Lower Bounds for Perfect Matching in Restricted Boolean Circuits *

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

GIT–CC–93/66

November, 1993

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

Abstract

We consider three restrictions on Boolean circuits: bijectivity, consistency and multilinearity. Our main result is that Boolean circuits require exponential size to compute the bipartite perfect matching function when restricted to be (i) bijective or (ii) consistent and multilinear. As a consequence of the lower bound on bijective circuits, we prove an exponential size lower bound for monotone arithmetic circuits that compute the 0-1 permanent function. We also define a notion of homogeneity for Boolean circuits and show that consistent homogeneous circuits require exponential size to compute the bipartite perfect matching function. Motivated by consistent multilinear circuits, we consider certain restricted $(\oplus, \wedge)$ circuits and obtain an exponential lower bound for computing bipartite perfect matching using such circuits. Finally, we show that the lower bound arguments for the bipartite perfect matching function on all these restricted models can be adapted to prove exponential lower bounds for the Hamiltonian circuit problem in all these models.

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

1 Introduction

The difficulty in obtaining non-trivial size or depth lower bounds for computing functions using general Boolean circuits has led researchers into studying restricted models of Boolean computation [1]. Monotone Boolean circuits is an example of such a restricted model for which several interesting results have been obtained [9, 3, 14, 7]. The problem of computing the bipartite perfect matching function has received considerable attention in the past. Deciding whether a bipartite graph has a perfect matching is equivalent to computing the Boolean permanent of the adjacency matrix of the graph. This problem can be solved by polynomial size Boolean circuits since it is known to be in $\mathcal{P}$ [4]. But no non-trivial lower bounds are known for the size or depth of general Boolean circuits that compute this function. For monotone Boolean circuits computing this function, a super-polynomial size lower bound [9] and a linear depth lower bound [7] are known. For constant depth unbounded fan-in circuits an exponential size lower bound for this function follows since it is constant depth reducible to PARITY [2].

In this paper we identify three new and non-trivial restrictions on Boolean circuits: bijectivity, consistency and multilinearity, obtained by defining restrictions on the formal polynomial associated with the circuit. Our main result is that Boolean circuits require exponential size to compute the bipartite perfect matching function when restricted to be: (i) bijective or (ii) consistent multilinear. We define a monomial to be consistent if it does not have both the positive and the negative literal of any variable appearing in it. A *consistent* circuit is one in which every monomial of the formal polynomial associated with the circuit is consistent. Note that monotone Boolean circuits are trivially consistent. We say that a Boolean circuit is *bijective* if the number of consistent monomials in the formal polynomial associated with the circuit is the same as the number of prime implicants of the function it computes. Although any Boolean function can be represented as a multilinear polynomial, formal polynomials associated with Boolean circuits for such functions need not be multilinear due to the idempotence property of the AND operation. A Boolean circuit is said to be *multilinear* if the formal polynomial associated with it is multilinear. Unlike monotone Boolean circuits, the restricted circuits we consider can compute all Boolean functions. This is because the circuit based on the representation of a Boolean function f as a disjunction of its prime implicants, is bijective, consistent and multilinear. Moreover, there are natural circuits for well known functions that are restricted. For instance, the $\mathcal{NC}^1$ circuit for PARITY is bijective, consistent and multilinear. Similarly, the polynomial size circuit implementing the Cocke-Kasami-Younger algorithm for context-free language recognition is consistent and multilinear.

To prove our lower bounds, we generalize a combinatorial framework developed by Jerrum and Snir [5] to obtain exponential lower bounds on the number of multiplications necessary to compute polynomials using circuits defined over semi-rings. The lower bound arguments in [5] are developed for the Boolean circuit model and then translated into lower bounds for circuits over other semi-rings. We observe that the Boolean circuits used in the lower bound arguments in [5] are bijective, monotone and multilinear. Since monotone circuits are necessarily consistent, our work can also be viewed as improvements to the results of [5] in several directions.

We also define a restriction called homogeneity on Boolean circuits. A *homogeneous* circuit is one in which every consistent formal monomial has the same number of positive literals. Unlike the three restrictions above, homogeneous circuits cannot compute all Boolean functions. However, they can compute all monotone homogeneous functions, such as bipartite perfect matching. We show that consistent homogeneous circuits require exponential size to compute this function.

One of the motivations for defining the bijectivity restriction on Boolean circuits is its

connection with counting classes. The circuit obtained by arithmetizing (that is, replacing the $\vee$ gates with $+$, $\wedge$ gates with $\times$ and each negated input $\bar{x}$ with $1-x$) a bijective Boolean circuit that decides whether a bipartite graph has a perfect matching, counts the number of perfect matchings in the graph. This is not true for general Boolean circuits due to the laws of additive idempotence and absorption of a Boolean algebra. Thus intuitively, the bijectivity restriction allows us to view arithmetic computation as restricted Boolean computation. This view has also been suggested by Valiant [17]. In fact, an interesting consequence of the lower bound result using bijective circuits is an exponential size lower bound for computing the 0-1 permanent using monotone arithmetic circuits over $(+, \times)$ with inputs $\{x_i | i \geq 1\}$, where each $x_i \in \{0, 1\}$. By defining monotone analogues of the counting classes $\sharp\mathcal{P}$ [18] and $\sharp\mathcal{LOGCFL}$ [21], this lower bound implies a separation between these two classes. Here $\sharp\mathcal{LOGCFL}$ is the counting version of the class $\mathcal{LOGCFL}$. Such a separation also follows from the result of Razborov [9] since it leads to a super-polynomial size lower bound for monotone arithmetic circuits for computing the permanent.

We also study the problem of computing the bipartite perfect matching function using $(\oplus, \wedge)$ circuits. To our knowledge, circuits over $(\oplus, \wedge)$ have not been studied in the past. The only related result is by Razborov [10] who studied constant depth unbounded fanin circuits over $\{\oplus, \wedge, 1\}$ and proved an exponential size lower bound for computing the MAJORITY function in this model. Without the constant 1, $\{\oplus, \wedge\}$ is an incomplete basis and circuits over this basis can only compute Boolean functions f such that $f(\mathbf{0}) = 0$. But interestingly enough, they can compute all such functions, bipartite perfect matching in particular, within the same size as that of general Boolean circuits [6]. As in the case of Boolean circuits, we identify a few interesting restrictions in this setting and prove an exponential size lower bound for computing the bipartite perfect matching function using $(\oplus, \wedge)$ circuits whose formal polynomials are multilinear and do not have monomials that occur an even number of times. This is a consequence of the lower bound for consistent multilinear Boolean circuits that compute this function.

Finally, we show that the lower bound arguments for the bipartite perfect matching function can be adapted to prove exponential size lower bounds for the Hamiltonian cycle problem on Boolean circuits that are: (i) bijective or (ii) consistent multilinear. This is because the prime implicants of this function are essentially the set of cyclic permutations, as opposed to being the set of all permutations in the case of bipartite perfect matching.

The following are some other noteworthy features of our work:

- By introducing the notion of consistency in Boolean circuits, we suggest a new approach to study the power of negations in Boolean circuits.
- Our lower bounds provide significant insight into the properties of polynomial size Boolean circuits that compute the bipartite perfect matching function. Namely, such circuits cannot be bijective or consistent multilinear.
- While bipartite perfect matching is known to be in $\mathcal{P}$ and Hamiltonian cycle is $\mathcal{NP}$-complete, the counting versions of both functions are complete for $\sharp\mathcal{P}$. This suggests that both these functions ought to be equally hard in some sense. By showing that both of them require exponential size on bijective Boolean circuits, we exhibit one such setting.

The rest of the paper is organized as follows. Section 2 introduces a few key concepts and contains some preliminary results. Section 3 defines the various restrictions on Boolean circuits. This lays the groundwork for the proofs of the lower bounds in section 4. Section 5 discusses the consequences in the arithmetic setting and presents the separation between

monotone $\sharp\mathcal{LOGCFL}$ and monotone $\sharp\mathcal{P}$. The lower bounds for $(\oplus, \wedge)$ circuits are contained in section 6. Section 7 discusses homogeneous circuits. Lower bounds for the Hamiltonian circuit problem are discussed in section 8. Section 9 concludes the paper.

2 Preliminaries

A Boolean circuit B_n is a rooted, directed, acyclic graph with interior nodes labeled from $\{\vee, \wedge\}$ and the leaf nodes labeled from $\{x_i, \bar{x}_i, 0, 1 | 1 \leq i \leq n\}$. B_n is *monotone* if the leaves are labeled only from $\{x_i, 0, 1 \mid 1 \leq i \leq n\}$. Without loss of generality, we will assume all $\wedge$-nodes have fanin 2. The *size* of a Boolean circuit is the number of non-leaf nodes in it, and its *depth* is the length of the longest path from any leaf to the root. Each B_n computes a unique Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$.

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 $P(B_n)$.

Definition 2.1 A *parse-graph* G in B_n is defined inductively as follows: G includes the root of B_n; for any $\vee$ 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 $\wedge$ gate v included in G, all the immediate predecessors of v in B_n are included as its predecessors in G.

Parse-graphs are a generalization of the notion of parse-trees in [5]. Every node of a parse-graph G computes a formal multivariate monomial, defined in the obvious way. 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 2.2 Each element in the set $\{x_i \mid 1 \leq i \leq n\}$ is a *variable*. A *literal* is a variable x in positive form x or negative form $\bar{x}$. A *term* is a conjunction of literals, both positive and negative. Each term t can be expressed as $t_+.t_-$, where each literal in t_+ is positive and each literal in t_- is negative. For any term t, t_+ is the *positive term* of t and t_- is its *negative term*. $var(t_+)$ denotes the set of variables in t_+ and $var(t_-)$ is the analogous set for t_-.

Definition 2.3 A term t is said to be *consistent* if $var(t_+) \cap var(t_-) = \emptyset$. A parse-graph is *consistent* if the term it computes is consistent.

Fact 2.1 [13] Each Boolean function f can be represented as a disjunction of its *prime implicants*, where each prime implicant is a term such that the set PI(f) of prime implicants satisfy the following properties: (i) for all $t \in$ PI(f), t is consistent and each literal appears at most once in t; (ii) PI(f) contains all terms $t = t_+.t_-$, such that setting the variables in $var(t_+)$ to 1 and those in $var(t_-)$ to 0 causes f to evaluate to 1, regardless of the values of the other variables; (iii) there do not exist terms $t, t' \in$ PI(f) such that both $var(t_+) \subseteq var(t'_+)$ and $var(t_-) \subseteq var(t'_-)$; (iv) when f is monotone, the terms in PI(f) do not contain negative literals.

2.1 Formal Monomials vs. Prime Implicants

Given a general Boolean circuit B_n computing a monotone function f, we can establish some relationships between the consistent monomials of $P(B_n)$ and the terms of PI(f). 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 consistent monomial ρ of $P(B_n)$, there exists a term $t \in \mathrm{PI}(f)$ such that $var(t) \subseteq var(\rho_+)$.

Proof: Suppose there is a consistent monomial ρ 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. □

Note that if f is non-monotone, then for each consistent monomial ρ, there is a term $t \in \mathrm{PI}(f)$ such that $var(t_+) \subseteq var(\rho_+)$.

In the other direction, we have the following lemma.

Lemma 2.2 For all terms $t \in \mathrm{PI}(f)$, there exists a consistent monomial ρ 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 ρ such that $var(\rho_+) \subseteq var(t)$, because otherwise B_n would evaluate to 0 on this input. Now, there cannot be any other $t' \in \mathrm{PI}(f)$ such that $var(t') \subseteq var(\rho_+)$, for otherwise $var(t') \subseteq var(t)$ which is impossible by fact 2.1 since $t, t' \in \mathrm{PI}(f)$. 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 $\mathrm{PI}(f)$ associated with each consistent parse-graph of B_n. By ordering the terms of $\mathrm{PI}(f)$, we associate a unique prime implicant with each consistent parse-graph of B_n. This allows us to partition the set of consistent parse-graphs of B_n into *parse-classes*, $PC_1, \ldots, PC_s$, where $s = |\mathrm{PI}(f)|$. 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 such a parse-graph as a *representative* of the parse-class.

3 Restricted Boolean Circuits

In this section, we define three restrictions on Boolean circuits: (a) bijectivity; (b) consistency and (c) multilinearity. As noted earlier, these restricted circuits can compute any Boolean function.

3.1 Bijective Circuits

By lemma 2.2 above, the formal polynomial of a general Boolean circuit computing a monotone function must contain *at least* $|\mathrm{PI}(f)|$ consistent monomials. This motivates the following restriction on Boolean circuits.

Definition 3.1 A Boolean circuit B_n computing the function f is said to be **bijective** if the number of consistent monomials in $P(B_n)$ equals the number of terms in $\mathrm{PI}(f)$.

If B_n is a bijective circuit that computes a monotone function f, then by lemmas 2.1 and 2.2, every consistent monomial ρ of $P(B_n)$ corresponds to a term $t \in \mathrm{PI}(f)$ such that $var(\rho_+) = var(t)$. It is then natural to ask whether negations are useful for bijective circuits computing a monotone function. We identify a property such that if a monotone Boolean function f satisfies this property then any bijective Boolean circuit computing f does not use negations to compute consistent monomials.

Definition 3.2 A Boolean function f is defined to be *suitable* if it is monotone and for any pair of terms $t, t' \in \mathrm{PI}(f)$, $\mid var(t') - var(t) \mid \geq 2$.

Lemma 3.1 If B_n is a bijective Boolean circuit computing a suitable function f, then for all consistent monomials ρ in $P(B_n)$, $var(\rho_-) = \emptyset$.

Proof: Let ρ be a consistent monomial in $P(B_n)$ with a negative literal $\bar{x}$. Consider the assignment $\mathcal{I}$ that sets all the variables in $var(\rho_+) \cup \{x\}$ to 1 and all the rest to 0. Since B_n is bijective and f is monotone, there exists a term t in $\mathrm{PI}(f)$, such that $var(t) = var(\rho_+)$. Therefore, f evaluates to 1 on input $\mathcal{I}$. But clearly, ρ evaluates to 0.

Since B_n computes f, there must be another consistent monomial ρ' in $P(B_n)$, such that ρ' evaluates to 1 on input $\mathcal{I}$. Since B_n is bijective, it follows from lemma 2.2 that there exists a term t' in $\mathrm{PI}(f)$ distinct from t, such that $var(t') = var(\rho'_+)$. In order for ρ' to evaluate to 1, the variables in $var(t')$ must be set to 1 on input $\mathcal{I}$. This implies that $var(t') \subseteq var(t) \cup \{x\}$, which is impossible since by definition of suitability, $\mid var(t') - var(t) \mid \geq 2$. This gives the desired contradiction. □

For a bijective circuit B_n computing a suitable function f, lemmas 2.1, 2.2 and 3.1 completely determine the variable sets of each consistent monomial in $P(B_n)$.

Theorem 3.1 If B_n is a bijective Boolean circuit that computes a suitable function f, then,

$$P(B_n) = \bigvee_{t \in PI(f)} \Big(\bigwedge_{x_i \in var(t)} x_i^{k_i} \Big) \vee \text{(inconsistent monomials)} ,$$

where each k_i is a natural number.

A natural question about bijective circuits is whether they can compute all Boolean functions within the same resources as general Boolean circuits. In section 4.1, we answer this in the negative by showing that bijective circuits require exponential size to compute bipartite perfect matching. This function is known to be in $\mathcal{P}$ and therefore can be computed within polynomial size using general Boolean circuits. Thus, there are functions for which the bijectivity restriction leads to an exponential blowup in size. However, there also are functions that bijective circuits can compute within the same resources as general Boolean circuits. For example, the $\mathcal{NC}^1$ circuit for PARITY is bijective (see section 3.4).

3.2 Consistent Circuits

Definition 3.3 An $\wedge$-node α in a Boolean circuit B_n is defined to be *consistent* if for any pair of leaves labeled with x and $\bar{x}$, that do not exist paths from both of these leaves to α.

Definition 3.4 A Boolean circuit B_n is said to be **consistent** if all its $\wedge$-nodes are consistent.

Fact 3.1 B_n is consistent if and only if $P(B_n)$ has no inconsistent monomials.

Consistent circuits can be viewed as Boolean circuits with restricted usage of negations, monotone Boolean circuits being trivially consistent. In the past, researchers have studied Boolean circuits with limited number of negations [12] to better understand the power of negations in Boolean computation. The study of consistent circuits is interesting for the same reason. However, in consistent circuits, the restriction is not on the number of negations but rather on the manner in which negations are used, namely, that they are not used for cancellation. This immediately leads to the question as to whether non-trivial usage of negation is at all possible if cancellation is not allowed. The answer is yes since there is a consistent $\mathcal{NC}^1$ circuit for PARITY (see section 3.4).

3.3 Multilinear Circuits

Multilinearity occurs naturally in the context of arithmetic computation over $(+, \times)$, where the formal polynomial of a circuit computing a multilinear function must be multilinear. However, this is not true if such circuits are restricted to have only 0-1 inputs (as we shall require in section 5), since 0 and 1 are idempotent with respect to the $\times$ operator. In such a case, multilinearity is a restriction on the model. This motivates the study of multilinear circuits in the Boolean setting.

Definition 3.5 A Boolean circuit B_n is said to be **multilinear** if $P(B_n)$ is multilinear.

Fact 3.2 $P(B_n)$ is multilinear if and only if all the parse-graphs of B_n are trees.

In section 3.4, we show that the circuit implementation of the Cocke-Kasami-Younger algorithm for context-free language recognition is multilinear. It is natural to ask whether multilinear circuits are as powerful as general Boolean circuits. Although we are unable to answer this question, we prove in section 4.2 that consistent multilinear circuits require exponential size to compute the bipartite perfect matching function. It follows that multilinearity conjunction with consistency can be exponentially restrictive over general Boolean circuits.

3.4 Examples of Natural Restricted Circuits

We first show that there is an $\mathcal{NC}^1$ circuit for PARITY that is bijective, multilinear and consistent.

Lemma 3.2 PARITY_n and $\overline{\text{PARITY}}_n$ can be computed with $\mathcal{NC}^1$ circuits that are bijective, multilinear and consistent.

Proof: The proof is by induction on n. Recall that both PARITY_n and $\overline{\text{PARITY}}_n$ have 2^{n-1} prime implicants.

For the base case, let $n = 2$. The circuits C_2 and $\bar{C}_2$ below compute $\text{PARITY}_2(x_1, x_2)$ and $\overline{\text{PARITY}}_2(x_1, x_2)$, respectively.

$$C_2 = (x_1 \wedge \bar{x_2}) \bigvee (\bar{x_1} \wedge x_2)$$

$$\bar{C}_2 = (x_1 \wedge x_2) \bigvee (\bar{x_1} \wedge \bar{x_2})$$

These circuits are clearly bijective, multilinear and consistent.

By inductive hypothesis, there exist circuits $C_{\frac{n}{2}}$, $\bar{C}_{\frac{n}{2}}$, $D_{\frac{n}{2}}$ and $\bar{D}_{\frac{n}{2}}$, such that $C_{\frac{n}{2}}$ ($\bar{C}_{\frac{n}{2}}$) computes $\text{PARITY}_{\frac{n}{2}}(x_1, \ldots, x_{\frac{n}{2}})$ ($\overline{\text{PARITY}}_{\frac{n}{2}}(x_1, \ldots, x_{\frac{n}{2}})$, resp.) and $D_{\frac{n}{2}}$ ($\bar{D}_{\frac{n}{2}}$) computes the same functions for the variable set $\{x_{\frac{n}{2}+1}, \ldots, x_n\}$. Also by hypothesis, each of these circuits are consistent, multilinear and bijective. That is, they must have exactly $2^{\frac{n}{2}-1}$ consistent parse-trees. The circuits C_n ($\bar{C}_n$) below compute PARITY_n and $\overline{\text{PARITY}}_n$, respectively.

$$C_n = (C_{\frac{n}{2}} \wedge \bar{D}_{\frac{n}{2}}) \bigvee (\bar{C}_{\frac{n}{2}} \wedge D_{\frac{n}{2}})$$

$$\bar{C}_n = (C_{\frac{n}{2}} \wedge D_{\frac{n}{2}}) \bigvee (\bar{C}_{\frac{n}{2}} \wedge \bar{D}_{\frac{n}{2}})$$

The number of consistent parse-graphs in C_n is $2 \cdot (2^{\frac{n}{2}-1})^2$ which clearly equals 2^{n-1}. Thus C_n is bijective. $\bar{C}_n$ is bijective for the same reason. To show that C_n is multilinear, we observe that although the inputs sets of $C_{\frac{n}{2}}$ and $\bar{C}_{\frac{n}{2}}$ are the same, subgraphs of $C_{\frac{n}{2}}$ and $\bar{C}_{\frac{n}{2}}$ do not appear in the *same* parse-graph in C_n. Moreover, subgraphs of $C_{\frac{n}{2}}$ and $\bar{D}_{\frac{n}{2}}$ do appear in the same parse-graph in C_n but their input sets are disjoint. Multilinearity of C_n then follows by inductive hypothesis. C_n is also consistent by a similar argument. The analysis for $\bar{C}_n$ is similar. Clearly, C_n and $\bar{C}_n$ are both $\mathcal{NC}^1$ circuits.□

We now show that the circuit implementation in [20] of the standard Cocke-Kasami-Younger algorithm for context-free language recognition is multilinear.

For a fixed context-free grammar in Chomsky normal form, a Boolean circuit family $\{G_n\}$ that accepts the language generated by the grammar can be derived from the Cocke-Kasami-Younger algorithm. A gate in the circuit G_n has one of the following forms [20]:

- $[A, i, j]$, for some nonterminal A and integers i and j such that $1 \leq i \leq j \leq n$. This is an $\vee$ gate that evaluates to 1 on input x if and only if $A \stackrel{*}{\Rightarrow} x_{ij}$.

- $[B, C, i, j, k]$, for some integers i and j such that $1 \leq i \leq j \leq n$, for all pairs of nonterminals B and C for which $A \rightarrow BC$ is a production for some nonterminal A, and for some integer k such that $i \leq k < j$. This is an $\wedge$ gate that has two inputs $[B, i, k]$ and $[C, k+1, j]$ and it evaluates to 1 on input x if and only if $B \stackrel{*}{\Rightarrow} x_{ik}$ and $C \stackrel{*}{\Rightarrow} x_{k+1,j}$.

The output gate is $[S, 1, n]$ where S is the start symbol in the grammar G. The input to a gate of the form $[A, i, j]$ are for $i < j$ all gates of the form $[B, C, i, j, k]$ where $i \leq k < j$ and $A \rightarrow BC$ is a production in the grammar. The gate $[A, i, i]$ has a single input which is one of the following: (a) the constant 1 if both the productions $A \rightarrow 0$ and $A \rightarrow 1$ are in the grammar, and the constant 0 otherwise; (b) X_i, the ith input if $A \rightarrow 1$ is a production in the grammar; (c) $\bar{X}_i$, if $A \rightarrow 0$ is a production in the grammar.

Let α be any $\wedge$ gate in G_n labeled $[A, B, i, j, k]$. By the labeling method described above, α has two children β and γ labeled $[A, i, k]$ and $[B, k+1, j]$ respectively, where both β and γ are $\vee$ gates. We show that the input sets of the circuits rooted at β and γ are disjoint. This in turn implies that the circuit G_n is multilinear.

Lemma 3.3 The interval indicated in the label of any $\vee$ node in the circuit rooted at β is a subinterval of $[i, k]$.

Proof: We can assume that G_n is layered, without any loss of generality, with the leaves at level 0. The proof is by induction on level h of the circuit rooted at β, $0 \leq h \leq t$, β being at level t.

For the base case $h = t$, $[i, k]$ is a subinterval of itself.

By the inductive hypothesis, all $\vee$ nodes at level h have labels marked with a subinterval of $[i, k]$.

We will show that all $\vee$ nodes at level $h-2$ have the same property (note that there are no $\vee$ nodes in level $h-1$). Let β_1 be an $\vee$ node at level $h-2$ labeled $[A, m_1, m_2]$. By construction, there is an $\wedge$ gate β_2 (β_1's parent) at level $h-1$ labeled $[B, C, n_1, n_2, n_3]$, where $A = B$ or $A = C$ and $n_1 \leq n_3 < n_2$. Moreover, $n_1 \leq m_1 \leq m_2 \leq n_3$ if $A = B$ and $n_3 + 1 \leq m_1 \leq m_2 \leq n_2$ if $A = C$. Also by construction, there is an $\vee$ gate β_3 (β_2's parent) at level h labeled $[D, n_1, n_2]$. By inductive hypothesis, $i \leq n_1 \leq n_2 \leq k$. Therefore, $i \leq m_1 \leq m_2 \leq k$. □

Corollary 3.1 All inputs to the circuit rooted at β are from the set $\{x_i, x_{i+1}, \ldots, x_k, \bar{x}_i, \bar{x}_{i+1}, \ldots, \bar{x}_k\}$.

Proof: By the above lemma, the interval indicated in the label of any $\vee$ node at level 1 is of the form $[m, m], i \leq m \leq k$. The result follows from the input assignment rules stated above. □

The inputs to the circuit rooted at γ must be from the set $\{x_{k+1}, x_{k+2}, \ldots, x_j, \bar{x}_{k+1}, \bar{x}_{k+2}, \ldots, \bar{x}_j\}$, by a similar argument.

4 Lower Bounds for Bipartite Perfect Matching

In this section, we prove exponential size lower bounds for computing the bipartite perfect matching function using (a) bijective circuits and (b) consistent multilinear circuits.

The bipartite perfect matching function BPM : $\{0,1\}^{n^2} \rightarrow \{0,1\}$ i takes as input the standard $n \times n$ adjacency matrix representation of a bipartite graph G and outputs 1 if and only if G has a perfect matching. Note that BPM computes the Boolean permanent of the input matrix. BPM is a monotone function and each of its prime implicants correspond to a perfect matching of the complete bipartite graph on $2n$ vertices. Let S_n be the set of all permutation functions on n elements and for each $\pi \in S_n$ let $p_\pi = \bigwedge_{i=1}^{n} x_{i,\pi(i)}$. Then, PI(BPM) is exactly $\{p_\pi \mid \pi \in S_n\}$. We shall refer to the set of variables in p_π as $var(p_\pi)$.

It is easy to verify that for any $\pi, \sigma \in S_n$, $\mid var(p_\pi) - var(p_\sigma) \mid \geq 2$. Therefore,

Fact 4.1 BPM is a suitable function.

4.1 Bijective Circuits

Let $m = n^2$ and let B_m be a bijective Boolean circuit with $2m + 2$ input nodes labeled from the set $\{x_{i,j}, \bar{x}_{i,j}, 0, 1 \mid 1 \leq i, j \leq n\}$, computing BPM.

Thus, by fact 4.1 and theorem 3.1, the consistent monomials of $P(B_m)$ have the form $\bigwedge_{i=1}^{n} x_{i,\pi(i)}^{k_i}$, where each k_i is an integer (we shall loosely refer to each consistent monomial in $P(B_m)$ as a *permutation*). That is, $P(B_m)$ is not necessarily multilinear, has exactly $n!$ consistent monomials and possibly some inconsistent monomials. Note that bijective circuits for BPM need not be monotone due to the presence of inconsistent monomials. Thus, Razborov's super-polynomial size lower bound [9] does not apply.

4.1.1 Adaptation of Jerrum and Snir's Framework

Throughout this section, we shall use the result that the consistent monomials of $P(B_m)$ are monotone and therefore, any term computed at any node of a consistent parse-graph of B_m is a conjunction of positive literals only. In this section, we generalize the lower bound arguments of [5] to hold for bijective circuits. The main differences between this model and that used in [5] are that the circuits used here need not be multilinear and could have limited usage of negations through inconsistent monomials.

Definition 4.1 For an $\wedge$-node α, let $m(\alpha)$ be the number of consistent parse-graphs in which α appears.

Definition 4.2 An $\wedge$-node is said to be *(r,d)-significant* for $1 \leq r \leq n$ and $0 \leq d \leq \lfloor \frac{r}{2} \rfloor$, if it participates in a consistent parse-graph with a term that has r variables, d of which are contributed by one of its immediate predecessors alone.

We note that if an $\wedge$-node α is not (r,d)-significant for any (r,d), then $m(\alpha) = 0$. Moreover, as a part of the proof of lemma 4.1 below, we show that α cannot be (r,d)-significant for more than one (r,d) pair.

Definition 4.3 Let H be a subgraph of a consistent parse-graph G. Define weight of H as follows: $W(H) = \sum_{\alpha \in \wedge-\text{nodes}(H)} \frac{1}{m(\alpha)}$, where $\wedge$-*nodes*(H) denotes the set of $\wedge$-nodes in H.

A lemma similar to the one below was proven in [5] for their model. To prove it for bijective circuits, we need to take into account the fact that the parse-graphs are not necessarily trees and that the circuit could be non-monotone.

Lemma 4.1 If α is an (r,d)-significant $\wedge$-node of B_m, then $m(\alpha) \leq d!(r-d)!(n-r)!$.

Proof: Let β and γ be the immediate predecessors of α in B_m. Let G be a consistent parse-graph in which α appears with an r-variable term. Let a be the term formed at β and b be the term formed at γ 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 (see figure 1). Let c be a term such that $a \cdot b \cdot c$ is the n-variable term formed at the root of G. Clearly, c has $n - r$ variables that are not in a or b. Note that the sets $var(a)$, $var(b)$ and $var(c)$ may have non-empty intersections since G is a parse-graph as opposed to a parse-tree.

Let α participate in another consistent parse-graph G'. Since α is an $\wedge$-node, β and γ participate in G' as well. In G', let a' be the term formed at β and b' be the term formed at γ. Let $a' \cdot b' \cdot c'$ be the term formed at the root of G'. Now since B_m is bijective, the term computed by G' must be different from that computed by G. Moreover, since α is an $\wedge$ node, there must be parse-graphs in B_m that compute the terms $a \cdot b \cdot c'$, $a \cdot b' \cdot c$ etc. But every term computed by a consistent parse-graph of B_m is a permutation. Thus, the number of consistent parse-graphs in which α participates is the product of the number of distinct a's, b's and c's, where a is a term formed at β, b is a term formed at γ and c is a term such that $a \cdot b \cdot c$ is a permutation.

We first show that for a fixed r and d, $m(\alpha)$ cannot exceed $d!(r-d)!(n-r)!$. Let

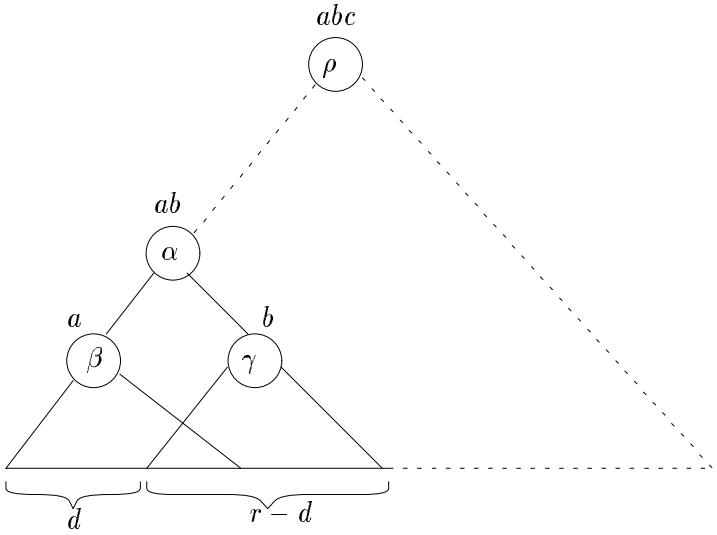

Figure 1: (r,d)-significant node α in a bijective circuit B_m with output node ρ.

$$\begin{aligned} A &= var(a) - \{var(b) \cup var(c)\} \\ B &= var(b) - \{var(a) \cup var(c)\} \\ C &= var(c) - \{var(a) \cup var(b)\} \end{aligned}$$

It is easy to see that α can participate in any consistent parse-graph that computes the term $a' \cdot b' \cdot c'$, where the variables in a' are obtainable by permuting within the indices of the variables in A and b', c' are obtainable similarly from B and C, respectively. Therefore, α can participate in $|A|!\,|B|!\,|C|!$ consistent parse-graphs. Clearly, $|A|!\,|B|!\,|C|! \leq d!\,(r-d)!\,(n-r)!$.

Suppose α participates in a consistent parse-graph G' different from the $|A|!\,|B|!\,|C|!$ parse-graphs described above such that in G', β computes the term a', γ computes the term b' and the output node computes the term $a' \cdot b' \cdot c'$. Now, by choice of G' and bijectivity of B_m, $a' \cdot b' \cdot c'$ must be a permutation different from the $|A|!\,|B|!\,|C|!$ permutations above. Thus, the indices of the variables in at least one of a', b' or c' are not obtainable by permuting within the indices of the variables in A, B or C, respectively. Without loss of generality, suppose this is true for a' (the argument is symmetrical for b' or c'). Then, there must be a variable x_{ij} in $var(a')$ such that either i is not the row index of any variable in A, or j is not the column index of any variable in A or both. But then, the term $a' \cdot b \cdot c$ is not a permutation and hence must be an inconsistent monomial of $P(B_m)$. This implies that the term $a' \cdot b \cdot c$ must contain at least one negative literal. But that is impossible since G and G' are both consistent parse-graphs and hence the terms a', b or c do not contain any negative literals. Therefore, G' cannot exist. Note that we did not make any assumptions about a' or b' other than that $a' \cdot b' \cdot c'$ is a permutation different from those counted earlier. Therefore, this argument holds even if $|var(a' \cdot b')| = r'$ and $|var(a') - var(b')| = d'$, where $r' \neq r$ and $d' \neq d$. Since this means that the node α contributes to $m(\alpha)$ only for a single value of r and a single value of d, $m(\alpha)$ is bounded above by $d!(r-d)!(n-r)!$. □

The lemma below is motivated by theorem 3.3 in [5], where $\{G_i \mid 1 \leq i \leq n!\}$, are the consistent parse-graphs of B_m. Let $\Lambda = \{\wedge\text{-node } \alpha \mid m(\alpha) \geq 1\}$.

Lemma 4.2 $\sum_{i=1}^{n!} W(G_i) = |\Lambda|$.

Proof: By definition,

$$\sum_{i=1}^{n!} W(G_i) = \sum_{i=1}^{n!} \sum_{\alpha \in \wedge\text{-nodes}(G_i)} \frac{1}{m(\alpha)}.$$

Fix an $\wedge$-node α. For each consistent parse-graph G_i, the contribution by α to the sum on the right-hand side of the above equation is either 0 (if α does not occur in it) or $\frac{1}{m(\alpha)}$. Thus, the total contribution by α is $m(\alpha)\frac{1}{m(\alpha)} = 1$ and therefore the right-hand side is the number of $\wedge$-nodes in Λ. $\square$

To obtain a lower bound on the weight of a parse-graph G, we consider the number of input variables covered by G, instead of the notion of degree used in [5]. This is primarily because $P(B_m)$ is not necessarily multilinear in our model.

For any subgraph H of a parse-graph of B_m, let $v(H)$ denote the number of positive variables in the term associated with H. Let $c(r,d) = d!(r-d)!(n-r)!$. The lemma below is adapted from theorem 3.4 in [5].

Lemma 4.3 Let H be a subgraph of of any consistent parse-graph G. Then, $W(H) \geq \sum_{i=2}^{v(H)} \frac{1}{c(i,1)}$.

Proof: The proof is by induction on the number of nodes in H. For the base case, H has a single node. Since it must be a leaf, $v(H) = 1$, and since H has no $\wedge$-nodes, $W(H) = 0$. Thus, the lemma holds.

For the induction step, let α be the root node of H and let $v(H) = r$. Without loss of generality, we assume that α is an $\wedge$-node. Since α appears in G, it must be (r,d)-significant for some d, $0 \leq d \leq \lfloor \frac{r}{2} \rfloor$. Let α have immediate predecessors β and γ and with β contributing d variables alone (see figure 1). Let H_γ be the subgraph of H rooted at γ. Clearly, $v(H_\gamma) = (r-d)$. Now, there is a subgraph $\tilde{H}_\beta$ rooted at β such that $v(\tilde{H}_\beta) = d$. For each leaf x in the set of d leaves in figure 1, there is a path P_x from x to β that is disjoint from H_γ. Let $\tilde{H}_\beta$ be the edge-induced subgraph of H_β defined on the union of the edge sets of P_x, for all x. We have,

$$W(H) \geq W(\tilde{H}_\beta) + W(H_\gamma) + \frac{1}{m(\alpha)}.$$

Note that $m(\alpha) \geq 1$ since α appears in G. Applying the induction hypothesis to the subgraphs $\tilde{H}_\beta$ and H_γ and using lemma 4.1, we get:

$$W(H) \geq \sum_{i=2}^{d} \frac{1}{c(i,1)} + \sum_{i=2}^{(r-d)} \frac{1}{c(i,1)} + \frac{1}{c(r,d)}.$$

The expression on the right is shown in [5] to be minimum at $d = 1$ in the range $1 \leq d \leq \lfloor \frac{r}{2} \rfloor$. In the range $0 \leq d \leq \lfloor \frac{r}{2} \rfloor$, this expression attains its minimum value at $d = 1$ as well, since $\frac{1}{c(r,0)} \geq 0$. $\square$

The above lemmas lead to the lower bound for BPM using bijective circuits.

Theorem 4.1 Any bijective Boolean circuit B_m requires size $\geq n(2^{n-1} - 1)$ to decide whether a bipartite graph has a perfect matching.

Proof: From lemmas 4.2 and 4.3 it follows that the size of any bijective Boolean circuit for this problem is at least $\sum_{j=1}^{n!} \sum_{i=2}^{v(G_j)} \frac{1}{c(i,1)}$. But $v(G_j) = n$ since every consistent parse-graph of B_m computes a permutation. Therefore, the above expression is equivalent to $n! \sum_{i=2}^{n} \frac{1}{c(i,1)}$. Substituting for $c(i,1)$ in $\sum_{i=2}^{n} \frac{1}{c(i,1)}$ we get, $\sum_{i=2}^{n} \frac{1}{(n-i)!.(i-1)!}$ which is exactly $\frac{2^{n-1}-1}{(n-1)!}$, from which the theorem follows.□

4.2 Consistent Multilinear Circuits

In this section, we show an exponential size lower bound for computing the bipartite perfect matching problem using Boolean circuits that are consistent and multilinear. As in section 4.1.1, we generalize the lower bound arguments in [5], but in another direction. The main differences between consistent multilinear circuits and the model used in [5] are that the circuits used here need not be bijective and could use negations as long as no inconsistent monomials are computed.

Let $m = n^2$ and let B_m be a consistent multilinear Boolean circuit computing BPM. All parse-graphs of B_m are trees. Applying the results in section 2, we know that B_m has exactly $n!$ parse-classes. Moreover, if T is a representative parse-tree of the parse-class PC_π, then the positive term of the monomial computed by T is exactly p_π. We shall refer to each p_π as a *permutation.*

For the lower bound argument, we fix a set of representative parse-trees $\{T_i | 1 \leq i \leq n!\}$, one from each parse-class. Henceforth, by representative parse-trees we shall mean the parse-trees in this set.

The contents of section 4.1.1 apply for consistent multilinear circuits with some modifications. Wherever applicable, we use "representative parse-trees" instead of "consistent parse-graphs" and use the set $\{T_i | 1 \leq i \leq n!\}$ defined above instead of the set $\{G_i | 1 \leq i \leq n!\}$ defined in section 4.1.1. With these substitutions, definitions 4.1, 4.3 apply and lemmas 4.2 and 4.3 hold. We alter definition 4.2 slightly to reflect the fact that all $\wedge$-nodes have 2 children and by definition of multilinearity, their input sets cannot overlap. This, in fact, simplifies the proof of lemma 4.3 for multilinear circuits.

Definition 4.4 An $\wedge$-node is said to be *(r,d)-significant* for $2 \leq r \leq n$ and $1 \leq d \leq \lfloor \frac{r}{2} \rfloor$, if it participates in a representative parse-tree with a positive term that has r variables, d of which are contributed by one of its immediate predecessors alone.

The only major change is in lemma 4.1. Next we prove its analogue for consistent multilinear circuits. The proof differs from that of lemma 4.1 primarily because the consistent monomials in this model need not be monotone. For this model, we can determine $m(\alpha)$ exactly rather than proving an upper bound.

Lemma 4.4 If α is an (r,d)-significant node in B_m, then $m(\alpha) = d!(r-d)!(n-r)!$.

Proof: Let β and γ be the immediate predecessors of α in B_m. Let T be a representative parse-tree in which α appears with an r-variable term. Let a_+ be the positive term formed at β and b_+ be the positive term formed at γ such that $a_+ \cdot b_+$ has $r \geq 1$ variables and a_+ has $1 \leq d \leq \lfloor r/2 \rfloor$ variables that are not in b_+ (see figure 2). Let c be the $(n-r)$-variable term such that $a_+ \cdot b_+ \cdot c_+$ is the positive term formed at the root of T.

Let α participate in another representative parse-tree T'. Since α is an $\wedge$-node, β and γ participate in T' as well. In T', let a'_+ be the positive term formed at β and b'_+ be the positive

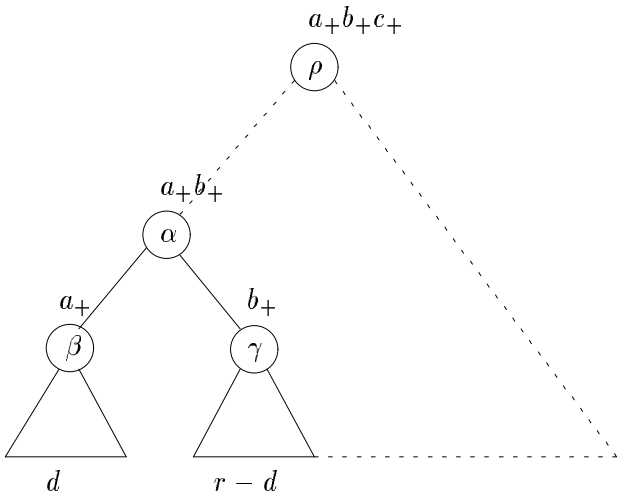

Figure 2: (r,d)-significant node α in consistent multilinear circuit B_m with output node ρ.

term formed at γ. Let $a'_+ \cdot b'_+ \cdot c'_+$ be the positive term formed at the root of G'. Now, since α is an $\wedge$ node, there must be parse-trees in B_m that whose positive terms are $a_+ \cdot b_+ \cdot c'_+$, $a_+ \cdot b'_+ \cdot c_+$ etc. Now, by definition, the positive term of the monomial computed by a representative parse-tree of B_m is a permutation. Thus, the number of representatives in which α participates is the product of the number of distinct a_+'s, b_+'s and c_+'s, where a_+ is a positive term formed at β, b_+ is a positive term formed at γ and c_+ is a positive term such that $a_+ \cdot b_+ \cdot c_+$ is a permutation.

We first show that for a fixed r and d, $m(\alpha)$ is exactly $d!(r-d)!(n-r)!$. It is easy to see that α can participate in any representative parse-tree whose positive term is $a'_+ \cdot b'_+ \cdot c'_+$, where the variables in a'_+ are obtainable by permuting within the indices of the variables in a_+ and b'_+, c'_+ are obtainable similarly from b_+ and c_+, respectively. Therefore, α can participate in at least $d!\,(r-d)!\,(n-r)!$ representative parse-trees.

Suppose α participates in a representative T' different from those counted above, such that the positive term of the monomial computed by T' is $a'_+ \cdot b'_+ \cdot c'_+$. Then, at least one of the terms $a'_+ \cdot b_+ \cdot c_+$, $a_+ \cdot b'_+ \cdot c_+$ etc. is not a permutation and therefore, must be the positive term of an inconsistent monomial of $P(B_m)$, which is impossible since B_m is consistent. Therefore, such a T' cannot exist. It follows that the number of representatives in which α can participate with an r-variable positive term with a child contributing d of those, is exactly $d!(r-d)!(n-r)!$.

Finally, suppose α participates in a representative parse-tree computing a positive term $a'_+ \cdot b'_+$ that has fewer than r variables. Then, $a'_+ \cdot b'_+ \cdot c_+$ must be the positive term of a consistent monomial of $P(B_m)$ with fewer than n variables, which is impossible by lemma 2.1. If $a'_+ \cdot b'_+$ has more than r variables, then there must be a c'_+ with less than $n-r$ variables such that $a'_+ \cdot b'_+ \cdot c'_+$ has exactly n variables, to satisfy lemma 2.2. But since α is an $\wedge$-node, this implies that $P(B_m)$ has a monomial whose positive term is $a_+ \cdot b_+ \cdot c'_+$, which has less than n variables violating lemma 2.1.

For similar reasons, β cannot participate in a consistent parse-graph computing a term that has fewer or more than d variables. Since this means that the node α contributes to $m(\alpha)$ only for a single value of r and a single value of d, $m(\alpha)$ is exactly $d!(r-d)!(n-r)!$. $\square$

This leads to the following theorem whose proof is similar to that of theorem 4.1.

Theorem 4.2 Any consistent multilinear Boolean circuit B_m requires size $\geq n(2^{n-1}-1)$ to

decide whether a bipartite graph has a perfect matching.

Since monotone Boolean circuits are consistent, the above bound holds for monotone multilinear Boolean circuits as well.

4.3 Upper Bounds

The Boolean circuit based on the permanent analogue of Laplace's expansion rule for determinants computes BPM within size $O(n(2^{n-1}-1))$ [5]. This circuit is bijective, monotone and multilinear. Since monotone circuits are trivially consistent, all the lower bounds presented in this section are tight.

5 Separation of Monotone Counting Classes

In this section, we establish a close connection between monotone arithmetic circuits for 0-1 permanent and bijective Boolean circuits for BPM. The lower bound result for bijective circuits in section 4.1 then implies an exponential size lower bound for monotone arithmetic circuits that compute the 0-1 permanent function.

An arithmetic circuit A_n is defined similarly to a Boolean circuit except that the interior nodes are labeled nodes labeled from $\{+,\times\}$ and the leaf nodes labeled from $\{x_i,(1-x_i),0,1|1\leq i\leq n\}$, where each $x_i\in\{0,1\}$. Note that -1 is not supplied as a constant. We say that A_n is *monotone* if the leaves are labeled only from $\{x_i,0,1\mid 1\leq i\leq n\}$. Each A_n computes a unique function $g:\{0,1\}^n\rightarrow\mathcal{N}$. The *size* and *depth* of A_n is defined as before. As in the case of Boolean circuits, we can associate a formal polynomial $P(A_n)$ with an arithmetic circuit A_n. We define the *degree* of A_n to be the degree of $P(A_n)$.

A function $\mathcal{F}:\{0,1\}^*\rightarrow\mathcal{N}$ is represented by a *function family* $\{f_n:\{0,1\}^n\rightarrow\mathcal{N}\mid n\geq 1\}$. An arithmetic circuit family $\{A_n\mid n\geq 1\}$ is said to compute the function $\mathcal{F}$, if for all $n\geq 1$, A_n computes f_n.

Let $m=n^2$ and let A_m be a monotone arithmetic circuit that computes the 0-1 permanent function PERM : $\{0,1\}^m\rightarrow\mathcal{N}$ which takes as input an $n\times n$ 0-1 matrix and outputs its permanent. Since A_m is monotone, the monomials of $P(A_m)$ do not have negative terms and therefore, all monomials are consistent.

Now, let B_m be the monotone Boolean circuit obtained by "Booleanizing" A_m, that is, replacing each $\times$ node with an $\wedge$-node and each $+$ node with an $\vee$ node. It is easy to verify that B_m computes BPM. By the analogues of lemmas 2.1 and 2.2, $P(A_m)$ has at least $n!$ monomials. If it had any more, then on the input that assigns all variables to 1, PERM evaluates to $n!$ but A_m would compute a value strictly greater than $n!$. Thus, $P(A_m)$ has exactly $n!$ monomials. Now, since $P(B_m)$ is formally identical to $P(A_m)$ by construction, it follows that B_m must be bijective. The lower bound below now follows from theorem 4.1.

Theorem 5.1 Any monotone arithmetic circuit A_m requires size $\geq n(2^{(n-1)}-1)$ to compute the 0-1 permanent function.

Proof: Suppose A_m had smaller size. Then, the bijective Boolean circuit obtained by Booleanizing A_m, would compute BPM within size smaller than $n(2^{(n-1)}-1)$. But by theorem 4.1, this is impossible. □

In [18], Venkateswaran characterized popular counting classes such as $\sharp\mathcal{P}$, in terms of arithmetic circuits. Subsequently, Vinay [21] gave a circuit definition of $\sharp\mathcal{LOGCFL}$, the counting

version of the class $\mathcal{LOGCFL}$ [19]. The circuit definition of $\sharp\mathcal{LOGCFL}$ stated below, follows from known simulations in [11, 19]. The uniformity condition used in these definitions is U_D-uniformity, defined by Ruzzo [11].

Theorem 5.2 [19, 18] $\sharp\mathcal{P}$ is the class of functions computable by uniform families of arithmetic circuits within polynomial depth and polynomial degree. $\sharp\mathcal{LOGCFL}$ is the class of functions computable by uniform families of arithmetic circuits within polynomial size and polynomial degree.

In this paper, we consider monotone analogues of the above classes defined using monotone arithmetic circuits.

Definition 5.1 $m\sharp\mathcal{P}$ is the class of functions computable by uniform families of monotone arithmetic circuits within polynomial depth and polynomial degree. $m\sharp\mathcal{LOGCFL}$ is the class of functions computable by uniform families of monotone arithmetic circuits within polynomial size and polynomial degree.

It can be verified that PERM $\in m\sharp\mathcal{P}$. But by theorem 5.1 and the circuit definition of $m\sharp\mathcal{LOGCFL}$, PERM $\notin m\sharp\mathcal{LOGCFL}$. Therefore,

Theorem 5.3 $m\sharp\mathcal{LOGCFL}$ is properly contained in $m\sharp\mathcal{P}$.

We note that this separation also follows from Razborov's super-polynomial size lower bound for monotone circuits that compute the Boolean permanent [9]. Using the argument in the proof of theorem 5.1 above, Razborov's lower bound can be shown to imply that monotone arithmetic circuits for PERM must have at least super-polynomial size. However, in the Booleanizing argument above we have observed that the Boolean circuit so obtained is not only monotone but also bijective leading to an exponential size lower bound. We note that this lower bound on arithmetic circuits computing PERM also does not follow from the work of Valiant [16].

Finally, we note that $\sharp\mathcal{LOGCFL}$ can be separated from $\sharp\mathcal{P}$ if the lower bound in theorem 5.1 can be shown to hold for arithmetic circuits that are not necessarily monotone.

6 Consequences for $(\oplus, \wedge)$ Circuits

A $(\oplus, \wedge)$ circuit C_n is a rooted, directed, acyclic graph with interior nodes labeled from $\{\oplus, \wedge\}$ and the leaf nodes labeled from $\{x_i, 0 | \ 1 \leq i \leq n\}$. Note that $\{\oplus, \wedge\}$ is an incomplete Boolean basis. Supplying the constant 1 to a $(\oplus, \wedge)$ circuit makes it computationally equivalent to a general Boolean circuit. The *size* and *depth* of $(\oplus, \wedge)$ circuits are defined as for Boolean circuits.

Circuits over $(\oplus, \wedge)$ can only compute Boolean functions f such that $f(\mathbf{0}) = 0$ (note that this includes all monotone functions). In fact, $(\oplus, \wedge)$ circuits can compute all such functions, BPM in particular, within size polynomial in that of the full Boolean basis [6]. Thus, by Razbarov's super-polynomial size lower bound on monotone circuits for BPM [9], it follows that $(\oplus, \wedge)$ circuits are at least super-polynomially more powerful than monotone circuits for this function since they can compute it within polynomial size.

Since $(\oplus, \wedge)$ circuits are as powerful as general Boolean circuits for the functions that they can compute, it is meaningful to identify restrictions on $(\oplus, \wedge)$ circuits. With every C_n we can associate a formal polynomial $P(C_n)$. We define the notion of parse-graphs of a $(\oplus, \wedge)$ circuit in a manner analogous to section 2, by replacing $\vee$ with $\oplus$ in definition 2.1. As before, each parse-graph of C_n computes a monomial of $P(C_n)$.

Definition 6.1 A $(\oplus, \wedge)$ circuit C_n is said to be *multilinear* if each variable in $\{x_i | \ 1 \leq i \leq n\}$ appears at most once in any monomial of $P(C_n)$.

Since $(\oplus, \wedge)$ circuits do not have negated inputs, no monomial of $P(C_n)$ is inconsistent in the sense of definition 2.3. However, $(\oplus, \wedge)$ circuits do have a property that is analogous to inconsistency in Boolean circuits in that they both result in cancellation.

Definition 6.2 A $(\oplus, \wedge)$ circuit C_n is said to have *even recurrence* if there is a monomial that occurs an even number of times in $P(C_n)$.

We note that the $(\oplus, \wedge)$ circuit based on the ring-sum expansion [13] of a Boolean function f is multilinear and has no even recurrence.

The set of distinct monomials of $P(C_n)$ can be partitioned into: those that occur an even number of times and those that do not. We shall refer to the former as *even monomials* and the latter as *odd monomials*. We can now prove the analogues of lemmas 2.1 and 2.2 using the same proof ideas. Let C_n be a $(\oplus, \wedge)$ circuit computing a monotone Boolean function f.

Lemma 6.1 For each odd monomial ρ of $P(C_n)$, there exists a term $t \in \text{PI}(f)$ such that $var(t) \subseteq var(\rho)$.

Lemma 6.2 For all terms $t \in \text{PI}(f)$, there exists an odd monomial ρ of $P(C_n)$ such that $var(t) = var(\rho)$.

The next lemma relates monotone Boolean circuits to $(\oplus, \wedge)$ circuits without even recurrence.

Lemma 6.3 Let f be a monotone Boolean function. If f is computable by a $(\oplus, \wedge)$ circuit without even recurrence within size s, then there is a monotone Boolean circuit that computes f within the same size s.

Proof: Let $\mathcal{C}$ be a $(\oplus, \wedge)$ circuit of size s that computes f without even recurrence. By definition, $P(\mathcal{C})$ has no even monomials. Consider the monotone Boolean circuit $\mathcal{B}$ obtained by replacing the $\oplus$ gates of $\mathcal{C}$ with $\vee$ gates. The polynomials $P(\mathcal{C})$ and $P(\mathcal{B})$ are formally identical. Thus, by lemma 6.1 every monomial of $P(\mathcal{B})$ contains a prime implicant of f. Moreover since $\mathcal{B}$ is monotone, for every prime implicant t of f there is a monomial ρ of $P(\mathcal{B})$ such that $var(t) = var(\rho)$, by lemma 6.2. It follows that by applying the absorption axiom of Boolean algebra, $x + xy = x$, $P(\mathcal{B})$ can be simplified to $\bigvee_{t \in PI(f)} t$. Thus, $\mathcal{B}$ computes f. $\square$

Note that if the circuit $\mathcal{C}$ above has even recurrence, then $P(\mathcal{B})$ has consistent monomials that need not contain any prime implicant and therefore, $\mathcal{B}$ does not necessarily compute the function f.

Lemma 6.3 combined with the super-polynomial size lower bound on monotone circuits for BPM [9] yields the following theorem:

Theorem 6.1 $(\oplus, \wedge)$ circuits without even recurrence require super-polynomial size to compute BPM.

Proof: Let $\mathcal{C}$ be a $(\oplus, \wedge)$ circuit of size less than super-polynomial that computes BPM without even recurrence. Then, by the above lemma, there is a monotone Boolean circuit of the same size for BPM. But this is impossible due to the lower bound by Razborov [9]. □

We can improve this bound to exponential if $(\oplus, \wedge)$ circuits are restricted further with multilinearity.

Theorem 6.2 Multilinear $(\oplus, \wedge)$ circuits without even recurrence require exponential size to compute BPM.

Proof: Let $\mathcal{C}$ be a multilinear $(\oplus, \wedge)$ circuit of size s that computes BPM without even recurrence. Then, by the proof of lemma 6.3, the Boolean circuit $\mathcal{B}$ obtained by replacing the $\oplus$ gates of $\mathcal{C}$ with $\vee$ gates is monotone, has size s and computes BPM. Moreover, $\mathcal{B}$ is multilinear since $P(\mathcal{B})$ is formally identical to $P(\mathcal{C})$ and $\mathcal{C}$ is multilinear. But by theorem 4.2, $\mathcal{B}$ must have size $\geq n(2^{n-1} - 1)$. Thus, $s \geq n(2^{n-1} - 1)$. □

The results in this section can be summarized as follows. It is known [6] that $(\oplus, \wedge)$ circuits can compute BPM within polynomial size. We identify two restrictions on $(\oplus, \wedge)$ circuits: (a) multilinearity and (b) no even recurrence. We show that $(\oplus, \wedge)$ circuits for BPM require super-polynomial size when restricted only with (b) and exponential size when restricted with both (a) and (b). The size requirement for BPM using $(\oplus, \wedge)$ circuits restricted only with (a) remains open.

7 Homogeneous Circuits

The notion of homogeneity has traditionally been associated with polynomials, where a homogeneous polynomial is one in which all the monomials have the same degree. But since every Boolean function can be expressed as a multilinear polynomial, the homogeneity property translates to requiring that each monomial of such a polynomial has the same number of literals.

Define a Boolean function f to be *homogeneous* if for all $t \in \text{PI}(f)$, $|var(t)|$ is the same. Since $var(t_+) \cap var(t_-) = \emptyset$ by fact 2.1, this is equivalent to requiring that for all $t \in \text{PI}(f)$, $|var(t_+)| + |var(t_-)|$ is the same. However, if f is monotone and homogeneous, then for all $t \in \text{PI}(f)$, $|var(t_+)|$ is the same. This motivates our notion of homogeneous circuits.

Definition 7.1 A Boolean circuit B_n is said to be **homogeneous** if all consistent monomials of $P(B_n)$ have the same number of positive literals.

Unlike bijective and multilinear circuits, homogeneous circuits cannot compute all Boolean functions. But they can compute all monotone homogeneous functions. For instance, homogeneous circuits can compute the BPM and MAJORITY functions but not the PARITY function. All of these functions are homogeneous, but the first two are monotone and the last is not. For any monotone homogeneous function f, the depth 2 circuit based on the representation of f as a disjunction of its prime implicants is homogeneous.

We now show that consistent homogeneous circuits require exponential size to compute BPM. Let $m = n^2$ and let B_m be a consistent homogeneous Boolean circuit computing BPM. Applying the results in the section 2, we know that B_m has exactly $n!$ parse-classes, one for each $\pi \in S_n$. Moreover, if G is a representative parse-graph of the parse-class PC_π computing the monomial ρ, then $var(\rho_+) = var(p_\pi)$.

Now, by lemma 2.1, every consistent monomial of $P(B_m)$ must have at least n positive variables. But since the representatives have exactly n positive variables, every consistent monomial of $P(B_m)$ must also have exactly n positive variables, by definition of homogeneity. But B_m is consistent, therefore for all monomials ρ of $P(B_m)$, $var(\rho_+) = var(p_\pi)$, for some $\pi \in S_n$. Thus every parse-graph of B_m is a representative of some parse-class.

As in section 4, we adapt the combinatorial framework developed by Jerrum and Snir [5] to consistent homogeneous circuits. The main difference between the consistent homogeneous model and that used in [5] are that the circuits used here need not be bijective or multilinear and could use negations as long as no inconsistent monomials are computed.

The lower bound is proved using a set of representatives $\{G_i | 1 \leq i \leq n!\}$, one from each parse-class instead of the consistent parse-graphs used in section 4.1.1. The contents of section 4.1.1 apply for consistent homogeneous circuits with some very minor changes. We use "representative parse-graphs," in place of "consistent parse-graphs" wherever applicable. In the proof of lemma 4.1, we consider only the positive terms as in the proof of lemma 4.4. Furthermore, the proof need not proceed beyond the point where existence of an inconsistent monomial is deduced. The lemmas 4.2 and 4.3 hold without change and we have,

Theorem 7.1 Any consistent homogeneous circuit B_m requires size $\geq n(2^{n-1} - 1)$ to decide whether a bipartite graph has a perfect matching.

8 Lower Bounds for Hamiltonian Cycle

The Hamiltonian cycle function HC : $\{0,1\}^{n^2} \rightarrow \{0,1\}$ takes as input the standard $n \times n$ adjacency matrix representation of a graph G and outputs 1 if and only if G has a Hamiltonian cycle. HC is a monotone function and each of its prime implicants correspond to a Hamiltonian cycle of the complete graph on n vertices. Let C_n be the set of all cyclic permutation functions on n elements and for each $\pi \in C_n$ let $p_\pi = \bigwedge_{i=1}^{n} x_{i,\pi(i)}$. Then, PI(HC) is exactly $\{p_\pi \mid \pi \in C_n\}$. As in the case of BPM.

Fact 8.1 HC is a suitable function.

Since HC is a monotone function, lemmas 2.1 and 2.2 hold and since it is also suitable, lemma 3.1 and theorem 3.1 hold. Note that $\mid C_n \mid = (n-1)!$, therefore, any Boolean circuit for HC has $(n-1)!$ parse-classes.

The arguments used in section 4 can be adapted to obtain exponential size lower bounds for computing HC using (i) bijective or (ii) consistent multilinear Boolean circuits. We outline the adaptation for the bijective model, the adaptation for consistent multilinear circuits then follows easily. Given an adjacency matrix of an n-vertex graph as input, let B_m be a bijective Boolean circuit for HC, where $m = n^2$.

Theorem 8.1 Any bijective Boolean circuit B_m requires size $\geq n(2^{n-1} - 1)$ to decide whether a graph has a Hamiltonian cycle.

Proof Sketch: The steps in the proof follow that of theorem 4.1. Defining (r,d)-significant node as before, the analogue of lemma 4.1 is stated below. The proof of this lemma is almost identical to that of lemma 4.1, considering only cyclic permutations in the place of all permutations.

restriction	*circuit size*
B	$\Omega(n.2^{n-1})$
B, C	"
B, M	"
B, C, M	"
C	?
M	?
M, C	$\Omega(n.2^{n-1})$
none	$O(n^{5/2})$ [4]

Figure 3: Circuit sizes for computing the Boolean permanent in various restricted models.

Lemma 8.1 *Let α be an (r,d)-significant $\wedge$-node in B_m. Then*

$$m(\alpha) \leq \begin{cases} (d-1)!(r-d-1)!(n-r-1)! & \text{if } r < n \\ (d-1)!(n-d-1)! & \text{if } r = n\,. \end{cases}$$

Lemma 4.2 holds without any modifications and lemma 4.3 holds as well by choosing $c(r,d)$ as suggested in [5]:

$$c(r,d) = \begin{cases} (d-1)!(r-d-1)!(n-r-1)! & \text{if } r < n \\ (d-1)!(n-d-1)! & \text{if } r = n\,. \end{cases}$$

As in the proof of theorem 4.1 it follows from these lemmas that the size of any bijective Boolean circuit for the Hamiltonian circuit problem is at least $(n-1)! \cdot \sum_{i=2}^{(n-1)} \frac{1}{c(i,1)}$. This expression evaluates to $n^2.2^{n-3}$ from which the theorem follows.□

The above size lower bound implies a linear depth lower bound for computing the HC using bijective circuits.

Moreover, as in the case of BPM, Jerrum and Snir [5] also show an $O(n^2.2^{n-3})$ size upper bound for computing the Hamiltonian circuit polynomial in their model. Since our model is a generalization of theirs, the size lower bound above in tight.

9 Concluding Remarks

In this paper we identify the following three restrictions on Boolean circuits: bijectivity (B), consistency (C) and multilinearity (M). The size lower bounds for computing the Boolean permanent function in various restricted models are summarized in figure 3. The exponential size lower bounds imply linear depth lower bounds in all these models. This immediately implies that if an $\mathcal{NC}$ circuit for BPM exists, it cannot be bijective or consistent multilinear.

Studying restricted models has many benefits such as identifying proof techniques that are applicable in the general case, and understanding the nature of difficulty in proving lower bounds. We hope that the restricted models considered in this paper and the lower bounds in these models would be likewise useful. This work raises several interesting questions. We list a few here:

- Are there other functions for which the approach presented here can be used to derive non-trivial lower bounds? The functions BPM and HC considered in this paper are both monotone. What about non-monotone functions such as CNF-SAT?

- Are there other interesting restrictions for which lower bounds for natural functions can be obtained?

  In our definition of homogeneity, we require that the consistent formal monomials have the same number of *positive* literals. A more natural variation of homogeneous circuits is one in which the formal monomials have the same number of literals. Such circuits can compute all homogeneous functions. It would be interesting to be able to prove a lower bound for BPM in this model.

- Do consistent circuits require exponential size to compute the Boolean permanent? Answering this in the affirmative would have two interesting consequences: (i) since monotone circuits are necessarily consistent, this would improve Razborov's super-polynomial lower bound [9] to exponential; (b) it would establish that cancellations are crucial for computing the Boolean permanent within polynomial size, thereby allowing us to comprehend the power of negation at a finer granularity than previously possible.

- We would like to be able to fill in the gaps in the table in figure 3, either with lower bounds or by exhibiting polynomial size restricted circuits.

- In all the examples of natural restricted circuits that we have looked at, bijectivity seems to appear only in conjunction with multilinearity. Are there natural circuits that are bijective but not multilinear?

- What can be said about the size requirements of $(\oplus, \wedge)$ circuits for BPM that are restricted only with multilinearity?

Acknowledgements We thank Paul Beame and Martin Tompa for their comments on an earlier version of this paper. We also thank Mauricio Karchmer for pointing out that $(\oplus, \wedge)$ circuits can compute matching within polynomial size.

References

[1] R. Boppana and M. Sipser, *The complexity of finite functions*, In *The Handbook iof Theoretical Computer Science*, J. Van Leeeuwen ed., Elsevier Science Publishers B.V. (1990) 759-804.

[2] A. Chandra, L. Stockmeyer and U. Vishkin, *Constant depth reducibility*, SIAM J. Comput. 13:2 (1984), 423-439.

[3] M. Grigni and M. Sipser, *Monotone separation of Logspace from* NC^1, Proc. 6th IEEE Structures (1991), 294-298.

[4] J.E. Hopcroft and R.M. Karp, *A* $n^{5/2}$ *algorithm for maximum matching in bipartite graphs*, SIAM J. Comput. 2 (1973), 225-231.

[5] M. Jerrum and M. Snir, *Some exact complexity results for straight-line computations over semi-rings*, J. Assoc. Comput. Mach. 29 (1982), 874-897.

[6] M. Karchmer, *private communication.*

[7] R. Raz and A. Wigderson, *Monotone circuits for matching require linear depth*, Proc. 22nd annual ACM STOC (1990), 287-292.

[8] A. A. Razborov, *Lower bounds on the monotone complexity of some Boolean functions*, Soviet Mathematics Doklady 31 (1985), 354-357.

[9] A.A. Razborov, *A lower bound on the monotone network complexity of the logical permanent*, Mathematischi Zametki 37 (1985), 887-900.

[10] A. A. Razborov, *Lower bounds for the size of bounded depth circuits with basis* $\{\oplus, \wedge\}$, Math. Notes of the Academy of Sciences of the USSR 41:4 (1987), 333-338.

[11] W. L. Ruzzo, *On uniform circuit complexity*, JCSS 22 (1981), 365-383.

[12] M. Santha and C. Wilson *Polynomial size constant depth circuits with a limited number of negations*, Proc. 8th STACS (1991), 228-237.

[13] J. E. Savage, *The complexity of computing*, R. E. Kreiger Publishing Co., Malabar, Florida, 1987.

[14] É. Tardos, *The gap between monotone and non-monotone circuit complexity is exponential*, Combinatorica 7 (1987), 141-142.

[15] L. Valiant, *The complexity of computing the permanent*, Theoretical Computer Science 8 (1979), 189-201.

[16] L. Valiant, *Negation can be exponentially powerful*, Theoretical Computer Science 12 (1980), 303-314.

[17] L. Valiant, *Why is Boolean complexity theory difficult?* In *Boolean Function Complexity Theory*, edited by M.S. Paterson, London Mathematical Society. Lecture Notes Series 169, Cambridge University Press, 1992.

[18] H. Venkateswaran, *Circuit definitions of nondeterministic complexity classes*, SIAM J. Comput. 21 (1992) 655-670.

[19] H. Venkateswaran, *Properties that characterize LOGCFL*, JCSS 43-2 (1991), 380-404.

[20] H. Venkateswaran, *Two dynamic programming algorithms for which interpreted pebbling helps*, Information and Computation 92:2 (1991), 237-252.

[21] V. Vinay, *Counting auxilliary pushdown automata and semi-unbounded arithmetic circuits*, Proc. 6th annual IEEE Conference on Structure in Complexity Theory (1991), 270-284.