On the Limitations of Ordered Representations of Functions

Jayram S. Thathachar

Technical Report UW-CSE-96-09-03
September, 1996

Department of Computer Science and Engineering, FR-35
University of Washington
Seattle, WA 98195

On the Limitations of Ordered Representations of Functions

Jayram S. Thathachar[*]

jayram@cs.washington.edu

Department of Computer Science and Engineering

University of Washington

Box 352350

Seattle, Washington 98195–2350

Abstract

We demonstrate the limitations of the various ordered representations that have been considered in the literature for symbolic model checking including OBDDs [Bry86], MTBDDs [CMZ$^+$93], EVBDDs [LS92], *-BMDs [BC95] and HDDs [CFZ95] by giving a variety of simple and natural functions for which each representation requires exponential size to represent them. We also show that there is a simple regular language that requires exponential size to be represented by any *-BMD; note that OBDDs can represent any regular language in linear size.

1. Introduction

In recent years, symbolic model checking has become one of the most important techniques for formal verification of hardware systems. Bryant [Bry86] introduced the OBDD representation for functions and showed that OBDDs can represent a rich class of functions succinctly and allow various operations to be performed efficiently. Subsequently, it was shown that OBDDs can also be used to handle the state explosion problem in symbolic model checking (see, for example, [BCM$^+$90], [BCL$^+$94] and [McM93]).

Despite its success, OBDDs have proved to be unsatisfactory for representing some important functions. Thus ordered representations like MTBDDs due to Clarke *et al.* [CMZ$^+$93] and EVBDDs due to Lai and Sastry [LS92] were defined. These have been effective for some additional functions but still have exponential size complexity for other functions like

[*]This work was supported by the National Science Foundation under Grant CCR-9303017.

multiplication and exponentiation. Progress in the direction of concisely representing the multiplication function was made by Bryant and Chen [BC95] who proposed the BMD and the *-BMD representations and showed how multiplication and other arithmetic functions can be represented efficiently at the word level. An immediate and important question that arose was whether *-BMDs are more powerful than MTBDDs or EVBDDs or, at the least, OBDDs. This question was answered in the negative by Enders [End95], who exhibited functions with exponential complexity in the *-BMD representation but need only polynomial size OBDDs. Recently, Clarke *et al.*(see [CFZ95] and [CZ95]) defined generalizations of MTBDDs and BMDs, called hybrid decision diagrams (HDDs), that combine the advantages of both the representations. However, we are still far from understanding the true power and limitations of these representations and an important question is characterizing the complexity of various important functions in these representations.

In this report, we demonstrate the *weakness* of all of the representations above by showing that *none* of them can represent many important *boolean* functions concisely. We show that a variety of functions from arithmetic, sorting, string matching, formal languages, and graph theory have *exponential* complexity in every one of these representations. Our bounds also apply to other ordered representations like FDDs [KSR92] and *-BDDs [End95].

We derive these results by first defining a general abstract representation, called *Binary Linear Diagram*(BLD) that encompasses all of these representations and then show by a simple argument that the size of a BLD is bounded by the *rank* of a certain matrix associated with the function that it computes. This matrix has previously been looked at by researchers studying VLSI and computational complexity in order to prove area-time-squared (AT^2) bounds for boolean functions (See [Len90] for references and a survey of this area). One of the successful methods for proving AT^2 lower bounds is to construct large "fooling sets". Dietzfelbinger *et al.* [DHS94] showed that the rank is at least the square-root of the size of any fooling set; applying this, we can recast all the fooling set based AT^2 bounds for boolean functions as size bounds in the BLD representation. Examples of boolean functions that have exponentially large fooling sets can be found in Lipton and Sedgewick [LS81], Papadimitriou and Sipser [PS84] and Bryant [Bry91].

Directly computing the rank is a more difficult problem and hence has been used less often to prove AT^2 bounds. Hajnal *et al.* [HMT88] have shown that many graph-theoretic predicates have associated matrices with exponentially high rank.

The rank bound also holds for functions that are defined on a word level. This can be used to get some insight into the contrast in the complexity when functions are dealt with either in the bit level or in the word level. For example, for the bit-level representation of multiplication, the associated matrix has exponential rank but the word-level definition gives rise to a matrix of constant rank! This gives insight as to why multiplication at the word level has linear-sized *-BMDs but at the bit level requires exponential size in the all of the ordered representations.

For dealing with the complexity of functions in specific representations, Enders [End95], in order to derive the result referred to above, introduced a general technique that can be adapted to each individual representation. This technique does not apply to HDDs but can be extended to include it as well, although handling the resulting formalism is quite complicated. Applying this method, Enders showed that the graph-predicate that checks whether a graph is a triangle requires *-BMDs of exponential size. (This separates *-BMDs from OBDDs because the same function can be represented by an OBDD in polynomial size.) An interesting and natural question is whether *-BMDs can represent *regular languages* efficiently, for which OBDDs have linear size. We answer this in the *negative*, by exhibiting a simple regular language requiring exponential size in the *-BMD representation. This provides a partial justification for Clarke *et al.*'s (see [CFZ95] and [CZ95]) approach for HDDs where they use OBDDs to represent control-logic based functions and BMDs for arithmetic functions.

The report is organized as follows. In the next section, we define the BLD representation and illustrate how it generalizes all the ordered representations. Next, we describe formally our basic lower bound technique that applies to these representations which we then use to give lower bounds for many functions, either from the fooling set approach or the direct rank approach. Then, we demonstrate for a simple regular language that the *-BMD complexity is exponential and conclude with a summary and related important open questions.

2. Binary Linear Diagrams

A *(boolean) input* $\sigma : X \to \{0,1\}$ is an assignment of 0-1 values to a *variable* set X. For technical reasons, we will also allow X to be the empty set in which case σ is the (unique) empty input. We will use monomials to denote inputs; for example, $\overline{x}yz$ denotes an input $\sigma : \{x,y,z\} \to \{0,1\}$, where $\sigma(x) = 0$ and $\sigma(y) = \sigma(z) = 1$. Given two inputs $\sigma : Y \to \{0,1\}$ and $\pi : Z \to \{0,1\}$, where Y and Z are *disjoint*, their composition $\sigma \cdot \pi : Y \cup Z \to \{0,1\}$ is defined in the natural way. Define f to be a *pseudo-boolean function on* X if its domain is the set of inputs that assign 0-1 values to the variables of X and its range is some fixed ground field.[1] Given an input $\sigma : Y \to \{0,1\}$, the *subfunction* f_σ of f denotes a function that maps each input $\pi : X \backslash Y \to \{0,1\}$ to $f(\sigma \cdot \pi)$, that is, $f_\sigma(\pi) = f(\sigma \cdot \pi)$. In this report, the term function will always refer to a pseudo-boolean function and we will not mention the variable set on which a function is defined if it can be understood from the context.

We now define our abstraction of the ordered representations, the *(Ordered) Binary Linear Diagram(BLD)*. Let $X = \{x_1, x_2, \ldots, x_n\}$ be a set of variables and let $x_{p_1}, x_{p_2}, \ldots, x_{p_n}$ be an order imposed on the variables of X. The basic structure of a BLD is a labeled, directed acyclic graph. The nodes that have out-degree zero are called the sinks; each sink is labeled with an element of the ground field. Every other node v has out-degree two and the two edges that are directed from v that are distinguished as the 0-edge and 1-edge, respectively. The node that the 0-edge (respectively, 1-edge) directs to is called the 0-child (respectively,

[1]For boolean functions, this field is $GF[2]$.

1-child). The node v is labeled with a variable and a 2×2 matrix $\begin{bmatrix} v_{00} & v_{01} \\ v_{10} & v_{11} \end{bmatrix}$, with entries in the ground field. The BLD is required to satisfy the constraint that in every directed path, the sequence of variables appearing in order along that path must conform with the order $x_{p_1}, x_{p_2}, \ldots, x_{p_n}$.

The 2×2 matrix associated with a node describes the linear relationship between the function computed at the node and the two functions computed at its children. Formally, we define the semantics of computation in a BLD by associating a *node* function g_v with each node v. For a sink node v, g_v is a constant function (on the empty variable set) as given by its label. For a non-sink node v labeled with the variable x_{p_k} for some k, $1 \leq k \leq n$, g_v is defined on the variable set $\{x_{p_k}, x_{p_{k+1}}, \ldots, x_{p_n}\}$ in terms of its 0-child u and 1-child v as follows:

$$\begin{bmatrix} (g_v)_{\overline{x_{p_k}}} \\ (g_v)_{x_{p_k}} \end{bmatrix} = \begin{bmatrix} v_{00} & v_{01} \\ v_{10} & v_{11} \end{bmatrix} \cdot \begin{bmatrix} g_u \\ g_w \end{bmatrix}.$$

Note: We interpret the above and any other functional equations as follows: let Y be the union of the variable sets of all the various functions appearing in an equation. Then, for any input $\pi : Y \rightarrow \{0,1\}$, the equation should hold when each function is evaluated at π *restricted* to the variable set that is defined on. For example, the equation above implies that for any input $\pi : \{x_{p_{k+1}}, \ldots, x_{p_n}\} \rightarrow \{0,1\}$,

$$(g_v)_{\overline{x_{p_k}}}(\pi) = v_{00} \cdot g_u(\pi') + v_{01} \cdot g_w(\pi''),$$

where $\pi'(y) = \pi(y)$ for each variable in the variable set of g_u and π'' is similarly defined with respect to g_w.

Finally, there is a designated node in the first level called the *source* and we say that the BLD *computes* the node function associated with the source node. It is important to note that in contrast to many of the ordered representations that have canonical representations of functions, there can be possibly many BLDs computing the same function. However, this is not a drawback since our technique for proving lower bounds for functions does not require their BLD representations to be unique.

Given the description of a function in any of the ordered representations, we can show that there is an equivalent BLD of the same size. For example, given an OBDD or an MTBDD for a function f, the BLD for f has the same underlying acyclic graph with the same variable and sink labelings and with the associated matrix for each node being the 2×2 identity matrix. The transformation of BMDs to BLDs is similar except that the associated matrix for each node is $\begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$, as given by the Galois expansion. The case for *-BMDs needs only a little bit more work and is best illustrated by Example 1 that we give below. HDDs are *oblivious* forms of BLDs. Here the BLD is a leveled acyclic graph with all its edges going between adjacent levels. All the nodes in any level (except the level corresponding to the sinks) are labeled with the same variable and the same matrix that the HDD associates with that variable.

Example 1: To illustrate the transformation of *-BMDs to BLDs, consider the word-level multiplication function for a pair of two-bit numbers. Figure 1 shows both the *-BMD representation and the corresponding BLD representation of that function. Note that in the BLD, the matrix associated with the node labeled by the variable x_1 abstracts both the Galois expansion and the weight associated with the 1-edge of the corresponding node in the *-BMD.

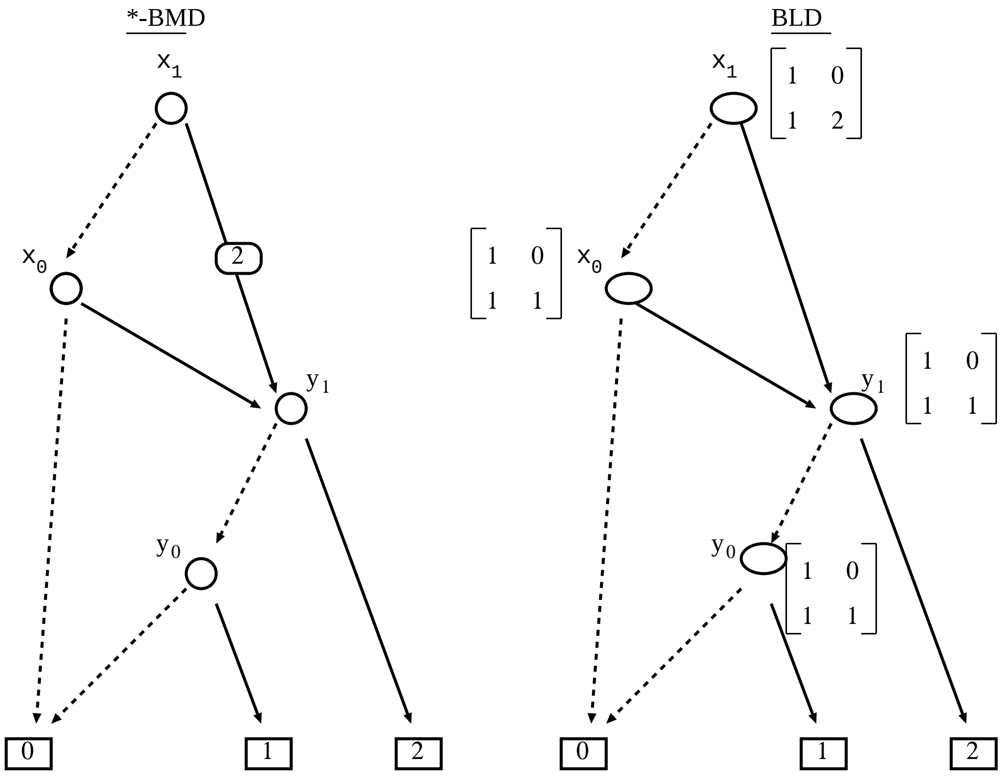

Figure 1: *The *-BMD (left) and BLD(right) for the multiplication function with the order of variables being x_1, x_0, y_1, y_0. The dashed lines denote the* 0*-edges and the solid lines denote the* 1*-edges.*

3. Lower Bounds for Functions under Ordered Representations

We now describe the rank method that we use to give complexity bounds for various functions that applies to all the ordered representations considered in this paper. First, we show how the rank of a certain matrix associated with a function bounds its BLD size. Then, we describe the various methods for getting bounds on the rank.

3.1. The Technique

For this section, let f be a function on $X = \{x_1, x_2, \ldots, x_n\}$ and let P be any BLD computing f with the order of variables being $x_{p_1}, x_{p_2}, \ldots, x_{p_n}$. Fix a k, $0 \leq k \leq n$, and let $L = \{x_{p_1}, x_{p_1}, \ldots, x_{p_k}\}$ be the first k variables in this order and R be the remaining variables. For each input $\sigma : L \to \{0,1\}$, we can associate a unique node in the BLD that can be reached from the source by tracing the path of 0-edges and 1-edges as defined by σ and stopping as soon as either a sink or a node labeled with a variable of R is reached. For example, in the BLD of Figure 1, the nodes corresponding to the inputs $\overline{x_1 x_0} y_1$ and $x_1 \overline{x_0}$ are the sink labeled with 0 and the node labeled with y_1 respectively. Let V_k denote the set of nodes associated, in the manner described above, with all the inputs that assign 0-1 values to the variables of L. The following lemma shows that the subfunction f_σ, for any input $\sigma : L \to \{0,1\}$, is linearly related to the node functions associated with the nodes in V_k.

Lemma 1: Let X, f, P, k, L and V_k be as defined above. Then, for any input $\sigma : L \to \{0,1\}$, there exist scalars $t_{\sigma,w}$, $w \in V_k$, in the ground field such that

$$f_\sigma = \sum_{w \in V_k} t_{\sigma,w} \cdot g_w$$

Proof: The proof is by induction on k.

Basis ($k = 0$:) Here $L = \Phi$ and σ is the empty input so $f = f_\sigma = g_s$, where $s \in V_0$ is the source node.

Induction Let $k > 0$ and suppose the statement is true for $k-1$. Let $L' = L \backslash \{x_k\}$ and $R' = X \backslash L'$. Fix any input $\sigma : L \to \{0,1\}$, and note that we can express it either as $\sigma' \cdot x_{p_k}$ or as $\sigma' \cdot \overline{x_{p_k}}$, for some input $\sigma' : L' \to \{0,1\}$. Assume that $\sigma = \sigma' \cdot x_{p_k}$; the proof for the other case is similar. By the induction hypothesis, there exist scalars $t_{\sigma',w'}$, $w' \in V_{k-1}$, such that $f_{\sigma'} = \sum_{w' \in V_{k-1}} t_{\sigma',w'} \cdot g_{w'}$.

Note that each w' in the sum above is either a sink or labeled with the variable x_{p_j}, for some $j \geq k$. Therefore, we have,

$$\begin{aligned} f_\sigma &= (f_{\sigma'})_{x_{p_k}} \\ &= \left(\sum_{\substack{w' \in V_{k-1} \\ x_{p_k} \text{ labels } w'}} t_{\sigma',w'} \cdot (g_{w'})_{x_{p_k}} \right) + \left(\sum_{\substack{w' \in V_{k-1} \\ x_{p_k} \text{ does not label } w'}} t_{\sigma',w'} \cdot g_{w'} \right) \qquad (1) \end{aligned}$$

Consider any $w' \in V_{k-1}$ in the first summand above. Since it is labeled by x_{p_k}, there exist the 0-child $u \in V_k$ and 1-child $v \in V_k$ in P such that $(g_{w'})_{x_{p_k}} = w'_{10} \cdot g_u + w'_{11} \cdot g_v$. Substitute this expression into Equation 1 for each such w'.

On the other hand, for any $w' \in V_{k-1}$ in the second summand above, w' is the (unique) node in P associated with the input σ so w' also belongs to V_k.

Combining the two observations above, we can see that f_σ is a linear combination of the node functions associated with the nodes in V_k, proving the lemma. $\square$

We will describe the linear relationship of Lemma 1 by a matrix equation. Again, fix a $0 \leq k \leq n$, and let L, R and V_k be as in the statement of Lemma 1 above. Define a matrix M_f associated with f of 2^k rows, one for each input $\sigma : L \rightarrow \{0,1\}$, and 2^{n-k} columns, one for each input $\pi : R \rightarrow \{0,1\}$. The (σ, π)-th entry of M_f is $f(\sigma \cdot \pi)$. Similarly, define a $|V_k| \times 2^{n-k}$ matrix M_g associated with the node functions in V_k. In this matrix, the (w, π)-th entry, for each $w \in V_k$ and each input $\pi : R \rightarrow \{0,1\}$, is $g_w(\pi')$, where π' is the input π restricted to the variable set of g_w. Finally, let T denote the $2^k \times |V_k|$ matrix that expresses the linear relationship between M_f and M_g. In other words, the (σ, w)-th entry of T, for each input $\sigma : L \rightarrow \{0,1\}$ and each node $w \in V_k$ is the $t_{\sigma,w}$ of Lemma 1.

The relationship between M_f and M_g, as given by Lemma 1 is $M_f = T \cdot M_g$. Therefore, we can infer from elementary linear algebra that

$$\text{rank}(M_f) \leq \text{rank}(M_g) \leq |V_k|.$$

Thus, the rank of M_f is a lower bound on the size of P.

Example 2: Let f be the multiplication function of Example 1. If we use the order as followed by the BLD in that example and set $k = 2$, the matrix M_f will have four rows corresponding to inputs that assign values to x_1 and x_0 and four columns corresponding to inputs that assign values to y_1 and y_0. The matrix is described below:

$$\begin{array}{c} \\ \overline{x_1 x_0} \\ \overline{x_1} x_0 \\ x_1 \overline{x_0} \\ x_1 x_0 \end{array} \begin{array}{c} \begin{array}{cccc} \overline{y_1 y_0} & \overline{y_1} y_0 & y_1 \overline{y_0} & y_1 y_0 \end{array} \\ \left[\begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 \\ 0 & 2 & 4 & 6 \\ 0 & 3 & 6 & 9 \end{array}\right] \end{array}$$

Notice that this matrix has rank one (over reals or rationals).

Referring to the BLD in Figure 1, we can see that V_k consists of two nodes, the sink labeled with zero and the node labeled with the variable y_1. Therefore, the matrix M_g has two rows corresponding to the node functions of these two nodes and four columns and has the following entries:

$$\begin{array}{r} \\ \text{sink labeled } 0 \\ \text{node labeled } y_1 \end{array} \begin{array}{c} \begin{array}{cccc} \overline{y_1 y_0} & \overline{y_1} y_0 & y_1 \overline{y_0} & y_1 y_0 \end{array} \\ \left[\begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3 \end{array}\right] \end{array}$$

In contrast, consider the bit-level multiplication function f' representing the second least significant bit of the product. In this case, the matrix $M_{f'}$ has rank two over $GF[2]$:

$$\begin{array}{c|cccc} & \overline{y_1 y_0} & \overline{y_1} y_0 & y_1 \overline{y_0} & y_1 y_0 \\ \hline \overline{x_1 x_0} & 0 & 0 & 0 & 0 \\ \overline{x_1} x_0 & 0 & 0 & 1 & 1 \\ x_1 \overline{x_0} & 0 & 1 & 0 & 1 \\ x_1 x_0 & 0 & 1 & 1 & 0 \end{array}$$

In determining the complexity of a function in the various representations, we must consider all possible orders of variables. Therefore, to prove lower bounds using the rank method above, we must show that for any order, there is always a partition of X into L and R such that the associated matrix M_f has a high rank. We state these observations as a theorem below:

Theorem 2: For any k, $0 \leq k \leq n$, and for any order of the variables $x_{p_1}, x_{p_2}, \ldots, x_{p_n}$, let $M_{f,k}^{p_1,p_2,\ldots,p_n}$ denote the matrix[2] where the (σ, π)-th entry is $f(\sigma \cdot \pi)$, for each σ and π that assign 0-1 values to $\{x_{p_1}, x_{p_2}, \ldots, x_{p_k}\}$ and $\{x_{p_{k+1}}, x_{p_{k+2}}, \ldots, x_{p_n}\}$, respectively. Then, any BLD that computes f must have size at least

$$\min_{p_1,p_2,\ldots,p_n} \max_k \operatorname{rank}(M_{f,k}^{p_1,p_2,\ldots,p_n}).$$

Corollary 3: The statement in Theorem 2 holds when we substitute any of the ordered representations like MTBDDs, EVBDDS, *-BMDs, HDDs in place of BLDs.

3.2. Rank versus Fooling Sets

We now consider the various methods that researchers have used to get lower bounds on the rank of many important *boolean* functions that holds independent of the order imposed on the variables. We will be primarily interested in those functions that have exponential rank.

For a boolean function f, and for a fixed partition of X into L and R, the matrix M_f is the matrix of the two-party *communication complexity* game, an area which has been extensively studied. In this game ([Yao79]), one player has the assignment of values to the variables of L and the other to the variables of R and their goal is to compute f by communicating the fewest number of bits. A related measure is the *best-partition* communication complexity ([PS84]) in which one computes the communication cost for the best choice of L and R, with the constraint that the sizes L and R be "large" enough. This measure has been heavily used to give AT^2 bounds in VLSI (see Lengauer [Len90] for references).

[2]We use this notation instead of the plain M_f to show the dependence explicitly.

One technique for getting lower bounds on the best-partition communication complexity is to construct large *boolean fooling sets*, one for each partition. If the minimum size of all of these sets is s, then $\log s$ is a lower bound on the best-partition communication complexity.

A fooling set $\mathcal{A}$ consists of pairs of inputs that assign values to the variables of L and R, respectively, such that

1. there exists a $\delta \in \{0,1\}$ such that for each pair (σ, π) in $\mathcal{A}$, $f(\sigma \cdot \pi) = \delta$, but
2. for any two distinct pairs (σ, π) and (σ', π') in $\mathcal{A}$, either $f(\sigma \cdot \pi') \neq \delta$ or $f(\sigma' \cdot \pi) \neq \delta$.

For our application, we are interested in knowing how the fooling set size relates to the rank. The following proposition due to Dietzfelbinger*et al.* [DHS94] shows that if the fooling set size is exponential, then so is the rank. Although they considered equipartitions, the same proof extends to unequal-sized partitions as well, which we give below for the sake of completeness.

Proposition 4 (Dietzfelbinger *et al.* **[DHS94]):** For a boolean function f, an order on its variable set $x_{p_1}, x_{p_2}, \ldots, x_{p_n}$ and an integer k, $0 \leq k \leq n$, let $M = M_{f,k}^{p_1,p_2,\ldots,p_n}$ be the matrix as defined in Theorem 2. Suppose s is the size of any fooling set and suppose r is the rank of M over any field.[3] Then, $r \geq \sqrt{s} - 1$.

Proof: Let $\mathcal{A}$ be a fooling set of size s. Without loss of generality assume that the δ in its definition is one and we will show below that the rank of M is at least $\sqrt{s}$. If the δ in its definition is zero, then the proof below will also apply to the matrix $J - M$, where J is a matrix whose every entry is one. Then, $\text{rank}(M) \geq \text{rank}(J - M) - 1 \geq \sqrt{s} - 1$.

As before, let $L = \{x_{p_1}, x_{p_1}, \ldots, x_{p_k}\}$ be the first k variables in the order and R be the remaining variables. The basic idea is to look at the *Kronecker* product $M \otimes M^T$. The rows of this matrix are indexed by the pairs (σ, π) and the columns by the pairs (π', σ'), for each $\sigma, \sigma' : L \rightarrow \{0,1\}$ and each $\pi, \pi' : R \rightarrow \{0,1\}$ and the entry corresponding to (σ, π)-th row and (π', σ')-th column is $f(\sigma \cdot \pi) f(\sigma' \cdot \pi')$. An important property that the Kronecker product satisfies is that $\text{rank}(M \otimes M^T) = \text{rank}(M)\text{rank}(M^T)$.

Let $\mathcal{A} = \{\sigma_i, \pi_i) : 1 \leq i \leq s\}$. Consider the $s \times s$ submatrix N of $M \otimes M^T$ induced by the rows corresponding to (σ_i, π_i) and the columns corresponding to (π_j, σ_j), for each $1 \leq i, j \leq s$. This matrix is an *identity* matrix because the (i,j)-th entry is $f(\sigma_i \cdot \pi_i) f(\sigma_j \cdot \pi_j)$ which equals one if and only if $i = j$. Thus, the rank of $M \otimes M^T$ is at least s from we can deduce that

$$s \leq \text{rank}(M \otimes M^T) = \text{rank}(M)\text{rank}(M^T) = \text{rank}(M)^2,$$

which proves the proposition. $\square$

Proposition 4 above, and Corollary 3 together imply that boolean functions that have fooling sets of exponential size, one for each order of the variables (with respect to some

[3]Note that M is a 0-1 matrix so this is well-defined.

partition), require exponential size in all the ordered representations. In general, constructing fooling sets is easier than computing the rank directly. However, there are functions for which the rank is exponentially larger than the size of any fooling set. In fact, Dietzfelbinger *et al.* [DHS94], in the same paper referred to above, showed that *almost all* boolean functions satisfy the property that the rank is exponential but no fooling set is larger than linear in size. Therefore, for some functions we have to resort to computing the rank directly. This again is a classic problem that has been extensively studied in communication complexity. As shown by Mehlhorn and Schmidt [MS82], the fixed-partition communication complexity of any function is bounded below by the logarithm of the rank of the associated matrix. Therefore, researchers have used this approach for proving bounds on the best-partition communication complexity by showing that for certain functions, all the matrices that arise by considering the various partitions have large rank. These results directly give bounds for the BLD representation as well.

3.3. Functions that have Exponential BLD Size

In this section, we list some important boolean functions which have exponential BLD-size for all orders of the variables; for each function, we will indicate the approach that was taken to show that the rank is exponentially large.

Selection/Equality Testing

Given $2n$ input bits, divide them into two halves, the *selector* and the *candidate*, of n bits each. Given that the selector has exactly $n/2$ bits set to one, function is defined to be one if and only if the $n/2$-bit number obtained by selecting those bits in the candidate at the positions corresponding to the zero bits in the selector equals the remaining $n/2$-bit number in the candidate (corresponding to the positions where the selector bits are one).

A Deterministic Context-Free Language

Here the input binary string is an encoding of a string in $\{0, 1, c, *\}^*$ and the function is defined to be one if and only if the input encodes a string that has the form wcw^R, for some $w \in \{0, 1\}^*$, when the $*$'s are removed from the string.

Pattern Matching

Here, the function is defined to be one if and only if the binary pattern string of αn bits occurs in the binary text string of $(1 - \alpha)n$ bits, where $0 < \alpha < 1$.

Comment: Lipton and Sedgewick [LS81] exhibit fooling sets of exponential size for each of the three functions above.

Shifted Equality

Given two n-bit numbers x and y and a $\log n$-bit number i, the function evaluates to one if and only if x equals the number y shifted circularly to the right by i bits.

Comment: The proof that this function has fooling sets of exponential size under all partitions illustrates a general technique by Lam and Ruzzo [LR92] who gave a transformation from the fixed-partition to the best-partition model. Using this technique, one can translate a result that shows that there is a large fooling set for a certain function under some *fixed* partition to another result that shows that there are large fooling sets for a *related* function under *all* partitions.

Multiplication

There are two versions that can be considered here. In the first version, the function verifies the product, that is, it takes two n-bit numbers x and y and a $2n$-bit number z as inputs and evaluates to one if and only if $x * y = z$. In the second version, the function computes the middle-bit of the product of two n-bit numbers.

Comment: For the verifier version, Lipton and Sedgewick [LS81] showed that this function has fooling sets of exponential size under all partitions of equal size. Bryant [Bry91] showed that even computing a single bit can be hard by exhibiting fooling sets of exponential size for the middle-bit version. In contrast, Bryant and Chen [BC95] showed that the word-level definition of multiplication has linear-sized *-BMDs.

For the following three problems, the input undirected graph is represented by $\binom{n}{2}$ bits.

Connectivity

The function is defined to be one if and only if the undirected graph is connected.

s-t-Connectivity

This function takes two additional vertices s and t as its input and evaluates to one if and only if there is a path between s and t in the graph.

Bipartiteness

Here, the function is defined to be one if and only if the undirected graph is bipartite, that is, the vertex set can be divided into two sets such that all the edges in the graph connect vertices of one set to vertices of the other.

Comment: For each of the three functions above, Hajnal *et al.* [HMT88] bounded the rank directly and showed that it is exponential.

4. *-BMDs and Regular Languages

In the earlier sections, we saw that the rank method is a useful tool for proving bounds that hold uniformly in all the ordered representations. A related and important problem is to contrast specific representations in order to understand what representations are best suited for a class of functions or languages. For example, MTBDDs and EVBDDs need exponential size to represent multiplication and exponentiation but *-BMDs can represent these and many other word-level arithmetic functions concisely. However, we already know by the result due to Enders [End95], referred to earlier, that *-BMDs are not more powerful than OBDDs. An interesting question is to contrast these representations for natural classes of languages. For *regular* languages, we know that OBDDs can represent any regular language in *linear* size by keeping track of the state in the automaton that represents it. In this section, we demonstrate that *-BMDs fail to represent all the languages in this class efficiently. We exhibit a simple regular language that any *-BMD representing it requires exponential size. In order to prove this, we use Enders' [End95] approach in bounding the number of distinct *path* functions.

Since all the ordered representations deal with functions and not languages, in order to define the complexity of a language in any of these representations, we formally define for each language S, the associated family of functions f_n^S, parameterized by the number of variables n, as follows: let $X = \{x_1, \ldots, x_n\}$ be the set of variables; then, for any input $\sigma : X \to \{0,1\}$, we have $f_n^S(\sigma) = 1$ if and only if $\sigma(1)\sigma(2)\ldots\sigma(n) \in S$. The complexity of S in a representation is defined to be the complexity of this family of functions in that representation.

We now state our main result of this section.

Theorem 5: Let the sets A_i, for $i = 0, 2, 3, 4$, be defined as follows:

$$A_i = \{w \in \{0,1\}^7 : w \text{ has } i \text{ ones }\}.$$

Then, any *-BMD representing the regular language

$$S = A_0^* A_3 (A_0 \cup A_2)^* \cup A_0^* A_4 A_0^*$$

requires size $2^{\Omega(n)}$.

We will first describe the technique that Enders introduced to derive bounds for the *-BMD representation and then apply it to prove our result.

Fix an order $x_{p_1}, x_{p_2}, \ldots, x_{p_n}$ on the variables and a *-BMD that computes $f = f_n^S$. For any input σ that assigns values to the first k variables in this order, for some $k \leq n$, let v_σ be the node reached in the *-BMD by taking the path corresponding to σ and let E_σ denote the product of the edge weights from the source to v_σ. As before, we will denote the node function corresponding to a node v by g_v.

Let the *path*[4] function $h_{(\sigma)}$, corresponding to σ, be defined as $h_{(\sigma)} = E_\sigma \cdot g_{v_\sigma}$. Enders showed that the path function can also be expressed in terms of the subfunctions using Mobius inversion. To describe this equation, we will need the following two notations: for any input σ, $|\sigma|$ denotes the number of variables set to one by σ and for any two inputs $\sigma, \tau : Y \rightarrow \{0,1\}$, we denote $\tau \leq \sigma$ to mean that $\tau(y) \leq \sigma(y)$ for each variable $y \in Y$. As before, partition X into L and R, where $L = \{x_{p_1}, x_{p_1}, \ldots, x_{p_k}\}$. For any input $\sigma : L \rightarrow \{0,1\}$, we have the following equations:

Proposition 6 (Enders [End95]):

$$\begin{aligned} f_\sigma &= \sum_{\tau \leq \sigma} h_{(\tau)} \\ h_{(\sigma)} &= \sum_{\tau \leq \sigma} (-1)^{|\sigma| - |\tau|} f_\tau \end{aligned} \tag{2}$$

To bound the number of nodes in any particular level, we define a variant of the fooling set that we used earlier. This variant is similar in spirit to the one used by Enders [End95] and Bryant [Bry91]. Here, the fooling set $\mathcal{A}$ consists of inputs that assign values to the variables of L only and satisfies the property that for any two distinct inputs $\sigma, \tau : L \rightarrow \{0,1\}$ in $\mathcal{A}$, there exist $\pi, \rho : R \rightarrow \{0,1\}$ such that

$$h_{(\sigma)}(\pi) h_{(\tau)}(\rho) \neq h_{(\sigma)}(\rho) h_{(\tau)}(\pi). \tag{3}$$

We claim that the *-BMD must have at least $|\mathcal{A}|$ nodes. For otherwise, by the pigeon-hole principle, there are two inputs σ and τ in $\mathcal{A}$ such that $v_\sigma = v_\tau$. But this implies that for all $\pi, \rho : R \rightarrow \{0,1\}$

$$h_{(\sigma)}(\pi) h_{(\tau)}(\rho) = E_\sigma E_\tau g_{v_\sigma}(\pi) g_{v_\tau}(\rho) = E_\sigma E_\tau g_{v_\tau}(\pi) g_{v_\sigma}(\rho) = h_{(\sigma)}(\rho) h_{(\tau)}(\pi),$$

which contradicts Equation 3.

With this machinery, we are now ready to prove the result.

Proof:[Of Theorem 5] Assume that $n = 14m$, for some large enough m. Divide the variables $X = \{x_1, x_2, \ldots, x_{14m}\}$ into $2m$ blocks of seven consecutive variables each. In other words, the block B_i, for $1 \leq i \leq 2m$, will contain the variables x_j, for $7(i-1) < j \leq 7i$. Any input σ that assigns values to the variables of a block B_i, for some i, induces a binary string $\sigma(7(i-1)+1)\sigma(7(i-1)+2)\ldots\sigma(7i)$. We will be interested in the various binary strings that are induced by the inputs in the appropriate blocks.

Fix an order $x_{p_1}, x_{p_2}, \ldots, x_{p_n}$ on X and also fix a *-BMD that uses this order to compute the characteristic function $f = f_n^S$ of S. Let $L = \{x_{p_1}, x_{p_1}, \ldots, x_{p_{n/2}}\}$ and $R = X \backslash L$ be an equipartition of X.

[4]We use $h_{(\sigma)}$ rather than h_σ to emphasize the fact that this is not a subfunction.

A simple counting argument shows that we can always find $2s$ blocks, for some *even* $s \geq m/8$, indexed by the set I ($|I| = 2s$), such that for $\ell \in I$, $|B_\ell \cap L| \geq 3$. For each $\ell \in I$, we will arbitrarily choose three variables in $B_\ell \cap L$ and call them *special.* Similarly, we can find a single block B_r, $r \notin I$, such that $|B_r \cap R| \geq 4$ and arbitrarily choose four special variables in $B_r \cap R$. We will call the blocks B_i, for $i \in I \cup \{r\}$, also as *special.*

The set of inputs that we will deal with will be obtained by carefully assigning values to the special variables. By default, the non-special variables in any input are always assigned to zero and we will not mention that henceforth. Note that this means that each non-special block induces the binary string 0000000, which is the unique element of A_0.

The fooling set $\mathcal{A}$ consists of inputs that correspond to the various ways of choosing a set $I' \subset I$ of cardinality s; for each such choice of an I', the corresponding input is defined as follows: For each $\ell \in I$,

(a) if $\ell \in I'$, then the three special variables in $B_\ell \cap L$ are each set to one.

(b) Otherwise, if $\ell \in I \backslash I'$, then set the special variable with the smallest index in $B_\ell \cap L$ to zero and the other two to one.

The fooling set has the right size of

$$\binom{2s}{s} = 2^{\Omega(n)}.$$

Note that if (a) (respectively, (b)) above was used to assign values to the special variables of a block, then assigning a zero to each of the other variables (which are non-special and hence get a zero value eventually anyway) induces a string in A_3 (respectively, A_2).

We will now show that the set we have constructed is indeed a fooling set by exhibiting two inputs $\pi, \rho : R \rightarrow \{0, 1\}$ such that

1. for all σ in $\mathcal{A}$, $h_{(\sigma)}(\pi) = 1$, and
2. $h_{(\sigma)}(\rho)$ is *distinct* for each $\sigma \in \mathcal{A}$.

Then, by Equation 3, we will have proved the theorem.

Define the input $\pi : R \rightarrow \{0, 1\}$ by setting all the four special variables in B_r to one. To see that $h_{(\sigma)}(\pi) = 1$, for any $\sigma \in \mathcal{A}$, via Equation 2, note that the induced string in block B_r is an element of A_4. The only way to ensure that $\tau \leq \sigma$ and $f_\tau(\pi) \neq 0$ is by having τ assign zeros to all the variables of L so that $\tau \cdot \pi$ induces an element of $A_0^* A_4 A_0^*$ in all the blocks put together. Then,

$$h_{(\sigma)}(\pi) = (-1)^{|\sigma| - |\tau|} f_\tau(\pi) = 1,$$

since s is even and σ assigns $5s$ variables to one whereas τ assigns no variable to one.

On the other hand, the input $\rho : R \rightarrow \{0,1\}$ is defined by assigning a zero to all the special variables in B_r (and hence to all variables in R). For each $\sigma \in \mathcal{A}$, we will obtain an exact expression for $h_{(\sigma)}(\rho)$ via Equation 2 and then show that it is distinct for each σ.

Notice that $\sigma \cdot \rho$ induces an element of A_3 in each of some s special blocks, an element of A_2 in each of some other s special blocks and the element of A_0 in each of the rest. To obtain a $\tau \leq \sigma$ such that $f_\tau(\pi) \neq 0$, the only choice is in assigning values to those blocks in which $\sigma \cdot \rho$ does *not* induce the element of A_0. Moreover, the string that $\tau.\rho$ induces contains no block that can be a member of A_4 therefore this string must be an member of $A_0^* A_3 (A_0 \cup A_2)^*$. This implies that $|\sigma| - |\tau|$ is odd so that $h_{(\sigma)}(\rho)$ is negative and its magnitude is the number of $\tau \leq \sigma$ such that $\tau \cdot \rho$ induces a string in $A_0^* A_3 (A_0 \cup A_2)^*$.

In the string induced by $\sigma \cdot \rho$, delete all the blocks corresponding to A_0 and call the resulting string of length $14s$ as w. Let $w = w_{2s} w_{2s-1} \ldots w_1$, where $w_i \in A_2 \cup A_3$, and let $n_s > n_{s-1} > \cdots > n_1$ be the positions such that for $1 \leq j \leq s$, $w_{n_j} \in A_3$. From the discussion above, it is clear that the magnitude of $h_{(\sigma)}(\rho)$ is exactly the number of strings u in $A_0^* A_3 (A_0 \cup A_2)^*$ such that u has the same length as w and the bit value for each position in u is no more than the bit value for the corresponding position in w. Let $u = u_{2s} u_{2s-1} \ldots u_1$, where $u_i \in A_0 \cup A_2 \cup A_3$, for $1 \leq i \leq 2s$. For each j, $1 \leq j \leq s$, observe that the number of u's in which $u_{n_j} \in A_3$ is exactly $2^{n_j - j} 4^{j-1} = 2^{n_j + j - 2}$. Summing up over all j, we have,

$$h_{(\sigma)}(\rho) = - \sum_{1 \leq j \leq s} 2^{n_j + j - 2}.$$

By a similar argument one can show that for an input $\sigma' \in \mathcal{A}$ different from σ, by deleting the blocks corresponding to A_0 in the string induced by $\sigma' \cdot \rho$, if $m_s > m_{s-1} > \cdots > m_1$ are the positions where the elements of A_3 appear, then,

$$h_{(\sigma')}(\rho) = - \sum_{1 \leq j \leq s} 2^{m_j + j - 2}.$$

By our construction of the fooling set, the vectors $(n_s, n_{s-1}, \ldots, n_1)$ and $(m_s, m_{s-1}, \ldots, m_1)$ are *different*. Let q be the largest integer such that $n_q \neq m_q$ and assume, without loss of generality, that $n_q > m_q$. Then,

$$h_{(\sigma')}(\rho) - h_{(\sigma)}(\rho) = \sum_{1 \leq j \leq q} 2^{n_j + j - 2} - \sum_{1 \leq j \leq q} 2^{m_j + j - 2} \geq 2^{n_q + q - 2} - (2^{m_q + q - 1} - 1) > 0.$$

This proves our claim that $h_{(\sigma)}(\rho)$ is distinct for each $\sigma \in \mathcal{A}$ and completes the proof of the theorem.

□

5. Conclusions

We have shown that a variety of boolean functions have exponential complexity in all the ordered representations by relating its complexity to two measures, the fooling set size and the rank. We also showed that there is a simple regular language that requires exponential size *-BMDs. We are currently investigating other boolean functions whose counter part multi-output functions are known to have high AT^2 bounds. There are two directions here– either look at some specific bit that is hard for that function or consider the function defined as a predicate by including the output of the function as part of the input of the predicate.

Another important issue is to study the limitations of the word-level representation of various functions. We are unaware of any results in this direction that show that certain word-level functions have high complexity in all the ordered representations. Since the bound in Theorem 2 applies for all functions, currently we are trying to show, via this technique, that even the word-level representations are inefficient for some important functions.

Acknowledgments

I am indebted to Paul Beame for his invaluable guidance and support during the course of this work and for his comments and clarifications in the report. I also thank Richard Anderson for directing me towards studying the *-BMD-complexity of regular languages.

References

[BC95] R.E. Bryant and Y.-A. Chen. Verification or Arithmetic Circuits with Binary Moment Diagrams. In *32nd ACM/IEEE Design Automation Conference*, Pittsburgh, June 1995. Carnegie Mellon University.

[BCL+94] J.R. Burch, E.M. Clarke, D.E. Long, K.L. MacMillan, and D.L. Dill. Symbolic model checking for sequential circuit verification. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13(4):401–424, April 1994.

[BCM+90] J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill, and L.J. Hwang. Symbolic Model Checking: 10^{20} States and Beyond. In *Proceedings of the Fifth Annual IEEE Symposium on Logic in Computer Science*, pages 1–33, Washington, D.C., June 1990. IEEE Computer Society Press.

[Bry86] R. E. Bryant. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, August 1986.

[Bry91] R. E. Bryant. On the Complexity of VLSI Implemetations and Graph Representations of Boolean Functions with Application to Integer Multiplication. *IEEE Transactions on Computers*, 40(2):205–213, February 1991.

[CFZ95] E. Clarke, M. Fujita, and X. Zhao. Overcoming the limitations of MTBDDs and BMDs. Technical Report CMU-CS-95-159, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, April 1995.

[CMZ$^+$93] E. Clarke, K.L. McMillian, X. Zhao, M. Fujita, and J.C.-Y. Yang. Spectral Transforms for large Boolean Functions with Application to Technologie Mapping. In *30th ACM/IEEE Design Automation Conference*, pages 54–60, Dallas, TX, June 1993.

[CZ95] E. Clarke and X. Zhao. A new approach for verifying arithmetic circuits. Technical Report CMU-CS-95-161, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, May 1995.

[DHS94] Dietzfelbinger, Hromkovic, and Schnitger. A Comparison of Two Lower Bound Methods for Communication Complexity. In *Symposium on Mathematical Foundations of Computer Science*, pages 326–335, 1994.

[End95] R. Enders. Note on the Complexity of Binary Moment Diagram Representations. Manuscript, 1995.

[HMT88] András Hajnal, Wolfgang Maass, and György Turán. On the communication complexity of graph properties. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 186–191, Chicago, Illinois, 2–4 May 1988.

[KSR92] U. Kebschull, E. Schubert, and W. Rosenstiel. Multilevel Logic Synthesis Based on Functional Decision Diagrams. In *29th ACM/IEEE Design Automation Conference*, pages 43–47, 1992.

[Len90] Lengauer. VLSI theory. In *Handbook of Theoretical Computer Science, Ed. Jan van Leeuwen, Elsevier and MIT Press (Volume A (= "1"): Algorithms and Complexity)*, volume 1. 1990.

[LR92] Lam and Ruzzo. Results on communication complexity classes. *Journal of Computer and System Sciences*, 44, 1992.

[LS81] Richard J. Lipton and Robert Sedgewick. Lower bounds for VLSI. In *Conference Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computation*, pages 300–307, Milwaukee, Wisconsin, 11–13 May 1981.

[LS92] Y.-T. Lai and S. Sastry. Edge-valued binary decision diagrams for multi-level hierarchical verification. In *29th ACM/IEEE Design Automation Conference*, pages 608–613, 1992.

[McM93] K.L. McMillan. *Symbolic Model Checking.* Kluwer Academic Publishers, Norwell Massachusetts, 1993.

[MS82] Kurt Mehlhorn and Erik M. Schmidt. Las Vegas is better than determinism in VLSI and distributed computing (extended abstract). In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*, pages 330–337, San Francisco, California, 5–7 May 1982.

[PS84] Papadimitriou and Sipser. Communication complexity. *Journal of Computer and System Sciences*, 28, 1984.

[Yao79] Andrew Chi-Chih Yao. Some complexity questions related to distributive computing (preliminary report). In *Conference Record of the Eleventh Annual ACM Symposium on Theory of Computing*, pages 209–213, Atlanta, Georgia, 30 April–2 May 1979.