NEW SIZE HIERARCHIES FOR TWO WAY AUTOMATA

1CENTER FOR QUANTUM COMPUTER SCIENCE, FACULTY OF COMPUTING, UNIVERSITY OF LATVIA RAINA BULV. 19, RĪGA, LV-1586, LATVIA
2 KAZAN FEDERAL UNIVERSITY, KREMLEVSKAYA STR. 18, 420008, KAZAN, RUSSIA
E-mail address: kamilhadi@gmail.com, rishat.ibrahimov@yandex.ru, abuzer@lu.lv

ABSTRACT. We introduce a new type of nonuniform two–way automaton that can use a different transition function for each tape square. We also enhance this model by allowing to shuffle the given input at the beginning of the computation. Then we present some hierarchy and incomparability results on the number of states for the types of deterministic, nondeterministic, and bounded-error probabilistic models. For this purpose, we provide some lower bounds for all three models based on the numbers of subfunctions and we define two witness functions.

1. INTRODUCTION

Nonuniform models (like circuits, branching programs, uniform models using advice, etc.) have played significant roles in computational complexity, and, naturally they have also been investigated in automata theory (e.g. [7, 12, 14, 6]). The main computational resource for nonuniform automata is the number of internal states that depends on the input size. Thus we can define linear, polynomial, or exponential size automata models. In this way, for example, nonuniform models allow us to formulate the analog of “P versus NP problem” in automata theory: Sakoda and Sipser [24] conjectured that simulating a two–way nondeterministic automaton by two–way deterministic automata requires exponential number of states in the worst case. But, the best known separation is only quadratic ($O(n^2)$) [9, 13] and the researchers have succeeded to obtain

2000 Mathematical Subject Classification. 46L51, 46L53, 46E30, 46H05.

Key words and phrases. Two–way nonuniform automaton, size hierarchy, deterministic and nondeterministic models, probabilistic computation.
slightly better bounds only for some modified models (e.g. [21, 15, 11, 16]). Researchers also considered similar question for OBDD model that can be seen as nonuniform automata (e.g. [18], [20], [4], [2], [1], [19]).

In this paper, we present some hierarchy results for deterministic, nondeterministic, and bounded-error probabilistic nonuniform two-way automata models, which can also be seen as a “two-way” version of ordered binary decision diagrams (OBDDs) [27]. For each input length \( n \), our models can have different number of states, and, like Branching programs or the data-independent models defined by Holzer [12], the transition functions can be changed during the computation. Holzer’s model can use a different transition function for each step. We restrict this property so that the transition function is the same for the same tape positions, and so, we can have at most \( n \) different transition functions. Moreover, we enhance our models by shuffling the input symbols at the beginning of the computation. We give the definitions and related complexity measures in Section 2.

In order to obtain our main results, we start with presenting some generic lower bounds (Section 3) by using the techniques given in [25] and [10]. Then, we define two witness Boolean functions in Section 3.2.1 \textit{Shuffled Address Function}, denoted 2-SAF\(_t\), which is a modification of Boolean functions given in [22] (see also [5], [8], [13], [17]), and its uniform version 2-USAF\(_t\). Moreover, regarding these functions, we provide two deterministic algorithms. In our results, we also use the well known \textit{Equality function} \( \text{EQ}(X) = \bigvee_{0}^{\lfloor n/2 \rfloor} \bigwedge_{x} x_{i} = x_{i+\lfloor n/2 \rfloor} \).

In Sections 4 and 5 we present our main results based on the size (the number of states) of models. We obtain linear size separations for deterministic models and quadratic size separations for nondeterministic and probabilistic models. Moreover, we investigate the effect of shuffling for all three types of models, and, we show that in some cases shuffling can save huge amount of states and in some other cases shuffling cannot be size efficient. We also show that the constant number of states does not increase the computational power of deterministic and nondeterministic nonuniform models without shuffling.

2. Definitions

Our alphabet is binary, \( \Sigma = \{0, 1\} \). We mainly use the terminologies of Branching programs: Our decision problems are solving/computing Boolean functions: The automaton solving a function accepts the inputs where the function gets the value true and rejects the inputs where the function gets the value false. For uniform models, on the other hand, our decision problems are recognizing languages: The automaton recognizing a language accepts any member and rejects any non-member.
A nonuniform head–position–dependent two–way deterministic automaton working on the inputs of length/size \( n \geq 0 \) \((2DA_n)\) \( D_n \) is a 6-tuple 
\[
D_n = (\Sigma, S, s_1, \delta = \{\delta_1, \ldots, \delta_n\}, s_a, s_r),
\]
where (i) \( S = \{s_1, \ldots, s_d\} \) is the set of states (\( d \) can be a function in \( n \)) and \( s_1, s_a, s_r \in S \) \((s_a \neq s_r)\) are the initial, accepting, and rejecting states, respectively; and, (ii) \( \delta \) is a collection of \( n \) transition functions such that \( \delta_i : S \setminus \{s_a, s_r\} \times \Sigma \rightarrow S \times \{←, ↓, →\} \) is the transition function that governs behaviour of \( D_n \) when reading the \( i \)th symbol/variable of the input, where \( 1 \leq i \leq n \). Any given input \( u \in \Sigma^n \) is placed on a read-only tape with a single head as \( u_1 u_2 \cdots u_n \) from the squares 1 to \( |u| = n \), where \( u_i \in \Sigma \) is the \( i \)th symbol of \( u \). When \( D_n \) is in \( s \in S \setminus \{s_a, s_r\} \) and reads \( u_i \in \Sigma \) on the tape, it switches to state \( s' \in S \) and updates the head position with respect to \( a \in \{←, ↓, →\} \) if \( \delta_i(s, u_i) \rightarrow (s', a) \). If \( a = "←" \) ("→"), the head moves one square to the left (the right), and, it stays on the same square, otherwise. The transition functions \( \delta_1 \) and \( \delta_n \) must be defined to guarantee that the head never leaves \( u \) during the computation. Moreover, the automaton enters \( s_a \) or \( s_r \) only on the right most symbol and then the input is accepted or rejected, respectively.

The nondeterministic counterpart of \( 2DA_n \), denoted \( 2NA_n \), can choose from more than one transition in each step. So, the range of each transition function is \( \mathcal{P}(S \times \{←, ↓, →\}) \), where \( \mathcal{P}(\cdot) \) is the power set of any given set. Therefore, a \( 2NA_n \) can follow more than one computational path and the input is accepted only if one of them ends with the decision of “acceptance”. Note that some paths end without any decision since the transition function can yield the empty set for some transitions.

The probabilistic counterpart of \( 2DA_n \), denoted \( 2PA_n \), is a \( 2NA_n \) such that each transition is associated with a probability. Thus, \( 2PA_n \)'s can be in a probability distribution over the deterministic configurations (the state and the position of head forms a configuration) during the computation. To be a well-formed machine, the total probability must be 1, i.e. the probability of outgoing transitions from a single configuration must be always 1. Thus, each input is accepted and rejected by a \( 2PA_n \) with some probabilities. An input is said to be accepted/rejected by a (bounded-error) \( 2PA_n \) if the accepting/rejecting probability by the machine is at least \( 1/2 + \epsilon \) for some \( \epsilon \in (0, 1/2] \).

A function \( f_n : \Sigma^n \rightarrow \Sigma \) is said to be computed by a \( 2DA_n \) \( D_n \) (a \( 2NA_n \) \( N_n \), a \( 2PA_n \) \( P_n \)) if each member of \( f_n^{-1}(1) \) is accepted by \( D_n \) (\( N_n \), \( P_n \)) and each member of \( f_n^{-1}(0) \) is rejected by \( D_n \) (\( N_n \), \( P_n \)).

The class \( 2DSIZE(d(n)) \) is formed by the functions \( f = \{f_0, f_1, f_2, \ldots\} \) such that each \( f_i \) is computed by a \( 2DA_i \) \( D_i \), the number of states of which is no more than \( d(i) \), where \( i \) is a non-negative integer. We can similarly define
nondeterministic and probabilistic counterparts of this class, denoted \(2\text{NSIZE}(d(n))\) and \(2\text{PSIZE}(d(n))\) respectively.

We also introduce a generalization of our nonuniform models that can shuffle the input at the beginning of the computation with respect to a permutation. A nonuniform head–position–dependent shuffling two–way deterministic automaton working on the inputs of length/size of \(n \geq 0\) (2DA\(\Theta\)\(n\)), say \(D\theta_n\), is a 2DA\(n\) that shuffles the symbols of input with respect to \(\theta\), a permutation of \(\{1, \ldots, n\}\), i.e. the \(j\)-th symbol of the input is placed on \(\theta(j)\)-th place on the tape \((1 \leq j \leq n)\), and then execute the 2DA\(n\) algorithm on this new input. The nondeterministic and probabilistic models can be respectively abbreviated as \(2\text{NA}_n\) and \(2\text{PA}_n\).

The class \(2\text{D}\Theta\text{SIZE}(d(n))\) is formed by the functions \(f = \{f_0, f_1, f_2, \ldots\}\) such that each \(f_i\) is computed by a 2DA\(\Theta\)\(i\) \(D\theta_i\) whose number of states is no more than \(d(i)\), where \(i\) is a non-negative integer and \(\theta\) is a permutation of \(\{1, \ldots, n\}\). The nondeterministic and probabilistic classes are respectively represented by \(2\Theta\text{NSIZE}(d(n))\) and \(2\Theta\text{PSIZE}(d(n))\).

Moreover we consider uniform versions of two-way automata, respectively 2DFA and 2NFA. We can define 2DFA in the same way as 2DA\(n\), but it is identical for all \(n\), \(|S| = \text{const}\) and \(\delta_i = \delta\) for any \(i \in \{1, \ldots, n\}\). Moreover, they can use end-markers, between which the given input is placed on the input tape. We can define 2NFA similarly. The corresponding classes of languages defined by 2DFAs and 2NFAs of size \(d\) are denoted \(2\text{DFASIZE}(d)\) and \(2\text{NFASIZE}(d)\), respectively.

### 3. Lower bounds, Boolean functions, and algorithms

Our key complexity measure behind our results is the number of subfunctions for a given function. It can be seen as the counterpart of “the equivalence classes of a language” with respect to Myhill-Nerode Theorem [23].

Let \(f\) be a Boolean function defined on \(X = \{x_1, \ldots, x_n\}\). We define the set of all permutations of \((1, \ldots, n)\) as \(\Theta(n)\). Let \(\theta \in \Theta(n)\) be a permutation. We can order the elements of \(X\) with respect to \(\theta\), say \((x_1', \ldots, x_n')\), and then we can split them into two disjoint non-empty (ordered) sets by picking an index \(i \in \{1, \ldots, n - 1\}\): \(X_A = (x_1', \ldots, x_i')\) and \(X_B = (x_{i+1}', \ldots, x_n')\). Let \(\rho\) be a mapping \(\{x_1', \ldots, x_i'\} \rightarrow \{0, 1\}\) that assigns a value to each \(x_j' \in (x_1', \ldots, x_i')\). Then, we define function \(f|\rho : X_B \rightarrow \{0, 1\}\) that returns the value of \(f\) where the values of the input from \(X_A\) are fixed by \(\rho\). The function \(f|\rho\) is called a subfunction.

The total number of different subfunctions with respect to \(\theta\) and \(i\) is denoted by \(N_i^\theta(f)\). Then, we focus on the maximum value by considering all possible indices:

\[
N^\theta(f) = \max_{i \in \{1, \ldots, n-1\}} N_i^\theta(f).
\]
After this, we focus on the best permutation that minimizes the number of subfunctions:

\[ N(f) = \min_{\theta \in \Theta(n)} N^\theta(f). \]

Now, we represent the relation between the number of subfunctions for Boolean function and the number of equivalence classes of a language.

Let \( L \) be a language defined on \( \Sigma = \{0, 1\} \). For a given non-negative integer \( n \), \( L_n \) is the language composed by all members of \( L \) with length \( n \), i.e. \( L_n = L \cap \Sigma^n = \{ w \mid w \in L, |w| = n \} \). For \( r < n \), two strings \( u \in \Sigma^r \) and \( v \in \Sigma^r \) are said to be equivalent if for any \( y \in \Sigma^{n-r} \), \( uy \in L_n \) if and only if \( vy \in L_n \).

We denote the number of non-equivalent strings of length \( r \) as \( R^r(L_n) \). Then, similar to the number of subfunctions,

\[ R(L_n) = \max_{r \in \{1, \ldots, n-1\}} R^r(L_n) \text{ and } R_n(L) = R(L_n). \]

The function \( f^L_n(X) \) denotes the characteristic Boolean function for language \( L_n \). Thus, we can say that

\[ N^{id}(f^L_n) = R_n(L). \]

for \( id = (1, \ldots, n) \) is natural order.

3.1. **Lower bounds.** First we give our lower bounds on the sizes of models in terms of \( N(f) \). Note that all of lower bounds for shuffling models are valid also for non-shuffling models.

**Theorem 1.** If the function \( f(X) \) is computed by a 2DA\( Q_n \) of size \( d \) for some permutations \( \theta \), then

\[ N(f) \leq (d + 1)^{d+1}. \]

**Proof.** This result is easily obtained by using the standard and well-known conversion given by Shepherdson [25]. \( \square \)

**Corollary 1.** If the language \( L \) is recognized by a 2DFA of size \( d \), then

\[ R_n(L) \leq (d + 1)^{d+1}, \]

**Theorem 2.** If the function \( f(X) \) is computed by a 2NA\( Q_n \) of size \( d \) for some permutations \( \theta \), then

\[ N(f) \leq 2^{(d+1)^2}. \]

**Proof.** This result follows from [26]. \( \square \)

**Corollary 2.** If the language \( L \) is recognized by a 2NFA of size \( d \), then

\[ R_n(L) \leq 2^{(d+1)^2}. \]
Based on Theorems 1 and 2, we can obtain the following result.

**Theorem 3.** For constant integer $d$, $2DSIZE(d)$ and $2NSIZE(d)$ contain only characteristic functions of regular languages.

**Proof.** If $d$ is constant and $2DA_n$ (or $2NA_n$) $A_n$ computes $f_n$ then $N(f_n)$ is constant and it is same for each $n$. Hence the $R(L_n)$ of the corresponding language $L_n$ is constant too, so the language $L = \bigcup L_n$ is regular since the number of equivalence classes is finite with respect to Myhill-Nerode Theorem [23]. □

**Theorem 4.** If the function $f(X)$ is computed by a $2PA_\Theta$ of size $d$ for some permutations $\theta$ with expected running time $T$ and error probability $\varepsilon$, then

$$N(f) \leq \left\lceil \frac{4d(8 + 3\log T)}{\log(1 + 2\varepsilon)(1 + \varepsilon)} \right\rceil^{(d+1)^2}.$$ 

**Proof.** This result follows from the techniques given in [10].

We need some additional definitions to present our lower bound for $2PA_\Theta$. Let $\beta \geq 1$. Two numbers $p$ and $p'$ are said to be $\beta$−close if either

- $p = p' = 0$ or
- $p \geq 0$, $p' \geq 0$, and $\beta^{-1} \leq p/p' \leq \beta$.

Two numbers $p$ and $p'$ are said to be $\beta$−close mod $\lambda$ if either

- $p \leq \lambda$ and $p' \leq \lambda$ or
- $p \geq \lambda$, $p' \geq \lambda$ and $p$ and $p'$ are $\beta$−close.

Let $P = \{p_{i,j}\}_{i,j=1}^{m}$ be an $m$-state Markov chain with starting state 1 and two absorbing states $m - 1$ and $m$; $a(P)$ denote the probability that Markov chain $P$ is absorbed in state $m$ when started in state 1; and, $T(P)$ denote the expected time to absorption into one of the states $m - 1$ or $m$. Two Markov chains $P = \{p_{i,j}\}_{i,j=1}^{m}$ and $P' = \{p'_{i,j}\}_{i,j=1}^{m}$ are said to be $\beta$−close mod $\lambda$ if, for each pair $i, j$, $p_{i,j}$ and $p'_{i,j}$ are $\beta$−close mod $\lambda$.

Let $\pi \in \Pi(\theta)$ be any partition such that $\theta \in \Theta(n)$ is the order of the inputs for $A_\theta$, and, $\pi = (X_A, X_B)$ and $|X_A| = u$.

Let us consider configurations $b_i$ for $i \in \{0, \ldots, 2d + 2\}$. Configuration $b_0$ is initial configuration of the automata, $b_{d+1}$ is accepting state and position of head on the last symbol, $b_{d+2}$ is similar but in rejecting state. For $i \in \{1, \ldots, d\}$, configuration $b_i$ is for position $u$ and state of the automata $i$. For $i \in \{d + 3, \ldots, 2d + 2\}$, configuration $b_i$ is for position $u + 1$ and state of the automata $i - d - 3$. We will use three object for describing computational process for the automata: matrix $M_A(\sigma, \gamma)$ and vectors $p^\theta, q$. Matrix $M_A(\sigma, \gamma)$ is $(2d + 3) \times (2d + 3)$ block diagonal matrix with two blocks $M_A(\sigma)$ and $M_A(\gamma)$. $i$-th
line and row of the matrix $M_A(\sigma, \gamma)$ corresponding to configuration $b_i$. Matrices $M_A(\sigma) = \{m_{i,j}\}$, $M_A(\sigma) = \{m'_{i,j}\}$ have following elements:

- If $A^\theta_n$ begins the computation from the configuration $b_i$, then it reaches the configuration $b_{j+d+3}$ early than another $b_r$ for $r \geq d+3$ within probability $m_{ij}$.
- If $A^\theta_n$ begins the computation from the configuration $b_{i+d+3}$, then it reaches the configuration $b_j$ early than another $b_r$ for $r \leq d$ within probability $m'_{ij}$.

The vector $p^0$ and $q$ have size $(2d+3)$ and $i$-th element of vectors also correspond to $b_i$. The vector $p^0$ represents initial distribution of probability for configuration of the automata. So $p^0_0 = 1$ and other elements are 0. The vector $q$ is characteristic vector of accepting state. So, $q_{b+1} = 1$ and other elements are 0.

Let us discuss some properties.

**Lemma 1.** If the function $f(X)$ is computed by $A^\theta_n$ of size $d$ within error probability $\varepsilon$, then for any $\nu \in \{0, 1\}^n$ satisfying $f(\nu) = 1$, there is a $t'$ such that

$$p^0 \cdot \left(M_A(\sigma, \gamma)\right)^{t'} \cdot q \geq \frac{1}{2} + \varepsilon$$

On the other hand, for any $\nu \in \{0, 1\}^n$ satisfying $f(\nu) = 0$ there is no such $t'$.

**Proof.** Since computation is probabilistic, there can be more than one path from the configuration $b_i$, where it reaches the configuration $b_{j+d+3}$ early than another $b_r$ for $r \geq d+3$. But by construction of matrix $M_A$, we consider all of them. In [10] have shown that we can model probabilistic computation in this way. \qed

We have shown that we model computation of $2\text{PA}^\Theta_n$ by Markov chain specified by matrix $M_A(\sigma, \gamma)$.

**Lemma 2.** Let $P$ and $P'$ are two $m$-state Markov chains and $a(P) \geq \frac{1}{2} + \varepsilon$. Let $T = \max (T(P), T(P'), m)$, $\lambda = \varepsilon^2/256 T^3$, $\beta = 2m \sqrt{\frac{1+\varepsilon^2}{1+\varepsilon}}$. If $P$ and $P'$ are $\beta$-close mod $\lambda$, then $a(P') \geq \frac{1}{2} + \varepsilon/4$.

**Proof.** Dwork and Stockmeyer [10] have shown that

$$a(P') \geq (1 - 2\lambda m^3) \beta^{-2m} a(P) - 4\sqrt{\lambda m T}.$$  

Therefore the bound on $a(P')$ may be obtained by substituting the values of $\lambda$ and $\beta$. \qed

Let $M$ be the set of all possible matrices $M_A(\sigma)$ such that $M = \{M_A(\sigma) : \sigma \in \{0, 1\}^{|X_A|}\}$ and, for any $P \in M$ and $P' \in M$, $P$ and $P'$ are $\beta$-close mod $\lambda$. 

The inequality $N^{id}(f) \leq |M|$ can be obtained in the same way. To estimate $|M|$ we use technique similar to one from [10]. Let $c = d(d + 1)$. Define an equivalence relation on matrices in $M$ as follows: $Q(w) \equiv Q'(w) \iff \forall i, j \quad (q_{i,j}(w) \leq \lambda \iff q'_{i,j}(w) \leq \lambda)$. Let $E$ be a largest equivalence class. Since there are at most $2^c$ equivalence classes, $|M| \leq 2^c|E(w)|$. Size of $E(w)$ is obtained in [10], since by substituting values $\lambda$ and $\beta$ we have:

$$N^{id}(f) \leq |M| \leq 2^c|E(w)| \leq \left\lceil \frac{4d(8 + 3 \log T)}{\log(1 + 2\varepsilon)/(1 + \varepsilon)} \right\rceil^c.$$ 

The proof of Theorem 3 is completed.

**Corollary 1.** If the function $f(X)$ is computed by a 2PA$_n^\Theta$ of size $d$ for some permutations $\theta$ with expected running time $T \geq 256$ and error probability $\varepsilon = \frac{1}{5}$, then

$$N(f) \leq (32d \log T)^{(d+1)^2}.$$ 

**Proof.** For $\varepsilon = \frac{1}{5}$, $\log (1 + 2\varepsilon) (1 + \varepsilon) > \frac{1}{2}$, and, for $T \geq 256$, $\log T \geq 8$. □

3.2. **Boolean Functions.** We define two Boolean functions: (1) A modification of Boolean function given in [3, 17, 22, 8, 13] **Shuffled Address Function**, denoted $2$-$SAF_t$, and (2) **Uniform Shuffled Address Function** $2$-$USAF_t$ as a modification of $2$-$SAF_t$. We also use the language $L_2USAF_t$, the characteristic function of which is $2$-$USAF_t$.

3.2.1. **Boolean Function $2$-$SAF_t$:** We divide all input into two parts, and each part into $t$ blocks. Each block has address and value. Formally, Boolean function $2$-$SAF_t(X) : \{0, 1\}^n \to \{0, 1\}$ for integer $t = t(n)$ such that

$$2t(2t + \lceil \log 2t \rceil) < n.$$ 

We divide the input variables (the symbols of the input) into $2t$ blocks. There are $\left\lfloor \frac{n}{2t} \right\rfloor = q$ variables in each block. After that, we divide each block into address and value variables (see Figure 1). The first $\lceil \log 2t \rceil$ variables of block are address and the other $q - \lceil \log 2t \rceil = b$ variables of block are value. We call $x^p_0, \ldots, x^p_{b-1}$

---

$M(\sigma)$ includes all information about the behaviour of automaton on $\sigma$ and so if there are two different $\sigma$ and $\sigma'$ with different subfunctions but their matrices $M(\sigma)$ and $M(\sigma')$ are $\beta - close$ mod $\lambda$, it should be different on some $\gamma$. But the behaviour of the automaton on this input is the same and therefore matrices $M(\sigma, \gamma)$ and $M(\sigma, \gamma')$ are $\beta - close$ mod $\lambda$. This is a contradiction.
and \( y_0^p, \ldots, y_{\lfloor \log 2^w \rfloor}^p \) are the \textit{value} and the \textit{address} variables of the \( p \)th block, respectively, for \( p \in \{0, \ldots, 2t-1\} \).

\[
\begin{array}{|c|c|c|c|c|c|}
\hline
\text{Address} & \text{Value} & \text{Address} & \text{Value} & \cdots & \text{Address} & \text{Value} \\
\hline
\text{block 0} & \text{block 1} & \cdots & \text{block t-1} & \text{block t} & \text{block 2t-1} \\
\hline
\end{array}
\]

\textbf{Figure 1. Address and value bits of blocks.}

Function \( 2\text{-SAF}_t(X) \) is calculated based on the following five sub-routines:

1. \( \text{Adr} : \{0,1\}^n \times \{0,\ldots,2t-1\} \rightarrow \{0,\ldots,2t-1\} \) gets the address of a block:

\[
\text{Adr}(X,p) = \sum_{j=0}^{\lfloor \log 2^t \rfloor - 1} y_j^p \cdot 2^j \mod (2^t).
\]

2. \( \text{Ind} : \{0,1\}^n \times \{0,\ldots,2t-1\} \rightarrow \{-1,\ldots,2t-1\} \) gets the number of block by address:

\[
\text{Ind}(X,a) = \left\{ \begin{array}{ll}
p & \text{where } p \text{ is the minimal number such that } \text{Adr}(X,p) = a, \\
-1 & \text{if there are no such } p.
\end{array} \right.
\]

3. \( \text{Val} : \{0,1\}^n \times \{0,\ldots,2t-1\} \rightarrow \{-1,\ldots,t-1\} \) gets the value of the block with address \( i \):

\[
\text{Val}(X,a) = \left\{ \begin{array}{ll}
\sum_{j=0}^{b-1} x_j^p \text{ (mod } t) & \text{if } \text{Ind}(X,a) = p \text{ for } p \geq 0, \\
-1 & \text{if } \text{Ind}(X,a) < 0.
\end{array} \right.
\]

Suppose that we are at the \( i \)-th step of iteration.

4. \( \text{Step}_1 : \{0,1\}^n \times \{0,\ldots,1\} \rightarrow \{-1,t,\ldots,2t-1\} \) gets the first part of the \( i \)th step of iteration:

\[
\text{Step}_1(X,i) = \left\{ \begin{array}{ll}
-1 & \text{if } \text{Step}_2(X,i-1) = -1, \\
\text{Val}(X,\text{Step}_2(X,i-1)) + t & \text{otherwise}
\end{array} \right.
\]

5. \( \text{Step}_2 : \{0,1\}^n \times \{-1,\ldots,1\} \rightarrow \{-1,\ldots,t-1\} \) gets the second part of the \( i \)th step of iteration:

\[
\text{Step}_2(X,i) = \left\{ \begin{array}{ll}
-1 & \text{if } \text{Step}_1(X,i) = -1, \\
2 & \text{if } i = -1, \\
\text{Val}(X,\text{Step}_1(X,i)) & \text{otherwise}
\end{array} \right.
\]

Function \( 2\text{-SAF}_t(X) \) is computed iteratively: \( 2\text{-SAF}_t(X) = \left\{ \begin{array}{ll}
0 & \text{if } \text{Step}_2(X,1) \leq 0, \\
1 & \text{otherwise}
\end{array} \right. \)

(1) We find the block with address 2 in the first part and compute the value of this block, which is the address of the block for the second part.
(2) We take the block from the second part with the computed address and compute value of the block, which is the address of the new block for the first part.

(3) We find the block with new address in the second part and check value of this block. If the value is greater than 0, then value of $2\cdot\text{SAF}_t$ is 1, and 0 otherwise.

If we do not find block with searching address $n$ in any phase then value of $2\cdot\text{SAF}_t$ is also 0. See the Figure 2 for the iterations of the function.

![Diagram](image)

**Figure 2.** $2\cdot\text{SAF}_t(X)$, $\text{Step}_1(X, i)$ and $\text{Step}_2(X, i)$ functions. In the picture we write $a$ in address bits of $p$-th block (gray box) if $A(X, p) = a$.

**Theorem 5.** For integer $t = t(n)$, $N(2\cdot\text{SAF}_t) \geq t^2 - 2$, where $t$ satisfies $2t(2t + \lceil \log 2t \rceil) < n$.

**Proof.** The proof is based on the following two technical Lemmas 3 and 4.

**Lemma 3.** Let $t = t(n)$ be some integers satisfying Inequality (1) and $\pi = (X_A, X_B)$ be a partition such that $X_A$ contains at least $t$ value variables from exactly $t$ blocks. Then, $X_B$ contains at least $t$ value variables from exactly $t$ blocks.

**Proof.** We define $I_A = \{i : X_A \text{ contains at least } t \text{ value variables from } i\text{th block}\}$. Let $i' \notin I_A$. Then, $X_A$ contains at most $t - 1$ value variables from $i'$th block, so $X_B$ contains at least $b - (t - 1) \text{ value variables from } i'$th block. By (1), we can get

$$b - (t - 1) = \left\lfloor \frac{n}{2t} \right\rfloor - \lceil \log 2t \rceil - (t - 1)$$

which is bigger than

$$(2t + \lceil \log 2t \rceil) - \lceil \log 2t \rceil - (t - 1) = 2t - (t - 1) = t + 1.$$
Let $I = \{0, \ldots, 2t - 1\}$ be the numbers of all blocks and $i' \in I \setminus I_A$. Then, we can follow that $|I \setminus I_A| = 2t - t = t$. \hfill \Box

Let $\theta \in \Theta(n)$ be any order. Then, we pick a partition $\pi = (X_A, X_B) \in \Pi(\theta)$ such that $X_A$ contains at least $t$ value variables from exactly $t$ blocks. We define $I_A = \{i : X_A \text{ contains at least } t \text{ value variables from } i\text{th block}\}$ and $I_B = \{0, \ldots, 2t - 1\} \setminus I_A$. By the proof of Lemma 3, we know that $|I_B| = t$.

Let $(\sigma, \gamma)$ be the partition for the input $\nu$ with respect to $\pi$. We define the sets $\Psi \subset \{0, 1\}^{|X_A|}$ and $\Gamma \subset \{0, 1\}^{|X_B|}$ for the input $\nu$ with respect to $\pi$ that satisfies the following conditions. For $\sigma, \sigma' \in \Psi, \gamma \in \Gamma, \nu = (\sigma, \gamma)$, and $\nu' = (\sigma', \gamma)$:

- For any $z \in \{0, \ldots, t - 1\}$, $\text{Ind}(\nu, z) \in I_A$;
- For any $z \in \{w, \ldots, 2t - 1\}$, $\text{Ind}(\nu, z) \in I_B$;
- There is $z \in \{2, \ldots, t - 1\}$ such that $\text{Val}(\nu', z) \neq \text{Val}(\nu, z)$;
- The value of $x^p_i$ is 0 for any $p \in I_B$ and $x^p_i \in X_A$;
- The value of $x^p_j$ is 0 for any $p \in I_A$ and $x^p_j \in X_B$;
- $\text{Val}(\nu, 0) = 2t - 2, \text{Val}(\nu', 1) = 2t - 1$; and,
- $\text{Val}(\nu, 2t - 2) = 0, \text{Val}(\nu, 2t - 1) = 1$.

Lemma 4. For any sequence $(a_2, \ldots, a_{2t-3})$, where $a_i \in \{0, \ldots, t - 1\}$, there are $\sigma \in \Psi$ and $\gamma \in \Gamma$ such that $\text{Val}(\nu, z_i) = a_i$ for $z_i \in \{2, \ldots, t - 1\}$ and $i \in \{2, \ldots, 2t - 3\}$.

Proof. Let $p_i \in I_A$ such that $p_i = \text{Ind}(\nu, z_i)$ for $i \in \{2, \ldots, t - 1\}$. Remember that the value of $x^{p_i}_j$ is 0 for any $p_j \in X_B$. Hence the value of $\text{Val}(\nu, z_i)$ depends only on the variables from $X_A$. At least $t$ value variables of $p_i$th block belong to $X_A$. Hence we can choose input with $a_i$ 1’s in the value variables of $p_i$th block which belongs to $X_A$. For set $\Gamma$ and $i \in \{t, \ldots, 2t - 3\}$, we can follow the same proof. \hfill \Box

Remember the statement of the theorem: For integer $t = t(n)$, if 2-SAF$_t$ satisfies Inequality [1], then

$$N(2\text{-SAF}_t) \geq t^{t-2}.$$}

Here are the details for the proofs.

Let $\theta \in \Theta(n)$ be an order. Then, we pick the partition $\pi = (X_A, X_B) \in \Pi(\theta)$ such that $X_A$ contains at least $t$ value variables from exactly $t$ blocks.

Let $\sigma, \sigma' \in \Psi$ be two different inputs and $\tau$ and $\tau'$ be their corresponding mappings, respectively. We show that the subfunctions 2-SAF$_t|_{\tau}$ and 2-SAF$_t|_{\tau'}$ are different. Let $z \in \{2, \ldots, t - 1\}$ such that $s' = \text{Val}(\nu', z) \neq \text{Val}(\nu, z) = s$.

If $z > 2$, then we choose $\gamma \in \Gamma$ providing that $\text{Val}(\nu, s + t) = 1, \text{Val}(\nu', s' + t) = 0$, and $\text{Val}(\nu, r) = \text{Val}(\nu', r) = z$, where $r = \text{Val}(\nu, 2)$. That is,

- $\text{Step}_1(\nu, 0) = \text{Step}_1(\nu', 0) = r$,
Theorem 6. There is a 2DA \( A_n \) of size \( 13t + 4 \) that computes 2-SAF.

Proof. Let \( \nu \in \{0, 1\}^n \) be the input. We begin with the first part of automaton \( A_n \) that computes \( \text{Step}_1(X, 0) \). Automaton \( A_n \) checks each \( j \)th block for predicate \( \text{Adr}(\nu, j) = 2 \). If it is true, then \( A_n \) computes \( \text{Val}(\nu, 2) = r' \), and, it checks the next block, otherwise. If \( A_n \) checks all blocks and does not find the block, it switches to the rejecting state. If \( A_n \) finds \( r' \), then it goes to one of the special state \( s'_{r'} \). From this state, the automaton returns back to the beginning of the input.

We continue with the second part of automaton \( A_n \) that computes \( \text{Step}_2(X, 0) \). From the state \( s'_{r'} \), \( A_n \) checks each \( j \)th block for predicate \( \text{Adr}(\nu, j) = r' \). If it is true, then \( A_n \) computes \( \text{Val}(\nu, r') = r'' \), and, it checks the next block, otherwise. If \( A_n \) checks all blocks and does not find the block, it switches to the rejecting state. If \( A_n \) finds \( r'' \), then it goes to one of special state \( s''_{r''} \). From this state, the automaton returns back to the beginning of the input.

Now, we describe the third part of automaton \( A_n \) that computes \( \text{Step}_1(X, 1) \). From the state \( s''_{r''} \), \( A_n \) checks each \( j \)th block for predicate \( \text{Adr}(\nu, j) = r'' \). If it is true, then \( A_n \) computes \( \text{Val}(\nu, r'') = r''' \), and, it checks the next block, otherwise. If \( A_n \) checks all blocks and does not find the block, then it switches to the rejecting state. If \( A_n \) finds \( r''' \), then it goes to one of special state \( s'''_{r'''} \). From this state, the automaton returns back to the beginning of the input.

The forth part of automaton \( A_n \) computes \( \text{Step}_2(X, 1) \). From the state \( s'''_{r'''} \), \( A_n \) checks each \( j \)th block for predicate \( \text{Adr}(\nu, j) = r''' \). If it is true, then \( A_n \) computes \( \text{Val}(\nu, r''') = r''IV \), and, it checks next block otherwise. If \( A_n \) checks
all blocks and does not find the block, it switches to the rejecting state. If \( A_n \) finds \( r^{IV} \) and \( r^{IV} = 1 \), the automaton accepts the input and rejects the input, otherwise.

In the first part, the block checking procedure uses only 2 states. Computing \( \text{Val}(\nu, 2) \) uses \( w \) states and there are \( t \) \( s'_r \) states. So, the size of the first part is \( 2 + 2t \). In the second part, the block checking procedure uses only 2 states and we have \( t \) blocks to check the state pairs for each value of \( r' \), that is \( 2t \) states. Computing \( \text{Val}(\nu, r') \) uses \( t \) states and also \( A_n \) has \( t \) \( s''_r \) states. Therefore, the size of the second part is \( 4t \). Similarly, we can show that the size of the third part is \( 4t \). In the fourth part, we need \( 3t \) states for procedure checking and computing \( \text{Val}(\nu, r''') \). \( A_n \) has also one accept and one reject states. So, the size of the forth part is \( 3t + 2 \). Thus, the overall size of \( A_n \) is \( 13t + 4 \). □

3.2.2. Boolean Function 2-\( \text{USAF}_t \): The definition of 2-\( \text{USAF}_t \) is as follows:

\[
\text{2-\( \text{USAF}_t \)}(X) : \{0, 1\}^n \rightarrow \{0, 1\} \text{ for integer } t = t(n) \text{ satisfying that }
\]

\[
4t(2t + \lceil \log 2t \rceil) < n.
\]

We denote its language version as \( L_{2\text{USAF}_t} \).

We divide the input variables (the symbols of the input) into \( 2t \) blocks. There are \( \lfloor n/2t \rfloor = q \) variables in each block. After that, we divide each block into mark, address, and value variables. All variables that are in odd positions are mark. The type of the bit on an even position is determined by the value of previous mark bit. The bit is address, if the previous mark bit’s value is 0, and value otherwise. The first \( \lceil \log 2t \rceil = c \) variables of block that are in the even positions denote the address and the other \( q/2 - \lceil \log 2t \rceil = b \) variables of block denote the value.

We call \( x^p_0, \ldots, x^p_{b-1}, y^p_0, \ldots, y^p_{\lceil \log 2t \rceil} \) and \( z^p_0, \ldots, z^p_{q/2} \) are the value, address, and the mark variables of the \( p \)th block, respectively, for \( p \in \{0, \ldots, 2t - 1\} \).

Function 2-\( \text{USAF}_t(X) \) is calculated based on the following five sub-routines:

1. \( \text{Adr} : \{0, 1\}^n \times \{0, \ldots, 2t - 1\} \rightarrow \{0, \ldots, 2t - 1\} \) gets the address of a block:

\[
\text{Adr}(X, p) = \sum_{j=0}^{c-1} y^p_j \cdot 2^{c-j-1} \text{ (mod } 2t).\]

2. \( \text{Ind} : \{0, 1\}^n \times \{0, \ldots, 2t - 1\} \rightarrow \{-1, \ldots, 2t - 1\} \) gets the number of block by address:

\[
\text{Ind}(X, a) = \begin{cases} 
p & \text{, where } p \text{ is the minimal number such that } \text{Adr}(X, p) = a, \\
-1 & \text{, if there are no such } p
\end{cases}.
\]
(3) $Val: \{0, 1\}^n \times \{0, \ldots, 2t - 1\} \rightarrow \{-1, \ldots, t - 1\}$ gets the value of the block with address $i$:

$$Val(X, a) = \begin{cases} \sum_{j=0}^{b-1} x_j^p \text{mod } t & , \text{ where } p = \text{Ind}(X, a) \text{ for } p \geq 0, \\ -1 & , \text{ if Ind}(X, i) < 0 \end{cases}.$$ 

Suppose that we are at the $i$-th step of iteration.

4. Step 1: $\{0, 1\}^n \times \{0, \ldots, 1\} \rightarrow \{-1, t, \ldots, 2t - 1\}$ gets the first part of the $i$th step of iteration:

$$\text{Step}_1(X, i) = \begin{cases} -1 & , \text{ if } \text{Step}_2(X, i - 1) = -1, \\ Val(X, \text{Step}_2(X, i - 1)) + t & , \text{ otherwise} \end{cases}.$$ 

5. Step 2: $\{0, 1\}^n \times \{-1, \ldots, 1\} \rightarrow \{-1, \ldots, t - 1\}$ gets the second part of the $i$th step of iteration:

$$\text{Step}_2(X, i) = \begin{cases} -1 & , \text{ if } \text{Step}_1(X, i) = -1, \\ 2 & , \text{ if } i = -1, \\ Val(X, \text{Step}_1(X, i)) & , \text{ otherwise} \end{cases}.$$ 

Remark that the address of the current block is computed on the previous step.

Function $2-\text{USAF}_t(X)$ is computed as: $2-\text{USAF}_t(X) = \begin{cases} 0 & , \text{ if } \text{Step}_2(X, 1) \leq 0, \\ 1 & , \text{ otherwise} \end{cases}.$

**Theorem 7.** For integer $t = t(n)$, $R_n(L2\text{USAF}_t) \geq t^{t-2}$, where $t$ satisfies $4t(2t + \lceil \log 2t \rceil) < n$.

**Proof.** We pick the partition $\pi = (X_A, X_B) \in \Pi(id)$ such that $X_A$ contains exactly $w$ blocks.

Let $\sigma, \sigma' \in \Psi$ be two different inputs and $\tau$ and $\tau'$ be their corresponding mappings, respectively. We show that the subfunctions $2-\text{USAF}_t|_{\tau}$ and $2-\text{USAF}_t|_{\tau'}$ are different. Let $z \in \{2, \ldots, t - 1\}$ such that $s' = \text{Val}(\nu', z) \neq \text{Val}(\nu, z) = s$.

If $z > 2$, then we choose $\gamma \in \Gamma$ providing that $\text{Val}(\nu, s+t) = 1$, $\text{Val}(\nu', s'+t) = 0$, and $\text{Val}(\nu, r) = \text{Val}(\nu', r) = z$, where $r = \text{Val}(\nu, 2)$. That is,

- $\text{Step}_1(\nu, 0) = \text{Step}_1(\nu', 0) = r$,
- $\text{Step}_2(\nu, 0) = s$ and $\text{Step}_2(\nu', 0) = s'$,
- $\text{Step}_1(\nu, 1) = 2t - 1$ and $\text{Step}_1(\nu', 1) = 2t - 2$, and,
- $\text{Step}_2(\nu, 1) = 1$ and $\text{Step}_2(\nu', 1) = 0$.

Thus, $2-\text{USAF}_t(\nu) = 1$ and $2-\text{USAF}_t(\nu') = 0$.

If $z = 2$, then we choose $\gamma \in \Gamma$ providing that $\text{Val}(\nu, s+t) = 1$ and $\text{Val}(\nu', s'+t) = 0$. That is,

- $\text{Step}_1(\nu, 0) = s$ and $\text{Step}_1(\nu', 0) = s'$,
- $\text{Step}_2(\nu, 1) = 1$ and $\text{Step}_2(\nu', 1) = 0$,
- $\text{Step}_1(\nu, 1) = 2t - 1$ and $\text{Step}_1(\nu', 1) = 2t - 2$, and,
- $\text{Step}_2(\nu, 1) = 1$ and $\text{Step}_2(\nu', 1) = 0$. 

Hence $2\text{-USAF}_{t}(\nu) = 1$ and $2\text{-USAF}_{t}(\nu') = 0$.

Therefore $2\text{-USAF}_{t}|_{\tau}(\gamma) \neq 2\text{-USAF}_{t}|_{\tau}(\gamma)$ and also $2\text{-USAF}_{t}|_{\tau} \neq 2\text{-USAF}_{t}|_{\tau'}$.

Now, we compute $|\Psi|$. For $\sigma \in \Psi$, we can get each value of $\text{Val}(\nu, i)$ for $2 \leq i \leq t-1$. It means $|\Psi| \geq t^{t-2}$ due to Lemma[4]. Therefore, $N^\pi(2\text{-USAF}_{t}) \geq t^{t-2}$, and, by definition of $N(2\text{-USAF}_{t})$, we have $N(2\text{-USAF}_{t}) \geq t^{t-2}. \qed$

**Theorem 8.** There is a 2DFA $A_n$ of size $23t + 2(1 + 3t) \log t + 6$ recognizing $L2\text{-USAF}_{t}$.

**Proof.** The computation of $A_n$ consists of four parts similar to the automaton given in Theorem[6].

In the first part, the block checking procedure uses $2 \log 2t + 2$ states. Computing $\text{Val}(\nu, 2)$ uses $2t$ states and there are $w$ states. So, the size of the first part is $3t + 2 \log 2t + 2$. In the second part, the block checking procedure uses $2 \log 2t + 2$ states and we have $t$ blocks to check the state pairs for each value of $\nu'$, that is $w(2 \log 2t + 2)$ states. Computing $\text{Val}(\nu, \nu')$ uses $2t$ states and also $A_n$ has $t$ states. Therefore, the size of the second part is $5t + 2t \log 2t$. Similarly, we can show that the size of the third part is $5t + 2t \log 2t$. In the fourth part, we use $4t + 2t \log 2t$ states for procedure checking and computing $\text{Val}(\nu, \nu'')$. $A_n$ has also one accept and one reject states. So, the size of the forth part is $4t + 2t \log 2t + 2t$. Thus, the overall size of $A_n$ is $23t + 2(1 + 3t) \log t + 6. \qed$

We can simplify the formula for the number of states for bigger $t$ values, e.g.:

- For $t > 2906$, there is a 2DFA of size $8t \log(t)$ recognizing $L2\text{USAF}_{t}$.
- For $t > 26$, there is a 2DFA of size $11t \log(t)$ recognizing $L2\text{USAF}_{t}$.
- For $t > 3$, there is a 2DFA of size $19t \log(t)$ recognizing $L2\text{USAF}_{t}$.

4. Hierarchies results

Now, we can follow our hierarchies and incomparability results.

**Theorem 9.** Let $d : \mathbb{N} \rightarrow \mathbb{N}$ be a function satisfying $13d + 43 < \sqrt{\pi}$. Then, $2\text{D\$\text{OSIZE}}(d) \subseteq 2\text{D\$\text{OSIZE}}(13d + 43)$.

**Proof.** By Theorem[5], we can follow that $(d + 1)^{d+1} < (d + 3)^{d+1} \leq N(2\text{-SAF}_{d+3})$. Suppose that there is a 2D$\text{A}_{n}^{\Theta}$ with size $d$ computing 2-SAF$_{d+3}$. Then, by Theorem[1], we can have $N(2\text{-SAF}_{d+3}) \leq (d + 1)^{d+1}$, which is a contradiction and so $2\text{-SAF}_{d+3} \notin 2\text{D\$\text{OSIZE}}(d)$.

Moreover, by Theorem[6], we have that 2-SAF$_{d+3}$ is in 2D$\text{SIZE}(13d + 43)$.

Lastly, the relation between $d$ and $n$ can be followed from the definition of 2-SAF$_{d}$ by also taking into account the parameter $(13d + 43)$. For 2-SAF$_{t}$, we have inequality $2t(2t + [\log(2t)]) < n$. For $t \geq 2$, we always have $t \geq [\log(2t)]$. 

Therefore, the following inequality also works $2t(3t) < n$, which gives us that $t < \sqrt{\frac{n}{12}}$.

**Theorem 10.** Let $d : \mathbb{N} \rightarrow \mathbb{N}$ be a function satisfying $121 < 13d(n) + 4 < \sqrt{\frac{n}{6}}$. Then, $2N\Theta\text{SIZE}(\lceil\sqrt{d}\rceil) \subseteq 2N\Theta\text{SIZE}(13d + 4)$.

**Proof.** By Theorem 5 we have $2\log_2 d(d-2) = d^{d-2} \leq N(2\text{-SAF}_d)$. For $d > 9$, we can follow that $2^\sqrt{d+1} < 2\log_2 d(d-2)$.

Suppose that there is a $2NA_n^\theta$ with size $\lceil\sqrt{d}\rceil$ computing $2\text{-SAF}_d$. Then, by Theorem 2, we can have $N(2\text{-SAF}_d) \leq 2^{1\lceil\sqrt{d}\rceil+1}$, which is a contradiction and so $2\text{-SAF}_d \not\in 2N\Theta\text{SIZE}(\lceil\sqrt{d}\rceil)$.

Moreover, by Theorem 6, we have that $2\text{-SAF}_d$ is in $2D\text{SIZE}(13d + 4)$.

The relation between $d$ and $n$ can be followed similar to the previous proof and the condition $d > 9$. \hfill $\Box$

**Theorem 11.** Let $d : \mathbb{N} \rightarrow \mathbb{N}$ be a function satisfying $1330 < \lceil 11d \log d \rceil < \sqrt{\frac{n}{12}}$. Then, $2D\text{FA}\text{SIZE}(d-3) \not\subseteq 2D\text{FA}\text{SIZE}(\lceil 11d \log d \rceil)$.

**Proof.** By Theorem 8, we have $(d-2)^{d-2} < d^{d-2} \leq N(2\text{-USAF}_d)$ and so we can also have $(d-2)^{d-2} < R_n(2\text{-USAF}_d)$.

Suppose that there is a $2DFA$ with size $d-3$ recognizing $2\text{-USAF}_d$. Then, by Corollary 1, we can have $R_n(2\text{-USAF}_d) \leq (d-2)^{d-2}$, which is a contradiction and so $2\text{-USAF}_d \not\in 2D\text{FA}\text{SIZE}(d-3)$.

By Theorem 10 and for $d > 26$, we have $2\text{-USAF}_d \in 2D\text{FA}\text{SIZE}(\lceil 11d \log d \rceil)$.

The relation between $d$ and $n$ can be followed by the definition of $2\text{-USAF}_t$: $4t(2t + \lceil \log(2t) \rceil) < n$. For $t \geq 2$, $t \geq \lceil \log(2t) \rceil$ and so we can also use the inequality $4t(3t) < n$, which implies $t \leq \sqrt{\frac{n}{12}}$. \hfill $\Box$

**Theorem 12.** Let $d : \mathbb{N} \rightarrow \mathbb{N}$ be a function satisfying $30 < 13d(n) + 4 < \sqrt{\frac{n}{6}}$ and $T \geq 256$ be the expected running time to finish computation. Then $2B\Theta\text{SIZE}\left(\left\lfloor\frac{\sqrt{d}}{32\log T}\right\rfloor\right) \not\subseteq 2B\Theta\text{SIZE}(13d + 4)$, where the error bound is at least $\frac{1}{3}$ for the classes.

**Proof.** By using the facts given in the two above proofs, for $d > 9$, we know that $2\text{-USAF}_d \not\in 2B\Theta\text{SIZE}(\lceil\sqrt{d}\rceil)$ and $2\text{-USAF}_d \in 2D\text{FA}\text{SIZE}(\lceil 11d \log d \rceil)$.

By Theorem 7, we have $(d-2)^{d-2} < R_n(2\text{-USAF}_d)$ and so we can also have $(d-2)^{d-2} < R_n(2\text{-USAF}_d)$.

Moreover, by Theorem 6, we have that $2\text{-USAF}_d$ is in $2D\text{FA}\text{SIZE}(13d + 4)$.

By using the fact that $4t(2t + \lceil \log(2t) \rceil) < n$. For $t \geq 2$, $t \geq \lceil \log(2t) \rceil$ and so we can also use the inequality $4t(3t) < n$, which implies $t \leq \sqrt{\frac{n}{12}}$. \hfill $\Box$
Proof. By using Theorem 5, we can obtain the following for \( d > 2 \).

\[
\left( \frac{32 \log(T) \sqrt{d}}{32 \log(T)} \right)^2 \left( \frac{32 \log(T) d}{32 \log(T)} \right)^{d-2} = d^{d-2} \leq N(2 \text{-SAF}_d).
\]

Then \( 2 \text{-SAF}_d \) cannot be solved by a \( 2PA_n^\Theta \) with size \( \lfloor \sqrt{\frac{d}{32 \log T}} \rfloor \). Otherwise, by Corollary 1, we obtain the following contradiction.

\[
N(2 \text{-SAF}_d) < \left( \frac{32 \log T}{32 \log T} \right)^{\left( \lfloor \sqrt{\frac{d}{32 \log T}} \rfloor + 1 \right)^2}.
\]

By Theorem 6, we have that \( 2 \text{-SAF}_d \) is in \( 2 \text{DSIZE}(13d + 4) \).

The relation between \( d \) and \( n \) can be followed similar to the previous proofs and the condition \( d > 2 \). \( \square \)

5. Incomparability results

In this section, we give evidences how shuffling can reduce the size of models. For this purpose, we use the well-known \( \text{Equality function} \)

\[
\text{EQ}(X) = \bigvee_{0}^{\lfloor n/2 \rfloor - 1} x_i = x_{i+\lfloor n/2 \rfloor}.
\]

The nice property of \( \text{EQ}(X) \) for our purpose is that \( N^\text{id}(\text{EQ}) = 2^{\lfloor n/2 \rfloor} \) and \( N^\theta(\text{EQ}) \leq 4 \) for \( \theta = (0, \lfloor n/2 \rfloor, 1, \lfloor n/2 \rfloor + 1, \ldots) \). Therefore, we can follow that \( \text{EQ} \in 2 \text{DThetaSIZE}(4) \).

Suppose that there is a \( 2NA_n \) with size \( \lfloor \sqrt{\frac{n}{2}} \rfloor - 2 \) solving \( \text{EQ} \). Then, by Theorem 2 we have

\[
N^\text{id}(\text{EQ}) \leq 2^\lfloor \sqrt{\frac{n}{2}} \rfloor - 1 \rfloor^2,
\]

which is a contradiction. Therefore, \( \text{EQ} \notin 2 \text{NSIZE}(\lfloor \sqrt{\frac{n}{2}} \rfloor - 2) \). We use this lower bound also for deterministic case.

Let \( d = d(n) \) be a function such that \( d \leq \lfloor \sqrt{\frac{n}{2}} \rfloor - 2 \), and \( d' \) be a function such that \( 4 \leq d' < d \). Then, due to the fact given for function \( \text{EQ} \), we can immediately follow that

\[
2 \text{DThetaSIZE}(d') \notin 2 \text{DSIZE}(d) \quad \text{and} \quad 2 \text{ThetaSIZE}(d') \notin 2 \text{NSIZE}(d).
\]

On the other hand, due to the results given in Section 4, the shuffling classes cannot contain the non-shuffling classes under certain size bounds, which leads us to our incomparability results.

**Theorem 14.** As a further restriction, if \( 13d + 43 < \sqrt{\frac{n}{6}} \), then for \( 94 \leq 13d' + 43 < d \), the classes \( 2 \text{DThetaSIZE}(d') \) and \( 2 \text{DSIZE}(d) \) are not comparable.
Proof. By the proof of Theorem 9, we know that $2D\Theta \text{SIZE}(d')$ does not have a function that is in $2D\text{SIZE}(13d' + 43)$. Since $13d' + 43 < d$, this function is also in $2D\text{SIZE}(d)$.

\begin{flushright}
\Box
\end{flushright}

\textbf{Theorem 15.} As a further restriction, if $121 < 13d + 4 < \sqrt{\frac{n}{6}}$, then for $58 \leq 13d' + 4 < d$, the classes $2N\Theta \text{SIZE}(\lfloor \sqrt{d'} \rfloor)$ and $2N\text{SIZE}(d)$ are not comparable.

Proof. By the proof of Theorem 10, we know that $2N\Theta \text{SIZE}(\lfloor \sqrt{d'} \rfloor)$ does not have a function that is in $2D\text{SIZE}(13d' + 4)$. Since $13d' + 4 < d$, this function is also in $2N\text{SIZE}(d)$.

\begin{flushright}
\Box
\end{flushright}

Any $2PA_n$ with size $d$ satisfying $(32d \log T) (d+1)^2 < 2^{\frac{n}{32}}$ cannot solve EQ due to Theorem 4. We bound $T < 2^{2d}$. Then, for $d \geq 2$, we can follow

\[(32d \log T) (d+1)^2 < \left(2^{5 \log_2 2} 2^d \right) (d+1)^2 \leq \frac{2d^3}{2} < \frac{2 \sqrt{d}}{2} < 2^{\frac{d}{3}}.\]

Then, for $4 \leq d' \leq d$ where $d < \sqrt{\frac{n}{2}}$, $2B\Theta \text{SIZE}(d') \notin 2B\text{SIZE}(d)$. Now, we can state the result for probabilistic case similar to the other cases.

\textbf{Theorem 16.} As a further restriction, if $30 < 13d + 4 < \sqrt{\frac{n}{2}}$, then for $4 \leq 13d' + 4 < d$ and $256 \leq T < 2^{2d'}$ be the expected running time to finish computation, the classes $2B\Theta \text{SIZE}\left(\lfloor \frac{\sqrt{d}}{32 \log T} \rfloor\right)$ and $2B\text{SIZE}(d)$ are not comparable.

Acknowledgements. The authors thank to A. Ambainis, A. Rivosh, K. Prūsis, and J. Vihrovs for useful discussions and comments.

The work is partially supported by ERC Advanced Grant MQC. The work is performed according to the Russian Government Program of Competitive Growth of Kazan Federal University

\begin{thebibliography}{99}

[1] F. Ablayev, A. Gainutdinova, K. Khadiev, and A. Yakaryılmaz. Very narrow quantum OBDDs and width hierarchies for classical OBDDs. \textit{Lobachevskii Journal of Mathematics}, 37(6):670–682, 2016.

[2] Farid Ablayev, Aida Gainutdinova, Kamil Khadiev, and Abuzer Yakaryılmaz. Very narrow quantum OBDDs and width hierarchies for classical OBDDs. In \textit{DCFS}, volume 8614 of \textit{LNCS}, pages 53–64. Springer, 2014.

[3] Farid Ablayev and Kamil Khadiev. Extension of the hierarchy for $k$-OBDDs of small width. \textit{Russian Mathematics}, 53(3):46–50, 2013.

[4] Farid Ablayev, Andris Ambainis, Kamil Khadiev and Aliya Khadieva. Lower Bounds and Hierarchies for Quantum Memoryless Communication Protocols and Quantum Ordered Binary Decision Diagrams with Repeated Test \textit{In SOFSEM 2018, LNCS, 2018} [arXiv:1703.05015]

\end{thebibliography}
[5] Farid M. Ablayev and Marek Karpinski. On the power of randomized branching programs. In *ICALP*, volume 1099 of *LNCS*, pages 348–356. Springer, 1996.

[6] Kaspars Balodis. One alternation can be more powerful than randomization in small and fast two-way finite automata. In *FCT*, volume 8070 of *LNCS*, pages 40–47, 2013.

[7] David A. Mix Barrington, Howard Straubing, and Denis Thérien. Non-uniform automata over groups. *Information and Computation*, 89(2):109–132, 1990.

[8] Beate Bollig, Martin Sauerhoff, Detlef Sieling, and Ingo Wegener. Hierarchy theorems for kOBDDs and kIBDDs. *Theoretical Computer Science*, 205(1):45–60, 1998.

[9] Marek Chrobak. Finite automata and unary languages. *Theoretical Computer Science*, 47(0):149 – 158, 1986.

[10] Cynthia Dwork and Larry Stockmeyer. A time complexity gap for two-way probabilistic finite-state automata. *SIAM Journal on Computing*, 19(6):1011–1123, 1990.

[11] Viliam Geffert, Bruno Guilhon, and Giovanni Pighizzini. Two-way automata making choices only at the endmarkers. In *Language and Automata Theory and Applications*, volume 7183 of *LNCS*, pages 264–276. Springer Berlin Heidelberg, 2012.

[12] Markus Holzer. Multi-head finite automata: data-independent versus data-dependent computations. *Theoretical Computer Science*, 286(1):97–116, 2002.

[13] Christos A. Kapoutsis. Removing bidirectionality from nondeterministic finite automata. In *MFCS*, volume 3618 of *LNCS*, pages 544–555. Springer, 2005.

[14] Christos A. Kapoutsis. Size complexity of two-way finite automata. In *Developments in Language Theory*, volume 5583 of *LNCS*, pages 47–66. Springer, 2009.

[15] Christos A. Kapoutsis. Nondeterminism is essential in small two-way finite automata with few reversals. *Information and Computation*, 222(0):208 – 227, 2013. 38th International Colloquium on Automata, Languages and Programming (ICALP 2011).

[16] Christos A. Kapoutsis, Richard Královic, and Tobias Mömke. Size complexity of rotating and sweeping automata. *Journal of Computer and System Sciences*, 78(2):537–558, 2012.

[17] K. Khadiev. Width hierarchy for k-OBDD of small width. *Lobachevskii Journal of Mathematics*, 36(2):178–183, 2015.

[18] K. Khadiev. On the hierarchies for deterministic, nondeterministic and probabilistic ordered read-k-times branching programs. *Lobachevskii Journal of Mathematics*, 37(6):682–703, 2016.

[19] Kamil Khadiev and Rishat Ibrahimov. Width hierarchies for quantum and classical ordered binary decision diagrams with repeated test. In: *Proceedings of the Fourth Russian Finnish Symposium on Discrete Mathematics*. No. 26 in *TUCS Lecture Notes*, Turku Centre for Computer Science (2017).

[20] Kamil Khadiev and Aliya Khadieva. Reordering method and hierarchies for quantum and classical ordered binary decision diagrams. In Pascal Weil, editor, *Computer Science Theory and Applications: 12th International Computer Science Symposium in Russia, CSR 2017*, Kazan, Russia, June 8-12, 2017, Proceedings, volume 10304 of *LNCS*, pages 162–175. Springer International Publishing, Cham, 2017.
[21] Hing Leung. Tight lower bounds on the size of sweeping automata. *Journal of Computer and System Sciences*, 63(3):384 – 393, 2001.

[22] Noam Nisan and Avi Widgerson. Rounds in communication complexity revisited. In *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pages 419–429. ACM, 1991.

[23] M.O. Rabin and D. Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3:114–125, 1959.

[24] William J. Sakoda and Michael Sipser. Nondeterminism and the size of two way finite automata. In *STOC’78: Proceedings of the tenth annual ACM symposium on Theory of computing*, pages 275–286, 1978.

[25] John C. Shepherdson. The reduction of two–way automata to one-way automata. *IBM Journal of Research and Development*, 3:198–200, 1959.

[26] Moshe Y. Vardi A Note on the Reduction of Two-Way Automata to One-Way Automata. *Information Processing Letters*, 30(5):261–264, 1989.

[27] Ingo Wegener. *Branching Programs and Binary Decision Diagrams*. SIAM, 2000.