Error-Free Affine, Unitary, and Probabilistic OBDDs

Rishat Ibrahimov, Kamil Khadiev, Krišjānis Prūsis, Abuzer Yakaryilmaz

▶ **To cite this version:**

Rishat Ibrahimov, Kamil Khadiev, Krišjānis Prūsis, Abuzer Yakaryilmaz. Error-Free Affine, Unitary, and Probabilistic OBDDs. 20th International Conference on Descriptional Complexity of Formal Systems (DCFS), Jul 2018, Halifax, NS, Canada. pp.175-187, 10.1007/978-3-319-94631-3_15 . hal-01905639

HAL Id: hal-01905639
https://inria.hal.science/hal-01905639

Submitted on 26 Oct 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Error-Free Affine, Unitary, and Probabilistic OBDDs[⋆]

Rishat Ibrahimov[1,3], Kamil Khadiev[1,2,3], Krišjānis Prūsis[2], and Abuzer Yakaryılmaz[2]

[1] Kazan Federal University, Kazan, Russia
[2] University of Latvia, Center for Quantum Computer Science, Rīga, Latvia
[3] Smart Quantum Technologies Ltd., Kazan, Russia
`rishat.ibrahimov@yandex.ru`, `kamilhadi@gmail.com`, `krisjanis.prusis@lu.lv`, `abuzer@lu.lv`

Abstract. We introduce the affine OBDD model and show that zero-error affine OBDDs can be exponentially narrower than bounded-error unitary and probabilistic OBDDs on certain problems. Moreover, we show that Las Vegas unitary and probabilistic OBDDs can be quadratically narrower than deterministic OBDDs. We also obtain the same results for the automata versions of these models.

Keywords: OBDDs, affine models, quantum and probabilistic computation, zero-error, Las Vegas computation, succinctness

1 Introduction

Using negative transition values (allowing interference between states and configurations) is unarguably the main distinguishing property of quantum computational models. In order to define a quantum-like classical system, one can also introduce "negative probabilities" but the system is no longer linear. In this direction, affine systems were introduced as an almost linear[4] generalization of probabilistic systems that can use negative transitions [10], and, due to their simplicity, affine finite automata (AfAs) have been examined in a series of papers by comparing them with classical and quantum finite automata [10, 27, 9, 14, 26]. Both bounded- and unbounded-error AfAs have been shown to be more powerful than their probabilistic and quantum counterparts and they are equivalent to quantum models in nondeterministic acceptance mode [10]. AfAs can also be very succinct on languages and promise problems [27, 9]. In this paper, we compare the computational power of error-free affine OBDDs with their probabilistic and unitary counterparts on solving total functions and recognizing languages.

[⋆] Part of the research work was done while Ibrahimov was visiting University of Latvia in February 2017.

[4] It evolves linearly but a non-linear operator is applied when we retrieve information from the state vector.

Ordered Binary Decision Diagrams (OBDD), also known as oblivious read once branching programs [28], are an important restriction of branching programs. It is a good model of data streaming algorithms. Since the length of an OBDD is fixed (linear), the main complexity measure is "width", analogous to the number of states for automata models (see [28]). OBDDs can also be seen as a variant of nonuniform automata that allow accessing the input in a predetermined order and using possibly different sets of instructions in each step (e.g. [1]). It is known that [4, 6, 3, 23, 18] the gap between the width of the following OBDD models can be at most exponential on total functions: deterministic and bounded-error probabilistic, deterministic and bounded-error unitary, and bounded-error probabilistic and bounded-error unitary. Each exponential gap has also been shown to be tight by presenting witness functions. On partial functions, on the other hand, the gap (width) between deterministic and exact unitary OBDDs was shown not to be bounded [5, 12, 2].

In this paper, we introduce affine OBDDs and then compare error-free (zero-error or Las Vegas) affine, unitary, probabilistic, and deterministic OBDD models. Zero-error probabilistic OBDDs are identical to deterministic OBDDs. Zero-error unitary OBDDs cannot be narrower than deterministic OBDDs. On the other hand, zero-error affine OBDDs can be exponentially narrower than not only deterministic OBDDs but also bounded-error probabilistic and unitary OBDDs. With success probability $\frac{1}{2}$, Las Vegas probabilistic and unitary OBDDs can be quadratically narrower than deterministic OBDDs. We give an example function that achieves this bound up to a logarithmic factor. Finally, we examine the automata counterpart of these models and obtain similar results.

The preliminaries and definitions are given in the next section. The generic lower bounds are given in Section 3. The results on zero-error affine OBDD are given in Section 4 and the results on Las Vegas OBDDs are given in Section 5. We close the paper with automata results in Section 6. The extended version of the paper can be accessible on arXiv (*1703.07184*).

2 Preliminaries

We assume the reader familiar with the basics of branching programs. An Ordered Binary Decision Diagram (OBDD) can be defined as a non-uniform automaton that can read the input symbols in a predetermined order (see [28]).

An OBDD P reads the variables in a specific order $\pi = (j_1, \ldots, j_n)$, called the order of P, and it can trace its computation on a finite set of states $S = \{s_1, \ldots, s_m\}$ such that (i) m is the width of OBDD, (ii) the initial state is the initial node, and (iii) the accepting states are the accepting sink nodes. Thus, each node in any level can be easily associated with a state in S. An OBDD can also have a different transition function at each level.

A probabilistic OBDD (POBDD) P_n of width m is formally defined as a 5-tuple: $P_n = (S, T, v_0, S_a, \pi)$, where $S = \{s_1, \ldots, s_m\}$ is the set of states corresponding to at most one node in each level[5], v_0 is the initial probabilistic state, which is a stochastic column vector of size m, S_a is the set of accepting states corresponding to the accepting sink nodes in the last level, π is a permutation of $\{1,\ldots,n\}$ defining the order of the input variables, and, $T = \{T_i^0, T_i^1 \mid 1 \leq i \leq n\}$ is the set of (left) stochastic transition matrices.

Let $x \in \{0,1\}^n$ be the given input. At the beginning of the computation P_n is in v_0. Then the input bits are read in the order $\pi(1), \pi(2), \ldots, \pi(n)$ and the corresponding stochastic operators are applied. That is, at the j-th step, we have $v_j = T_j^{x_{\pi(j)}} v_{j-1}$, where v_{j-1} and v_j are the probabilistic states before and after the transition, respectively, $\pi(j)$ represents the input bit read in this step, $x_{\pi(j)}$ is the value of this bit, and $T_j^{x_{\pi(j)}}$ is the stochastic operator applied in this step. We represent the final state as $v_f = v_n$. The input x is accepted (the value of 1 is returned) with probability $f_{P_n}(x) = \sum_{s_i \in S_a} v_f[i]$.

If all stochastic elements of transition matrices of a POBDD are composed of only 0s and 1s, then it is a deterministic OBDD.

Quantum OBDDs (QOBDD) using superoperators are non-trivial generalizations of POBDDs [2]. In this paper, we use the most restricted version of quantum OBDDs called unitary OBDDs (UOBDDs) [13]. Note that UOBDDs and POBDDs are incomparable [23]. (We refer the reader to [25] for a basic introduction to the basics of quantum computation.)

A UOBDD with width m, say M_n, is a 5-tuple $M_n = (Q, T, |v_0\rangle, Q_a, \pi)$, where $Q = \{q_1, \ldots, q_m\}$ is the set of states, $|v_0\rangle$ is the initial quantum state, Q_a is the set of accepting states, π is a permutation of $\{1,\ldots,n\}$ defining the order of the variables, and $T = \{T_i^0, T_i^1 \mid 1 \leq i \leq n\}$ is the set of unitary transition function matrices such that at the i-th step T_i^0 (T_i^1) is applied if the corresponding input bit is 0 (1).

Let $x \in \{0,1\}^n$ be the given input. At the beginning of the computation M_n is in $|v_0\rangle$. Then the input bits are read in the order $\pi(1), \pi(2), \ldots, \pi(n)$ and the corresponding unitary operators are applied: $|v_j\rangle = T_j^{x_{\pi(j)}} |v_{j-1}\rangle$. This represents the transition at the j-th step, where $|v_{j-1}\rangle$ and $|v_j\rangle$ are the quantum states before and after the transition, respectively, $\pi(j)$ represents the input bit read in this step, $x_{\pi(j)}$ is the value of this bit, and $T_j^{x_{\pi(j)}}$ is the unitary operator applied in this step. We represent the final state as $|v_f\rangle = |v_n\rangle$. At the end of the computation, the final state is measured in the computational basis and the input is accepted if the observed state is an accepting one. Thus, the input x is accepted with probability $f_{M_n}(x) = \sum_{q_i \in Q_a} |\langle q_i | v_f \rangle|^2$, where $|q_i\rangle$ represents the basis state corresponding to state q_i and $\langle q_i | v_f \rangle$ gives the amplitude of q_i in the final state.

An m-state affine system [10] can be represented by the space $\mathbb{R}^m$, where $\mathbb{R}$ is the set of real numbers. The set of (classical) states is denoted $E = \{e_1, \ldots, e_m\}$. Any affine state (similar to a probabilistic state) is represented as a column vector $v = (\alpha_1, \alpha_2, \ldots, \alpha_m)^T \in \mathbb{R}^m$ such that $\sum_{i=1}^m \alpha_i = 1$. Each e_i also corresponds to a standard basis vector of $\mathbb{R}^m$ having value 1 in its i-th entry. An affine operator

[5] Suppose we have $w_i \leq m$ nodes on a level i; then the node u_j of this level corresponds to the state s_j, for $j \in \{1, \ldots, w_i\}$

is an $m \times m$ matrix, each column of which is an affine state, where the (j,i)-th entry represents the transition from state e_i to state e_j. If we apply an affine operator A to the affine state v, we obtain the new affine state $v' = Av$. To get information from the affine state, a non-linear operator called weighting is applied, which returns any state with probability equal to the weight of the corresponding vector element in the l_1 norm of the affine state. If it is applied to v, the state e_i is observed with probability $\frac{|\alpha_i|}{\sum_{j=1}^{n} |\alpha_j|} = \frac{|\alpha_i|}{|v|}$, where $|v|$ is the l_1 norm of v.

Here we define affine OBDDs (AfOBDDs) as a model with both classical and affine states, which is similar to the quantum model having quantum and classical states [7]. This addition does not change the computational power of the model, but helps in algorithm construction.

An AfOBDD, say M_n, having m_1 classical and m_2 affine states is an 9-tuple $M_n = (S, E, \delta, T, s_I, v_0, S_a, E_a, \pi)$, where $S = \{s_1, \dots, s_{m_1}\}$ is the set of classical states, $s_I \in S$ is the initial classical state, $S_a \subseteq S$ is the set of classical accepting states, $E = \{e_1, \dots, e_{m_2}\}$ is the set of affine states, $v_0 \in \mathbb{R}^{m_2}$ is the initial affine state, $E_a \subseteq E$ is the set of affine accepting states, π is a permutation of $\{1, \dots, n\}$ defining the order of the variables, $\delta = \{\delta_i : S \times \{0,1\} \to S \mid 1 \leq i \leq n\}$ is the classical transition function such that at the i-th step the classical state is set to $\delta_i(s, x_{\pi(i)})$ when in state $s \in S$ and corresponding input bit is $x_{\pi(i)}$, and, $T = \{T_i^{s,0}, T_i^{s,1} \mid s \in S \text{ and } 1 \leq i \leq n\}$ is the set of affine transition matrices such that at the i-th step $T_i^{s,0}$ is applied if the corresponding input bit is 0 (or $T_i^{s,1}$ if it is 1) and the current classical state is s. The width of M_n is equal to $m_1 \cdot m_2$.

Let $x \in \{0,1\}^n$ be the given input. At the beginning of the computation M_n is (s_I, v_0). Then the input bits are read in the order $\pi(1), \pi(2), \dots, \pi(n)$. In each step, depending on the current input bit and classical state, the affine state is updated and then the classical state is updated based on the current input bit. Let (s, v_{j-1}) be the classical-affine state pair at the beginning of the j-th step. Then the new affine state is updated as $v_j = T_j^{s, x_{\pi(j)}} v_{j-1}$. After that the new classical state is updated by $\delta_j(s, x_{\pi(j)})$. At the end of the computation we have (s_F, v_f). If $s_F \notin S_a$, then the input is rejected. Otherwise, the weighting operator is applied to v_f, and the input is accepted with probability $f_{M_n}(x) = \sum_{e_i \in E_a} \frac{|v_f[i]|}{|v_f|}$. Note that if we use only non-negative numbers for an AfOBDD, then we obtain a POBDD.

Any OBDD with $\pi = (1, \dots, n)$ is called an id-OBDD. If we use the same transitions at each level of an id-OBDD, then we obtain a finite automaton (FA). A FA can also read an additional symbol after reading the whole input called the right end-marker (\$) for the post-processing. We abbreviate the FA versions of OBDD, POBDD, QOBDD, UOBDD, and AfOBDD as DFA, PFA, QFA, UFA, and AfA, respectively. Remark that UFAs are also known as Measure-Once or Moore-Crutchfield quantum finite automata [21, 8].

A Las Vegas automaton can make three decisions: "accept", "reject", and "don't know". Therefore, its set of states is split into three disjoint sets, the set of accepting, rejecting, and neutral states in which the aforementioned decisions are given, respectively. To be a well-defined Las Vegas algorithm, each member (resp., non-member) is rejected (resp., member) with zero probability, i.e. the algorithm never makes false classification, and the correct decision is given with probability at least $p > 0$.

We assume the reader is familiar with the basic terminology of computing functions and recognizing languages. Here we revise some necessary notations.

A function $f : \{0,1\}^n \to \{1,0\}$ is computed by a bounded-error machine if each member of $f^{-1}(1)$ is accepted with probability at least $1 - \varepsilon$ and each member of $f^{-1}(0)$ is accepted with probability no more than ε for some non-negative $\varepsilon < \frac{1}{2}$. If $\varepsilon = 0$, then the computation (and the machine) is called zero-error or exact.

In the case of FAs, languages are considered instead of functions and the term "language recognition" is used instead of "computing a function".

A Las Vegas FA can recognize a language L with success probability $p < 1$ (with error bound $1 - p$) if each member is accepted and each non-member is rejected with probability at least p.

For a given language L, $\mathsf{DFA(L)}$, $\mathsf{LV}_\varepsilon\mathsf{(L)}$, and $\mathsf{ULV}_\varepsilon\mathsf{(L)}$ denote the number of states of a minimal DFA, a minimal LV-PFA and a minimal LV-QFA recognizing language L, respectively, where the error bound is ε for the probabilistic and quantum models. For a given Boolean function f, $\mathsf{OBDD(f)}$, $\mathsf{LV{-}OBDD}_\varepsilon\mathsf{(f)}$, and $\mathsf{ULV{-}OBDD}_\varepsilon\mathsf{(L)}$ denote the widths of a narrowest OBDD, a narrowest LV-POBDD and a narrowest LV-UOBDDs computing f, respectively, where the error bound is ε for the probabilistic and quantum models. Remark that in the case of zero-error we set $\varepsilon = 0$.

3 Lower Bounds

Let $X = \{X_1, \dots, X_n\}$ be the set of variables. Let $\theta = (X_A, X_B)$ be a partition of the set X into two parts X_A and $X_B = X \backslash X_A$. Let $f|_\rho$ be a "subfunction" of f, where ρ is a mapping $\rho : X_A \to \{0,1\}^{|X_A|}$. The function $f|_\rho(X_B)$ is obtained from f by fixing each variable from X_A to its value under ρ. The concept of subfunction can be seen as a counterpart of Myhill-Neroda equivalence classes. Let $N^\theta(f)$ be the number of different subfunctions with respect to the partition θ. Let $\Pi(n)$ be the set of all permutations of $\{1, \dots, n\}$. Let $\theta(\pi, u) = (X_A, X_B) = (\{X_{j_1}, \dots, X_{j_u}\}, \{X_{j_{u+1}}, \dots, X_{j_n}\})$, for the permutation $\pi = (j_1, \dots, j_n) \in \Pi(n)$, $1 < u < n$. We denote $\Theta(\pi) = \{\theta(\pi, u) : 1 < u < n\}$. Let $N^\pi(f) = \max_{\theta \in \Theta(\pi)} N^\theta(f)$, $N(f) = \min_{\pi \in \Pi(n)} N^\pi(f)$.

Based on techniques from communication complexity theory, it has been shown that exact quantum and probabilistic protocols have at least the same complexity as deterministic ones [19, 15]. The followings are also known:

Fact 1 *[11, 19, 16, 15] For any regular language L and error bound $\varepsilon < 1$, we have the following lower bounds for PFAs and QFAs:* $(\mathsf{DFA(L)})^{1-\varepsilon} \leq \mathsf{LV}_\varepsilon\mathsf{(L)}$ *and* $(\mathsf{DFA(L)})^{1-\varepsilon} \leq \mathsf{ULV}_\varepsilon\mathsf{(L)}$.

Fact 2 *[23] For any Boolean function f over $X = (X_1, \ldots, X_n)$ and error bound $\varepsilon < 1$:* $(\mathsf{OBDD(f)})^{1-\varepsilon} \leq \mathsf{ULV{-}OBDD}_{\varepsilon}(\mathsf{f})$.

These results are followed from certiain facts from communication complexity. We briefly remind the notion of communication complexity (see [20]). Let $h : \{0,1\}^q \times \{0,1\}^m \to \{0,1\}$ be a Boolean function. We have two players called Alice and Bob, who compute $h(x, y)$. The function h is known by both of them, but Alice can see only x and Bob can see only y. First, Alice sends message(s) to Bob and then Bob returns the answer. Alice's aim is trying to minimize the number of sending bits. The protocol is called Las Vegas if Alice chooses her messages randomly, and then Bob returns the correct answer with probability at least $1 - \varepsilon$ and gives-up (has error) with the remaining probability.

The communication complexity of function $h(x, y)$ equals to c if and only if the minimum number of bits sent during the communication is equal to c. It is denoted as $C(h)$ if the protocol is deterministic and $LV{-}C_{\varepsilon}(h)$ if the protocol is Las Vegas and the probability of giving-up is bounded by ε. The relation between these two complexity measures is presented in the following fact:

Fact 3 *[11, 19, 16, 15] For any Boolean function h and an error bound $\varepsilon < 1$, we have the following lower bound:* $(1 - \varepsilon)C(h) \leq LV{-}C_{\varepsilon}(h)$.

Facts 1 and 2 follows from Fact 3 and a simulation of automata and OBDD by communication protocols (see [16, 17]).

We can easily extend the result from Fact 2 for the probabilistic OBDD model as well. For this purpose, we also use the following fact.

Theorem 1. *For any Boolean function f over $X = \{X_1, \ldots, X_n\}$ and error bound $\varepsilon < 1$:* $(\mathsf{OBDD(f)})^{1-\varepsilon} \leq \mathsf{LV{-}OBDD}_{\varepsilon}(\mathsf{f})$.

Proof. Let $d = \mathsf{OBDD(f)}$. Due to [28], we have $N(f) = d$. Assume that there is a Las Vegas OBDD P of width w, i.e. $w < d^{1-\varepsilon}$. Let $u = \mathtt{argmax}_t N^{\theta(\pi(P),t)}(f)$ and $\theta = \theta(\pi(P), u)$. Then P can be simulated by a Las Vegas probabilistic protocol (see [17]) with $\log_2 w < (1 - \varepsilon) \log_2 d$ bits. By the definition of the number of subfunctions, we have $N^{\theta}(f) \geq d$. Moreover, it is known that the deterministic communication complexity of a function is $\log_2(N^{\theta}(f)) = \log_2 d$. But we also have a Las Vegas communication protocol which uses $\log_2 w < (1 - \varepsilon) \log_2 d$ communication bits, which contradicts with Fact 3. $\square$

It is trivial that these results imply equivalence for exact (zero-error) computation, where $\varepsilon = 0$.

4 Zero-Error Affine OBDDs

We show that exact AfOBDDs can be exponentially narrower than classical and unitary quantum OBDD models. For this purpose we use two different functions.

The hidden weighted bit function [28] $\mathtt{HWB_n} : \{0,1\}^n \to \{0,1\}$ returns the value of x_z on the input $x = (x_1, \ldots, x_n)$ where $z = x_1 + \cdots + x_n$, taking $x_0 = 0$.

It is known [28] that any OBDD solving $\mathtt{HWB_n}$ has a minimum width of $2^{n/5}/n$. Due to Fact 2 and Theorem 1, the same bound is also valid for exact POBDDs and UOBDDs.

Theorem 2. *An exact id-AfOBDD M with n classical and n affine states can solve* $\mathtt{HWB_n}$.

Proof. The classical states are $s_0, \ldots, s_{n-1}$ where s_0 is the initial and only accepting state. The affine states are $e_0, \ldots, e_{n-1}$ where e_0 is the only accepting affine state and v_0 is e_0.

Until reading the last input bit (x_n), for each value 1, the index of the classical state is increased by 1. Meanwhile, the value of x_i is written to the value of e_i in the affine state: the affine state after the $(i-1)$-th step becomes $v_{i-1} = (\overline{1} \; x_1 \; \cdots \; x_{i-1} \; 0 \; \cdots \; 0)^T$, where $\overline{1} = 1 - \sum_{j=1}^{i-1} x_j$. If $x_i = 0$, then the identity operator is applied. If $x_i = 1$, then the following affine transformation is applied

$$\left(\begin{array}{c|ccc|c|c}
0 & -1 & \cdots & -1 & 0 & \\ \hline
0 & & & & 0 & \\
\vdots & & \mathbf{I_{(i-1)\times(i-1)}} & & \vdots & \mathbf{0} \\
0 & & & & 0 & \\ \hline
1 & 1 & \cdots & 1 & 1 & \\ \hline
 & & \mathbf{0} & & & \mathbf{I}
\end{array}\right),$$

which updates the values of e_0 and e_i as $-\sum_{j=1}^{i-1} x_j$ and 1, respectively, and does not change the other entries. Remark that the first entry can also be written as $1 - \sum_{j=1}^{i} x_j$. Thus, before reading x_n, the classical state is s_t for $t = x_1 + \cdots + x_{n-1}$, and, the affine state is $v_{n-1} = (\overline{1} \; x_1 \; x_2 \; \cdots \; x_{n-1})^T$, where $\overline{1}$ is $1 - \sum_{j=1}^{n-1} x_j$. After reading x_n, the classical state is always set to s_0 and so the decision is made based on the affine state. Each pair of s_t and x_n determine an appropriate last affine transformation that sets the final affine state to $v_f = (x_z \; 1-x_z \; 0 \; \cdots \; 0)^T$, where x_z is set to either 0 or 1. The details of each last transition are as follows: (i) if $t = 0$ and $x_n = 0$, then the corresponding last transitions sets $x_z = 0$, (ii) if $t = n - 1 > 0$, then regardless the value of x_n, the corresponding last transition sets $x_z = 1$, and (iii) in any other case, $z = t + x_n$ and the corresponding last transition sets x_z to the $(z+1)$-th entry of v_{n-1}. Based on x_z, the final decision is given: if $x_z = 0$ (resp., $x_z = 1$), then the input is accepted with probability 0 (resp., 1). □

For a positive integer n, let x, y be the inputs of size n, let $p(n)$ be the smallest prime number greater than n, and let $s_n(x) = (\sum_{i=1}^{n} i \cdot x_i) \mod p(n)$. The mixed weighted sum function [22] $\mathtt{MWS_n} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ is defined as $\mathtt{MWS_n}(x, y) = x_i \oplus y_i$ if $i = s_n(x) = s_n(y) \in \{1, \ldots, n\}$, and 0 otherwise. Any bounded-error POBDD or UOBDD solving $\mathtt{MWS_n}$ has a width of at least $2^{\Omega(n)}/n$ [24, 22]. The same bound is also valid for OBDDs, and so for exact POBDDs and UOBDDs as well.

Before presenting our affine algoroithm for $\mathtt{MWS_n}$, we consider its simpler version: the weighted sum function [22] $\mathtt{WS_n}(x) : \{0,1\}^n \rightarrow \{0,1\}$, defined as $\mathtt{WS_n}(x) = x_{s_n(x)}$ if $s_n(x) \in \{1,\ldots,n\}$, and 0 otherwise.

Theorem 3. *An exact id-AfOBDD can solve* $\mathtt{WS_n}$ *with* $p(n)$ *classical states and* n *affine states.*

Proof. We use almost the same algorithm given in the previous proof. The affine part is the same. The new classical states are $s_0,\ldots,s_{p(n)-1}$ with the same initial and single accepting state s_0. Until reading x_n, the same affine transitions are applied: $v_{n-1} = (\overline{1} \ \ x_1 \ \ x_2 \ \ \cdots \ \ x_{n-1})^T$, where $\overline{1}$ is $1 - \sum_{j=1}^{n-1} x_j$. The classical transitions are modified as follows: The classical state before reading x_i $(i < n)$, say s_j, is set to $s_{j+i\cdot x_i \bmod p(n)}$. Thus, before reading x_n, the classical state is s_t where $t = \left(\sum_{i=1}^{n-1} i \cdot x_i\right) \bmod p(n)$. At this point, the pair s_t and x_n sets the affine state to $v_f = (x_{s_n(x)} \ 1 - x_{s_n(x)} \ 0 \ \cdots \ 0)^T$ and then the classical state is set to s_0. Here $x_{s_n(x)} = 0$ if $t + n \cdot x_n \bmod p(n) \notin \{1,\ldots,n\}$. Otherwise, depending on the value of $s_n(x)$, $x_{s_n(x)}$ is set to the corresponding value from v_{n-1} or directly to x_n. □

Theorem 4. *An exact id-AfOBDD can solve* $\mathtt{MWS_n}$ *with* $p^2(n)$ *classical states and* $(n+1)$ *affine states.*

Proof. The classical states are $\{s_{i,j} \mid 0 \leq i,j < p(n)\}$ where $s_{0,0}$ is the initial state and the accepting states are $\{s_{i,i} \mid 1 \leq i \leq n\}$. While reading all bits, the values of $s_n(x)$ and $s_n(y)$ are calculated and stored as the values of the first and second index of the classical states, respectively. Let $s_{i,j}$ be the final state. It is clear that if $i \neq j$ or $i = j$ but not in $\{1,\ldots,n\}$, the input is rejected classically. In the remaining part, we assume that the final classical state is $s_{i,i}$ with $i \in \{1,\ldots,n\}$. Thus, the final decision is given based on the final affine state.

The affine states are $\{e_0,\ldots,e_n\}$ where e_0 is the single accepting state. The initial affine state is $v_0 = e_0$.

While reading the first part of the input, all x values are encoded in the affine state as $(\overline{1} \ \ (-1)^{x_1} \ \ (-1)^{x_2} \ \ \cdots \ \ (-1)^{x_n})^T$, where $\overline{1} = 1 - \sum_{i=1}^{n}(-1)^{x_i}$. Then, for each y_j $(1 \leq j \leq n)$, we multiply the j-th entry of the affine state by $(-1)^{y_j}$ (and update the first entry accordingly). Thus, the j-th entry becomes $(-1)^{x_j+y_j}$, which is equal to 1 if $x_j = y_j$ and -1 if $x_j \neq y_j$.

The last affine transformation is a composition of three affine transformations. The first transformation is the one explained above for y_n. By using the second one, the affine state is set to $((-1)^{x_i+y_i} \ 1 - (-1)^{x_i+y_i} \ 0 \ \cdots \ 0)^T$.

Here the first two entries are $(1 \ \ 0)^T$ for members and $(-1 \ \ 2)^T$ for non-members. By using the third transformation, we can add half of the second entry to the first entry and so get respectively $(1 \ \ 0)^T$ and $(0 \ \ 1)^T$. Thus the AfOBDD can separate members from non-members with zero error. □

5 Las Vegas POBDDs and UOBDDs

For OBDDs with $\varepsilon = \frac{1}{2}$, the lower bounds given in Section 3 can be at most quadratic. Up to a logarithmic factor, this quadratic gap was achieved by using the $\mathtt{SA_d}$ function in [23] for id-OBDDs. Here we give the same result for OBDDs (for any order) using the $\mathtt{SSA_n}$ function and also provide an LV-UOBDD algorithm with the same size as the LV-OBDD.

We start with the well-known *Storage Access* Boolean Function $\mathtt{SA_d}(x, y) = x_y$, where the input is split into the *storage* $x = (x_1, \ldots, x_{2^d})$ and the *address* $y = (y_1, \ldots, y_d)$.

By using the idea of "Shuffling" from [3, 6, 5, 2], we define the *Shuffled Storage Access* Boolean function $\mathtt{SSA_n} : \{0,1\}^n \to \{0,1\}$ for even n. Let $x \in \{0,1\}^n$ be an input. We form two disjoint sorted lists of even indexes of bits $I_0 = (2i_1, \ldots, 2i_m)$ and $I_1 = (2j_1, \ldots, 2j_k)$ with the following properties: (i) if $x_{2i-1} = 0$ then $2i \in I_0(x)$, (ii) if $x_{2i-1} = 1$ then $2i \in I_1(x)$, (iii) $i_r < i_{r+1}$, for $r \in \{1, \ldots, m-1\}$ and $j_r < j_{r+1}$, for $r \in \{1, \ldots, k-1\}$.

Let d be such that $2^d + d = n/2$. We can construct two binary strings by the following procedure: initially $\alpha(x) := 0^{2^d}$, then for r from 1 to m we do $\alpha(x) := \mathtt{ShiftX}(\alpha(x), x_{2i_r})$, where $\mathtt{ShiftX}((a_1, ..., a_m), b) = (a_2, ..., a_m, a_1 \oplus b)$. And initially $\beta(x) := 0^d$, then for r from 1 to k: $\beta(x) := \mathtt{ShiftX}(\beta(x), x_{2j_r})$. Then, $\mathtt{SSA_n}(x) = \mathtt{SA_d}(\alpha(x), \beta(x))$. Firstly, we provide a lower bound for OBDDs.

Theorem 5. $\mathsf{OBDD}(\mathtt{SSA_n}) \geq 2^{2^d}$*, for* $2^d + d = n/2$.

Now, we provide an upper bound for LV-OBDDs.

Theorem 6. $\mathsf{LV{-}OBDD_{0.5}}(\mathtt{SSA_n}) \leq 2^{2^d/2+d+3}$*, for* $2^d + d = n/2$.

Theorem 7. $\mathsf{ULV{-}OBDD_{0.5}}(\mathtt{SSA_n}) \leq 2^{2^d/2+d+3}$*, for* $2^d + d = n/2$.

Affine OBDDs can be exponentially narrower also for $\mathtt{SSA_n}$.

Theorem 8. *An exact AfOBDD A can solve* $\mathtt{SSA_n}$ *with* 2^{d+1} *classical states and* $2^d + 1$ *affine states, for* $2^d + d = n/2$.

Proof. The set of classical states is $\{(p, s) \mid p \in \{p_0, p_1\}$ and $s \in \{s_0, \ldots, s_{2^d-1}\}\}$ having 2^{d+1} states. The states p_j denote whether the next even bit to read is a storage or address bit. The state s_i corresponds to the current address being i. Every classical state is accepting, and so the decision is made based on the final affine state. The AfOBDD also has 2^d+1 affine states, $\{e_1, \ldots, e_{2^d+1}\}$. The initial state is e_{2^d+1}, i.e. $v_0 = (0 \ \cdots \ 0 \ 1)^T$, and the only accepting state is e_1.

During the computation, it keeps the value of the storage in the first 2^d states. Specifically, if (i) the next position to read is odd or (ii) even but an address bit (we are in a state (p_1, s_i)), then we perform the identity transformation on the affine state and change only the classical state. If we are reading a storage bit, the classical state remains unchanged and we implement the $\mathtt{ShiftX}$ operation on the storage: first, the 2^d entries are shifted to the left by one and the first entry becomes the 2^d-th entry. Then, depending on the scanned symbol, the value of the 2^d-th entry is updated:

– If the scanned symbol is 0, then, for calculating the XOR value, the 2^d-th entry is multiplied by 1, i.e., $0 \to 0$ and $1 \to 1$.
– If the scanned symbol is 1, then, for calculating the XOR value, the 2^d-th entry is multiplied by -1 and then 1 is added to this result, i.e., $0 \to 0 \to 1$ and $1 \to -1 \to 0$.

The last entry in the affine state is used to make the state vector well-formed. For example, if the state vector has $0 \leq t \leq 2^d$ 1s in its first 2^d entries, then the last entry is $1 - t$.

After reading the whole input, the first 2^d entries keep the storage. We know the address i from our classical state s_i – we move the corresponding storage value from e_i to e_1 and sum all other entries in e_2. Then, if the first entry is 1, the rest of the vector contains only 0. If the first entry is 0, the second entry is 1 and the rest of the vector contains 0. Therefore, any member (resp., non-member) of $\mathtt{SSA_n}$ is accepted by A with probability 1 (resp., 0). □

6 Las Vegas Automata and Zero-Error AfAs

Similar to OBDDs, for $\varepsilon = \frac{1}{2}$, the lower bound for finite automata (Section 3) is at most quadratic. Up to a constant, this quadratic gap is achieved by $\mathtt{END_k} = \{u1v \mid u, v \in \{0,1\}^* \text{ and } |v| = k-1\}$: $\mathsf{DFA}(\mathtt{END_k}) = 2^k$ and $\mathsf{LV}_{0.5}(\mathtt{END_k}) \leq 4{\cdot}2^{k/2}$.

Here, we propose a new language $\mathtt{MODXOR_k}$ based on which we improve the above constant for LV-PFAs and provide a LV-UFA algorithm with the same size as LV-PFAs. Then, we show that an AfA can recognize it with exponentially fewer states with zero error. The language $\mathtt{MODXOR_k}$ for $k > 0$ is formed by the strings $\{0,1\}^{<2k}x_1\{0,1\}^{2k-1}x_2\{0,1\}^{2k-1}\cdots x_m\{0,1\}^{2k-1}$ where $m > 0$, each $x_i \in \{0,1\}$ for $1 \leq i \leq m$, and $\bigoplus_{i=0}^{m} x_i = 1$, taking $x_0 = 0$. First, we give a lower bound for DFAs.

Theorem 9. $\mathsf{DFA}(\mathtt{MODXOR_k}) \geq 2^{2k}$ *for each* $k > 0$.

Theorem 10. $\mathsf{LV}_{0.5}(\mathtt{MODXOR_k}) \leq 2 \cdot 2^k$ *for any* $k > 0$.

Theorem 11. $\mathsf{ULV}_{0.5}(\mathtt{MODXOR_k}) \leq 2 \cdot 2^k$ *for any* $k > 0$.

Similarly to OBDDs, exact AfAs can also be exponentially more efficient than their classical and quantum counterparts.

Theorem 12. *The language* $\mathtt{MODXOR_k}$ *for* $k > 0$ *can be recognized by a* $(2k+1)$*-state AfA* A *with zero-error.*

Proof. The AfA A does not use any classical state and it has $2k+1$ affine states, $\{e_1, \ldots, e_{2k+1}\}$. The initial state is e_{2k+1} and the only accepting state is e_1. It starts its computation in $v_0 = (0 \ \cdots \ 0 \ 1)^T$.

During the computation, it keeps the results in the values of the first $2k$ states, i.e. it sets each value to 0 or 1 depending the previous results and the current scanning symbols. More specifically, before each transition, the first $2k$ entries are shifted to the right by one and the $2k$-th entry becomes the first entry. Then, depending on the scanned symbol, the value of the first entry is updated:

– If the scanned symbol is 0, then, for calculating XOR value, the first entry is multiplied by 1, i.e., $0 \to 0$ and $1 \to 1$.
– If the scanned symbol is 1, then, for calculating XOR value, the first entry is multiplied by -1 and then 1 is added to this result, i.e., $0 \to 0 \to 1$ and $1 \to -1 \to 0$.

The last entry in the affine state is used to make the state vector well-formed. For example, if the state vector has $0 \leq t \leq 2k$ 1s in its first $2k$ entries, then the last entry is $1 - t$. After reading the whole input, the first entry has the result and the rest of entries are summed to the second entry: if the first entry is 1 (resp., 0), then the rest of the vector contains only 0 (resp., 1). Therefore, any member (resp., non-member) is accepted by A with probability 1 (resp., 0). □

Acknowledgements. We thank Evgenijs Vihrovs (University of Latvia) for his helpful discussions and anonymous reviewers for their very helpful comments.

The work is partially supported by ERC Advanced Grant MQC, Latvian State Research Programme NeXIT project No. 1. The work is also performed according to the Russian Government Program of Competitive Growth of Kazan Federal University. The research on Las-Vegas OBDDs (Section 5) is supported by Russian Science Foundation Grant 17-71-10152

References

1. Ablayev, F., Gainutdinova, A.: Complexity of quantum uniform and nonuniform automata. In: DLT'2005. LNCS, vol. 3572, pp. 78–87. Springer (2005)
2. Ablayev, F., Gainutdinova, A., Khadiev, K., Yakaryılmaz, A.: Very narrow quantum OBDDs and width hierarchies for classical OBDDs. Lobachevskii Journal of Mathematics 37(6), 670–682 (2016)
3. Ablayev, F.: Randomization and nondeterminism are incomparable for ordered read-once branching programs. ECCC (021) (1997)
4. Ablayev, F., Gainutdinova, A., Karpinski, M., Moore, C., Pollett, C.: On the computational power of probabilistic and quantum branching program. Information and Computation 203(2), 145–162 (2005)
5. Ablayev, F., Gainutdinova, A., Khadiev, K., Yakaryılmaz, A.: Very narrow quantum OBDDs and width hierarchies for classical OBDDs. In: Descriptional Complexity of Formal Systems, LNCS, vol. 8614, pp. 53–64. Springer (2014)
6. Ablayev, F.M., Karpinski, M.: On the power of randomized branching programs. In: ICALP. LNCS, vol. 1099, pp. 348–356. Springer (1996)
7. Ambainis, A., Watrous, J.: Two–way finite automata with quantum and classical states. Theoretical Computer Science 287(1), 299–311 (2002)
8. Ambainis, A., Yakaryılmaz, A.: Automata and quantum computing. Tech. Rep. 1507.01988, arXiv (2015)
9. Belovs, A., Montoya, J.A., Yakaryılmaz, A.: On a conjecture by Christian Choffrut. Int. J. Found. Comput. Sci. 28(5), 483–502 (2017)
10. Díaz-Caro, A., Yakaryılmaz, A.: Affine computation and affine automaton. In: Computer Science - Theory and Applications. LNCS, vol. 9691, pp. 146–160. Springer (2016)

11. Ďuriš, P., Hromkovič, J., Rolim, J.D., Schnitger, G.: Las Vegas versus determinism for one-way communication complexity, finite automata, and polynomial-time computations. In: STACS. LNCS, vol. 1200, pp. 117–128. Springer (1997)
12. Gainutdinova, A.F.: Comparative complexity of quantum and classical OBDDs for total and partial functions. Russian Mathematics 59(11), 26–35 (2015)
13. Gainutdinova, A., Yakaryılmaz, A.: Nondeterministic unitary OBDDs. In: Computer Science - Theory and Applications. LNCS, vol. 10304, pp. 126–140 (2017)
14. Hirvensalo, M., Moutot, E., Yakaryılmaz, A.: On the computational power of affine automata. In: Language and Automata Theory and Applications. LNCS, vol. 10168, pp. 405–417 (2017)
15. Hirvensalo, M., Seibert, S.: Lower bounds for Las Vegas automata by information theory. RAIRO-Theoretical Informatics and Applications 37(1), 39–49 (2003)
16. Hromkovič, J., Schnitger, G.: On the power of Las Vegas for one-way communication complexity, OBDDs, and finite automata. Information and Computation 169(2), 284–296 (2001)
17. Khadiev, K.: On the hierarchies for deterministic, nondeterministic and probabilistic ordered read-k-times branching programs. Lobachevskii Journal of Mathematics 37(6), 682–703 (2016)
18. Khadiev, K., Khadieva, A.: Reordering method and hierarchies for quantum and classical ordered binary decision diagrams. In: Computer Science – Theory and Applications, LNCS, vol. 10304. Springer (2017)
19. Klauck, H.: On quantum and probabilistic communication: Las Vegas and one-way protocols. In: STOC'00. pp. 644–651 (2000)
20. Kushilevitz, E., Nisan, N.: Communication complexity. Cambridge University Press (1997)
21. Moore, C., Crutchfield, J.P.: Quantum automata and quantum grammars. Theoretical Computer Science 237(1-2), 275–306 (2000)
22. Sauerhoff, M.: Quantum vs. classical read-once branching programs. In: Complexity of Boolean Functions. No. 06111 in Dagstuhl Seminar Proceedings, Internationales Begegnungs und Forschungszentrum für Informatik (2006)
23. Sauerhoff, M., Sieling, D.: Quantum branching programs and space-bounded nonuniform quantum complexity. Theoretical Computer Science 334(1), 177–225 (2005)
24. Savickỳ, P., Žák, S.: A read-once lower bound and a $(1,+k)$-hierarchy for branching programs. Theoretical Computer Science 238(1), 347–362 (2000)
25. Say, A.C.C., Yakaryılmaz, A.: Quantum finite automata: A modern introduction. In: Computing with New Resources. LNCS, vol. 8808, pp. 208–222. Springer (2014)
26. Villagra, M., Yakaryılmaz, A.: Language recognition power and succinctness of affine automata. Natural Computing DOI: (10.1007/s11047-017-9652-z)
27. Villagra, M., Yakaryılmaz, A.: Language recognition power and succinctness of affine automata. In: Unconventional Computation and Natural Computation. LNCS, vol. 9726, pp. 116–129. Springer (2016)
28. Wegener, I.: Branching Programs and Binary Decision Diagrams: Theory and Applications. SIAM (2000)