM*, 51:595-605, 2004.

[BH 97] G. Brassard and P. Hyer. An exact quantum polynomialtime algorithm for Simons problem. *In Proceedings of the 5th Israeli Symposium on Theory of Computing and Systems (ISTCS97)*, pages 1223, 1997.

[BL 95] D. Boneh and R. J. Lipton. Quantum cryptanalysis of hidden linear functions (extended abstract). *In Advances in Cryptology (CRYPTO95), Lecture Notes in Computer Science*, volume 963, pages 424437. Springer, 1995.

[BP 99] A. Brodsky, N. Pippenger. Characterizations of 1-Way Quantum Finite Automata.
`http://arxiv.org/abs/quant-ph/9903014`

[BV 97] E. Bernstein, U. Vazirani. Quantum Complexity Theory. *SIAM Journal on Computing*, Vol. 26(5), pp. 1411-1473, 1997.

[BW 02] H. Buhrman, R. deWolf. Complexity measures and decision tree complexity: a survey. *Theoretical Computer Science*, 288:21-43, 2002.

[BBBV 97] C. Bennett, E. Bernstein, G. Brassard, U. Vazirani. Strengths and weaknesses of quantum computing. *SIAM Journal on Computing*, 26(3):1510-1523, 1997.

[BBCMW 01] R. Beals, H. Buhrman, R. Cleve, M. Mosca, R. de Wolf. *Quantum lower bounds by polynomials.* Journal of ACM, 48: 778-797, 2001. Earlier versions at FOCS98 and quant-ph/9802049.

[BBHT 98] M. Boyer, G. Brassard, P. Hyer, and A. Tapp. Tight bounds on quantum searching. *Fortschritte der Physik*, 46(45):493505, 1998.

[BHMT 02] G. Brassard, P. Hyer, M. Mosca and A. Tapp. Quantum amplitude amplification and estimation, *Quantum Computation and Information, Contemporary Mathematics 305*, 5374, 2002.

[BHT 98] G. Brassard, P. Hyer, and A. Tapp. Quantum counting. *In Proceedings of 25th ICALP, Lecture Notes in Computer Science*, volume 1443, pages 820831. Springer, 1998.

[BMP 03] Alberto Bertoni, Carlo Mereghetti, and Beatrice Palano. Quantum Computing: 1-Way Quantum Automata, *DLT 2003, Lecture Notes in Computer Science*, Vol. 2710, pp. 120, 2003.

[BSS 03] H. Barnum, M. Saks, M. Szegedy. Quantum decision trees and semidefinite programming. *Complexity2003*, pp. 179-193.

[C 66] E. W. Cheney, *Introduction to Approximation Theory.* New York: McGraw- Hill, 1966.

[De 85] D. Deutsch. Quantum Theory, the Church-Turing Principle and the Universal Quantum Computer. *Proceedings of the Royal Society (London)*, Vol. A-400, pp. 97-117, 1985.

[DJ 92] D. Deutsch and R. Jozsa. Rapid solution of problems by quantum computation. *In Proceedings of the Royal Society of London*, volume A439, pages 553558, 1992.

[DSa 96] C. Dürr, M. Santha. A Decision Procedure for Unitary Linear Quantum Cellular Automata. *Proc. 37th FOCS*, pp. 38-45, 1996.
`http://arxiv.org/abs/quant-ph/9604007`

[FGK 04] R. Freivalds, M. Golovkins, A. Ķikusts. On the Properties of Probabilistic Reversible Automata. *SOFSEM 2004, Proceedings Vol. 2*, pp. 75-84, MatFyzPress, Prague, 2004.

[G 02] M. Golovkins. Quantum automata and quantum computing. Doctoral thesis. 2002.

[GK 02] M. Golovkins, M. Kravtsev. Probabilistic Reversible Automata and Quantum Automata. *COCOON 2002, Lecture Notes in Computer Science*, Vol. 2387, pp. 574-583, 2002.
`http://arxiv.org/abs/cs.CC/0209018`

[GKK 05] M.Golovkins, M. Kravcevs, V. Kravcevs. On the Class of Languages Recognizable by Probabilistic Reversible Decide-and-Halt Automata. Extended Abstract. *5th int. ERATO Conference on Quantum Information Systems. Proceedings, ERATO project*, pp. 131-132, 2005.

[GKK 07] M. Golovkins, M. Kravtsev, and V. Kravcevs. On a Class of Languages Recognizable by Probabilistic Reversible Decide-and-Halt Automata. *DLT 2007 - 11th International Conference on Developments in Language Theory, Satellite Workshop on Probabilistic and Quantum Automata. Proceedings.*, TUCS General Publication No 45, pp. 37-54, 2007.

[GS 97] C. M. Grinstead and J. L. Snell. Introduction to Probability. *American Mathematical Society*, 1997.
`http://www.dartmouth.edu/~chance/teaching_aids/articles.html`

[Gr 99] J. Gruska. Quantum Computing, McGraw Hill, 439 p, 1999

[GR 96] L. K. Grover: A fast quantum mechanical algorithm for database search. *In Proc. of 28th ACM STOC*, pp. 212219, 1996.

[GR 97] L. Grover Quantum Mechanics Helps in Searching for a Needle in a Haystack Phys. Rev. Lett. 79, pp. 325328, 1997 `http://arxiv.org/abs/quant-ph/9706033`

[Gu 89] E. Gurari. An Introduction to the Theory of Computation. *Computer Science Press*, 1989.

[HMW 03] P. Hoyer, M. Mosca, AND R. deWolf: Quantum search on bounded-error inputs. *In Proc. of 30th ICALP*, pp. 291299, 2003.

[HNS 02] P. Hoyer, J. Neerbek, AND Y. Shi: Quantum complexities of ordered searching, sorting, and element distinctness. *Algorithmica, Special issue on Quantum Computation and Cryptography*, 34(4):429448, 2002.

[HS 66] J. Hartmanis and R.E.Stearns. Algebraic Structure Theory of Sequential Machines. *Prentice Hall*, 1966.

[K 04] V. Kravcevs. Quantum query algorithm complexity for graph circuit problem *6th International Baltic Conference on Databases and Information Systems. Proceedings.*, Latvian University Press, 673:66-72, 2004.

[K 02] V. Kravcevs. Boolean function computation by probabilistic and quantum decision trees. *3rd International Workshop on Quantum Computation and Learning. Proceedings, Malardaren University Press, pp. 32-41,2002.*

[Ku 03] S. Kutin. A quantum lower bound for the collision problem, quant-ph/0304162.

[KKG 06] V. Kravcevs, M. Kravtsev, and M. Golovkins. Closure Properties of Probabilistic Reversible DH Automata. QCMC 2006 - 8th International Conference on Quantum Communication, Measurement and Computing. Proceedings.*, pp. 117-120, NICT Press, Japan, 2006.*

[KS 76] J. G. Kemeny and J. L. Snell. Finite Markov Chains. Springer Verlag*, 1976.*

[KW 97] A. Kondacs, J. Watrous. On The Power of Quantum Finite State Automata. Proc. 38th FOCS*, 1997, pp. 66-75.*

[LM 04] S. Laplante, F. Magniez. Lower bounds for randomized and quantum query complexity using Kolmogorov arguments. Proceedings of Complexity04*, pp. 294-304.*

[LV 97] . Li and P. M. B. Vitanyi: An Introduction to Kolmogorov Complexity and its Applications. Springer, Berlin, second edition*, 1997. 2.2.*

[M 98] M.Mosca. Quantum searching, counting and amplitude amplification by eigenvector analysis. In MFCS98 workshop on Randomized Algorithms*, 1998.*

[ME 98] M. Mosca and A. Ekert. The hidden subgroup problem and eigenvalue estimation on a quantum computer. In Proceedings of NASA QCQC conference, Lecture Notes in Computer Science*, volume 1509, Springer, 1998.*

[MI 04] G. Midrijanis. Exact quantum query complexity for total Boolean functions. quant-ph/0403168.

[MO 79] A. Marshall, I. Olkin. Inequalities: Theory of Majorization and Its Applications. Academic Press*, 1979.*

[MC 00] C. Moore, J. P. Crutchfield. Quantum automata and quantum grammars. Theoretical Computer Science*, 2000, Vol. 237(1-2), pp. 275-306.*
`http://arxiv.org/abs/quant-ph/9707031`

[N 99] A. Nayak. Optimal Lower Bounds for Quantum Automata and Random Access Codes. Proc. 40th FOCS*, 1999, pp. 369-377.*
`http://arxiv.org/abs/quant-ph/9904093`

[NC 00] M. Nielsen,I. Chuang. Quantum Computation and Quantum Information. Cambridge University Press, 675 p.,2000.

[NW 99] A. Nayak, F. Wu. The quantum query complexity of approximating the median and related statistics. Proceedings of STOC99*, pp. 384-393, quant-ph/9804066.*

[NS 91] N. Nisan. CREW PRAMs and decision trees. SIAM Journal of Computing*, 20(6):9991007, 1991.*

[NS 94] N. Nisan and M. Szegedy. On the degree of Boolean functions as real polynomials. Computational Complexity*, 4(4):301313, 1994.*

[R 63] M. O. Rabin. Probabilistic Automata. Information and Control*, 1963, Vol. 6(3), pp. 230-245.*

[RA 03] A. Razborov. Quantum communication complexity of symmetric predicates, Izvestiya of the Russian Academy of Science*, mathematics, 67:159-176, 2003. Also quant-ph/0204025.*

[Si 94] Simon, D., On the power of quantum computation, 35th Annual IEEE Symposium on Foundations of Computer Science*, 1994,pp. 116 123.*

[Sh 94] P. W. Shor. Algorithms for Quantum Computation: Discrete Logarithms and Factoring. Proc. 35th FOCS*, pp. 124-134, 1994.*
`http://arxiv.org/abs/quant-ph/9508027`

[Sh 97] P.W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM Journal of Computing*, 26(5):14841509, 1997.*

[SS 04] R. Spalek, M. Szegedy. All quantum adversary methods are equivalent, quant-ph/0409116*.*

[SZ 03] M. Szegedy: On the quantum query complexity of detecting triangles in graphs. quantph/0310107*, 2003.*

[T 68] G. Thierrin. Permutation Automata. Mathematical Systems Theory*, Vol. 2(1), pp. 83-90.*

[VZ 98] U. Vazirani. On the power of quantum computation. Philosophical Transactions of the Royal Society of London, Series A: Mathematical and Physical Sciences*, 356:1759-1768, August 1998.*

[Y 93] A. Yao, Quantum Circuit Complexity, Proc. 34th IEEE Symp. on Foundations of Computer Science*, 1993, pp. 352-361.*

[YKTI 00] T. Yamasaki, H. Kobayashi, Y. Tokunaga, H. Imai. One-Way Probabilistic Reversible and Quantum One-Counter Automata. COCOON 2000, Lecture Notes in Computer Science*, Vol. 1858, pp. 436-446, 2000.*

[ZA 97] C. Zalka. Grovers quantum searching algorithm is optimal. quant-ph/9711070*, 26 Nov 1997.*

[ZH 04] S. Zhang. On the power of Ambainiss lower bounds. Proceedings of ICALP04, Lecture Notes in Computer Science*, 3142:1238-1250.*