Electronic Colloquium on Computational Complexity, Report No. 17 (2001)

A Hierarchy Result for Read-Once Branching Programs with Restricted Parity Nondeterminism*

Petr Savický[1,**] and Detlef Sieling[2,***]

[1] Institute of Computer Science, Academy of Sciences of the Czech Republic, Prague, Czech Republic, email: savicky@cs.cas.cz
[2] Universität Dortmund, FB Informatik, LS 2, 44221 Dortmund, Germany, email: sieling@ls2.cs.uni-dortmund.de

Abstract. Restricted branching programs are considered in complexity theory in order to study the space complexity of sequential computations and in applications as a data structure for Boolean functions. In this paper $(\oplus, k)$-branching programs and $(\vee, k)$-branching programs are considered, i.e., branching programs starting with a $\oplus$- (or $\vee$-)node with a fan-out of k whose successors are k read-once branching programs. This model is motivated by the investigation of the power of nondeterminism in branching programs and of similar variants that have been considered as a data structure. Lower bound methods and hierarchy results for polynomial size $(\oplus, k)$- and $(\vee, k)$-branching programs with respect to k are presented.

1 Introduction

Branching Programs or Binary Decision Diagrams are a well-established model for the representation and manipulation of Boolean functions in computer programs and for the investigation of their space complexity. In complexity theory the goal is to prove superpolynomial lower bounds on the size of branching programs for explicitly defined functions, because such lower bounds imply superlogarithmic lower bounds on the sequential space complexity of those functions. However, the best lower bound on the branching program size for explicitly defined functions is due to Nečiporuk [12] and is merely of size $\Omega(n^2/\log^2 n)$. In order to study lower bound methods a lot of restricted variants of branching programs have been introduced and proofs of exponential lower bounds for those restricted variants have been presented. The strongest results in this direction are presented by Ajtai [1] and by Beame, Saks, Sun and Vee [2]. For further references, see [13] and [16].

* An extended abstract of this paper appeared in the Proceedings of the 25th International Symposium on Mathematical Foundations of Computer Science 2000.
** Supported by The Ministry of Education of the Czech Republic, project LN00A056.
*** Supported in part by DFG grant We 1066/9.

ISSN 1433-8092

Several restricted types of branching programs, in particular OBDDs, which are defined below, are used to represent Boolean functions in computer programs for applications like hardware design and verification. In such applications, data structures for Boolean functions are needed that allow to store as many important functions as possible in small space and to manipulate them efficiently. For more information on the application of restricted branching programs as a data structure we refer to Bryant [5, 6] and Wegener [16].

In the present paper, we investigate a generalization of read-once branching programs (see below) obtained by combining k read-once branching programs by a parity or a disjunction. We prove a hierarchy result for these models with respect to k, i.e., we prove for some explicit functions that the size may decrease from exponential to polynomial if k is increased by 1. This result holds for $k \leq (2/3)\log^{1/2} n$.

We recall the definitions of deterministic and nondeterministic branching programs. Let $X = \{x_0, \ldots, x_{n-1}\}$ be a set of Boolean variables. A deterministic branching program over X is a directed acyclic graph. The graph consists of sink nodes without outgoing edges and of internal nodes with a fan-out of 2. Each sink is labeled by $c \in \{0, 1\}$. Each internal node v is labeled by a variable from X and has an outgoing 0-edge and an outgoing 1-edge. Furthermore the branching program has a source node, i.e., a node without incoming edges. The function represented by the branching program is evaluated in the following way: For some input $a = (a_0, \ldots, a_{n-1})$ the evaluation starts at the source. At each internal node v labeled by x_i the computation proceeds to the successor of v that is reached via the a_i-edge leaving v. The label of the sink that is finally reached is equal to value of the represented function on the input a. The path that is followed for the input a is called the computation path for a.

In a read-once branching program on each path from the source to a sink each variable may be tested at most once. An OBDD (Ordered Binary Decision Diagram) is a read-once branching program where an ordering of the variables is fixed and during each computation the variables are tested according to this ordering. OBDDs have been proposed by Bryant [5] as a data structure for the representation and manipulation of Boolean functions.

A nondeterministic read-once branching program may contain "guessing" nodes, i.e., nodes not labeled by any variable and with an arbitrary number of outgoing edges. Then there may be multiple computation paths for the same input, and an input is accepted, i.e. the value of the represented function is 1, if and only if there is an accepting path for it, i.e., a path leading to the 1-sink. A parity read-once branching program is a nondeterministic read-once branching program with the parity acceptance mode, i.e., an input is accepted, iff there is an odd number of accepting paths for it. For more details on the different variants of nondeterminism in branching programs we refer to Meinel [11].

In the present paper, we consider $(\oplus, k)$-branching programs. The source of such a branching program is a nondeterministic node (labeled by $\oplus$) with a fan-out of k and parity acceptance mode. The k successors of the source are deterministic read-once branching programs $P_1, \ldots, P_k$. The semantics of such

a branching program is defined in a straightforward way: It computes the value 1 for some input a iff an odd number of the read-once branching programs $P_1, \ldots, P_k$ compute the value 1 for a. Similarly, we define $(\vee, k)$-branching programs. Now the source is a node labeled by $\vee$ with a fan-out of k, where the k outgoing edges point to deterministic read-once branching programs $P_1, \ldots, P_k$. The value 1 is computed for the input a if at least one of the branching programs $P_1, \ldots, P_k$ computes a 1 for a.

The results of Jukna [8], Krause, Meinel and Waack [10] and Borodin, Razborov and Smolensky [4] imply exponential lower bounds for $(\vee, k)$-branching programs. In order to prove lower bounds for $(\oplus, k)$-branching programs one may transform a given $(\oplus, k)$-branching program into a syntactic read-k-times branching program and apply the lower bound methods of Borodin, Razborov and Smolensky [4]. A syntactic read-k-times branching program has the restriction that on each computation path each variable may be tested at most k times. The transformation of a $(\oplus, k)$-branching program consisting of $P_1, \ldots, P_k$ into a read-k-times branching program is straightforward; it suffices to combine one copy of P_1 and two copies of $P_2, \ldots, P_k$ in such a way that the parity of the results of $P_1, \ldots, P_k$ is computed.

In the present paper, we provide a new method to prove exponential lower bounds for $(\oplus, k)$-branching programs and $(\vee, k)$-branching programs and prove a hierarchy result for these two models. The hierarchy result means that we present a function with polynomial size $(\oplus, k+1)$-branching programs but only exponential size $(\oplus, k)$-branching programs (and a different function proving a similar statement for $(\vee, k)$-branching programs). By de Morgan's rules the hierarchy result for $(\vee, k)$-branching programs implies a similar hierarchy result for $(\wedge, k)$-branching programs.

Our result generalizes the hierarchy results for $(\vee, k)$-OBDDs due to Bollig and Wegener [3] and Sauerhoff [14]. A $(\vee, k)$-OBDD is a branching program with a $\vee$-node at the source with k outgoing edges pointing to OBDDs $P_1, \ldots, P_k$ (with possibly different variable orderings). Bollig and Wegener [3] and Sauerhoff [14] provided functions with polynomial size $(\vee, k+1)$-OBDDs but only exponential size $(\vee, k)$-OBDDs.

The motivation to consider $(\vee, k)$-OBDDs was given by Jain, Bitner, Fussell and Abraham [7] who suggested to use so-called Partitioned BDDs, which are in fact restricted $(\vee, k)$-OBDDs, as a data structure for Boolean functions. Another work considering restricted nondeterminism is the due to Sauerhoff [15]. He shows that restricting nondeterminism to the source of a nondeterministic OBDD may cause an exponential blow-up of the size compared with ordinary nondeterministic OBDDs.

The paper is organized as follows. In the following section we describe the general lower bound methods for $(\oplus, k)$- and $(\vee, k)$-branching programs. In Section 3 we show how to apply these methods to particular functions and in Section 4 we prove the hierarchy results.

2 The Lower Bound Method

We first describe the lower bound method for $(\oplus, k)$-branching programs. The method is applicable to all (m, k)-full-degree functions, defined in Definitions 1 and 2. The lower bound for such functions is stated in Theorem 3. At the end of this section we show how to adapt this lower bound method to $(\vee, k)$-branching programs. The lower bound is shown for (m, k)-full-sensitive functions (Definition 8, Theorem 9). In the following, let $X = \{x_0, \ldots, x_{n-1}\}$ denote the set of variables.

Definition 1. Let $A \subseteq X$. A mapping $\phi : \{0,1\}^d \to \{0,1\}^A$ is called a projection of degree d, if each of the $|A|$ coordinates of $\phi(y_1, \ldots, y_d)$ is defined by a constant or a literal in one of the variables y_i, $i = 1, \ldots, d$, and, moreover, each of the variables $y_1, \ldots, y_d$ is used (positively or negatively) in at least one of the coordinates.

Definition 2. A Boolean function f is called (m, k)-full-degree, if the following is satisfied. For any partition of its variables into subsets A, B, where $|A| \leq m$, and every projection $\phi : \{0,1\}^d \to \{0,1\}^A$ of degree $d \leq k$, there is a setting b to the variables B, such that substituting $\phi(y_1, \ldots, y_d)$ for the variables in A and b for the variables in B leads to a function $f(\phi(y_1, \ldots, y_d), b)$, which is an $\mathbf{F}_2$-polynomial of degree d in the variables $y_1, \ldots, y_d$.

Let us point out that the (m, k)-full-degree property generalizes the m-mixed property introduced by Jukna [6], since a function is m-mixed if and only if it is $(m, 1)$-full-degree. To see this, note that if $d = 1$, then it is only required that $f(\varphi(y_1), b)$ is non-constant. The following theorem will be applied in situations, where $k = \Theta(\log n)$ and $m/k^2 4^k = \Omega(n^\varepsilon)$.

Theorem 3. *If a Boolean function f of n variables is (m, k)-full-degree, then each $(\oplus, k)$-branching program for f has at least $2^{\Omega(m/k^2 4^k) - \log n}$ nodes.*

Proof. Let f be (m, k)-full-degree, and let a $(\oplus, k)$-branching program P for f be given. Let P consist of the read-once branching programs $P_1, \ldots, P_k$. In the following, we assume that $P_1, \ldots, P_k$ are complete read-once branching programs, i.e., on each computation path each variable is tested exactly once. Since making read-once branching programs complete increases the size by a factor of at most $O(n)$, the lower bound $2^{m/k^2 4^k - 1}$ on the total size of the complete branching program, which we prove in the following, implies the claimed lower bound.

Let $t = \lfloor m/k \rfloor$, and for $i \in \{1, \ldots, k\}$ let V_i be the set of all nodes on the $(t+1)$-th level of P_i, i.e. the nodes that are reached after t tests have been performed. For every input a k-tuple $(v_1, \ldots, v_k) \in V_1 \times \cdots \times V_k$ of nodes is reached. Now, let $(v_1, \ldots, v_k)$ be fixed. Since the read-once branching programs $P_1, \ldots, P_k$ are complete, on each path from the source of P_i to v_i the same set X_i of variables is tested. Let $A = \bigcup_{i=1}^k X_i$ and let $B = X - A$. By the choice of v_i we have $|X_i| = t$ and $|A| \leq m$. Let T be the set of all settings of the variables

in A for which $v_1, \ldots, v_k$ are reached. We are going to prove the upper bound $2^{|A|+1}/2^{t/4^k}$ on the size of T. Any upper bound U on the size of T implies the lower bound $2^{|A|}/U$ on the number of tuples $(v_1, \ldots, v_k)$. Since the total size of the branching program is at least the kth root of the number of such tuples, the claimed lower bound follows.

Let us remark that, if T is large, after reaching v_i the branching program P_i "forgets much information" about the values of the bits read before. We show that if T is large enough, then it contains a subset of size 2^d, $d \leq k$, on which P can compute only polynomials of degree at most $d-1$. This contradicts the assumption that the computed function is (m,k)-full-degree. The critical subset used for this is an image of an appropriate projection with the following property.

Definition 4. A projection $\phi : \{0,1\}^d \to \{0,1\}^A$ is called a covering projection for sets $X_1, \ldots, X_k$ if for every $i = 1, \ldots, k$, there is a variable among $y_1, \ldots, y_d$ such that all its occurrences (negative and positive) are only used to determine the values of X_i-variables in the output of ϕ.

We split the proof of the upper bound on the size of T into two lemmas. If $|T|$ is large, the first lemma guarantees the existence of a suitable covering projection. By the second lemma, this implies that the computed function is not an (m,k)-full-degree function in contradiction to the assumptions of the theorem. Hence, the two lemmas imply the upper bound $2^{|A|+1}/2^{t/4^k}$ on $|T|$, which completes the proof of Theorem 3 as mentioned above.

Lemma 5. *If $|T| \geq 2^{|A|+1}/2^{t/2^{2k}}$, then there is a covering projection ϕ of some degree d, $1 \leq d \leq k$, such that $\phi(\{0,1\}^d) \subseteq T$.*

Lemma 6. *Let ϕ be a covering projection of degree $d \leq k$, and let $\phi(\{0,1\}^d) \subseteq T$. For each setting b of the variables in B the following holds: If the variables in A are substituted in P by $\phi(y_1, \ldots, y_d)$ and the variables in B are substituted by b, the represented function is a polynomial of degree at most $d-1$ over $y_1, \ldots, y_d$.*

Proof. We first consider the effect of substituting the variables in A by $\phi(y)$ and the variables in B by b on the function represented by the read-once branching program P_i. Let $P_i(\phi(y), b)$ denote the result of this substitution. All the variables tested on paths from the source to v_i belong to A. Since $\phi(\{0,1\}^d) \subseteq T$, for each setting of the y-variables the computation of P_i goes through the node v_i. Let y_j be the variable whose occurrences in ϕ only determine X_i-variables. Then the computation of $P_i(\phi(y), b)$ does not test the variable y_j at v_i or after v_i, i.e., the function computed at v_i does not essentially depend on y_j. It follows that the function computed by P_i is a polynomial of degree at most $d-1$. Then also the function represented by P is a polynomial of degree at most $d-1$, since it is the parity of the functions represented by P_i for $i = 1, \ldots, k$. □

Proof of Lemma 5. In order to construct a covering projection we first select subsets $A_1, \ldots, A_d$ of $X_1, \ldots, X_k$. Later on all variables in A_j are determined by y_j. Different sets X_i and $X_{i'}$ may share the same subset A_j. Then we inductively

construct the correspondence between the variables in A_j and y_j. Finally, by a counting argument we show that this is possible in such a way that $\phi(\{0,1\}^k) \subseteq T$.

Since the set X_i contains at most 2^{k-1} cells of the Venn diagram of the sets $X_1, \ldots, X_k$, we may choose for each set X_i a cell contained in X_i of size at least $|X_i|/2^{k-1} = t/2^{k-1}$. The same cell may be used for two different sets X_i. Let $A_1, \ldots, A_d$ be the list of the distinct selected cells. These sets are disjoint, each has size at least $t/2^k$ and for each X_i there is a set $A_{j(i)}$ among $A_1, \ldots, A_d$ such that $A_{j(i)} \subseteq X_i$. Let $A_{d+1} = A - (A_1 \cup \ldots \cup A_d)$.

We are going to construct a covering projection by considering special rectangular sets. Let $s \in \{0, \ldots, d\}$ and let

$$D_s = \mathcal{A}_1^{(2)} \times \cdots \times \mathcal{A}_s^{(2)} \times \mathcal{A}_{s+1} \times \cdots \times \mathcal{A}_{d+1},$$

where $\mathcal{A}_i$ is the set of all settings of the variables of A_i, and $\mathcal{A}_i^{(2)}$ is the set of all unordered pairs of such settings. The elements of D_s are $(d+1)$-tuples of the form $(\{a_1, p_1\}, \ldots, \{a_s, p_s\}, \{a_{s+1}\}, \ldots, \{a_{d+1}\})$, where $a_i, p_i \in \{0,1\}^{A_i}$ and $a_i \neq p_i$ for $1 \leq i \leq s$, and $a_i \in \{0,1\}^{A_i}$ for $s+1 \leq i \leq d+1$. We interpret each element of D_s as the product

$$\{a_1, p_1\} \times \cdots \times \{a_s, p_s\} \times \{a_{s+1}\} \times \cdots \times \{a_{d+1}\},$$

which is a set of 2^s settings of the variables in A. We call such sets *rectangular sets* of dimension s.

We may consider elements of T as rectangular sets of dimension 0, i.e. $T \subseteq D_0$. For any $0 \leq s \leq d$, let $T_s \subseteq D_s$ be the set of all rectangular sets of dimension s that are subsets of T. In particular, $T_0 = T$. We shall prove that T_d is not empty, provided that all sets A_i, $i = 1, \ldots, d$, are large enough. Then T_d contains all elements of a rectangular set $\{a_1, p_1\} \times \cdots \times \{a_d, p_d\} \times \{a_{d+1}\}$. Let ϕ be the projection defined by

$$\phi(y_1, \ldots, y_d) = (c_1, \ldots, c_d, a_{d+1}), \text{where } c_i = \begin{cases} a_i, \text{ if } y_i = 0, \\ p_i, \text{ if } y_i = 1. \end{cases}$$

The choice of the partition $A_1, \ldots, A_{d+1}$ implies that ϕ is a covering projection. Since ϕ is constructed from a rectangular set in T_d, we have $\phi(\{0,1\}^d) \subseteq T$.

It remains to prove that the set T_d is not empty. Let $\text{density}(T_s) = |T_s|/|D_s|$. The following lemma shows how to obtain lower bounds on the density of T_{s+1} from a lower bound on the density of T_s. By applying this lemma inductively, one can obtain that the density of T_d is larger than 0, i.e., that T_d is not empty.

Lemma 7. *Let $s \in \{0, \ldots, d-1\}$, let $a = |\mathcal{A}_{s+1}|$ and let $\varepsilon = \text{density}(T_s)$. Then*

$$\text{density}(T_{s+1}) \geq \varepsilon^2 \left(1 - \frac{1}{\varepsilon a}\right).$$

Proof. Partition $D_s = \mathcal{A}_1^{(2)} \times \cdots \times \mathcal{A}_s^{(2)} \times \mathcal{A}_{s+1} \times \cdots \times A_{d+1}$ into classes of elements that coincide in all coordinates except the $(s+1)$-th one. Each of these

classes has size $a = |\mathcal{A}_{s+1}|$. Let $N = |D_s|/a$ be the number of these classes and let l_i for $i = 1, 2, \ldots, N$ be the size of the intersection of T_s and the ith class. Clearly, $(1/N)\sum_{i=1}^{N} l_i = |T_s|/N = \varepsilon a$. Since there are $\binom{l_i}{2}$ pairs of elements of the ith class, we obtain from the ith class $\binom{l_i}{2}$ elements of T_{s+1}. Furthermore, the size of D_{s+1} is $N\binom{a}{2}$. Hence, we have the estimate

$$\text{density}(T_{s+1}) = \frac{1}{N\binom{a}{2}} \sum_{i=1}^{N} \binom{l_i}{2} \geq \frac{1}{\binom{a}{2}} \binom{\varepsilon a}{2} \geq \frac{\varepsilon a(\varepsilon a - 1)}{a^2} = \varepsilon^2 \left(1 - \frac{1}{\varepsilon a}\right),$$

where the first inequality follows from the convexity of $\binom{x}{2}$. □

Since we apply Lemma 7 only for $s \in \{0, \ldots, d-1\}$, in all applications of the lemma we have $a = 2^{|A_{s+1}|} \geq 2^{t/2^k}$.

Let ε_0 be the density of T_0 $(= T)$. By the assumption of Lemma 5 we have $|T| \geq 2^{|A|+1}/2^{t/2^{2k}}$ and, therefore, $\varepsilon_0 \geq \frac{2}{2^{t/2^{2k}}}$. Let ε_s be the lower bound on the density of T_s that we obtain after the sth application of Lemma 7. Clearly, $\varepsilon_0 a \geq 2 \cdot 2^{t/2^k - t/2^{2k}} \geq 2$. Hence, the first application of Lemma 7 yields $\text{density}(T_1) \geq \varepsilon_1 \geq \varepsilon_0^2/2 = 2(\varepsilon_0/2)^2$. It is easy to verify that $\varepsilon_1 a \geq 2$ and we can estimate the density after the second application of the lemma in a similar way. In general, after the sth application of the lemma, we obtain $\text{density}(T_s) \geq \varepsilon_s \geq 2(\varepsilon_0/2)^{2^s}$. For every $s < d$, we have $\varepsilon_s a \geq 2 \cdot 2^{t/2^k - t/2^{2k-s}} \geq 2$, which allows to perform the next step. Hence, after d applications of Lemma 7 we obtain a positive lower bound on the density of T_d, which implies the existence of a covering projection. □

The proofs of Lemmas 5–7 complete the proof of Theorem 3. □

Finally, we present the adaptation of the lower bound method to $(\vee, k)$-branching programs. The lower bound method can be applied to functions that are (m, k)-full-sensitive – a property that is defined in the following definition.

Definition 8. A function g on d variables is called full-sensitive, if there is an input c for g such that $g(c) = 1$ and the shortest prime implicant covering c has length d.

A function f is called (m, k)-full-sensitive, if the following is satisfied. For any partition of its variables into subsets A, B, where $|A| \leq m$, and every projection $\phi : \{0,1\}^d \to \{0,1\}^A$ of degree $d \leq k$, there is a setting b to the variables B such that substituting $\phi(y_1, \ldots, y_d)$ for the variables in A and b for the variables in B leads to a full-sensitive function $f(\phi(y_1, \ldots, y_d), b)$.

It is easy to see that a full-sensitive function g has the following property: If c is the input only covered by a prime implicant of length d, then for all inputs c' obtained from c by flipping one variable it holds that $g(c') = 0$. Similarly to the case of (m, k)-full-degree functions the property (m, k)-full-sensitive generalizes the notion m-mixed: A function is m-mixed iff it is $(m, 1)$-full-sensitive.

Theorem 9. *If a Boolean function f of n variables is (m,k)-full-sensitive, then each $(\vee,k)$-branching program for f has at least $2^{\Omega(m/k^2 4^k)-\log n}$ nodes.*

Proof. Let f be (m,k)-full-sensitive, and let a $(\vee,k)$-branching program P for f be given. As in the proof of Theorem 3 we assume that P consists of the complete read-once branching programs $P_1,\ldots,P_k$, and we prove the lower bound $2^{m/k^2 4^k-1}$ on the total size of the complete branching program. Let t and $V_1,\ldots,V_k$ be defined as in the proof of Theorem 3. Moreover, let any selection of elements $v_1,\ldots,v_k$ from $V_1,\ldots,V_k$ and corresponding sets $X_1,\ldots,X_k,A,B$ and T be also as in the proof of Theorem 3.

If $|T| \geq 2^{|A|+1}/2^{t/4^k}$, then, by Lemma 5, there is a covering projection ϕ of degree d such that $d \leq k$ and $\phi(\{0,1\}^d) \subseteq T$. Using Lemma 10 below, this contradicts the assumption that P computes an (m,k)-full-sensitive function. Altogether, we have the upper bound $2^{|A|+1}/2^{t/4^k}$ on the size of T, which, by the same arguments as in the proof of Theorem 3, implies the claimed lower bound on the total size of the read-once branching programs P_i.

Lemma 10. *Let ϕ be a covering projection of degree $d \leq k$, and let $\phi(\{0,1\}^d) \subseteq T$. For each setting b of the variables in B the followings holds: If the variables in A are substituted in P by $\phi(y_1,\ldots,y_d)$ and the variables in B are substituted by b, the resulting function $P(\phi(y),b)$ is not full-sensitive.*

Proof. If $P(\phi(y),b)$ is the zero function, it is not full-sensitive. Otherwise, let c be any setting of the y-variables such that $P(\phi(c),b)=1$. This implies that there is some $i \in \{1,\ldots,k\}$ such that $P_i(\phi(c),b)=1$. Let y_j be the variable whose occurrences in $\phi(y)$ only belong to X_i. Then, the computation of $P_i(\phi(c),b)$ does not test the variable y_j at v_i or after v_i. Let c^* be the input obtained from c by flipping the value of y_j. Since $\phi(c^*) \in T$, the computation of $P_i(\phi(c^*),b)$ goes through the node v_i and continues exactly as the computation of $P_i(\phi(c),b)$. Consequently, in the functions $P_i(\phi(y),b)$ and $P(\phi(y),b)$, the input c is covered by an implicant of length at most $d-1$. Since this holds for every c satisfying $P(\phi(c),b)=1$, $P(\phi(y),b)$ is not full-sensitive. □

This concludes the proof of Theorem 9. □

3 The Lower Bounds

Let us start with the definitions of the functions which we use. The considered functions are multipointer functions where the pointers are obtained similarly to functions used in [9]. Let n be a power of 2 and k an integer that possibly depends on n. In order to compute $f_n^k(x_0,\ldots,x_{n-1})$ and $g_n^k(x_0,\ldots,x_{n-1})$ the input $X=\{x_0,\ldots,x_{n-1}\}$ is partitioned into $k(k+1)$ blocks $B_{i,j}$, where $i \in \{1,\ldots,k+1\}$ and $j \in \{1,\ldots,k\}$ and, if necessary, to some remaining variables. Each block $B_{i,j}$ consists of $\log n$ subblocks of size

$$s = \left\lfloor \frac{n}{k(k+1)\log n} \right\rfloor .$$

Each of the blocks $B_{i,j}$ determines a binary representation of an integer $p_{i,j}$, $0 \leq p_{i,j} \leq n-1$. Each of the $\log n$ bits of $p_{i,j}$ is determined by the majority of the s bits in one of the $\log n$ subblocks of the block $B_{i,j}$.

The function $f_n^k(x)$ takes the value 1 iff

1. $\forall j \in \{1,\ldots,k\} : p_{1,j} = \cdots = p_{k+1,j}$ and
2. $x_{p_{1,1}} \wedge \ldots \wedge x_{p_{1,k}} = 1$.

The function $g_n^k(x)$ takes the value 1 iff

1. $\forall j \in \{1,\ldots,k\} : p_{1,j} = \cdots = p_{k+1,j}$ and
2. $x_{p_{1,1}} \oplus \cdots \oplus x_{p_{1,k}} = 1$.

Let us point out that the conditions 1 in these definitions are only needed to prove the upper bound of the hierarchy result. For the lower bound alone, the functions $x_{p_{1,1}} \wedge \ldots \wedge x_{p_{1,k}}$ and $x_{p_{1,1}} \oplus \cdots \oplus x_{p_{1,k}}$ are sufficient. Moreover, the lower bounds could be made slightly larger, if we don't use blocks $B_{i,j}$ for $i \geq 2$ at all and split all the variables among blocks $B_{1,j}$.

For each $i = 1,\ldots,k+1$, the blocks $B_{i,j}$ for $j = 1,\ldots,k$ determine a collection of k pointers. The functions may take the value 1 only if all of these $k+1$ collections coincide. Note that the fact that two pointers coincide does not imply that the blocks from which the pointers are derived are identical. The lower bound results use the following lemma.

Lemma 11. *If $c \in \{0,1\}$ is a constant and some set A of at most $s/2-2$ variables is selected, then for each choice $\bar{p}_1,\ldots,\bar{p}_d \in \{0,\ldots,n-1\}$, where $d \leq k$, one can find settings of the variables not in A such that*

1. *The pointers $p_{i,j}$ for all $i = 1,\ldots,k+1$ and $j = 1,\ldots,k$ do not depend on the variables in A.*
2. *For all $i = 1,\ldots,k+1$ and $j \leq d$, we have $p_{i,j} = \bar{p}_j$ and for all $i = 1,\ldots,k+1$ and $j \geq d+1$, $p_{i,j}$ is an index of the same variable set to the constant c.*

Proof. Set one of the variables not in A to c and let q be its index. Define, moreover, $\bar{p}_j = q$ for all $j = d+1,\ldots,k$. Since $|A| \leq s/2-2$, less than one half of the bits of each of the blocks is contained in $A \cup \{q\}$. Thus, setting the remaining bits may force the majority of the bits in each block to any predetermined value independently from the values of the bits in A.

We use this to force the majority of the bits in $B_{i,j}$ for all $i = 1,\ldots,k+1$ and $j = 1,\ldots,k$ according to the binary representation of $\bar{p}_1,\ldots,\bar{p}_k$. □

In the following theorems we state the lower bounds for the above defined functions.

Theorem 12. *Each $(\oplus,k)$-branching program for f_n^k has at least*

$$2^{\Omega\left(\frac{n}{k^4 4^k \log n}\right) - \log n}$$

nodes. This number grows exponentially, if $k \leq (1/2-\gamma)\log n$ for some $\gamma > 0$.

Proof. By Theorem 3 it suffices to prove that f_n^k is $(s/2-2,k)$-full-degree. Let $A \subseteq X$ such that $|A| \leq s/2-2$. Let $d \leq k$ and let $\phi : \{0,1\}^d \to \{0,1\}^A$ be any projection of degree $d \leq k$. Since by the definition of the projections each variable $y_1, \ldots, y_d$ occurs at least once in the projection, we can define $\bar{p}_1, \ldots, \bar{p}_d$ in such a way that $x_{\bar{p}_i}$ is an occurrence of y_i or $\neg y_i$. Using Lemma 11 with $c = 1$, we can find a setting b of the variables in $B = X - A$ in such a way that the resulting function $f_n^k(\phi(y), b)$ is the conjunction of the y-variables or their negations, which is an $\mathbf{F}_2$-polynomial of degree d. □

Theorem 13. *Each $(\vee,k)$-branching program for g_n^k has at least*

$$2^{\Omega\left(\frac{n}{k^4 4^k \log n}\right) - \log n}$$

nodes. This number grows exponentially, if $k \leq (1/2-\gamma)\log n$ for some $\gamma > 0$.

Proof. By Theorem 9 it suffices to prove that g_n^k is $(s/2-2,k)$-full-sensitive. Let $A \subseteq X$ such that $|A| \leq s/2-2$. Let $\phi : \{0,1\}^d \to \{0,1\}^A$ be a projection of degree $d \leq k$. Again let $\bar{p}_1, \ldots, \bar{p}_d$ be such that $x_{\bar{p}_i}$ is an index of an occurrence of y_i or its negation. Using Lemma 11 with $c = 0$, one can find a setting b of the variables in $B = X - A$ in such a way that the resulting function $g_n^k(\phi(y), b)$ is the parity of all the y-variables or their negations, which is obviously a full-sensitive function. □

4 The Hierarchy Result

In order to obtain the hierarchy result we first prove a polynomial upper bound on the size $(\oplus, k+1)$- and $(\vee, k+1)$-branching programs for f_n^k and g_n^k, resp., where k is a constant.

Theorem 14. *There are $(\oplus, k+1)$-branching programs for the function f_n^k and $(\vee, k+1)$-branching programs for the function g_n^k of size $O(n^{k+2})$. These branching programs even consist of $k+1$ OBDDs.*

Proof. We start with the construction of a $(\oplus, k+1)$-branching program P for f_n^k. We call the set of input variables contained in $B_{i,1}, B_{i,2}, \ldots, B_{i,k}$ the ith sector of the input. We first describe the OBDDs P_i, $i \in \{1, \ldots, k+1\}$, that P consists of. The OBDD P_i works in the following way. It first reads the subblocks in the ith sector and computes for each subblock the majority of its variables. For each subblock the majority is stored. Storing means that the computation paths for the inputs x and y do not join before a sink, if the majority of some subblock in x is 1, while the majority of the same subblock in y is 0. For computing the majority of s variables width s is sufficient. Since for each of the $k \log n$ subblocks the majority of s variables is computed, width $O(sn^k)$ is sufficient. In particular, after reading the ith sector all pointers derived from the ith sector are known.

If there is a pointer addressing a bit in ith sector, the OBDD P_i computes a 0. If there is some $j < i$ such that no pointer addresses any bit of jth sector, also a 0 is computed. Otherwise i is the smallest number such that there is no pointer addressing a bit of the ith sector. Then P_i sequentially reads the other sectors and compares the stored pointers $p_{i,1}, \ldots, p_{i,k}$ with the corresponding pointers of the other sectors in order to test condition 1 of the definition of f_n^k. If this condition is not fulfilled, a 0 is computed. Since the pointers are stored, it is possible to compute the conjunction of the addressed bits during the comparison of the pointers.

The correctness of P follows from the observation that exactly one of the branching programs P_i, namely that where i is the smallest number of a sector without an addressed bit, computes the correct function value, while all P_j, where $j \neq i$, compute a 0. The branching program P_i is able to compute the function value since it has not read any of the addressed bits before it knows all pointers.

The width of P_i is bounded by $O(sn^k)$ and, hence, its size is bounded by $O(sn^{k+1})$. The total size of P is bounded by $O(ksn^{k+1}) = O(n^{k+2})$. It is easy to construct P_i in such a way that a variable ordering is respected, i.e. such that P_i is an OBDD.

For the function g_n^k and $(\vee, k+1)$-branching programs the same arguments work with the only exception that the parity of the addressed bit has to be computed instead of the conjunction. This may increase the width by a factor of at most 2. □

In order to state the hierarchy result, let P-$(\oplus, k)$-BP denote the set of all Boolean functions with polynomial size $(\oplus, k)$-branching programs, and let P-$(\vee, k)$-BP and P-$(\wedge, k)$-BP be defined similarly.

Theorem 15. *If $k \leq (2/3)\log^{1/2} n$, it holds that*

$$\begin{aligned} &\text{P-}(\oplus, k)\text{-BP} \subsetneq \text{P-}(\oplus, k+1)\text{-BP}, \\ &\text{P-}(\vee, k)\text{-BP} \subsetneq \text{P-}(\vee, k+1)\text{-BP and} \\ &\text{P-}(\wedge, k)\text{-BP} \subsetneq \text{P-}(\wedge, k+1)\text{-BP}. \end{aligned}$$

Proof. The third inequality follows from the second one by de Morgan's rules. For constant k the first and second inequalities follow directly from Theorems 12–14. In order to prove the hierarchy results for nonconstant k we apply a padding argument. The following arguments are given for the hierarchy of $(\oplus, k)$-branching programs, but they work for the hierarchy of $(\vee, k)$-branching programs in the same way. Let $\tilde{n} = \lceil n^{1/k} \rceil$. We define the function $h_n^k(x_0, \ldots, x_{n-1}) = f_{\tilde{n}}^k(x_0, \ldots, x_{\tilde{n}-1})$. This means we consider the same function as above with a large number of dummy variables. The upper bound for $(\oplus, k+1)$-branching programs for h^k is then $O(\tilde{n}^{k+2}) = O(n^2)$. The lower bound is

$$2^{\Omega\left(\frac{\tilde{n}}{k^4 2^{2k} \log \tilde{n}}\right) - \log n} = 2^{\Omega\left(2^{\frac{\log n}{k} - 4\log k - 2k - \log\log n}\right) - \log n}.$$

The last term is superpolynomial, since it is bounded below by $2^{2^{\Omega(\log^{1/2} n)}} = 2^{\log^{\alpha(n)} n}$ for a function α such that $\alpha(n) \to \infty$. □

Acknowledgement. The authors are grateful to Jiří Sgall for pointing out the connection between $(\oplus, k)$-branching programs and syntactic read-k-times branching programs.

References

1. Ajtai, M. (1999). A non-linear time lower bound for Boolean branching programs. In *Proc. of 40th Symposium on Foundations of Computer Science*, 60–70.
2. Beame, P., Saks, M., Sun, X. and Vee, E. (2000). Super-linear time-space tradeoff lower bounds for randomized computation. To appear in *Proc. of 41st Symposium on Foundations of Computer Science*.
3. Bollig, B. and Wegener, I. (1999). Complexity theoretical results on partitioned (nondeterministic) binary decision diagrams. *Theory of Computing Systems* 32, 487–503.
4. Borodin, A., Razborov, A. and Smolensky, R. (1993). On lower bounds for read-k-times branching programs. *Computational Complexity* 3, 1–18.
5. Bryant, R.E. (1986). Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers* 35, 677–691.
6. Bryant, R.E. (1992). Symbolic Boolean manipulation with ordered binary-decision diagrams. *ACM Computing Surveys* 24, 293–318.
7. Jain, J., Bitner, J., Fussell, D.S. and Abraham, J.A. (1992). Functional partitioning for verification and related problems. *Brown MIT VLSI Conference*, 210–226.
8. Jukna, S. (1988). Entropy of contact circuits and lower bounds on their complexity. *Theoretical Computer Science* 57, 113–129.
9. Jukna, S., Razborov, A., Savický, P. and Wegener, I. (1997). On P versus $NP \cap co\text{-}NP$ for decision trees and read-once branching programs. In *Proc. of Mathematical Foundations of Computer Science*, Springer, Lecture Notes in Computer Science 1295, 319–326.
10. Krause, M., Meinel, C. and Waack, S. (1991). Separating the eraser Turing machine classes L_e, NL_e, $co\text{-}NL_e$ and P_e. *Theoretical Computer Science* 86, 267–275.
11. Meinel, C. (1990). Polynomial size Ω-branching programs and their computational power. *Information and Computation* 85, 163–182.
12. Nečiporuk, E. I. (1966). A Boolean function. *Soviet Mathematics Doklady* 7, 999–1000.
13. Razborov, A. A. (1991). Lower bounds for deterministic and nondeterministic branching programs. In *Proc. of Fundamentals in Computing Theory*, Springer, Lecture Notes in Computer Science 529, 47–60.
14. Sauerhoff, M. (1999). An improved hierarchy result for partitioned BDDs. To appear in *Theory of Computing Systems*.
15. Sauerhoff, M. (1999). Computing with restricted nondeterminism: the dependence of the OBDD size on the number of nondeterministic variables. In *Proc. of 19th Conference on Foundations of Software Technology and Theoretical Computer Science*, Springer, Lecture Notes in Computer Science 1738, 342–355.
16. Wegener, I. (2000). *Branching Programs and Binary Decision Diagrams—Theory and Applications*. SIAM Monographs on Discrete Mathematics and Its Applications.

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'