The Expressive Power of Voting Polynomials

James Aspnes*† Richard Beigel*‡ Merrick Furst§ Steven Rudich§

October 14, 1993

Abstract

We consider the problem of approximating a Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$ by the sign of an integer polynomial p of degree k. For us, a polynomial $p(x)$ predicts the value of $f(x)$ if, whenever $p(x) \geq 0$, $f(x) = 1$, and whenever $p(x) < 0$, $f(x) = 0$. A low-degree polynomial p is a good approximator for f if it predicts f at almost all points. Given a positive integer k, and a Boolean function f, we ask, "how good is the best degree k approximation to f?" We introduce a new lower bound technique which applies to any Boolean function. We show that the lower bound technique yields tight bounds in the case f is parity. Minsky and Papert [10] proved that a perceptron can not compute parity; our bounds indicate exactly how well

*Yale University, Dept. of Computer Science, P.O. Box 208285, New Haven CT 06520-8285.

†Email: aspnes-james@cs.yale.edu.

‡Email: beigel-richard@cs.yale.edu. Supported in part by NSF grants CCR-8808949 and CCR-8958528.

§Carnegie-Mellon University, School of Computer Science, Pittsburgh, PA 15213-3890.

a perceptron can *approximate* it. As a consequence, we are able to give the first correct proof that, for a random oracle A, PP^A is properly contained in PSPACE^A. We are also able to prove the old AC^0 exponential-size lower bounds in a new way. This allows us to prove the new result that an AC^0 circuit with one majority gate cannot approximate parity. Our proof depends only on basic properties of integer polynomials.

1. Introduction

Linial, Mansour, and Nisan [9]; Tarui [16]; and Beigel, Reingold, and Spielman [2] have shown that polynomial-size, bounded-depth circuits can be closely approximated as the sign of a low-degree polynomial over the rationals. This result closely ties the class $\mathbf{AC}^0$ [7, 18] to the class of low-degree polynomials over the rationals, much as previous work by Razborov [12] and Smolensky [14] ties $\mathbf{AC}^0$ to the class of low-degree polynomials over finite fields. Unfortunately, the lower bound techniques known for polynomials over finite fields do not generalize to the *signs* of a polynomials over an ordered field; and despite considerable recent interest in this latter representation [4, 5, 11, 8], to date few techniques have been developed which yield lower bounds on the degree of such polynomials.

In this paper we describe a result which relates the degree of a polynomial over the rationals to its ability to accurately approximate particular Boolean functions. Our bounds are tight in the case of the parity function. In the language of perceptrons [10], this shows the maximum number of inputs on which a perceptron of order k can output parity. Much as similar results for polynomials over finite fields yield lower bounds for circuits containing MOD_p gates, this result yields lower bounds for circuits of AND and OR gates with unbounded fan-in and a single threshold gate. While this class of circuits is somewhat limited compared to more general classes of threshold circuits, it is still surprisingly powerful. This can be seen by considering a few simple but instructive "puzzles". These demonstrate that there are cases where the subcircuits which are the inputs of the threshold gate are uncorrelated with the function that the full circuit approximates, yet the full circuit can manage to compute the function for all but a small fraction of the inputs.

1.1. Voting Puzzles

Puzzle 1:

Let n be an odd number of women. Let each have a uniformly chosen random bit on her forehead. Each person can see all the bits except her own. They wish to vote on the parity of the bits. No communication between the voters is allowed. More precisely, each person casts a private vote (1 or 0); the outcome of the election is the value which the majority cast. The n women are said to win the election in the case when the outcome is equal to the parity of the n bits. *What is a collective strategy for the voters which gives them a high probability of winning the election?*

Note that in this puzzle, no individual voter ever learns any information about the parity of the bits, and each voter will be wrong exactly half of the time. It is tempting to believe that the voters as a group will be able to win only half the time as well. Surprisingly, the voters can manage to win with high probability with a very simple strategy.

Solution 1A:

If a voter sees as many 0's as 1's, she casts a vote of 0. Otherwise, she assumes that the bit on her forehead is the same as the majority of the bits she sees; she then casts a vote consistent with this assumption.

In the case where the number of 0's and 1's differs by more than one, each person will vote assuming her bit is the same as the majority of bits. The majority of women are correct in assuming that the bit on their head is the same as the bit which is in the majority— thus the majority of votes cast are for the parity of the n bits. The voters win the election.

In the case where the number of 0's is exactly one more than the number of 1's, the majority of women will vote 0. Zero is the correct answer. The voters win the election.

In the remaining case where the number of 0's is exactly one less than the number of 1's, the majority of women will again vote 0. This value is wrong. The voters lose the election. This case occurs in $\binom{n}{n/2}$ of 2^n cases, i.e. with probability $\frac{1}{\theta(\sqrt{n})}$.

The voters thus have a $1 - \frac{1}{\theta(\sqrt{n})}$ chance of winning the election, which is much better than 50%.

Though the above strategy is optimal over all strategies where the voters behave identically, it is not the optimal solution over unrestricted strategies.

Solution 1B:

For convenience, assume n is of the form $2^k - 1$. Divide the voters into k groups numbered from 0 to $k-1$, where group i contains 2^i voters. Further divide the voters in each group except the first into two equal-sized groups: the 0-half and 1-half. Denote the parity of the forehead bits in group i by P_i. The strategy of a voter in the b-half of group j is given as follows: If there exists an $i < j$ where $P_i = 0$, vote b. If not, assume $P_j = 0$ and vote accordingly.

We argue that in all the cases where there exists a j such that $P_j = 0$ the above strategy wins the election. Let j be the lowest numbered group such that $P_j = 0$. Each voter in groups numbered $i < j$ voted incorrectly because they falsely assumed $P_i = 0$. This accounts for $2^j - 1$ incorrect votes. Each voter in group j votes correctly. There are 2^j voters in group j; thus there is one more correct vote than incorrect vote cast among the votes of the first j groups. The voters in the 0- and 1-halves of each higher group cancel each other out and have no effect on the election.

The remaining case, where $P_j = 1$ for all j, occurs with probability $1/2^k = 1/(n+1)$. In this case the voters all vote incorrectly. Thus the above strategy wins with probability $1 - 1/(n+1)$, which is better than the strategy presented in solution 1A.

Now, given any strategy, tallying the votes over all possible assignments will always count as many correct votes as incorrect votes, since each voter is correct only half of the time. We conclude that solution 1B is optimal, because the incorrect votes are optimally distributed. When the voters win, they win by exactly one vote; when they lose, they all vote incorrectly.

Let's generalize the puzzle a little.

Puzzle 2:

The voters have moved to Chicago, a city famous for its lax election laws. Each voter can now cast as many votes as she wants! Fix the other parameters as before. Once again, *what is a collective strategy for the voters which has a high probability of their winning the election?*

Solution 2:

Have a each voter behave as did a whole group in solution 1B. Number the voters from 0 to $n-1$. The strategy for voter j is as follows: If a lower numbered voter has a 0 on her forehead, she abstainsn from voting (votes 0 times). Otherwise, she assumes she has a 0 on her forehead and cast 2^j votes accordingly.

The analysis is as in 1B except that we now have n groups. The probability of winning is $1-1/2^n$. This is clearly optimal, for across all assignments as many incorrect votes as correct votes must be cast. Thus there must be one assignment on which the voters are wrong.

Puzzle 3:

Let S be a set of n uniformly chosen random bits. This time there are $\binom{n}{k}$ voters, each of whom sees a distinct subset of S of size k. The voting is Chicago style, each voter casting an integer vote. If the sum of the votes is positive, then the outcome of the election is 0. If the sum is negative, the outcome is 1. If the sum is 0, the outcome is undefined. Again, the voters wish to decide parity. *Given n and k, what is a collective strategy for the voters which maximizes their probability of winning the election?*

When $k = n-1$, Puzzle 3 reduces to Puzzle 2. For general k, the puzzle can be restated as a natural question about the subject of this paper: *voting polynomials.*

1.2. Voting Polynomials

If we are given a voting strategy in which each voter sees at most k bits, we can represent each voter's net vote (counting a vote for b as $(-1)^b$) by a polynomial in the k bits that it sees. Thus the total vote can be computed as a polynomial of degree at most k in all of the input bits, and determining the winner of an election consists of taking the sign of this polynomial. Conversely, given a degree-k polynomial with integer coefficients, we can assign each term to a voter who can see the bits on which its value depends; to determine its vote, each voter computes the sum of all terms it has been assigned. Thus there is an equivalence between computing a Boolean function as the result of an election with limited information and computing it as the sign of a low-degree polynomial with rational coefficients. For this reason we refer to the latter representation as a **voting polynomial**.

Voting polynomials are a special kind of **perceptron** [10], a circuit which computes a weighted threshold of a set of arbitrary predicates. The **order** of a perceptron is the maximum number of input bits available to any one predicate. An order k perceptron is essentially the same as what we have been calling a voting strategy in which each voter sees at most k bits. That voting strategies compute the same functions as voting polynomials tells us that when considering order k perceptrons we can limit ourselves to ones in which each predicate is the parity of a subset of the input, a critical restriction which allows us to apply algebraic techniques to the question of what functions a perceptron can approximate well.

To define the computation of voting polynomials more formally, we say that a function p **strongly represents**[1] a Boolean function f just in case $\mathrm{sgn}(p(x)) = f(x)$ for all input vectors x. Though our terminology is unusual the underlying concept is a standard one; see for example [4, 5, 8]. The **strong degree** of f, written $d_s(f)$, is the least k for which there exists a degree-k polynomial which strongly represents f. Henceforth, when we say a polynomial p represents f we mean it strongly represents f. For example, majority is represented by the degree 1 polynomial $\Sigma x_i - n/2$. Hence, $d_s(MAJ) = 1$.

[1] As distinct from *weakly represents*, introduced in Section 2

For the most part we are interested not so much in computing functions exactly as in approximating them. We say that a polynomial p **approximates** a function f with e errors if the number of inputs x on which $\mathrm{sgn}(p(x)) \neq \mathrm{sgn}(f(x))$ is no greater than e. Our central concern will be to determine for various functions the minimum number of errors in their best approximations by polynomials of fixed degree. This is equivalent to determining how well a set of voters each of whom sees only a bounded-size subset of the inputs can compute these functions. For example, to know how well a degree k polynomial can approximate parity is to know the answer to "puzzle" 3.

We will answer this particular question in Section 3. First, however, it is necessary to state a few definitions.

1.3. Boolean Functions

A Boolean function on n variables will be represented as a function from $\{1, -1\}^n$ to $\{1, -1\}$, where each bit b is replaced by the real value $(-1)^b$. In this representation the parity function on a set of variables $x_1, x_2, \ldots x_k$ is simply the monomial $\prod_{i=1}^{k} x_i$.

The set of all real-valued functions on $\{1, -1\}^n$ can be thought of as a 2^n-dimensional vector space F_n where $(f+g)(x) = f(x)+g(x)$ and $(af)(x) = af(x)$ for any functions f and g and scalar a. A natural basis for this vector space is the set of functions which take on the value 1 on exactly one possible input and take on the value 0 for all others; this basis suggests the inner product $f \cdot g = \sum_{x \in \{1,-1\}^n} f(x)g(x)$.

An alternative basis can be constructed from the monomials $\phi_S(x) = \prod_{i \in S} x_i$. Because $x_i^2 = 1$, the function obtained by taking the product of any two monomials ϕ_S and $\phi_{S'}$ is itself a monomial, $\phi_{S \triangle S'}$. This fact allows one to show that any two distinct monomials are orthogonal, for $\sum \phi_S(x)\phi_{S'}(x) = \sum \phi_{S \triangle S'}(x)$, and if $S \triangle S'$ is nonempty, $\phi_{S \triangle S'}(x)$ will be -1 for exactly half of all inputs x. Since the number of monomials is precisely 2^n, the dimension of F_n, the monomials form an alternative orthogonal basis for the space.

A function is a **polynomial of degree** k if it can be expressed as a linear combination of monomials over sets of size k or less. The set of all polynomials $\mathcal{P}_k$ of degree $\leq k$ is a subspace of F_n of dimension $\sum_{i=0}^{k} \binom{n}{i}$. It should be clear that $F_n = P_n$.

A function is **symmetric** if it is invariant under all permutations of its input variables, or, looking at equivalence classes, if $f(x) = f(x')$ whenever $\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} x'_i$. We will often find it convenient to treat a symmetric polynomial p in $x_1 \ldots x_n$ as polynomial in the single integer quantity $\sum_{i=1}^{n} x_i$. If we restrict ourselves to polynomials of degree at most n, this latter form is unique and can be obtained by polynomial interpolation. It is not difficult to see that the degree of the polynomial will be the same in either form.

2. Lower Bounds on Approximations

In this section we describe a property of Boolean functions which partially characterizes how difficult they are to approximate.

Relax the conditions for representing a function as follows: let p **weakly represent** f just in case p is not the constant zero function, and $\operatorname{sgn}(p(x)) = \operatorname{sgn}(f(x))$ for all x where $p(x)$ is nonzero. In electoral terms, those values x where $p(x) = 0$ correspond to situations in which the voters "deadlock", delivering a majority to neither value; the difference between strong and weak representation is that in the latter such deadlocks are allowed, so long as the majority casts the correct vote in those situations where deadlock does not occur.

We can define the **weak degree** of a function f, $d_w(f)$, analogously to its strong degree, as the least k for which there exists a degree k polynomial which weakly represents f. This notion is useful because for functions of known weak degree it is possible to place a lower bound on the distance to any low-degree polynomial approximation. The method is to show that any function p which closely approximates a function f can be converted into a weak representative for f without substantially increasing its degree. We do so by multiplying p by a low-degree polynomial q which sends all the inputs x for which $\mathrm{sgn}(p(x)) \neq \mathrm{sgn}(f(x))$ to 0 without changing the sign of any input not sent to 0. The following lemma tells us when we can find such a function.

Lemma 2.1 *Let S be a set of inputs such that $|S| < \sum_{i=0}^{k} \binom{n}{i}$. Then there exists a degree $2k$ polynomial q such that $q \neq 0$, $q(x) \geq 0$ for all x, and $q(x) = 0$ for all $x \in S$.*

Proof: Any degree k polynomial has $\sum_{i=0}^{k} \binom{n}{i}$ coefficients, and its value on any particular input is a linear combination of those coefficients. Thus if r stands for a degree k polynomial, the constraints $r(x) = 0$ for all x in S are a homogeneous system of $|S|$ linear equations in $\sum_{i=0}^{k} \binom{n}{i}$ variables, and have a non-trivial solution since $|S| < \sum_{i=0}^{k} \binom{n}{i}$. But then $q = r^2$ is a non-trivial degree $2k$ polynomial that is 0 on S and non-negative elsewhere. ∎

The full result is stated in the following theorem.

Theorem 2.2 *If p is a degree k polynomial and f any Boolean function. Let S be the set of all x*

for which $\operatorname{sgn}(p(x)) \neq \operatorname{sgn}(f(x))$. *Then if* $k < d_w(f)$,

$$|S| \geq \sum_{i=0}^{\lfloor \frac{d_w(f)-k-1}{2} \rfloor} \binom{n}{i}$$

Proof: Suppose otherwise; then by the preceding lemma there is a non-trivial degree $d_w(f)-k-1$ polynomial q which is 0 on S and non-negative elsewhere. Now, pq weakly represents f, as for any x either $pq(x) = 0$ or $x \notin S$ and $\operatorname{sgn}(pq(x)) = \operatorname{sgn}(p(x)) = \operatorname{sgn}(f(x))$. But then pq is a weak representative of f with degree $d_w(f) - 1$, a contradiction. ∎

2.1. Computing Weak Degrees

Unfortunately, it is not trivial to determine the weak degree of an arbitrary function. However, there are functions whose weak degrees are easily determined. One such is the parity function:

Lemma 2.3 *The weak degree of the parity function* ϕ *is* n.

Proof: Suppose p weakly represents parity. Then $p \cdot \phi > 0$, since each term in $\sum p(x)\phi(x)$ is nonnegative and at least one term is nonzero. But parity is orthogonal to all other monomials, thus if p is in $\mathcal{P}_{n-1}$, $p \cdot \phi = 0$. ∎

The parity function is central to the connection between strong and weak degrees. We will first need a small technical lemma which is a disguised form of an old result in the theory of linear inequalities:

Lemma 2.4 *Let S be a linear subspace of the space of real-valued functions on $\{1,-1\}^n$, and let f be a function from $\{1,-1\}^n$ to $\{1,-1\}$. Then there exists some $p \in S$ which strongly represents f if and only if there does not exist any nonzero $q \in S^\perp$ such which weakly represents f.*

Proof: Let σ be the linear transformation which maps g to fg. Then $\sigma(f)$ is the constant 1 and, since σ is easily seen to be an isometry, $\sigma(S)^\perp = \sigma(S^\perp)$. Thus we may assume without loss of generality that f is 1.

Regard S as the set of solutions p to $Ap = 0$ for some appropriate matrix A; then $S^\perp$ is the row space of A consisting of all vectors of the form $q = yA$. We may thus recast the original statement as: there exists p such that $Ap = 0$ and $p > 0$ if and only if there does not exist $q = yA$ such that $q \neq 0$ and $q \geq 0$. This restatement is Stiemke's Transposition Theorem [15] (see [13, 6].) ∎

We may now show the connection between strong and weak degrees.

Lemma 2.5 *For any function f on n bits, $d_s(f) + d_w(f\phi) = n$.*

Proof: Suppose p has degree $d_s(f)$ and strongly represents f and q has degree $d_w(f\phi)$ and weakly represents $f\phi$. Then pq has degree at most $d_s(f) + d_w(f\phi)$ and weakly represents parity. Hence $d_s(f) + d_w(f\phi) \geq n$ by Lemma 2.3.

To show $d_s(f)+d_w(f\phi) \leq n$, suppose that $d_s(f) = k$. Then there is no degree $k-1$ representative of f, so the subspace $\mathcal{P}_{k-1}$ of the space of all real-valued functions on n bits contains no p such that $p(x)\,\mathrm{sgn}(f(x)) > 0$ for all x. But then Lemma 2.4 tells us that there is a nonzero q in $\mathcal{P}_{k-1}^\perp$ such that $q(x)\,\mathrm{sgn}(f(x)) \geq 0$ for all x. Since $\mathcal{P}_{k-1}^\perp$ is spanned by the monomials of degree k and higher, when we multiply q by parity we get a polynomial consisting of monomials of degree $n-k$ and lower which is a weak representative for $f\phi$. Thus $d_w(f\phi) \leq n-k$ and $d_s(f)+d_w(f\phi) \leq n$. ∎

2.1.1. Symmetric Functions

Using Lemma 2.5 it is straightforward to characterize both the weak and strong degree of any symmetric function.

Lemma 2.6 *If f is a symmetric function, then $d_s(f)$ and $d_w(f)$ are both equal to the number of times f changes sign when expressed as a univariate function in $\sum x_i$.*

Proof: Clearly if f changes sign only k times there is a degree-k polynomial which strongly (and, *a fortiori*, weakly) represents f. Furthermore, $f\phi$ changes sign $n-k$ times so there is a degree $n-k$ polynomial which represents $f\phi$. Thus $d_w(f) \leq k$ but by Lemma 2.5 $d_w(f) = n - d_s(f\phi) \geq k$. Thus $d_w(f) = k$, and since $d_w(f) \leq d_s(f) \leq k$, $d_s(f) = k$. ∎

2.1.2. Random Functions

What is the expected weak degree of a randomly-chosen function? Combining Lemma 2.5 and the fact that $d_w(f) \leq d_s(f)$ we can easily see that the average weak degree is at most $n/2$. We conjecture that both the strong and weak degrees are in fact exactly $n/2$ (when n is even) for almost all functions. This conjecture is supported by empirical evidence for small values of n, but we have been unable to prove a lower bound on the weak degree of a random function stronger than the $O(n/\log n)$ bound implied by Gotsman [8].

3. Approximating Parity

Suppose that we wish to approximate the parity function on n bits with a polynomial of degree k (or, equivalently, to construct a solution to Puzzle 3; or to approximate parity with a perceptron of order k.) Theorem 2.2 tells us that any degree k polynomial must differ in sign from parity on at least $\sum_{i=0}^{\lfloor (n-k-1)/2 \rfloor} \binom{n}{i}$ inputs. How close to this bound can we get?

One approach we might take is to approximate parity with a symmetric function, expressed as a univariate polynomial $v_{n,k}$ in $\sum x_i$. As we are only concerned with the sign of $v_{n,k}$ our only relevant decision is how to place its k zeroes to maximize the number of inputs x on which it correctly predicts parity. Since s is binomially distributed we are likely to be best off "spending" zeroes to get the correct values when s is close to 0.

This rule of thumb suggests the following plausible definition for $v_{n,k}$. To simplify the analysis we express the polynomial in the variable $s = \sum (1 - x_i)/2$, the count of the number of times -1 appears in the input vector, making the parity of x simply $(-1)^s$. Thus $v_{n,k}(s)$ is:

$$(-1)^{\lfloor (n-k)/2 \rfloor} \prod_{i=0}^{k-1} \left(\left\lfloor \frac{n-k}{2} \right\rfloor + i + \frac{1}{2} - s \right)$$

Here the term before the product sets the sign when $s = \left\lfloor \frac{n-k}{2} \right\rfloor$, and the product itself places zeroes to set the sign correctly for all values in the range $\left\lfloor \frac{n-k}{2} \right\rfloor \leq s \leq \left\lfloor \frac{n-k}{2} \right\rfloor + k$. Outside of this range $v_{n,k}$ will predict parity correctly on roughly half the inputs. It is not difficult to see that $v_{n,k}$ can be expanded into a polynomial of degree k on the inputs $x_1 \ldots x_n$.

What is more difficult to see is how well $v_{n,k}$ predicts parity. The first step is to write an expression for the number of inputs x on which $v_{n,k}$ has the "wrong" sign in the outer regions of the distribution of s:

$$\sum_{i\geq 0}\binom{n}{\lfloor (n-k)/2\rfloor - 1 - 2i}$$
$$+\quad \sum_{i\geq 0}\binom{n}{\lfloor (n-k)/2\rfloor + k + 1 + 2i}$$

When $n-k$ is odd, an application of the symmetry identity $\binom{n}{i} = \binom{n}{n-i}$ causes the terms in the summation on the right to exactly fill in the "gaps" in the summation on the left, leaving (after a change of the summation variable) $\sum_{i=0}^{\lfloor (n-k)/2\rfloor}\binom{n}{i}$, which is just the lower bound $\sum_{i=0}^{\lfloor (n-k-1)/2\rfloor}\binom{n}{i}$ since $n-k$ is odd. Thus in this case $v_{n,k}$ approximates parity optimally.

The case where $n-k$ is even is slightly more complicated. Here, an application of symmetry yields:

$$2\sum_{i\geq 0}\binom{n}{\lfloor (n-k)/2\rfloor - 1 - 2i}$$

Applying the Pascal's triangle identity expands each term in this summation into two terms in the following summation:

$$2\sum_{i=0}^{\lfloor (n-k)/2\rfloor - 1}\binom{n-1}{i} \tag{1}$$

which is precisely twice the lower bound $\sum_{i=0}^{\lfloor (n-k-2)/2\rfloor}\binom{n-1}{i}$ on the number of inputs on which any degree k polynomial on $n-1$ bits must disagree with the parity of those bits. This pleasant coincidence shows that (1) is in fact a lower bound on the number of errors of any degree k approximation to parity on n inputs, for if some f had fewer errors, at least one of the degree k polynomials on $n-1$ bits obtained by fixing x_n in f to 1 or -1 would approximate parity on the remaining $n-1$ bits better than allowed by the lower bound of Theorem 2.2.

We have just completed the proof the following theorem:

Theorem 3.1 *The symmetric polynomial $v_{n,k}$ on n inputs approximates parity optimally for all values of n and k.*

Thus the voting strategy based on $v_{n,k}$ is a complete solution to Puzzle 3.

3.1. Approximating other symmetric functions

In general, Lemma 2.6 suggests that the difficulty of approximating a symmetric function increases as the number of sign changes increases. Unfortunately, the bound of Theorem 2.2 is not tight for symmetric functions in general. Perhaps surprisingly, it is not even the case that for every symmetric function the best approximation of given degree is symmetric. Consider the function on four bits which is positive only when exactly two of its input bits are 1. This symmetric function changes sign twice, and the best symmetric linear approximation, which matches one of the function's sign changes, fails on five inputs. However, the asymmetric polynomial $2(x_1 - 1) - (x_2 + x_3 + x_4)$ fails on only four inputs: when $x_1 = -1$ and two of the other inputs are 1 (three cases) and when $x_1 = 1$ and all of the other inputs are -1 (one case.)[2]

There is a curious contrast here. If we wish to compute a symmetric function exactly, Lemma 2.6 shows that we can do so with the lowest possible degree using a symmetric polynomial. But allowing the polynomial to be incorrect on some inputs breaks symmetry. It is an open question whether the symmetric functions do share any property which would yield a tighter lower bound than that of Theorem 2.2.

[2] One can verify by exhaustive search that no degree 1 approximation fails on fewer inputs.

4. $\mathbf{PP}^A \neq \mathbf{PSPACE}^A$ for random A

Bennet and Gill [3] assert that $\mathbf{PP}^A \neq \mathbf{PSPACE}^A$ with probability 1 relative to a random oracle A. Unfortunately, their proof has a bug [1] and the question of separating the two classes relative to a random oracle has remained open. Fortunately, the lower bound of Theorem 2.2 gives us a way to show that the two classes are in fact distinct.

Theorem 4.1 $\mathbf{PP}^A \neq \mathbf{PSPACE}^A$ *for random* A *with probability* 1.

Proof: Let ODD^A be the set of all x such that an odd number of strings of length $|x|$ are in A. Clearly ODD^A is in $\mathbf{PSPACE}^A$. Now suppose that M^A is a probabilistic oracle machine which makes at most $|x|^c$ (c constant) oracle queries on any input x. We can convert the computation of M^A on a particular input x to a voting strategy in which each voter represents all computation paths which query a particular set of strings of size $|x|^c$, and the vote cast is the net difference between the number of paths which decide 0 and the number of such paths which decide 1. The resulting voting strategy thus attempts to compute the parity of $n = 2^{|x|}$ bits with each voter seeing at most $|x|^c = O(\log n)$ bits. By Theorem 2.2 it must fail with probability at least $2^{-n}\sum_{i=0}^{n/2-O(\log n)}\binom{n}{i}$; for n sufficiently large (e.g., when $|x|^c < \sqrt{n}$) this value will always be larger than some positive constant ϵ.

Since the choice of M^A and c were arbitrary, we know that *any* probabilistic polynomial-time oracle machine will fail to compute $\mathrm{ODD}^A(x)$ for any typical x with probability at least ϵ. Lemma 1 from [3] shows that this property is sufficient to prove that $\mathrm{ODD}^A \notin \mathbf{PP}^A$ with probability 1. ∎

5. Relation to Bounded-Depth Circuits

In this section we describe a method for constructing from an $\mathbf{AC}^0$ circuit with a majority gate at its root a voting polynomial which closely approximates it. The construction is based on that of Beigel, Reingold, and Spielman [2]. The following key lemma is a simplification of their Lemma 5. Note that in this construction bits will be represented by the real values 0 and 1.

Lemma 5.1 *For any $\epsilon > 0$ and any distribution of the inputs there exists a degree $O(\log(1/\epsilon)\log(n))$ polynomial on $x_1 \ldots x_n$ which computes their OR, $\bigvee x$, with probability at least $1-\epsilon$.*

Proof: We will use a stripped-down version of a theorem of Valiant and Vazirani [17]. Let S_0 be the set of variables. For each $i \leq \log n + 1$, let S_i be constructed randomly from S_{i-1} by removing each variable with probability 1/2. Let $p_i = \sum_{x_j \in S_i} x_j$; clearly p_i is a degree 1 polynomial in x. Now consider some input x in which some x_j is nonzero. Then one of the following cases must hold:

1. $p_i > 1$ for all i. This occurs with probability at most $n2^{-(\log n+1)} = 1/2$.

2. $p_0 = 1$.

3. There is some i such that $p_i > 1$ but $p_{i+1} \leq 1$. Now for any j, the probability that $p_{j+1} = 0$ given p_j is 2^{-p_j} and the probability that $p_{j+1} = 1$ is $p_j 2^{-p_j}$; hence the probability that $p_{j+1} = 1$ conditioned on $p_{j+1} \leq 1$ is $p_j/(p_j+1)$. Thus for any i where $p_i > 1$ and $p_{i+1} \leq 1$, $p_{i+1} = 1$ with probability at least 2/3.

Since one of the latter two cases occurs with probability at least 1/2, the probability that some $p_i = 1$ is at least 1/3.

Now let $p = \prod_{i=0}^{\log n+1}(1 - p_i)$; p is thus a randomly-chosen polynomial of degree $O(\log n)$. If all the x_i are 0, each p_i is 0 and p will be 1. Otherwise, some p_i will be 1 with probability 1/3, and p will be 0. Thus for each input x, $1 - p$ computes $\bigvee x$ with probability 1/3. The probability of correctness can be amplified to $1 - \epsilon$ by using $1 - p'$ where p' is the product of $O(\log(1/\epsilon))$ different p's, each generated independently of the others; $1 - p'$ has degree $O(\log(1/\epsilon)\log(n))$. Now, since a randomly-chosen $1 - p'$ computes $\bigvee x$ with probability at least $1 - \epsilon$ for any individual x, for each input distribution there must exist some fixed $1 - p'$ which computes $\bigvee x$ with probability at least $1 - \epsilon$ when x is randomly chosen from that distribution. ∎

Corollary 5.2 *For any $\epsilon > 0$ and any function f computed by an AND-OR circuit of depth d and size s, there exists a polynomial of degree $O((\log(s/\epsilon)\log(s))^d)$ which computes f for all but $2^n\epsilon$ inputs.*

Proof: Replace each AND gate in the circuit with an OR gate whose inputs and output are negated. Consider the distribution of the inputs of each gate when the inputs of the circuit are chosen uniformly at random; by Lemma 5.1, using s as an upper bound on the number of inputs to the gate, there exists some polynomial of degree $O(\log(s/\epsilon)\log(s))$ which computes the value of the gate with probability at least $1 - \epsilon/s$ when the inputs to the circuit are generated uniformly. The composition of these polynomials is a polynomial of degree $O((\log(s/\epsilon)\log(s))^d)$ which computes f with probability at least $1 - \epsilon$ when the inputs are generated uniformly, i.e. which computes f for all but $2^n\epsilon$ inputs. ∎

Lemma 5.3 *Let $\epsilon > 0$ and let f be a function computed by an AND-OR circuit of depth $d+1$ and size s with a single majority gate at the root. Then there exists a voting polynomial of degree $O((\log(s^2/\epsilon)\log(s))^d)$ which approximates f with at most $2^n\epsilon$ errors.*

Proof: Suppose that the majority gate has k inputs; for the subcircuit generating the i-th input use Corollary 5.2 to construct a polynomial p_i of degree $O((\log(sk/\epsilon)\log(s))^d)$ which computes that input for all but $2^n\epsilon/k$ inputs. Then $\operatorname{sgn}\sum_{i=1}^{k} p_i - k/2$ is a polynomial of degree $O((\log(sk/\epsilon)\log(s))^d)$ which computes f for all but at most $2^n\epsilon$ inputs. But since k is bounded by s we may rewrite the degree as $O((\log(s^2/\epsilon)\log(s))^d)$. ∎

An immediate consequence of the preceding Lemma and Theorem 2.2 is a lower bound on the size of a bounded-depth circuit with at most one majority gate which computes parity:

Lemma 5.4 *If a depth $d+1$ AND-OR circuit with majority gate at its root computes parity, its size is $2^{\Omega(n^{1/4d})}$.*

Proof: Suppose the size of the circuit is s. Then by Lemma 5.3 there exists a voting polynomial p of degree $O((\log(4s^2)\log(s))^d) = O(\log(s)^{2d})$ which approximates parity with at most $2^n/4$ errors. But by Theorem 2.2 the degree of p must then be $\Omega(\sqrt{n})$, and thus $s = 2^{\Omega(n^{1/(4d)})}$. ∎

Corollary 5.5 *If a depth $d+1$ AND-OR circuit containing one majority gate (not necessarily at the root) computes parity, its size is $2^{\Omega(n^{1/4(d+3)})}$.*

Proof: Transform the circuit into a circuit with a majority gate at its root as follows. Let C be the circuit, and let C_0 and C_1 be the circuits obtained by replacing the majority gate m in C with a constant 0 or 1, respectively. Send the output of circuits which compute C_0, C_1, and the inputs to m into a circuit with a majority gate m' at its root which acts as follows:

- If $C_0 = C_1$, set all the inputs to m' to C_0.
- If $C_0 = 0$ and $C_1 = 1$, set the inputs to m' to the values of the inputs to m. Thus if m would have computed the value x, m' will compute the same value x, which gives the correct answer for C since $C_x = x$.
- If $C_0 = 1$ and $C_1 = 0$, set the inputs to m' to the negations of the inputs to m. Thus if m would have computed the value x, m' will compute $\overline{x}$, which gives the correct answer for C since in this case $C_x = \overline{x}$.

This combining circuit can be built in depth 3; thus the depth of the entire new circuit is $d+4$ and its size is easily seen to be at most $3s + O(n)$ where s is the size of the original circuit. The result follows. ∎

6. Acknowledgments

Bert Enderton supplied the slick proof in Solution 1B of puzzle 1. We would like to thank David Applegate, Bob Floyd, Simon Kasif, Dick Lipton, and Gabor Tardos for useful discussions.

References

[1] Richard Beigel. Relativized counting classes: Relations among thresholds, parity, and mods. *Journal of Computer and System Sciences*, 42(1):76–96, February 1991.

[2] Richard Beigel, Nick Reingold, and Daniel Spielman. The perceptron strikes back. In *Proceedings of the 6th Annual Conference on Structure in Complexity Theory*, pages 286–291, 1991.

[3] Charles H. Bennett and John Gill. Relative to a random oracle A, $\mathrm{P}^A \neq \mathrm{NP}^A \neq \text{co-NP}^A$ with probability 1. *SIAM Journal on Computing*, 10(1):96–113, February 1981.

[4] Jehoshua Bruck. Harmonic analysis of polynomial threshold functions. *SIAM Journal on Discrete Mathematics*, 3(2):168–177, May 1990.

[5] Jehoshua Bruck and Roman Smolensky. Polynomial threshold functions, AC^0 functions and spectral norms. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, pages 632–641, 1990.

[6] Vašek Chvátal. *Linear Programming*. W.H. Freeman and Company, 1983.

[7] M. Furst, J. Saxe, and M. Sipser. Parity, circuits and the polynomial time hierarchy. *Mathematical Systems Theory*, 17:13–27, 1984.

[8] Craig Gotsman. On boolean functions, polynomials, and algebraic threshold functions. Unpublished manuscript.

[9] Nathan Linial, Yishay Mansour, and Noam Nisan. Constant depth circuits, fourier transform, and learnability. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 574–579, 1989.

[10] Marvin L. Minsky and Seymour Papert. *Perceptrons*. MIT press, Cambridge, MA, 1988. Expanded Edition. The first edition appeared in 1968.

[11] Ramamohan Paturi and Michael E. Saks. On threshold circuits for parity. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, pages 397–404, 1990.

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

[13] Alexander Schrijver. *Theory of Linear and Integer Programming.* John Wiley and Sons., 1986.

[14] Roman Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*, pages 77–82, 1987.

[15] Erich Stiemke. Über positive Lösungen homogener linearer Gleichungen. *Mathematische Annalen*, 76:340–342, 1915.

[16] Jun Tarui. Randomized polynomials, threshold circuits, and the polynomial hierarchy. Manuscript, August 1990.

[17] L. G. Valiant and V.V. Vazirani. NP is as easy as detecting unique solutions. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing*, 1985.

[18] Andrew Yao. Separating the polynomial-time hierarchy by oracles. In *Proceedings of the 26th Annual IEEE Symposium on Foundations of Computer Science*, pages 1–10, 1985.