The Complexity of Finite Functions

Ravi B. Boppana *

Department of Computer Science
New York University
New York, NY 10012

Michael Sipser **

Mathematics Department
Massachusetts Institute of Technology
Cambridge, MA 02139

August 1989

Abstract: This paper surveys the recent results on the complexity of Boolean functions in terms of Boolean circuits, formulas, and branching programs. The primary aim is to give accessible proofs of the more difficult theorems proving lower bounds on the complexity of specific functions in restricted computational models. These include bounded depth circuits, monotone circuits, and bounded width branching programs. Application to other areas are described including Turing machine complexity, relativization, and first order definability.

Keywords: Alternation, Approximation, Boolean Circuit, Boolean Formula, Boolean Function, Branching Program, Clique Function, Connectivity Function, Depth, Fanin, Fanout, First Order Definability, Gate, Log Time Hierarchy, Majority Function, Monotone Circuit, Monotone Function, Nonconstructive Proof, Nondeterminism, Oracle, Parity Function, Polynomial Time Hierarchy, Relativization, Restriction, Size, Slice Function, Space, Sunflower, Symmetric Function, Threshold Function, Time, Turing Machine.

* Supported by an NSF Mathematical Sciences Postdoctoral Fellowship and NSF Grant CCR-8902522.

** Supported by NSF Grant DCR-8602062 and Air Force Contract AFOSR-86-0078.

1. Introduction

The classification of problems according to computational difficulty comprises two subdisciplines of different character. One, the theory of algorithms, gives upper bounds on the amount of computational resource needed to solve particular problems. This endeavor has enjoyed much success in recent years with a large number of strong results and fruitful connections with other branches of mathematics and engineering. The other, called the theory of computational complexity, is an attempt to show that certain problems cannot be solved efficiently by establishing lower bounds on their inherent computational difficulty. This has not been as successful. Except in a few special circumstances, we have been unable to demonstrate that particular problems are computationally difficult, even though there are many which appear to be so. The most fundamental questions, such as the famous P versus NP question which simply asks whether it is harder to find a proof than to check one, remain far beyond our present abilities. This chapter surveys the present state of understanding on this and related questions in complexity theory.

The difficulty in proving that problems have high complexity seems to lie in the nature of the adversary: the algorithm. Fast algorithms may work in a counterintuitive fashion, using deep, devious, and fiendishly clever ideas. How does one prove that there is no clever way to quickly solve a problem? This is the issue confronting the complexity theorist. One way to make some progress on this is to limit the capabilities of the computational model, thereby limiting the class of potential algorithms. In this way it has been possible to achieve some interesting results. Perhaps these may lead the way to lower bounds for more powerful computational models.

There is by now an extensive literature on bounds for various models of computation. We have excluded some of these from this survey using the following criteria. First, we will concentrate on recent, mostly combinatorial bounds for Boolean circuits, formulas, and branching programs. We will not include the older work on lower bounds via the diagonalization method. Though important early progress was made in that way, the relativization results of Baker, Gill, and Solovay (1975) show that such methods from recursive function theory are inadequate for the remaining interesting questions. Second, we will mostly focus on bounds that differentiate polynomial from nonpolynomial rather than lower level bounds. This reflects our feeling that this type of result is closer to the interesting unsolved questions. Finally, in several cases we refrain from giving the tightest result known to keep the exposition as clear as possible.

1.1. Brief History

Circuit complexity theory dates from Shannon's seminal 1949 paper. There he proposed the size of the smallest circuit computing a function as a measure of its complexity. His motivation was simply to minimize the hardware necessary for computation. He proved an upper bound on the complexity of all n input functions and used a counting argument to show that for most functions this bound is not too far off.

The 1960's saw the introduction of the algorithm as a way of measuring the complexity of functions. Edmonds (1965) gave a polynomial time algorithm for the matching problem and foresaw the issue of polynomial versus exponential complexity. Hartmanis and Stearns (1965) formalized this measure as time on a Turing machine. Savage (1972) established a close relationship between the time required to compute a function on a Turing machine and its circuit complexity. At this point the importance of proving good lower bounds on circuit size was apparent, but it was also becoming clear that this was going to be difficult to accomplish. See Harper and Savage (1972) for a discussion of this. By the end of the 1970's essentially the only results known were the linear lower bound of Paul (1977) and the nearly quadratic lower bound of Nečiporuk (1966) for the special case of formula size.

A new direction in the 1980's brought on a burst of activity. By placing sufficiently strong restrictions on the class of circuits it became possible to prove strong lower bounds. The first results of this kind were independently obtained by Furst, Saxe, and Sipser (1984) and Ajtai (1983) for the bounded depth circuit model. Razborov (1985a) and subsequently Andreev (1985) gave strong lower bounds for the monotone circuit model. Numerous papers strengthening these results and giving others in the same vein have since appeared.

Our chapter emphasizes this latter work giving only a sketchy overview of the proceeding period. For a more comprehensive discussion of the earlier work see the survey paper of Paterson (1976), and the book of Savage (1976). Much of the recent work also is covered in the books of Wegener (1987) and Dunne (1988).

Contents

2. General Circuits

In an early paper Shannon (1949) considered the size of Boolean circuits as a measure of computational difficulty. Circuits are an attractive model for proving lower bounds for several reasons. They are closely related in computational power to Turing machines so that a good lower bound on circuit size directly gives a lower bound on time complexity. Among computational models the circuit model has an especially simple definition and so may be more amenable to combinatorial analysis. Even so, we are currently able to prove only very weak lower bounds on circuit size.

A *Boolean circuit* is a directed acyclic graph. The nodes of indegree 0 are called *inputs*, and are labeled with a variable x_i or with a constant 0 or 1. The nodes of indegree $k > 0$ are called *gates* and are labeled with a Boolean function on k inputs. We refer to the indegree of a node as its *fanin* and its outdegree as its *fanout*. Unless otherwise specified we restrict to the Boolean functions AND, OR, and NOT. One of the nodes is designated the *output* node. The *size* is the number of gates, and the *depth* is the maximum distance from an input to the output. A *Boolean formula* is a special kind of circuit whose underlying graph is a tree.

A Boolean circuit represents a Boolean function in a natural way. Let N denote the natural numbers, $\{0,1\}^n$ the set of binary strings of length n, and $\{0,1\}^*$ the set of all finite binary strings. Let $f\colon \{0,1\}^n \to \{0,1\}$. Then $C(f)$ is the size of the smallest circuit representing f. Let $g\colon \{0,1\}^* \to \{0,1\}$ and $h\colon N \to N$. Say g has *circuit complexity* h if for all n, $C(g_n) = h(n)$ where g_n is g restricted to $\{0,1\}^n$. A *language* is a subset of $\{0,1\}^*$. The circuit complexity of a language is that of its characteristic function.

2.1. Boolean Circuits and Turing Machines

In this subsection we establish a relationship between the circuit complexity of a problem and the amount of time that is required to solve it. First we must select a model of computation on which to measure time. The *Turing machine (Tm)* model was proposed by Alan Turing (1936) as a means of formalizing the notion of effective procedure. This intuitively appealing model serves as a convenient foundation for many results in complexity theory. The choice is arbitrary among the many polynomially equivalent models. The complexity of Turing machine computations was first considered by Hartmanis and Stearns (1965). We briefly review here the variant of this model that we will use. For a more complete introduction see Hopcroft and Ullman (1979) or the chapter on machine models in this handbook by van Emde Boas (1989).

A *deterministic Turing machine* consists of a finite control and a finite collection of tapes each with a head for reading and writing. The finite control is a finite collection of states. A tape is an infinite list of cells each containing a symbol. Initially, all tapes have blanks except for the first, which contains the input string. Once started, the machine goes from state to state, reading the symbols under the heads, writing new ones, and moving the heads. The exact action taken is governed by the current state, the symbols read, and the next move function of the machine. This continues until a designated halt state is entered. The machine indicates its output by the halting condition of the tapes.

In a *nondeterministic Turing machine* the next move function is multivalued. There may be several computations on a given input, and several output values. We say the machine accepts its input if at least one of these outputs signals acceptance. An *alternating Turing machine* is a nondeterministic Turing machine whose states are labeled $\wedge$ and $\vee$. Acceptance of an input is determined by evaluating the associated $\wedge, \vee$ tree of computations in the natural way. A Σ_i (Π_i) Turing machine is an alternating Turing machine which may have at most i runs of $\vee$ and $\wedge$ states and must begin with $\vee$ ($\wedge$) states. See Chandra, Kozen, and Stockmeyer (1981) for more information about alternating Turing machines.

Say a Turing machine M *accepts* language A if M accepts exactly those strings in A. We now define Turing machine computations within a time bound T. Let $T\colon N \rightarrow N$. A deterministic, nondeterministic, or alternating Turing machine M

runs in time $T(n)$ if for every input string w of length n, every computation of M on w halts within $T(n)$ steps. We define the time complexity classes.

$$\begin{aligned}
\mathrm{TIME}(T(n)) &= \{A\text{: some deterministic Tm accepts } A \text{ in time } \mathrm{O}(T(n))\} \\
\mathrm{NTIME}(T(n)) &= \{A\text{: some nondeterministic Tm accepts } A \text{ in time } \mathrm{O}(T(n))\} \\
\mathrm{ATIME}(T(n)) &= \{A\text{: some alternating Tm accepts } A \text{ in time } \mathrm{O}(T(n))\} \\
\Sigma_i\mathrm{TIME}(T(n)) &= \{A\text{: some } \Sigma_i \text{ Tm accepts } A \text{ in time } \mathrm{O}(T(n))\} \\
\Pi_i\mathrm{TIME}(T(n)) &= \{A\text{: some } \Pi_i \text{ Tm accepts } A \text{ in time } \mathrm{O}(T(n))\}
\end{aligned}$$

$$\mathrm{P} = \bigcup_k \mathrm{TIME}(n^k)$$

$$\mathrm{NP} = \bigcup_k \mathrm{NTIME}(n^k)$$

$$\Sigma_i\mathrm{P} = \bigcup_k \Sigma_i\mathrm{TIME}(n^k)$$

$$\Pi_i\mathrm{P} = \bigcup_k \Pi_i\mathrm{TIME}(n^k)$$

The connection between circuit complexity and Turing machine complexity was first shown by Savage (1972). The bound in the following theorem is due to Pippenger and Fischer (1979).

Theorem 2.1: *If language A is in* $\mathrm{TIME}(T(n))$ *then A has circuit complexity* $O(T(n)\log(T(n)))$.

Proof sketch: To see the main idea of the simulation we first prove the weaker bound $O(T^4(n))$. Let M accept A in time $T(n)$. Convert M to a 1-tape machine. This increases the time to $O(T^2(n))$. View a computation of M on some input as a table where row i is the tape at the ith step. At any point in time we will consider the cell currently scanned by the head to contain a symbol representing both the actual symbol and the state of the machine. Let cell(i, j) be the contents of the ith cell at time j. It is easy to see that cell(i, j) only depends upon its predecessors cell$(i-1, j-1)$, cell$(i, j-1)$, and cell$(i+1, j-1)$. We may encode the possible values of a cell in binary and build a small circuit for each cell which computes its value from its predecessors. Assuming the machine indicates its acceptance in the 1st tape cell upon halting, we designate the appropriate gate from cell$(1, l)$ to be the output where l is the index of the last row. Since the total number of cells is $O((T^2(n))^2) = O(T^4(n))$, the simulating circuit has size $O(T^4(n))$.

To obtain the tighter bound of the theorem, we first modify the original machine so that it is *oblivious,* i.e., the head motions are only dependent upon the length of the input but not the input itself. An $O(T(n) \log T(n))$ time simulation using two tapes accomplishes this. Once the motions of the heads are known the above construction may be repeated, except now it is only necessary to build circuitry for the cells which contain the head because the others do not change symbol. ∎

Thus every language in P has polynomial circuit complexity. The converse of this is false since there are nonrecursive languages with low circuit complexity. To obtain a converse we define the nonuniform extension of P. Alternatively, we may impose a uniformity condition on the circuit families saying that the nth circuit is easy to find as a function of n. But, since lower bounds for nonuniform classes imply lower bounds for the corresponding uniform classes, we only consider the former here.

Definitions: Let $f\colon N \to N$. An $f(n)$-*advice sequence* is a sequence of binary strings $A = (a_1, a_2, \ldots)$, where $|a_n| \leq f(n)$. For a language $B \subseteq \{0, 1, \#\}^*$ let $B@A = \{x \mid x\#a_{|x|} \in B\}$. Let $\mathrm{P}/f(n) = \{B@A\colon\ B \in \mathrm{P}$ and A is an $f(n)$-advice sequence$\}$. Let P/poly$= \bigcup_k \mathrm{P}/n^k$. Apply this notation to other complexity classes, e.g., NP/poly. These classes are sometimes referred to as "nonuniform P" or "nonuniform NP".

Theorem 2.2: *C is in P/poly iff C has polynomial circuit complexity.*

Proof: If C is in P/poly then $C = B@A$ for some B in P and polynomial advice sequence A. By the above theorem B has polynomial circuit complexity. Presetting the advice strings into the appropriate inputs of the circuits for B obtains the polynomial size circuits for C.

For the other direction we encode the polynomial size circuits for C as an advice sequence. ∎

Thus one may hope to show P $\neq$ NP by giving a superpolynomial lower bound on circuit size for an NP problem. Of course, it may be that all NP problems have polynomial circuit complexity even though P $\neq$ NP. The following theorem of Karp, Lipton, and Sipser (1982) shows that if this were the case then there would be other peculiar consequences.

Theorem 2.3: *If NP $\subseteq$ P/poly then the polynomial time hierarchy collapses to $\Sigma_2 P$.*

2.2. Nonexplicit Lower Bounds

Although we are mostly concerned with explicit problems (i.e., those in NP), it is worth understanding what can be said about nonexplicit problems. Muller (1956), based on an argument of Shannon (1949), proved an exponential lower bound on the circuit complexity of a nonexplicit problem. We prove this lower bound below.

Counting arguments can help establish lower bounds for nonexplicit problems. On one hand, there are not too many small circuits; on the other hand, there are very many Boolean functions of n variables. Thus some Boolean function must require circuits of exponential size. Muller obtains the following result.

Theorem 2.4: *Almost every Boolean function of n variables requires circuits of size $\Omega(2^n/n)$.*

Proof: We first show the number of circuits with n variables and size s is bounded above by $(2 \cdot (s + 2n + 2)^2)^s$. Each gate in a circuit is assigned an AND or OR operator that acts on two previous nodes. Each previous node can either be a previous gate (at most s choices), a literal, i.e., a variable or its negation ($2n$ choices), or a constant (2 choices). Thus each gate has at most $2 \cdot (s + 2n + 2)^2$ choices. Compounding these choices for all s gates gives the claimed upper bound.

Notice that for $s = 2^n/(10n)$, the above bound is approximately $2^{2^n/5} \ll 2^{2^n}$. Since there are 2^{2^n} Boolean functions of n variables, almost every Boolean function requires circuits of size larger than $2^n/(10n)$. ∎

The above lower bound is optimal up to a constant factor. Expressed in disjunctive normal form (i.e., as an OR of ANDs of literals), every Boolean function has circuits of size $O(2^n \cdot n)$. By being more careful, Muller showed that every Boolean function has circuits of size $O(2^n/n)$.

Theorem 2.4 shows the existence of a Boolean function with exponential circuit complexity, but says nothing about the explicitness of the function. The theorem can be sharpened to yield a Boolean function computable in exponential space but requiring exponential-size circuits. Let f_n be the lexically-first function of n variables that requires circuits of size $2^n/(10n)$, and let f be the union of these functions over all n. By definition f requires circuits of exponential size. By enumerating in lexical order all functions of n variables, and also enumerating all small circuits, we can compute f in space $O(2^n)$. Unfortunately, this argument fails to provide an example of a problem in NP (or even in exponential time) that provably requires circuits of superpolynomial size.

As we did for circuit complexity above, one can ask about the complexity of most Boolean functions under other complexity measures. For example, the formula complexity (defined in section 5) of almost every Boolean function is $\Theta(2^n/\log n)$. The branching program complexity (defined in section 6) of almost every Boolean function is $\Theta(2^n/n)$.

2.3. Explicit Lower Bounds

Despite the importance of lower bounds on the circuit complexity of explicit problems, the best bounds known are only linear. Blum (1984), improving a bound of Paul (1977), proved a $3n - o(n)$ lower bound. In this subsection, we prove a weaker but easier lower bound of size $2n - O(1)$. These bounds apply to circuits with all binary gates allowed.

The proofs of these lower bounds use the following gate-elimination method. Given a circuit for the function in question, we first argue that some variable (or set of variables) must fan out to several gates. Setting this variable to a constant will eliminate several gates. By repeatedly applying this process, we conclude that the original circuit must have had many gates. As an example, we apply the gate-elimination method to threshold functions. Let $\mathrm{TH}_{k,n}$ be the function that outputs 1 iff at least k of its n variables are 1. We have the following lower bound.

Theorem 2.5: *For $n \geq 2$, the function $\mathrm{TH}_{2,n}$ requires circuits of size at least $2n - 4$.*

Proof: The proof is by induction on n. For $n = 2$ and $n = 3$, the bound is trivial. Otherwise, let C be an optimal circuit for $\mathrm{TH}_{2,n}$, and suppose without loss of generality that the bottom-most gate of C acts on variables x_i and x_j (where $i \neq j$). Notice that under the four possible settings of x_i and x_j, the function $\mathrm{TH}_{2,n}$ has three possible subfunctions (namely $\mathrm{TH}_{0,n-2}$, $\mathrm{TH}_{1,n-2}$, and $\mathrm{TH}_{2,n-2}$). It follows that either x_i or x_j fans out to another gate in C, for otherwise C would have only two inequivalent subcircuits under the settings of x_i and x_j; suppose it is x_i that fans out to another gate. Setting x_i to 0 will eliminate the need for at least two gates from C. The resulting function is $\mathrm{TH}_{2,n-1}$, which by induction requires circuits of size $2(n-1) - 4$. Adding the two eliminated gates to this bound shows that C has at least $2n - 4$ gates, which completes the induction. ∎

3. Bounded Depth Circuits

As we have seen in the previous section, our methods for proving lower bounds on the circuit complexity of explicit Boolean functions are presently very weak. We have only been able to obtain strong lower bounds by imposing sharp limitations on the types of computation performed by the circuit. In this and the next section we will survey the results on two types of limited circuit model, bounded depth circuits and monotone circuits.

The first strong lower bounds for bounded depth circuits were given by Furst, Saxe, and Sipser (1984) and Ajtai (1983) who demonstrated a superpolynomial lower bound for constant depth circuits computing the parity function. Subsequently Yao (1985), by giving a deeper analysis of the method of Furst, Saxe, and Sipser, was able to give a much sharper exponential lower bound. Hastad (1989) further strengthened and simplified this argument, obtaining near optimal bounds. Ajtai used different but related probabilistic combinatorial methods in his proof. We present Hastad's proof here.

3.1. Definitions

In this section we consider circuits whose depth is very much smaller than n, the number of inputs. We allow arbitrary fanin so that the circuit may access the entire input. Equivalently one may allow only bounded fanin and measure alternation depth.

For the following definitions we assume that the circuits are of the special form where all AND and OR gates are organized into alternating levels with edges only between adjacent levels and all negations appear only on the inputs. Any circuit may be converted to one of this form without increasing the depth and by at most squaring the size.

Definitions: A circuit C is called a Σ_i^S circuit if it has a total of at most S gates organized into at most i levels with an OR gate at the output. Say C is a $\Sigma_i^{S,t}$ circuit if there are a total of at most S gates organized into at most $i+1$ levels with an OR gate at the output and gates of fanin at most t at the input level. Call C a *t-open* circuit if it is $\Sigma_1^{S,t}$ for some S, i.e., an OR of ANDs of fanin at most t. Dually define Π_i^S, $\Pi_i^{S,t}$, and *t-closed* by exchanging AND and OR.

A peculiar feature of these definitions is that we define a $\Sigma_i^{S,t}$ circuit to be one of depth $i+1$. We defend this terminology by noting the analogy of unbounded fanin gates to unbounded quantifiers and bounded fanin gates to bounded quantifiers. Generally the subscript in the Σ_i symbol refers to the number of unbounded quantifiers. A further justification is that the statements of a number of the coming theorems are rendered more elegantly by adopting this convention.

In the remainder of this section the statements made regarding Σ circuits have a natural dual form for Π circuits. To avoid continual repetition we omit this dual form where obvious.

We will speak of t-open, t-closed, Σ_i^S, and $\Sigma_i^{S,t}$ functions and subsets of $\{0,1\}^n$ as those which are defined by the respective type of circuits.

A *family* of circuits is a sequence $(C_1, C_2, \ldots)$ where C_n takes n input variables. A family may be used to define a language (subset of $\{0,1\}^*$) or a function from $\{0,1\}^*$ to $\{0,1\}$. We may use the notation $t(n)$-open, $t(n)$-closed, $\Sigma_i^{f(n)}$, and $\Sigma_i^{f(n),t(n)}$ to describe the complexities of families and their associated languages and functions. Occasionally we will use $\Sigma_i^{f(n)}$, or $\Sigma_i^{f(n),t(n)}$ to mean the collection of such families, languages, or functions. A *uniform family* is one where the description of C_n may be easily computed from n. Since the lower bounds we present later in the chapter apply even in the stronger nonuniform case we will not occupy ourselves further with notions of uniformity.

Let Σ_i^{poly} and $\Sigma_i^{\mathrm{poly,const}}$ mean $\bigcup_k \Sigma_i^{n^k}$ and $\bigcup_k \Sigma_i^{n^k,k}$ respectively.

Definition: Let $\mathrm{AC}^0 = \bigcup_i \Sigma_i^{\mathrm{poly}}$. The classes NC^i and AC^i for $i \geq 0$ were proposed by Pippenger (1979) and Cook (1985) to be those functions computable by a uniform family of polynomial size, $\mathrm{O}(\log^i n)$ depth circuits with constant and unbounded fanin respectively. For the remainder of this chapter we ignore the uniformity condition.

Thus AC^0 is the class of polynomial size, constant depth, unbounded fanin circuits. This is equivalent to the above definition. NC^0 is the class of functions depending upon a number of input variables which is a constant, independent of n. As an aside to our main story, the following theorem, discovered independently by a number of people, is an interesting exercise:

Theorem 3.1: $\mathrm{NC}^0 = \Sigma_1^{\mathrm{poly,const}} \cap \Pi_1^{\mathrm{poly,const}}$.

In fact, one may prove the following stronger separation property.

Theorem 3.2: *If A and B are disjoint $\Sigma_1^{\mathrm{poly,const}}$ languages then there is an* NC^0 *language C separating them, i.e., $A \subseteq C \subseteq \overline{B}$.*

This theorem is false for disjoint $\Pi_1^{\mathrm{poly,const}}$ languages.

3.2. Restrictions

Furst, Saxe, and Sipser introduced the method of probabilistic restrictions as a way of proving lower bounds on the size of bounded depth circuits. Here a randomly selected subset of the variables is preset so that some of the gates in the circuit become determined, resulting in a simpler circuit on fewer variables.

Definition: Let $X = \{x_1, \ldots, x_n\}$ be the input variables to a circuit C computing a function f. A *restriction* ρ is a mapping from X to $\{0, 1, *\}$.

We interpret ρ as presetting the variables assigned 0 or 1 and leaving variable those assigned star. Under ρ we may simplify C by eliminating gates whose values become determined. Call this the *induced circuit* $C|_\rho$ computing the *induced function* $f|_\rho$.

In the probabilistic arguments to follow we will be selecting restrictions from certain probability distributions. Fix $0 < p < 1$. Let $\mathcal{R}_p$ be the probability distribution on restrictions over X where each $x_i \in X$ is independently assigned a value in $\{0, 1, *\}$ so that $\Pr[\rho(x_i) = *] = p$ and $\Pr[\rho(x_i) = 0] = \Pr[\rho(x_i) = 1] = \frac{(1-p)}{2}$.

3.3. Hastad Switching Lemma

The following lemma of Hastad (1989) states that a t-closed function is very likely to become s-open under a restriction chosen at random from a suitable probability distribution.

Lemma 3.3: *Let f be a t-closed function and ρ a random restriction from $\mathcal{R}_p$. Then*

$$\Pr\left[f|_\rho \textit{ is not } s\textit{-open}\right] \leq \alpha^s$$

where $\alpha = \gamma pt$ and $\gamma = 2/\ln\phi \approx 4.16$ for $\phi = (1+\sqrt{5})/2$, the golden ratio.

Proof: We introduce some notation. A *minterm* of f is a minimal assignment to some subset of the variables which determines f to be 1. Say that $min(f)$ is the size of the largest minterm of f. By showing that $min(f|_\rho) \leq s$ we show that $f|_\rho$ is s-open because a function may be written as an OR of its minterms. Below we will prove that the slightly stronger inequality $min(f|_\rho) < s$ occurs with high probability.

To better understand the proof of this lemma it is helpful to first consider a special case. Say f is given by a t-closed circuit C where we assume that all of the ORs of C are on disjoint sets of variables. Then the following straightforward induction on l, the number of ORs in C, gives the lemma. The intuition behind this induction is easy. Consider the ORs one by one. If some variable in an OR gets a 1 under ρ then the OR is determined to be 1 and it contributes nothing to any minterm. Only a starred variable in an OR without 1's may be part of a minterm. But such an OR is more likely to get all 0's. If this ever happens then the function is determined to be 0 and has no minterms at all. Thus it is very unlikely that there will be large minterms. The details follow.
Basis: ($l = 0$). Obvious since f is the constant function.
Induction: (Assume for $l-1$ and prove for l). Let C_1 be the first OR in C. By renaming variables we may assume that all of the variables in C_1 occur without negation. Let D be C without C_1. Then

$$\Pr\left[min(C|_\rho) \geq s\right] = \Pr\left[C_1|_\rho \equiv 1\right] \cdot (\mathrm{A}) + \Pr\left[C_1|_\rho \not\equiv 1\right] \cdot (\mathrm{B})$$

where

$$(\mathrm{A}) = \Pr\left[min(C|_\rho) \geq s \;\middle|\; C_1|_\rho \equiv 1\right]$$

and

$$(\mathrm{B}) = \Pr\left[min(C|_\rho) \geq s \;\middle|\; C_1|_\rho \not\equiv 1\right].$$

Since the right hand side is a weighted average of (A) and (B) it is sufficient to show that both (A) and (B) are at most α^s to conclude that the left hand side is at most α^s. Here and in all expressions throughout the proof of lemma 3.3 and 3.3′ we adopt the convention that $\Pr[A|B] = 0$ if $\Pr[B] = 0$.

Bounding (A): If $C_1|_\rho \equiv 1$ then C_1 will not contribute to the minterm and so $min(C|_\rho) = min(D|_\rho)$. Since we are assuming that the ORs are disjoint, the conditioning is irrelevant and so $(\mathrm{A}) = \Pr[min(D|_\rho) \geq s]$. Now the bound follows from the induction hypothesis.

Bounding (B): We first divide up the probability, summing it according to the number of stars in C_1. Since the number of stars in C_1 is an upper bound on its contribution to any minterm we have $(\mathrm{B}) \leq \sum_{k>0}^{t} (\mathrm{C}) \cdot (\mathrm{D})$ where

$$(\mathrm{C}) = \Pr\left[C_1|_\rho \text{ gets } k \text{ stars} \;\middle|\; C_1|_\rho \not\equiv 1\right].$$

and

$$(\mathrm{D}) = \Pr\left[min(D|_\rho) \geq s-k \;\middle|\; C_1|_\rho \text{ gets } k \text{ stars and } C_1|_\rho \not\equiv 1\right].$$

Actually 1 is an upper bound on the contribution of C_1 so we can say something a bit stronger here. We chose to estimate it this way to preserve the similarity with the following lemma 3.3′. It is important to note that the term $k = 0$ is excluded from the above sum since it implies that $C_1|_\rho$ and hence $C|_\rho$ are equivalent to 0.

To bound (C) observe that the condition $C_1|_\rho \not\equiv 1$ is equivalent to saying that all variables in C_1 receive 0 or star in ρ. The conditional probability that a variable receives a star is

$$\frac{p}{1-(1-p)/2} = \frac{p}{(1+p)/2} = \frac{2p}{1+p} \leq 2p.$$

So $(\mathrm{C}) \leq \binom{t}{k}(2p)^k$.

To bound (D), again observe that the conditioning has no effect under the disjointness assumption, so by induction (D) $\leq \alpha^{s-k}$. Therefore,

$$
\begin{aligned}
\text{(B)} &\leq \sum_{k>0}^{t} \binom{t}{k} (2p)^k \alpha^{s-k} = \alpha^s \sum_{k>0}^{t} \binom{t}{k} (2p/\alpha)^k \\
&= \alpha^s((1+2p/\alpha)^t - 1) = \alpha^s((1+2p/(\gamma pt))^t - 1) = \alpha^s((1+2/\gamma t)^t - 1) \\
&\leq \alpha^s(e^{2/\gamma} - 1) \\
&< \alpha^s.
\end{aligned}
$$

To eliminate the disjointness assumption we may no longer ignore the conditioning which occurs in (A) and (B). To handle this we will prove a version of the lemma with a stronger induction hypothesis. The intuition behind this is that at every stage in the induction the only information about the probability distribution that is known is that it sets certain variables to star, others to 0 or 1, and that some ORs are determined to be 1. Conditioning on some variable being set to star does not hurt since these variables were already counted. The other conditioning does not hurt since it can only make it less likely that a variable is starred and contributes to a big minterm. In the following lemma, F is an arbitrary function defined on $X = \{x_1, \ldots, x_n\}$ the variables of C, as well as any other variables.

Stronger Lemma 3.3′: *Let f be a t-closed function, F an arbitrary function, and ρ a random restriction from $\mathcal{R}_p$. Then*

$$\Pr\left[f|_\rho \textit{ not } s\textit{-open } \;\middle|\; F|_\rho \equiv 1\right] \leq \alpha^s$$

where $\alpha = \gamma pt$ and $\gamma = 2/\ln\phi \approx 4.16$ for $\phi = (1+\sqrt{5})/2$, the golden ratio.

Proof: Let C be a t-closed circuit for f. We modify the first equation in the previous argument to include the new condition $F|_\rho \equiv 1$ in all probability statements. As before, we assume that the first OR, C_1, has only positive occurrences of variables. Again we show that the modified (A) and (B) are at most α^s.

$$
\begin{aligned}
&\Pr\left[min(C|_\rho) \geq s \;\middle|\; F|_\rho \equiv 1\right] = \\
&\Pr\left[C_1|_\rho \equiv 1 \;\middle|\; F|_\rho \equiv 1\right] \cdot (A) + \Pr\left[C_1|_\rho \not\equiv 1 \;\middle|\; F|_\rho \equiv 1\right] \cdot (B)
\end{aligned}
$$

where

$$(A) = \Pr\left[min(C|_\rho) \geq s \;\middle|\; C_1|_\rho \equiv 1 \text{ and } F|_\rho \equiv 1\right]$$

and

$$(B) = \Pr\left[min(C|_\rho) \geq s \;\middle|\; C_1|_\rho \not\equiv 1 \text{ and } F|_\rho \equiv 1\right].$$

Bounding (A): $\Pr\left[min(C|_\rho) \geq s \;\middle|\; C_1|_\rho \equiv 1 \text{ and } F|_\rho \equiv 1\right]$ is clearly equivalent to $\Pr\left[min(D|_\rho) \geq s \;\middle|\; (C_1 \wedge F)|_\rho \equiv 1\right]$. This is bounded by α^s using the induction hypothesis.

Bounding (B): Recalling (B) we have

$$(\mathrm{B}) = \Pr\left[min(C|_\rho) \geq s \;\middle|\; C_1|_\rho \not\equiv 1 \text{ and } F|_\rho \equiv 1\right].$$

In the special case of the lemma we estimated (B) by breaking up the probability, dividing the minterms according to the number of variables in C_1 that they contain. Now we must use a finer scalpel and divide the minterms according to precisely which variables in C_1 they contain.

Let T be the collection of variables in C_1 and let $Y \subseteq T$. Let $min^Y(C)$ be the size of the largest minterm of C which sets all of the variables of Y and no others in T. Let $\sigma \in \{0,1\}^Y$ be an assignment of Y to 0 and 1. Let $min^{Y \leftarrow \sigma}(C)$ be the size of the largest minterm of C which sets all of the variables in Y to σ and sets no other variables in T. Both $min^Y(C)$ and $min^{Y \leftarrow \sigma}(C)$ are defined to be 0 if such minterms do not exist. Let $\rho(Y) = *$ denote the event that ρ assigns star to all variables in Y.

Then we can write

$$\begin{aligned}
(\mathrm{B}) &= \Pr\left[min(C|_\rho) \geq s \;\middle|\; C_1|_\rho \not\equiv 1 \text{ and } F|_\rho \equiv 1\right] \\
&\leq \sum_{Y \subseteq T} \Pr\left[min^Y(C|_\rho) \geq s \;\middle|\; C_1|_\rho \not\equiv 1 \text{ and } F|_\rho \equiv 1\right] \\
&= \sum_{Y \subseteq T} \begin{pmatrix} \Pr\left[\rho(Y) = * \;\middle|\; C_1|_\rho \not\equiv 1 \text{ and } F|_\rho \equiv 1\right] \\ \cdot \Pr\left[min^Y(C|_\rho) \geq s \;\middle|\; \rho(Y) = * \text{ and } C_1|_\rho \not\equiv 1 \text{ and } F|_\rho \equiv 1\right] \end{pmatrix}.
\end{aligned}$$

We estimate the first term in this sum using an elementary fact from probability theory.

Lemma 3.4: *For arbitrary events A and B*

$$\Pr[A \mid B] \leq \Pr[A] \iff \Pr[B \mid A] \leq \Pr[B].$$

This is easily verified using the definition of conditional probability.

Lemma 3.5: $\Pr\left[\rho(Y) = * \mid C_1|_\rho \not\equiv 1 \text{ and } F|_\rho \equiv 1\right] \leq (2p)^{|Y|}$.

Proof: Without the condition $F|_\rho \equiv 1$ that lemma would follow immediately. We have already done essentially that calculation in the proof of lemma 3.3. The present lemma also holds in the presence of the extra condition because the condition can only decrease the probability and so works in our favor. Intuitively, this is true because knowing that a restriction forces some function to 1 can not make it more likely that any given variable is assigned star. To prove this formally we use the previous lemma. Let A be the event $\rho(Y) = *$ and B the event $F|_\rho \equiv 1$. To show

$$\Pr\left[\rho(Y) = * \mid F|_\rho \equiv 1 \text{ and } C_1|_\rho \not\equiv 1\right] \leq \Pr\left[\rho(Y) = * \mid C_1|_\rho \not\equiv 1\right].$$

it is enough to show that

$$\Pr\left[F|_\rho \equiv 1 \mid \rho(Y) = * \text{ and } C_1|_\rho \not\equiv 1\right] \leq \Pr\left[F|_\rho \equiv 1 \mid C_1|_\rho \not\equiv 1\right].$$

This last inequality holds because the condition $C_1|_\rho \not\equiv 1$ means that ρ assigns the variables in T either 0 or star and the condition $\rho(Y) = *$ means that the variables in the subset Y of T must all be star. Any restriction ρ satisfying both conditions corresponds to a set of $2^{|Y|}$ restrictions satisfying only the condition $C_1|_\rho \not\equiv 1$. If $F|_\rho \equiv 1$ then $F_{\rho'} \equiv 1$ for every ρ' in that set, since any restriction forcing F to be 1 still does even if some of the starred variables are assigned 0.

We now estimate the second term of the sum in (B). That is

$$(\mathrm{D}) = \Pr\left[min^Y(C|_\rho) \geq s \mid \rho(Y) = * \text{ and } C_1|_\rho \not\equiv 1 \text{ and } F|_\rho \equiv 1\right].$$

First we further divide it according to how the minterm sets Y.

$$(\mathrm{D}) \leq \sum_{\substack{\sigma \in \{0,1\}^Y \\ \sigma \neq 0^Y}} \Pr\left[min^{Y \leftarrow \sigma}(C|_\rho) \geq s \mid \rho(Y) = * \text{ and } C_1|_\rho \not\equiv 1 \text{ and } F|_\rho \equiv 1\right].$$

The case $\sigma = 0^Y$ is excluded because a minterm must set some variable in Y to 1.

Estimating (D) is now greatly simplified. Break up the restriction ρ into ρ_1 which assigns values to T and ρ_2 which assigns values to the remaining variables. Then

$$(\mathrm{D}) \le \sum_{\substack{\sigma \in \{0,1\}^Y \\ \sigma \neq 0^Y}} \max_{\rho_1} \Pr_{\rho_2} \left[min^{Y \leftarrow \sigma}(C|_\rho) \ge s \;\middle|\; \rho(Y) = * \text{ and } C_1|_\rho \not\equiv 1 \text{ and } F|_\rho \equiv 1 \right].$$

The maximum is taken over all ρ_1 assigning 0's and stars to the variables in T and only stars to the variables in Y. Having now fixed how ρ_1 sets the variables in C_1 we would like to estimate the above probability over ρ_2 using the induction hypothesis. To do this we must set all of the variables that ρ_1 assigns star to be 0 or 1. For the variables in Y we take the assignments given by σ. For the variables in $T - Y$ we take the worst case setting τ of these variable to 0 and 1.

Fix σ and ρ_1. Let W be the variables in $T - Y$ that are assigned star by ρ_1. Recall that D is C without C_1. We may obtain an upper bound on the above probability by writing,

$$\max_{\tau \in \{0,1\}^W} \Pr_{\rho_2} \left[min((D|_{\sigma\tau\rho_1})|_{\rho_2}) \ge s - |Y| \;\middle|\; \rho(Y) = * \text{ and } C_1|_\rho \not\equiv 1 \text{ and } F|_\rho \equiv 1 \right].$$

Since the probability is only over ρ_2 and the first two conditions do not depend upon ρ_2 they may be dropped. Fixing the maximizing τ this is

$$\Pr_{\rho_2} \left[min((D|_{\sigma\tau\rho_1})|_{\rho_2}) \ge s - |Y| \;\middle|\; (F|_{\rho_1})|_{\rho_2} \equiv 1 \right].$$

By induction this is at most $\alpha^{s-|Y|}$.

Pulling this together we get

$$(\mathrm{D}) \le \sum_{\substack{\sigma \in \{0,1\}^Y \\ \sigma \neq 0^Y}} \max_{\rho_1} \alpha^{s-|Y|} = \sum_{\substack{\sigma \in \{0,1\}^Y \\ \sigma \neq 0^Y}} \alpha^{s-|Y|}$$

$$= (2^{|Y|} - 1)\alpha^{s-|Y|}$$

and thus

$$
\begin{aligned}
(\mathrm{B}) &\le \sum_{Y \subseteq T} (2p)^{|Y|}(2^{|Y|}-1)\alpha^{s-|Y|} \\
&= \alpha^s \sum_{k \ge 0} \binom{|T|}{k} (2p)^k (2^k - 1)/\alpha^k \\
&\le \alpha^s \left(\sum_{k \ge 0} \binom{t}{k} (4p/\alpha)^k - \sum_{k \ge 0} \binom{t}{k} (2p/\alpha)^k \right) \\
&= \alpha^s((1+4/\gamma t)^t - (1+2/\gamma t)^t) \\
&\le \alpha^s(e^{4/\gamma} - e^{2/\gamma}) \\
&\le \alpha^s \quad (\text{recalling } \gamma = 2/\ln\phi). \quad \blacksquare
\end{aligned}
$$

3.4. Lower Bound for the Parity Function

Now we use the Hastad switching lemma to derive a lower bound for small depth circuits computing the parity function.

Theorem 3.6: *For all $p, d, (0 \le k \le d-1)$, if f is $\Sigma_d^{S,t}$ then for a random ρ from $\mathcal{R}_{p^k}$*

$$\Pr\left[f|_\rho \text{ is not } \Sigma_{d-k}^{S,t}\right] < S(\gamma p t)^t$$

where $\gamma = 2/\ln\phi \approx 4.16$.

Proof: Consider the random restriction ρ as being composed from k restrictions $\rho = \rho_1\rho_2\cdots\rho_k$ each drawn from $\mathcal{R}_p$. Obtain the sequence of functions $f = f_1, f_2, \ldots, f_{k+1}$ where $f_{i+1} = f_i|_{\rho_i}$. At each step of this sequence there is a collection of t-open (or t-closed if $d-i$ is odd) bottom level subcircuits in the circuit for f_i which may become t-closed (or t-open) under ρ_i and then merge with the gates above them. If this successfully occurs for each subcircuit in every f_i then f_{k+1} is $\Sigma_{d-k}^{S,t}$. The probability that it fails at any particular subcircuit is at most $(\gamma p t)^t$ by the Hastad switching lemma. Hence the probability that it fails at any of the at most S subcircuits encountered is bounded above by $S(\gamma p t)^t$. $\blacksquare$

The following corollary is independently interesting as a type of Ramsey theorem (see Graham, Rothschild, and Spencer (1980)).

Corollary 3.7: *If f is $\Sigma_d^{S,t}$ where $t \geq \log S$ then there is a restriction ρ assigning at least $n/3(10t)^{d-1} - t$ stars such that $f|_\rho$ is a constant function.*

Proof: By the theorem above, if $p = 1/10t$ and ρ is drawn from $\mathcal{R}_{p^{d-1}}$ then the probability that $f|_\rho$ is not $\Sigma_1^{S,t}$ is at most $S(\gamma pt)^t = S\beta^t \leq (2\beta)^t$ where $\beta = \gamma/10 < .42$. Hence $\Pr[f|_\rho$ is not $\Sigma_1^{S,t}] < .84$. Furthermore, ρ is expected to have np^{d-1} stars. An easy calculation shows that $\Pr[\rho$ has fewer than $np^{d-1}/3$ stars$] < .15$. Since the sum of these probabilities is less than 1, there is a restriction ρ for which neither event occurs. Finally, since any nonconstant $\Sigma_1^{S,t}$ function may be forced to 1 by setting at most t inputs we may extend ρ by including these t additional settings and guarantee that $f|_\rho$ is constant. ∎

Using the preceding corollary we now can obtain the desired lower bounds for the parity function, $\text{PARITY}(x_1, \ldots, x_n) = \sum x_i \bmod 2$.

Theorem 3.8: *For all $n, d > 0$, PARITY is not $\Sigma_d^{S,\log S}$ where $S < 2^{\frac{1}{10}n^{1/d}}$.*

Proof: If PARITY were $\Sigma_d^{S,\log S}$ for $S < 2^{\frac{1}{10}n^{1/d}}$ then by the above corollary there would be a restriction ρ assigning at least one star such that $\text{PARITY}|_\rho$ is constant. This contradiction proves the theorem. ∎

Corollary 3.9: PARITY$\notin$ AC^0.

Corollary 3.10: Polynomial size parity circuits must have depth at least $\frac{\log n}{c+\log\log n}$ for some constant c.

The bound in the above theorem cannot be significantly improved as it is quite close to the easily obtained upper bound.

Theorem 3.11: *For all n and d, PARITY is $\Sigma_d^{S,\log S}$ where $S = n2^{n^{1/d}}$.*

We may also derive lower bounds for other simple Boolean functions as a corollary to the lower bound for the parity function, using the notion of AC^0-reducibility. This notion was proposed by Furst, Saxe, and Sipser who gave a few examples of reductions and further investigated by Chandra, Stockmeyer, and Vishkin (1984) who gave additional examples.

Definition: Let $f, g\colon \{0,1\}^* \to \{0,1\}^*$. Say f is AC^0-*reducible* to g if there is a family of constant depth, polynomial size circuits computing f using AND, OR, and NOT gates, and polynomial fanin gates computing g.

As an example it is easy to see that PARITY is AC^0-reducible to the majority function MAJORITY$(x_1, \ldots, x_n)$ which is 1 iff at least half of the x_i are 1.

Corollary 3.12: MAJORITY$\notin$ AC^0.

3.5. Depth Hierarchy

Thus far, restrictions have been used to establish the limitations of small, shallow circuits. In a similar way they may be used to show that the power of small circuits forms a hierarchy according to depth. Sipser (1983) showed that $\Sigma_{d+1}^{\mathrm{poly}} \neq \Sigma_d^{\mathrm{poly}}$ for every d. It follows as an easy corollary that $\Sigma_d^{\mathrm{poly,const}} \neq \Pi_d^{\mathrm{poly,const}}$ for every d. Exponential lower bounds for the depth d to $d-1$ conversion were claimed without proof by Yao (1985) and proved by Hastad (1989) as a consequence of a variant of his switching lemma.

Let

$$f_d^m = \bigwedge_{i_1 \leq m_1} \bigvee_{i_2 \leq m_2} \bigwedge_{i_3 \leq m_3} \cdots \bigodot_{i_d \leq m_d} [x_{i_1 \ldots i_d} = 1],$$

where $\bigodot = \bigwedge$ or $\bigvee$ depending on the parity of d. The variables $x_1, \ldots, x_n$ appear as $x_{i_1,\ldots,i_d}$ for $i_j \leq m_j$ where $m_1 = \sqrt{m/\log m}$, $m_2 = m_3 = \cdots = m_{d-1} = m$, $m_d = \sqrt{dm \log m/2}$, and $m = (n\sqrt{2/d})^{1/(d-1)}$. Since $(\prod_{i \leq d} m_i) = n$ we see that f_d^m is Π_d^n.

Theorem 3.13: *For every $m, d > 1$ the function f_d^m is not $\Sigma_d^{S,\log S}$ for $S < 2^{\frac{1}{20} n^{1/2d} (\log n)^{-1/2}}$.*

3.6. Monotone Bounded Depth Circuits

Monotone circuits have AND and OR but not NOT gates. Boppana (1986) and Klawe, Paul, Pippenger, and Yannakakis (1984) were the first to obtain strong lower bounds on the size of constant depth monotone circuits. Boppana gave an exponential lower bound for the majority function and Klawe, Paul, Pippenger and Yannakakis showed that the depth d to $d-1$ conversion is exponential. These results were superseded by the previously described results of Yao and Hastad. Ajtai and Gurevich (1987) and Okol'nishnikova (1982) consider the relative power of monotone and general circuits for computing monotone functions. They give the following family of AC^0 functions not computable by monotone, constant depth, polynomial size circuits.

For each m we define the function $f_m(x_1, \ldots, x_n)$ where $n = m \log m$. Index the variables $x_1, \ldots, x_n$ as x_{ij} for $i \leq \log m$ and $j \leq m$. For each row i let s_i be the index of the last 1 such that there are no 0's preceding it in that row and set $s_i = 0$ if the row has all 0's. Then f_m is 1 if $\sum s_i \geq \frac{n}{2}$.

3.7. Probabilistic Bounded Depth Circuits

A probabilistic circuit is one which has in addition to its standard inputs some specially designated inputs called random inputs. When these random inputs are chosen from a uniform distribution the output of the circuit is a random variable. We say a probabilistic circuit (a, b)-*accepts* a language if it outputs 1 with probability at most a for strings not in the language and outputs 1 with probability at least b for strings in the language. The circuit accepts with ϵ-*error* if it $(\epsilon, 1-\epsilon)$-accepts and it accepts with ϵ-*advantage* if it $(\frac{1}{2}, \frac{1}{2}+\epsilon)$-accepts.

A simple nonconstructive argument due to Adleman (1978) and Bennett and Gill (1981) shows that any language accepted by polynomial size probabilistic circuits with $(1/n^k)$-advantage for any fixed k may also be accepted by polynomial size deterministic circuits. Ajtai and Ben-Or (1984) consider the analogous question for constant depth circuits and give a similar answer provided the advantage is at least $1/\log^k n$.

Theorem 3.14: *Every probabilistic circuit C of size s and depth d that accepts a language with $(\log^{-k} n)$-advantage (for fixed k) has an equivalent deterministic circuit of size $poly(n) \cdot s$ and depth $d + 2k + 2$.*

Proof: Our plan is to first amplify the advantage until the error is exponentially small. Then there exists a setting of the random inputs which always gives the correct answer. In order to amplify the advantage we define two mappings on circuits. Let $and^l(C) = \bigwedge_{1 \le i \le l} C_i$ and $or^l(C) = \bigvee_{1 \le i \le l} C_i$ where each C_i is an independent copy of C sharing the same standard inputs but with its own set of random inputs. It is easy to see that if C is (a,b)-accepting then $and^l(C)$ is (a^l, b^l)-accepting and $or^l(C)$ is $\big(1-(1-a)^l, 1-(1-b)^l\big)$-accepting. It follows that for $k > 1$ if C accepts with $(\log^{-k} n)$-advantage then $or^{l_2}(and^{l_1}(C))$ accepts with $(\log^{-(k-1)} n)$-advantage for $l_1 = 2\log n$ and $l_2 = n^2 \ln 2$. If C accepts with $(\log^{-1} n)$-advantage then $or^{l_4}(and^{l_3}(or^{l_2}(and^{l_1}(C))))$ has error at most $e^{-n} < 2^{-n}$ for $l_1 = 2\log n$, $l_2 = 2n^2 \log n$, $l_3 = n^3$, and $l_4 = n$. ∎

3.8. Razborov-Smolensky Lower Bound for Circuits with MOD_p Gates

A natural way to extend the results of the last few subsections is to obtain lower bounds for more powerful circuit models. Razborov (1987) took an important step in this direction when he proved an exponential lower bound for computing the majority function with small depth circuits having AND, OR, and PARITY gates. His method is similar to that of his earlier paper on monotone circuits for the clique function, described in section 4. Subsequently, Smolensky (1987) simplified and strengthened this theorem showing that for any p and q powers of distinct primes, the MOD_p function cannot be computed with AND, OR, NOT, MOD_q circuits of polynomial size and constant depth.

The majority function is the Boolean function $\mathrm{MAJORITY}(x_1, \ldots, x_n) = 1$ iff $\sum x_i \ge n/2$. The MOD_p function is the Boolean function which is 1 iff $\sum x_i \not\equiv 0 (\bmod p)$. We will prove the special case of the Razborov-Smolensky theorem where $p = 2$ and $q = 3$.

Theorem 3.15: *The* MOD_2 *function cannot be computed with a size* $\frac{1}{50} 2^{\frac{1}{2} n^{1/2d}}$*, depth* d *circuit of AND, OR, NOT, and* MOD_3 *gates for sufficiently large* n*.*

Proof: Let C be a depth d circuit computing MOD_2 with these types of gates. Think of the gates of C as operating on functions rather than merely on Boolean values. Our plan is to slightly adjust the results of each of the AND and OR operators in such a way that 1) each adjustment alters the output of C on few input settings, while 2) the end result is a function which differs from MOD_2 in many input settings. Hence many adjustments must occur. This gives a lower bound on the circuit size.

More precisely, we will assign to each subcircuit of C a new function called a *b-approximator*. A b-approximator is a polynomial on the input variables $x_1, \ldots, x_n$ of degree at most b over GF(3), the three element field $\{-1, 0, 1\}$, where on inputs from $\{0, 1\}$ it takes values in $\{0, 1\}$. A subcircuit of height h is assigned an $n^{h/2d}$-approximator. The assignments are done inductively, first to the inputs, then working up to the output. Each assignment introduces some *error* which is the number of output deviations between it and the result of applying the true operator at that gate to the approximators of the subcircuits feeding into it, looking only at inputs drawn from $\{0, 1\}^n$.

Let D be a subcircuit of C all of whose proper subcircuits have been assigned b-approximators. If the top gate of D is a NOT gate and the unique subcircuit feeding into it has approximator f, then we assign the b-approximator $1 - f$ to D. Since f and $1 - f$ are correct on the same input settings, this approximator introduces no new error. If the top gate of D is a MOD_3 gate and its inputs have b-approximators $f_1, \ldots, f_k$ then we assign the $2b$-approximator $\left(\sum f_i\right)^2$ to D. Since $0^2 = 0$ and $-1^2 = 1^2 = 1$ this introduces no new error.

Before considering the case where the top gate is an AND or an OR let us examine how these operators affect degree. If $f_1, \ldots, f_k$ are Boolean functions that are polynomials of degree at most b then $\mathrm{AND}(f_1, \ldots, f_k)$ is represented by the kb degree polynomial $f_1 \cdot f_2 \cdot \ldots \cdot f_k$. Since f_i and $\neg f_i = 1 - f_i$ have the same degree, $\mathrm{OR}(f_1, \ldots, f_k)$ also has degree at most kb. We cannot in general use this as an approximator because k may be polynomial in n and this bound is too large to use directly. To find a low degree approximation for the results of these operators we will allow some error to be introduced. Let us first fix a parameter l which determines the tradeoff between error and degree.

If the top gate of D is an OR and its subcircuits have approximators $f_1, \ldots, f_k$ then we find a $2lb$-approximator for D as follows. Select $F_1, \ldots, F_l$ subsets of $\{f_i\}$, let $g_i = (\sum_{f \in F_i} f)^2$, and let $a = \mathrm{OR}(g_1, \ldots, g_l)$. For an appropriate choice of F's, the polynomial a is the desired $2lb$-approximator. To bound the error observe that if for a particular input setting, $\mathrm{OR}(f_1, \ldots, f_k) = 0$ then all f_i and hence all g_i are 0 and so a is 0. To analyze the case where the OR is 1 take the F's to be independently selected random subsets of $\{f_i\}$. Since at least one f_i is 1 each g_i independently has at least a $\frac{1}{2}$ chance of being 1, so $\Pr[a = 0] \leq 2^{-l}$. By an averaging argument there must be some collection of F's so that the number of input settings on which $\mathrm{OR}(f_1, \ldots, f_k) = 1$ and $a = 0$ is at most 2^{n-l}. A dual argument applies if the top gate of D is an AND gate using the identity $\neg f = 1 - f$. Thus, in either case, the approximator introduces at most 2^{n-l} error. This gives the following lemma.

Lemma 3.16: *Every circuit C of depth d has a $(2l)^d$-approximator differing with C on at most $size(C) \cdot 2^{n-l}$ input settings.*

Proof: The inputs to C are assigned the corresponding 1-approximators. Each level increases the degree of the approximators by a factor of at most $2l$. Each assignment of an approximator contributes at most 2^{n-l} error. ∎

Fixing $l = n^{1/2d}/2$ we obtain a $\sqrt{n}$-approximator for C.

Lemma 3.17: *Any $\sqrt{n}$-approximator a differs from the* MOD_2 *function on at least $\frac{1}{50}2^n$ input settings for sufficiently large n.*

Proof: Let U be the 2^n possible input settings and $G \subseteq U$ be the settings on which a agrees with MOD_2. For each input variable x_i let $y_i = x_i + 1$. Consider MOD_2 as a function from $\{-1, 1\}^n \to \{-1, 1\}$ under this change of variables. We may write $\mathrm{MOD}_2(y_1, \ldots, y_n)$ as the nth degree polynomial $\prod y_i$. Since this change of variables does not alter the degrees of polynomials, a is a polynomial of degree at most $\sqrt{n}$ which agrees with $\prod y_i$ on G. Let $\mathcal{F}_G$ be the collection of $f: G \to \{-1, 0, 1\}$. Since $|\mathcal{F}_G| = 3^{|G|}$ we may bound the size of G by showing that $|\mathcal{F}_G|$ is small. We do this by assigning each $f \in \mathcal{F}_G$ a different polynomial of degree at most $\frac{n}{2} + \sqrt{n}$ and then estimating the number of such polynomials.

Let $f \in \mathcal{F}_G$. Extend it in an arbitrary way to an $\bar{f}: U \to \{-1, 0, 1\}$. Let p be a polynomial in the y variables representing $\bar{f}$. Let $cy_{i_1} \cdots y_{i_l}$ be a term of p where $c \in \{-1, 1\}$. Notice that it is multilinear, i.e. without powers higher than 1, since $y^2 = 1$ for $y \in \{-1, 1\}$. Let $Y = \{y_1, \ldots, y_n\}$ and $T \subseteq Y$ be the set of y_{i_j} appearing in this term and $\overline{T} = Y - T$. Then we may rewrite this term $c \prod T$ as $c \prod Y \prod \overline{T}$ again using $y^2 = 1$. Within G this is equivalent to the polynomial $c \cdot a \cdot \prod \overline{T}$ of degree $\sqrt{n} + (n - |T|)$. If we rewrite in this way all terms in p of degree greater than $\frac{n}{2}$ we obtain a polynomial that agrees with p in G and hence with f. It has degree at most $(n - \frac{n}{2}) + \sqrt{n} = \frac{n}{2} + \sqrt{n}$.

The number of multilinear monomials of degree at most $\frac{n}{2} + \sqrt{n}$ is $\sum_{i=0}^{\frac{n}{2}+\sqrt{n}} \binom{n}{i}$ which for large n is approximately $.9772 \cdot 2^n < \frac{49}{50}2^n$. The number of polynomials with monomials of this kind is thus less than $3^{\frac{49}{50}2^n}$, so this is an upper bound on the size of $\mathcal{F}_G$. Hence $|G| = \log_3 |\mathcal{F}_G| \leq \frac{49}{50}2^n$. Therefore the number of input settings where a differs from MOD_2 is at least $2^n - \frac{49}{50}2^n = \frac{1}{50}2^n$. ∎

Now we may conclude the proof of the theorem. We know that $size(C)$ is at least the total error introduced divided by the error introduced at each assignment of an approximator. Thus

$$size(C) \;\geq\; \frac{\frac{1}{50}2^n}{2^{n-l}} \;\geq\; \frac{1}{50}2^{\frac{1}{2}n^{1/2d}}.$$

∎

3.9. Applications

In this subsection we give some consequences of the previous lower bounds for relativized computation, log-time computation, and first order definability.

3.9.1. Relativization of the Polynomial Time Hierarchy

Baker, Gill, and Solovay (1975) proposed relativization as a means of showing that many familiar problems in complexity theory cannot be settled within pure recursive function theory. Among their results they give oracles under which P =? NP has an affirmative and a negative answer. They leave open the problem of finding an oracle under which the levels of the polynomial time hierarchy are all distinct. Baker and Selman (1979) extended this giving an oracle separating $\Sigma_2\mathrm{P}$ from $\Pi_2\mathrm{P}$. Furst, Saxe, and Sipser established a connection between exponential lower bounds for constant depth circuits and the existence of the sought after oracle. The bounds given by Yao and Hastad settled this question.

Theorem 3.18: *There is an oracle A such that for every i, $\Sigma_i\mathrm{P}^A \neq \Pi_i\mathrm{P}^A$.*

3.9.2. Log Time Hierarchy

Chandra, Kozen, and Stockmeyer (1981) introduced the notion of alternating Turing machines operating in sub-linear time. In their definition the sub-linear time alternating Turing machine may access the input in a random access manner by specifying the address of the bit to be read on a special address tape which by convention is reset to blank after each read. The alternating log time hierarchy is defined by analogy with the polynomial time hierarchy. Sipser (1983) showed that as a consequence of the AC^0 hierarchy theorem, the levels of the log time hierarchy are all distinct.

Theorem 3.19: $\Sigma_i\mathrm{TIME}(\log n)/\mathrm{poly} = \Sigma_i^{\mathrm{poly,const}}$.

Proof: Given a $\Sigma_i\mathrm{TIME}(\log n)$ machine M we may convert it to a machine M' which reads its input at most once on any computation path and then only after all nondeterministic branches have been made. M' operates by simulating M except that it guesses the answer whenever M reads the input. The guesses are made with the same type of branching ($\wedge$ or $\vee$) in effect at the time. Then M' accepts iff M accepted on that branch and all guesses $g_1, \ldots, g_c$ are correct or if the first incorrect guess was made with $\wedge$-branching. Since M runs in time $O(\log n)$ and each input address has length $\log n$, we know that c is bounded above by a constant independent of n.

We convert M' to a $\Sigma_i^{\mathrm{poly,const}}$ circuit family by taking its $\vee$-$\wedge$ computation tree, collapsing adjacent $\vee$'s and adjacent $\wedge$'s on every branch and adding subcircuits to check the g's. This gives a level of gates for each non-alternating run of M' where the fanin is given by the number of possible configurations (excluding the input tape) of M', except for the lowest level of the circuit where the g's are checked with fixed size subcircuits. Then the inputs to the circuit corresponding to the advice may be preset accordingly.

For the other direction, we may represent each circuit in a $\Sigma_i^{\mathrm{poly,const}}$ circuit family as an advice string of polynomial length in such a way that a $\Sigma_i\mathrm{TIME}(\log n)$ may use its alternation to select a path of the circuit to one of the constant size subcircuits and then evaluate it deterministically. ∎

Corollary 3.20: *For all i, $\Sigma_i\mathrm{TIME}(\log n) \neq \Pi_i\mathrm{TIME}(\log n)$.*

Proof: The function f_i is in $\Pi_i\mathrm{TIME}(\log n)$ but by theorem 3.13 not in $\Sigma_i^{\mathrm{poly,const}}$

and hence not in Σ_iTIME$(\log n)$. ∎

3.9.3. First Order Definability on Finite Structures

As observed by Ajtai (1983) and Immerman (1987) there is a close correspondence between the descriptive power of first order sentences and AC^0. By a *first order sentence* we mean an expression built up from relations, equality, quantifiers ranging over a specified universe, and variables taking values in the specified universe all appearing within the scope of some quantifier. A *structure* specifies the universe and the values of the relations. We say a structure $\mathcal{M}$ *satisfies* a sentence ϕ, or also ϕ *is true in* $\mathcal{M}$, written $\mathcal{M} \models \phi$, if it specifies all of the relation symbols appearing in ϕ and ϕ is true in the familiar mathematical sense with these specifications.

To develop the relationship with AC^0 let us focus on structures with universes U of finite cardinality. We fix a special relation symbol R to be used as the *input relation.* The other predicate symbols $S_1, \ldots, S_l$ are called the *built-in relations.* Given a sentence ϕ, if we specify U and the values of the S_i then we may associate with ϕ the collection of R values which, together with the prespecified S_i values, make ϕ true. For example, if $\phi = \forall x \forall y \, [(S(x) \wedge \neg S(y)) \rightarrow \neg R(x,y)]$ and the universe $U_n = \{1, \ldots, n\}$ with $S = \{1, \ldots, \lfloor n/2 \rfloor\}$ then ϕ defines the class of labeled directed graphs associated with R which contain no edges from the first half of the nodes to the second half. More precisely, if $\mathcal{S} = \{\mathcal{S}_1, \mathcal{S}_2, \ldots\}$ where each $\mathcal{S}_n = \langle S_{n1}, \ldots, S_{nl} \rangle$ and each S_{ni} is a value for the relation symbol S_i in the universe U_n and if $\mathcal{R} = \{\mathcal{R}_1, \mathcal{R}_2, \ldots\}$ where each $\mathcal{R}_n = \{R_{n1}, R_{n2}, \ldots\}$ and each R_{ni} is a value for R in the universe U_n then we say ϕ *defines* $\mathcal{R}$ given $\mathcal{S}$ if for each n the structure $\langle U_n, S_{n1}, \ldots, S_{nl},\, R \rangle \models \phi$ iff $R \in \mathcal{R}_n$.

Say ϕ is Σ_d if it is in prenex normal form, i.e., all quantifiers out in front, and has d alternating blocks beginning with $\exists$. Such an $\mathcal{R}$ above is then called *nonuniformly* Σ_d *first order definable.* Say that $\mathcal{R}$ is *nonuniformly first order definable* if there exists a d such that it is nonuniformly Σ_d first order definable. The built-in predicates S_{ni} play the same role in providing the nonuniform information as does the advice in nonuniform Turing machines. We identify collections of sets of relations $\mathcal{R}$ with the language $A_{\mathcal{R}}$ by encoding each relation with its characteristic binary string.

Theorem 3.21: *For each d, $\mathcal{R}$ is nonuniformly Σ_d first order definable iff $A_{\mathcal{R}}$ is in $\Sigma_d^{\mathrm{poly,const}}$.*

Proof: The easier direction ($\rightarrow$) of this equivalence is seen by constructing circuits simulating a given sentence ϕ. Each non-alternating block of quantifiers becomes a level of unbounded gates, the quantifier free part of ϕ becomes fixed size subcircuits, each atomic formula $S_i(x_1, \ldots, x_{j_i})$ becomes a preset input to the circuit and each $R(x_1, \ldots, x_k)$ becomes an input variable.

For the other direction we are given a $\Sigma_d^{n^l,c}$ circuit C to convert to a Σ_d sentence ϕ. The variables of ϕ are w_i and x_i for $1 \leq i \leq c$ and y_{jk} for $1 \leq j \leq d$, and $1 \leq k \leq l$. Let y_j denote the sequence $y_{j1}, \ldots, y_{jl}$, and y denote the sequence $y_1, \ldots, y_d$. For quantifier Q write Qy_j as a shorthand for $Qy_{j1}Qy_{j2} \cdots Qy_{jk}$. Then ϕ is $\exists y_1 \forall y_2 \cdots Qy_d[\psi]$ where ψ will be described shortly.

Expand all gates of C, except those at the bottom level, so that they have fanin n^l, by adding redundant copies of subcircuits. Then each value for y gives a path through C from the output gate to one of the bottom gates of size at most c. For each y, let g_y denote the associated gate. Expand all bottom gates, by adding redundant variables if necessary, so that each contains exactly c exclusively positive variables, exactly c exclusively negative variables, or a combination of c positive and c negative variables. Call these gates of type p, n,and np respectively. Now we define several relations. Let w denote the sequence $w_1, \ldots, w_c$ and x the sequence $x_1, \ldots, x_c$.

$I^n = \{y\colon\ g_y \text{ is type } n\}$
$I^p = \{y\colon\ g_y \text{ is type } p\}$
$I^{np} = \{y\colon\ g_y \text{ is type } np\}$

$P = \{(y, x)\colon\ x_1, \ldots, x_c \text{ are the positive variables in } g_y\}$
$N = \{(y, x)\colon\ x_1, \ldots, x_c \text{ are the negative variables in } g_y\}$

To complete the description of ϕ, if d is odd we let $\psi = \exists w \exists x[\zeta]$ where

$$\zeta = \left[\begin{array}{l} \left(\Big(I^p(y) \vee I^{np}(y)\Big) \rightarrow \Big(P(y,w) \wedge \bigwedge_{b \leq c} R(w_b)\Big)\right) \\ \\ \wedge \left(\Big(I^n(y) \vee I^{np}(y)\Big) \rightarrow \Big(N(y,x) \wedge \bigwedge_{b \leq c} \neg R(x_b)\Big)\right) \end{array}\right].$$

If d is even we let $\psi = \forall w \forall x[\zeta]$ where

$$\zeta = \left[\begin{array}{l} \left(\left(I^p(y) \vee I^{np}(y) \right) \rightarrow \left(P(y,w) \rightarrow \bigvee_{b \leq c} R(w_b) \right) \right) \\ \\ \wedge \left(\left(I^n(y) \vee I^{np}(y) \right) \rightarrow \left(N(y,x) \rightarrow \bigvee_{b \leq c} \neg R(x_b) \right) \right) \end{array} \right].$$

∎

As a corollary to this theorem and theorem 3.8 we have the theorem of Ajtai (1983) that parity is not nonuniformly first order definable. Let ODD$=\{\mathcal{R}_1, \mathcal{R}_2, \ldots\}$ where $\mathcal{R}_n = \{R \colon R$ is a unary relation on U_n defining a set of odd cardinality$\}$.

Corollary 3.22: *ODD is not nonuniformly first order definable.*

We also obtain a hierarchy theorem as a corollary to the $\Sigma_d^{\mathrm{poly,const}}$ hierarchy theorem.

Corollary 3.23: *For all d there is a class of Σ_d first order definable relations which is not nonuniformly Π_d first order definable.*

It is interesting to contrast these results about nonuniform definability over finite structures with results about definability with built-in relations over infinite structures. For example let $\mathcal{G} = \{R \colon R$ is a binary relation representing a connected infinite graph$\}$. It is easy to show that $\mathcal{G}$ is not first order definable. But if we build in the $+$ and $\times$ relations then all recursive, in fact all arithmetic, relations are definable including $\mathcal{G}$. In essence, the sentence Gödel-encodes a path from a to b as a single integer. The finite analog of $\mathcal{G}$ is not first order definable with any built-in relations since, by reduction from parity, we know that connectedness is not in AC^0. The same proof fails because the encoded path is an exponentially large integer not representable in the finite universe.

4. Monotone Circuits

4.1. Background

This section will survey the lower bounds known for monotone circuits. A *monotone circuit* is a circuit with AND gates and OR gates, but with no NOT gates; the gates may have fanin two and unlimited fanout. A Boolean function f is called *monotone* if $x \leq y$ implies that $f(x) \leq f(y)$, under the usual Boolean ordering. Notice that the only functions computable by monotone circuits are monotone functions. The *monotone circuit complexity* of a monotone function is the size of the smallest monotone circuit computing it.

Many important functions in complexity theory are monotone. As an example, consider the clique function from graph theory. The clique function (written $\text{CLIQUE}_{k,n}$) has $\binom{n}{2}$ variables, one for each potential edge in a graph on n vertices, and outputs 1 iff the associated graph contains a clique (complete subgraph) on some k vertices. The clique function is monotone because setting more edges to 1 can only increase the size of the largest clique.

Strong lower bounds are known for monotone circuits. Razborov (1985a), in a major breakthrough, obtained a superpolynomial lower bound of size $n^{\Omega(\log n)}$ for the monotone circuit complexity of the clique function. (To appreciate the significance of this result, note that the best previous lower bound on the monotone circuit complexity of an explicit, single-output, monotone problem was only $4n$, due to Tiekenheinrich (1984).) Shortly thereafter, Andreev (1985), using methods similar to Razborov, proved an exponential (not just superpolynomial) lower bound for a monotone problem in NP. This implies an exponential lower bound for the clique function since the clique function is complete (with respect to polynomial monotone projections) for "monotone NP" (see Valiant (1979) and Skyum and Valiant (1985)). Alon and Boppana (1987), by strengthening the combinatorial arguments of Razborov, proved a lower bound for $\text{CLIQUE}_{k,n}$ (where $k = n^{2/3}$) of size exponential in $\Omega((n/\log n)^{1/3})$.

If general circuits computing monotone functions could be converted into equivalent monotone circuits with only a polynomial blowup in size, then the above lower bounds would extend to general circuit complexity. Razborov (1985b), using methods similar to his clique lower bound, dashed this possibility by proving that the perfect matching problem for bipartite graphs, known to be in P, requires monotone circuits of superpolynomial size. Tardos (1988) improved the gap to truly exponential for another monotone problem in P.

In spite of the exponential gap between monotone and general circuit complexity, there is a special class of functions for which the two complexities are polynomially related. This is the class of slice functions introduced by Berkowitz (1982). A function f is called a *slice function* if for some integer k, the value of $f(x)$ is 0 when the number of 1's in x is fewer than k, and $f(x)$ is 1 when the number is more than k (but $f(x)$ may be arbitrary when the number is exactly k). Although slice functions may appear to be limited, there do exist NP-complete slice functions. Berkowitz showed that a general circuit computing a slice function can be converted into a monotone circuit by adding only a polynomial number of extra gates. Superpolynomial lower bounds on the *monotone* circuit complexity of explicit slice functions would thus imply that $\mathrm{P} \neq \mathrm{NP}$.

4.2. Razborov Lower Bound for Monotone Circuit Size

To prove lower bounds on monotone circuit complexity, the behavior of small monotone circuits must be shown to be constrained. In Razborov's method to be described below, certain input settings will be designated "test" inputs that compare the circuit's behavior with the behavior of the clique function.

A *positive test graph* is a graph on n vertices that consists of a clique on some set of k vertices, and no other edges; these graphs are called "positive" because the function $\mathrm{CLIQUE}_{k,n}$ outputs 1 on them. Observe that there are $\binom{n}{k}$ such graphs. A *negative test graph* is formed by assigning each vertex a color from the set $\{1, 2, \ldots, k-1\}$, and then putting edges between those pairs of vertices with different colors; these graphs are called "negative" because the function $\mathrm{CLIQUE}_{k,n}$ outputs 0 on them. There are $(k-1)^n$ possible colorings, and although different colorings can lead to the same graph, negative test graphs formed from different colorings will be considered different for counting purposes.

Positive and negative test graphs are designed to measure how closely a circuit agrees with the function $\mathrm{CLIQUE}_{k,n}$. The main goal of Razborov's method is the following.

Goal: *Show that every small monotone circuit either outputs* 0 *on most positive test graphs or outputs* 1 *on most negative test graphs.*

How can the goal be established? Monotone circuits can be amorphous, so to analyze their behavior directly is difficult. Instead, every small monotone circuit will be approximated by a special type of monotone circuit, called an *approximator circuit*. The behavior of approximator circuits will be much easier to analyze than the behavior of arbitrary monotone circuits.

The class of approximator circuits will now be defined. For a subset X of vertices, set the *clique indicator* of X (written $\lceil X \rceil$) to be the function of $\binom{n}{2}$ variables that is 1 if the associated graph contains a clique on the vertices X, and is 0 otherwise. An *approximator circuit* is an OR of at most m clique indicators, each of whose underlying vertex sets have cardinality at most l. Here $l \geq 2$ and $m \geq 2$ will have fixed values, depending only on the values of k and n.

Approximator circuits will be important for establishing the goal. Every monotone circuit C will be assigned an approximator $\widetilde{C}$. The goal will be proved by dividing it into the following two subgoals.

Subgoal 1: *Show that if C is a small monotone circuit, then $C \leq \widetilde{C}$ holds for most positive test graphs, and $C \geq \widetilde{C}$ holds for most negative test graphs.*

Subgoal 2: *Show that every approximator either outputs* 0 *on most positive test graphs or outputs* 1 *on most negative test graphs.*

How can arbitrary monotone circuits be approximated by such special approximator circuits? The approach to be taken is a "bottom-up" construction. Every subcircuit of the original circuit is assigned its own approximator, starting from the input variables and then working up. An input variable is of the form $x_{i,j}$, where i and j are two different vertices; it is equivalent to the clique indicator $\lceil \{i, j\} \rceil$. Hence an input variable is already an approximator.

Suppose that each proper subcircuit of a circuit C has been assigned an approximator circuit. What approximator should be assigned to the entire circuit? Assume, for argument's sake, that the top gate of the circuit C is an OR gate. One natural idea to form the desired approximator is to OR together the approximators of the two subcircuits feeding into the top gate. Let the two approximators be denoted by $A = \bigvee_{i=1}^{r} \lceil X_i \rceil$ and $B = \bigvee_{i=1}^{s} \lceil Y_i \rceil$, where r and s are at most m. The OR of the two approximators is an OR of $r + s$ clique indicators. Unfortunately, $r + s$ can be as large as $2m$, so the OR of the two approximators need not be an approximator itself.

How can the number of clique indicators be reduced? The procedure used here is to replace several clique indicators with their "common" part. To implement this procedure, a combinatorial object called a sunflower is introduced. A *sunflower* is a collection of distinct sets $Z_1, Z_2, \ldots, Z_p$, called *petals*, such that the intersection $Z_i \cap Z_j$ is the same for every pair of distinct indices i and j; the common part $Z_i \cap Z_j$ is called the *center* of the sunflower. In the application to approximator circuits, each petal will be a subset of vertices.

Sunflowers can be used to reduce the number of clique indicators. Fix a value for $p \geq 2$, and look at the current collection of vertex sets $\{X_1, \ldots, X_r, Y_1, \ldots, Y_s\}$. If some p of these vertex sets form a sunflower, replace these p sets with their center. This operation is called a *plucking*. Repeatedly perform such pluckings until no more are possible. This entire procedure is called the *plucking procedure*. Since the number of vertex sets decreases with each plucking, at most $2m$ pluckings will occur. Regarding the number of vertex sets remaining after the plucking procedure is completed, the following combinatorial lemma on sunflowers is useful, due to Erdős and Rado (1960).

Lemma 4.1: *Let $\mathcal{Z}$ be a collection of sets each of cardinality at most l. If $|\mathcal{Z}| > (p-1)^l \cdot l!$, then the collection contains a sunflower with p petals.*

Proof: The proof is by induction on l. The case $l = 1$ is obvious. For $l \geq 2$, let $\mathcal{M}$ be a maximal subcollection of disjoint sets in $\mathcal{Z}$, and let S be the union of the sets in $\mathcal{M}$. If $|\mathcal{M}| \geq p$, then $\mathcal{M}$ itself forms the desired sunflower and we are done. Otherwise we have $|S| \leq (p-1)l$. Since $\mathcal{M}$ is maximally disjoint, the set S intersects every set in $\mathcal{Z}$. By averaging, some element i in S intersects a fraction at least $\frac{1}{(p-1)l}$ of the sets in $\mathcal{Z}$. Consider the following collection of sets of cardinality at most $l-1$:

$$\mathcal{Z}' = \{Z - \{i\} : i \in Z \text{ and } Z \in \mathcal{Z}\}.$$

From the choice of i, we have

$$|\mathcal{Z}'| \geq \frac{|\mathcal{Z}|}{(p-1)l} > (p-1)^{l-1} \cdot (l-1)!.$$

Thus by induction, the collection $\mathcal{Z}'$ contains a sunflower with p petals. Adding i back to all these petals gives the desired sunflower in $\mathcal{Z}$. ∎

To apply the Erdős–Rado lemma to the present situation, set $m = (p-1)^l \cdot l!$. The lemma implies that after the plucking procedure is completed, at most m vertex sets remain. The clique indicators of the remaining vertex sets are then ORed together to form the approximator for the entire circuit. The resulting approximator is called the *approximate OR* of the two approximators A and B, written $A \sqcup B$.

The second case to consider is when the top gate is an AND gate; again, let $A = \bigvee_{i=1}^{r} \lceil X_i \rceil$ and $B = \bigvee_{i=1}^{s} \lceil Y_i \rceil$ be the approximators of the two subcircuits feeding into the top gate. (For technical reasons, assume without loss of generality that none of the sets X_i or Y_i are singleton sets.) Forming the AND of the two approximators yields, by the distributive law, the expression $\bigvee_{i=1}^{r} \bigvee_{j=1}^{s} (\lceil X_i \rceil \wedge \lceil Y_j \rceil)$. Two reasons why this expression is not an approximator itself are that the term $\lceil X_i \rceil \wedge \lceil Y_j \rceil$ is not a clique indicator and that there can be as many as m^2 terms.

To overcome these difficulties, apply the following three steps. First, replace the term $\lceil X_i \rceil \wedge \lceil Y_j \rceil$ by the clique indicator $\lceil X_i \cup Y_j \rceil$. Second, erase those clique indicators $\lceil X_i \cup Y_j \rceil$ for which the cardinality of $X_i \cup Y_j$ is more than l. Finally, apply the plucking procedure (described above for OR gates) to the remaining clique indicators; there will be at most m^2 pluckings. These three steps guarantee that a valid approximator is formed. The resulting approximator is called the *approximate AND* of the approximators A and B, written $A \sqcap B$.

The two operations described above, approximate OR and approximate AND, complete the bottom-up construction of the approximator $\widetilde{C}$ from the monotone circuit C.

4.3. Lower Bound for the Clique Function

The previous subsection observed that lower bounds on the monotone circuit complexity of the clique function follow from proving two subgoals. In this subsection, the two subgoals will be formally stated and proved. The proof given here will combine Razborov's original proof with some of the improvements due to Alon and Boppana. The second subgoal is demonstrated first, since it is the easier of the two subgoals.

Lemma 4.2: *Every approximator circuit either is identically* 0 *or outputs* 1 *on at least* $\left[1 - \binom{l}{2}/(k-1)\right] \cdot (k-1)^n$ *of the negative test graphs.*

Proof: Let $A = \bigvee_{i=1}^{r} \lceil X_i \rceil$ be an approximator circuit. If A is identically 0, then the first conclusion holds. If not, then $A \geq \lceil X_1 \rceil$. A negative test graph is rejected by the clique indicator $\lceil X_1 \rceil$ iff its associated coloring assigns some two vertices of X_1 the same color. Suppose a random coloring is chosen, with each of the $(k-1)^n$ possible colorings equally likely. The probability that some two vertices of X_1 are assigned the same color is bounded above by $\binom{|X_1|}{2}/(k-1) \leq \binom{l}{2}/(k-1)$. Hence the probability that $\lceil X_1 \rceil$ outputs 1 on the associated negative test graph is at least $1 - \binom{l}{2}/(k-1)$. Rewriting this probabilistic statement as a counting statement yields the desired result. ∎

Subgoal 1 will be established by the following two lemmas on the relationship of a circuit C to its approximator $\widetilde{C}$.

Lemma 4.3: *For every monotone circuit C, the number of positive test graphs for which the inequality $C \leq \widetilde{C}$ does not hold is at most size*$(C) \cdot m^2 \cdot \binom{n-l-1}{k-l-1}$.

Proof: Let $A = \bigvee_{i=1}^{r} \lceil X_i \rceil$ and $B = \bigvee_{i=1}^{s} \lceil Y_i \rceil$ be two approximators. Both of the inequalities $A \vee B \leq A \sqcup B$ and $A \wedge B \leq A \sqcap B$ will be shown to fail for at most $m^2 \cdot \binom{n-l-1}{k-l-1}$ positive test graphs. This will imply the lemma because in the transformation from C to $\widetilde{C}$ there are $size(C)$ approximate AND and OR gates.

The inequality $A \vee B \leq A \sqcup B$ is always true, since $A \sqcup B$ is obtained from $A \vee B$ by the plucking procedure. Each plucking can only enlarge the class of accepted graphs.

Next, consider the inequality $A \wedge B \leq A \sqcap B$. The first step in the transformation from $A \wedge B$ to $A \sqcap B$ is to replace $\lceil X_i \rceil \wedge \lceil Y_j \rceil$ by $\lceil X_i \cup Y_j \rceil$. These two functions behave identically on positive test graphs. The second step is to erase those clique indicators $\lceil X_i \cup Y_j \rceil$ for which $|X_i \cup Y_j| \geq l+1$. For each such clique indicator, at most $\binom{n-l-1}{k-l-1}$ of the positive test graphs are lost. Since there are at most m^2 such clique indicators, at most $m^2 \cdot \binom{n-l-1}{k-l-1}$ positive test graphs are lost in the second step. The third and final step, applying the plucking procedure, only enlarges the class of graphs accepted, as noted in the previous paragraph. Summing up the three steps, at most $m^2 \cdot \binom{n-l-1}{k-l-1}$ positive test graphs fail to satisfy $A \wedge B \leq A \sqcap B$, completing the proof. ∎

Lemma 4.4: *For every monotone circuit C, the number of negative test graphs for which $C \geq \widetilde{C}$ does not hold is at most $size(C) \cdot m^2 \cdot \left[\binom{l}{2}/(k-1)\right]^p \cdot (k-1)^n$.*

Proof: Let $A = \bigvee_{i=1}^{r} \lceil X_i \rceil$ and $B = \bigvee_{i=1}^{s} \lceil Y_i \rceil$ be two approximators. The inequalities $A \vee B \geq A \sqcup B$ and $A \wedge B \geq A \sqcap B$ will be shown to fail for at most $m^2 \cdot \left[\binom{l}{2}/(k-1)\right]^p \cdot (k-1)^n$ negative test graphs. As in the proof of lemma 4.3, this will imply the desired result.

First, consider the inequality $A \vee B \geq A \sqcup B$. Recall that $A \sqcup B$ is obtained by performing at most $2m$ pluckings on $A \vee B$. Each plucking will be shown to accept only a few additional negative test graphs. Color the vertices randomly, with all $(k-1)^n$ possible colorings equally likely, and let G be the associated negative test graph. Let $Z_1, Z_2, \ldots, Z_p$ be the petals of a sunflower with center Z. What is the probability that $\lceil Z \rceil$ accepts G, but none of the terms $\lceil Z_1 \rceil, \lceil Z_2 \rceil, \ldots, \lceil Z_p \rceil$ accept G? This event occurs iff the vertices of Z are assigned distinct colors (called a proper coloring, or PC), but every petal Z_i has two vertices colored the same. We have

$$
\begin{aligned}
\Pr[Z \text{ is PC and } Z_1, \ldots, Z_p \text{ are not PC}] &\leq \Pr[Z_1, \ldots, Z_p \text{ are not PC} | Z \text{ is PC}] \\
&= \prod_{i=1}^{p} \Pr[Z_i \text{ is not PC} | Z \text{ is PC}] \\
&\leq \prod_{i=1}^{p} \Pr[Z_i \text{ is not PC}].
\end{aligned}
$$

The first inequality holds by the definition of conditional probability; the second inequality holds by the mutual independence of the events $\{Z_i \text{ is not PC} | Z \text{ is PC}\}$;

and the third inequality holds because the event "Z is PC" is negatively correlated with the other events.

As in the proof of lemma 4.2, we have $\Pr[Z_i \text{ is not PC}] \le \binom{l}{2}/(k-1)$. Substituting this inequality into the chain of inequalities in the previous paragraph shows that

$$\Pr[Z \text{ is PC and } Z_1, \ldots, Z_p \text{ are not PC}] \le \left[\binom{l}{2}/(k-1)\right]^p.$$

Thus to the class of negative test graphs accepted each plucking adds at most $\left[\binom{l}{2}/(k-1)\right]^p \cdot (k-1)^n$ new graphs. There are at most $2m$ pluckings, so the number of negative test graphs violating the inequality $A \vee B \ge A \sqcup B$ is at most $2m \cdot \left[\binom{l}{2}/(k-1)\right]^p \cdot (k-1)^n$. This settles the case of approximate ORs.

Next, consider the inequality $A \wedge B \ge A \sqcap B$. In the transformation from $A \wedge B$ to $A \sqcap B$, the first step introduces no new violations, since $\lceil X_i \rceil \wedge \lceil Y_j \rceil \ge \lceil X_i \cup Y_j \rceil$. The second step of erasing large clique indicators also introduces no new violations. Only the third step, the plucking procedure, introduces new violations. This step was analyzed in the previous two paragraphs; the only difference now is that there can be m^2 pluckings instead of just $2m$. This settles the case of approximate ANDs, thus completing the proof. ∎

Subgoals 1 and 2 have thus been proved; combining them yields the following exponential lower bound on the monotone circuit complexity of the clique function.

Theorem 4.5: *For $k \le n^{1/4}$, the monotone circuit complexity of the function* $\mathrm{CLIQUE}_{k,n}$ *is* $n^{\Omega(\sqrt{k})}$.

Proof: Set $l = \lfloor \sqrt{k} \rfloor$ and $p = \lceil 10\sqrt{k} \log_2 n \rceil$, and recall that $m = (p-1)^l \cdot l!$. Let C be a monotone circuit that computes the function $\mathrm{CLIQUE}_{k,n}$. By lemma 4.2, the approximator $\widetilde{C}$ either is identically 0 or outputs 1 on at least $\frac{1}{2} \cdot (k-1)^n$ of the negative test graphs. If the former case holds, then apply lemma 4.3 to obtain

$$size(C) \cdot m^2 \cdot \binom{n-l-1}{k-l-1} \ge \binom{n}{k}.$$

A simple calculation shows that in this case $size(C)$ is $n^{\Omega(\sqrt{k})}$. Suppose instead that the latter case holds. Applying lemma 4.4 shows that

$$size(C) \cdot m^2 \cdot 2^{-p} \cdot (k-1)^n \geq \frac{1}{2} \cdot (k-1)^n.$$

Another simple calculation shows that in this case $size(C)$ is $n^{\Omega(\sqrt{k})}$. ∎

4.4. Polynomial Lower Bounds

This subsection will describe the known results on functions with polynomial monotone circuit complexity. We present results for both single-output functions and multi-output functions.

Before Razborov's work, only linear lower bounds were known for the monotone circuit complexity of single-output monotone functions in NP. Tiekenheinrich (1984) gave a $4n$ monotone lower bound for a simple explicit function. Dunne (1984) proved a $3.5n$ lower bound on the monotone circuit complexity of the majority function. Majority is known to have monotone circuits of $O(n \log n)$ size by the work of Ajtai, Komlós, and Szemerédi (1983) discussed below.

Turning to multi-output functions, consider the Boolean sorting problem: given n Boolean variables, output their values in nondecreasing order. Boolean sorting is the same problem as simultaneously computing the threshold functions $\mathrm{TH}_{k,n}$ (defined in subsection 2.3) for all k between 1 and n. Ajtai, Komlós, and Szemerédi (1983) gave a very clever construction of monotone circuits of size $O(n \log n)$ for Boolean sorting. Lamagna and Savage (1974) established an $\Omega(n \log n)$ lower bound on the monotone circuit complexity of Boolean sorting. Later, Pippenger and Valiant (1976) and Lamagna (1979) independently showed that Boolean merging, a special case of Boolean sorting, has monotone circuit complexity $\Omega(n \log n)$. Muller and Preparata (1975) observed that Boolean sorting has nonmonotone circuits of linear size, exposing a small gap between monotone and general circuit complexity.

A larger gap was obtained for the problem of Boolean matrix multiplication. This problem takes two n-by-n Boolean matrices as input, and outputs their n-by-n Boolean matrix product. It is trivial to show that Boolean matrix multiplication has monotone circuits of size $2n^3 - n^2$. Pratt (1974) was the first to demonstrate that its monotone circuit complexity is $\Omega(n^3)$. Later, Paterson (1975) and

Mehlhorn and Galil (1976) independently showed that its monotone circuit complexity is exactly $2n^3 - n^2$. Interestingly enough, its general circuit complexity is known to be asymptotically smaller. For example, Coppersmith and Winograd (1987) show that Boolean matrix multiplication has nonmonotone circuits of size $O(n^{2.38})$.

Wegener (1982) has proved monotone circuit lower bounds for a generalization of matrix multiplication, called Boolean direct product, which takes as input several n-by-n matrices. Wegener's results on Boolean direct product give an $\Omega(n^2/\log n)$ lower bound on the monotone circuit complexity of an explicit monotone problem with n inputs and n outputs.

Finally, lower bounds have been discovered for a class of multi-output functions called Boolean sums. A *Boolean sum* has n inputs and n outputs, each output being an OR of some subset of the inputs. Nečiporuk (1969) constructed an explicit Boolean sum that has $\Omega(n^{3/2})$ monotone circuit complexity. Later, Mehlhorn (1979) and Pippenger (1980) independently obtained $\Omega(n^{5/3})$ monotone lower bounds for another explicit Boolean sum. Since then, Andreev (1986) has explicitly contructed, for every fixed $\epsilon > 0$, a Boolean sum with monotone complexity $\Omega(n^{2-\epsilon})$.

5. Formulas

A formula is the special type of circuit whose gates have fanout 1, i.e., a circuit whose underlying graph is a tree. The main motivation for studying formulas is their close relationship to circuit depth. Spira (1971) showed that, over a complete basis, a Boolean function is computable by polynomial-size formulas iff it is computable by logarithmic-depth circuits. Wegener (1983) showed the analogous result for the monotone basis, and Ugol'nikov (1987) announced the analogous result for all fixed bases. Thus strong lower bounds on formula size may lead to lower bounds on circuit depth. In this section, we present the major results known on lower bounds for formula size.

The *size* of a formula is defined to be the number of occurrences of literals (variables or their negations) in the formula. Notice that the size of a formula with binary gates is precisely one more than the number of gates in the formula. Given a collection of Boolean functions Ω, called a *basis*, define the *formula complexity* $L_\Omega(f)$ to be the size of the smallest formula with gates from Ω computing the Boolean function f.

We will consider three bases: the full binary basis B consisting of all 16 binary gates, the DeMorgan basis $D = \{\text{AND, OR, NOT}\}$, and the monotone basis $M = \{\text{AND, OR}\}$. We refer to formulas over these bases as binary formulas, DeMorgan formulas, and monotone formulas respectively. Trivially we have $L_B \leq L_D \leq L_M$. Pratt (1975) showed that $L_D \leq O({L_B}^c)$ for $c = \log_3 10 \approx 2.09$.

The best explicit lower bound known differs for the three bases. For monotone formulas, Karchmer and Wigderson (1988) showed a superpolynomial lower bound on a problem with polynomial monotone circuits. For DeMorgan formulas, Andreev (1987) showed a lower bound of size $\Omega(n^{5/2-\epsilon})$ for every fixed $\epsilon > 0$. For binary formulas, Nečiporuk (1966) showed an $\Omega(n^2/\log n)$ lower bound. All of these results will be proved in this section.

5.1. Karchmer and Wigderson Lower Bound for Monotone Formula Size

This subsection deals with lower bounds on the size of monotone formulas. Of course, the lower bounds for monotone circuits discussed in section 4 imply as a special case lower bounds for monotone formulas. In fact, Razborov (1989b) gives a simpler proof that some monotone NP problems require monotone formulas of superpolynomial size. Nevertheless, these results leave open the question of whether some problem with monotone circuits of polynomial size requires monotone formulas of superpolynomial size. Karchmer and Wigderson (1988) answered this question affirmatively.

The graph s-t connectivity function (written $\mathrm{CONNECT}_n$) takes as input an undirected graph on n ordinary vertices and two distinguished vertices s and t; the function outputs 1 iff the graph has a path from s to t. This function is well-known to be computable in polynomial time, to have monotone circuits of polynomial size and $O((\log n)^2)$ depth, and to have monotone formulas of size $n^{O(\log n)}$. Karchmer and Wigderson originally proved that this function requires monotone circuits of depth $\Omega((\log n)^2/\log\log n)$ depth and monotone formulas of $n^{\Omega(\log n/\log\log n)}$ size. Later R. Boppana and J. Hastad independently simplified the proof and improved the bounds to the optimal $\Omega((\log n)^2)$ depth and $n^{\Omega(\log n)}$ formula size. By the result of Wegener (1983), the depth bound and size bound are actually equivalent, so the proof just focuses on circuit depth. Below we give an overview of the proof, and in the next subsection we give the complete proof.

The Karchmer-Wigderson method uses certain graphs to test a circuit's behavior. An *l-path graph* consists of a path from s to t with a sequence of l ordinary vertices (not necessarily distinct) in between, and no other edges. (If the same vertex appears in two consecutive positions of the sequence, then ignore that edge.) There are n^l possible sequences, and though two of them can lead to the same graph, graphs formed from different sequences will be considered different for counting purposes. A *cut graph* is formed by partitioning the vertices into two components, with s and t in different components, and placing edges between those vertex pairs in the same component. There are 2^n possible cut graphs.

These test graphs will measure how closely a circuit agrees with the function $\mathrm{CONNECT}_n$. Say that a monotone circuit is an (α,β) separator if it outputs 1 for a fraction at least α of the l-path graphs, and outputs 0 for a fraction at least

β of the cut graphs. The main goal of the proof is to show that no shallow monotone circuit can be a good separator; the goal will be established by a "top-down" argument on the structure of a circuit.

Assume C is a shallow monotone circuit computing the function CONNECT_n; clearly C is a (1,1) separator. We will explore C from top to bottom. If the top gate of C is an OR gate, then it is easy to see that one of the two subcircuits feeding into this gate must be a $(\frac{1}{2}, 1)$ separator. Similarly, if the top gate of C is an AND gate, one of the two subcircuits must be a $(1, \frac{1}{2})$ separator. In either case, this subcircuit is certainly a $(\frac{1}{2}, \frac{1}{2})$ separator. By repeating this argument j times, we see that one of the subcircuits j levels from the top is a $(2^{-j}, 2^{-j})$ separator.

Unfortunately, repeating the above argument too many times makes the path-accepting and cut-rejecting densities too low to be of interest. Is there a way to increase the densities? Yes, as follows. Select $\sqrt{n}$ vertices at random and collapse them into vertex s, identifying their edges with the corresponding edges of s. Similarly, select $\sqrt{n}$ other vertices at random and collapse them into vertex t. At the same time, halve the value of the path length l. The crucial fact, and the hardest one to prove, is that performing this vertex-collapsing step will likely convert an (α, β) separator into a $(\frac{\sqrt{\alpha}}{2}, \frac{\beta}{n})$ separator. In other words, the path-accepting density increases greatly, while the cut-rejecting density decreases only slightly.

With the above tools, the proof outline becomes clearer. We constantly strive to maintain a high path-accepting density. When the path-accepting density becomes too low, we apply a vertex-collapsing step to make it high again. The cut-rejecting density constantly, but controllably, decreases. When we finally reach the bottom of the circuit, we will have a depth-0 monotone circuit that is a good separator; since we assumed the original circuit was shallow, the path length l will still be large. But a depth-0 monotone circuit is too limited to be a good separator, yielding a contradiction. Thus the original circuit must have been deep to begin with.

A similar top-down method was used by Klawe, Paul, Pippenger, and Yannakakis (1984) to prove a lower bound for monotone constant-depth circuits. Both they and Karchmer-Wigderson point out a connection between circuit depth and communication complexity.

5.2. Lower Bound for the Connectivity Function

This subsection will present a formal proof of the Karchmer-Wigderson's lower bound for the graph s-t connectivity function, based on the above proof overview.

Definitions: Let f be a monotone Boolean function acting on graphs with n ordinary vertices and two distinguished vertices. Say that f is an (n, l, α) *path acceptor* if it outputs 1 on a fraction at least α of the l-path graphs. Say that f is an (n, β) *cut rejector* if it outputs 0 on a fraction at least β of the cut graphs. Finally, say that f is an (n, l, α, β) *separator* if it is both an (n, l, α) path acceptor and an (n, β) cut rejector.

Definition: Let f be a function acting on graphs with vertex set V, and let ρ be a mapping from V to V'. The *induced function* f_ρ is the following function acting on graphs with vertex set V'. Given a graph $G = (V', E')$, form the graph $G_\rho = (V, E)$ by setting $\{i, j\} \in E$ iff $\rho(i) = \rho(j)$ or $\{\rho(i), \rho(j)\} \in E'$. The value of f_ρ on G is assigned the value of f on G_ρ.

Definition: Let V be a vertex set with two distinguished vertices s and t. The probability distribution $\mathcal{R}_k$ of mappings ρ is defined as follows. First, choose a random k-subset W of $V - \{s, t\}$. Second, place each vertex of W into either set W_s or set W_t, randomly and independently. Finally, form the mapping $\rho: V \to V - W$ by mapping all of W_s into s, mapping all of W_t into t, and mapping every other vertex to itself.

Our first lemma shows that the path-accepting density of a function is likely to increase greatly by switching to a randomly induced function.

Lemma 5.1: *Let f be an (n, l, α) path acceptor, and let ρ be a random mapping from $\mathcal{R}_k$. Suppose that $\frac{100l}{\alpha} \le k \le \frac{n}{100l}$. Then with probability at least $\frac{3}{4}$, the induced function f_ρ is an $(n-k, \frac{l}{2}, \frac{\sqrt{\alpha}}{2})$ path acceptor.*

Proof: Let V be the set of n ordinary vertices. Let P be the collection of l-path graphs that f accepts; we will identify each l-path graph with its sequence of intermediate vertices. Let $L = \{1, \ldots, \frac{l}{2}\}$ and $R = \{\frac{l}{2} + 1, \ldots, l\}$. Given a subpath p in V^L (or V^R), an *extension* of p is a path in V^l that agrees with p on L (R). Define P_L (P_R) to be the collection of subpaths in V^L (V^R) with more than $\frac{\alpha}{4} n^{l/2}$ extensions in P.

We first show that either P_L or P_R is large. Observe that every path in P is either: (a) an element of $P_L \times P_R$, or (b) an extension of some subpath in $V^L - P_L$, or (c) an extension of some subpath in $V^R - P_R$. The number of type-(a) paths in P is at most $|P_L| \cdot |P_R|$. The number of type-(b) paths in P is at most $\frac{\alpha}{4}n^l$, since there are at most $n^{l/2}$ subpaths in $V^L - P_L$ each with at most $\frac{\alpha}{4}n^{l/2}$ extensions in P. Similarly, the number of type-(c) paths in P is at most $\frac{\alpha}{4}n^l$. In total we obtain $|P| \leq |P_L| \cdot |P_R| + \frac{\alpha}{2}n^l$. On the other hand, by hypothesis we have $|P| \geq \alpha n^l$. Comparing the two bounds, it follows that $|P_L| \cdot |P_R| \geq \frac{\alpha}{2}n^l$. Hence either P_L or P_R has size at least $\sqrt{\frac{\alpha}{2}}\, n^{l/2}$; without loss of generality, suppose it is P_L.

Next, consider the following collection of $\frac{l}{2}$-subpaths:

$$P_\rho = \{p \in P_L : p \in (V - W)^L \text{ and } \exists p' \in (W_t)^R \text{ such that } (p, p') \in P\}.$$

From the definition of f_ρ, it is straightforward to see that f_ρ accepts all paths in P_ρ. Thus we only need show that P_ρ is large with high probability.

Toward this goal, we estimate the probability that an arbitrary path p in P_L belongs to P_ρ. The probability that $p \notin (V - W)^L$ is at most $\frac{l}{2} \cdot \frac{k}{n}$, since there are $\frac{l}{2}$ coordinates and each has probability $\frac{k}{n}$ of being in W. This probability is at most $\frac{1}{200}$ by hypothesis.

We next estimate the probability that p fails the second condition of being in P_ρ. Since W_t is a random subset of W, it has size at least $\frac{k}{4}$ with exponentially high probability. In that case, let X be a random $\frac{k}{4}$-subset of W_t; observe that X has the distribution of a random $\frac{k}{4}$-subset of V. One way to form such a random $\frac{k}{4}$ subset is as follows: select $\frac{k}{2l}$ subpaths p'_1, p'_2, $\ldots$, $p'_{k/2l}$ randomly and independently from V^R. Obtain X by taking the union of the vertices of all these subpaths, and if necessary randomly adding new vertices until there are $\frac{k}{4}$ vertices. The probability that p fails the second condition of P_ρ is

$$\begin{aligned}
\Pr[\nexists p' \in (W_t)^R \text{ s.t. } (p, p') \in P] &\leq \Pr[|W_t| < k/4] + \Pr[\nexists i \text{ s.t. } (p, p'_i) \in P] \\
&\leq 2^{-k/10} + \left(1 - \frac{\alpha}{4}\right)^{k/2l} \\
&\leq \frac{1}{200}.
\end{aligned}$$

We now complete the proof. Combining the two estimates above shows that p belongs to P_ρ with probability at least $1 - \frac{1}{200} - \frac{1}{200} = \frac{99}{100}$. Hence the expected size of P_ρ is at least $\frac{99}{100}|P_L|$. On the other hand, we always have $|P_\rho| \leq |P_L|$. Thus with probability at least $\frac{3}{4}$ we have $|P_\rho| \geq \frac{24}{25}|P_L|$, which is greater than $\frac{\sqrt{\alpha}}{2}n^{l/2}$ by our prior estimate of $|P_L|$. This is what we wanted to prove. ∎

Our second lemma shows that the cut-rejecting density of a function is likely to decrease only slightly by switching to a randomly induced function.

Lemma 5.2: *Let f be an (n, β) cut rejector, and let ρ be a random mapping from $\mathcal{R}_k$. Suppose that $\beta \geq 2 \cdot (\frac{3}{4})^{n/k}$. Then with probability at least $\frac{3}{4}$, the induced function f_ρ is an $(n-k, \frac{\beta k}{2n})$ cut rejector.*

Proof: Again let V be the set of n ordinary vertices. Let Q be the collection of cut graphs that f rejects; we will identify each cut graph with a vector in $\{s,t\}^V$. Given a k-subset X of V and a vector x in $\{s,t\}^X$, an *extension* of x is a cut in $\{s,t\}^V$ that agrees with x on X. Define $Q(X)$ to be those vectors in $\{s,t\}^X$ with more than $\frac{\beta k}{2n} \cdot 2^{n-k}$ extensions in Q.

Our first goal is to show that $|Q(X)|$ has large expectation for a random k-subset X. Consider an arbitrary partition of V into $\frac{n}{k}$ subsets $X_1, \ldots, X_{n/k}$ each of size k. A similar argument to the first part of the proof of lemma 5.1 will show that $|Q(X_1)| \ldots |Q(X_{n/k})| \geq \frac{\beta}{2}2^n$. From the arithmetic-geometric mean inequality, we obtain

$$\frac{|Q(X_1)| + \cdots + |Q(X_{n/k})|}{n/k} \geq \left(|Q(X_1)| \ldots |Q(X_{n/k})|\right)^{k/n}$$

$$\geq \left(\frac{\beta}{2}2^n\right)^{k/n}$$

$$\geq \frac{3}{4}2^k.$$

This bound holds for an arbitrary such partition, so by averaging over all such partitions, the expected size of $Q(X)$ is at least $\frac{3}{4}2^k$, where X is a random k-subset of V.

We use this estimate to complete the proof. Let x be the vector in $\{s,t\}^W$ that is s on W_s and t on W_t. Consider the set Q_ρ of subcuts q in $\{s,t\}^{V-W}$ such that (q,x) is in Q. From the definition of f_ρ, it is straightforward to see that f_ρ rejects all the cuts in Q_ρ. It thus suffices to show that Q_ρ is large with high probability. Notice that the event "$|Q_\rho| > \frac{\beta k}{2n} \cdot 2^{n-k}$" is identical to the event "$x \in Q(W)$". Thus by the previous paragraph, we have

$$\Pr[|Q_\rho| > \frac{\beta k}{2n} \cdot 2^{n-k}] = \Pr[x \in Q(W)] = \frac{E[Q(W)]}{2^k} \geq \frac{3}{4},$$

which completes the proof. ∎

Combining lemmas 5.1 and 5.2 yields the following corollary showing how to improve a separator.

Corollary 5.3: *Let f be an (n,l,α,β) separator. Suppose that $\frac{100l}{\alpha} \leq k \leq \frac{n}{100l}$ and $\beta \geq 2 \cdot (\frac{3}{4})^{n/k}$. Then for some mapping ρ in the support of $\mathcal{R}_k$ the induced function f_ρ is an $(n-k, \frac{l}{2}, \frac{\sqrt{\alpha}}{2}, \frac{\beta k}{2n})$ separator.*

We finally prove the superlogarithmic lower bound on the monotone depth complexity of the graph s-t connectivity function.

Theorem 5.4: CONNECT$_n$ *requires monotone circuits of depth $\Omega((\log n)^2)$.*

Proof: Let C be a monotone circuit for CONNECT$_n$, and assume it has depth at most $\frac{1}{25}(\log_2 n)^2$, for n sufficiently large. Adding dummy gates if necessary, we can suppose that the underlying graph of C is a complete binary tree of depth $\frac{1}{25}(\log_2 n)^2$. Set the path length $l = n^{1/4}$ and the path density $\alpha = \frac{1}{4}n^{-1/5}$.

We do a top-down exploration of the circuit C. The circuit C is an $(n,l,1,1)$ separator, and in particular is an $(n,l,\alpha,1)$ separator. Break C into $\frac{1}{5}\log_2 n$ blocks of $\frac{1}{5}\log_2 n$ gate levels each. By following C down one block as described in the proof overview, we obtain a subcircuit that is an $(n,l,\alpha n^{-1/5}, n^{-1/5})$ separator. Applying corollary 5.3 gives a new subcircuit that is an $(n-n^{2/3}, \frac{l}{2}, \alpha, n^{-1})$ separator. Notice that the path-accepting density is back up to α.

Repeat the above process successively for each block of the circuit. Upon reaching the bottom of the circuit, we end up with a monotone depth-0 circuit that is an $(n', l', \alpha, n^{-\log_2 n})$ separator, where $n' = n - \frac{1}{5}n^{2/3}\log_2 n$ and $l' = l/n^{1/5} = n^{1/20}$.

A monotone depth-0 circuit is either a single input variable or a constant. An input variable $x_{i,j}$, for $\{i,j\} \cap \{s,t\} = \emptyset$, accepts a fraction at most $l'/\binom{n'}{2}$ of the l'-path graphs. An input variable of the form $x_{s,i}$ or $x_{i,t}$ accepts a fraction at most $1/n'$ of the l'-path graphs. A constant either rejects all l'-path graphs or accepts all cut graphs. None of these cases gives a good separator, so we have a contradiction. Thus the original circuit C must have had depth larger than $\frac{1}{25}(\log_2 n)^2$. ∎

5.3. Nonmonotone Formulas

This subsection will prove the largest lower bounds known for nonmonotone formulas, the results of Andreev and Nečiporuk.

Let us start with lower bounds for DeMorgan formulas. Observe that negations in DeMorgan formulas may always be pushed down to the bottom level. Random restrictions, as they were for bounded depth circuits (section 3), will be useful here. The probability distribution $\mathcal{R}_k$ of restrictions used here is the following: randomly assign k variables to be $*$, and assign all other variables to be 0 or 1 randomly and independently. Intuitively, a random restriction should reduce considerably the size of a formula. The following lemma of Subbotovskaya (1961) makes this intuition precise.

Lemma 5.5: *Let f be a Boolean function of n variables, and let ρ be a random restriction from $\mathcal{R}_k$. Then with probability at least $\frac{3}{4}$,*

$$L_D(f|_\rho) \leq 4 \cdot \left(\frac{k}{n}\right)^{3/2} \cdot L_D(f).$$

Proof: Let F be an optimal DeMorgan formula for the function f, of size $s = L_D(f)$. Construct the restriction ρ in $n-k$ stages as follows. At any stage, choose a variable randomly from the ones remaining, and assign it 0 or 1 randomly. We analyze the effect of this restriction stage-by-stage.

Suppose the first stage chooses the variable x_i. When the variable x_i is set, the literals x_i and $\overline{x_i}$ will disappear from the formula F. By averaging, the expected number of such literals is $\frac{s}{n}$.

In fact, the formula is likely to be reduced even further. For each of the literals x_i or $\overline{x_i}$, consider the gate which it feeds into. For example, suppose the gate is

$x_i \wedge G$ for some subformula G. We may assume without loss of generality that G does not contain the literals x_i or $\overline{x_i}$. If the variable x_i is assigned 0, then the subformula G will disappear from the formula F, thereby erasing at least one more literal. Since x_i is assigned 0 or 1 randomly, we expect at least $\frac{1}{2} \cdot \frac{s}{n}$ literals to disappear because of these secondary effects. In total, we thus expect at least $\frac{s}{n} + \frac{1}{2} \cdot \frac{s}{n} = \frac{3}{2} \cdot \frac{s}{n}$ literals to disappear in the first stage, yielding a new formula with expected size at most $s \cdot (1 - 3/(2n)) \leq s \cdot (1 - 1/n)^{3/2}$.

The succeeding stages of the restriction can be analyzed in the same way. After each stage the number of variables decrements by one. Hence the expected size of the final formula is

$$E[L_D(f|_\rho)] \leq s \cdot \left(1 - \frac{1}{n}\right)^{3/2} \cdot \left(1 - \frac{1}{n-1}\right)^{3/2} \cdot \dots \cdot \left(1 - \frac{1}{k+1}\right)^{3/2}$$

$$= s \cdot \left(\frac{k}{n}\right)^{3/2}.$$

The probability that $L_D(f|_\rho)$ is more than 4 times its expected value is less than $\frac{1}{4}$, which completes the proof. ∎

Lemma 5.5 is useful for proving lower bounds on formula size. For example, Subbotovskaya (1961) applied it to show that the parity function of n variables requires DeMorgan formulas of size $\Omega(n^{3/2})$. Khrapchenko (1971) later improved the parity lower bound to $\Omega(n^2)$ using a method we describe in subsection 5.4. For now we show how Andreev (1987) used lemma 5.5 to prove an $\Omega(n^{5/2-\epsilon})$ lower bound for another explicit function.

First we define Andreev's function. Given a Boolean function f of b variables, let $f^{\oplus m}$ be the Boolean function of bm variables defined by:

$$f^{\oplus m}(x_1, \dots, x_{bm}) = f(x_1 \oplus \dots \oplus x_m, x_{m+1} \oplus \dots \oplus x_{2m}, \dots, x_{(b-1)m+1} \oplus \dots \oplus x_{bm}).$$

The variables $x_{(i-1)m+1}, \dots, x_{im}$ are said to form the ith *block*. Andreev's function, denoted $A_{b,m}$, is a Boolean function of $2^b + bm$ variables. The first 2^b variables of $A_{b,m}$ will specify a Boolean function of b variables by listing its truth table; call the resulting function f. The value of $A_{b,m}$ is then obtained by applying the function $f^{\oplus m}$ to the last bm variables of $A_{b,m}$. The collection of functions $\{A_{b,m}\}$ is easy to compute in time polynomial in the number of variables. Andreev proved the following lower bound for $A_{b,m}$ over the DeMorgan basis.

Theorem 5.6: *Let $b = \lfloor \log_2 n - 1 \rfloor$ and $m = \lfloor n/(2b) \rfloor$. Then the function $A_{b,m}$ of at most n variables requires DeMorgan formulas of size $\Omega(n^{5/2-\epsilon})$ for every fixed $\epsilon > 0$.*

Proof: Let f be the Boolean function of b variables that requires the largest DeMorgan formulas. An argument analogous to that of theorem 2.4 will show that $L_D(f)$ is $\Theta(2^b/\log b)$. Since $f^{\oplus m}$ is a subfunction of $A_{b,m}$, we have $L_D(A_{b,m}) \geq L_D(f^{\oplus m})$. Let ρ be a random restriction from $\mathcal{R}_k$ for $k = \lceil b \ln(4b) \rceil$.

Applying lemma 5.5, with probability at least $\frac{3}{4}$ we have

$$L_D(f^{\oplus m}|_\rho) \leq 4 \cdot \left(\frac{k}{bm}\right)^{3/2} \cdot L_D(f^{\oplus m}).$$

An easy probability calculation shows that, also with probability at least $\frac{3}{4}$, the restriction ρ assigns at least one $*$ to each of the b blocks of variables. Some restriction ρ will thus satisfy both conditions. From the second condition, we have $L_D(f^{\oplus m}|_\rho) \geq L_D(f)$. Combining the above inequalities, we obtain

$$\begin{aligned} L_D(A_{b,m}) &\geq L_D(f^{\oplus m}) \\ &\geq \frac{1}{4} \cdot \left(\frac{bm}{k}\right)^{3/2} \cdot L_D(f) \\ &= \Omega\left(\left(\frac{bm}{k}\right)^{3/2} \cdot \frac{2^b}{\log b}\right) \\ &= \Omega(n^{5/2-\epsilon}). \end{aligned}$$

This establishes the theorem. ∎

We next turn to lower bounds for binary formulas. Let f be a Boolean function on a variable set X. A *subfunction* of f on $Y \subseteq X$ is a function obtained from f by setting the variables of $X - Y$ to constants. Let $N_Y(f)$ be the number of different nonconstant subfunctions of f on Y. Intuitively, if f has many subfunctions, then it is complicated and hence should require large formulas. This intuition was made precise by Nečiporuk (1966) who proved the following theorem with a weaker constant factor; the version below is due to M. Paterson (unpublished) and Zwick (1987).

Theorem 5.7: *Let f be a Boolean function on a variable set X, and let Y_1, Y_2, $\ldots$, Y_k be disjoint subsets of X. Then*

$$L_B(f) \geq \sum_{i=1}^{k} \log_5[2N_{Y_i}(f) + 1].$$

Proof: Let $S_Y(f)$ be the collection of nonconstant functions g on a variable set Y for which either g or $\neg g$ is a subfunction of f. Clearly $N_Y(f) \leq |S_Y(f)|$. Let $size_Y(F)$ be the number of occurrences of variables of Y in a Boolean formula F. It is simple to see that the theorem follows from the claim below.

Claim 5.8: *For every binary formula F and every variable set Y, we have*

$$2|S_Y(F)| + 1 \leq 5^{size_Y(F)}.$$

The claim is proved by induction on the size of F. The base case $F = x_i$ divides into two subcases ($i \in Y$ or $i \notin Y$); both subcases satisfy the claim. Assume by induction that $F = F_1 * F_2$, where F_1 and F_2 satisfy the claim and $*$ is a binary operator. For brevity let $S_1 = S_Y(F_1)$ and $S_2 = S_Y(F_2)$. Consider the following two collections of Boolean functions on Y:

$$T = \{g_1 * g_2 : g_1 \in S_1 \text{ and } g_2 \in S_2\}$$
$$\widetilde{T} = \{\neg g : g \in T\}.$$

It is straightforward to check that

$$S_Y(F) \subseteq T \cup \widetilde{T} \cup S_1 \cup S_2,$$

and therefore

$$\begin{aligned}
2|S_Y(F)|+1 &\leq 2\cdot(|T|+|\widetilde{T}|+|S_1|+|S_2|)+1 \\
&\leq 4|S_1||S_2|+2|S_1|+2|S_2|+1 \\
&= (2|S_1|+1)\cdot(2|S_2|+1) \\
&\leq 5^{size_Y(F_1)}\cdot 5^{size_Y(F_2)} \\
&= 5^{size_Y(F)}.
\end{aligned}$$

This completes the proof of the claim and hence the proof of the theorem. ∎

Theorem 5.7 can be used to prove lower bounds of size $\Omega(n^2/\log n)$ for many explicit functions. For instance, it applies to Andreev's function $A_{b,m}$, where each subset Y_i consists of one variable from every block. The theorem also applies to the element distinctness function defined in section 6. Unfortunately the theorem is inherently unable to give a lower bound larger than $\Theta(n^2/\log n)$.

5.4. Symmetric Functions

Symmetric functions are those Boolean functions whose value only depends on the number of variables that are 1. Being so natural, symmetric functions have been studied by many researchers in formula complexity. In this subsection, we present the best upper and lower bounds known on the formula complexity of symmetric functions.

We start with upper bounds. That all symmetric functions have polynomial-size formulas is implicit in the work of Ofman (1962) and Wallace (1964), who independently showed how to add n integers with n bits each in logarithmic depth. The best bounds known are due to Khrapchenko (1972) and Peterson (1978). Khrapchenko showed that all symmetric functions have DeMorgan formulas of size $O(n^{4.93})$. Peterson, improving constructions of Pippenger (1974) and Paterson (1977), showed that all symmetric functions have binary formulas of size $O(n^{3.37})$.

For monotone symmetric functions (i.e., threshold functions), it seems more natural to use monotone formulas. Valiant (1984), using an elegant argument, shows that all threshold functions have monotone formulas of size $O(n^{5.3})$. Boppana (1989), improving results of Khasin (1969) and Friedman (1986), showed that the threshold function $\mathrm{TH}_{k,n}$ has monotone formulas of size $O(k^{4.3} n \log n)$. Unfortunately, all these results use probabilistic methods and hence do not explicitly construct the formulas. Ajtai, Komlós, and Szemerédi (1983) give a very clever explicit construction of polynomial-size monotone formulas for all threshold functions, where the degree of the polynomial is a large constant. Using their construction, Friedman (1986) explicitly constructed monotone formulas for $\mathrm{TH}_{k,n}$ of size $O(k^c n \log n)$ for a large constant c.

What about lower bounds for symmetric functions? Over the DeMorgan basis, Khrapchenko (1971) presented a method for obtaining lower bounds of size $\Omega(n^2)$ for certain symmetric functions, which we describe next. Let A and B be two disjoint subsets of $\{0,1\}^n$. A Boolean formula F *separates* A and B if it outputs 0 for every input in A and outputs 1 for every input in B. Define the set

$$A \otimes B = \{(a,b) : a \in A \text{ and } b \in B \text{ and } a \sim b\},$$

where $a \sim b$ means that the inputs a and b differ on exactly one bit. Intuitively, if $A \otimes B$ is large, then every formula separating A and B should be large, since the formula must distinguish many pairs of adjacent inputs. The following theorem of Khrapchenko makes this intuition precise.

Theorem 5.9: *Let F be a DeMorgan formula that separates A and B. Then*

$$size(F) \geq \frac{|A \otimes B|^2}{|A| \cdot |B|}.$$

Proof: (due to M. Paterson) The proof is by induction on the size of F. If the size of F is 1, then F is just a single literal. In that case, it is easy to see that $|A \otimes B| \leq |A|$ and $|A \otimes B| \leq |B|$, settling the base case of the induction.

Assume by induction the theorem holds for all formulas smaller than F, and suppose that $F = F_1 \wedge F_2$ (the case $F = F_1 \vee F_2$ is similar). Define

$$\begin{aligned} A_1 &= \{a \in A : F_1(a) = 0\} \\ A_2 &= A - A_1. \end{aligned}$$

Notice that F_i is a separator of A_i and B for $i = 1, 2$. Define $a_i = |A_i|$, $c_i = |A_i \otimes B|$, and $b = |B|$. Applying the induction hypothesis to the subformula F_i yields

$$size(F_i) \geq \frac{{c_i}^2}{a_i \cdot b}.$$

Thus we obtain

$$\begin{aligned} size(F) &= size(F_1) + size(F_2) \\ &\geq \frac{{c_1}^2}{a_1 \cdot b} + \frac{{c_2}^2}{a_2 \cdot b} \\ &\geq \frac{(c_1 + c_2)^2}{(a_1 + a_2) \cdot b} \end{aligned}$$

where the last inequality can be established by cross-multiplication. Since $c_1 + c_2 = |C|$ and $a_1 + a_2 = |A|$, this completes the induction step for F. ∎

Khrapchenko's theorem shows that the parity function of n variables requires DeMorgan formulas of size $\Omega(n^2)$, which is tight. It also shows that the threshold function $\mathrm{TH}_{k,n}$ requires DeMorgan formulas of size $\Omega(k \cdot (n-k+1))$. For $2 \leq k \leq n-1$, Krichevskii (1963) showed that $\mathrm{TH}_{k,n}$ requires DeMorgan formulas of size $\Omega(n \log n)$, which beats the Khrapchenko bound for small k.

Over the full binary basis, the best lower bounds known for symmetric functions are of size $\Omega(n \log n)$, due to Fischer, Meyer, and Paterson (1982). Their bound applies to the majority function and many other symmetric functions. Pudlák (1984a), improving a result of Hodes and Specker (1968), showed that all symmetric functions of n variables, except for 16 of them having linear-size binary formulas, require binary formulas of size $\Omega(n \log \log n)$. This bound applies for instance to the function $\mathrm{TH}_{k,n}$ for $2 \leq k \leq n-1$.

6. Branching Programs

Branching programs are a natural model on which to investigate the amount of space necessary to compute various functions.

Definitions: A *branching program* is a directed acyclic graph all of whose nodes of nonzero outdegree are labeled with a variable x_i and whose nodes of outdegree zero are labeled with an output value. The edges are labeled by 0 or 1. One of the nodes is designated the start node. A setting of the inputs determines a collection of paths from the start node to output nodes giving a collection of output values. The branching program is *deterministic* if every nonoutput node has exactly one 0 edge and one 1 edge leaving it. Otherwise it is *nondeterministic*. It *accepts* its input if at least one path leads to an accepting output node. The *size* of a branching program is the number of nodes. The branching program complexity of functions and languages is defined analogously with that of circuits. If the nodes are arranged into a sequence of levels with edges going only from one level to the next, then the *width* is the size of the largest level.

6.1. Relationship with Space Complexity

We again select Turing machines as a formal model to define space complexity. Designate one of the tapes to be a read-only input tape and call the others work tapes. The space used by this machine on a given input is defined to be the number of work tape cells scanned by a head at any point in the computation. Define the deterministic, nondeterministic, and alternating space complexity classes by analogy with the time complexity classes. Of particular interest are:

$$\begin{aligned} L &= \mathrm{SPACE}(\log n) \\ NL &= \mathrm{NSPACE}(\log n) \\ PSPACE &= \bigcup_k \mathrm{SPACE}(n^k) \end{aligned}$$

The following connection between the size of branching programs and space complexity was first observed by Masek (1976).

Theorem 6.1: *For $S(n) \geq \log n$, if $A \in SPACE(S(n))$ then A has branching program complexity at most $c^{S(n)}$ for some constant c.*

Proof: The possible configurations of the machine on an input of length n form the nodes of a branching program operating on inputs of length n. Place an edge labeled 0 or 1 from a node a to node b if configuration a reading 0 or 1 respectively can yield configuration b. ∎

The above theorem also applies for nondeterministic space and nondeterministic branching programs.

Theorem 6.2: *For $S(n) \geq \log n$, if $A \in NSPACE(S(n))$ then A has nondeterministic branching program complexity at most $c^{S(n)}$ for some constant c.*

Thus every language in L has polynomial branching program complexity. We may obtain a converse by extending L to the nonuniform class L/poly.

Theorem 6.3: *$A \in$L/poly iff A has polynomial branching program complexity.*

Theorem 6.4: *$A \in$NL/poly iff A has polynomial nondeterministic branching program complexity.*

6.2. Bounds on Size

There have been a number of results bounding the size of branching programs with various restrictions. None of these is yet strong enough to have any implications for space complexity. Just as for Boolean circuits, we know via a counting argument that most functions have exponential branching program complexity. The best lower bound for an explicit function is $\Omega\left(\frac{n^2}{\log^2 n}\right)$ due to Nečiporuk (1966). Using the same method Beame and Cook (unpublished) have observed that the following "element distinctness" function may be proved to require large branching programs. Consider the input to represent m strings $s_1, \ldots, s_m$ each of length $2\log m$ where $n = 2m\log m$. Define the function so that it is 1 iff the s_i are all distinct.

Theorem 6.5: *The element distinctness function requires branching programs of size $\Omega(\frac{n^2}{\log^2 n})$.*

Proof: We show that a function is hard if its inputs may be partitioned into blocks b_i so that there are a large number of functions on each block b_i obtained by setting the remaining variables. By letting b_i be the inputs for s_i we see that the given function has this property. For each b_i there are $\binom{m^2}{m-1}$ ways of setting the remaining s's distinctly and each way gives a different induced function. Each such setting also yields an induced branching program on the nodes labeled from b_i plus an accept and a reject node. Say there are h_i such nodes. The number of branching programs on h_i nodes is at most $n^{h_i} h_i^{2h_i}$. Thus $n^{h_i} h_i^{2h_i} \geq \binom{m^2}{m-1}$ and so $h_i \geq m/2$. The total size of the original branching program is $\sum (h_i - 2) + 2$ since the accept and reject nodes are common but the others are not. This is $\Omega(m^2)$ and hence $\Omega(\frac{n^2}{\log^2 n})$. ∎

Since stronger lower bounds on general branching programs seem hard to come by, researchers have turned to restricted versions. Furst, Saxe, and Sipser (1984) and Borodin, Dolev, Fich, and Paul (1983) conjectured that the majority function required superpolynomial size if the width of the branching program is held fixed. Partial progress was made by Chandra, Furst, and Lipton (1983) who gave a superlinear lower bound. Pudlak (1984b) improved this by eliminating the width restriction. Then Barrington (1989), with a surprising construction, disproved this conjecture. He showed that constant width branching programs are unexpectedly powerful, being able to accept all NC^1 languages.

Theorem 6.6: *A language has polynomial size, width 5 branching programs iff it is in nonuniform* NC^1.

Proof: One direction ($\rightarrow$) is easy since a fixed amount of circuitry can compose two levels into one. Doing this in parallel across the branching program and repeating it $\mathrm{O}(\log n)$ times yields the desired NC^1 circuit.

For the other direction we construct a special kind of width 5 branching program from the $\mathrm{O}(\log n)$ depth circuit. All levels have 5 nodes and all nodes in a given level are labeled with the same variable. Additionally, at each level the 0 edges and the 1 edges going to the next level form permutations. Then an input setting yields a permutation which is the composition of the selected permutations at each level. Call such a branching program p a *permuting* branching program and let $p(x)$ be the resulting permutation on input x. For a language B and permutation σ say that branching program p *σ-accepts* B if for each string $x \in B$ we have $p(x) = \sigma$ and for each $x \notin B$ we have $p(x) = e$, the identity permutation.

A permutation is *cyclic* if it is composed of a single cycle on all of its elements. In the following lemmas let σ and τ be cyclic permutations, B and C be languages, and p and q be permuting branching programs.

Lemma 6.7: *If p σ-accepts B then there is a permuting branching program of the* *same size τ-accepting B.*

Proof: Since σ and τ are cyclic we may write $\tau = \gamma\sigma\gamma^{-1}$ for some permutation γ. Then simply reorder the left and right nodes of p according to γ to obtain the τ-accepting branching program.

Lemma 6.8: *If p σ-accepts B then there is a permuting branching program of the same size σ-accepting $\overline{B}$.*

Proof: Use the previous lemma to obtain a σ^{-1}-acceptor for B. Then reorder the final level by σ so that it becomes a σ-acceptor for $\overline{B}$.

Lemma 6.9: *If p σ-accepts B and q τ-accepts C then there is a permuting branching program $\sigma\tau\sigma^{-1}\tau^{-1}$-accepting $B \cap C$ of size $2(size(p) + size(q))$.*

Proof: Use lemma 6.7 to get a σ^{-1}-acceptor for B and a τ^{-1}-acceptor for C. Now compose these 4 acceptors in the order $\sigma, \tau, \sigma^{-1}, \tau^{-1}$. This has the desired effect because replacing either σ or τ by e in $\sigma\tau\sigma^{-1}\tau^{-1}$ yields e.

Lemma 6.10: *There are cyclic permutations σ and τ in S_5 such that $\sigma\tau\sigma^{-1}\tau^{-1}$ is cyclic.*

Proof: $(12345)(13542)(54321)(24531) = (13254)$.

The above lemma is the only place where the value 5 is important.

Now we may finish the proof of the theorem. If a depth d circuit contains only NOT gates and AND gates of fanin 2, then it is easily seen by the above lemmas that we may construct a width 5 branching program of length at most 4^d. Hence, if the depth is $O(\log n)$ then the branching program has polynomial size. ∎

7. Conclusion

Let us speculate on the future for research on lower bounds on the complexity of finite functions. Looking back, it is clear that the last decade has seen some important progress on restricted models of computation. It is hard to say whether this has any bearing on the unrestricted case. An optimist might argue that by considering models with weaker and weaker restrictions one may incrementally approach the unrestricted case. A very recent result of Razborov (1989a) indicates however that the unrestricted case may be qualitatively different from the cases that have been successfully treated so far. This result shows that the "approximation method," which lies at the heart of all of the strong lower bounds presented in this chapter except for the Karchmer-Wigderson monotone depth bound, cannot be used to prove superpolynomial lower bounds on the size of general circuits. It seems that a fundamentally different idea is needed.

A different approach was proposed by Sipser (1981). Using an analogy between countability and polynomiality one may derive a suggestive correspondence between finite complexity and definability in descriptive set theory (see Moschovakis (1980) for a comprehensive treatment). In this way the class AC^0 corresponds to the class of Borel sets and the class NP corresponds to the class of analytic sets. The former correspondence played a critical role in the discovery of the proof of the theorem of Furst, Saxe, and Sipser (1984). By first formulating and solving an infinite version of their conjecture about the parity function they were guided in the search for a solution to the conjecture.

The latter correspondence may have more direct bearing on the P=NP, or more precisely, the NP=co-NP problem. Sipser (1984) gives a new proof of the classical theorem due to Lebesgue (1905) stating that the analytic sets are not closed under complement. There is a possibility that this proof contains a hint of how to proceed with showing that NP is not closed under complement.

Acknowledgements

We wish to thank M. Karchmer, M. Paterson, N. Pippenger, A.A. Razborov and U. Zwick for extensive comments on an earlier draft of this paper.

8. References

L. Adleman (1978), "Two theorems on random polynomial time," *Proceedings of 19th Annual IEEE Symposium on Foundations of Computer Science*, 75–83.

M. Ajtai (1983), "Σ_1^1-formulae on finite structures," *Annals of Pure and Applied Logic* 24, 1–48.

M. Ajtai and M. Ben-Or (1984), "A theorem on probabilistic constant depth circuits," *Proceedings of 16th Annual ACM Symposium on Theory of Computing*, 471–474.

M. Ajtai and Y. Gurevich (1987), "Monotone versus positive," *Journal of the ACM* 34:4, 1004–1015.

M. Ajtai, J. Komlós, and E. Szemerédi (1983), "An $O(n \log n)$ sorting network," *Proceedings of 15th Annual ACM Symposium on Theory of Computing*, 1–9. Journal version in *Combinatorica* 3:1, 1–19.

N. Alon and R. B. Boppana (1987), "The monotone circuit complexity of Boolean functions," *Combinatorica* 7:1, 1–22.

A. E. Andreev (1985), "On a method for obtaining lower bounds for the complexity of individual monotone functions," *Doklady Akademii Nauk SSSR* 282:5, 1033–1037 (in Russian). English translation in *Soviet Mathematics Doklady* 31:3, 530–534.

A. E. Andreev (1986), "On a family of Boolean matrices," *Vestnik Moskovskogo Universiteta, Matematika* 41:2, 97–100 (in Russian). English translation in *Moscow University Mathematics Bulletin* 41:2, 79–82.

A. E. Andreev (1987), "On a method for obtaining more than quadratic effective lower bounds for the complexity of π-schemes," *Vestnik Moskovskogo Universiteta, Matematika* 42:1, 70–73 (in Russian). English translation in *Moscow University Mathematics Bulletin* 42:1, 63–66.

T. P. Baker, J. Gill, and R. Solovay (1975), "Relativizations of the P =?NP question," *SIAM Journal on Computing* 4:4, 431-442.

T. P. Baker and A. L. Selman (1979), "A second step toward the polynomial hierarchy," *Theoretical Computer Science* 8, 177–187.

D. A. Barrington (1989), "Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1," *Journal of Computer and System Sciences* 38, 150–164.

C. Bennett and J. Gill (1981), "Relative to a random oracle A, $\mathrm{P}^A \neq \mathrm{NP}^A \neq$ co-NP^A with probability 1," *SIAM Journal on Computing* 10, 96–113.

S. J. Berkowitz (1982), "On some relationships between monotone and non-monotone circuit complexity," Technical Report, Computer Science Department, University of Toronto.

N. Blum (1984), "A Boolean function requiring $3n$ network size," *Theoretical Computer Science* 28, 337–345.

R. B. Boppana (1986), "Threshold functions and bounded depth monotone circuits," *Journal of Computer and System Sciences* 32:2, 222–229.

R. B. Boppana (1989), "Amplification of probabilistic Boolean formulas," *Advances in Computer Research*, Volume 5 on Randomness and Computation, Editor S. Micali, JAI Press, to appear.

A. Borodin, D. Dolev, F. E. Fich, and W. J. Paul (1983), "Bounds for width two branching programs," *Proceedings of 15th Annual ACM Symposium on Theory of Computing*, 87–93.

A. K. Chandra, M. L. Furst, and R. J. Lipton (1983), "Multiparty protocols," *Proceedings of 15th Annual ACM Symposium on Theory of Computing*, 94–99.

A. K. Chandra, D. Kozen, and L. Stockmeyer (1981), "Alternation," *Journal of the ACM* 28, 114–133.

A. K. Chandra, L. J. Stockmeyer, and U. Vishkin (1984), "Constant depth reducibility," *SIAM Journal on Computing* 13:2, 423–439.

S. A. Cook (1985), "A taxonomy of problems with fast parallel algorithms," *Information and Control* 64, 2–22.

D. Coppersmith and S. Winograd (1987), "Matrix multiplication via arithmetic progressions," *Proceedings of 19th Annual ACM Symposium on Theory of Computing*, 1–6.

P. E. Dunne (1984), "Lower bounds on the monotone network complexity of threshold functions," *Proceedings of 22nd Annual Allerton Conference on Communication, Control and Computing*, 911–920.

P. E. Dunne (1988), *The Complexity of Boolean Networks*, Academic Press.

J. Edmonds (1965), "Paths, trees, and flowers," *Canadian Journal of Mathematics*, 17, 449–467.

P. van Emde Boas (1989), "Machine models and simulations," this handbook.

P. Erdős and R. Rado (1960), "Intersection theorems for systems of sets," *Journal of London Mathematical Society* 35, 85–90.

M. J. Fischer, A. R. Meyer, and M. S. Paterson (1982), "$\Omega(n \log n)$ lower bounds on length of Boolean formulas," *SIAM Journal on Computing* 11:3, 416–427.

J. Friedman (1986), "Constructing $O(n \log n)$ size monotone formulae for the k-th elementary symmetric polynomial of n Boolean variables," *SIAM Journal on Computing* 15:3, 641–654.

M. Furst, J. Saxe, and M. Sipser (1984), "Parity, circuits and the polynomial time hierarchy," *Mathematical Systems Theory* 17, 13–27.

R. L. Graham, B. L. Rothschild, and J. H. Spencer (1980), *Ramsey Theory*, John Wiley & Sons, New York.

L. H. Harper and J. E. Savage (1972), "The complexity of the marriage problem," *Advances in Mathematics* 9, 299–312.

J. Hartmanis and R. E. Stearns (1965), "On the computational complexity of algorithms," *Transactions of the AMS* 117, 285–306.

J. Hastad (1989), "Almost optimal lower bounds for small depth circuits," *Advances in Computer Research*, Volume 5 on Randomness and Computation, Editor S. Micali, JAI Press, to appear. See also *Computational Limitations for Small Depth Circuits*, MIT Press, 1986.

L. Hodes and E. Specker (1968), "Length of formulas and elimination of quantifiers I," *Contributions to Mathematical Logic*, Editors H. A. Schmidt, K. Schutte, and H.-J. Thiele, North-Holland, Amsterdam, 175–188.

J. E. Hopcroft and J. D. Ullman (1979), *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley.

N. Immerman (1987), "Languages that capture complexity classes," *SIAM Journal on Computing* 16:4, 760–778.

M. Karchmer and A. Wigderson (1988), "Monotone circuits for connectivity require super-logarithmic depth," *Proceedings of 20th Annual ACM Symposium on Theory of Computing*, 539–550.

R. M. Karp and R. Lipton (1982), "Turing machines that take advice," *L'enseignment Mathematique* 28, 191–209.

L. S. Khasin (1969), "Complexity bounds for the realization of monotone symmetrical functions by means of formulas in the basis $+$, $\cdot$, $-$," *Doklady Akademii Nauk SSSR* 189:4, 752–755 (in Russian). English translation in *Soviet Physics Doklady* 14:12 (1970), 1149–1151.

V. M. Khrapchenko (1971), "A method of determining lower bounds for the complexity of Π-schemes," *Matematicheskie Zametki* 10:1, 83–92 (in Russian). English translation in *Mathematical Notes of the Academy of Sciences of the USSR* 10:1, 474–479.

V. M. Khrapchenko (1972), "The complexity of the realization of symmetrical functions by formulae," *Matematicheskie Zametki* 11:1, 109–120 (in Russian). English translation in *Mathematical Notes of the Academy of Sciences of the USSR* 11:1, 70–76.

M. Klawe, W. J. Paul, N. Pippenger, and M. Yannakakis (1984), "On monotone formulae with restricted depth," *Proceedings of 16th Annual ACM Symposium on Theory of Computing*, 480–487.

R. E. Krichevskii (1963), "Complexity of contact circuits realizing a function of logical algebra," *Doklady Akademii Nauk SSSR* 151:4, 803–806 (in Russian). English translation in *Soviet Physics Doklady* 8:8 (1964), 770–772.

E. A. Lamagna (1979), "The complexity of monotone networks for certain bilinear forms, routing problems, sorting and merging," *IEEE Transactions on Computing* 28, 773–782.

E. A. Lamagna and J. E. Savage (1974), "Combinational complexity of some monotone functions," *Proceedings of 15th Annual IEEE Symposium on Switching and Automata Theory*, 140–144.

H. Lebesgue (1905), "Sur les fonctions représentables analytiquement," Journal de Math. 6^e serie 1, 139–216 (in French).

W. Masek (1976), "A fast algorithm for the string editing problem and decision graph complexity," M.S. thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology.

K. Mehlhorn (1979), "Some remarks on Boolean sums," *Acta Informatica* 12, 371–375.

K. Mehlhorn and Z. Galil (1976), "Monotone switching circuits and Boolean matrix product," *Computing* 16, 99–111.

Y. N. Moschovakis (1980), *Descriptive Set Theory*, North-Holland.

D. E. Muller (1956), "Complexity in electronic switching circuits," *IRE Transactions on Electronic Computers* 5, 15–19.

D. E. Muller and F. P. Preparata (1975), "Bounds to complexities of networks for sorting and switching," *Journal of the ACM* 22:2, 195–201.

E. I. Nečiporuk (1966), "A Boolean function," *Doklady Akademii Nauk SSSR* 169:4, 765–766 (in Russian). English translation in *Soviet Mathematics Doklady* 7:4, 999–1000.

E. I. Nečiporuk (1969), "On a Boolean matrix," *Problemy Kibernetiki* 21, 237–240 (in Russian). English translation in *Systems Theory Research* 21 (1971), 236–239.

Yu. Ofman (1962), "On the algorithmic complexity of discrete functions," *Doklady Akademii Nauk SSSR* 145:1, 48–51 (in Russian). English translation in *Soviet Physics Doklady* 7:7 (1963), 589–591.

E. A. Okol'nishnikova (1982), "On the influence of negations on the complexity of a realization of monotone Boolean functions by formulas of bounded depth," *Metody Diskretnogo Analiza* 38, 74–80 (in Russian).

M. S. Paterson (1975), "Complexity of monotone networks for Boolean matrix product," *Theoretical Computer Science* 1:1, 13–20.

M. S. Paterson (1976), "An introduction to Boolean function complexity," Astérisque 38–39, 183–201.

M. S. Paterson (1977), "New bounds on formula size," *Proceedings of 3rd GI Conference on Theoretical Computer Science, Lecture Notes in Computer Science* 48, Springer-Verlag, Berlin, 17–26.

W. J. Paul (1977), "A 2.5n lower bound on the combinational complexity of Boolean functions," *SIAM Journal on Computing* 6:3, 427–443.

G. L. Peterson (1978), "An upper bound on the size of formulae for symmetric Boolean functions," Technical Report 78-03-01, Department of Computer Science, University of Washington.

N. Pippenger (1974), "Short formulae for symmetric functions," IBM Research Report RC–5143, Yorktown Heights.

N. Pippenger (1979), "On simultaneous resource bounds," *Proceedings of 20th Annual IEEE Symposium on Foundations of Computer Science*, 307–311.

N. Pippenger (1980), "On another Boolean matrix," *Theoretical Computer*

Science 11, 49–56.

N. Pippenger and M. J. Fischer (1979), "Relations among complexity measures," *Journal of the ACM* 26, 361–381.

N. Pippenger and L. G. Valiant (1976), "Shifting graphs and their applications," *Journal of the ACM* 23, 423–432.

V. R. Pratt (1974), "The power of negative thinking in multiplying Boolean matrices," *SIAM Journal on Computing* 4, 326–330.

V. R. Pratt (1975), "The effect of basis on size of Boolean expressions," *Proceedings of 16th Annual IEEE Symposium on Foundations of Computer Science*, 119–121.

P. Pudlák (1984a), "Bounds for Hodes-Specker theorem," *Proceedings of Symposium on Rekursive Kombinatorik*, *Lecture Notes in Computer Science* 171, Springer-Verlag, Berlin, 421–445.

P. Pudlák (1984b), "A lower bound on complexity of branching programs," *Proceedings of 11th Mathematical Foundations of Computer Science*, *Lecture Notes in Computer Science* 176, Springer-Verlag, Berlin, 480–489.

A. A. Razborov (1985a), "Lower bounds on the monotone complexity of some Boolean functions," *Doklady Akademii Nauk SSSR* 281:4, 798–801 (in Russian). English translation in *Soviet Mathematics Doklady* 31, 354–357.

A. A. Razborov (1985b), "A lower bound on the monotone network complexity of the logical permanent," *Matematicheskie Zametki* 37:6, 887–900 (in Russian). English translation in *Mathematical Notes of the Academy of Sciences of the USSR* 37:6, 485–493.

A. A. Razborov (1987), "Lower bounds on the size of bounded depth networks over a complete basis with logical addition," *Matematicheskie Zametki* 41:4, 598–607 (in Russian). English translation in *Mathematical Notes of the Academy of Sciences of the USSR* 41:4, 333–338.

A. A. Razborov (1989a), "On the method of approximations," *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, 167–176.

A. A. Razborov (1989b), "Applications of matrix methods to the theory of lower bounds in computational complexity," *Combinatorica*, to appear.

J. E. Savage (1972), "Computational work and time on finite machines," *Journal of the ACM* 19:4, 660–674.

J. E. Savage (1976), *The Complexity of Computing*, Wiley.

C. E. Shannon (1949), "The synthesis of two-terminal switching circuits," *Bell Systems Technical Journal* 28:1, 59–98.

M. Sipser (1981), "On polynomial vs. exponential growth," unpublished.

M. Sipser (1983), "Borel sets and circuit complexity," *Proceedings of 15th Annual ACM Symposium on Theory of Computing*, 61–69.

M. Sipser (1984), "A topological view of some problems in complexity theory," *Colloquia Mathematica Societatis János Bolyai* 44, 387–391.

S. Skyum and L. G. Valiant (1985), "A complexity theory based on Boolean algebra," *Journal of the ACM* 32:2, 484–502.

R. Smolensky (1987), "Algebraic methods in the theory of lower bounds for Boolean circuit complexity," *Proceedings of 19th Annual ACM Symposium on Theory of Computing*, 77–82.

P. M. Spira (1971), "On time-hardware complexity tradeoffs for Boolean functions," *Proceedings of 4th Hawaii Symposium on System Sciences*, Western Periodicals Company, North Hollywood, 525–527.

B. A. Subbotovskaya (1961), "Realizations of linear functions by formulas using $+, \cdot, -$," *Doklady Akademii Nauk SSSR* 136:3, 553–555 (in Russian). English translation in *Soviet Mathematics Doklady* 2, 110–112.

É. Tardos (1988), "The gap between monotone and non-monotone circuit complexity is exponential," *Combinatorica* 8:1, 141–142.

J. Tiekenheinrich (1984), "A $4n$ lower bound on the monotone network complexity of a one-output Boolean function," *Information Processing Letters* 18, 201–202.

A. M. Turing (1936), "On computable numbers with an application to the Entscheidungsproblem," *Proceedings of the London Mathematical Society* 2:42, 230–265. Correction in *ibid.* 2:43, 544–546.

A. B. Ugol'nikov (1987), "Complexity and depth of formulas realizing functions from closed classes," *Proceedings of Fundamentals of Computation Theory, Lecture Notes in Computer Science* 278, Springer-Verlag, Berlin, 456–461.

L. G. Valiant (1979), "Completeness classes in algebra," *Proceedings of 11th ACM Annual Symposium on Theory of Computing*, 249–261.

L. G. Valiant (1984), "Short monotone formulae for the majority function," *Journal of Algorithms* 5, 363–366.

C. S. Wallace (1964), "A suggestion for a fast multiplier," *IEEE Transactions on Computers* 13:1, 14–17.

I. Wegener (1982), "Boolean functions whose monotone complexity is of size $n^2/\log n$," *Theoretical Computer Science* 21, 213–224.

I. Wegener (1983), "Relating monotone formula size and monotone depth of Boolean functions," *Information Processing Letters* 16, 41–42.

I. Wegener (1987), *The Complexity of Boolean Functions*, Wiley-Teubner.

A. C. Yao (1985), "Separating the polynomial-time hierarchy by oracles," *Proceedings of 26th Annual IEEE Symposium on Foundations of Computer Science*, 1–10.

U. Zwick (1987), "Optimizing Nečiporuk's theorem," Ph.D. thesis, Department of Computer Science, Tel Aviv University.