On Comparative Computational Power of Classical and Quantum Branching Programs

Farid Ablayev*

Abstract

We describe a model of a Quantum Branching Program (QBP) and study its computational power. We define several natural restrictions of a general QBP model, such as a constant width and a read-once QBP.

- *Constant width.* We present upper and lower bounds on the power of quantum and stochastic branching programs of bounded width. We show any NC^1 function can be accepted exactly by a width-2 quantum branching program of polynomial length, in contrast to the classical case where width 5 is necessary unless $\mathrm{NC}^1 = \mathrm{ACC}$. This separates width-2 quantum programs from width-2 doubly stochastic programs as we show the latter cannot compute the middle bit of multiplication.

  Next, we show that bounded-width quantum and stochastic programs can be simulated by classical programs of larger but bounded width, and thus are in NC^1.

- *Read-once.* We display certain symmetric Boolean function which is computable by a read-once QBP with a $O(\log n)$ width, which requires a width $\Omega(n)$ on any deterministic read-once BP and (classical) randomized read-once BP with the permanent transitions on each level.

  We present also a general lower bound on the width of read-once QBPs, showing that the upper bound for the considered symmetric function is almost tight.

*MPIM, Bonn, Dept. of Theoretical Cybernetics of Kazan State University. Email: `ablayev@ksu.ru`

1 Preliminaries

Interest in quantum computation has been steadily increasing since Shor's discovery of a polynomial time quantum algorithm for factoring [17]. A number of models of quantum computation have been considered, including quantum versions of Turing machines, simple automata, circuits, and decision trees. The goal of much of this research has been to understand in what ways quantum algorithms do and do not offer a speed-up over the classical case, and to understand what classical techniques for proving upper and lower complexity bounds transfer to the quantum setting.

Branching programs have proven useful in a variety of domains, such as hardware verification, model checking, and other CAD applications [18]. Remind that polynomial size branching programs as powerful as log space Turing machines.

Also branching program (BP) (see for example book [18]) is a very convenient nonuniform model for implementing different restricted variants of the computational model. Leveled oblivious permutation BPs are well known in complexity theory, their computational power is strong enough. Remind that 5 width polynomial size BPs as powerful as log depth circuits.

Moreover it is appeared that branching programs are very natural model for comparing the computational power of the quantum models with the deterministic ones. Recently, several models of quantum branching programs have been proposed [1, 2, 7, 14].

In this paper we present several results for quantum branching programs of bounded width [1]. We show that width-2 quantum programs are more powerful than width-2 doubly stochastic programs, and are as strong as deterministic branching programs of width 5. Specifically, we show that polynomial-length, width-2 quantum branching programs can recognize any NC^1 function exactly. This is surprising, since such programs act on a single qubit. On the other hand, we show that polynomial-length, width-2 doubly stochastic programs cannot compute the middle bit of the multiplication function. In the classical case, Yao [19] showed that width-2 deterministic programs require superpolynomial length to compute the majority function, and Barrington [8] showed that width 5 is sufficient for deterministic programs to capture NC^1.

Next, we show that bounded-error quantum and stochastic programs can be simulated by deterministic programs of the same length and larger, but still bounded, width. Therefore these classes are contained in NC^1, and in fact for bounded-width quantum programs exact acceptance is just as strong as acceptance with bounded probability. We use the techniques of this result to show that polynomial-length width-2 stochastic programs accepting with error margin more than $\epsilon = 1/4$ cannot compute majority. In addition, we show that polynomial-length width-2 stochastic programs accepting with error margin more than $\epsilon = 1/8$ and polynomial-length width-3 stochastic programs accepting with error margin more than $\epsilon = 1/\sqrt{5}$ must compute functions in ACC.

Finally, we investigate the properties of the quantum read-once BPs [2]. We first show that read-once exact quantum BPs can compute an arbitrary Boolean function. Next we display a certain symmetric Boolean function which is computable by a read-once *QBP* with $O(\log n)$ width, which requires a width $\Omega(n)$ on any deterministic read-once *BP* and (classical) randomized read-once *BP* with permanent transitions on each level. We present a general lower bound on the width of read-once QBPs, showing that the upper bound for some symmetric functions is almost tight.

1.1 Branching Programs

We begin by discussing the classical model of branching programs and then show how to quantize it. A good source of information on branching programs is Wegener's book [18], and for an introduction to quantum computation see Nielsen and Chuang [15].

Definition 1 *A* branching program *is a finite directed acyclic graph which recognizes some subset of* $\{0,1\}^n$. *Each node (except for the sink nodes) is labeled with an integer* $1 \leq i \leq n$ *and has two outgoing arrows labeled* 0 *and* 1. *This corresponds to querying the ith bit* x_i *of the input, and making a transition along one outgoing edge or the other depending on the value of* x_i. *There is a single source node corresponding to the start state, and there is a subset* A *of the sink nodes corresponding to accepting states. An input* x *is* accepted *if and only if it induces a chain of transitions leading to a sink node in* A.

A branching program is oblivious *if the nodes can be partitioned into levels* $V_1, \ldots, V_l$ *such that the nodes in* V_l *are the sink nodes, nodes in each level* V_j *with* $j < l$ *have outgoing edges only to nodes in the next level* V_{j+1}, *and all nodes in a given level* V_j *query the same bit* x_{i_j} *of the input. Such a program is said to have* length l, *and* width k *if each level has at most* k *nodes.*

Oblivious branching programs have an elegant algebraic definition. Recall that a *monoid* is a set with an associative binary operation $\cdot$ and an identity 1 such that $1 \cdot a = a \cdot 1 = a$ for all a.

Definition 2 *Let M be a monoid and $S \subset M$ an accepting set. Let x_i, $1 \leq i \leq n$ be a set of Boolean variables. A* branching program *over M of length l is a string of l* instructions*; the jth instruction is a triple $(i_j, a_j, b_j) \in \{1, \ldots, n\} \times M \times M$, which we interpret as a_j if $x_{i_j} = 0$ and b_j if $x_{i_j} = 1$. Given an input x, the* yield *$Y(x)$ of the program is the product in M of all its instructions. We say that the input x is* accepted *if $Y(x) \in S$, and the set of such inputs is the function f recognized by the program.*

Such programs are often called *non-uniform deterministic finite automata* (NUDFAs); a computation over a deterministic finite automaton consists of taking a product in its syntactic monoid, while in a NUDFA we allow the same variable to be queried many times, and for "true" and "false" to be mapped into a different pair of monoid elements in each query.

A common monoid is T_k, the set of functions from a set of k objects into itself. Then the program makes transitions among k states, and we can equivalently define oblivious, width-k branching programs by choosing an initial state and a set of accepting states, where the k states correspond, according to an arbitrary ordering, to the k vertices in each level V_j.

Definition 3 *An* oblivious width-k branching program *is a branching program over T_k, where the accepting set $S \subset T_k$ consists of those elements of T_k that map an initial state $s \in \{1, \ldots, k\}$ to a final state $t \in F$ for some subset $F \subset \{1, \ldots, k\}$.*

2 Bounded Width Branching Programs

We define function classes recognized by (non-uniform) families of bounded-width branching programs whose length increases polynomially with n:

Definition 4 k-BWBP *is the class of functions recognized by polynomial-length branching programs of width k, and* BWBP $= \cup_k\, k$-BWBP.

Recall that a *group* is a monoid where every element has an inverse, and a group is *Abelian* if $ab = ba$ for all a, b. A subgroup $H \subseteq G$ is *normal* if the left and right cosets coincide, $aH = Ha$ for all $a \in G$. A group is *simple* if it has no normal subgroups other than itself and $\{1\}$.

Barrington [8] studied branching programs over the permutation group on k objects $S_k \subset T_k$; such programs are called *permutation programs.* He showed that polynomial-length programs over S_5, and therefore width-5 branching programs, can recognize any function in NC^1, the class of functions recognizable by boolean circuits of polynomial width and logarithmic depth. The version of Barrington's result that we will use is:

Theorem 1 ([8, 13]) *Let G be a non-Abelian simple group, and let $a \neq 1$ be any non-identity element. Then any function f in NC^1 can be recognized by a family of polynomial-length branching programs over G such that their yield is $Y(x) = a$ if $x \in L$ and 1 otherwise.*

Since the smallest non-Abelian simple group is $A_5 \subset S_5$, the group of even permutations of 5 objects, and since we can choose a to map some initial state s to some other final state t, width 5 suffices. Conversely, note that we can model a width-k branching program as a boolean product of l transition matrices of dimension k, and a simple divide-and-conquer algorithm allows us to calculate this product in $O(\log l)$ depth. Thus BWBP $\subset \mathrm{NC}^1$, so we have

$$\text{5-BWBP} = \text{BWBP} = \mathrm{NC}^1 \ .$$

To define stochastic and quantum branching programs, we write the probability of acceptance as an inner product. Let e_s and e_t be k-dimensional vectors whose entries are 1 for the initial state and accepting final states respectively and 0 otherwise, and M_j the matrix corresponding to the jth instruction. Then write

$$P(x) = \left\langle e_s \left| \prod_{j=1}^{l} M_j \right| e_t \right\rangle$$

In the deterministic case $P = 1$ or $P = 0$ for all x, since the transition matrix corresponding to an element of T_k has exactly one 1 in each column. For a group this is true of the rows as well, in which case the M_j are permutation matrices. We can generalize this by letting the M_j be *stochastic* matrices, i.e. mmatrices with non-negative entries where each column sums to 1, and letting e_s be an initial probability distribution over the set of states. Then P is the probability that the program accepts. If the transpose of a stochastic matrix is also stochastic the matrix is called *doubly stochastic*. If all the matrices in a program are doubly stochastic then we say the program is a *doubly stochastic* program.

In the quantum case, we let the M_j be complex-valued and *unitary*, i.e. $M_j^{-1} = M_j^\dagger$ where $\dagger$ denotes the Hermitian conjugate, and e_s and e_t be initial and final state vectors with $|e_s|^2 = |e_t|^2 = 1$. Then the program accepts with probability

$$P(x) = \left| \left\langle e_s \left| \prod_{j=1}^{l} M_j \right| e_t \right\rangle \right|^2$$

Note that this is a "measure-once" model analogous to the quantum finite automata of [12], in which the system evolves unitarily except for a single measurement at the end. We could also allow multiple measurements during the computation, by representing the state as a density matrix and making the M_i superoperators instead of purely unitary operators; we do not do this here.

We can define recognition in several ways for the quantum case. We say that a function f is accepted *with bounded probability* if there is some $\epsilon > 0$ such that $P(x) > 1/2 + \epsilon$ if $x \in L$ and $P(x) < 1/2 - \epsilon$ if $x \notin L$, and accepted *exactly* if $P(x) = 1$ if $x \in L$ and $P(x) = 0$ if $x \notin L$ as in the deterministic case.

We denote by $B\cdot$ (where we will drop the '$\cdot$' when clear) the function classes recognized with bounded probability, and $E\cdot$ those recognized exactly. Writing SBP and QBP for stochastic and quantum branching programs respectively, we define the classes of functions recognized by width-k stochastic and quantum programs of polynomial length k-BSBP, k-BQBP, and k-EQBP. Note that we remove "BW" to avoid acronym overload. We write BSBP for $\cup_k k$-BSBP and define BQBP and EQBP similarly. We have

$$\text{BWBP} \subseteq \text{EQBP} \subseteq \text{BQBP}$$

and

$$\text{BWBP} \subseteq \text{BSBP}$$

but in principle k-BSBP could be incomparable with k-EQBP or k-BQBP.

2.1 Width-2 Doubly Stochastic and Quantum Programs

In this section we show that width-2 quantum programs with exact acceptance contain NC^1 and also show that these programs are stronger than width-2 doubly stochastic programs.

First we note that stochastic programs are stronger than permutation programs for width 2. It is easy to see that any program over $\mathbb{Z}_2$ simply yields the parity of some subset of the x_i. The AND_n function, which accepts only the input with $x_i = 1$ for all i, is not of this form, and so this function cannot be recognized by a width-2 permutation program. However, it can easily be recognized by a stochastic program P with bounded error which queries each variable once as follows: for $i < n$ it maps $x_i = 1$ and 0 to the identity $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ and the matrix $\begin{pmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{pmatrix}$ respectively, and for x_n it maps 1 and 0 to $\begin{pmatrix} 3/4 & 1/4 \\ 0 & 1 \end{pmatrix}$ and $\begin{pmatrix} 3/8 & 5/8 \\ 3/8 & 5/8 \end{pmatrix}$ respectively. Taking the first state to be both the initial and final state, P accepts with probability 3/4 if $x_i = 1$ for all i and 3/8 otherwise. Note that except for one matrix this is in fact a doubly stochastic program. If we had treated the variable x_n in the same fashion as the other variables we would have gotten a doubly stochastic program accepting AND_n with one-sided error.

Despite being stronger than their permutation counterparts, the next result shows width-2 doubly stochastic branching programs are not that strong. Let $MULT_k^n$ be the boolean function which computes the kth bit of the product of two n-bit integers. Define $MULT^n$ to be $MULT_{n-1}^n$. i.e., the middle bit of the product. We will argue that this function requires at least exponential lengthed width 2 stochastic programs.

Lemma 1 *Any width-2 doubly stochastic program on n variables is equivalent to one which queries each variable once and in the order 1,2,3,...,n.*

Proof. Any 2×2 stochastic matrix can be written as $\begin{pmatrix} p & 1-p \\ 1-p & p \end{pmatrix}$ for some $p \in [0, 1]$. It is easy to verify that matrices of this kind commute. Hence, if we have a product of such matrices $\prod_j^n M_{x_{j_i}}$ we can rewrite it so that we first take the product of all the matrices that depend on x_1, then those that depend on x_2, and so on. To finish the proof we note that products of doubly stochastic matrices are again doubly stochastic, so we can use a single doubly stochastic matrix for the product of all the matrices that depend on a given x_i. $\square$

The above lemma shows we can convert any width-2 doubly stochastic program into one which is read-once and with a fixed variable ordering. i.e. a randomized ordered binary decision diagram (OBDD). The next result is proved in Ablayev and Karpinski [3].

Theorem 2 *A BP-OBDD that correctly computes $MULT^n$ has length at least $2^{\Omega(n/\log n)}$.*

So by Lemma 1 we have immediately:

Corollary 1 *Any width 2 doubly stochastic program correctly computing $MULT^n$ with bounded error has length at least $2^{\Omega(n/\log n)}$.*

So width-2 stochastic programs are not that strong. However, width-2 *quantum* programs are surprisingly strong, as the next result shows. Note that a width-2 quantum program has a state space equivalent to a single qubit such as a single spin-1/2 particle [15].

Theorem 3 NC^1 *is contained in* 2-EQBP.

Proof. First, recall that A_5, the smallest non-Abelian simple group, is the set of rotations of the icosahedron. Therefore, the group $SO(3)$ of rotations of $\mathbb{R}^3$, i.e. the 3×3 orthogonal matrices with determinant 1, contains a subgroup isomorphic to A_5.

There is a well-known 2-to-1 mapping from $SU(2)$, the group of 2×2 unitary matrices with determinant 1, to $SO(3)$. Consider a qubit $a|0\rangle+b|1\rangle$ with $|a|^2 + |b|^2 = 1$; we can make a real by multiplying by an overall phase. The *Bloch sphere* representation (see e.g. [15]) views this state as the point on the unit sphere with latitude θ and longitude ϕ , i.e. $(\cos\phi\cos\theta, \sin\phi\cos\theta, \sin\theta)$, where $a = \cos\theta/2$ and $b = e^{i\phi}\sin\theta/2$.

Given this representation an element of $SU(2)$ is equivalent to some rotation of the unit sphere. Recall the *Pauli matrices*

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \sigma_y = \begin{pmatrix} 0 & i \\ -i & 0 \end{pmatrix}, \quad \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Then we can rotate an angle α around the x, y or z axes with the following operators:

$$\begin{aligned} R_x(\alpha) = e^{i(\alpha/2)\sigma_x} &= \begin{pmatrix} \cos\alpha/2 & -i\sin\alpha/2 \\ -i\sin\alpha/2 & \cos\alpha/2 \end{pmatrix} \\ R_y(\alpha) = e^{i(\alpha/2)\sigma_y} &= \begin{pmatrix} \cos\alpha/2 & -\sin\alpha/2 \\ \sin\alpha/2 & \cos\alpha/2 \end{pmatrix}, \text{ and} \\ R_z(\alpha) = e^{i(\alpha/2)\sigma_z} &= \begin{pmatrix} e^{-i\alpha/2} & 0 \\ 0 & e^{i\alpha/2} \end{pmatrix}. \end{aligned}$$

This makes $SU(2)$ a *double cover* of $SO(3)$, where each element of $SO(3)$ corresponds to two elements $\pm U$ in $SU(2)$. (Note that angles get halved by this mapping.) Therefore, $SU(2)$ has a subgroup which is a double cover of A_5. One way to generate this subgroup is with $2\pi/5$ rotations around two adjacent vertices of an icosahedron. Since two such vertices are an angle

$\tan^{-1} 2$ apart, if one is pierced by the z axis and the other lies in the x-z plane we have

$$\begin{aligned} a &= R_z(2\pi/5) = \begin{pmatrix} e^{i\pi/5} & 0 \\ 0 & e^{-i\pi/5} \end{pmatrix} \\ b &= R_y(\tan^{-1} 2) \cdot a \cdot R_y(-\tan^{-1} 2) \\ &= \frac{1}{\sqrt{5}} \begin{pmatrix} e^{i\pi/5}\tau + e^{-i\pi/5}\tau^{-1} & -2i \sin \pi/5 \\ -2i \sin \pi/5 & e^{-i\pi/5}\tau + e^{i\pi/5}\tau^{-1} \end{pmatrix} \end{aligned}$$

where $\tau = (1+\sqrt{5})/2$ is the golden ratio. Now consider the group element $c = a \cdot b \cdot a$; this rotates the icosahedron by π around the midpoint of the edge connecting these two vertices. In $SU(2)$, this maps each of the eigenvectors of σ_y to the other times an overall phase. Taking these as the initial and final state,

$$e_s = \frac{|0\rangle + i|1\rangle}{\sqrt{2}}, \quad e_t = \frac{|0\rangle - i|1\rangle}{\sqrt{2}}$$

we have

$$|\langle e_s|c|e_t\rangle|^2 = 1$$

while, since the two eigenvectors are orthogonal,

$$|\langle e_s|1|e_t\rangle|^2 = 0 \ .$$

Now, Theorem 1 tells us that for any function in NC^1 we can construct a polynomial-length program over A_5 that yields the element equivalent to c if the input is in the function and 1 otherwise. Mapping this function to $SU(2)$ gives a program which yields $\pm c$ or 1, and accepts with probability 1 or 0. □

2.1.1 Deterministic Classical Simulations of Stochastic and Quantum Branching Programs

In this section we give general results on simulating stochastic and quantum programs by classical ones. We use this to show that width-2 quantum programs can be simulated by bounded-width deterministic programs, and to show that if $\epsilon > 1/4$ then width-2 stochastic programs for majority require super-polynomial length. We also get that polynomial-length width-2 stochastic programs accepting with error margin more than $\epsilon = 1/8$ and polynomial-length width-3 stochastic programs accepting with error margin more than $\epsilon = 1/\sqrt{5}$ must compute functions in ACC.

Theorem 4 *If a function is recognized with bounded probability* $1/2+\epsilon$ *by a width-w stochastic branching program, then it is also recognized by a deterministic branching program* D *of the same length, and width*

$$w' \leq \left(1+\frac{1}{\epsilon}\right)^{w}.$$

Similarly, if a function is recognized with bounded probability $1/2+\epsilon$ *by a width-w quantum branching program, it is recognized by a deterministic program of the same length, and width*

$$w' \leq \left(1+\frac{1}{\epsilon}\right)^{2w}.$$

Proof of Theorem 4 We start with the idea of lower bounds proof and notations we use for formal proof.

Let us denote (w,l)-P an w width and l length BP. The idea of proofs of theorem 4 is that having stochastic (quantum) (w,l)-P that computes a function f with margin ϵ we construct a deterministic BP (w',l) - P' such that P' computes the same function f and

$$w' \leq \left(1+\frac{1}{\epsilon}\right)^{w}$$

when P is stochastic BP and

$$w' \leq \left(1+\frac{1}{\epsilon}\right)^{2w}$$

when P is quantum BP.

The construction of P' is based on the following properties. We will view on computation by oblivious (stochastic and quantum) BP (w,l)-P as an l step linear process of transformation of vectors α of a current internal description (ID) of P. In stochastic case $\alpha=\mu$, where $\mu=(p_1,\ldots,p_w)$ is a current probability distribution of a states of P, and in quantum case $\alpha=|\psi\rangle$, where $|\psi\rangle=(z_1,\ldots,z_w)^T$ is a current distribution of amplitudes of states of P.

We represent this l step linear process as an $(l+1)$-leveled deterministic branching program DP, as follows: each node of DP labeled by vector α corresponds to an ID of P, and each level $i\in\{0,\ldots,l\}$ represents a step of the computation. The level 0 contains initial node labeled α_0 — the initial distribution of P. From each node α on the level i, $i\in\{0,\ldots,l-1\}$, out goes two edges labeled $x_{j_i} = 0$ and $x_{j_i} = 1$ where x_{j_i} is a variable tested in the computational step i. Edge $x_{j_i} = 1$, directed from parent α on a level i to child α' on the level $i+1$ iff P being on the step i of computation in ID α tests $x_{j_i} = 1$ and transforms its ID on the step $i+1$ to α'.

Denote *Accept* (*Reject*) a set of all accepting (rejecting) sink nodes of P. Sink nodes of DP on the level l are labeled in addition by 1 ("accept") and 0 ("reject") as follows: in stochastic case sink node $\alpha = (p_1, \dots, p_w)$ labeled 1 if $Pr_{accept}(\alpha) = \sum_{j \in Accept} p_j \geq 1/2 + \epsilon$, and labeled 0 if $Pr_{accept}(\alpha) = \sum_{j \in Accept} p_j \leq 1/2 - \epsilon$; in quantum case node $\alpha = (z_1, \dots, z_w)^T$ labeled 1 if $Pr_{accept}(\alpha) = \sum_{j \in Accept} |z_j|^2 \geq 1/2 + \epsilon$, and labeled 0 if $Pr_{accept}(\alpha) = \sum_{j \in Accept} |z_j|^2 \leq 1/2 - \epsilon$. Denote $\mathcal{A}$ ($\mathcal{R}$) a set of all sink nodes of DP labeled 1 (0).

As usual DP accepts (rejects) an input $\vec{a} = a_1 \dots a_n$ if the computation of DP on $\vec{a}$ ends in accepting (rejecting) node.

From the above we have the following property.

Property 1 *Deterministic branching program DP computes the same Boolean function f as P.*

In the next section we use metric point of view to DP for constructing deterministic BP (w', l)-P' that computes the same function as DP and hence the same function as P.

Metric Properties of DP Denote Ψ_i, $i \in \{0, \dots, l\}$, a set all possible IDs of P on step i of computation. Let $\Psi = \cup_{i=0}^{l} \Psi_i$. For stochastic P we define metric on the space Ψ based on norm $|| \cdot ||_1$. That is, for $\alpha = (p_1, \dots, p_w)$ and $\alpha' = (p'_1, \dots, p'_w)$ $\rho(\alpha, \alpha') = ||\alpha - \alpha'||_1 = \sum_{i=1}^{w} |p_i - p'_i|$. For quantum P we define metric on the space Ψ based on norm $|| \cdot ||_2$: for $\alpha = (z_1, \dots, z_w)$ and $\alpha' = (z'_1, \dots, z'_w)$ $\rho(\alpha, \alpha') = ||\alpha - \alpha||_2$. We will also use notation $|| \cdot ||$ for norm $|| \cdot ||_2$.

Recall known notions of metric spaces we need in the proof (see for example [5]). Denote $\mathcal{H}_w$ an w-dimensional vector space (real valued or complex valued) with metric ρ. Points μ, μ' from $\mathcal{H}_w$ are connected through θ-chain if there exists a finite set of points $\mu_1, \mu_2, \dots, \mu_m$ from $\mathcal{H}_w$ such that $\mu_1 = \mu, \mu_m = \mu'$ and $\rho(\mu_i, \mu_{i+1}) < \theta$ for $i \in \{1, \dots, m-1\}$. For $\mathcal{H}_w$ its subset C is called θ-component if arbitrary two points $\mu, \mu' \in \mathcal{C}$ are connected through θ-chain. It is known [5] that if $\mathcal{D}$ is a finite diameter subset of a subspace of $\mathcal{H}_w$ (diameter of $\mathcal{D}$ is defined as $\sup_{\mu, \mu' \in \mathcal{D}} \{\rho(\mu, \mu')\}$ then for $\theta > 0$ $\mathcal{D}$ is partitioned to a finite number t of its θ-components.

Lemma 2 *Let f be a Boolean function $(1/2+\epsilon)$-computed by P. Let DP be a corresponding deterministic BP for P. Let $\theta > 0$ and let for sink nodes of DP the following holds: for arbitrary $\alpha \in \mathcal{A}$ and $\alpha' \in \mathcal{R}$ it is holds that $\rho(\alpha, \alpha') \geq \theta$. Then, there exists a deterministic BP (w', l)-P which computes f and*

$$w' \leq \left(1 + \frac{2}{\theta}\right)^{w}$$

when P is stochastic BP, and

$$w' \leq \left(1 + \frac{2}{\theta}\right)^{2w}$$

when P is quantum BP.

Proof. Consider $\mathcal{D}$ a sphere of radius 1 with center in $(0, \ldots, 0)$. We clearly have that $\Psi \subseteq \mathcal{D}$. From the condition of the lemma it follows that subsets $\mathcal{A}$ and $\mathcal{R}$ of Ψ_l is a union of some θ-components of Ψ_l. Next. Oblivious property of BP P provides the following: on the each level i, $i \in \{0, \ldots, l-1\}$, of DP for all nodes $\alpha \in \Psi_i$ it is tested the same variable x_{j_i} and applied the same linear transition $M_{x_{j_i}}(1)$ if $x_{j_i} = 1$ ($M_{x_{j_i}}(0)$ if $x_{j_i} = 0$) where $M_{x_{j_i}}(a)$ is stochastic matrix when P is stochastic BP and $M_{x_{j_i}}(a)$ is unitary when P is quantum BP.

It is known (and it can be easily verified) that transformations determined by stochastic matrix M does not increase the distance. That is, if $\alpha' = M\alpha$ and $\beta' = M\beta$ then $\rho(\alpha', \beta') \leq \rho(\alpha, \beta)$. Unitary matrix M preserves the distance. That is, it is holds that $||\alpha' - \beta'|| = ||\alpha - \beta||$.

Denote $\mathcal{C}_i$ the set of all θ-components of Ψ_i. For $C \in \mathcal{C}_i$ and matrix M we denote $MC = \{\alpha' : \alpha = M\alpha, \alpha \in C\}$. From the property of non increasing distance linear transformations (stochastic or uniform) it is holds that for $C \in \mathcal{C}_i$ and for $a \in \{0,1\}$ there exists $C' \in \mathcal{C}_{i+1}$ such that $MC \subseteq C'$ for the stochastic P and $MC = C'$ for the quantum P.

Now we describe deterministic BP P' that computes f as follows: P' is an l-leveled oblivious BP. On the level i tested variable x_{j_i} (as in BP) and all nodes are labeled by θ-components $C \in \mathcal{C}_i$. From the node $C \in \mathcal{C}_i$ edge labeled $x_{j_i} = a$ goes to to node $C' \in \mathcal{C}_{i+1}$ iff $M_{x_{j_i}}(a)C \subseteq C'$.

From the above it follows that P' computes the same function as DP

The width $w(P')$ of P' is $w' = \max\{t_0, \ldots, t_l\}$ where t_i is the number θ-components of Ψ_i. Let $w' = t_i$. We estimate the number t of θ-components (number of nodes of B) of Ψ_j as follows.

For each θ-component $C \in \mathcal{C}_i$ select one point $\alpha \in C$. If we draw a sphere of the radius $\theta/2$ with the center $\alpha \in C$ then all such spheres do not intersect pairwise. All these w' spheres are in large sphere of radius $1+\theta/2$ which has center $(0,0,\dots,0)$. The volume of a sphere of a radius r in $\mathcal{H}_w$ is cr^w when $\mathcal{H}_w$ is real space and is cr^{2w} when $\mathcal{H}_w$ is complex space (in the complex space $\mathcal{H}_w$ each complex point is a 2-dimensional point). Constant c depends on the metric of $\mathcal{H}_w$. Now for stochastic and quantum case we have respectively that

$$w' \le \frac{c\,(1+\theta/2)^w}{c\,(\theta/2)^w} = \left(1+\frac{2}{\theta}\right)^w, \qquad w' \le \frac{c\,(1+\theta/2)^{2w}}{c\,(\theta/2)^{2w}} = \left(1+\frac{2}{\theta}\right)^{2w},$$

$\square$

Below we present technical lemma that estimates parameter θ of the lemma 2 depending on margin ϵ of computation.

Lemma 3 *Let f be a Boolean function $(1/2+\epsilon)$-computed by (w,l)-P. Let DP be a corresponding deterministic BP for P.*

Then for arbitrary $\alpha \in \mathcal{A}$ and $\alpha' \in \mathcal{R}$ for the case of stochastic P it is holds that:

$$||\alpha - \alpha'||_1 \ge \theta = 2\epsilon,$$

for the case of quantum P it is holds that:

$$||\alpha - \alpha'|| \ge \theta = 2\epsilon.$$

Proof. Consider the case of stochastic P. $\alpha = (p_1 \dots, p_w)^T$, $\alpha' = (p'_1 \dots, p'_w)^T$,

$$||\alpha - \alpha'||_1 = \sum_{i=1}^{w} |p_i - p'_i| \ge \sum_{i\in Accept} |p_i - p'_i| \ge |\sum_{i\in Accept} p_i - \sum_{i\in Accept} p'_i|$$

From the condition of the lemma we have that $\sum_{i\in Accept} p_i \ge 1/2+\epsilon$ and $\sum_{i\in Accept} p'_i \le 1/2-\epsilon$. From this we have that

$$||\alpha - \alpha'||_1 \ge 1/2+\epsilon-(1/2-\epsilon) = 2\epsilon.$$

Consider the case of quantum P. $\alpha = (z_1,\dots,z_d)^T$ and $\alpha' = (z'_1,\dots z'_d)^T$.

From the condition of the lemma it is holds that

$$2\epsilon \le \sum_{i\in Accept} (|z_i|^2 - |z'_i|^2) = \sum_{i\in Accept} (|z_i| - |z'_i|)(|z_i| + |z'_i|) \le \sum_{i\in Accept} (|z_i - z'_i|)(|z_i| + |z'_i|)$$

and similarly

$$2\epsilon \le \sum_{i\in Reject} (|z_i'|^2 - |z_i|^2) = \sum_{i\in Reject} (|z_i'| - |z_i|)(|z_i| + |z_i'|) \le \sum_{i\in Reject} (|z_i - z_i'|)(|z_i| + |z_i'|)$$

or

$$4\epsilon \le \sum_{i=1}^{w} (|z_i - z_i'|)(|z_i| + |z_i'|).$$

From the above using known inequality $\sum_{i=1}^d a_i b_i \le \sqrt{\sum_{i=1}^d a_i^2}\sqrt{\sum_{i=1}^d b_i^2}$ and triangle inequality for the norm we get that

$$4\epsilon \le ||\alpha - \alpha'||(||\,|\alpha| + |\alpha'|\,|| \le ||\alpha - \alpha'||(||\alpha|| + ||\alpha'|| = 2||\alpha - \alpha'||$$

□

Now the lower bounds of the theorem 4 follows immediately from lemmas 2 and 3. This completes the proof of theorem 4 □

While we may need an exponentially larger width to simulate a stochastic or quantum branching program, the width is still constant. Thus bounded-width programs of all three types are equivalent, and since bounded-width classical programs are contained in NC^1 they all are. Conversely, we showed in Theorem 3 that NC^1 is contained in width-2 quantum programs, so we have

Corollary 2

$$\text{2-EQBP} = \text{2-BQBP} = \text{BQBP} = \text{BSBP} = \text{BWBP} = \mathrm{NC}^1\ .$$

In other words, width-2 quantum programs with exact acceptance are exactly as strong as quantum or stochastic programs of arbitrary bounded width, with exact or bounded-error acceptance.

3 Read-once Branching Programs (OBDDs)

We investigate computational power of read-once branching programs in this section. In according to definitions 3 we use here the following model of branching programs:

1. each variable of a branching program is tested only once during a computation.

2. We do not restrict a width of branching program to constant.

We clearly have that read-once branching program has length n where n is a number of Boolean variables of branching program. This model of branching programs is well known and usually called Ordered Binary Decision Diagram (quantum OBDD) [18].

The next statement says that exponential width read-once quantum branching can computes arbitrary Boolean function. This is not surprised. This is true for classical deterministic OBDDs as well. We present this statement for the complete of presentation.

Property 2 *For an arbitrary Boolean function $f(x_1,\ldots,x_n)$, there exists a read-once 2^n QBP that exact computes f.*

Proof. The proof is simple. For example, the following read-once 2^n QBP P is a QBP satisfied the proposition. All possible configurations of P are trivial. That is, a configuration $|\psi\rangle$ of P contains exactly one 1 and all the rest components of $|\psi\rangle$ are 0. An initial configuration of P is $|\psi_0\rangle = (1,0,\ldots,0)$. P reads input variables in order $x_1, x_2, \ldots, x_n$.

In each step $i, 1 \leq i \leq n$, P reads input σ_i and transforms its current configuration $|\psi\rangle$ as follows. If $\sigma_i = 0$ then $|\psi\rangle$ does not changed. If $\sigma_i = 1$, then the 1 of the configuration $|\psi\rangle$ is "moved" to 2^{n-i} positions to the right in the next configuration $|\psi'\rangle$.

For an input sequence $\sigma = \sigma_1, \ldots . \sigma_n$ denote l_σ the number of position of 1 in the final (after reading σ) configuration of P. Clearly we have that $l_\sigma \neq l_{\sigma'}$ iff $\sigma \neq \sigma'$.

Now determine the set of accepting states F of P as follows: if $f(\sigma) = 1$, then $q_{l_\sigma} \in F$. If $f(\sigma) = 0$, then $q_{l_\sigma} \notin F$. □

Consider the following symmetric Boolean function MOD_{p_n}: For an input $\sigma = \sigma_1, \ldots, \sigma_n \in \{0,1\}^n$ we have $MOD_{p_n}(\sigma) = 1$ iff a number of ones in σ is divisible by p_n, where p_n is a prime and $p_n \leq n/2$.

Theorem 5 *The function MOD_{p_n} can be presented by a read-once $O(\log p_n)$ QBP with one-sided error $\epsilon \geq 0$.*

The proof of this theorem will be presented in the section below. We have clearly that any deterministic OBDD for MOD_{p_n} has $\Omega(p_n)$ width.

3.1 Proof of the Theorem 5

We construct a QBP P accepting inputs $\sigma \in MOD_{p_n}^{-1}(1)$ with probability 1 and rejecting inputs $\sigma \in MOD_{p_n}^{-1}(0)$ with probability at least 1/8. Consider $2QBP$ P^k for $k \in \{1, \dots, p_n - 1\}$. Quantum program $P^k = \langle T^k, |\psi_0^k\rangle, F^k\rangle$ based on 2-state quantum system. Here $T^k = (\langle i, U^k(0), U^k(1)\rangle)_{i=1}^n$ where

$$U^k(0) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, U^k(1) = \begin{pmatrix} \cos(2\pi k/p_n) & -\sin(2\pi k/p_n) \\ \sin(2\pi k/p_n) & \cos(2\pi k/p_n) \end{pmatrix}, |\psi_0^k\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, F^k = \{1\}.$$

Denote by $l(\sigma)$ a number of 1-s in the sequence σ, $l(\sigma) = \sum_{i=1}^n \sigma_i$.

Lemma 4 *After reading an input $\sigma = \sigma_1, \dots, \sigma_n$ the superposition of P^k is*

$$|\psi\rangle = \cos\left(\frac{2\pi\, l(\sigma)k}{p_n}\right)|1\rangle + \sin\left(\frac{2\pi\, l(\sigma)k}{p_n}\right)|2\rangle.$$

Proof: The proof follows from the description of QBP P^k.

□

If the number of ones in an input σ is divisible by p_n then $2\pi\, l(\sigma)k/p_n$ is a multiple of 2π and $\cos\Big(2\pi\, l(\sigma)k/p_n\Big) = 1$, $\sin\Big(2\pi\, l(\sigma)k/p_n\Big) = 0$. Therefore all QBP P^k accept inputs $\sigma \in f_{n,p_n}^{-1}(1)$ with probability 1.

Call P^k "good" for input $\sigma \in MOD_{p_n}^{-1}(0)$ if P^k reject σ with probability at least 1/2.

Lemma 5 *For any $\sigma \in MOD_{p_n}^{-1}(0)$, at least $(p_n - 1)/2$ of all P^k are "good".*

Proof: According to the Lemma 4 after reading an input $\sigma = \sigma_1, \dots, \sigma_n$ the superposition of P^k is

$$|\psi\rangle = \cos\left(\frac{2\pi\, l(\sigma)k}{p_n}\right)|1\rangle + \sin\left(\frac{2\pi\, l(\sigma)k}{p_n}\right)|2\rangle.$$

Therefore, the probability of accepting the input $\sigma \in MOD_{p_n}^{-1}(0)$ is $\cos^2\Big(2\pi\, l(\sigma)k/p_n\Big)$. $\cos^2\Big(2\pi\, l(\sigma)k/p_n\Big) \le 1/2$ iff $\Big|\cos\Big(2\pi\, l(\sigma)k/p_n\Big)\Big| \le 1/\sqrt{2}$. This happens if and only if $\Big(2\pi\, l(\sigma)k/p_n\Big)$ is in $[\pi/4 + 2\pi j, 3\pi/4 + 2\pi j]$ or in $[5\pi/4 + 2\pi j, 7\pi/4 + 2\pi j]$ for some $j \in N$. $\Big(2\pi\, l(\sigma)k/p_n\Big) \in [\pi/4 + 2\pi j, 3\pi/4 + 2\pi j]$ iff $\Big(2\pi\, (l(\sigma)k \bmod p_n)/p_n\Big) \in [\pi/4, 3\pi/4]$. p_n is prime, $l(\sigma)$ is relatively prime with p_n. Therefore, $l(\sigma) \bmod p_n$, $2l(\sigma) \bmod p_n, \dots,$

$(p_n-1)l(\sigma) \bmod p_n$ is $1, 2, \ldots, p_n-1$ in different order. Consequently, it is enough to find the power of a set $I = \{i_1, \ldots, i_l\} \subset \{1, \ldots, p_n-1\}$ such that $2\pi\, i_j\,/p_n \in [\pi/4, 3\pi/4]$ or $2\pi\, i_j\,/p_n \in [5\pi/4, 7\pi/4]$. Since p_n points $2\pi/p_n, \ldots, 2\pi(p_n-1)/p_n, 2\pi$ are regularly distributed on a circumference and sectors $[\pi/4, 3\pi/4]$ and $[5\pi/4, 7\pi/4]$ are exactly half of the circumference, therefore $|I| \geq \lfloor p_n/2 \rfloor \geq (p_n-1)/2$.

$\square$

We call a set of quantum programs $S = \{P^{i_1}, \ldots, P^{i_t}\}$ "good" for $\sigma \in f^{-1}_{n,p_n}(0)$ if at least $1/4$ of all its elements are "good" for this σ.

Lemma 6 *There is a set S of 1QBPs with $|S| = t = \lceil 16 \ln p_n \rceil$ which is "good" for all inputs $\sigma \in MOD^{-1}_{p_n}(0)$.*

Proof: We consider following procedure $\mathcal{A}$ for construction the set S.

> For a fixed input σ with $l(\sigma) \leq p_n - 1$ $\mathcal{A}$ selects quantum branching program uniformly at random from $\{P^1, \ldots, P^{p_n-1}\}$.

The probability of selecting a "good" QBP on each step is at least $1/2$. Using Chernoff inequality, we have the probability that less than $1/4$ fraction of all QBPs from the set S are "good" for any fixed σ with $l(\sigma) \leq p_n - 1$ is at most

$$\exp((-16 \ln p_n)/2(1/2)^2/2 = 1/p_n.$$

Hence the probability that constructing set is not "good" for at least one input σ with $l(\sigma) \leq p_n - 1$ is at most $(p_n-1)/p_n > 0$. Therefore there exist a set which is "good" for all inputs σ with $l(\sigma) \leq p_n - 1$. This set is "good" for inputs σ with $l(\sigma) > p_n$ as well since any QBP P^k returns a currently superposition to starting superposition after reading every p_n ones, and hence works the same way on inputs σ, σ' with $l(\sigma) = l(\sigma') \bmod p_n$.

$\square$

We construct a $1QBP$ P accepting inputs $\sigma \in MOD^{-1}_{p_n}(1)$ with probability 1 and rejecting inputs $\sigma \in MOD^{-1}_{p_n}(0)$ with probability at least $1/8$ as follows. The $1QBP$ P consists of QBPs from "good" set $S = \{P^{i_1}, \ldots, P^{i_t}\}$, which work in parallel. In the starting superposition of P all these programs represented with equal amplitudes.

Inputs $\sigma \in MOD^{-1}_{p_n}(1)$ are always accepted with probability 1 because all P^ks accept them. For any input $\sigma \in MOD^{-1}_{p_n}(0)$ at least $1/4$ of all

$P^k \in S$ reject it with probability at least $1/2$ and the total probability of rejecting any $\sigma \in MOD_{p_n}^{-1}(0)$ is at least $1/8$.

We can make the error as small as possible using standard technique for reducing error for one-sided error computation. That is, we take $d = d(\epsilon)$ copies of such $1QBP$ P and run them uniformly at random. In this case the width of $1QBP$ will be $O(\log p_n)$. $\square$

- We call a branching program P *stable* if its transformations does not depend on the level of P.

From the proof of the theorem we have that a constructed QBP for MOD_{p_n} is a stable branching program.

Corollary 3 *The function MOD_{p_n} can be presented by a stable read-once $(O(\log p_n)QBP$ with one-sided error $\epsilon > 0$.*

Below we show that MOD_{p_n} function is hard for randomized OBDD.

3.2 Lower Bound for randomized OBDD for MOD

Randomized OBDDs are defined and investigated first in [3]. See also book [18].

Theorem 6 *Any stable probabilistic OBDD computing MOD_{p_n} has width at least p_n.*

Proof: Assume that there is a stable probabilistic OBDD P of width $q < p_n$ computing MOD_{p_n} with probability $1/2 + \epsilon$ for a fixed $\epsilon \in (0, 1/2]$. We can suppose each level of P has exactly q nodes without loss of generality. Let $\mu^j = (\mu_1^j, \ldots, \mu_q^j)$ be a probability distribution of states of P on j-th level, where μ_i^j is probability of being in the i-th node of the j-th level. We can describe a computational process of P on an input $\tilde{\sigma} = \sigma_1, \ldots, \sigma_n$ as follows:

- Computation of P starts from the initial probability distributions vector μ^0.
- On j-th step, $1 \leq j \leq n$ P reads input σ_{i_j} and transforms vector μ^{j-1} to $\mu^j = \mu^{j-1}A$, where A is $q \times q$ stochastic matrix, $A = A(0)$ if $\sigma_{i_j} = 0$ and $A = A(1)$ if $\sigma_{j_k} = 1$.

- After the last (n-th) step of computation P accepts the input $\tilde{\sigma}$ with probability $P_{acc}(\tilde{\sigma}) = \sum_{i \in F} \mu_i$. If $f(\tilde{\sigma}) = 1$ then we have $P_{acc}(\tilde{\sigma}) \geq 1/2 + \epsilon$, else we have $P_{acc}(\tilde{\sigma}) \leq 1/2 - \epsilon$.

We assume P reads inputs in the natural order $x_1, \dots, x_n$ without loss of generality. We consider all inputs $\tilde{\sigma}_n, \dots, \tilde{\sigma}_1$, such that $\tilde{\sigma}_i = \tilde{\sigma}_i^0 \tilde{\sigma}_i^1$, where $\tilde{\sigma}_i^0 = \underbrace{0 \dots 0}_{n-i}$, $\tilde{\sigma}_i^1 = \underbrace{1 \dots 1}_{i}$.

For $i \in \{1, \dots, n\}$ we denote μ^i probability distribution after reading the part $\tilde{\sigma}_i^0$. That is, $\mu^i = \mu^0 A(0) \cdots A(0) = \mu^0 A^{n-i}(0)$. There are only ones in the $\tilde{\sigma}_i^1$, hence a computation after reading $\tilde{\sigma}_i^0$ can be described by a Markov chain. In this case μ^i is the initial probability distribution for a Markov process, $A(1)$ is the transition probability matrix. According to the classification of Markov chains described in the Section 2 of [10], the states of the Markov chain are decided into ergodic and transient states. An ergodic set of states is set which a process cannot leave if it once entered. A transient set of states is a set which a process can leave, but cannot return if it once left. An ergodic state is an element of an ergodic set. A transient state is an element of transient set.

An arbitrary Markov chain C has at least one ergodic set. C can be a Markov chain without any transient set. If a Markov chain C has more than one ergodic set, then there is absolutely no interaction between these sets. Hence we have two or more unrelated Markov chains lumped together. These chains we can study separately. If a Markov chain consist of a single ergodic set, then the chain called an ergodic chain. According to the classification, every ergodic chain is either regular or cyclic.

If an ergodic chain is regular, then sufficiently high power of the state transition matrix A has only positive elements. Thus no matter where the process starts, after sufficient large time of steps it can be in any state. Moreover there is the limiting vector of probabilities of being in the states of the chain, not dependent of the initial state.

If a Markov chain is cyclic, then the chain has a period t and all its states are subdivided into t cyclic subsets ($t > 1$). For a given starting state a process moves through the cyclic subsets in a definite order, returning to the subset with the starting state after every t steps. It is know that after sufficient time has elapsed, the process can be in any state of the cyclic subset appropriate for the moment. Hence for every of t cyclic subsets the t-th power of state transition matrix A^t describes a regular Markov chain. Moreover if an ergodic chain is cyclic chain with the period t it has at least t states.

From the assumption $q < p_n$ of the proof we get that $t < p_n$ for every of cyclic chains. We denote D the least common multiply of all such t. Because p_n is prime, t is relatively prime to p_n, D is relatively prime to p_n, and so is any positive degree D^m of D.

Let α^k be a probability distribution after reading the part $\tilde{\sigma}_k^1$ of $\tilde{\sigma}_k$. That is, $\alpha^k = \mu^k A^k(1)$. We can assume that there is a single accepting state without loss of generality. Let α_{acc}^k be a probability of being on accepting state after reading the input $\tilde{\sigma}_k^1$. Since after every D steps a process can be in any state of set consisting accepting state, D-th power of matrix A describes a regular Markov chain for this set. From the theory of Markov chains we have that there exist α_{acc} that $\lim_{k\to\infty} \alpha_{acc}^{kD} = \alpha_{acc}$. Hence for any $\epsilon > 0$ it holds that

$$|\alpha_{acc}^{D^m} - \alpha_{acc}^{D^m p_n}| < 2\epsilon$$

for m large enough. As P $1/2+\epsilon$ computes MOD_{p_n} we have $\alpha_{acc}^{D^m p_n} \geq 1/2+\epsilon$ and $\alpha_{acc}^{D^m} \leq 1/2 - \epsilon$. The contradiction with the inequation above.

□

4 General Lower Bound for Quantum OBDDs

Below we present a general lower bound on the width of read-once quantum branching programs.

Theorem 7 *Let $\epsilon \in (0, 1/2)$. Let $f(x_1, \ldots, x_n)$ be a Boolean function $(1/2+\epsilon)$-computed (computed with margin ϵ) by quantum read-once branching program Q. Then it is holds that*

$$width(Q) = \Omega(\log width(P))$$

where P is a deterministic OBDD of minimal width computing $f(x_1, \ldots, x_n)$.

Proofs of the theorem 7 use the same idea used for proving theorem 4 and directly follows the proof. We construct a deterministic OBDD P that presents the same function f and

$$width(P) \leq \left(1 + \frac{1}{\epsilon}\right)^{2width(Q)}.$$

The lower bound for $width(Q)$ of the theorem 7 proves that the quantum OBDD constructed for MOD_{p_n} function has possible optimal width.

References

[1] F. Ablayev, C. Moore, and C. Pollett, Quantum and Stochastic Branching Programs of Bounded Width, *in Proceedings of the ICALP'2002, Lecture Notes in Computer Science, Springer-Verlag*, 343-354. See also *Electronic Colloquium on Computational Complexity*, TR02-013, 2002, available at `http://www.eccc.uni-trier.de/eccc/`

[2] F. Ablayev, A. Gainutdinova, and M. Karpinski. On computational Power of quantum branching programs. *Proc. FCT 2001*, Lecture Notes in Computer Science 2138: 59–70, 2001.

[3] F. Ablayev and M. Karpinski. A lower bound for integer multiplication on randomized read-once branching programs. *Electronic Colloquium on Computational Complexity* TR 98-011, 1998. `http://www.eccc.uni-trier.de/eccc`

[4] A. Ambainis and R. Freivalds, 1-way quantum finite automata: strengths, weaknesses and generalization, In *Proceeding of the 39th IEEE Conference on Foundation of Computer Science*, 1998, 332-342. See also quant-ph/9802062 v3.

[5] P. Alexandrov. *Introduction to set theory and general topology.* Berlin, 1984.

[6] A. Ambainis and R. Freivalds. 1-way quantum finite automata: strengths, weakness, and generalizations. *Proc. 39th IEEE Symp. on Foundations of Computer Science (FOCS)*, 332–342, 1998.

[7] A. Ambainis, L. Schulman, and U. Vazirani. Computing with Highly Mixed States. *Proc. 32nd Annual ACM Symp. on Theory of Computing (STOC)*, 697–704, 2000.

[8] D.A. Barrington. Bounded-width polynomial branching programs recognize exactly those languages in NC^1. *Journal of Computer and System Sciences* 38(1): 150–164, 1989.

[9] D.A. Barrington and D. Therien. Finite Monoids and the Fine Structure of NC^1 *Journal of the ACM* 35(4): 941–952, 1988.

[10] J. G. Kemeny, J. L. Snell, Finite Markov Chains, Van Nostrand Company, INC, 1960.

[11] A. Kondacs and J. Watrous On the power of quantum finite automata. *Proc. of the 38th IEEE Symp. on Foundations of Computer Science (FOCS)*, 66–75, 1997.

[12] C. Moore and J.P. Crutchfield. Quantum automata and quantum grammars. *Theoretical Computer Science* 237: 275–306, 2000.

[13] C. Moore, D. Thérien, F. Lemieux, J. Berman, and A. Drisko. Circuits and Expressions with Non-Associative Gates. *Journal of Computer and System Sciences* 60: 368–394, 2000.

[14] M. Nakanishi, K. Hamaguchi, and T. Kashiwabara. Ordered quantum branching programs are more powerful than ordered probabilistic branching programs under a bounded-width restriction. *Proc. 6th Annual International Conference on Computing and Combinatorics (COCOON)* Lecture Notes in Computer Science 1858: 467–476, 2000.

[15] M.A. Nielson and I.L. Chuang. *Quantum Computation and Quantum Information.* Cambridge University Press. 2000.

[16] M. Rabin. Probabilistic automata. *Information and Control* 6: 230–245, 1963.

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

[18] Ingo Wegener. *Branching Programs and Binary Decision Diagrams.* SIAM Monographs on Discrete Mathematics and Applications. 2000.

[19] A.C. Yao. Lower Bounds by Probabilistic Arguments *Proc. of the 24th IEEE Symp. on Foundations of Computer Science (FOCS)*, 420–428, 1983.