On the Power of Randomized Branching Programs

Farid Ablayev * Marek Karpinski †

Abstract

We define the notion of a randomized branching program in the natural way similar to the definition of a randomized circuit. We exhibit an explicit function f_n for which we prove that:

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

2) f_n cannot be computed in polynomial size by deterministic read-once branching programs;

3) f_n cannot be computed in polynomial size by deterministic read-k-times ordered branching program for $k = o(n/\log n)$ (the required deterministic size is $\exp\left(\Omega\left(\frac{n}{k}\right)\right)$).

1 Preliminaries

Different models of branching programs introduced in [13, 15], have been studied extensively in the last decade (see for example [19]). A survey of known lower bounds for different models of branching programs can be found in [17].

Developments in the field of digital design and verification have led to the introduction of restricted forms of branching programs. In particular, ordered read-once branching programs are now commonly used in the circuit verification [9], [20]. But many important functions cannot be computed by read-once branching programs of polynomial size. For more information see the survey [9] and papers [18], [16].

It is known that different models of randomized circuits with weak enough restrictions on the error of randomized computation have only polynomial

*Dept. of Computer Science, University of Bonn. Visiting from University of Kazan. Research partially supported by the Volkswagen-Stiftung and the Basic Research Grant 96-01-01962 Email: ablayev@ksu.ras.ru

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

advantage over nonuniform deterministic models (see [2], [4], [3], and survey [6]). In the paper we define the 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 for read-once polynomial branching programs can be more powerful than deterministic ones. The argument that can help the intuition in this direction is that amplification method does not work for the case of restricted number of input verifications. Note that in the paper [7] it is presented an explicit function which needs exponential size for presentation by a nondeterministic read-k-times branching program for $k = o(\log n)$.

We use the variant of a definition of a branching program from the paper [7]. A deterministic branching program P for computing a function $g : \Sigma^n \to \{0,1\}$, where Σ is a finite set, is a directed acyclic multi-graph with a single source node, distinguished sink nodes labeled "accept" and "reject". For each non-sink node there is a variable x_i such that all out-edges from this node are labeled by "$x_i = \delta$" for some $\delta \in \Sigma$ and for each δ there is exactly one such labeled edge. The label "$x_i = \delta$" indicates that only inputs satisfying $x_i = \delta$ may follow this edge in the computation. We call a node v an x_i-node if all output edges of the node v are labeled by "$x_i = \delta$", $\delta \in \Sigma$.

A deterministic branching program P computes a function $g : \Sigma^n \to \{0,1\}$, in the obvious way; that is, $g(\sigma_1, \ldots, \sigma_n) = 1$ iff there is a computation on $< \sigma_1, \ldots, \sigma_n >$ starting in the source state and leading to the accepting state.

A randomized branching program is a one which has in addition to its standard (deterministic) inputs some specially designated inputs called random inputs. When these random inputs are chosen from the uniform distribution, the output of the branching program is a random variable. We call a node v of the randomized branching program a "random generator" node if output edges of the node v are labeled by random inputs.

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

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

From the definition of the complexity of a branching program it follows that the size of randomized branching program is the sum of random generator nodes and x_i-nodes.

Read-once branching programs are branching programs in which for every path, every variable is tested no more than once. A read-once ordered branching program is a read-once branching program which respects a fixed ordering π of the 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 path in the program. A read-k-times ordered branching program is a read-k-times branching program which is partitioned into k layers such that the each layer is a read-once ordered respecting the same ordering π. In [5] it is proved that deterministic ordered read-$(k+1)$-times branching programs are more powerful than deterministic ordered read-k-times branching programs. Namely classes of functions computed by deterministic polynomial-size read-k-times ordered branching programs form proper hierarchy for $k = o(n^{1/2}/\log^2 n)$.

We exhibit an explicit function $f_n : \{0,1,\hat{0},\hat{1}\}^{2n} \to \{0,1\}$, for which we prove that:

(i) Function f_n can be computed with one sided $\varepsilon(n)$-error by randomized read-once ordered branching program with the size $O\left(\frac{n^6}{\varepsilon^3(n)}\log^2\frac{n}{\varepsilon(n)}\right)$ (Theorem 1).

(ii) Any deterministic read-once branching program that computes function f_n has the size no less than 2^n (Theorem 2).

(iii) Any deterministic read-k-times ordered branching program for computing function f_n has size no less than $2^{(n-1)/(2k-1)}$ (Theorem 3).

Function f_n can be easily defined as a boolean function. For technical reasons in the proofs we prefer to use the above notation.

Note that one can think of each internal node of a branching program as a state of a computation. This point of view is essential for the investigation of the amount of space necessary to compute functions. Restricted models of branching programs are useful for the investigation of time-space tradeoffs. We can think of read-k-time ($k \geq 1$) restrictions as a restriction on time, say time $\leq kn$ (see survey [8] for more information). This approach draws 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 [1] and [11].

2 Function

Consider the finite alphabet $\Sigma = \{0,1,\hat{0},\hat{1}\}$. As usual Σ^* and Σ^n denote the set of all words of finite length and the length n over Σ respectively.

For $\sigma_1, \sigma_2 \in \Sigma$, $x \in \Sigma^*$ define $Proj_{\sigma_1,\sigma_2}(x)$ to be the longest subsequence x' of the sequence x that consists only of symbols σ_1 and σ_2.

Define function $f_n : \Sigma^{2n} \to \{0,1\}$ as follows: $f(x) = 1$ iff

1) $Proj_{0,1}(x)$ and $Proj_{\hat{0},\hat{1}}(x)$ have the same length and

2) i-th symbol in $Proj_{0,1}(x)$ is σ_i iff the i-th symbol in $Proj_{\hat{0},\hat{1}}(x)$ is $\hat{\sigma}_i$ for all i.

Informally speaking inputs of f_n are words over the alphabet Σ which consists of two kinds of zeroes and two kinds of ones. $f_n(x) = 1$ iff a subsequence z of x formed by the first kind of zeroes and ones and a subsequence y of x

formed by the second kind of zeroes and ones are binary notations of the same natural number.

As it is mentioned in the section above, function f_n can be easily defined as a boolean function $f'_n : \{0,1\}^{4n} \to \{0,1\}$. One can encode, for example, 0 by 00, 1 by 01, $\hat{0}$ by 10, and $\hat{1}$ by 11. Our presentation help us to make main ideas of proof methods more clear and help us to avoid several technical details in proofs.

3 Results

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

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

Proof: Randomized read-once ordered branching program P that computes f_n works as follows:

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

P selects a prime number p in the following way. P use $t = \lceil \log d(n) \rceil$ random 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 fixed order $y_1, y_2, \ldots, y_t$. Sequence $y = y_1 y_2 \ldots y_t$ is interpreted as binary notation of the number $N(y)$. P selects i-th prime number $p_i \in Q_{d(n)}$ iff $N(y) = i \bmod d(n)$.

Phase 2. (deterministic). Let $\sigma \in \Sigma^{2n}$ be a valuation of x. Denote $\alpha = Proj_{0,1}(\sigma)$, $\beta = Proj_{\hat{0},\hat{1}}(\sigma)$. We treat $\hat{\sigma}_i$ to be the number 0 if $\hat{\sigma}_i = \hat{0}$, and to be the number 1 if $\hat{\sigma}_i = \hat{1}$. Sequences α and β are interpreted as binary notations of numbers $N(\alpha)$ and $N(\beta)$. P reads input sequence $x = \sigma$ in the order $x_1, \ldots, x_{2n}$.

Along the computation path, P

a) verifies if $|\alpha| = |\beta|$,

b) counts modulo p the numbers $N(\alpha)$ and $N(\beta)$ ($a = N(\alpha) \bmod p$ and $b = N(\beta) \bmod p$) in the following way. In the beginning of computation $a := 0$ and $b := 0$. When P reads i-th input symbol $\sigma_i \in \{0,1\}$ of the sequence α (respectively i-th input symbol $\hat{\sigma}_i \in \{\hat{0},\hat{1}\}$ of the sequence β) then $a := a + \sigma_i 2^i \bmod p$ (respectively $b := b + \hat{\sigma}_i 2^i \bmod p$).

Let α' and β' be first parts of the length t and k respectively of subsequences α and β that were tested during the path from the source to the internal node (state) v. For the realization of the procedure described in the *phase 2* it is sufficient to store in the state v four numbers: $t, k \in \{0, 1, \ldots, n\}$, $a = N(\alpha') \pmod p$, and $b = N(\beta') \pmod p$.

If $|\alpha| \neq |\beta|$ then P outputs 1 correct answer with probability 1.

Consider the case $|\alpha| = |\beta|$.

If $N(\alpha) = N(\beta) \pmod p$ then P outputs 1 else P outputs 0.

From the description of P it follows that if $N(\alpha) = N(\beta))$ then P with probability 1 outputs correct answer. If $N(\alpha) \neq N(\beta)$ then it can happen that $N(\alpha) = N(\beta) \pmod p$ for some $p \in Q_{d(n)}$. In these cases P make error output.

For $x = \sigma$ it holds that $|N(\alpha) - N(\beta)| \leq 2^n < p_1 p_2 \cdots p_n$ where $p_1, p_2, \ldots, p_n$ are first n prime numbers. This means that in the case when $N(\alpha) \neq N(\beta)$ the probability $\varepsilon(n)$ of the error of P on the input $x = \sigma$ is no more than $2n/d(n)$.

The size of P is no more than

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

It is known from the number theory that the value of the i-th prime is of order $O(i \log i)$. Therefore from the above upper bound for the $size(P)$ and from the upper bound for $\varepsilon(n)$ it follows that

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

∎

Theorem 2. *Any deterministic read-once branching program that computes the function f_n has the size* of *no less than 2^n.*

Proof: Consider an arbitrary deterministic read-once branching program P that computes function f_n. Let v be a node of the P. Let $\sigma = \sigma_1 \sigma_2 \ldots \sigma_l$ be a sequence of symbols over Σ. We will write $v = v(\sigma)$ if there is a sequence $x_{i_1}, x_{i_2}, \ldots, x_{i_l}$ of variables such that edges $x_{i_1} = \sigma_1, x_{i_2} = \sigma_2, \ldots, x_{i_l} = \sigma_l$ form a path P from the source to the node v. Denote $x(\sigma) = \{x_{i_1}, x_{i_2}, \ldots, x_{i_l}\}$.

For the node $v(\sigma)$ denote $f_{v(\sigma)}$ the function which is computed by P when the node $v(\sigma)$ is considered as a source node. $f_{v(\sigma)}$ is the sub function of f_n where we have replaced the variables read on $x(\sigma)$ by the proper constants from σ.

For proving the lower bound of the theorem it is enough to show that for any $\sigma, \sigma' \in \{0,1\}^n$, $\sigma \neq \sigma'$ it holds that $v(\sigma) \neq v(\sigma')$.

Assume that there are sequences $\sigma = \sigma_1 \sigma_2 \ldots \sigma_n \in \{0,1\}^n$ and $\sigma' = \sigma'_1 \sigma'_2 \ldots \sigma'_n \in \{0,1\}^n$ such that $\sigma \neq \sigma'$ and $v(\sigma) = v(\sigma') = v$. P is read-once. This means that $f_{v(\sigma)}$ and $f_{v(\sigma')}$ are functions over the same set of variables and $f_{v(\sigma)} = f_{v(\sigma')}$. From the definition of the function f_n we have that there exists a sequence $\hat{\sigma} \in \{\hat{0}, \hat{1}\}^n$ such that $f_{v(\sigma)}(\hat{\sigma}) = 1$ but $f_{v(\sigma')}(\hat{\sigma}) = 0$. This means that $f_{v(\sigma)} \neq f_{v(\sigma')}$. ∎

Note that the proof of the theorem 2 can be also obtained as a corollary from theorem 2.1 [18].

Below we prove an exponential lower bound for the complexity of presentation of the function f_n by deterministic read-k-times ordered branching program. For proving it we use a method based on two-way communication game. We present this method in the lemma below for a more common notion of ordering variables for a branching program than the traditional ones.

Note that the method based on communication game is used in the paper [12] and later in [5] for proving lower bound for deterministic read-k-times ordered branching programs.

Definition 1. Call a read-once branching program a π-weak-ordered read-once branching program if it respects an ordered partition π of the variables into two parts X_1 and X_2, i.e. 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.

Call a read-k-times branching program read-k-times π-weak-ordered if it is partitioned to k layers such that the each layer is a π-weak-ordered read-once respecting the same ordered 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 on the rest part of computation path no variables from X_1 can be tested.

We call 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 read-once (read-k-times) branching program is ordered then it is weak-ordered.

For a function $g : \Sigma^n \to \{0,1\}$ and for a partition π of the set of variables x of g into two parts X_1 and X_2, denote by $C_{k,\pi}(g)$ a k-round deterministic communication complexity of g for the communication game with two players A and B where A obtains variables from the first part X_1 of variables and B obtains variables from the second part X_2 of variables of g.

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

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

Proof: Consider the following communication game with two players A and B for computing function f_n. Let X_1 and X_2 be two sets determined by partition π of a set of variables x of P. Part of the input corresponding to X_1 is known to A, and part of the input corresponding to X_2 is known to B. Players A and B have the copy of P. In order to compute f_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 one to send a message. The output is produced by B. Let $\sigma \in \Sigma^{2n}$ be a valuation of x.

Denote σ_A and σ_B parts of input σ which correspond to variables from X_1 and X_2 (inputs of A and B), respectively.

For each i, $1 \leq i \leq k-1$, communication protocol ϕ simulates computation on the i-th layer of P by two communication rounds $2i-1$ and $2i$.

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

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

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

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

Denote by V_i the set of all internal nodes which can be send 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 computations on Σ^{2n}. Denote $d_i = |V_i|$. From our notation it follows that the number of all full messages that can be exchanged on inputs from Σ^{2n} according to protocol ϕ is no more than $\prod_{i=1}^{2k-1} d_i$.

The number of full messages used by ϕ cannot be less than $2^{C_{2k-1,\pi}(g)-1}$

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

The lower bound of the lemma follows from the inequality above (with $d = \max\{d_i : i \in \{1, 2, \ldots, 2k-1\}\}$) for which it holds that

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

and hence

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

∎

Theorem 3. *Any deterministic ordered read-k-times branching program that computes function f_n has the size no less than $2^{(n-1)/(2k-1)}$.*

Proof: Let P be an ordered read-k-times branching program with an ordering π of variables which computes the function f_n. Consider the following partition $\hat{\pi}$ of variables x of f_n into two parts $X_1 = \{x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(n)}\}$ and $X_2 = \{x_{\pi(n+1)}, x_{\pi(n+2)}, \ldots, x_{\pi(2n)}\}$. It is obvious that P is read-k-times $\hat{\pi}$-weak-ordered.

Denote CM a communication matrix of the function f_n for the partition $\hat{\pi}$ of the variables x. Consider the $2^n \times 2^n$ sub-matrix CM' of CM which is formed by strings that correspond to the part of inputs from $\{0,1\}^n$ and columns that correspond to the part of inputs from $\{\hat{0},\hat{1}\}^n$. Matrix CM' is the E matrix (elements of the main diagonal are 1 and all rest elements are 0). This means that

$$C_{t,\pi}(f_n) \geq n$$

for $t \geq 1$. From the lower bound for $C_{t,\pi}(f_n)$ above and the lemma 1 it follows that

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

The lower bound of the theorem follows from considering the best read-k-times ordered branching program that computes f_n. ∎

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

4 Further Research and Open Problems

We conclude with two open problems:

1. It will be interesting to describe how to separate "hard functions" (functions for which *randomization* does not improve their branching program complexity for the restricted number of testing variables) from the functions which can be computed more efficiently using randomization. Another words it is an interesting open problem to develop new randomized lower bound techniques for branching programs.

2. What is the exact dependence of the size of randomized branching programs on the error of computation?

Acknowledgments. We thank Sasha Razborov and Roman Smolensky for a number of interesting discussions on the subject of the paper.

References

[1] F. Ablayev, Lower bounds for probabilistic space complexity: communication-automata approach, *in Proceedings of the LFCS'94, Lecture Notes in Computer Science, Springer-Verlag*, 813, (1994), 1-7.

[2] L. Adelman, Two theorems on random polynomial time, *in Proceedings of the 19-th FOCS*, (1978), 75-83.

[3] M. Ajtai and M. Ben-Or, A theorem on randomized constant depth circuits, *in Proceedings of the 16-th STOC*, (1984), 471-474.

[4] C. Bennet and J. Gill, Relative to a random oracle A, $P^A \neq NP^A \neq co-NP^A$ with probability 1, *SIAM J. Comput*, 10, (1981), 96-113.

[5] B. Bolling, M. Sauerhoff, D. Sieling, and I. Wegener, On the power of different types of restricted branching programs, *ECCC Reports 1994*, TR94-025.

[6] R. Boppana and M. Sipser, The complexity of finite functions, *in Handbook of Theoretical Computer Science*, Vol A: Algorithms and Complexity, MIT Press and Elsevier, The Netherlands, 1990, ed. J Van Leeuwen, 757-804.

[7] A. Borodin, A. Razborov, and R. Smolensky, On lower bounds for read-k-times branching programs, *Computational Complexity*, 3, (1993), 1-18.

[8] A. Borodin, Time-space tradeoffs (getting closer to barrier?), *in Proceedings of the ISAAC'93, Lecture Notes in Computer Science, Springer-Verlag*, 762, (1993), 209-220.

[9] R. Bryant, Symbolic boolean manipulation with ordered binary decision diagrams, *ACM Computing Surveys*, 24, No. 3, (1992), 293-318.

[10] R. Frevalds, Fast probabilistic algorithms, *in Proceedings of the Conference Mathematical Foundation of Computer Science 1979, Lecture Notes in Computer Science, Springer-Verlag*, 74, (1979), 57-69.

[11] R. Freivalds and M. Karpinski, Lower time bounds for randomized computation, *in Proceedings of the ICALP'95, Lecture Notes in Computer Science, Springer-Verlag*, 944, (1995), 183-195.

[12] M. Krause, Lower bounds for depth-restricted branching programs, *Information and Computation*, 91, (1991), 1-14.

[13] C. Y. Lee, Representation of switching circuits by binary-decision programs, *Bell System Technical Journal*, 38, (1959), 985-999.

[14] L. Lovasz, Communication complexity: a survey, *in "Paths, Flows and VLSI Layout", Korte, Lovasz, Proemel, Schrijver Eds., Springer-Verlag* (1990), 235-266.

[15] W. Masek, A fast algorithm for the string editing problem and decision graph complexity, M.Sc. Thesis, Massachusetts Institute of Technology, Cambridge, May 1976.

[16] S. Ponzio, A lower bound for integer multiplication with read-once branching programs, *Proceedings of the 27-th STOC*, (1995), 130-139.

[17] A. Razborov, Lower bounds for deterministic and nondeterministic branching programs, *in Proceedings of the FCT'91, Lecture Notes in Computer Science, Springer-Verlag*, 529, (1991), 47–60.

[18] 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), 183-193.

[19] I. Wegener, *The complexity of Boolean functions*. Wiley-Teubner Series in Comp. Sci., New York – Stuttgart, 1987.

[20] I. Wegener, Efficient data structures for boolean functions, *Discrete Mathematics*, 136, (1994), 347-372.