Electronic Colloquium on Computational Complexity, Report No. 19 (1999)

Lower Bounds for Linear Transformed OBDDs and FBDDs

Detlef Sieling[1]
FB Informatik, LS 2, Univ. Dortmund
44221 Dortmund, Fed. Rep. of Germany

Abstract: Linear Transformed Ordered Binary Decision Diagrams (LTOBDDs) have been suggested as a generalization of OBDDs for the representation and manipulation of Boolean functions. Instead of variables as in the case of OBDDs parities of variables may be tested at the nodes of an LTOBDD. By this extension it is possible to represent functions in polynomial size that do not have polynomial size OBDDs, e.g., the characteristic functions of linear codes. In this paper lower bound methods for LTOBDDs and some generalizations of LTOBDDs are presented and applied to explicitly defined functions. By the lower bound results it is possible to compare the set of functions with polynomial size LTOBDDs and their generalizations with the set of functions with polynomial size representations for many other restrictions of BDDs.

1 Introduction

Branching Programs or Binary Decision Diagrams (BDDs) are a representation of Boolean functions with applications in complexity theory and in programs for hardware design and verification as well. In complexity theory branching programs are considered as a model of sequential computation. The goal is to prove upper and lower bounds on the branching program size for particular Boolean functions in order to obtain upper and lower bounds on the sequential space complexity of these functions. Since for unrestricted branching programs no method to obtain exponential lower bounds is known, a lot of restricted variants of branching programs has been considered; for an overview see e.g. Razborov [21].

In hardware design and verification data structures for the representation and manipulation of Boolean functions are needed. The most popular data structure are Ordered Binary Decision Diagrams (OBDDs), which were introduced by Bryant [3]. They allow the compact representation and the efficient manipulation of many important functions. However, there are a lot of other important functions for which OBDDs are much too large. For this reason a large number of generalizations of OBDDs has been proposed as a data structure for Boolean functions. Many of these generalizations are restricted branching programs that are also investigated in complexity theory. Hence, the lower and upper bound results and methods from complexity theory are also useful in order to compare the classes of functions for which the different extensions of OBDDs have polynomial size.

[1]Supported in part by DFG grant We 1066/8.

ISSN 1433-8092

In this paper we consider several variants of branching programs that are obtained by introducing linear transformations in OBDDs or generalized variants of OBDDs. In order to explain the differences between ordinary OBDDs and OBDDs with linear transformations we first repeat the definition of BDDs/branching programs and of OBDDs. A Binary Decision Diagram (BDD) or Branching Program for a function $f(x_1, \ldots, x_n)$ is a directed acyclic graph with one source node and two sinks. The sinks are labeled by the Boolean constants 0 and 1. Each internal node is labeled by a variable x_i and has an outgoing 0-edge and an outgoing 1-edge. For each input $a = (a_1, \ldots, a_n)$ there is a computation path from the source to a sink. The computation path starts at the source and at each internal node labeled by x_i the next edge of the computation path is the outgoing a_i-edge. The label of the sink reached by the computation path for a is equal to $f(a)$. The size of a branching program or BDD is the number of internal nodes. In OBDDs the variables have to be tested on each computation path at most once and according to fixed ordering.

In the following we call an expression $x_{i(1)} \oplus \cdots \oplus x_{i(k)}$ a *linear test.* A *generalized variable ordering* over $x_1, \ldots, x_n$ is a sequence of n linear independent linear tests. In Linear Transformed OBDDs (LTOBDDs) the internal nodes may be labeled by linear tests instead of single variables as in the case of OBDDs. However, on each computation path the tests have to be arranged according to a fixed generalized variable ordering. The function f represented by an LTOBDD is evaluated in the obvious way: The computation path for some input $a = (a_1, \ldots, a_n)$ starts at the source. At an internal node labeled by $x_{i(1)} \oplus \cdots \oplus x_{i(k)}$ the outgoing edge labeled by $a_{i(1)} \oplus \cdots \oplus a_{i(k)}$ has to be chosen. The label of the sink at the end of the computation path is equal to $f(a)$.

In Section 2 we present an alternate definition of LTOBDDs. There we also define several extensions of LTOBDDs. An example of an LTOBDD is shown in the left of Figure 1. We remark that the linear independence of the linear tests of a generalized variable ordering is necessary, since otherwise not all inputs can be distinguished by the LTOBDD and, therefore, not all functions can be represented.

The evaluation of linear tests instead of single variables at the nodes of BDDs was already suggested by Aborhey [1] who, however, only considers decision trees. Linear Transformed OBDDs have been suggested as a generalization of OBDDs (Meinel, Somenzi and Theobald [19]), since they are a more compact representation of Boolean functions than OBDDs. The results of Bern, Meinel and Slobodová [2] on Transformed BDDs imply that the algorithms for the manipulation of OBDDs can also be applied to LTOBDDs so that existing OBDD packages can easily be extended to LTOBDDs. Furthermore, the well-known sifting algorithm due to Rudell [22] for the computation of (heuristically) good variable orderings for OBDDs can be adapted to compute good generalized variable orderings for LTOBDDs (Meinel, Somenzi and Theobald [19]). Günther and Drechsler [10] present an algorithm for computing optimal generalized variable orderings.

An example that shows the power of LTOBDDs are the characteristic functions of linear codes. For more details about linear codes we refer to MacWilliams and Sloane [18]. It is easy to see that all characteristic functions of linear codes can be represented by LTOBDDs of linear size: In order to check whether a word x belongs to a linear code it suffices to test whether the inner product of x and each row of the parity check matrix of the code is equal to 0. For each row we can choose a linear test that is equal to the inner product of the row and

the input. Since the rows of the parity check matrix are linearly independent, we can choose these linear tests as a generalized variable ordering of an LTOBDD, and an LTOBDD computing the NOR of these linear tests also computes the characteristic function of the code. On the other hand, exponential lower bounds on the size of many restrictions of branching programs are known for the characteristic functions of certain linear codes: Exponential lower bounds for syntactic read-k-times branching programs are proved by Okol'nishnikova [20], for nondeterministic syntactic read-k-times branching programs by Jukna [12], for semantic $(1, +k)$-branching programs by Jukna and Razborov [14], and for $\oplus$OBDDs by Jukna [13]. The definition of $\oplus$OBDDs is given in Section 2. We omit the definitions of the other variants of branching programs since they are not subject of this paper, but we would like to point out that these variants of branching programs are the most powerful ones for which exponential lower bounds can be proved.

The aim of this paper is to present methods to prove exponential lower bounds on the size of LTOBDDs and some generalizations of LTOBDDs. Many lower bounds for restricted BDDs have been proved by arguments based on communication complexity theory. Roughly, the BDD is cut into two parts so that some part of the input is known only in the first part of the BDD and the other part of the input only in the second part of the BDD. If the computation of the considered function requires the exchange of a large amount of information between those two parts of the input, the cut through the BDD and, therefore, also the BDD has to be large. For LTOBDDs this approach is more difficult to apply. If in one part of the LTOBDD $x_1 \oplus x_2$ is tested and in the other part $x_1 \oplus x_3$, one can hardly say that nothing about x_1 is known in one of the parts of the LTOBDD. The main result of this paper is to show how to overcome this problem.

The paper is organized as follows. In Section 2 we repeat the definitions of several variants of BDDs and define the corresponding variants of LTOBDDs. In Section 3 we present lower bound methods for LTFBDDs (i.e., LTOBDDs with a relaxed variable ordering condition) and in Section 4 for $\oplus$LTOBDDs (i.e., nondeterministic OBDDs with parity accepting mode). Finally, we summarize our results and compare the classes of functions with polynomial size LTOBDDs with the corresponding classes for other variants of BDDs.

2 Further Definitions

Before we define some generalizations of LTOBDDs we discuss an alternate definition of LTOBDDs, which is equivalent to that given in the Introduction and will be useful in our lower bound proofs. In order to simplify the notation we always assume that vectors are column vectors and we also use vectors as arguments of functions. Furthermore we only consider vector spaces over $\mathbb{F}_2$. Then an LTOBDD for some function f consists of an OBDD for some function g and a regular matrix A, so that $f(x) = g(A \cdot x)$. In order to illustrate this definition Figure 1 shows an LTOBDD for some function f and an isomorphic OBDD

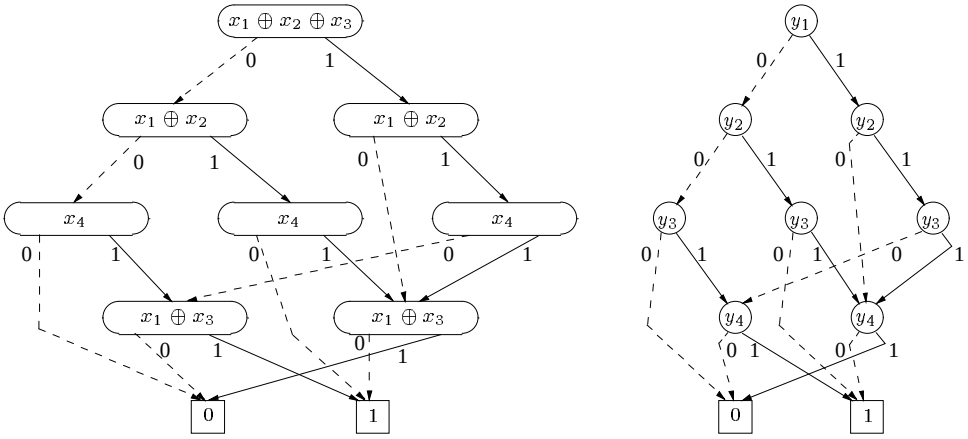

Figure 1: An example of an LTOBDD with the generalized variable ordering $x_1 \oplus x_2 \oplus x_3$, $x_1 \oplus x_2$, x_4, $x_1 \oplus x_3$ for some function f and of an OBDD for some function g so that $f(x) = g(A \cdot x)$.

for some function g, so that $f(x) = g(A \cdot x)$ for

$$A = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}.$$

We now define some generalizations of OBDDs and the linear transformed variants of these generalizations. In FBDDs (Free BDDs, also called read-once branching programs) on each computation path each variable is tested at most once. This property is also called the read-once property. An LTFBDD for some function f consists of an FBDD for some function g and a regular matrix A so that $f(x) = g(A \cdot x)$. If we draw LTFBDDs as FBDDs with linear tests at the internal nodes, we see that at most n different linear tests may occur in an LTFBDD. Another possibility to define LTFBDDs is to allow an arbitrary number of different linear tests. We call the resulting variant of LTFBDDs *strong LTFBDDs*: In a strong LTFBDD the linear tests of each computation path have to be linearly independent. This definition is quite natural, since the term "free" in the name FBDD means that a path leading from the source to some node v can be extended to a computation path (a path corresponding to some input) via the 0-edge leaving v and the 1-edge as well. In a BDD this is obviously equivalent to the read-once property. In linear transformed BDDs this is possible iff on each path the linear tests performed on this path are linearly independent.

Obviously, an LTFBDD is also a strong LTFBDD, while the opposite is not true. We shall even see in the following section that polynomial size strong LTFBDDs are more powerful than polynomial size LTFBDDs so that the name strong LTFBDD is justified.

A nondeterministic OBDD is an OBDD where each internal node may have an arbitrary number of outgoing 0-edges and 1-edges. Hence, for each input there may be more than one computation path. A function represented by a nondeterministic OBDD takes the value 1 on

the input a, if there is at least one computation path for a that leads to the 1-sink. $\oplus$OBDDs are syntactically defined as nondeterministic OBDDs. However, a $\oplus$OBDD computes the value 1 on the input a, if the number of computation paths for a from the source to the 1-sink is odd. We may define nondeterministic LTOBDDs and $\oplus$LTOBDDs by introducing a regular transformation matrix A as described above or by allowing linear tests at the internal nodes, where a generalized variable ordering has to be respected.

The investigation of $\oplus$OBDDs is motivated by polynomial time algorithms for several important operations on Boolean functions, which are presented by Gergov and Meinel [8] and Waack [24]. It is straightforward to extend most of these algorithms to $\oplus$LTOBDDs. We shall see that polynomial size $\oplus$LTOBDDs can represent a larger class of functions than $\oplus$OBDDs. On the other hand, we also obtain exponential lower bounds for $\oplus$LTOBDDs.

Another motivation for lower bound proofs for $\oplus$OBDDs and $\oplus$LTOBDDs is their relationship to other variants of OBDDs that are used in programs for VLSI design. Such a variant are Ordered Functional Decision Diagrams (OFDDs), which were introduced by Kebschull, Schubert and Rosenstiel [16]. OFDDs are syntactically defined as OBDDs, but evaluated in a different way: Each computation path starts at the source. At an internal node labeled by x_i a computation path may proceed via the outgoing 0-edge, if $a_i = 0$, and it may proceed via the outgoing 0-edge *or* the outgoing 1-edge, if $a_i = 1$. Similar to $\oplus$OBDDs $f(a) = 1$ iff the number of computation paths for a to the 1-sink is odd. By the given definition it is easy to observe that OFDDs can be replaced by $\oplus$OBDDs of the same size (Gergov and Meinel [8]). The same holds for so-called OKFDDs (Ordered Kronecker FDDs), which are an extension of OFDDs (Drechsler, Sarabi, Theobald, Becker and Perkowski [7]). In the same way as described above linear transformations can be introduced into OFDDs and OKFDDs, and many of the algorithms on OFDDs and OKFDDs easily generalize to the linear transformed versions. Since linear transformed OFDDs and OKFDDs can be simulated by $\oplus$LTOBDDs without increasing the size, our lower bound for $\oplus$LTOBDDs implies the same lower bound for the linear transformed variants of OFDDs and OKFDDs.

3 Lower Bounds for LTFBDDs and a Comparison of LTFBDDs and Strong LTFBDDs

Lower bounds for FBDDs can be proved by cut-and-paste arguments as first shown by Wegener [25] and Žák [26]. The following lemma describes an extension of the cut-and-paste method that is suitable for LTFBDDs.

Lemma 1: *Let $f : \{0,1\}^n \to \{0,1\}$ and let $k \geq 1$. If for all $k \times n$ matrices D with linearly independent rows, for all $c = (c_0, \ldots, c_{k-1}) \in \{0,1\}^k$ and for all $z \in \{0,1\}^n$, $z \neq \vec{0}$, there is an $x \in \{0,1\}^n$ so that $D \cdot x = c$ and $f(x) \neq f(x \oplus z)$, the LTFBDD size for f is bounded below by $2^{k-1} - 1$.*

Proof: Let an LTFBDD G with the transformation matrix A for the function $f(x_0, \ldots, x_{n-1})$ be given. Let $(l_0(x), \ldots, l_{n-1}(x)) = A \cdot (x_0, \ldots, x_{n-1})$ (i.e., we may interpret G as a BDD in which at the internal nodes the linear tests $l_0(x), \ldots, l_{n-1}(x)$ are performed). Then $l_i(x)$

is the result of the ith linear test on input x. We are going to show that the assumptions of the lemma imply that the top $k-1$ levels of G' are a complete binary tree, which implies the lower bound. Assume for contrary that there are partial computation paths p and p' that join after r tests, where $r \leq k-1$.

First, we assume that on p and p' the same linear tests, w.l.o.g. $l_0, \ldots, l_{r-1}$, are performed, possibly in different orderings. Hence, there are constants $c_0, \ldots, c_{r-1}$ and $d_0, \ldots, d_{r-1}$ so that the partial computation path p is chosen for exactly those inputs x for which $l_0(x) = c_0, \ldots, l_{r-1}(x) = c_{r-1}$, and p' is chosen for exactly those inputs y for which $l_0(y) = d_0, \ldots, l_{r-1}(y) = d_{r-1}$. Let $c_r, \ldots, c_{k-1}$ be defined as 0.

We are going to construct a partial computation path q that completes p and p' to a path to a sink. Then on q the linear tests $l_r, \ldots, l_{n-1}$ are performed. Let D denote the matrix consisting of the rows of A corresponding to $l_0, \ldots, l_{k-1}$. Let $z = A^{-1} \cdot (c_0 \oplus d_0, \ldots, c_{r-1} \oplus d_{r-1}, 0, \ldots, 0)$. By the assumptions of the lemma there is an x so that $D \cdot x = (c_0, \ldots, c_{k-1})$ and $f(x) \neq f(x \oplus z)$. This implies that the computation path for x starts with the partial computation path p. Let q be the partial computation path consisting of the computation path for x between the end of p and the sink.

Now consider the input $x \oplus z$. Note that $l_i(z)$ is equal to the inner product of the ith row of A and z. By the definition of z, for $0 \leq i \leq r-1$ we have $l_i(z) = c_i \oplus d_i$ and $l_i(x \oplus z) = l_i(x) \oplus l_i(z) = c_i \oplus (c_i \oplus d_i) = d_i$. Hence, for $x \oplus z$ the partial computation path p' is chosen. Again by the definition of z, for $r \leq i \leq n-1$ we have $l_i(z) = 0$ and $l_i(x \oplus z) = l_i(x) \oplus l_i(z) = l_i(x)$. Hence, for $r \leq i \leq n-1$ and the input $x \oplus z$ the linear tests $l_i(x \oplus z)$ yield the same results as $l_i(x)$, i.e., the path q is chosen. Hence, for x and $x \oplus z$ the same sink is reached, which is a contradiction to $f(x) \neq f(x \oplus z)$.

Finally, we discuss the situation that on p and p' not the same set of linear tests is performed. Then there is some linear test l that is performed w.l.o.g. on p but not on p'. Now we introduce a marking of the nodes v on p'. If there is a path from some node labeled by l to v, we mark v with 'A', and if there is a path from v to some node labeled by l, we mark v with 'B'. There is no node marked with A and B since otherwise there is a path on which the linear test l is performed twice. For the same reason from a node marked by A no node marked by B is reachable. The first node of p', i.e. the source, is marked by B since the test of l on p is reachable from this node, and for a similar reason the node where p and p' join is marked by A. Hence, p' starts with one or more nodes marked by B, possible followed by some unmarked nodes, and at the end there may be nodes marked by A. On the edge (v, w) leaving the last node v marked by B we insert a dummy node u performing the linear test l, where both outgoing edges lead to w. By the choice of (v, w) the resulting BDD is still an LTFBDD, which represents the same function. Let p_0 be the path consisting of p' up to the node u where at v the outgoing 0-edge is chosen, and let p_1 be the path consisting of p' up to u where at v the outgoing 1-edge is chosen. Then on p_0 and p_1 the same sets of linear tests are performed and they join after at most k tests. Now, we may apply the same arguments as above to obtain a contradiction, since the arguments also hold for $r \leq k$. □

It remains to apply this method to a particular function. We call the function defined in the following the matrix storage access function *MSA*. We remark that a similar function was considered by Jukna, Razborov, Savický and Wegener [15]. Let n be a power of 2 and let

$b = \lfloor \lfloor (n-1)/\log n \rfloor^{1/2} \rfloor$. Let the input $x_0, \ldots, x_{n-1}$ be partitioned into x_0, into $t = \log n$ matrices $C^0, \ldots, C^{t-1}$ of size $b \times b$ and possibly some remaining variables. Let $s_i(x) = 1$ if the matrix C^i contains a row consisting of ones only, and let $s_i(x) = 0$ otherwise. Let $s(x)$ be the value of $(s_{t-1}(x), \ldots, s_0(x))$ interpreted as a binary number. Then

$$\mathit{MSA}(x_0, \ldots, x_{n-1}) = \begin{cases} x_0 & \text{if } s(x) = 0, \\ x_0 \oplus x_{s(x)} & \text{if } s(x) > 0. \end{cases}$$

Theorem 2: *LTFBDDs for MSA have size* $2^{\Omega((n/\log n)^{1/2})}$.

Proof: It suffices to show that for $k = b - 2$ the assumptions of Lemma 1 are fulfilled. Let a $k \times n$-matrix D, a vector $c = (c_0, \ldots, c_{k-1})$ and a vector $z \in \{0,1\}^n$, $z \neq \vec{0}$ be given. We are going to construct an input x for which $D \cdot x = c$ and $\mathit{MSA}(x) \neq \mathit{MSA}(x \oplus z)$.

If $z_0 = 1$, we choose $s^* = 0$. Otherwise there is some s^* for which $z_{s^*} = 1$. We shall construct an input x so that $s(x) = s^*$ and $s(x \oplus z) = s^*$ as well. If $s^* = 0$, it holds that $\mathit{MSA}(x) = x_0 \neq x_0 \oplus z_0 = \mathit{MSA}(x \oplus z)$ and, if $s^* \neq 0$, we have $\mathit{MSA}(x) = x_0 \oplus x_{s(x)} \neq (x_0 \oplus z_0) \oplus (x_{s(x \oplus z)} \oplus z_{s(x \oplus z)}) = \mathit{MSA}(x \oplus z)$ as required.

Let $(s^*_{t-1}, \ldots, s^*_0)$ be the representation of the chosen value for s^* as a binary number. For $i = 0, \ldots, t-1$ we successively construct linear equations of the form $x_j = 0$ or $x_j = 1$, which make sure that for x and $x \oplus z$ the matrix C^i contains a row consisting of ones only, if $s^*_i = 1$, or that C^i contains a column consisting of zeros only, if $s^*_i = 0$. Hence, for a solution of the system of equations the number $s(x)$ takes the value s^* and $\mathit{MSA}(x) \neq \mathit{MSA}(x \oplus z)$. However, we also have to make sure that the equations $D \cdot x = c$ are fulfilled. Hence, we shall choose the equations $x_j = 0$ or $x_j = 1$ in such a way that the vectors of coefficients of all equations together with the rows of D are a linearly independent set. Then there is a solution x for the system of all considered linear equations so that $D \cdot x = c$ and $\mathit{MSA}(x) \neq \mathit{MSA}(x \oplus z)$.

For each $i = 0, \ldots, t-1$ we inductively construct $2b$ equations where the left-hand-sides are single variables from the matrix C^i. These equations make sure that $s_i(x)$ takes the value s^*_i. Let $i \geq 0$. Assume that for the matrices $C^0, \ldots, C^{i-1}$ the equations are already constructed. We show how to choose the equations for the matrix C^i. Up to now we have $2bi + k = 2bi + b - 2$ equations (for the matrices $C^0, \ldots, C^{i-1}$ and for the rows of D) with linearly independent vectors of coefficients. Let B be the set of vectors of coefficients of all these equations.

In the following we use the notation $x^i_{p,r}$ in order to refer to the input bit in the pth row and rth column of the matrix C^i, and the notation $z^i_{p,r}$ for the corresponding bit of z.

W.l.o.g. let $s^*_i = 1$. We shall choose two rows, the pth and qth row of C^i, which consist of the variables $x^i_{p,1}, \ldots, x^i_{p,b}$ and $x^i_{q,1}, \ldots, x^i_{q,b}$, respectively, and construct the linear equations $x^i_{p,1} = 1, \ldots, x^i_{p,b} = 1, x^i_{q,1} = z^i_{q,1} \oplus 1, \ldots, x^i_{q,b} = z^i_{q,b} \oplus 1$. The equations for the pth row make sure that in the input x this row only consists of ones and the equations for the qth row make sure that in the input $x \oplus z$ this row only consists of ones. It is easy to see that in a similar way we can enforce columns only consisting of zeros in the case that $s^*_i = 0$.

It remains to show how to choose p and q so that the set of vectors of coefficients of the resulting system of linear equations is linearly independent. For $p \in \{1, \ldots, b\}$ let V_p denote the set of vectors of coefficients of the equations $x^i_{p,1} = 1, \ldots, x^i_{p,b} = 1$.

Claim: There is some $p \in \{1, \ldots, b\}$ so that $B \cup V_p$ is linearly independent.

In order to prove the claim we assume for contrary that for all $j \in \{1, \ldots, b\}$ the set $B \cup V_j$ is not linearly independent. Then for each j, there is a linear combination $w_j \neq \vec{0}$ of vectors of V_j which simultaneously is a linear combination of vectors of B. Hence, the dimension of the vector space spanned by $B \cup \{w_1, \ldots, w_b\}$ is equal to the dimension of the vector space spanned by B, i.e., $2bi + b - 2$.

On the other hand, remember that B consists of the rows of D and of a set B' of $2bi$ vectors for equations with a single variable on the left-hand side. Obviously $B' \cup \{w_1, \ldots, w_b\}$ is a linearly independent set and the dimension of the vector space spanned by $B' \cup \{w_1, \ldots, w_b\}$ is $2bi + b$. Since this vector space is a subset of the vector space spanned by B, which has the dimension $2bi + b - 2$, we obtain a contradiction, which implies the claim.

Hence, there is some row p so that $B \cup V_p$ is linearly independent. We remove this row from C^i and obtain a matrix with $b - 1$ rows. Then we may apply the same arguments in order to obtain a second row q so that $B \cup V_p \cup V_q$ is a linearly independent set.

Altogether, we obtain a system of linear equations where the set of vectors of coefficients is linearly independent. Let x be a solution. Then the linear equations enforce that $D \cdot x = c$, that $s(x) = s(x \oplus z)$, and, by the case distinction above, that *MSA*$(x) \neq$ *MSA*$(x \oplus z)$. This completes the proof of Theorem 2. □

In the following theorem we state a polynomial upper bound on the size of strong LTFBDDs for *MSA*. Hence, polynomial size strong LTFBDDs can represent a larger class of functions than polynomial size LTFBDDs, and we get a justification to distinguish between these two restrictions of linear transformed BDDs.

Theorem 3: *There is a strong LTFBDD for MSA with $O(n^2/\log n)$ nodes.*

Proof: It is easy to construct an OBDD P_i for the computation of $s_i(x)$. The OBDD tests the variables of C^i in a rowwise variable ordering. If a row only consisting of ones is found, the 1-sink is reached. If in some row a 0-entry is found, the remaining variables of the row are skipped and for the next row it is tested whether it is a row consisting of ones only. The size of this OBDD is b^2.

In order to compute $s(x)$ we arrange the OBDDs P_i in a complete binary tree of depth t. At the root there is a copy of P_{t-1}. In the cases $s_{t-1}(x) = 0$ and $s_{t-1}(x) = 1$ different copies of P_{t-2} are reached, and so on. At the leaves of the tree we know the value of $s(x)$. If $s(x) = 0$, a test of x_0 suffices to compute the value of the function. At the leaf that is reached for some $s(x) > 0$ we perform the linear test $x_0 \oplus x_{s(x)}$ in order to compute the value of the function.

It is easy to see that this linear transformed BDD represents the function *MSA* and that its size is bounded by $O(n^2/\log n)$. Since in the binary tree only single variables are tested and x_0 is not tested there, the tests x_0 and $x_0 \oplus x_{s(x)}$ performed at the last level are not linear combinations of the tests of the previous levels. Hence, the constructed linear transformed BDD is a strong LTFBDD. □

4 Lower Bounds for LTOBDDs, $\oplus$LTOBDDs and Nondeterministic LTOBDDs

LTOBDDs, $\oplus$LTOBDDs and nondeterministic LTOBDDs have in common that they respect a generalized variable ordering. Hence, we shall apply communication complexity based arguments in order to prove lower bounds. For an introduction into communication complexity theory we refer to the monographs of Hromkovič [11] and Kushilevitz and Nisan [17]. We are going to prove lower bounds on the communication complexity by constructing large fooling sets. In order to introduce the notation we repeat the definition of fooling sets.

Definition 4: *Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function. Let (L, R) be a partition of the set of input variables. For an input x let $x^{(l)}$ denote the assignment to the variables in L according to x and let $x^{(r)}$ denote the assignment to the variables in R according to x. Let $(x^{(l)}, y^{(r)})$ be the input consisting of $x^{(l)}$ and $y^{(r)}$.*

A fooling set for f and the partition (L, R) of the input variables is a set $M \subseteq \{0,1\}^n$ of inputs which has for some $c \in \{0,1\}$ the following properties.

1. $\forall x \in M : f(x) = c$.
2. $\forall x, y \in M, x \neq y : [f(x^{(l)}, y^{(r)}) = \bar{c}] \vee [f(y^{(l)}, x^{(r)}) = \bar{c}]$.

We say that M is a strong fooling set if it has the following property 2′ instead of property 2 from above.

2′. $\forall x, y \in M, x \neq y : [f(x^{(l)}, y^{(r)}) = \bar{c}] \wedge [f(y^{(l)}, x^{(r)}) = \bar{c}]$.

We call M a 1-fooling set or strong 1-fooling set, respectively, if it has the above properties for $c = 1$.

It is well-known that the size of a fooling set for a function f and a partition (L, R) is a lower bound on the size of OBDDs for f and all variable orderings where the variables in L are arranged before the variables in R. However, in an LTOBDD for f and the transformation matrix A the function $g(y) = f(A^{-1} \cdot y)$ is represented. Hence, we have to construct large fooling sets for g in order to obtain lower bounds on the LTOBDD size for f.

In order to simplify the notation let $B = A^{-1}$ throughout this section. Furthermore, let the number n of variables be an even number. We always partition the set $\{y_0, \ldots, y_{n-1}\}$, which g depends on, into $L = \{y_0, \ldots, y_{n/2-1}\}$ and $R = \{y_{n/2}, \ldots, y_{n-1}\}$. Furthermore, we use the notation $y^{(l)}$ and $y^{(r)}$ to denote $(y_0, \ldots, y_{n/2-1})$ and $(y_{n/2}, \ldots, y_{n-1})$, respectively. We shall apply the following lemmas to prove the lower bounds. Lemma 6 is inspired by the presentation of Dietzfelbinger and Savický [6].

Lemma 5: *If for all regular matrices B there is a fooling set of size at least b for the function $g(y) = f(B \cdot y)$ and the partition (L, R), the LTOBDD size for f and all generalized variable orderings is at least b.*

Proof: Assume that there is an LTOBDD G of size less than b for the function f. Let B be the matrix for which $(x_0, \ldots, x_{n-1}) = B \cdot (y_0, \ldots, y_{n-1})$, where $y_0, \ldots, y_{n-1}$ is the generalized variable ordering of G. Then G is simultaneously an OBDD for the function $g(y) = f(B \cdot y)$ and the variable ordering $y_0, \ldots, y_{n-1}$ (where $y_0, \ldots, y_{n-1}$ are considered as variables). On the other hand, the existence of a fooling set of size b implies the lower bound b on the OBDD size for g (see e.g. Kushilevitz and Nisan [17]). This follows by a well-known cut-and-paste argument: If for different inputs y and z of the fooling set the partial computation paths for $y^{(l)}$ and $z^{(l)}$ lead to the same node, the computation paths for y, z, $(y^{(l)}, z^{(r)})$ and $(z^{(l)}, y^{(r)})$ lead to the same sink which is a contradiction to the fact that by the definition of the fooling sets the function does not take the same value for all these inputs. Hence, we get a contradiction to the assumption that the size of G is less than b. □

Lemma 6: *If for all regular matrices B there is a fooling set of size at least b for the function $g(y) = f(B \cdot y)$ and the partition (L, R), the $\oplus$LTOBDD size for f and all generalized variable orderings is at least $b^{1/2} - 1$. If for all regular matrices B there is even a strong fooling set of size at least b' for the function $g(y) = f(B \cdot y)$ and the partition (L, R), the $\oplus$LTOBDD size for f and all generalized variable orderings is at least b'.*

Proof: Again it suffices to prove a lower bound on the size of $\oplus$OBDDs for g. The results of Waack [24] imply that the rank of the communication matrix for g and the partition (L, R) is a lower bound on the size of $\oplus$OBDDs for g and the variable ordering $y_0, \ldots, y_{n-1}$. The lower bound b' on the size of a strong fooling set for g and the partition (L, R) implies that the $b' \times b'$ identity matrix is a submatrix of the communication matrix for g and the partition (L, R). This clearly implies the lower bound b' on the rank of the communication matrix. If b is merely a lower bound on the size of an ordinary fooling set, by the results of Dietzfelbinger, Hromkovič and Schnitger [5] the rank of the communication matrix is at least $b^{1/2} - 1$. □

Lemma 7: *If for all regular matrices B there is a 1-fooling set of size at least b for the function $g(y) = f(B \cdot y)$ and the partition (L, R), the nondeterministic LTOBDD size for f and all generalized variable orderings is at least b.*

Proof: The same cut-and-paste argument as described in the proof of Lemma 5 implies that the size of nondeterministic OBDDs for g and the variable ordering $y_0, \ldots, y_{n-1}$ is bounded below by the size of a 1-fooling set for g and the partition (L, R). Hence, we can apply the same proof as for Lemma 5. □

Although the lemmas imply that the well-known fooling-set method can be extended to prove lower bounds on the size of LTOBDDs and their generalizations, it remains the problem to apply this method to an explicitly defined function. In the following we define the function *INDEX-EQ*, a combination of the functions *INDEX* and *EQ*, which are both well-known functions in communication complexity theory. We get lower bounds on the size of LTOBDDs, $\oplus$LTOBDDs and nondeterministic LTOBDDs by constructing large fooling sets which are even simultaneously strong fooling sets and 1-fooling sets.

Definition 8: *Let k be a power of 2 and let $N = 2^k$. The function INDEX-EQ is defined on $n = 3N/2$ variables $x_0, \ldots, x_{n-1}$. The variables $x_0, \ldots, x_{N-1}$ are interpreted as a memory and the $N/2$ variables $x_N, \ldots, x_{n-1}$ are interpreted as $N/(2 \log N)$ pointers each consisting of $\log N$ bits. Let $m = N/(4 \log N)$. Let $a(1), \ldots, a(m), b(1), \ldots, b(m)$ denote the values of the pointers. Then INDEX-EQ$(x_0, \ldots, x_{n-1})$ takes the value 1 iff the following conditions hold.*

1. $\forall i \in \{1, \ldots, m\} : x_{a(i)} = x_{b(i)}$.
2. $a(1) < \cdots < a(m)$ *and* $b(1) < \cdots < b(m)$.
3. $a(m) < b(1)$ *or* $b(m) < a(1)$.

Because of the first condition the computation of the function includes the test whether the words whose bits are addressed by the pointers are equal. The second and the third condition ensure that the equality test has only to be performed if the pointers are ordered and if either all a-pointers are smaller than all b-pointers or vice versa. We remark that the last two conditions are not necessary for the proof of the lower bound. These conditions allow to prove a polynomial upper bound on the FBDD size of *INDEX-EQ*, which we shall state at the end of this section.

Theorem 9: *The size of LTOBDDs, $\oplus$LTOBDDs and nondeterministic LTOBDDs for the function INDEX-EQ is bounded below by $2^{\Omega(n/\log n)}$.*

Proof: By Lemmas 5–7 it suffices to show that for all regular $n \times n$ matrices B there is a strong 1-fooling set of size at least 2^m for the function $g(y) = \textit{INDEX-EQ}_n(B \cdot y)$ and the partition $(\{y_0, \ldots, y_{n/2-1}\}, \{y_{n/2}, \ldots, y_{n-1}\})$. The construction of the strong 1-fooling set essentially consists of the following steps. In the first one we construct sets I and J of indices of memory variables. In the second step we construct a set which will be the fooling set. The set I has the property that for all inputs of the fooling set the values of the variables with indices in I only depend on the results of the linear tests in $\{y_0, \ldots, y_{n/2-1}\}$. Similarly, for the inputs of the fooling set the values of the memory variables with indices in J only depend on the linear tests in $\{y_{n/2}, \ldots, y_{n-1}\}$. Since an equality test of those memory variables has to be performed, a large amount of information has to be exchanged between $\{y_0, \ldots, y_{n/2-1}\}$ and $\{y_{n/2}, \ldots, y_{n-1}\}$ in order to evaluate *INDEX-EQ*. Finally, we prove that the constructed set is really a fooling set.

Let B a regular $n \times n$ matrix. We always keep in mind that B is the matrix for which $(x_0, \ldots, x_{n-1}) = B \cdot (y_0, \ldots, y_{n-1})$. In particular, each row of B corresponds to one of the x-variables and each column of B to one of the y-variables.

Notation. We use the following notation. Let $B^{(l)}$ be the left half of B, i.e., the $n \times n/2$ matrix consisting of the first $n/2$ elements of each row of B. Similarly, let $B^{(r)}$ be the right half of B. Let $B[x_i]$ denote the ith row of B (the row corresponding to x_i), and let $B^{(l)}[x_i]$ and $B^{(r)}[x_i]$ be the left and right half of this row, respectively. Let each pointer $a(i)$ consist of the $k = \log N$ bits $a_{k-1}(i), \ldots, a_0(i)$ which are interpreted as a binary number. Similarly let each pointer $b(i)$ consist of the bits $b_{k-1}(i), \ldots, b_0(i)$. We shall use the notations x_j and $a_l(i)$ simultaneously even of both denote the same bit. Then $B[a_l(i)]$ denotes the row of B corresponding to the bit $a_l(i)$ of the input.

Construction of I and J. The choice of I and J can to be done in such a way that I and J have the following properties.

(P1) $|I| = N/16$, $|J| = N/16$ and $I, J \subseteq \{0, \ldots, N-1\}$.

(P2) The set $\{B^{(l)}[x_i] | i \in I\}$ is linearly independent and
$\mathrm{span}\{B^{(l)}[x_i]) | i \in I\} \cap \mathrm{span}\{B^{(l)}[x_j] | j \in J \vee N \leq j \leq n-1\} = \{\vec{0}\}$.

(P3) The set $\{B^{(r)}[x_j] | j \in J\}$ is linearly independent and
$\mathrm{span}\{B^{(r)}[x_j] | j \in J\} \cap \mathrm{span}\{B^{(r)}[x_i] | i \in I \vee N \leq i \leq n-1\} = \{\vec{0}\}$.

We shall apply property (P2) (and similarly (P3)) in order to prove that a system of linear equations whose vectors of coefficients are $B^{(l)}[x_i]$, where $i \in I \cup J \cup \{j | N \leq j \leq n-1\}$, has a solution. (P2) and (P3) also imply that $I \cap J = \emptyset$.

As an intermediate step for the construction of I and J we choose a set $I' \subseteq \{0, \ldots, N-1\}$, where $|I'| = N/8$. Let $\mathcal{B}$ be a basis of $\mathrm{span}\{B^{(l)}[x_\nu] | N \leq \nu \leq n-1\}$. Then $|\mathcal{B}| \leq N/2$. We extend this basis to a basis of the vector space generated by the rows of $B^{(l)}$. Since the rank of $B^{(l)}$ is equal to $n/2 = 3N/4$ and since $|\mathcal{B}| \leq N/2$, at least $N/4$ rows have to be chosen. Let I' be the set of indices of exactly $N/8$ of those rows.

Now we choose J. Let $\mathcal{C}$ be a basis of the vector space $\mathrm{span}\{B^{(r)}[x_i] | i \in I' \vee N \leq i \leq n-1\}$. Then $|\mathcal{C}| \leq N/8 + N/2$. We extend $\mathcal{C}$ to a basis of the vector space generated by $B^{(r)}$. Since the rank of $B^{(r)}$ is equal to $3N/4$, at least $N/8$ rows have to be chosen. Let J be the set of indices of exactly $N/16$ of those rows.

Finally, we choose $I \subseteq I'$. Let $\mathcal{D}$ be a basis of the vector space $\mathrm{span}\{B^{(l)}[x_j] | j \in J \vee N \leq j \leq n-1\}$. Then $|\mathcal{D}| \leq N/2 + N/16$. We extend $\mathcal{D}$ to a basis of the vector space $\mathrm{span}\{B^{(l)}[x_j] | j \in I' \vee j \in J \vee N \leq j \leq n-1\}$, where we only choose rows $B^{(l)}[x_i]$, where $i \in I'$. Since the dimension of the resulting vector space is at least $N/2 + N/8$, at least $N/16$ rows have to be chosen. Let I be the set of indices of exactly $N/16$ of those rows.

It is easy to see that I and J have the property (P1). Since I and J are chosen using the theorem that any linearly independent set can be extended to a basis, it follows that $\{B^{(l)}[x_i] | i \in I\}$ and $\{B^{(r)}[x_j] | j \in J\}$ are both linearly independent sets. This theorem has also been applied in order to choose I in such a way that $\{B^{(l)}[x_i] | i \in I\}$ and a basis of $\mathrm{span}\{B^{(l)}[x_j] | j \in J \vee N \leq j \leq n-1\}$ are linearly independent. Hence, the second condition of (P2) is fulfilled. Finally, J is chosen in such a way that (P3) even holds if I is replaced by its superset I'.

Construction of the fooling set. Let i^* be the $N/32$ smallest element of I and let j^* be the $N/32$ smallest element of J. If $i^* < j^*$, we choose for I^* the m smallest elements of I and for J^* the m largest elements of J. W.l.o.g. $k \geq 32$. Then $m < N/32$ and all elements of I are smaller than all elements of J. If $i^* > j^*$, we choose for I^* the m largest elements of I and for J^* the m smallest elements of J. Then all elements of I are larger than all elements of J. Let $I^* = \{i(1), \ldots, i(m)\}$ and $J^* = \{j(1), \ldots, j(m)\}$ such that $i(1) < \cdots < i(m)$ and $j(1) < \cdots < j(m)$. We shall only construct inputs where the chosen addresses $a(1), \ldots, a(m), b(1), \ldots, b(m)$ are equal to $i(1), \ldots, i(m), j(1), \ldots, j(m)$. Hence, for all considered inputs the second and the third condition of the definition of *INDEX-EQ* are fulfilled so that the value of *INDEX-EQ* on these inputs only depends on the first condition.

Let $i_\nu(\alpha)$ and $j_\nu(\alpha)$ denote the νth bit of $i(\alpha)$ and $j(\alpha)$, respectively. Let $s = (s_0, \ldots, s_{n-1})$ be an arbitrary solution of the system of linear equations that consists for all $\alpha \in \{1, \ldots, m\}$ and all $\nu \in \{0, \ldots, k-1\}$ of the following equations.

$$\begin{array}{rcl} B[a_\nu(\alpha)] \cdot s & = & i_\nu(\alpha) \\ B[b_\nu(\alpha)] \cdot s & = & j_\nu(\alpha) \end{array} \tag{1}$$

This system of linear equations has a solution since all rows of B are linearly independent.

Now we construct the fooling set. For all $(c(1), \ldots, c(m)) \in \{0,1\}^m$ we construct a system of linear equations. We shall prove that this system has at least one solution. We select an arbitrary solution and include it into the fooling set M. Since for different assignments to $c(1), \ldots, c(m)$ we get different systems of linear equations, which have disjoint sets of solutions, we obtain a set M of size 2^m.

In the following system of linear equations the only variables are denoted by y; all other identifiers denote constants. The linear equations are arranged as a table which shows the connections between the different equations. Note that the left column only contains variables in $y^{(l)}$ and the right column only variables in $y^{(r)}$. Hence, we may also consider the equations as two independent systems, one determining the values of $y^{(l)}$ and the other one the values of $y^{(r)}$.

1st block: For all $\alpha \in \{1, \ldots, m\}$:
$B^{(l)}[x_{i(\alpha)}] \cdot y^{(l)} = c(\alpha) \oplus B^{(r)}[x_{i(\alpha)}] \cdot s^{(r)}$ and $B^{(r)}[x_{i(\alpha)}] \cdot y^{(r)} = B^{(r)}[x_{i(\alpha)}] \cdot s^{(r)}$

2nd block: For all $\alpha \in \{1, \ldots, m\}$:
$B^{(l)}[x_{j(\alpha)}] \cdot y^{(l)} = B^{(l)}[x_{j(\alpha)}] \cdot s^{(l)}$ and $B^{(r)}[x_{j(\alpha)}] \cdot y^{(r)} = c(\alpha) \oplus B^{(l)}[x_{j(\alpha)}] \cdot s^{(l)}$

3rd block: For all $\alpha \in \{1, \ldots, m\}$ and for all $\nu \in \{0, \ldots, k-1\}$:
$B^{(l)}[a_\nu(\alpha)] \cdot y^{(l)} = B^{(l)}[a_\nu(\alpha)] \cdot s^{(l)}$ and $B^{(r)}[a_\nu(\alpha)] \cdot y^{(r)} = B^{(r)}[a_\nu(\alpha)] \cdot s^{(r)}$
$B^{(l)}[b_\nu(\alpha)] \cdot y^{(l)} = B^{(l)}[b_\nu(\alpha)] \cdot s^{(l)}$ and $B^{(r)}[b_\nu(\alpha)] \cdot y^{(r)} = B^{(r)}[b_\nu(\alpha)] \cdot s^{(r)}$

We have to prove that the system of linear equations has a solution. Since the left and the right column work on disjoint sets of variables, we can consider the solvability of both parts separately. Consider the left column. The system of linear equations consisting of the second and third block is solvable, since $s^{(l)}$ is a solution. Now we choose a basis from $\{B^{(l)}[x_{j(1)}], \ldots, B^{(l)}[x_{j(m)}]\} \cup \{B^{(l)}[a_\nu(\alpha)], B^{(l)}[b_\nu(\alpha)] | \nu \in \{0, \ldots, k-1\}, \alpha \in \{1, \ldots, m\}\}$ and remove from the second and third block all equations with vectors of coefficients not contained in the basis. The set of solutions does not change. To the resulting system of equations we add the equations of the first block. Remember that $a_\nu(\alpha)$ and $b_\nu(\alpha)$ belong to $\{x_i | N \leq i \leq n-1\}$. Hence, by property (P2) we obtain a system of linear equations where the vectors of coefficients are linearly independent. This implies that this system has a solution. The solvability of the system of equations in the right column follows with similar arguments and property (P3).

Proof of the fooling set properties. Let us start with the first property of fooling sets. Let $y \in M$, where y is a solution of the system of equations for $c(1), \ldots, c(m)$. Then the value

of $a_\nu(\alpha)$ in the input $B \cdot y$ is

$$\begin{aligned} a_\nu(\alpha) &= B[a_\nu(\alpha)] \cdot y \\ &= B^{(l)}[a_\nu(\alpha)] \cdot y^{(l)} \oplus B^{(r)}[a_\nu(\alpha)] \cdot y^{(r)} \\ &= B^{(l)}[a_\nu(\alpha)] \cdot s^{(l)} \oplus B^{(r)}[a_\nu(\alpha)] \cdot s^{(r)} \\ &= B[a_\nu(\alpha)] \cdot s \\ &= i_\nu(\alpha). \end{aligned}$$

The first equality follows from the definition of B. The third equality follows from the fact that y fulfills the equations of the third block. The last equality is true since s is a solution of (1). In the same way it follows $b_\nu(\alpha) = j_\nu(\alpha)$. In other words, for the input $B \cdot y$ the addresses $i(1), \ldots, i(m), j(1), \ldots, j(m)$ are selected.

Now we look for the value of $x_{i(\alpha)}$.

$$\begin{aligned} x_{i(\alpha)} &= B[x_{i(\alpha)}] \cdot y \\ &= B^{(l)}[x_{i(\alpha)}] \cdot y^{(l)} \oplus B^{(r)}[x_{i(\alpha)}] \cdot y^{(r)} \\ &= (c(\alpha) \oplus B^{(r)}[x_{i(\alpha)}] \cdot s^{(r)}) \oplus B^{(r)}[x_{i(\alpha)}] \cdot s^{(r)} \\ &= c(\alpha). \end{aligned}$$

The third equality follows from the fact that y fulfills the equations of the first block. In the same way, we obtain $x_{j(\alpha)} = c(\alpha)$. Hence, for all $\alpha \in \{1, \ldots, m\}$ it holds that $x_{a(\alpha)} = x_{b(\alpha)} (= c(\alpha))$ and *INDEX-EQ*$(B \cdot y) = 1$.

Now consider the condition $2'$ of the definition of strong fooling sets. Let y and z be elements of M, where $y \neq z$. Let y be a solution of the system of linear equations for $c(1), \ldots, c(m)$ and let z be a solution of the system of linear equations for $d(1), \ldots, d(m)$. Since $y \neq z$, there is some μ such that $c(\mu) \neq d(\mu)$.

The same computation as above shows that for the inputs $B \cdot y$, $B \cdot z$, $B \cdot (y^{(l)}, z^{(r)})$ and $B \cdot (z^{(l)}, y^{(r)})$ the addresses $i(1), \ldots, i(m), j(1), \ldots, j(m)$ are chosen. Furthermore, similar computations show that in the input $B \cdot (y^{(l)}, z^{(r)})$ the variable $x_{i(\alpha)}$ takes the value $c(\alpha)$ and $x_{j(\alpha)}$ takes the value $d(\alpha)$. Then $x_{i(\mu)} = c(\mu) \neq d(\mu) = x_{j(\mu)}$. Hence, *INDEX-EQ*$(B \cdot (y^{(l)}, z^{(r)})) = 0$. Similarly, it follows that *INDEX-EQ*$(B \cdot (z^{(l)}, y^{(r)})) = 0$. Hence, M is a strong 1-fooling set for $g(y) =$ *INDEX-EQ*$(B \cdot y)$.

The computations also show that from $y \in M$ the values $c(1), \ldots, c(m)$ for which y is included into M can be computed. It suffices to compute $x = B \cdot y$ and to look for $x_{i(1)}, \ldots, x_{i(m)}$. Hence, for different $c(1), \ldots, c(m)$ different inputs y are included into M as claimed above. This implies $|M| = 2^m$. □

We conclude this section with a polynomial upper bound on the FBDD size for *INDEX-EQ*.

Theorem 10: *There are FBDDs of size $O(n^6)$ for INDEX-EQ.*

Proof: Let k be a power of 2 and let n, m and N be defined as in Definition 8. The FBDD consists of m layers. We start with the description of the first one. In the top there is a complete binary tree of depth $2 \log N$ in which the variables of the pointers $a(1)$ and $b(1)$ are

tested. If both pointers are equal, the 0-sink is reached. In the following we only consider the case $a(1) < b(1)$. The other case can be handled similarly. Since $a(1) < b(1)$, the FBDD has also to test whether all $a(\cdot)$ are smaller than $b(1)$. Hence, the FBDDs stores $b(1)$, i.e., for inputs with different values $b(1)$ the computation paths do not join before a sink.

After testing the variables of the pointers $a(1)$ and $b(1)$ the variables $x_{a(1)}$ and $x_{b(1)}$ are tested and compared (i.e. if they are different, the 0-sink is reached). The pointers $a(1)$ and $b(1)$ remain stored.

Now we describe the νth layer, where $\nu \in \{2, \ldots, m\}$. We assume that in the previous layer the pointers $b(1)$, $a(\nu-1)$ and $b(\nu-1)$ are stored and passed to the νth layer. In the νth layer the variables of the pointers $a(\nu)$ and $b(\nu)$ are tested in complete binary trees of depth $2 \log N$. Since $b(1)$, $a(\nu-1)$ and $b(\nu-1)$ are stored, it is easy to test whether $a(\nu) < b(1)$, whether $a(\nu-1) < a(\nu)$ and $b(\nu-1) < b(\nu)$. In the negative case the 0-sink is reached. Otherwise, $x_{a(\nu)}$ and $x_{b(\nu)}$ are tested and compared. Finally, the values of $b(1)$, $a(\nu)$ and $b(\nu)$ are given to the next layer, or, after the last layer, the 1-sink is reached if all tests are passed.

In each layer at most 5 pointers and the fact $a(1) < b(1)$ is stored. Hence, width $2N^5$ is sufficient, and the total size is bounded by $O(n^6)$. It is obvious that in the constructed BDD on each computation path each pointer variable is tested at most once. It remains to show that this is also true for the memory variables. We observe that in the νth layer the bit $x_{a(\nu)}$ is tested only if $a(\nu) > a(\nu-1)$ and $a(\nu) < b(1)$. Similarly, $x_{b(\nu)}$ is tested only if $b(\nu) > b(\nu-1)$. If two a-pointers or two b-pointers address the same variable, the pointers are not ordered and this is recognized at the latest when the second pointer is read. Then the computation is aborted so that the addressed bit is read only once. If there is an a-pointer and a b-pointer addressing the same bit, the a-pointers or the b-pointers are not ordered or the a-pointer is not smaller than $b(1)$. Again the addressed bit is read only once. □

5 A Comparison of Complexity Classes of Linear Transformed BDDs

Let P-LTOBDD, P-LTFBDD, P-sLTFBDD, NP-LTOBDD and $\oplus$P-LTOBDD denote the sets of functions that have polynomial size LTOBDDs, polynomial size LTFBDDs, polynomial size strong LTFBDDs, polynomial size nondeterministic LTOBDDs and polynomial size $\oplus$LTOBDDs, respectively. Let P-OBDD, P-FBDD,... be defined similarly. In Figure 2 some inclusions between these classes are summarized. $A \to B$ means that $A \subsetneq B$, and a dotted lines between classes A and B means that these classes are not comparable, i.e., $A \not\subseteq B$ and $B \not\subseteq A$. The numbers in the figure refer to the following list of functions proving that the corresponding inclusion is proper or proving that the classes are not comparable. In order to make Figure 2 clearer, the relations between P-LTOBDD and NP-LTOBDD and some related classes are drawn separately.

First we remark that it is easy to see that all inclusions shown in Figure 2 hold. Besides the functions mentioned in the following, in the literature a lot of functions can be found that witness that the inclusions (1), (3) and (15) are proper. E.g., the so-called Hidden

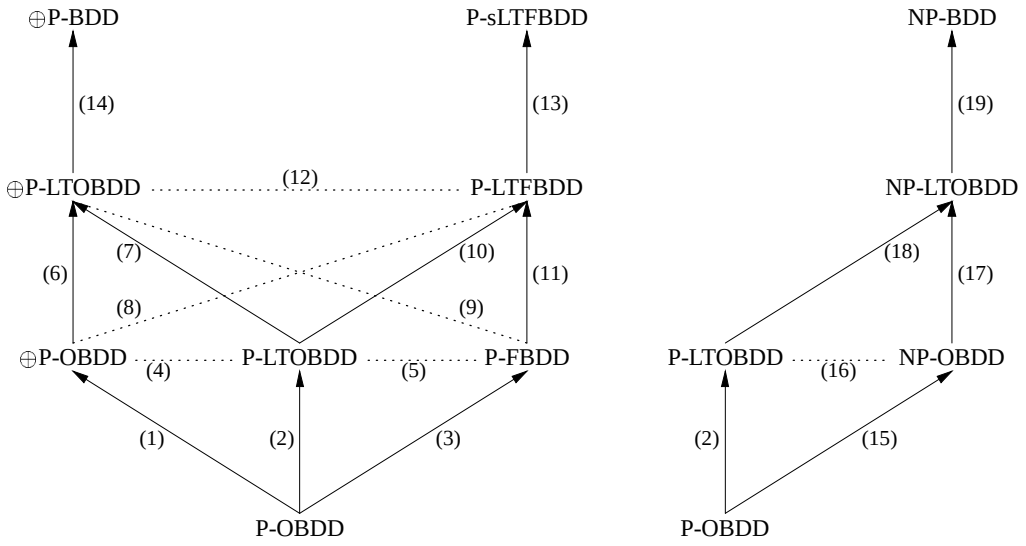

Figure 2: Comparison of the complexity classes of polynomial size LTOBDDs and related BDD variants.

Weighted Bit Function *HWB* only has exponential size OBDDs (Bryant [4]) but polynomial size FBDDs (Sieling and Wegener [23]) and polynomial size $\oplus$OBDDs (Gergov and Meinel [8]), which are simultaneously polynomial size nondeterministic OBDDs.

Our results on the function *MSA* prove that the inclusions (1), (7), (13), (15) and (18) are proper. The polynomial upper bounds for nondeterministic OBDDs and $\oplus$OBDDs are straightforward: The nondeterministic OBDD or $\oplus$OBDD "guesses" some number s and during the tests of the variables it computes x_0, if $s = 0$, or $x_0 \oplus x_s$, if $s > 0$, and simultaneously it evaluates $s(x)$ and compares s with $s(x)$. The upper bound for strong LTFBDDs is stated in Theorem 3. The lower bound for LTFBDDs is stated in Theorem 2. It clearly implies the same lower bound on the size of OBDDs, FBDDs and LTOBDDs for *MSA*. The characteristic functions of linear codes prove that the inclusions (2), (6), (11) and (17) are proper. The upper bound and references for the lower bounds are given in the Introduction. In particular, the lower bound for nondeterministic read-k-times branching programs of Jukna [12] implies lower bounds of the same size for OBDDs, FBDDs and nondeterministic OBDDs. We remark that Günther and Drechsler [9] presented a different function to prove that (2) is a proper inclusion. Their results implicitly imply that also (6) and (17) are proper inclusions. By our results for *INDEX-EQ* it follows that (3), (10), (14) and (19) are proper inclusions (Theorems 9 and 10).

It remains to discuss the incomparability results. (4) and (16) follow from the bounds on *MSA* and on the characteristic functions of linear codes, (5) follows from the bounds on *INDEX-EQ* and the characteristic functions of linear codes, and (8), (9) and (12) follow from our results on *INDEX-EQ* and *MSA*.

6 Conclusion

We conclude that the methods presented in this paper allow to prove exponential lower bounds for several variants of linear transformed BDDs. In particular, it is possible to separate the classes of functions with polynomial size representations for many variants of linear transformed BDDs. It remains an open problem to prove exponential lower bounds for strong LTFBDDs.

Acknowledgment

I thank Beate Bollig, Rolf Drechsler, Wolfgang Günther, Martin Sauerhoff, Stephan Waack and Ingo Wegener for fruitful discussions and helpful comments.

References

[1] S. Aborhey, Binary decision tree test functions, *IEEE Transactions on Computers* **37** (1988), 1461–1465.

[2] J. Bern, C. Meinel, and A. Slobodová, Efficient OBDD-based Boolean manipulation in CAD beyond current limits, *in* Proceedings of 32nd Design Automation Conference (1995), 408–413.

[3] R.E. Bryant, Graph-based algorithms for Boolean function manipulation, *IEEE Transactions on Computers* **35** (1986), 677–691.

[4] R.E. Bryant, On the complexity of VLSI implementations and graph representations of Boolean functions with application to integer multiplication, *IEEE Transactions on Computers* **40** (1991), 205–213.

[5] M. Dietzfelbinger, J. Hromkovič, and G. Schnitger, A comparison of two lower-bound methods for communication complexity, *Theoretical Computer Science* **168** (1996), 39–51.

[6] M. Dietzfelbinger, and P. Savický, Parity OBDDs cannot represent the multiplication succinctly, Technical Report, Universität Dortmund (1997).

[7] R. Drechsler, A. Sarabi, M. Theobald, B. Becker, and M.A. Perkowski, Efficient representation and manipulation of switching functions based on ordered Kronecker functional decision diagrams, *in* Proceedings of 31st Design Automation Conference (1994), 415–419.

[8] J. Gergov, and C. Meinel, Mod-2-OBDDs—a data structure that generalizes EXOR-sum-of-products and ordered binary decision diagrams, *Formal Methods in System Design* **8** (1996), 273–282.

[9] W. Günther, and R. Drechsler, BDD minimization by linear transformations, *in* Proceedings of Advanced Computer Systems, Szczecin, Poland (1998), 525–532.

[10] W. Günther, and R. Drechsler, Linear transformations and exact minimization of BDDs, *in* Proceedings of IEEE Great Lakes Symposium on VLSI (1998), 325–330.

[11] J. Hromkovič, "Communication Complexity and Parallel Computing," Springer, 1997.

[12] S. Jukna, A note on read-k times branching programs, *RAIRO Theoretical Informatics and Applications* **29** (1995), 75–83.

[13] S. Jukna, Linear codes are hard for oblivious read-once parity branching programs, *Information Processing Letters* **69** (1999), 267–270.

[14] S. Jukna and A. Razborov, Neither reading few bits twice nor reading illegally helps much, *Discrete Applied Mathematics* **85** (1998), 223–238.

[15] S. Jukna, A. Razborov, P. Savický and I. Wegener, On P versus $NP \cap co\text{-}NP$ for decision trees and read-once branching programs, *in* Proceedings of International Symposium on Mathematical Foundations of Computer Science, Lecture Notes in Computer Science 1295 (1997), 319–326.

[16] U. Kebschull, E. Schubert, and W. Rosenstiel, Multilevel logic synthesis based on functional decision diagrams, *in* Proceedings of European Design Automation Conference (1992), 43–47.

[17] E. Kushilevitz and N. Nisan, "Communication Complexity," Cambridge University Press, 1997.

[18] F.J. MacWilliams, and N.J.A. Sloane, "The Theory of Error-Correcting Codes," North-Holland, 1977.

[19] C. Meinel, F. Somenzi and T. Theobald, Linear sifting of decision diagrams, *in* Proceedings of 34th Design Automation Conference (1997), 202–207.

[20] E.A. Okol'nishnikova, On lower bounds for branching programs, *Metody Diskretnogo Analiza* **51** (1991), 61–83 (in Russian), English Translation in *Siberian Advances in Mathematics* **3** (1993), 152–166.

[21] A.A. Razborov, Lower bounds for deterministic and nondeterministic branching programs, *in* Proceedings of Fundamentals of Computing Theory, Lecture Notes in Computer Science 529 (1991), 47–60.

[22] R. Rudell, Dynamic variable ordering for ordered binary decision diagrams, *in* Proceedings of International Conference on Computer-Aided Design (1993), 42–47.

[23] D. Sieling and I. Wegener, Graph driven BDDs—a new data structure for Boolean functions, *Theoretical Computer Science* **141** (1995), 283–310.

[24] S. Waack, On the descriptive and algorithmic power of parity ordered binary decision diagrams, *in* Proceedings of Symposium on Theoretical Aspects of Computer Science, Lecture Notes in Computer Science 1200 (1997), 201–212.

[25] I. Wegener, On the complexity of branching programs and decision trees for clique functions, *Journal of the Association for Computing Machinery* **35** (1988), 461–471.

[26] S. Žák, An exponential lower bound for one-time-only branching programs, *in* Proceedings of International Symposium on Mathematical Foundations of Computer Science, Lecture Notes in Computer Science 176 (1984), 562–566.

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'