Multilinearity Can Be Exponentially Restrictive*
(Preliminary version)

Rimli Sengupta and H. Venkateswaran
e-mail: {rimli, venkat}@cc.gatech.edu

GIT–CC–94–40

September, 1994

College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332–0280

Abstract

We define a Boolean circuit to be multilinear if the formal polynomial associated with it is multilinear as well. We consider the problem of computing the connectivity function using circuits that are monotone and multilinear. Our main result is that monotone multilinear circuits for this function require exponential size. Since connectivity can be computed by monotone Boolean circuits within size \( O(n^3) \), our lower bound establishes that at least in the context of monotone computation, multilinearity can be exponentially restrictive. As an application of our lower bound, we show that connectivity is not in \( m\mathcal{NC} \), thereby improving a recent result by Yao that connectivity is not in \( m\mathcal{NC}^1 \).

*This work was supported by NSF grant CCR-9200878.
1 Introduction

The study of restricted models of Boolean computation has been largely motivated by the difficulty in obtaining non-trivial size or depth lower bounds for computing functions using general Boolean circuits. Monotone Boolean circuits is an example of such a restricted model for which several interesting results have been obtained [5, 9, 8, 13]. In this work, we study the power of multiplicative idempotence for monotone Boolean circuits. We define a Boolean circuit to be multilinear if the formal polynomial associated with it is multilinear as well. Unlike monotone circuits, multilinear circuits can compute all Boolean functions because the circuit based on the representation of a Boolean function as a disjunction of its prime implicants is multilinear. However, since elements of a Boolean algebra satisfy the multiplicative idempotence property, not all formal polynomials representing a Boolean function need be multilinear. We consider the problem of computing the connectivity function using circuits that are monotone and multilinear. The connectivity function has been studied extensively in the past [17]. Recently, Yao has shown a $\Omega(\log^{1.5}(n))$ monotone depth bound for this function [18], thereby showing that it is not in $mNC^1$. Our main result is that monotone multilinear circuits for this function require exponential size. Since connectivity can be computed by monotone Boolean circuits within size $O(n^3)$, our lower bound establishes that at least in the context of monotone computation, multilinearity can be exponentially restrictive. As an application of our lower bound, we show that connectivity is not in $mNC$, thereby improving Yao's result.

The proof of our lower bound proceeds in three steps. We first show that the formal polynomial of a monotone Boolean circuit computing a function $f$ has a monomial whose variables are exactly those of the prime implicant, for each prime implicant of $f$. We then extend a framework developed by Jerrum and Snir [6] and use it in our lower bound argument. In [6], Jerrum and Snir developed a combinatorial framework to obtain size lower bounds for computing certain multilinear polynomials, including the spanning tree polynomial, using algebraic circuits over positive reals. We extend it in the context of the spanning tree polynomial to show that the bound holds for circuits whose formal polynomial corresponds to a linear combination of the terms in the multilinear polynomial being computed. Finally, we give a construction to convert a monotone multilinear circuit into one for which the above bound applies.

2 Preliminaries

Let $f : \{0, 1\}^n \rightarrow \{0, 1\}$ be a monotone Boolean function. We consider here Boolean circuits $B_n$ for $f$ over the monotone basis $\{\lor, \land\}$. Circuits have fanin 2. The size and depth of a circuit are defined as usual.

**Definition 2.1** A term is a conjunction of variables. Let $\text{var}(t)$ denote the set of variables in term $t$.

We shall use $\Pi(f)$ to denote the set of prime implicants of $f$ (see for instance [11] for a definition of prime implicants).

**Definition 2.2** A parse-graph $G$ in $B_n$ is defined inductively as follows: $G$ includes the root of $B_n$; for any $\lor$ gate $v$ included in $G$, exactly one immediate predecessor of $v$ in $B_n$ is included as its only predecessor in $G$; and for any $\land$ gate $v$ included in $G$, all the immediate predecessors of $v$ in $B_n$ are included as its predecessors in $G$. 

1
With every node of $B_n$, we can associate a formal multivariate polynomial defined inductively in a natural fashion. The formal polynomial associated with $B_n$ is the one formed at its root and it will be denoted as $P(B_n)$.

Every node of a parse-graph $G$ computes a monomial. The monomial computed at the root of $G$ is a monomial of $P(B_n)$. Thus, the number of monomials in $P(B_n)$ is the same as the number of parse-graphs in $B_n$.

**Definition 2.3** A Boolean circuit $B_n$ is said to be multilinear if $P(B_n)$ is multilinear.

It can be verified that $P(B_n)$ is multilinear if and only if all the parse-graphs of $B_n$ are trees. We shall use the term parse-trees instead of parse-graphs when considering multilinear circuits.

### 2.1 The Canonical Formal Polynomial

Given a monotone Boolean circuit $B_n$ computing a monotone function $f$, we establish some relationships between the monomials of $P(B_n)$ and the terms of $PI(f)$ leading to a canonical form for $P(B_n)$. The proofs of the following lemmas are based on the idea that $P(B_n)$ and $f$ must agree on every input assignment, since $B_n$ computes $f$.

**Lemma 2.1** For each monomial $\rho$ of $P(B_n)$, there exists a term $t \in PI(f)$ such that $var(t) \subseteq var(\rho)$.

**Proof:** Suppose there is a monomial $\rho$ for which this claim is not true. On the input assignment that sets the variables in $var(\rho)$ to 1 and all the rest to 0, $B_n$ evaluates to 1 but $f$ is 0, leading to a contradiction. □

In the other direction, we have the following lemma.

**Lemma 2.2** For all terms $t \in PI(f)$, there exists a monomial $\rho$ of $P(B_n)$, such that $var(t)$ = $var(\rho)$.

**Proof:** Let $t$ be any prime implicant of $f$. Consider the input that assigns 1 to the variables in $t$ and 0 to all the rest. On this input, $f$ evaluates to 1. Since $B_n$ computes $f$, $P(B_n)$ must have a monomial $\rho$ such that $var(\rho) \subseteq var(t)$, because otherwise $B_n$ would evaluate to 0 on this input. Now, there cannot be any other $t' \in PI(f)$ such that $var(t') \subseteq var(\rho)$, for otherwise $var(t') \subseteq var(t)$ which is impossible since $t, t' \in PI(f)$. It follows from lemma 2.1 that $var(\rho) = var(t)$. □

Since each parse-graph in $B_n$ computes a monomial in $P(B_n)$, by the above lemmas there is a term in $PI(f)$ associated with each parse-graph of $B_n$. By ordering the terms of $PI(f)$, we associate a unique prime implicant with each parse-graph of $B_n$. This allows us to partition the set of parse-graphs of $B_n$ into parse-classes, $PC_1, \ldots, PC_s$, where $s = |PI(f)|$. By lemma 2.2, each parse-class has at least one parse-graph whose positive variables correspond exactly with those of the prime implicant associated with the parse-class. We shall refer to one such parse-graph as a *representative* of the parse-class.

Thus, for any monotone $B_n$ computing a monotone $f$, we can put $P(B_n)$ in the following normal form: the monomials of $P(B_n)$ can be partitioned into $|PI(f)|$ parse-classes; in each parse-class, there are one or more monomials whose variable set coincides with that of the
prime-implicant corresponding to the class and each of the rest of the monomials in the parse-class contains
this set as a subset of its variable set. If $B_n$ is restricted further to be multilinear, then none of the monomials have repeated literals and hence the terms corresponding to the representatives of each parse-class look exactly like the prime implicant corresponding to the class.

2.2 The Connectivity Function

Consider the function $\text{UCONN}: \{0, 1\}^{n^2} \to \{0, 1\}$, which takes as input the adjacency matrix of a graph and outputs a 1 if and only if the graph is connected. Now a graph is connected if and only if it has a spanning tree. Thus, the prime implicants of $\text{UCONN}$ are the conjuncts corresponding to the spanning trees of a complete graph. Let $T_n = \{ t : \{2, 3, \ldots, n\} \to \{1, 2, \ldots, n\} \mid \forall i \exists k t^k(i) = 1 \}$ and for each $t \in T_n$, let $p_t = \wedge_{i=2}^{n} x_{t(i)}$. Then $\Pi(\text{UCONN})$ is exactly $\{p_t \mid t \in T_n\}$. Each $p_t$ is a directed tree rooted on the vertex labeled 1, spanning the complete graph on $n$ vertices. It is well known that $|T_n| = n^{n-2}$.

$\text{UCONN}$ can be computed by taking the transitive closure of the input adjacency matrix and performing $\wedge$ on the $n^2$ outputs. The circuit based on the well known Floyd-Warshall’s algorithm [1] for transitive closure is monotone and has size $O(n^3)$. Therefore, there is a $O(n^3)$ size monotone circuit for $\text{UCONN}$. In the next section we show that there are no polynomial size monotone multilinear circuits for this function.

3 A Lower Bound for Connectivity

In [6], Jerrum and Snir proved an exponential size lower bound on Boolean circuits computing $\text{UCONN}$ whose formal polynomial looks like: $P = \bigvee_{t \in T_n} p_t$. We begin by showing that this lower bound can be extended to hold for circuits whose formal polynomial is a linear combination of the terms in $P$. (In other words, some of the $p_t$'s could occur more than once in the formal polynomial.) Let us call such circuits normalized.

3.1 Adaptation of Jerrum and Snir’s Framework

Let $m = n^2$ and let $B_m$ be a normalized circuit for $\text{UCONN}$. As mentioned above, the main difference between $B_m$ and the circuits considered in [6] is that $B_m$ could have more than one parse-tree in $B_m$ for each prime implicant $p_t$. To extend the lower bound, we simply fix $n^{n-2}$ representative parse-trees $\{G_t \mid t \in T_n\}$ of $B_m$ such that $G_t$ computes $p_t$, for all $t$. The lower bound in [6] then applies to the sub-circuit of $B_m$ that computes the disjunction of the monomials corresponding to the representative parse-trees. We have simplified the presentation in [6] to adapt it to the model we are considering.

**Definition 3.1** For an $\wedge$-node $\alpha$, let $m(\alpha)$ be the number of representative parse-trees of $B_m$ in which $\alpha$ appears.

**Definition 3.2** An $\wedge$-node is said to be $(r,d)$-significant for $1 \leq r \leq n$ and $0 \leq d \leq \left\lfloor \frac{r}{2} \right\rfloor$, if it participates in a parse-tree with a term that has $r$ variables, $d$ of which are contributed by one of its immediate predecessors alone.

**Definition 3.3** Let $H$ be a sub-tree of a parse-tree $T$. Define weight of $H$ as follows: $W(H) = \sum_{\alpha \in \wedge \text{-nodes}(H)} \frac{1}{m(\alpha)}$, where $\wedge \text{-nodes}(H)$ denotes the set of $\wedge$-nodes in $H$. 

3
Lemma 3.1 below corresponds to theorem 3.3 in [6].

Lemma 3.1 \(\sum_{i=1}^{n-2} W(G_i) = |\{\land\text{-node } \alpha \mid m(\alpha) \geq 1\}|.\)

Proof: By definition,

\[\sum_{i=1}^{n-2} W(G_i) = \sum_{i=1}^{n-2} \sum_{\alpha \in \land\text{-nodes}(G_i)} \frac{1}{m(\alpha)}.\]

Fix an \(\land\text{-node } \alpha.\) For each representative parse-tree \(G_i,\) the contribution by \(\alpha\) to the sum on the right-hand side of the above equation is either 0 (if \(\alpha\) does not occur in it) or \(\frac{1}{m(\alpha)}.\) Thus, the total contribution by \(\alpha\) is \(m(\alpha) \frac{1}{m(\alpha)} = 1\) and therefore the right-hand side is the number of \(\land\text{-nodes } \alpha\) for which \(m(\alpha) \geq 1.\)

The following lemma summarizes the arguments in section 4.5 of [6]. These arguments are presented in the appendix for the sake of completeness.

Lemma 3.2 [6] Let \(n/2 < r \leq n - 1\) and \(1 \leq d < n/2.\) If \(\alpha\) is an \((r,d)\)-significant \(\land\text{-node of } B_m,\) then \(m(\alpha) \leq (\frac{3n}{4})^{n-1}.\)

For any subtree \(H\) of a representative parse-tree of \(B_m,\) let \(v(H)\) denote the number of variables in the term associated with \(H.\)

Lemma 3.3 Let \(H\) be a subgraph of any representative parse-tree \(T_r\) such that \(v(H) \geq n/2.\) Then, \(W(H) \geq (\frac{3r}{4})^{1-n}.\)

Proof: Since \(v(H) \geq n/2,\) there must be at least one \((r,d)\)-significant \(\land\text{-node } \alpha\) such that \(n/2 < r \leq (n - 1)\) and \(1 \leq d \leq (r - d) < n/2.\) The proof then follows using lemma 3.2 since \(W(H) \geq \frac{1}{m(\alpha)}.\)

From lemmas 3.1 and 3.3 it follows that the size of any normalized circuits for \(\text{UCONN}\) has size at least \(n^{n-2} (\frac{3n}{4})^{1-n}.\) Therefore we have,

Theorem 3.1 Any normalized Boolean circuit requires size \(\geq \frac{1}{n} \cdot (\frac{4}{3})^{n-1}\) to decide whether a graph is connected.

### 3.2 Monotone Multilinear Circuits

Let \(B_n\) be a monotone multilinear Boolean circuit for \(\text{UCONN}.\) We present the construction of an equivalent normalized circuit \(B'_n\) from \(B_n\) with size at most the square of the size of \(B_n.\) (Note that the main difference between \(B_n\) and \(B'_n\) is that there could be monomials in \(P(B_n)\) that have more than \(n - 1\) variables.) An exponential lower bound on the size of monotone multilinear circuits for \(\text{UCONN}\) then follows from theorem 3.1.

Recall the canonical formal polynomial of a monotone multilinear Boolean circuit for a monotone function (section 2.1). Each parse-class has one or more monomials that look exactly like the prime implicant corresponding to the class. The construction below produces a circuit in which only these monomials survive. Thus, the resulting circuit is normalized and computes the same function as the original circuit.

Define a function to be \(\text{homogeneous}\) with \(p\) variables if each of its prime implicants have the \(p\) variables. It is readily verified that \(\text{UCONN}\) is homogeneous with \(n - 1\) variables.
Lemma 3.4 Given a monotone multilinear circuit $B_n$ of size $s$ computing a homogeneous function $f$ with $p$ variables, there is a normalized circuit $B'_n$ that computes $f$ within size $O(s^2)$.

Proof: Given $B_n$, we first construct an equivalent circuit $C_n$ that has the following normal form: (i) $C_n$ has alternating $\lor$ and $\land$ layers; (ii) the output gate is an $\lor$ gate and all circuit inputs are inputs to $\lor$ gates; and (iii) each $\land$ gate has fan-in two. It is easily verified that the size of $C_n$ is at most twice that of $B_n$.

Given $C_n$, we construct an equivalent circuit $B'_n$ such that each monomial of $P(B'_n)$ has exactly $p$ variables. This is achieved by essentially keeping a count of the number of variables covered at a node.

- For every leaf node $A$ in $C_n$ create a leaf node $A$ in $B'_n$.
- For every $\lor$ gate $A$ in $C_n$ create the $\lor$ gates $[A, i, 0], 0 \leq i \leq p$, in $B'_n$.
- For every $\land$ gate $A$ in $C_n$ create the $\land$ gates $[A, i, 1], 0 \leq i \leq p$, in $B'_n$.
- For all $0 \leq i \leq p$, the inputs to an $\lor$ gate of the form $[A, i, 1]$ are $\land$ gates $[A, i, j, k]$, for all $j, k$ such that $0 \leq j, k \leq p$ and $j + k = i$.
- For all $i, j, k$, inputs to the $\land$ gate $[A, i, j, k]$ are the $\lor$ gates $[B, j, 0]$ and $[C, k, 0]$, where $B$ and $C$ are the inputs of the $\land$ gate $A$ in $C_n$.
- For all $0 \leq i \leq p$, the inputs to an $\lor$ gate of the form $[A, i, 0]$ are set as follows: for each input $B$ of the $\lor$ gate $A$ in $C_n$, (a) if $B$ is an $\land$ gate, make $[B, i, 1]$ an input of $[A, i, 0]$, for all $0 \leq i \leq p$; (b) if $B$ is a leaf node labeled with a variable, $[A, 1, 0]$ has $B$ as its input, and for all $i \neq 1$, $[A, i, 0]$ gets the constant 0 as an input; and (c) if $B$ is a leaf node labeled with a constant, $[A, 0, 0]$ has $B$ as its input, and for all $1 \leq i \leq p$, $[A, i, 0]$ gets the constant 0 as an input.

The size of $B'_n$ is at most a square of that of $C_n$ and therefore is $O(s^2)$. It is also easily verified that the formal monomials of $B'_n$ are those of $B_n$ that have exactly $p$ variables. Since the construction preserves monotonicity and multilinearity, $B'_n$ is a normalized circuit computing $f$. □

From lemma 3.4 and theorem 3.1 we obtain the following theorem:

Theorem 3.2 Monotone multilinear circuits require size $\geq \sqrt{\frac{1}{n} \cdot \left(\frac{4}{3}\right)^{n-1}}$ to compute UCONN.

Consider Boolean circuits in which not all formal monomials are multilinear but which have at least one representative in each parse-class which is multilinear. Let us call such circuits nearly multilinear. By the above construction, a monotone nearly multilinear circuit computing a homogeneous function can be efficiently made multilinear. Thus, the lower bound of theorem 3.2 above holds for monotone nearly multilinear circuits as well.

Theorem 3.3 Monotone nearly multilinear circuits require size $\geq \sqrt{\frac{1}{n} \cdot \left(\frac{4}{3}\right)^{n-1}}$ to compute UCONN.
4 Connectivity versus Reachability

Consider the reachability function \textsc{ustconn}: \(\{0,1\}^* \rightarrow \{0,1\}\), which takes as input the adjacency matrix of a graph \(G\) and two distinguished vertices \(s\) and \(t\) and outputs a 1 if and only if there is a path from \(s\) to \(t\) in \(G\). \textsc{ustconn} has been studied extensively in the past, both alone as well as in comparison with \textsc{uconn} [16]. These two functions are known to be projection equivalent to each other [2]. In his survey on connectivity [16], Wigderson mentions that under almost any choice of reducibility, \textsc{ustconn} is harder than \textsc{uconn}. For example, in the Boolean decision tree model and under monotone \(p\)-projections, \textsc{ustconn} is provably harder than \textsc{uconn} [3, 12]. The status of knowledge about the complexity of these two functions in the monotone Boolean circuit setting is consistent with this relative hardness. While a \(\Omega(\log^2(n))\) depth bound is known for \textsc{ustconn} [7], a similar bound is expected for \textsc{uconn} [16] but seems elusive. Recently, Yao has shown a \(\Omega(\log^{1.5}(n))\) monotone depth bound for \textsc{uconn} [18], thereby showing that \textsc{uconn} \(\notin mNL^c\).

However, the relative hardness of these two functions are reversed in the context of expressibility. \textsc{ustconn} is known to be a monadic \(\Sigma^1_1\) property\(^1\) but \textsc{uconn} is not [4]. Since \textsc{ustconn} is considered harder than \textsc{uconn}, this result is mentioned as being surprising and counter-intuitive in [16]. In this section, we exhibit two circuit settings in which \textsc{uconn} is provably harder than \textsc{ustconn}. As a consequence of the second, we show that \textsc{uconn} \(\notin mNL\).

4.1 Monotone Multilinear Circuits

The reachability function can be computed as the \((i, j)\)-th output of the transitive closure circuit based on the well known \(O(n^3)\) Floyd-Warshall’s algorithm [1]. This is a monotone circuit that can be verified to be nearly multilinear as well. Thus, by the remarks following theorem 3.2, there is a polynomial size monotone multilinear circuit for \textsc{ustconn}. Therefore, there is a monotone multilinear circuit for \textsc{uconn} of polynomial size but such circuits require exponential size to compute \textsc{uconn}, by theorem 3.2.

As was mentioned in section 2, there is an \(O(n^3)\) circuit for \textsc{uconn} obtained by taking the transitive closure of the input matrix and performing \(\wedge\) of the \(n^2\) outputs. This circuit is clearly monotone, but by theorem 3.2, it cannot be multilinear.

4.2 Monotone Skew Circuits

A Boolean circuit is said to be \textit{skew} if none of its \(\wedge\) nodes have more than one non-leaf input [15]. The class \(NL\) is characterized by uniform families of polynomial size skew circuits [15]. (The uniformity condition used is the notion of \(U_d\)-uniformity defined by Ruzzo [10].) The monotone analogue \(mNL\) of \(NL\) is then characterized by uniform families of monotone skew circuits of polynomial size (see also [5]).

In this section, we provide a construction by which a skew circuit can be made nearly multilinear in a uniform fashion. We will assume that the circuit is layered with alternating layers of \(\lor\) and \(\wedge\) gates, and all its \(\wedge\)-gates have fan-in 2.

Let \(G\) be a skew circuit computing a function \(f\).

**Phase 1:**

1. Construct the circuit \(G'\) which consists of \(n\) copies of \(G: G_1, G_2, \ldots, G_n\). In \(G'\), each \(\wedge\) gate is replaced with a \(\lor\) gate.

\(^1\)Expressible as \(\exists A_1 \exists A_2 \ldots \exists A_n \Psi\), where \(\Psi\) is a first order sentence and each \(A_i\) is a subset of the universe.
2. In $G_i$, the copy of an interior node $v$ of $G$ is labeled $[v, x_i]$, for all $1 \leq i \leq n$.

3. The leaves of each $G_i$ are labeled with constants such that the input to $G_i$ is the $i$th $n$-bit identity vector.

It is easily verified that,

**Fact:** In $G'$, the gate $[v, x_i]$ evaluates to 1 if and only if there is a path from the leaf $x_i$ to gate $v$ in $G$.

**Phase 2:** Construct the circuit $G''$ as follows:

1. For each $\lor$ gate $A$ in $G$, add the $\lor$ gate $[A, 0]$ to $G''$.

2. For each $\land$ node $A$ in $G$ with input $x$ (leaf) and $B$ ($\lor$ node), add the $\lor$ node $[A, 1]$ in $G''$ with input $\land$ nodes $A'$ and $A''$. The inputs to $A'$ are the leaf $x$ and the $\lor$ node $[B, 0]$ and the inputs to $A''$ are the $\lor$ node $[B, 0]$ and the node labeled $[B, x]$ in $G'$.

The polynomial computed at the node $[A, 1]$ is $x. P(B) + P(B). P([B, x])$, where $P(B)$ is the polynomial computed at $B$, and $P([B, x])$ is simply a disjunction of constants since the leaves of the circuit rooted at the node $[B, x]$ are all labeled with constants and each interior node is an $\lor$ gate.

The resulting circuit has size at most twice that of $G$. It can be made skew by observing that in step 2 of phase 2 in the above construction, the non-skew $\land$ gate $A''$ computes $P(B). \sum_{i=1}^{k} c_i$, where $c_i$'s are the labels of the leaves of the circuit rooted at the node $[B, x]$. $A''$ can thus be replaced by the skew circuit that computes $\sum_{i=1}^{k} c_i P(B)$.

The following fact can be verified using an inductive argument on the circuit depth.

**Fact:** $G''$ is nearly multilinear and computes $f$.

Now, from the lower bound in theorem 3.3 we get,

**Theorem 4.1** UCONN $\not\in mN\mathcal{L}$.

Since USTCONN is known to be in $mN\mathcal{L}$, connectivity is clearly harder than reachability in the monotone skew circuit setting. Moreover, since UCONN is known to be in $m\mathcal{SAC}^1$, we have the following separation.

**Corollary 4.1** $mN\mathcal{L} \subset m\mathcal{SAC}^1$.

## 5 Questions

- The monotone nearly multilinear circuit for USTCONN also has $O(n^3)$ depth. But monotone circuits for USTCONN are known to require $\Omega(log^2(n))$ depth [7]. So it is relevant to ask whether there are polynomial size monotone multilinear circuits that compute USTCONN within $O(log^2(n))$ depth.
• The construction in section 4.2 for converting a skew circuit into a nearly-multilinear circuit clearly holds for non-monotone circuits as well. Are nearly-multilinear circuits efficiently and uniformly multilinearizable in the non-monotone setting? This would place $\mathcal{NL}$ within multilinear-$\mathcal{P}$.

• Are there natural non-monotone functions in $\mathcal{P}$ for which multilinear circuits require exponential size?

• The problem of detecting whether a circuit is non-multilinear can be shown to be $\mathcal{NL}$-complete. What is the complexity of removing non-multilinearity in general Boolean circuits?

References


Appendix

We present the arguments in [6] that lead to lemma 3.2. Let $B_m$ be a monotone multilinear circuit for $\text{CONN}$.

Lemma 1 If $\alpha$ is an $(r,d)$-significant $\wedge$-node of $B_m$, then $m(\alpha) \leq \left[\frac{(n-1)^2-d(r-d)-(r(r-1))}{n-1}\right]^{n-1}$.

Proof: Let $\beta$ and $\gamma$ be the immediate predecessors of $\alpha$ in $B_m$. Let $G$ be a representative parse-tree in which $\alpha$ appears in an $r$-variable term. Let $a$ be the term formed at $\beta$ and $b$ be the term formed at $\gamma$ such that $a \cdot b$ has $r \geq 1$ variables and $a$ has $0 \leq d \leq \lfloor r/2 \rfloor$ variables that are not in $b$. Let $c$ be a term such that $a \cdot b \cdot c$ is the $n-1$-variable term formed at the root of $G$. Clearly, $c$ has $n-1-r$ variables that are not in $a$ or $b$. Moreover, for every representative parse-tree that $\alpha$ participates in, the term formed at $\beta$ ($\gamma$) must have exactly $d$ ($r-d$, resp.) variables with the same set of row indices.

For $1 \leq i \leq n$, let $X_i$ be the set of variables $\{x_{ij}\}$ such that $x_{ij}$ appears in the monomial computed by some representative parse-tree that $\alpha$ participates in. Since the monomial computed by each representative parse-tree is exactly a prime implicant, $m(\alpha)$ is bounded above by the number of functions $t \in T_n$ such that $x_{i,t(i)} \in X_i$ for all $2 \leq i \leq n$. This in turn is at most the number of functions $t : \{2, 3, \ldots, n\} \to \{1, 2, \ldots, n\}$ such that $x_{i,t(i)} \in X_i$ for all $2 \leq i \leq n$. This number is exactly $\prod_{i=2}^{n} |X_i|$, since each term $\prod_{i=2}^{n} x_{ij}$, $x_{ij} \in X_i$, corresponds to a different function $t$. This product is maximized when $|X_i|$ is independent of $i$, therefore,

$$\prod_{i=2}^{n} |X_i| \leq \left[\frac{\sum_{i=2}^{n} |X_i|}{n-1}\right]^{n-1}.$$

The expression $\sum_{i=2}^{n} |X_i|$ is easily seen to be bounded above by $(n-1)^2$. This can be further refined to $(n-1)^2 - d(r-d) - r(n-r-1)$ by observing that the representative parse-trees of $B_m$ cannot compute a monomial that contains both $x_{ij}$ and $x_{ji}$, for any $i, j$. □

Let $c(r,d) = \left[\frac{(n-1)^2-d(r-d)-(r(r-1))}{n-1}\right]^{n-1}$. In the range $n/2 < r \leq (n-1)$ and $1 \leq d \leq (r-d) < n/2$, holding $r$ constant, the expression is maximum at $d = r - \lfloor n/2 \rfloor + 1$ and allowing $r$ to vary, the overall maximum is at $r = (n-1)$.

Therefore,
\[c(r, d) \leq \left( \frac{(n-1)^2 - n/2(n/2-1)}{n-1} \right)^{n-1} \leq \left( \frac{3n^2/4 - 3n/2 + 1}{n-1} \right)^{n-1} \leq (\frac{2n}{4})^{n-1}\] 

This in conjunction with lemma 1 above leads to lemma 3.2.