On the Computation Power of Randomized Branching Programs

Marek Karpinski*

Abstract

We survey some upper and lower bounds established recently on the sizes of *randomized* branching programs computing explicit boolean functions. In particular, we display boolean functions on which *randomized* read-once ordered branching programs are exponentially more powerful than deterministic or nondeterministic read-k-times branching programs for any $k = o(n/\log n)$. We investigate further computational power of *randomized* read-once order branching programs (OBDDs) and their basic manipulation properties for verification of boolean functions and for testing graphs of arithmetic functions.

*Dept. of Computer Science, University of Bonn, 53117 Bonn, Email: marek@cs.uni-bonn.de. Research partially supported by the International Computer Science Institute, Berkeley, California, by the DFG Grant KA 673/4-1, and by the ESPRIT BR Grants 7097, 21726, and EC-US 030, and by the Max–Planck Research Prize.

1 Introduction

The model of restricted branching programs has recently been found very useful in a number of applications. Its special variant, *ordered read-once branching programs* become an important computational model and a technical tool in the fields of circuit design and hardware verification. They are also known as "OBDDs" (ordered binary decision diagrams). The approach used depends on converting independently the circuit description, and the function specification to a common intermediate representation (being the OBDD), and then testing whether the two representations are equivalent (c.f., e.g. [W94]). This approach has an apparent shortcoming, in that we cannont hope in general for a polynomial size intermediate representation in the form of an OBDD. It turns out in fact that many important elementary functions do not have polynomial size read-once branching progams. Examples are: multiplication, squaring, and inversion [P95a]. During the last decade there were several attempts to find generalization of OBDDs model more powerful computationally and still algorithmically manipulable. In this paper we are concerned with the randomized extension of the read-once branching programs and analize their computational power compared with deterministic and nondeterministic models.

2 Randomized Branching Programs

A *deterministic* branching program P for computing a boolean function $f: \{0,1\}^n \to \{0,1\}$ is a directed acyclic multi-graph with a distinguished source node s and a distinguished (accepting) sink node t. The outdegree of each non-sink node is exactly 2, and the two outgoing edges are labeled by "$x_i = 0$" and "$x_i = 1$" for a variable 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 f 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 $\sigma = \sigma_1\sigma_2 \ldots \sigma_n$. The size of P is its number of internal nodes.

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 f, in the obvious way; that is, $f(\sigma) = 1$ iff there exists (at least one) computation over σ starting in the source node s and leading to the accepting node t.

Define a *probabilistic* 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 probabilistic branching program P (a, b)-*computes* a function f if it outputs 1 with the probability at most a for an input σ such that $f(\sigma) = 0$, and it outputs 1 with the probability at least b for an input σ such that $f(\sigma) = 1$. A probabilistic branching program is called *randomized* if it $(\varepsilon, 1 - \varepsilon)$-computes the function f for some $\varepsilon < 1/2$. ε is called an error probability of P.

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 *probabilistic* 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.

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 branching program respecting the same ordering π. Ordered branching programs can be *layered*. In this case all nodes that test the same variable should have the same distance from the source node. This can be accomplished easily by introducing redundant nodes. In the case of probabilistic branching programs we stipulate additionally that the deterministic and probabilistic *layers* alternate. The *width* of such programs is the maximum size of a layer.

3 Explicit Boolean Functions

In this section we define some explicit boolean functions for which we are going to prove computational upper and lower bounds on different types of branching programs.

Firstly, we define a boolean function $F_n: \{0,1\}^{4m} \to \{0,1\}$ as follows. For $x \in \{0,1\}^{4m}$ we shall call the *odd* bits, the "*type*" bits, and the *even* bits, the "*value*" bits. We say that the *even* bit x_i, $i \in \{2,4,\ldots,4m\}$ is of "*type*" $0(1)$ if the corresponding *odd* bit x_{i-1} is $0(1)$. For $x \in \{0,1\}^{4m}$, we denote by $x^0(x^1)$ a subsequence of x that consists of all even bits of type $0(1)$.

Now we define a boolean function $f_n: \{0,1\}^n \to \{0,1\}$ as follows: $f_n(x) = 1$ iff $x^0 = x^1$.

We are going to define now the second class of boolean functions. For a given integer n denote by $p[n]$ the smallest prime greater or equal to n. For every integer s, define

$$\omega_n(s) = \begin{cases} j & \text{if } j = s \bmod p[n] \text{ and } 1 \le j \le p[n], \\ 1 & \text{otherwise.} \end{cases}$$

Define a boolean function $g_n: \{0,1\}^n \to \{0,1\}$ as follows. $g_n(x) = x_j$ for $j = \omega_n(\sum_{i=1}^n i x_i)$.

We define a function PERM_n (cf. [KMW88], [J89]) on a boolean $n \times n$ matrix $x = [x_{ij}]_{1 \le i,j \le n}$, $\mathrm{PERM}_n: \{0,1\}^{n^2} \to \{0,1\}$. For a given $x \in \{0,1\}^{n^2}$,

$\mathrm{PERM}_n(x) = 1$ iff x is a permutation matrix, i.e., each row and each column of x contains exactly one 1 entry.

We introduce now a boolean function DMULT: $\{0,1\}^{4n} \to \{0,1\}$ of testing integer multiplication such that $\mathrm{DMULT}(x,y,z) = 1$ iff $xy = z$ (x, y, and z are binary representations of integer numbers, and $|x| = |y| = n$, $|z| = 2n$).

Further, we define the integer multiplication function MULT as follows. The function MULT_k: $\{0,1\}^{2n} \to \{0,1\}$ computes the kth bit of the product of two n-bit integers, i.e., $\mathrm{MULT}_k(x,y) = z_k$ where $x = x_{n-1} \ldots x_0$, $y = y_{n-1} \ldots y_0$, and $z = z_{2n-1} \ldots z_0$ for $0 \leq k \leq 2n-1$. Now define MULT to be MULT_{n-1} computing the *middle* bit in the product xy. It is known that the middle bit is the "*hardest*" bit in the multiplication (cf., e.g., [P95a]). It is also well known that MULT besides being *hard* for many arithmetic functions is also reducible under *read-once* reductions to other arithmetic functions like *squaring*, *inversion*, and *division* (cf. [P95b]).

4 Randomized Upper Bounds and Deterministic Lower Bounds

We are going to characterize the computational power of randomized OBDDs on the explicit boolean functions introduced in Section 3, and formulate also corresponding lower bounds on the deterministic branching programs. The techniques for randomized upper bounds and first separating deterministic lower bounds were introduced by Ablayev and Karpinski [AK96], [AK98a]. The other bounds were proven by Sauerhoff [S97a], Krause, Meinel and Wack [KMW88], and Jukna [J89], [J95].

Theorem 1. ([AK96], [AK98a])

1. *The function f_n can be computed by an $\varepsilon(n)$-error randomized OBDD of size*

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

2. *The size lower bound on any nondeterministic ordered read-k-times branching program computing* f_n *is* $2^{\Omega(n/k)}$.

Theorem 2. ([S97a], [AK98a], [KMW88], [J89])

1. *The function* PERM_n *can be computed by an* $\varepsilon(n)$*-error randomized OBDD of size*
$$O\left(\frac{n^5}{\varepsilon^2(n)}\log^3 n\right) \quad .$$
2. *The size lower bound on any nondeterministic read-once branching program computing* PERM_n *is* $2^{\Omega(n)}$.

The first part of the next theorem formulates a surprising fact on the power of randomized OBDDs for testing graphs of arithmetic functions.

Theorem 3. ([AK98b], [J95])

1. *The* test function *for* integer multiplication DMULT *can be computed by an* $\varepsilon(n)$*-error randomized OBDD of size*
$$O\left(\frac{n^6}{\varepsilon^5(n)}\log^4\frac{n}{\varepsilon(n)}\right) \quad .$$
2. *The size lower bound on any nondeterministic syntactic read-k-times branching program computing* DMULT *is* $O(2^{\Omega(n^{1/4}/k^{2k})})$.

5 Randomized Lower Bounds

The following randomized lower bounds of [A97], and [AK98b] were established using the property of the entropy function, and the one-way probabilistic communication complexity arguments.

Theorem 4. ([A97])

1. *The size lower bound on any randomized OBDD computing the function* g_n *is* $2^{\Omega(n/\log n)}$.

2. *The function g_n can be computed by a nondeterministic ordered read-once branching program in size $O(n^3)$.*

Theorem 5. ([AK98b])

The size lower bound on any randomized OBDD computing the integer multiplication function MULT *is* $2^{\Omega(n/\log n)}$.

6 Manipulability and Satisfiability Problem for Randomized OBDDs

It is easy to see that randomized OBDDs are closed under boolean combinations, and that various boolean model checking combinations of randomized OBDDs stay in the class of randomized OBDDs. In particular, equivalence problem for randomized OBDDs can be reduced to the satisfiability problem.

We call a *width* of a randomized OBDD to be a maximum number of nodes in a layer of a program.

The following recent results of Agrawal and Thierauf [AT97] relate the computational complexity of the satisfiability problem for randomized OBDDs to their error probability.

Theorem 6. ([AT97])

1. *The* satisfiability problem *for randomized OBDDs is NP-complete.*
2. *Given a randomized OBDD P with an error probability $\varepsilon < (1/W + 2)$ for W the width of P. There is a polynomial time algorithm for solving the satisfiability problem for P.*

7 Randomized Read-k-Times Branching Programs

It was observed by Borodin, Razborov and Smolensky [BRS93] that there are two different types of read-k-times branching programs, the first, *syntactic* type

where the restriction on readings applies to all paths in the program, and the second, *semantic* type where the restriction on readings applies only to consistent computational paths. The corresponding classes of functions are potentially different. The two classes coincide for $k = 1$. For $k \geq 2$, up to now no non-trivial lower bounds are known for semantic read-k branching programs. In the sequel we deal only with *syntactic* read-k-times branching and their randomized, deterministic, and nondeterministic variants.

For a number p we call a branching program *p-way* if its outgoing edges are labeled with "$x_i = 0$", "$x_i = 1$", ..., and "$x_i = p - 1$".

We define now the function SIP: $\mathbb{Z}_3^n \times \mathbb{Z}_3^n \to \{0,1\}$ (Silvester inner product) for $n = 2^\ell$, by

$$\mathrm{SIP}(x,y) = 1 \quad \text{iff} \quad x^\top A y = 0$$

for $A = [a_{ij}]_{1 \leq i,j \leq 2^\ell}$ the *Sylvester matrix* of dimension $2^\ell \times 2^\ell$,

$$a_{i+1,j+1} = (-1)^{\langle \mathrm{bin}(i), \mathrm{bin}(j) \rangle}$$

for $0 \leq i, j \leq 2^\ell - 1$, and with $\mathrm{bin}(i)$ the binary representation of i, and $\langle \, , \rangle$ the inner product in $\mathbb{Z}_2^\ell$.

A boolean variant SIP_B of SIP can be obtained by a straightforward encoding of $\mathbb{Z}_3$ over $\{0,1\}^2$ (see for details [S97a]).

Using a modified technique of *rectangles* of Borodin, Razborov and Smolensky [BRS93], and combining it with the communication complexity arguments, Sauerhoff [S97a] was able to prove

Theorem 7. ([S97a])

The size lower bound on any randomized 3-way (2-way) read-k-times branching program computing SIP *(*SIP_B*) is* $2^{\Omega(n/c^k k^3)}$ *for some constant c.*

8 Some Further Results

Quite recently some further exponential lower bounds on randomized read-k-times branching programs were obtained by Thathacher [T98]. Also some in-

teresting insights about computational power of Las Vegas (zero-error) branching program were gained recently by Sauerhoff [S98]. [S98] displays an explicit boolean function ("addressing functions") and designs for it a polynomial size Las Vegas read-once branching program. It is well known that this function cannot be computed by polynomial size deterministic read-once branching programs.

On other hand Karpinski and Mubarakzjanov [KM98] proved using communication complexity techniques, that *Las Vegas public coin* (all random variables are read at the beginning of computation) OBDDs are equivalent to deterministic OBDDs.

One can also construct an explicit boolean function which is computable by polynomial size randomized OBDD but not computable in polynomial size by any nondeterministic or co-nondeterministic OBDD (cf. [AKM98])

9 Open Problems

It remains an important open problem to develop new more powerful lower bound techniques for randomized read-once (and read-k-times) branching programs. A development of new two-way probabilistic communication complexity techniques could be a possible way to accomplish it.

Also an important open problem remains the status of the integer multiplication function MULT on randomized read-once and read-k-times branching programs on both types syntactic, and semantic programs.

A challenging open remains still a constuction of an explicit boolean function which can be computed in polynomial size by both nondeterministic and co-nondeterministic read-once branching programs and which is not computable by any polynomial size randomized read-once branching program (cf. also [JRSW97]).

Another question concerns the computational power of Las Vegas OBDDs and Las Vegas ordered read-k-times branching programs.

It would be also very interesting to shed some light on computational power of randomized branching with restricted readings of variables and additionally

equipped with algebraic branching elements like evaluation of polynomials and branching on the sign (for the corresponding randomized decision tree model see [GKMS96]).

References

[A97] F. Ablayev, *Randomization and Nondeterminism are Incomparable for Ordered Read-Once Branching Programs*, Proc. ICALP'97, LNCS 1256, Springer, 1997, pp. 195–202.

[AK96] F. Ablayev and M. Karpinski, *On the Power of Randomized Branching Problems*, Proc. ICALP'96, LNCS 1099, Springer, 1996, pp. 348–356; also available as ECCC TR95-054 (1995) at `http://www.eccc.uni-trier.de/eccc/`.

[AK98a] F. Ablayev and M. Karpinski, *On the Power of Randomized Ordered Branching Programs*, ECCC TR98-004 (1998), available at `http://www.eccc.uni-trier.de/eccc/`.

[AK98b] F. Ablayev and M. Karpinski, *A Lower Bound for Integer Multiplication on Randomized Read-Once Branching Programs*, ECCC TR98-011 (1998), available at `http://www.eccc.uni-trier.de/eccc/`.

[AKM98] F. Ablayev, M. Karpinski and R. Mubarakzjanov, *On BPP versus* $NP \cup \mathrm{co}NP$ *for Ordered Read-Once Branching Programs and* AC^0 *class*, 1998, this volume.

[AT97] M. Agrawal and T. Thierauf, *The Satisfiability Problem for Probabilistic Ordered Branching Programs*, ECCC TR97-060 (1997), 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.

[BSSW93] B. Bollig, M. Sauerhoff, D. Sieling and I. Wegener, *Read k-Times Ordered Binary Decision Diagrams-Efficient Algorithms in the Presence of Null-Chains*, Technical Report Nr. 474, Univ. Dortmund, 1993.

[B86] R. Bryant, *Graph-Based Algorithms for Boolean Function Manipulation*, IEEE Trans. Comput., C-35, (8), (1986), pp. 677–691.

[B91] B. Bryant, *On the Complexity of VLSI Implementations and Graph Representations of Boolean Functions with Applications to Integer Multiplication*, IEEE Trans. Comput. **40** (1991), pp. 205–213.

[Br92] R. Bryant, *Symbolic Boolean Manipulation with Ordered Binary Decision Diagrams*, ACM Computing Surveys, **24**, No. 3, (1992), pp. 293–318.

[Bu92] R. Buss, *The Graph of Multiplication is Equivalent to Counting*, Information Processing Letters, **41**, (1992), pp. 199–201.

[G68] R. Gallager, *Information Theory and Reliable Communication*, Wiley, New York, 1968.

[G94] J. Gergov, *Time-Space Tradeoffs for Integer Multiplication on Various Types of Input Oblivious Sequential Machines*, Information Processing Letters **51** (1991), pp. 265–269.

[GKMS96] D. Grigoriev, M. Karpinski, F. Meyer auf der Heide and R. Smolensky, *A Lower Bound for Randomized Algebraic Decision Trees*, Proc. 28th ACM STOC (1996), pp. 612–619; also in Computational Complexity **6** (1997), pp. 357–375.

[J89] S.Jukna, *On the Effect of Null-Chains on the Complexity of Contact Schemes*, Proc. FCT'89, LNCS 380, Springer, 1989, 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.

[J95] S. Jukna, *The Graph of Integer Multiplication is Hard for Read-k-Times Networks*, TR 95-10 Mathematik/Informatik, Univ. of Trier, 1995.

[JRSW97] S. Jukna, A. Razborov, P. Savický, and I. Wegener, *On P versus $NP \cup \text{co}-NP$ for Decision Trees and Read-Once Branching Programs*, Proc. 22th MFCS'98, LNCS 1295, Springer, 1997, pp. 319–326.

[KM98] M. Karpinski, R. Mubarakzjanov, *Some Separation Problems on Randomized OBDDs*, Manuskript, 1998.

[KMW88] M. Krause, C. Meinel and S. Waack, *Seperating the Eraser Turing Machine Classes L_e, NL_e, $co-NL_e$, and P_e*, Proc. MFCS'88, LNCS 324, Springer, 1988, pp. 405–413.

[K97] E. Kushilevitz and N. Nisan, *Communication Complexity*, Cambridge University Press, 1997.

[P95a] S. Ponzio, *A Lower Bound for Integer Multiplication with Read-Once Branching Programs*, Proc. 27th ACM STOC (1995), pp. 130–139.

[P95b] S. Ponzio, *Restricted Branching Programs and Hardware Verification*, Technical Report, MIT/LCS-TR-633, MIT, 1995.

[R91] A. Razborov, *Lower Bounds for Deterministic and Nondeterministic Branching Programs*, Proc. FCT'91, LNCS 529, Springer, 1991, pp. 47–60.

[SZ96] P. Savicky and S. Zak, *A Large Lower Bound for 1-Branching Programs*, ECCC, Revision 01 of TR96-036 (1996), available at `http://www.eccc.uni-trier.de/eccc/`.

[S97a] 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/`.

[S97b] M. Sauerhoff, *On Nondeterminism Versus Randomness for Read-Once Branching Programs*, ECCC, TR97-030, (1997), available at `http://www.eccc.uni-trier.de/eccc/`.

[S98] M. Sauerhoff, *Randomness and Nondeterminism are Incomparable for Read-Once Branching Programs*, ECCC TR98-018, 1998, also see the Correction; available at `http://www.eccc.uni-trier.de/eccc/`.

[T98] J.S. Thathacher, *On Seperating the Read-k-Times Branching Program Hierarchy*, ECCC TR98-002 (1998), available at `http://www.eccc.uni-trier.de/eccc/`; to appear in Proc. 30th ACM STOC (1998).

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

[Y79] A.C. Yao, *Some Complexity Questions Related to Distributive Computing*, Proc. 11th Annual ACM Symposium on the Theory of Computing, (1979), pp. 209-213.

[Y83] A.C. Yao, *Lower Bounds by Probabilistic Arguments*, Proc. 27th Annual IEEE Symposium on Foundations of Computer Science (1983), pp. 420-428.