Electronic Colloquium on Computational Complexity, Report No. 4 (1998)

On the Power of Randomized Ordered Branching Programs*

Farid Ablayev † Marek Karpinski‡

Abstract

We introduce a model of a *randomized branching program* in a natural way similar to the definition of a randomized circuit. We exhibit an explicit boolean function $f_n : \{0,1\}^n \to \{0,1\}$ for which we prove that:

1) f_n can be computed by a polynomial size randomized ordered read-once branching program with a small one-sided error.

2) f_n cannot be computed in polynomial size by any nondeterministic ordered *read-k-times* branching program for any $k = o(n/\log n)$. The required nondeterministic size is $2^{\Omega(n/k)}$.

1 Preliminaries

Various models of branching programs have recently been found very useful in the field of digital design and hardware verification. They were introduced in [L59] and [M76], and studied extensively during the last decade (see, e.g., [W87] and [W94]) under various computational efficiency aspects. For the known lower bounds results we refer also to [R91], and [BRS93].

Developments in the field of digital design and verification have led to the restricted forms of branching programs. A most common model used for verifying circuits is an

*A preliminary version of this paper appeared in [AK96]

†Dept. of Computer Science University of Bonn. Email: ablayev@cs.uni-bonn.de. Visiting from University of Kazan. Research partially supported by the Volkswagen-Stiftung and Russia Fund for Basic Research 96-01-01962

‡Dept. of Computer Science University of Bonn, and International Computer Science Institute, Berkeley, California. Email: marek@cs.uni-bonn.de. Research partially supported by DFG Grant KA 673/4-1, by the ESPRIT BR Grants 7097, and EC-US 030, by the Volkswagen-Stiftung, and by the Max-Planck Research Prize.

ISSN 1433-8092

ordered binary decision diagram (OBDD), also called an *ordered read-once branching program* (see [B92], [W94]). It has turned out recently that many important functions cannot be computed by read-once branching programs in polynomial size (see [B92], [SS93], and [P95]).

In this paper we define a notion of a randomized branching program in a natural way similar to the definition of a randomized circuit. Our goal is to show that randomized computation with a small error probability for ordered read-once polynomial branching programs can be (exponentially) more efficient than the deterministic one. We exhibit an explicit boolean function f_n which is *easy* for ordered read-once randomized branching programs, but it is *hard* for ordered read-k-times nondeterminstic branching programs. Note that for read-poly-times randomized branching programs, the error probability can be made less than 2^{-n} by repeating computations and taking majority of the results. This means the resulting branching programs can be made (nonuniformly) deterministic (cf. [A78], [AB84] and [S97]).

The results presented in this paper are generalization of the results presented in [AK96] for the case of boolean functions.

A *deterministic* branching program P for computing a function $g : \{0,1\}^n \to \{0,1\}$ is a directed acyclic multi-graph with a distinguished source node s and a distinguished (accepting) sink node t. The outdegree of each nonsink node is exactly 2, and the two outgoing edges are labeled by $x_i = 0$ and $x_i = 1$ for a variable x_i associated with this node. Call such a node an x_i-node. The label "$x_i = \delta$" indicates that only inputs satisfying $x_i = \delta$ may follow this edge in the computation. The branching program P computes a function g in the obvious way: for each $\sigma \in \{0,1\}^n$ we let $g(\sigma) = 1$ iff there is a directed s-t path starting in the source s and leading to the accepting node t such that all labels $x_i = \sigma_i$ along this path are consistent with $\sigma = \sigma_1, \sigma_2, \ldots, \sigma_n$.

A branching program becomes *nondeterministic* if we allow "guessing nodes" that is nodes with two outgoing edges being unlabeled. Unlabeled edges allow all inputs to proceed to the next node. A *nondeterministic* branching program P computes a function g, in the obvious way; that is, $g(\sigma) = 1$ iff there exists (at least one) computation over σ starting in the source node s and leading to the accepting node t.

Define a *randomized* branching program as a branching program which has in addition to its standard (deterministic) inputs especially designed random ("coin-toss") input nodes. When these random inputs are chosen from the uniform distribution, the output of the branching program is a random variable.

We say that a randomized branching program (a,b)-*computes* a function g if it outputs 1 with the probability at most a for an input x such that $g(x) = 0$, and it outputs 1 with the probability at least b for an input x such that $g(x) = 1$. The randomized branching program computes the function g with a one-sided ε-error if it $(\varepsilon, 1)$-*computes* the function g.

For a branching program P, we define size of P size(P) (complexity of P) as the number of its internal nodes of P.

For a *randomized* branching program P, size(P) is the sum of numbers of its internal and random nodes.

The size of a nondeterminstic branching program is the number of its internal nodes (without "guessing" nodes).

A *read-once* branching program is a branching program in which no variable appears more than once on any computation path. An *ordered read-once* branching program is a read-once branching program which respects a fixed ordering π of variables, i.e., if an edge leads from an x_i-node to an x_j-node, the condition $\pi(i) < \pi(j)$ has to be fulfilled.

A *read-k-times branching program* is a branching program with the property that no input variable x_i appears more than k times on any consistent computation path in the program (a path is *consistent* if for all i the labels "$x_i = 0$" and "$x_i = 1$" do not both appear on the path).

A *syntactic read-k-times branching program* [BRS93] is a branching program with the property that no input variable x_i appears more than k times on any path (*consistent or not*) in the program. (Non-syntactic read-k-times devices can be more powerful than the syntactic ones [J94].)

An *ordered read-k-times branching program* is a read-k-times branching program which is partitioned into k layers such that each layer is an ordered read-once respecting the same ordering π. In [BSSW94] it is proved that deterministic ordered read-$(k+1)$-times branching programs are more powerful than deterministic ordered read-k-times branching programs. Namely the classes of functions computed by deterministic polynomial size ordered read-k-times branching programs form a proper hierarchy for $k = o(n^{1/2}/\log^2 n)$.

We will use a slightly more general notion of a *non-syntactic* read-k-times branching program in this paper. We call a branching program *read^A-k-times* if it has the following property: no input variable x_i appears more than k times on any consistent *accepting* computation s–t path in the program. We call a branching program *read^R-k-times* if it has the following property: no input variable x_i appears more than k times on any consistent *rejecting* computation path in the program. Clearly, a branching program is read-k-times if it is *read^A-k-times* and *read^R-k-times* .

Note that one can think of each internal node of a branching program as a state of the computation. This observation is important for the lower bound proofs in our paper. Note also that this point of view is essential for investigating the amount of space necessary for computing certain functions. Restricted models of branching programs are useful for investigating the time-space tradeoffs. We can think of read-k-time ($k \geq 1$) restrictions as a restriction on time, say time $\leq kn$ (see, e.g., survey [B93]). This approach draws a time-space tradeoff point of view to our results. Recent results on the general lower bounds on randomized space and time can be found in [A94] and [FK95].

2 The Functions

We introduce now the following two boolean functions.

First, we define the boolean function f_n of $n = 4l$ variables as follows. For a sequence $\sigma \in \{0,1\}^{4l}$ we call the odd bits "type" bits, and the even bits "value" bits. We say that even bit $\sigma_i, i \in \{2,4,\ldots,4l\}$, has "type" 0 (1) if corresponding odd bit σ_{i-1} is 0 (1). For

a sequence $\sigma \in \{0,1\}^{4l}$, denote by σ^0 (σ^1) a *subsequence* of σ that consists of all even bits of type 0 (1).

We define the boolean function $f_n : \{0,1\}^n \to \{0,1\}$ as follows: $f_n(\sigma) = 1$ iff $\sigma^0 = \sigma^1$.

The function $PERM$ [KMW88], [J89] is defined on $n \times n$ matrix $X = (x_{ij})_{1 \le i,j \le n}$ of boolean variables. For an input $\sigma \in \{0,1\}^{2n}$ $PERM(\sigma) = 1$ if and only if σ is a permutation matrix, i.e., if each row and each column of σ contains exactly one 1 entry. It is known (see [KMW88], [J89]) that *nondeterministic* read-once branching programs for $PERM$ have exponential size.

3 Results

In Theorem 1 below we present a randomized read-once branching program of polynomial size for the function f_n. Our design uses a variant of a *fingerprint* technique (cf., e.g., [F79], [KR87] and [GKPR96]).

Theorem 1 *The function f_n can be computed with one sided $\varepsilon(n)$-error by a randomized read-once ordered branching program of size*

$$O\left(\frac{n^6}{\varepsilon^3(n)} \log^2 \frac{n}{\varepsilon(n)}\right).$$

Proof: We design a randomized read-once ordered branching program P for f_n that works in two phases as follows.

Phase 1. (randomized). Choose $d(l)$ to be some function in $O(l)$, s.t. $d(l) > 2l$. P randomly selects a prime number p from the set $Q_{d(l)} = \{p_1, p_2, \ldots, p_{d(l)}\}$ of the first $d(l)$ prime numbers.

P selects a prime number p in the following way. P uses $t = \lceil \log d(l) \rceil$ *random* input variables $y_1, y_2, \ldots, y_t$ where $y_i \in \{0,1\}$ and $Prob(y_i = 1) = Prob(y_i = 0) = 1/2$. The branching program P reads its random inputs in the fix order $y_1, y_2, \ldots, y_t$. A sequence $y = y_1 y_2 \ldots y_t$ is interpreted as the binary notation of a number $N(y)$. P selects i-th prime number $p_i \in Q_{d(l)}$ iff $N(y) = i \bmod d(l)$.

Phase 2. (deterministic). Let $\sigma \in \{0,1\}^{4l}$ be an input sequence. The sequences σ^0 and σ^1 are interpreted as the binary notations of numbers $N(\sigma^0)$ and $N(\sigma^1)$ respectively. P reads an input sequence $x = \sigma$ in the order $x_1, x_2, \ldots, x_{4l}$.

Now P does the following.

a) it verifies if $|\sigma^0| = |\sigma^1| = l$,

b) it computes modulo p the numbers $N(\sigma^0)$ and $N(\sigma^1)$ ($a = N(\sigma^0) \bmod p$ and $b = N(\sigma^1) \bmod p$) in the following way. At the beginning of the computation, P sets $a := 0$ and $b := 0$. Next, if P reads the j-th symbol $\sigma_j \in \{0,1\}$ of the sequence $\sigma^0 = \sigma_1, \sigma_2, \ldots$ (respectively j-th input symbol $\sigma'_j \in \{0,1\}$ of the sequence $\sigma^1 = \sigma'_1, \sigma'_2, \ldots$) then $a := a + \sigma_j 2^j \bmod p$ (respectively $b := b + \sigma'_j 2^j \bmod p$).

Let α and β be the first parts of length t and k, respectively, of the subsequences σ^0 and σ^1 that were tested along the path from the source to the internal node (state) v.

For $p \in Q_{d(l)}$, denote by S_p a deterministic subprogram of P that carries out the deterministic part of the computations of the *phase 2* with a prime p. For the realization of the *phase 2* it is sufficient to store in a state v four numbers: $t, k \in \{0, 1, \ldots, l\}$, $a = N(\alpha) (\mathrm{mod}\, p)$, $b = N(\beta) (\mathrm{mod}\, p)$, and the *type* bit $c \in \{0, 1\}$.

If $|\sigma^0| \neq |\sigma^1|$ then P outputs the correct answer with probability 1.

Consider now the case $|\sigma^0| = |\sigma^1|$.

If $N(\sigma^0) = N(\sigma^1) (\mathrm{mod}\, p)$ then P outputs 1 else P outputs 0.

From the description of P it follows that if $N(\sigma^0) = N(\sigma^1)$ then P outputs the correct answer with probability 1. If $N(\sigma^0) \neq N(\sigma^1)$ then it can happen that $N(\sigma^0) = N(\sigma^1) (\mathrm{mod}\, p)$ for some $p \in Q_{d(l)}$. In this case P makes an error.

For $x = \sigma$ we have $|N(\sigma^0) - N(\sigma^1)| \leq 2^l < p_1 p_2 \cdots p_l$ where $p_1, p_2, \ldots, p_l$ are the first l prime numbers. This means that in the case when $N(\sigma^0) \neq N(\sigma^1)$ the probability $\varepsilon(n)$ of an error of P on the input $x = \sigma$ is no more than $2l/d(l)$ (no more than $l/d(l)$ if t is a power of 2).

The size of S_p is at most

$$\sum_{i=1}^{4l} 2(n+1)^2 p^2.$$

The size of P is at most

$$2^{t+1} - 1 + \sum_{p \in Q_{d(l)}} size(S_p).$$

The value of the i-th prime is $O(i \log i)$. Therefore, the above upper bounds for the $size(S_p)$, and $size(P)$ yield

$$size(P) = O(n^3 d^3(l) \log^2 d(l)) = O\left(\frac{n^6}{\varepsilon^3(n)} \log^2 \frac{n}{\varepsilon(n)}\right).$$

∎

Using the proof technique of Theorem 1 the following result was also proved in [S97].

Theorem 2 *([S97]) The function $PERM$ can be computed with a one sided $\varepsilon(n)$-error by randomized read-once ordered branching program of size*

$$O\left(\varepsilon(n)^{-2} n^5 \log^3 n\right).$$

∎

Below we prove an exponential lower bound for the complexity of computing the function f_n by nondeterministic ordered read-k-times branching programs. We use a proof method based on a two-way communication game (cf., [Y79], [KN97]). We present this

method in the lemma below for a more general notion of ordering variables than the traditional one.

Note that the methods based on a communication game were also used in [K91], and later in [BSSW94] for proving lower bounds in a context of branching programs.

Definition 1 *We call a read-once branching program π-weak-ordered if its respects a partition π of the variables into two parts X_1 and X_2 such that if an edge leads from an x_i-node to an x_j-node, where $x_i \in X_t$ and $x_j \in X_m$, then the condition $t \leq m$ has to be fulfilled.*

We call a read-k-times branching program read-k-times π-weak-ordered if it is partitioned into k layers such that each layer is π-weak-ordered read-once respecting the same partition π of variables in each layer.

A π-weak-ordering of variables of a branching program P means that if some input $x_i \in X_2$ is tested by P, then along the remaining part of the computation path no variables from X_1 can be tested.

We call a branching program P a read-k-times weak-ordered if it is read-k-times π-weak-ordered for some ordered partition π of the set of variables of P into two sets.

From the definition it follows that if a read-k-times branching program is ordered then it is weak-ordered.

Let $g : \{0,1\}^n \to \{0,1\}$ be a boolean function. Let π be a partition of the set of variables X of g into two parts X_1 and X_2. Consider a k-round nondeterminstic communication computation. We use a standard model of a k-round nondeterminstic communication computation for boolean functions, cf., e.g., [KN97]. For a valuation σ of x, two players A and B receive respectively, parts of σ: $\sigma_A \in X_1$ and $\sigma_B \in X_2$. A k-round protocol specifies for each input σ, a sequence of k binary strings (messages) $m_1, m_2, \ldots, m_k$ to be sent alternatively between the players such that at the end one of the players accepts (outputs 1) or rejects (outputs 0) an input σ. Call $m = m_1 m_2 \ldots m_k$ a full message. The player A is always the first one to send a message. We say that a k-round nondeterminstic communication protocol ϕ computes a function g with $g(\sigma) = 1$ iff there exists an accepting computation of ϕ on an input σ. The complexity of a k-round protocol is $\max|m|$ (here $|m|$ is the length of m) over all inputs $\sigma \in \{0,1\}^n$. Denote by $NC_{k,\pi}(g)$ the complexity of the best k-round nondeterministic communication protocol that computes g.

Define the *accepting* (*rejecting*) complexity of a k-round protocol as $|m|$ (m is a full message) maximized over all inputs $\sigma \in g^{-1}(1)$ ($\sigma \in g^{-1}(0)$). Denote by $NC^A_{k,\pi}(g)$ the accepting complexity of the best k-round nondeterministic communication protocol that computes g and by $NC^R_{k,\pi}(g)$ the rejecting complexity of the best k-round nondeterministic communication protocol that computes g.

Lemma 1 *1. Let P be a nondeterministic read-k-times π-weak-ordered branching program that computes a function $g : \{0,1\}^n \to \{0,1\}$. Then*

$$size(P) \geq 2^{(NC_{2k-1,\pi}(g)-1)/(2k-1)}.$$

2. Let P be a nondeterministic read^A-k-times π-weak-ordered branching program that computes a function $g : \{0,1\}^n \to \{0,1\}$. Then

$$size(P) \geq 2^{(NC^A_{2k-1,\pi}(g)-1)/(2k-1)}.$$

3. Let P be a nondeterministic read^R-k-times π-weak-ordered branching program that computes a function $g : \{0,1\}^n \to \{0,1\}$. Then

$$size(P) \geq 2^{(NC^R_{2k-1,\pi}(g)-1)/(2k-1)}.$$

Proof: Case 1. Consider the following communication game with two players A and B for computing a function g. Players A and B have a copy of P. In order to compute g on the set of inputs $\{0,1\}^n$, A and B *communicate* with each other in $(2k-1)$ rounds by sending messages in each round according to the following protocol ϕ. Player A is the first to send a message. Let $\sigma \in \{0,1\}^n$ be a valuation of x. For each i, $1 \leq i \leq k-1$, the communication protocol ϕ on obtaining σ, simulates computation on the i-th layer of P by two communication rounds $2i-1$ and $2i$. The output is produced by the player B.

First round: Player A starts simulation of P on his part σ_A of an input σ from the source of P. Let v_1 be a node which is reachable by P on σ_A from the source during this simulation. Player A sends a node v_1 (binary code of v_1) to B.

Second round: Player B on obtaining message v_1 from A starts its simulation of P on his part σ_B of an input σ from node v_1. Let v_2 be a node which is reachable by P on σ_B from v_1 during this simulation. Player B sends a node v_2 to A.

Last round (round $2k-1$): Player A on obtaining message v_{2k-2} from B starts its computation from a node v_{2k-2} on his part σ_A of an input σ. Let v_{2k-1} be a node which is reachable by P on σ_A from a node v_{2k-2} during this simulation. Player A sends a node v_{2k-1} to B. Player B on obtaining v_{2k-1} starts its part of simulation of P from v_{2k-1} on σ_B and then outputs the result of the computation.

The full message that A and B have exchanged during the computation is $m = v_1 v_2 \ldots v_{2k-1}$.

Denote by M_i the set of all internal nodes which can be sent on the i-th round by player A to B if i is odd (by player B to A if i is even) during the computation on $\{0,1\}^n$. Denote $d_i = |M_i|$. Using this notation, it follows that the number of all full messages that can be exchanged over the inputs from $\{0,1\}^n$ according to protocol ϕ is no more than $\prod_{i=1}^{2k-1} d_i$.

The number of full messages used by ϕ must be greater than equal to $2^{NC_{2k-1,\pi}(g)-1}$, and therefore,

$$\prod_{i=1}^{2k-1} d_i \geq 2^{NC_{2k-1,\pi}(g)-1}.$$

The lower bound of our lemma follows from this inequality.

For $d = \max\{d_i : i \in \{1,2,\ldots,2k-1\}\}$ we have

$$d^{2k-1} \geq \prod_{i=1}^{2k-1} d_i \geq 2^{NC_{2k-1,\pi}(g)-1}$$

and hence

$$d \geq 2^{(NC_{2k-1,\pi}(g)-1)/(2k-1)}.$$

Proofs of cases 2 and 3 of Lemma 1 are similar to the case 1. The only difference is that instead of the set $\{0,1\}^n$, one uses the set $g^{-1}(1)$ of "one" inputs of g (case 2), and the set $g^{-1}(0)$ of "zero" inputs of g (case 3), respectively. ∎

Theorem 3 *Any nondeterministic ordered read^A-k-times branching program that computes function f_n has size at least $2^{(n/4-1)/(2k-1)}$.*

Proof: Let P be an ordered read-k-times branching program (with an ordering τ of variables) which computes the function f_n. For an ordering τ denote by $\tau^0 = \{i_1, i_2, \ldots, i_l\}$ a subsequence of τ that consists of the first l even numbers of τ. Respectively, denote by $\tau^1 = \{j_1, j_2, \ldots, j_l\}$ a subsequence of τ that consists of the last l even numbers of τ.

Consider the partition π of variables X of f_n into two parts X_1 and X_2, such that X_1 contains all variables with indexes from τ^0, and X_2 contains all variables with indexes from τ^1. Clearly P is read-k-times and π-weak-ordered.

For the function f_n we have

$$NC^A_{t,\pi}(f_n) \geq l = n/4 \tag{1}$$

for $t \geq 1$. To prove the inequality (1) we consider an arbitrary t-round nondeterministic communication protocol ϕ that computes the function f_n and respects the partition π of the inputs.

Call a sequence $\sigma \in f_n^{-1}(1)$ τ-*hard* if all its "value" bits σ_i, $i \in \tau^0$, are of "type" 0 and all its "value" bits σ_j, $j \in \tau^1$, are of "type" 1. Denote

$$X^\tau = \{\sigma \in \{0,1\}^{4l} : \sigma \text{ is } \tau\text{-hard}\}.$$

We call X^τ a τ-hard set of inputs.

Let $am(\sigma)$ be a set of messages of accepting computations of protocol ϕ on an input $\sigma \in X^\tau$. Then using "crossing-sequence" argument we have that for an arbitrary distinct pair of inputs $\sigma, \sigma' \in X^\tau$, it holds that $am(\sigma) \cap am(\sigma') = \emptyset$. Clearly $|X^\tau| = 2^l$. This means that the protocol ϕ should use at least 2^n different full messages during the computation over the inputs from a τ-hard set X^τ. From the definition of accepting communication complexity we have that the communication complexity of the protocol ϕ is at least n. Inequality (1) will follow by considering the best possible protocol.

From the lower bound for $NC^A_{t,\pi}(f_n)$ above, and Lemma 1 we have

$$size(P) \geq 2^{(l-1)/(2k-1)} = 2^{(n/4-1)/(2k-1)}.$$

∎

We formulate now following Corollaries.

Corollary 1 *f_n cannot be computed by nondeterministic ordered $read^A$-k-times branching programs in polynomial size for $k = o(n/\log n)$.*

Corollary 2 *f_n cannot be computed by nondeterministic ordered read-k-times branching programs in polynomial size for $k = o(n/\log n)$.*

Note that from Theorem 1 it follows that the complement of function f_n has a polynomial nondeterministic ordered read-once branching program. Below we give a more straightforward proof of this fact.

Property 1 *The complement $\neg f_n$ can be computed by nondeterminstic ordered read-once branching programs of polynomial size.*

Proof: The function $\neg f_n$ is OR of n functions $G_i(x_1, \ldots, x_{4l})$, $1 \leq i \leq n$, where $G_i(x_1, \ldots, x_{4l}) = 1$ if and only if 1) $|\sigma^0| \neq |\sigma^1|$ or 2) if $|\sigma^0| = |\sigma^1|$ then i-th even bit of type 0 is not equal to the i-th even bit of type 1. Clearly, $G_i(x_1, \ldots, x_{4l})$ has a polynomial size read once branching program for the ordering $\tau = \{x_1, x_2, \ldots, x_{4l}\}$. ∎

4 Concluding Remarks

It is known that the complexity class P—OBDD of boolean functions computed by deterministic ordered read-once branching programs of polynomial size is incomparable with AC^0 (class of boolean functions computed by polynomial size unbounded fanin and constant depth boolean circuits): $PARITY$ which is not in AC^0 has linear size ordered read-once branching programs (cf. [KN97]), and $PERM$ function is hard for ordered read-once branching programs but it is in AC^0 (see the property below). We prove now that the complexity class BPP—OBDD of functions computed by randomized ordered read-once branching programs of polynomial size is incomparable with the complexity class AC^0. This sheds some new light on the relationship between randomized, nondeterministic and deterministic classes asked in [JRSW97].

Property 2 *$PERM$ is in AC^0, and f_n is not in AC^0.*

Proof. The function $PERM$ can be represented formally as follows.

$$PERM(X) = ROWS(X) \wedge COLMNS(X),$$

where $ROWS(X) = 1$ if and only if each row of matrix X contains exactly one entrance of 1. We have,

$$ROW(X) = ROW_1(X) \wedge \ldots \wedge ROW_n(X),$$

where $ROW_i(X) = 1$ if and only if i-th row of matrix X contains exactly one 1.

$$ROW_i(X) = K_1 \vee \ldots \vee K_n,$$

where $K_j(X) = \bar{x}_{i1} \wedge \ldots \wedge \bar{x}_{ij-1} \wedge \bar{x}_{ij} \wedge \bar{x}_{ij+1} \wedge \ldots \bar{x}_{in}$. This representation proves that $PERM \in \mathtt{AC}^0$.

To prove that $f_n \notin \mathtt{AC}^0$, we show that $PARITY$ is AC^0-reducible to f_n. Set all even bits of f_n to 0. We get a subfunction of f_n which is a symmetric function. Using now a property that the value of this subfunction is 1 iff a half of remaining inputs are set to 1, we can represent $PARITY$ as OR of linearly many such boolean subfunctions. ∎

5 Open Problems and Further Research

We have displayed an explicit boolean function outside the class $\mathtt{AC}^0$ which can be computed by polynomial size randomized ordered read-once programs, and which requires for any deterministic or nondeterministic ordered read-k-times branching program an exponential size (for $k = o(n/\log n)$).

It will be very interesting to develop new lower bound techniques for randomized branching programs in order to separate the classes of boolean functions computable *efficiently* by randomized branching programs (with restricted number of testing variables) from the class that remains *hard* for randomized programs. Another interesting issue is an *exact dependence* of the sizes of the read-once branching programs on their respective error probabilities.

Acknowledgements

We would like to thank Stephen Ponzio, Sasha Razborov, Roman Smolensky, and Thomas Thierauf for interesting discussions on the subject of this paper.

References

[A94] F. Ablayev, *Lower Bounds for Probabilistic Space Complexity: Communication-Automata Approach*, Proc. LFCS '94, Lecture Notes in Computer Science, Springer-Verlag, 813, (1994), pp. 1-7.

[AK96] F. Ablayev and M. Karpinski, *On the Power of Randomized Branching Programs*, in Proc. ICALP'96, Lecture Notes in Computer Science, Springer-Verlag, 1099, (1996), 348-356; see also ECCC TR95-054, 1995. Available at `http://www.eccc.uni-trier.de/eccc/`.

[A78] L. Adelman, *Two Theorems on Random Polynomial Time*, Proc. 19th IEEE FOCS (1978), pp. 75-83.

[AB84] M. Ajtai and M. Ben-Or, *A Theorem on Randomized Constant Depth Circuits*, Proc. 16th ACM STOC (1984), pp. 471-474.

[BSSW94] B. Bolling, M. Sauerhoff, D. Sieling and I. Wegener, *On the Power of Different Types of Restricted Branching Programs*, ECCC, TR94-025, 1994. Available at `http://www.eccc.uni-trier.de/eccc/`.

[BRS93] A. Borodin, A. Razborov and R. Smolensky, *On Lower Bounds for Read-k-Times Branching Programs*, Computational Complexity, 3, (1993), pp. 1-18.

[B93] A. Borodin, *Time–Space Tradeoffs (Getting Closer to the Barrier?)*, in Proceedings of the ISAAC'93, Lecture Notes in Computer Science, Springer-Verlag, 762, (1993), pp. 209-220.

[B92] R. Bryant, *Symbolic Boolean Manipulation with Ordered Binary Decision Diagrams*, ACM Computing Surveys, 24, No. 3, (1992), pp. 293-318.

[F79] R. Freivalds, *Fast Probabilistic Algorithms*, in Proc. MFCS'79, Lecture Notes in Computer Science, Springer-Verlag, 74, (1979), pp. 57-69.

[FK95] R. Freivalds and M. Karpinski, *Lower Time Bounds for Randomized Computation*, Proc. ICALP'95, Lecture Notes in Computer Science, Springer-Verlag, 944, (1995), pp. 183-195.

[GKPR96] L. Gasieniec, M. Karpinski, W. Plandowski and W. Rytter, *Randomized Efficient Algorithms for Compressed Strings: The Finger-Print Approach*, Proc. 7th Annual Symposium on Combinatorial Pattern Matching (1996), pp.39-49.

[HR88] B. Halstenberg and R. Reischuk, *On Different Models of Communication*, Proc. 20th ACM STOC, (1988), pp. 162-172.

[J89] S. Jukna, *On the Effect of Null-Chains on the Complexity of Contact Schemes*, Proc. FCT'89, Lecture Notes in Computer Science, Springer-Verlag, 380, pp. 246-256.

[J94] S. Jukna, *A Note on Read-k-Times Branching Programs*, RAIRO Theoretical Informatics and Applications, 29, No 1 (1995), pp. 75-83.

[JRSW97] S. Jukna, A. Razborov, P. Savicky and I. Wegener, *On P versus $NP \cap co$–NP for Decision Trees and Read-Once Branching Programs*, ECCC, TR97-023, 1997. Available at `http://www.eccc.uni-trier.de/eccc/` .

[KR87] R. Karp and M. Rabin, *Efficient Randomized Pattern-Matching Algorithm*, IBM Journal of Research and Development, 31, (1987), pp. 249-260.

[KMW88] M. Krause, C. Meinel and S. Waack, *Separating the Eraser Turing Machine Classes L_e, NL_e, $co-NL_e$, and P_e*, Proc. MFCS'88, Lecture Notes in Computer Science, Springer-Verlag, 324, pp. 405-413.

[K91] M. Krause, *Lower Bounds for Depth-Restricted Branching Programs*, Information and Computation, 91, (1991), pp. 1-14.

[KN97] E. Kushilevitz and N. Nisan, *Communication Complexity*, Cambridge University Press, 1997.

[L59] C. Lee, *Representation of Switching Circuits by Binary-Decision Programs*, Bell System Technical Journal, 38, (1959), pp. 985-999.

[M76] W. Masek, *A Fast Algorithm for the String Editing Problem and Decision Graph Complexity*, M.Sc. Thesis, Massachusetts Institute of Technology, Cambridge, May 1976.

[P95] S. Ponzio, *A Lower Bound for Integer Multiplication with Read-Once Branching Programs*, Proc. 27th ACM STOC, (1995), pp. 130-139.

[R91] A. Razborov, *Lower Bounds for Deterministic and Nondeterministic Branching Programs*, Proc. FCT'91, Lecture Notes in Computer Science, Springer-Verlag, 529, (1991), pp. 47-60.

[S97] M. Sauerhoff, *A Lower Bound for Randomized Read-k-Times Branching Programs*, ECCC, TR97-019, 1997. Available at `http://www.eccc.uni-trier.de/eccc/`.

[SS93] J. Simon and M. Szegedy, *A New Lower Bound Theorem for Read-Only-Once Branching Programs and its Applications*, Advances in Computational Complexity Theory, ed. Jin-Yi Cai, DIMACS Series, 13, AMS (1993), pp. 183-193.

[W87] I. WEGENER, *The Complexity of Boolean Functions*, Wiley-Teubner Series in Computer Science, 1987.

[W94] I. WEGENER, *Efficient Data Structures for Boolean Functions*, Discrete Mathematics, 136, (1994), pp. 347-372.

[Y79] A. YAO, *Some Complexity Questions Related to Distributive Computing*, Proc. 11th ACM STOC (1979), pp. 209-213.

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'