Very Narrow Quantum OBDDs and Width Hierarchies for Classical OBDDs

F. Ablayev1*, A. Gainutdinova1**, K. Khadiyev1***, and A. Yakaryılmaz2****

(Submitted by M. M. Arslanov)

1Department of Theoretical Cybernetics, Institute of Computational Mathematics and Information Technologies, Kazan (Volga Region) Federal University, Kremlevskaya ul. 18, Kazan, 420008 Tatarstan, Russia

2Hurriyet Mah. 1755. Sok. 8/5, Yenisehir, Mersin, 33120, Turkey

Received March 12, 2016

Abstract—In the paper we investigate Ordered Binary Decision Diagrams (OBDDs) — a model for computing Boolean functions. We present a series of results on the comparative complexity for several variants of OBDD models.

• We present results on the comparative complexity of classical and quantum OBDDs. We consider a partial function depending on a parameter $k$ such that for any $k > 0$ this function is computed by an exact quantum OBDD of width 2, but any classical OBDD (deterministic or stable bounded-error probabilistic) needs width $2^k + 1$.

• We consider quantum and classical nondeterminism. We show that quantum nondeterminism can be more efficient than classical nondeterminism. In particular, an explicit function is presented that is computed by a quantum nondeterministic OBDD of constant width but any classical nondeterministic OBDD for this function needs non-constant width.

• We also present new hierarchies on widths of deterministic and nondeterministic OBDDs.

DOI: 10.1134/S199508021606007X

Keywords and phrases: OBDD, quantum computation, nondeterminism, partial functions, width hierarchy.

1. INTRODUCTION

Branching programs (PBs) are one of the well known models of computation. This model have been shown useful in a variety of domains, such as hardware verification, model checking, and other applications (see for example the book by Wegener [25]). It is known that the class of Boolean functions computed by polynomial size branching programs coincides with the class of functions computed by non-uniform log-space machines. Moreover, branching programs are a convenient model for considering their various (natural) restrictive variants and various complexity measures such as size (number of inner nodes), length, and width.

An important class of restrictive branching programs is that of oblivious read–once branching programs, also known in applied computer science as Ordered Binary Decision Diagrams (OBDD)[25]. Oblivious model of BPs is a branching program whose set $S$ of nodes is partitioned into subsets (levels) $S_1, S_2, \ldots$ such that for each level all nodes query the same variable. Read–once property means that BP has exactly $n + 1$ levels and each variable $x \in \{x_1, \ldots, x_n\}$ is tested only once. Since the length of

---

*E-mail: fablayev@gmail.com
**E-mail: aida.ksu@gmail.com
***E-mail: kamilhadi@gmail.com
****E-mail: abuzer@boun.edu.tr
an OBDD is exactly \( n \) (the number of input variables), the main complexity measure is “width”, where width is the maximum number of nodes on a level.

OBDDs can also be seen as nonuniform automata (see for example [2]). Different variants of OBDDs were considered, i.e. deterministic, nondeterministic, probabilistic, and quantum, and many results have been proved on the comparative power of deterministic, nondeterministic, and randomized OBDDs [25]. In particular, Ablayev and Karpinski [5] presented the first function that is polynomially easy for randomized OBDDs and exponentially hard for deterministic and even nondeterministic OBDDs. More specifically, it was proven that the OBDD variants of \( \text{coRP} \) and \( \text{NP} \) are different.

In the last decades a quantum model of OBDD came into play [3, 19, 22]. It was proven that quantum OBDDs can be exponentially cheaper than classical ones and it was shown that this bound is tight [4].

In this paper we present the first results on the comparative complexity for classical and quantum OBDDs computing partial functions. In the last part of the paper, we focus on width complexity of deterministic and nondeterministic OBDDs, which have been investigated in several papers (see for more information and citations [13, 14]). Here we present very strict hierarchies for the classes of Boolean functions computed by deterministic and nondeterministic OBDDs.

The paper is organized as follows. Section 2 contains the definitions and notation used in the paper. In Section 3, we compare classical and exact quantum OBDDs. We consider a partial function, depending on a parameter \( k \), such that for any \( k > 0 \) this function is computed by an exact quantum OBDD of width \( 2 \) but deterministic, nondeterministic, and bounded-error probabilistic OBDDs need width \( 2^{k+1} \). In Section 4, we consider quantum and classical nondeterminism. We show that quantum nondeterministic OBDDs can be more efficient than their classical counterparts. We present an explicit function which is computed by a quantum nondeterministic OBDD with constant width, but any classical nondeterministic OBDD needs non-constant width. Section 5 contains our results on hierarchies on the sublinear (5.1) and larger (5.2) widths of deterministic and nondeterministic OBDDs.

2. PRELIMINARIES

We refer to [25] for more information on branching programs. The main model investigated throughout the paper is OBDD (Ordered Binary Decision Diagram), a restricted version of branching programs.

In this paper we use the following notation for vectors. We use subscripts for enumerating the elements of vectors and strings and superscripts for enumerating vectors and strings. For a binary string \( \nu \), \( \#_1(\nu) \) and \( \#_0(\nu) \) are the numbers of 1s and 0s in \( \nu \), respectively. We denote \( \#^k_0(\nu) \) and \( \#^k_1(\nu) \) to be the numbers of 1s and 0s in the first \( k \) elements of string \( \nu \), respectively.

For a given \( n > 0 \), a probabilistic OBDD \( P_n \) of width \( d \) is a natural generalization of deterministic OBDD and can be described as a specific linear BP as follows. \( P_n \), defined on the set \( X = \{x_1, \ldots, x_n\} \) of boolean variables, is a 4-tuple \( P_n = (v^0, T, \text{Accept}, \pi) \), where

- \( v^0 \) is a \( d \)-dimensional zero–one stochastic vector (initial state of \( P_n \) – initial probability distribution of nodes on the level \( S_1 \));
- \( T = \{T_j : 1 \leq j \leq n\} \) such that \( T_j = (A_j(0), A_j(1)) \) is an ordered pair of (left) stochastic \( d \times d \)-matrices, representing the transitions, where, at the \( j \)-th step, \( A_j(0) \) or \( A_j(1) \), determined by a corresponding input bit, is applied;
- \( \text{Accept} \subseteq \{1, \ldots, d\} \) is the set of accepting nodes;
- \( \pi \) is a permutation of \( \{1, \ldots, n\} \) defining the order of testing the input bits.

For any given input \( \nu \in \{0, 1\}^n \), computation of \( P_n \) on \( \nu \) can be traced by a stochastic vector which is initially \( v^0 \). In each step \( j, 1 \leq j \leq n \), the input bit \( x_{\pi(j)} \) is tested and then the corresponding stochastic operator is applied:

\[
v^j = A_j(x_{\pi(j)})v^{j-1},
\]
where \( \nu^j \) represents the probability distribution vector of nodes after the \( j \)-th step, \( 1 \leq j \leq n \).

The accepting probability \( Pr_{\text{accept}}(\nu) \) of \( P_n \) on \( \nu \) is

\[
Pr_{\text{accept}}(\nu) = \sum_{i \in \text{Accept}} \nu^n_i.
\]

We say that a function \( f \) is computed by \( P_n \) with bounded error if there exists an \( \varepsilon \in (0, 1/2] \) such that \( P_n \) accepts all inputs from \( f^{-1}(1) \) with a probability at least \( 1/2 + \varepsilon \) and \( P_n \) accepts all inputs from \( f^{-1}(0) \) with a probability at most \( 1/2 - \varepsilon \). We say that \( P_n \) computes \( f \) exactly if \( \varepsilon = 1/2 \).

In particular, a deterministic OBDD is a probabilistic OBDD restricted to use only 0-1 transition matrices. In other words, the system is always in a single node and, from each node, there is exactly one outgoing transition for each tested input bit.

A nondeterministic OBDD (NOBDD) can have the ability of making more than one outgoing transition for each tested input bit from each node and so the program can follow more than one computational path and if one of the paths ends with an accepting node, then the input is accepted (rejected, otherwise).

- An OBDD is called stable if each transition set \( T_j \) is identical for each level.
- An OBDD is called ID (ID-OBDD) if the input bits are tested in the order \( \pi = (1, 2, \ldots, n) \). If a stable ID-OBDD has a fixed width and transition rules for each \( n \), then it can be considered as a realtime finite automaton.

Quantum computation is a generalization of classical computation [24]. Therefore, each quantum model can simulate its probabilistic counterparts. In some cases, on the other hand, the quantum models are defined in a restricted way, e.g., using only unitary operators during the computation followed by a single measurement at the end, and so they may not simulate their probabilistic counterparts. The literature on quantum automata contains many results of this kind such as [18, 6, 9]. A similar result was also given for OBDDs in [22], in which a function with a small size of deterministic OBDD was given but the quantum OBDD defined in a restricted way needs exponential size to solve this function.

Quantum OBDDs that are defined with the general quantum operators, i.e. superoperators [23, 24, 27], followed by a measurement on the computational basis at the end can simulate their classical counterpart with the same size and width. So we can always conclude that any quantum class contains its classical counterpart.

In this paper, we follow our quantum results based on stable ID-OBDDs, which are simply realtime quantum finite automata algorithms, already given in the literature. But, we give the definition of quantum OBDDs for the completeness of the paper. We refer the reader to [8] for a review of the basics of quantum operators and quantum automata models.

A quantum OBDD is the same as a probabilistic OBDD with the following modifications:

- The quantum part is represented by a \( d \)-dimensional Hilbert space. The state of the system (mixed state) can be represented by a density operator (matrix). The initial one is \( \rho^0 = |q_0\rangle \langle q_0| \) where \( q_0 \) corresponds to the initial node.
- Instead of a stochastic matrix, we apply a superoperator in each step. That is, \( T = \{ T_j : 1 \leq j \leq n \text{ and } T_j = (\mathcal{E}_j^0, \mathcal{E}_j^1) \} \), where, at the \( j \)-th step, \( \mathcal{E}_j^0 \) or \( \mathcal{E}_j^1 \), determined by the corresponding input bit, is applied,
- At the end, we make a measurement on the computational basis.

The (mixed) state of the system is updated as follows after the \( j \)-th step:

\[
\rho^j = \mathcal{E}_j^{x(n)}(\rho^{j-1}),
\]

where \( \rho^{j-1} \) and \( \rho^j \) represent the (mixed) state of the system after the \( (j-1) \)-th and \( j \)-th steps, respectively, where \( 1 \leq j \leq n \). The accepting probability of the quantum program on \( \nu \) is calculated from \( \rho^n \) as

\[
\sum_{i \in \text{Accept}} \rho_{i,i}^n.
\]
3. EXACT QUANTUM OBDDs

In [7], Ambainis and Yakaryılmaz defined a new family of unary promise problems: for any \( k > 0 \), \( A^k = (A^k_{\text{yes}}, A^k_{\text{no}}) \) such that \( A^k_{\text{yes}} = \{a^{(2i+1)}2^k : i \geq 0 \} \) and \( A^k_{\text{no}} = \{a^{(2i+1)}2^k : i \geq 0 \} \). They showed that each member of this family \( A^k \) can be solved exactly by a 2-state realtime quantum finite automaton (QFA), but any exact probabilistic finite automaton (PFA) needs at least \( 2^{k+1} \) states. Recently, Rashid and Yakaryılmaz [21] showed that bounded-error realtime PFAs also need at least \( 2^{k+1} \) states for solving \( A^k \). Based on this promise problem, we define a partial function:

\[
\text{PartialMOD}^k_\ell(\nu) = \begin{cases} 
1, & \text{if } \#_1(\nu) = 0 \pmod{2^{k+1}}, \\
0, & \text{if } \#_1(\nu) = 2^k \pmod{2^{k+1}}, \\
*, & \text{otherwise}, 
\end{cases}
\]

where the function is not defined for the inputs mapping to “*”. We call the inputs, where the function takes the value of 1 (0), as yes-instances (no-instances).

**Theorem 1.** For any \( k \geq 0 \), \( \text{PartialMOD}^k_\ell \) can be solved exactly by a stable quantum ID-OBDD with width 2.

The OBDD can be constructed in the same way as a QFA that solves problem \( A^k \) [7].

We show that width of deterministic, nondeterministic, or bounded-error stable probabilistic OBDDs that solve \( \text{PartialMOD}^k_\ell \) cannot be less than \( 2^{k+1} \).

**Remark.** Note that the proof for deterministic OBDDs is not similar to the proof for automata because potentially the nonstability can give profit. Also this proof differs from proofs for total functions (for example, \( \text{MOD}_q \)) due to the existence of incomparable inputs. Note that classical one-way communication complexity techniques also fail for partial functions (for example, it can be shown that the communication complexity of \( \text{PartialMOD}^k_\ell \) is 1), and we need to use a more careful analysis in the proof.

A deterministic stable ID-OBDD of width \( 2^{k+1} \) for \( \text{PartialMOD}^k_\ell \) can be easily constructed. We left open the case of bounded-error non-stable probabilistic OBDDs.

**Theorem 2.** For any \( k \geq 0 \), there are infinitely many \( n \) such that any deterministic OBDD, computing the partial function \( \text{PartialMOD}^k_\ell \), has width at least \( 2^{k+1} \).

**Proof.** Let \( \nu \in \{0,1\}^n \) be an input and \( \nu = \sigma \gamma \). We call \( \gamma \) valid for \( \sigma \) if \( \nu \in (\text{PartialMOD}^k_\ell)^{-1}(0) \cup (\text{PartialMOD}^k_\ell)^{-1}(1) \). We call two substrings \( \sigma' \) and \( \sigma'' \) comparable if for all \( \gamma \) it holds that \( \gamma \) is valid for \( \sigma' \) iff \( \gamma \) is valid for \( \sigma'' \). We call two substrings \( \sigma' \) and \( \sigma'' \) nonequivalent if they are comparable and there exists a valid substring \( \gamma \) such that \( \text{PartialMOD}^k_\ell(\sigma' \gamma) \neq \text{PartialMOD}^k_\ell(\sigma'' \gamma) \).

Let \( P \) be a deterministic OBDD computing \( \text{PartialMOD}^k_\ell \). Note that paths associated with nonequivalent strings must lead to different nodes. Otherwise, if \( \sigma \) and \( \sigma' \) are nonequivalent, then there exists a valid string \( \gamma \) such that \( \text{PartialMOD}^k_\ell(\sigma \gamma) \neq \text{PartialMOD}^k_\ell(\sigma' \gamma) \) and computations on these inputs lead to the same final node.

Let \( N = 2^k \) and \( \Gamma = \{\gamma : \gamma \in \{0,1\}^{2N-1}, \gamma = 0 \cdots 01 \cdots 1\} \). We will naturally identify any string \( \nu \) with the element \( a = \#_1(\nu) \pmod{2N} \) of the additive group \( \mathbb{Z}_{2N} \). We call two strings of the same length different if the numbers of ones modulo \( 2N \) in them are different. We denote by \( \rho(\gamma^1, \gamma^2) = \gamma^1 - \gamma^2 \) the distance between numbers \( \gamma^1, \gamma^2 \). Note that \( \rho(\gamma^1, \gamma^2) \neq \rho(\gamma^2, \gamma^1) \) in general case.

Let width of \( P \) be \( t < 2N \). At each step \( i (i = 1, 2, \ldots) \) of the proof we will count the number of different strings that lead to the same node (denote this node by \( v_i \)). At the \( i \)-th step we consider the \( (2N-1)i \)-th level of \( P \).

Let \( i = 1 \). By the pigeon-hole principle there exist two different strings \( \sigma^1 \) and \( \sigma^2 \) from the set \( \Gamma \) such that the corresponding paths lead to the same node \( v_1 \) of the \( (2N-1) \)-th level of \( P \). Note that \( \rho(\sigma^1, \sigma^2) \neq N \), because in this case \( \sigma^1 \) and \( \sigma^2 \) are nonequivalent and cannot lead to the same node.

We will show by induction that at each step of the proof the number of different strings that lead to the same node increases.

\(^1\)The same result is also proved for two-way nondeterministic finite automata by Geffert and Yakaryılmaz [12].
Step 2. By the pigeon-hole principle there exist two different strings \( \gamma^1 \) and \( \gamma^2 \) from the set \( \Gamma \) such that corresponding paths, going from the node \( v_1 \), lead to the same node \( v_2 \) of the \( (2N - 1) \)-th level of \( P \). In this case the strings \( \sigma^1\gamma^1, \sigma^2\gamma^1, \sigma^1\gamma^2, \) and \( \sigma^2\gamma^2 \) lead to the node \( v_2 \). Note that \( \rho(\gamma^1, \gamma^2) \neq N \) because \( \sigma^1\gamma^1 \) and \( \sigma^1\gamma^2 \) are nonequivalent and cannot lead to the same node.

Adding the same number does not change the distance between the numbers, so we have

\[
\rho(\sigma^1 + \gamma^1, \sigma^2 + \gamma^1) = \rho(\sigma^1, \sigma^2)
\]

and

\[
\rho(\sigma^1 + \gamma^2, \sigma^2 + \gamma^2) = \rho(\sigma^1, \sigma^2).
\]

Let \( \gamma^2 > \gamma^1 \). Denote \( \Delta = \gamma^2 - \gamma^1 \). Let us count the number of different numbers among \( \sigma^1 + \gamma^1, \sigma^2 + \gamma^1, \sigma^1 + \gamma^1 + \Delta, \) and \( \sigma^2 + \gamma^1 + \Delta \). Because \( \sigma^1 \) and \( \sigma^2 \) are different and \( \rho(\sigma^1, \sigma^2) \neq N \), the numbers from the pair \( \sigma^1 + \gamma^1, \sigma^2 + \gamma^1 + \Delta \) coincide with corresponding numbers from the pair \( \sigma^1 + \gamma^1 + \Delta \) and \( \sigma^2 + \gamma^1 + \Delta \) if\( \Delta = 0 \) (mod 2N). But \( \Delta \neq 0 \) (mod 2N) since the numbers \( \gamma^1 \) and \( \gamma^2 \) are different and \( \gamma^1, \gamma^2 < 2N \). The numbers \( \sigma^1 + \gamma^1 + \Delta \) and \( \sigma^2 + \gamma^1 + \Delta \) cannot be a permutation of numbers \( \sigma^1 + \gamma^1 \) and \( \sigma^2 + \gamma^1 \) since \( \rho(\gamma^1, \gamma^2) \neq N \) and \( \rho(\sigma^1, \sigma^2) \neq N \). In this case, at least 3 numbers from \( \sigma^1 + \gamma^1, \sigma^2 + \gamma^1, \sigma^1 + \gamma^2, \) and \( \sigma^2 + \gamma^2 \) are different.

Step of induction. Let at the step \( i - 1 \) the numbers \( \sigma^1, \ldots, \sigma^i \) are different and the corresponding paths lead to the same node \( v_{i-1} \) of the \( (2N - 1)(i - 1) \)-th level of \( P \).

By the pigeonhole principle, there exist two different strings \( \gamma^1 \) and \( \gamma^2 \) from the set \( \Gamma \) such that the corresponding paths, going from the node \( v_{i-1} \), lead to the same node \( v_i \) of the \( (2N - 1)i \)-th level of \( P \). So the paths \( \sigma^1\gamma^1, \ldots, \sigma^i\gamma^1, \sigma^1\gamma^2, \ldots, \sigma^i\gamma^2 \) lead to the same node \( v_i \). Let us estimate the number of different strings among them. Note that \( \rho(\gamma^1, \gamma^2) \neq N \), because the strings \( \sigma^1\gamma^1 \) and \( \sigma^1\gamma^2 \) are nonequivalent and cannot lead to the same node.

The numbers \( \sigma^1, \ldots, \sigma^i \) are different and \( \rho(\sigma^l, \sigma^j) \neq N \) for each pair \( (l, j) \) such that \( l \neq j \). Let \( \sigma^1 < \cdots < \sigma^i \). We will show that among \( \sigma^1 + \gamma^1, \ldots, \sigma^i + \gamma^1 \) and \( \sigma^1 + \gamma^1 + \Delta, \ldots, \sigma^i + \gamma^1 + \Delta \) at least \( i + 1 \) numbers are different.

The sequence of numbers \( \sigma^1 + \gamma^1, \ldots, \sigma^i + \gamma^1 \) coincides with the sequence \( \sigma^1 + \gamma^1 + \Delta, \ldots, \sigma^i + \gamma^1 + \Delta \) if\( \Delta = 0 \) (mod 2N). But \( \Delta \neq 0 \) (mod 2N) since \( \gamma^1 \) and \( \gamma^2 \) are different and \( \gamma^1, \gamma^2 < 2N \).

Suppose that the sequence \( \sigma^1 + \gamma^1 + \Delta, \ldots, \sigma^i + \gamma^1 + \Delta \) is a permutation of the sequence \( \sigma^1 + \gamma^1, \ldots, \sigma^i + \gamma^1 \). In this case we have numbers \( a_0, \ldots, a_r \) from \( \mathbb{Z}_{2N} \) such that all \( a_j \) are from the sequence \( \sigma^1 + \gamma^1, \ldots, \sigma^i + \gamma^1, a_0 = a_r = \sigma^1 + \gamma^1, \) and \( a_j = a_{j-1} + \Delta \), where \( j = 1, \ldots, r \). In this case we have that \( r\Delta = 2Nm \). Because \( N = 2^k, \Delta < 2N, \) and \( \Delta \neq N \) we get that \( r \) is even. For \( z = r/2 \) we have \( z\Delta = Nm \). Since all numbers from \( \sigma^1 + \gamma^1, \ldots, \sigma^i + \gamma^1 \) are different, then \( \rho(a_0, a_z) = N \). So we have that \( a_0 \) and \( a_z \) are nonequivalent, but the corresponding strings lead to the same node \( v_i \). From this we conclude that after the \( i \)-th step at least \( i + 1 \) different strings lead to the same node \( v_i \).

At the \( N \)-th step, we have that \( N + 1 \) different strings lead to the same node \( v_N \). There must be at least two nonequivalent strings among these strings. Thus we can conclude that \( P \) does not compute the function \( \text{PartialMOD}^N_k \) correctly.

**Theorem 3.** For any \( k \geq 0 \), there are infinitely many \( n \) such that any nondeterministic OBDD, computing the partial function \( \text{PartialMOD}^N_k \), has width at least \( 2^{k+1} \).

**Proof.** The proof is similar to the deterministic case with the following modifications. Denote \( N = 2^k \). Assume that there exists an NOBDD \( P \) that computes \( \text{PartialMOD}^N_k \) such that width of \( P \) is \( t < 2^{k+1} \). We consider only the accepting paths in \( P \). Note that accepting paths, corresponding to nonequivalent strings, can not pass through the same node. Let \( \Gamma = \{ \gamma : \gamma \in \{0, 1\}^{2N-1}, \gamma = \underbrace{0\ldots0}_{2N-1-j}1\ldots1, j = 0, \ldots, 2N - 1 \} \).

We denote by \( V_l \) a set of nodes of the \( l \)-th level of \( P \). By our assumption, \( |V_l| < 2N \) for each \( l = 0, \ldots, n \).

Let \( V_l(\gamma, V) \) be a set of nodes of the \( l \)-th level through which accepting paths, leading from the nodes of a set \( V \) and corresponding to the string \( \gamma \), pass.
We will prove the theorem step by step. At the $i$-th step ($i = 1, 2, \ldots$) of the proof we consider the $(2N - 1)i$-th level of $P$. Because $|V_{2N-1}| < 2N$, at the first step of the proof we have that there exist two different strings $\sigma^1, \sigma^2 (\sigma^1, \sigma^2 \in \Gamma)$ such that $V_{2N-1}(\sigma^1, V_0) \cap V_{2N-1}(\sigma^2, V_0) \neq \emptyset$. Denote by $G_1$ this non empty intersection. Next, we continue our proof by considering only such accepting paths that pass through the nodes of $G_1$.

Using the same ideas as in the deterministic case we have that the number of strings with different amounts of 1’s by modulo 2N, such that corresponding accepting paths lead to the same set of nodes, increases with each step of the proof.

Arguing as in the proof of Theorem 2, we see that at the $N$-th step of the proof there must exist two different non-equivalent strings such that corresponding accepting paths lead to the same set $G_N$ of nodes of the $(2N - 1)N$-th level. This implies that $P$ does not compute the function $\text{PartialMOD}_n^k$ correctly.

**Theorem 4.** For any $k \geq 0$, there are infinitely many $n$ such that any stable probabilistic OBDD, computing $\text{PartialMOD}_n^k$ with bounded error, has width at least $2^{k+1}$.

**Proof.** The proof is based on the technique of Markov chains.

We assume that there is a stable probabilistic OBDD $P$ of width $d < 2^{k+1}$ computing $\text{PartialMOD}_n^k$ with probability $1/2 + \epsilon$ for a fixed $\epsilon \in (0, 1/2]$. Let $v^j = (v^j_1, \ldots, v^j_d)$ be a probability distribution of nodes of $P$ at the $j$-th level, where $v^j_i$ is the probability of being in the $i$-th node at the $j$-th level. We can describe the computation of $P$ on the input $\nu = \nu_1, \ldots, \nu_n$ as follows:

- The computation of $P$ starts from the initial probability distribution vector $\nu^0$.

- At the $j$-th step, $1 \leq j \leq n$, $P$ reads input $\nu_{i_j}$ and transforms the vector $\nu^{j-1}$ to $\nu^j = A\nu^{j-1}$, where $A$ is the $q \times q$ stochastic matrix, $A = A(0)$ if $\nu_{i_j} = 0$ and $A = A(1)$ if $\nu_{i_j} = 1$.

- After the last ($n$-th) step of the computation $P$ accepts the input $\nu$ with the probability $P_{\text{acc}}(\nu) = \sum_{j \in \text{Accept}} v^j$. If $\text{PartialMOD}_n^k(\nu) = 1$, then we have $P_{\text{acc}}(\nu) \geq 1/2 + \epsilon$ and if $\text{PartialMOD}_n^k(\nu) = 0$, then we have $P_{\text{acc}}(\nu) \leq 1/2 - \epsilon$.

Without loss of generality we assume that $P$ reads the inputs in the natural order $x_1, \ldots, x_n$. We consider only inputs $\hat{\nu}_n, \ldots, \hat{\nu}_1$ of the form $\hat{\nu}_i = \hat{\nu}^0_i \hat{\nu}^1_i$, where $\hat{\nu}^0_i = 0 \cdots 0, \hat{\nu}^1_i = 1 \cdots 1$.

For $i \in \{1, \ldots, n\}$, we denote by $\alpha^i$ the probability distribution after reading the part $\hat{\nu}^0_i$, i.e. $\alpha^i = A^{n-i}(0)\nu^0$. There are only 1’s in $\hat{\nu}^1_i$, hence the computation after reading $\hat{\nu}^1_i$ can be described by a Markov chain. In this case $\alpha^i$ is the initial probability distribution for a Markov process and $A(1)$ is the transition probability matrix.

According to the classification of Markov chains described in the Section 2 of the book by Kemeny and Snell [16], the states of the Markov chain are subdivided into ergodic and transient states. An ergodic set of states is a set which a process cannot leave once it has entered. A transient set of states is a set which a process can leave, but cannot return once it has left. An ergodic state is an element of an ergodic set. A transient state is an element of a 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 can be studied separately. If a Markov chain consists of a single ergodic set, then the chain is called an ergodic chain. According to the classification mentioned above, every ergodic chain is either regular or cyclic.

If an ergodic chain is regular, then for sufficiently high powers of the state transition matrix, $A$ has only positive elements. Thus, no matter where the process starts, after a sufficiently large number of steps it can be in any state. Moreover, there is a limiting vector of probabilities of being in the states of the chain, that does not depend on 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 known that after sufficient time has elapsed, the process can be in any state of the cyclic subset appropriate for the moment. Hence for each of \( t \) cyclic subsets the \( t \)-th power of the state transition matrix \( A^t \) describes a regular Markov chain. Moreover, if an ergodic chain is a cyclic chain with the period \( t \), it has at least \( t \) states.

Let \( C_1, \ldots, C_t \) be cyclic subsets of states of Markov chain with periods \( t_1, \ldots, t_t \) respectively, and \( D \) be the least common multiple of \( t_1, \ldots, t_t \).

**Lemma 1.** \( D \) must be a multiple of \( 2^{k+1} \).

**Proof.** Assume that \( D \) is not a multiple of \( 2^{k+1} \). After every \( D \) steps, the process can be in any set of states containing the accepting state and the \( D \)-th power of \( A \) describes a regular Markov chain. From the theory of Markov chains we have that there exists an \( \alpha_{\text{acc}} \) such that \( \lim_{r \to \infty} \alpha_{\text{acc}}^{r \cdot D} = \alpha_{\text{acc}} \), where \( \alpha_{\text{acc}}^{r \cdot D} \) represents the probability of process being in accepting state(s) after the \( i \)-th step. Hence for any \( \varepsilon > 0 \) it holds that

\[
|\alpha_{\text{acc}}^{r \cdot D} - \alpha_{\text{acc}}^{r' \cdot D}| < 2\varepsilon
\]

for some big enough \( r, r' \). Since \( D \) is not a multiple of \( 2^{k+1} \), it can be represented as \( D = m \cdot 2^l \) (\( l \leq k \), \( m \) is odd). For any odd \( s \), the number \( s \cdot D \) is not a multiple of \( 2^{k+1} \). Because \( P \) is supposed to solve Partial\( k \) with probability \( 1/2 + \varepsilon \) so we have \( \alpha_{\text{acc}}^{s \cdot m \cdot 2^{k+1+l}} \geq 1/2 + \varepsilon \) and \( \alpha_{\text{acc}}^{s \cdot m \cdot 2^{k+1+l}} \leq 1/2 - \varepsilon \). This contradicts with the inequality above for big enough \( s \). \( \square \)

**Lemma 2.** There is a circle of period \( t \), where \( t \) is a multiple of \( 2^{k+1} \).

**Proof.** The proof follows from the fact that \( D \) is a multiple of \( 2^{k+1} \) that implies existence of \( t \in \{t_1, \ldots, t_t\} \) such that \( t \) is a power of \( 2 \). Among such \( t \) there must be a multiple of \( 2^{k+1} \). Otherwise \( D \) (the least common multiple of \( t_1, \ldots, t_t \)) can not be a multiple of \( 2^{k+1} \). \( \square \)

Since there exists a circle of period \( t \), where \( t \) is a multiple of \( 2^{k+1} \), we conclude that \( q \geq 2^{k+1} \). \( \square \)

### 4. NonDeterministic Quantum and Classical OBDDs

In [26], Yakaryılmaz and Say showed that nondeterministic QFAs define a superset of regular languages, called exclusive stochastic languages [20]. This class contains the complements of some interesting languages: \( \text{PAL} = \{w \in \{0, 1\}^* : w = w^r\} \), where \( w^r \) is the reverse of \( w \), \( \text{O} = \{w \in \{0, 1\}^* : \#_1(w) = \#_0(w)\} \), \( \text{SQUARE} = \{w \in \{0, 1\}^* : \#_1(w) = (\#_0(w))^2\} \), and \( \text{POWER} = \{w \in \{0, 1\}^* : \#_1(w) = 2\#_0(w)\} \).

Based on these languages we define three symmetric functions. For any input \( \nu \in \{0, 1\}^n \):

\[
\text{NotO}_n(\nu) = \begin{cases} 0, & \text{if } \#_0(\nu) = \#_1(\nu), \\ 1, & \text{otherwise}; \end{cases}
\]

\[
\text{NotSQUARE}_n(\nu) = \begin{cases} 0, & \text{if } (\#_0(\nu))^2 = \#_1(\nu), \\ 1, & \text{otherwise}; \end{cases}
\]

\[
\text{NotPOWER}_n(\nu) = \begin{cases} 0, & \text{if } 2\#_0(\nu) = \#_1(\nu), \\ 1, & \text{otherwise}. \end{cases}
\]

**Theorem 5.** The functions \( \text{NotO}_n, \text{NotSQUARE}_n, \) and \( \text{NotPOWER}_n \) can be computed by a nondeterministic quantum OBDD of constant width.

For each of these three functions we can define a nondeterministic quantum (stable ID-) OBDDs with constant width based on nondeterministic QFAs for the languages \( O, \text{SQUARE}, \) and \( \text{POWER} \) respectively [26].

The complements of \( \text{PAL}, O, \text{SQUARE}, \) and \( \text{POWER} \) cannot be recognized by classical nondeterministic finite automata. But, for example, the function version of the complement of \( \text{PAL}, \)
NotPAL\(_n\), which returns 1 only for the non-palindrome inputs, is quite easy since it can be computed by a deterministic OBDD of width 3. Note that the order of reading variables used by such an OBDD is not natural (1, \ldots, n). We show that this is not the case for the function versions of the complements of the three other languages.

**Theorem 6.** There are infinitely many \(n\) such that any NOBDD \(P_n\), computing NotO\(_n\), has width at least \(\lfloor \log n \rfloor + 1\).

**Proof.** The proof is based on the complexity properties of the function NotO\(_n\), which we discuss in the next lemma.

**Lemma 3.** There are infinitely many \(n\) such that any OBDD, computing NotO\(_n\), has width at least \(n/2 + 1\).

**Proof.** Let \(n\) be an even integer and \(P_n\) be an OBDD that computes NotO\(_n\). Assume that \(P_n\) uses the natural order of reading variables. For any other order the proof is similar. We consider the \(n/2\)-th level.

Let \(\Sigma = \{\sigma^i \in \{0, 1\}^{n/2} : \sigma^i = 1 \ldots 1 0 \ldots 0, 0 \leq i \leq n/2\}, \sigma^i\) and \(\sigma^j (i \neq j)\) be any pair from \(\Sigma\).

Assume that \(P_n\) reaches a node \(v\) by the input \(\sigma^i\) and a node \(v'\) by the input \(\sigma^j\).

We show that \(v \neq v'\). For this we take \(\gamma \in \{0, 1\}^{n/2}\) such that \(\#_0(\gamma) = i (\#_1(\gamma) = n/2 - i)\). Let us consider the computation on the input \(\sigma^i \gamma\). Note that \(P_n\) reaches a rejecting node from \(v\) by using \(\gamma\) because

\[
\#_1(\sigma^i) + \#_1(\gamma) = n/2 = \#_0(\sigma^i) + \#_0(\gamma).
\]

Let us consider the computation on the input \(\sigma^j \gamma\). The OBDD \(P_n\) reaches an accepting node from \(v\) by using \(\gamma\) because

\[
\#_1(\sigma^j) + \#_1(\gamma) = j + (n/2 - i) = n/2 + (j - i) \neq n/2.
\]

This yields \(v \neq v'\). Note that \(|\Sigma| = n/2 + 1\). Hence the width of the \(n/2\)-th level is at least \(n/2 + 1\). This means that width of \(P_n\) is at least \(n/2 + 1\).

The proof of the theorem is based on the well-known relation between nondeterministic and deterministic space complexity. Namely, if a Boolean function \(f(X)\) is computed by an NOBDD \(P\) of width \(d\), then there exists a deterministic OBDD \(P'\), computing \(f\), which has width \(2^d\). By Lemma 3 we have that an OBDD for the function NotO\(_n\) has width at least \(n/2 + 1\). Thus any NOBDD computing NotO\(_n\) has width \(\geq \log(n/2 + 1) > \log n - 1\).

Using the same reasoning we can show that there are infinitely many \(n\) such that any NOBDD \(P_n\) that computes the function NotSQUARE\(_n\) has width at least \(\Omega(\log(n))\), and any NOBDD \(P'_n\) that computes the function NotPOWERN\(_n\) has width at least \(\Omega(\log \log(n))\).

### 5. Hierarchies for Deterministic and Nondeterministic OBDDs of Small Width

We denote OBDD\(_d\) and NOBDD\(_d\) to be the sets of Boolean functions that can be computed by OBDDs and NOBDDs of width \(d = d(n)\) respectively, where \(n\) is the number of variables. In this section, we present some width hierarchies for OBDD\(_d\) and NOBDD\(_d\). Moreover, we discuss relations between these classes. We consider OBDD\(_d\) and NOBDD\(_d\) with small (sublinear) widths and large widths. The similar width hierarchy for read \(k\) times OBDDs (\(k\)-OBDDs) for small width is presented in[17]. OBDD model is particular case of \(k\)-OBDD (for \(k = 1\)). Note that hierarchy in[17] is not tight, but following hierarchy is tight.

We have the following width hierarchy for the deterministic and nondeterministic models.

**Theorem 7.** For any integer \(n\), \(d = d(n)\), and \(1 < d \leq n/2\), we have

\[
\text{OBDD}^{d-1} \subsetneq \text{OBDD}^d \quad \text{and} \quad \text{NOBDD}^{d-1} \subsetneq \text{NOBDD}^d.
\]

**Proof.** It is obvious that OBDD\(_{d-1}\) \(\subset\) OBDD\(_d\) and NOBDD\(_{d-1}\) \(\subset\) NOBDD\(_d\). Let us show the inequalities of these classes. For this purpose we use the complexity properties of the Boolean function MOD\(_n\).

LOBACHEVSKII JOURNAL OF MATHEMATICS  Vol. 37  No. 6  2016
Let $k$ be a number such that $1 < k \leq n/2$. For any given input $\nu \in \{0, 1\}^n$,

$$\text{MOD}_n^k(\nu) = \begin{cases} 1, & \text{if } \#_1(\nu) \equiv 0 \pmod{k}, \\ 0, & \text{otherwise}. \end{cases}$$

**Lemma 4.** There is an OBDD (and so an NOBDD) $P_n$ of width $d$ that computes the function $\text{MOD}_n^k$ and $d = k$.

*Proof.* At each level, $P_n$ counts the number of 1’s by modulo $k$. $P_n$ answers 1 if the number in the last step is zero. It is clear that width of $P_n$ is $k$. \hfill \Box

**Lemma 5.** Any OBDD and NOBDD, computing $\text{MOD}_n^k$, has width at least $k$.

*Proof.* The proof is based on the pigeonhole principle. Let $P$ be a deterministic OBDD computing the function $\text{MOD}_n^k$. For each input $\nu$ from the set $(\text{MOD}_n^k)^{-1}(1)$ there must be exactly one path in $P$ leading from the source node to an accepting node. Let us consider $k$ inputs $\{\nu^1, \nu^2, \ldots, \nu^k\}$ from this set such that the last $k$ bits in $\nu^j (j = 1, \ldots, k)$ contain exactly $j$’s and $(k - j)$ 0’s. Let us consider the $(n - k)$-th level of $P$. The acceptance paths for different inputs from $\{\nu^1, \nu^2, \ldots, \nu^k\}$ must pass through different nodes of the $(n - k)$-th level of $P$. So width of the $(n - k)$-th level of $P$ is at least $k$.

The proof for the nondeterministic case is similar to the deterministic one. For each input from $(\text{MOD}_n^k)^{-1}(1)$ for the function $\text{MOD}_n^k$ there must be at least one path in $P$ leading from the source node to an accepting node labelling this input. The accepting paths for different inputs from the set $\{\nu^1, \nu^2, \ldots, \nu^k\}$ must go through different nodes of the $(n - k)$-th level of $P$.

We have $\text{MOD}_n^k \in \text{OBDD}^d$ and $\text{MOD}_n^d \in \text{NOBDD}^d$ due to Lemma 4, and $\text{MOD}_n^d \not\in \text{OBDD}^{d-1}$ and $\text{MOD}_n^d \not\in \text{NOBDD}^{d-1}$ due to Lemma 5.

We say that complexity classes $C$ and $C'$ are not comparable if $C \not\subseteq C'$, $C \not\supseteq C'$ and $C' \not\subseteq C$. We have the following relationships between the deterministic and nondeterministic models.

**Theorem 8.** For any integer $n$, $d = d(n)$, and $d' = d'(n)$ such that $d \leq n/2$ and $O(\log^2 d \log \log d) < d' \leq d - 1$, we have $\text{NOBDD}^{[\log(d)]} \subseteq \text{OBDD}^d$ and $\text{OBDD}^d$ and $\text{NOBDD}^{d'}$ are not comparable.

*Proof.* We start with (1). Using the same reasoning as in the proof of Theorem 3 we can show that $\text{NOBDD}^{[\log(d)]} \subseteq \text{OBDD}^d$ and, from Lemma 5, we know that $\text{MOD}_n^d \not\in \text{NOBDD}^{[\log(d)]}$. Then we have $\text{OBDD}^d \not\subseteq \text{NOBDD}^{[\log(d)]}$.

We continue with (2). Let $k$ be even and $1 < k \leq n$. For any given input $\nu \in \{0, 1\}^n$,

$$\text{Not}_n^{\text{OBDD}}(\nu) = \begin{cases} 0, & \text{if } \#_0(\nu) = \#_1(\nu) = k/2, \\ 1, & \text{otherwise}. \end{cases}$$

Note that the function $\text{Not}_n^{\text{OBDD}}$ is identical to $\text{Not}_n$.

**Lemma 6.** Any OBDD, computing $\text{Not}_n^{\text{OBDD}}$, has width at least $k/2 + 1$.

*Proof.* The proof can be obtained by the same technique as given in the proof of Lemma 3. \hfill \Box

**Lemma 7.** There is an NOBDD $P_n$ of width $d$ that computes the function $\text{Not}_n^{\text{OBDD}}$ and $d \leq O(\log^2 k \log \log k)$.

*Proof.* We construct an NOBDD $P_n$. We use the fingerprinting method given in [11]. Let $p_1, \ldots, p_r$ be the first $r$ prime numbers satisfying $p_1p_2\cdots p_r > k$,

where $r$ is the minimum value. Note that $r \leq \log k$ since $p_r \geq 2$.

The NOBDD $P_n$ consists of $r$ parts, each of them corresponds to one of $p$ from $\{p_1, \ldots, p_r\}$. At the first step, $P_n$ nondeterministically picks a $p$. Then, this part counts the number of 1’s by modulo $p$. If this number is not equal to $(k/2 \pmod{p})$, then $P_n$ gives the answer 1, otherwise $P_n$ gives the answer 0. We need $p$ nodes for each value from 0 to $p - 1$. 

LOBACHEVSKII JOURNAL OF MATHEMATICS Vol. 37 No. 6 2016
By the Chinese Remainder Theorem [15], if \( \#_k^1(\nu) \neq k/2 \mod p_i \), then there exists at least one \( p_i \) such that \( \#_k^1(\nu) \neq k/2 \mod p_i \). This means that at least one branch gives the answer 1. But if \( \#_k^1(\nu) = k/2 \), then for all \( p_i \) we have \( \#_k^1(\nu) = k/2 \mod p_i \). This means that each branch gives the answer 0. Hence \( P_n \) computes \( \text{NotO}_k^n \).

By the Prime Number Theorem [10], we have \( p_r = O(r \ln r) \). Therefore \( p_i \leq O(r \ln r) \) for \( 1 \leq i \leq r \) since \( p_i \leq p_r \). This means the width of the \( i \)-th part is \( p_i \leq O(r \ln r) \). Because \( r \leq \log k \), the width of \( P_n \) is \( d \leq r \cdot O(r \ln r) = O(\log^2 k \ln \log k) \).

Recall that \( O(\log^2 d \log d) \leq d' \leq d - 1 \) and, by Lemma 4 and Lemma 5, we have \( \text{MOD}^d \in \text{OBDD}^d \) and \( \text{MOD}^d \notin \text{NOBDD}^d \); by Lemma 7, we have \( \text{NotO}_2^{d-1} \in \text{NOBDD}^d \); and, by Lemma 6, we have \( \text{NotO}_2^{d-1} \notin \text{OBDD}^d \). Therefore we cannot compare these classes. □

6. Hierarchies and relations for deterministic and nondeterministic OBDDs of large width

In this section we consider OBDDs of large width. We prove some hierarchies, which differ from the hierarchies presented in the previous section (Theorem 7).

**Theorem 9.** For any integer \( n, d = d(n), 16 \leq d \leq 2^n/4 \), we have

\[
\text{OBDD}^{[d/8]-1} \subset \subset \text{OBDD}^d \quad \text{and} \quad \text{NOBDD}^{[d/8]-1} \subset \subset \text{NOBDD}^d.
\]

**Proof:** Each of inclusions \( \text{OBDD}^{[d/8]-1} \subset \subset \text{OBDD}^d \) and \( \text{NOBDD}^{[d/8]-1} \subset \subset \text{NOBDD}^d \) is obvious.

We define a Boolean function \( \text{EQS}_n^k \) as a modification of the Boolean function \( \text{Shuffled Equality} ([5, 1]) \). The proofs of the inequalities are based on the complexity properties of \( \text{EQS}_n^k \).

Let \( k \) be a multiple of 4 such that \( 4 \leq k \leq 2^n/4 \). The Boolean function \( \text{EQS}_n^k \) depends only on the first \( k \) bits.

For any given input \( \nu \in \{0, 1\}^n \), we define two binary strings \( \alpha(\nu) \) and \( \beta(\nu) \) as follows. We call the odd bits of the input \( (\nu_1, \nu_3, \ldots) \) marker bits and the even bits \( (\nu_2, \nu_4, \ldots) \) value bits. For any \( i \) satisfying \( 1 \leq i \leq k/2 \), the value bit \( \nu_{2i} \) belongs to \( \alpha(\nu) \) if the corresponding marker bit \( \nu_{2i-1} \) is 0 and \( \nu_{2i} \) belongs to \( \beta(\nu) \) otherwise.

\[
\text{EQS}_n^k(\nu) = \begin{cases} 
1, & \text{if } \alpha(\nu) = \beta(\nu), \\
0, & \text{otherwise}.
\end{cases}
\]

**Lemma 8.** There is an OBDD \( P_n \) of width \( 8 \cdot 2^{k/4} - 5 \) that computes the Boolean function \( \text{EQS}_n^k \).

**Proof.** We construct an OBDD \( P_n \) for the function \( \text{EQS}_n^k \). \( P_n \) uses the natural order of reading variables. Let \( \nu \in \{0, 1\}^n \) be the input. The main idea is to remember the bits from \( \alpha(\nu) \) and \( \beta(\nu) \) that have not been compared yet. Suppose that at the \( 2i \)-th level \( P_n \) has already read \( j \) bits of \( \alpha(\nu) \) and \( l \) bits of \( \beta(\nu) \). Each node of the level is associated with the value of the string \( c = (c_1, \ldots, c_r) \) of bits which are not compared yet. For example, if \( j > l \), then \( c = (\alpha_{i+1}(\nu), \ldots, \alpha_j(\nu)) \); if \( j < l \), then \( c = (\beta_{j+1}(\nu), \ldots, \beta_l(\nu)) \); if \( j = l \), then \( c \) is the empty string. Note that \( c \) always contains the bits either from \( \alpha(\nu) \) or from \( \beta(\nu) \) and never both. If some nodes \( c \) contains the bits from \( \alpha(\nu) \) and \( P_n \) reads a bit from \( \alpha(\nu) \), then it adds this bit to \( c \). Otherwise this bit belongs to \( \beta(\nu) \) and \( P_n \) compares it with the first bit of \( c \). If these bits are equal, then \( P_n \) removes the first bit from \( c \) and rejects the input otherwise.

More formally, let us consider the \( 2i \)-th level. It contains four groups of nodes. First of them are the nodes associated with such \( c \) that contain the bits from \( \alpha(\nu) \). The second group are the nodes associated with such \( c \) that contain the bits from \( \beta(\nu) \). The third group contains the only one “equals” node for empty \( c \) and the fourth group contains the only one “rejecting” node.

Let \( \nu_c \) be a node from the first group and \( c = (c_1, \ldots, c_r) \) be a string associated with \( \nu_c \) (\( c \) contains the bits from \( \alpha(\nu) \)). At the \( 2i \)-th level, \( P_n \) reads a marker bit \( \nu_{2i+1} \). If \( \nu_{2i+1} = 0 \), then the next value bit \( \nu_{2i+2} \) belongs to \( \alpha(\nu) \) and \( P_n \) stores bit \( \nu_{2i+2} \) at the \( (2i+1) \)-th level and goes to the node corresponding to \( c' = (c_1, \ldots, c_r, \nu_{2i+2}) \). Otherwise, the next value bit \( \nu_{2i+2} \) belongs to \( \beta(\nu) \) and \( P_n \) compares it with
If these bits are equal, then \( P_n \) goes to the node corresponding to \( c'' = (c_2, \ldots, c_r) \) or to the “equals” node if \( r = 1 \). If the bits are different, then \( P_n \) goes to the “rejecting” node. If \( c \) is empty, then \( P_n \) goes to the node of the first (the second) group associated to \( c = (\nu_{2i+2}) \), which contains bits from \( \alpha (\beta) \). If the length of \( c \) is greater than \( k/4 \), then \( P_n \) goes to the “rejecting” node.

For the second group of nodes, \( P_n \) works in the same way but the string \( c \) stores bits from \( \beta (\nu) \). \( P_n \) gives the answer 1 if it reaches the “equals” node at the last level.

Now we compute the width of \( P_n \). At the \( 2i \)-th level, the first two groups of nodes contain nodes for each possible value of \( c \) both for \( \alpha (\nu) \) and for \( \beta (\nu) \). The third and the forth groups contain the “equal” node and the “rejecting” node respectively. The width of the \( 2i \)-th level is

\[
d = (2 + 4 + \cdots + 2^{k/4}) + (2 + 4 + \cdots + 2^{k/4}) + 2 = (2^{k/4+1} - 2) + (2^{k/4+1} - 2) + 2 = 4 \cdot 2^{k/4} - 2.
\]

The \((2i+1)-th\) level has twice more nodes for the first three groups since \( P_n \) has to remember the value of marker bit, which indicates if the next bit belongs to \( \alpha (\nu) \) or to \( \beta (\nu) \). Therefore the width of the \( 2i + 1 \)-th level is \( 8 \cdot 2^{k/4} - 5 \) and we conclude that the width of \( P_n \) is \( 8 \cdot 2^{k/4} - 5 \).

Note that OBDD is a particular case of NOBDD and the same result holds for NOBDDs. \( \square \)

**Lemma 9.** There are infinitely many \( n \) such that any OBDD and NOBDD \( P_n \), computing \( \text{EQS}_n^k \), has width at least \( 2^{k/4} \).

**Proof.** Let \( P_n \) be an OBDD that computes \( \text{EQS}_n^k \) and let \( P_n \) reads the variables in the order \( \pi = (j_1, \ldots, j_n) \). Let on the \( l \)-th level \( P_n \) has already read exactly \( k/4 \) value bits from \( X_K = \{x_1, \ldots, x_k\} \).

We consider a partition of the set of variables \( \{x_{j_1}, \ldots, x_{j_l}\} \), \( \{x_{j_{l+1}}, \ldots, x_{j_n}\} \) = \( (X_A, X_B) \) and an input \( \nu = (\sigma, \gamma) \) with respect to this partition. Denote by \( X_V \) a set of all value bits from \( X_A \cap X_K \) and by

\[
X_M = (X_A \cap X_K) \setminus X_V
\]

a set of all marker bits from \( X_A \cap X_K \).

Let the set \( \Sigma = \{\sigma \in \{0, 1\}^l \mid \text{marker bits for variables from } X_M \text{ fixed such that value bits for variables from } X_V \text{ belong to } \alpha (\nu) \text{ and other value bits from } X_K \setminus X_V \text{ belong to } \beta (\nu)\} \). Note that \(|\Sigma| = 2^{k/4}\).

In the same way as in Lemma 5 we can show that all \( \sigma \in \Sigma \) reach different nodes on the \( l \)-th level, and therefore the width of the \( l \)-th level is at least \( 2^{k/4} \).

By Lemma 8, we have that Boolean function \( \text{EQS}_n^{4\lceil \log (d+5) \rceil - 12} \in \text{OBDD}^d \) and \( \text{EQS}_n^{4\lceil \log (d+5) \rceil - 12} \in \text{NOBDD}^d \). By Lemma 9, we have that the function \( \text{EQS}_n^{4\lceil \log (d+5) \rceil - 12} \notin \text{OBDD}^{d/8 - 1} \) and \( \text{EQS}_n^{4\lceil \log (d+5) \rceil - 12} \notin \text{NOBDD}^{d/8 - 1} \). Therefore we have \( \text{OBDD}^{d/8 - 1} \neq \text{OBDD}^d \) and \( \text{NOBDD}^{d/8 - 1} \neq \text{NOBDD}^d \). These inequalities prove Statements (3) and (4).

In the following theorem, we present a relationship between the deterministic and nondeterministic models.

**Theorem 10.** For any integer \( n \), \( d = d(n) \), and \( d' = d'(n) \), satisfying \( d \leq 2^n/4 \) and \( O(\log^4(d + 1) \log \log (d + 1)) < d' < d/8 - 1 \), we have

\[
\text{NOBDD}^{\log(d)} \subsetneq \text{OBDD}^d \quad \text{and} \quad \text{OBDD}^d \text{ and } \text{NOBDD}^{d'} \text{ are not comparable.}
\]

**Proof.** We start with (5). Using the same reasoning as in the proof of Theorem 3 we can show that \( \text{NOBDD}^{\log(d)} \subsetneq \text{OBDD}^d \). By Lemma 9, we have \( \text{EQS}_n^{4\lceil \log(d+5) \rceil - 12} \notin \text{NOBDD}^{\log(d)} \). This means \( \text{OBDD}^d \neq \text{NOBDD}^{\log(d)} \).

Now we continue with (6). We use the complexity properties of the Boolean function \( \text{NotEQS}_n^k \) – the negation of \( \text{EQS}_n^k \).

**Lemma 10.** There are infinitely many \( n \) such that any OBDD \( P_n \), computing \( \text{NotEQS}_n^k \), has width at least \( 2^{k/4} \).

**Proof.** The proof is similar to the proof of Lemma 9. \( \square \)
Lemma 11. There is a NOBDD $P_n$ of width $d$ that computes the Boolean function NotEQS$^k_n$, where \( d \leq O(k^4 \log k) \).

Proof. We construct an NOBDD $P_n$ for the function NotEQS$^k_n$. $P_n$ reads variables in the natural order \((1, \ldots, n)\). We use the fingerprinting method given in [11]. We will use the same notation as given in the proof of Lemma 8. Let $p_1, \ldots, p_z$ be the first $z$ prime numbers satisfying
\[
p_1p_2 \cdots p_z > 2^{k/4},
\]
where $z$ is the minimum value. Note that $r \leq k/4$ since $p_z \geq 2$.

The NOBDD $P_n$ consists of $z$ parts, each of the parts corresponds to one of $p$ from \( \{p_1, \ldots, p_z\} \). For an input $\nu \in \{0, 1\}^n$ we denote $\nu^i = (\nu_1, \ldots, \nu_i)$, where $1 \leq i \leq k$.

At the first step of a computation $P_n$ nondeterministically picks a $p$. Then, this branch computes $\nu^i = bin(\alpha(\nu)) - bin(\beta(\nu)) \mod p$. Here $bin(\alpha(\nu))$ is the binary representation of $\alpha(\nu)$.

For computing $\nu^i$ at the $i$-th step we should know three numbers: the value of $\nu^i = bin(\alpha(\nu^i)) - bin(\beta(\nu^i)) \mod p_i$, the length of $\alpha(\nu^i)$, and the length of $\beta(\nu^i)$. Note that $\nu^i \in \{0, \ldots, p - 1\}$ and $|\alpha(\nu^i)|, |\beta(\nu^i)| \leq k/2$. If at the last step $\nu^i$ is not zero or $|\alpha(\nu^i)| \neq |\beta(\nu^i)|$, then this part answers 1 and 0 otherwise. We need $p \cdot k^2/4$ nodes for each possible value of the triple \( (\nu^i, |\alpha(\nu^i)|, |\beta(\nu^i)|) \) and two times more nodes in order to check the value of odd bits for knowing whether the following bit belongs to $\alpha(\nu)$ or to $\beta(\nu)$. This means that the width of this branch is $p \cdot k^2/2$.

By the Chinese Remainder Theorem [15], if $bin(\alpha(\nu)) \neq bin(\beta(\nu))$, then there is at least one $p_i$ such that $bin(\alpha(\nu)) \neq bin(\beta(\nu)) \mod p_i$. That is, at least one branch gives the answer 1. If the lengths are different, then each branch gives the answer 1. But if $\alpha(\nu) = \beta(\nu)$, then for all $p_i$ we have $bin(\alpha(\nu)) = bin(\beta(\nu)) \mod p_i$ and $|\alpha(\nu)| = |\beta(\nu)|$. That is, each branch gives the answer 0. Hence $P_n$ computes NotEQS$^k_n$ correctly.

By the Prime Number Theorem [10], we know that $p_z = O(z \ln z)$ and therefore $p_i \leq O(z \ln z)$ for $1 \leq i \leq z$ since $p_i \leq p_z$. This means that the width of the $i$-th part is $p_i \cdot k^2/2 \leq O(z \cdot k^2 \ln z)$. Since $z \leq k/4$, the width of $P_n$ is $d \leq z \cdot O(z \cdot k^2 \ln z) = O(k^4 \log k)$.

Recall that $O(\log^4(d+1) \log \log(d+1)) \leq d' \leq \lfloor d/8 \rfloor - 1$ and, by Lemma 9 and Lemma 8, we have $\text{EQS}_{4[\log(d+6)]}^{[\log(d+6)]-12} \in \text{OBDD}^d$ and $\text{EQS}_{4[\log(d+5)]}^{[\log(d+5)]-12} \notin \text{OBDD}^d$; by Lemma 10, we have $\text{NotEQS}_{4[\log(d)]+4}^{[\log(d)]+4} \in \text{OBDD}^{d'}$; and, by Lemma 11, we have $\text{NotEQS}_{4[\log(d)]+4}^{[\log(d)]+4} \notin \text{OBDD}^d$. Therefore we cannot compare these classes. 

ACKNOWLEDGMENTS

F. Ablayev was partially supported by RFBR Grant 14-01-06036, A. Gainutdinova and K. Khadiev were partially supported by RFBR Grant 14-07-00557 and 15-37-21160, A. Yakaryılmaz did this work in part while visiting the Kazan Federal University in 2013 and he was partially supported by CAPES with grant 88881.030338/2013-01, ERC Advanced Grant MQC, and FP7 FET project QALGO.

We thank the anonymous reviewers for their helpful comments.

The work is performed according to the Russian Government Program of Competitive Growth of Kazan Federal University.

REFERENCES

1. F. Ablayev, “Randomization and nondeterminism are incomparable for ordered read-once branching programs,” Electronic Colloquium on Computational Complexity 4 (21), (1997).
2. F. Ablayev and A. Gainutdinova, “Complexity of quantum uniform and nonuniform automata,” Developments in Language Theory, LNCS 3572, 78–87 (2005).
3. F. Ablayev, A. Gainutdinova, and M. Karpinski, “On computational power of quantum branching programs,” FCT, LNCS 2138, 59–70 (2001).
4. F. M. Ablayev, A. Gainutdinova, and M. Karpinski, Cristopher Moore, and Chris Pollett, “On the computational power of probabilistic and quantum branching programs,” Information Computation 203 (2), 145–162 (2005).
5. F. M. Ablayev and M. Karpinski, “On the power of randomized branching programs,” ICALP, LNCS 1099, 348–356 (1996).
6. A. Ambainis and R. Freivalds, in Proceedings of 39th Annual Symposium on Foundations of Computer Science, 1998 (IEEE Computer Society, 1998), pp. 332–341.
7. A. Ambainis and A. Yakaryilmaz, “Superiority of exact quantum automata for promise problems,” Information Processing Letters 112 (7), 289–291 (2012).
8. A. Ambainis and A. Yakaryilmaz, Automata and Quantum Computing, arXiv:1507.01988 (2015).
9. A. Bertoni and M. Carpentieri, “Analogies and differences between quantum and stochastic automata,” Theoretical Computer Science 262 (1–2), 69–81 (2001).
10. H. G. Diamond, “Elementary methods in the study of the distribution of prime numbers,” Bulletin of The American Mathematical Society 7, 553–589 (1982).
11. R. Freivalds, “Fast probabilistic algorithms,” Mathematical Foundations of Computer Science, LNCS 74, 57–69 (1979).
12. V. Gelfert and A. Yakaryilmaz, “Classical automata on promise problems,” Discrete Mathematics & Theoretical Computer Science 17 (2), 126–137 (2015).
13. J. Hromkovic and M. Sauerhoff, “Tradeoffs between nondeterminism and complexity for communication protocols and branching programs,” STACS, LNCS 1770, 145–156 (2000).
14. J. Hromkovic and M. Sauerhoff, “The power of nondeterminism and randomness for oblivious branching programs,” Theory of Computing Systems 36 (2), 159–182 (2003).
15. K. Ireland and M. Rosen, A Classical Introduction to Modern Number Theory, 2nd ed. Vol. 84 (Graduate Texts in Mathematics, 1990).
16. J. G. Kemeny and J. L. Snell, Finite Markov Chains (Van Nostrand Company, INC, 1960).
17. K. Khadiev, “Width hierarchy for k-OBDD of small width,” Lobachevskii Journal of Mathematics 36 (2), 178–183 (2015).
18. A. Kondacs and J. Watrous, “On the power of quantum finite state automata,” FOCS’97 Proceedings of the 38th Annual Symposium on Foundations of Computer Science (IEEE Computer Society, Washington, 1997), pp. 66–75.
19. M. Nakanishi, K. Hamaguchi, and T. Kashiwabara, “Ordered quantum branching programs are more powerful than ordered probabilistic branching programs under a bounded-width restriction,” COCOON, LNCS 1858, 467–476 (2000).
20. A. Paz, Introduction to Probabilistic Automata (Academic Press, New York, 1971).
21. J. Rashid and A. Yakaryilmaz, “Implications of quantum automata for contextuality,” Implementation and Application of Automata, LNCS 8587, 318–331 (2014).
22. M. Sauerhoff and D. Sieling, “Quantum branching programs and space-bounded nonuniform quantum complexity,” Theoretical Computer Science 334 (1–3), 177–225 (2005).
23. J. Watrous, “On the complexity of simulating space-bounded quantum computations,” Computational Complexity 12 (1–2), 48–84 (2003).
24. J. Watrous, “Quantum computational complexity,” in Encyclopedia of Complexity and Systems Science, arXiv:0804.3401, 7174–7201 (2009).
25. I. Wegener, Branching Programs and Binary Decision Diagrams (SIAM, 2000).
26. A. Yakaryilmaz and A. C. Cem Say, “Languages recognized by nondeterministic quantum finite automata,” Quantum Information and Computation 10 (9–10), 747–770 (2010).
27. A. Yakaryilmaz and A. C. Cem Say, “Unbounded-error quantum computation with small space bounds,” Information and Computation 279 (6), 873–892 (2011).