Polynomials and Combinatorial Definitions of Languages

Kenneth W. Regan[1]

ABSTRACT Using polynomials to represent languages and Boolean functions has opened up a new vein of mathematical insight into fundamental problems of computational complexity theory. Many notable advances in the past ten years have been obtained by working directly with these polynomials. This chapter surveys important results and open problems in this area, with special attention to low-level circuit classes and to the issues of "strong" vs. "weak" representations raised by Barrington, Smolensky, and others. Other combinatorial representations for languages besides polynomials are worthy of attention, and a new example characterizing parity-of-(ands-of)-threshold circuits is presented in the last section.

1 Introduction

Turing machines and complexity measures are great for *defining* classes of languages, but many researchers are finding that they are not so hot for *analyzing* these classes, especially for lower bounds. As formal tools they mostly stand by themselves; they do not build on or easily link to the great progression of mathematical concepts and tools. Turing machines are unstructured; their work environment is a *tabula rasa*; their computational process is not known to have anything like the overt properties and hooks for analysis of other mathematical processes. Even chaos is structured. These remarks apply to other general machine models, and in large part to Boolean circuits.

Machine-independent characterizations of complexity classes seek to answer these concerns. A prominent main line of research has been "capturing" these classes by systems of first and second-order logic. The chapter by Barrington and Immerman in this volume covers some of this, and some recent successes in lower and upper bounds may be found in [AF90, FSV93, Sch94]. Here we will survey a second line that seeks to characterize languages and complexity notions directly in terms of mathematical entities that have been studied for a long time.

[1]Department of Computer Science, State University of New York at Buffalo, 226 Bell Hall, Buffalo, NY 14260-2000 USA. E-mail: *regan@cs.buffalo.edu*. Supported in part by the National Science Foundation under grant CCR-9409104.

Polynomials over various rings and fields have been the touchstone for many notable advances over the past ten years. The main complexity notion for polynomials is that of *degree*, which in turn is a chief actor in many areas of mathematics, and the results covered here show how well it corresponds to standard complexity measures for the languages and functions represented by the polynomials. Although they had been used as early as 1968 by Minsky and Papert [MP68] for lower bounds on *perceptrons*, polynomials really erupted onto the map with papers by Razborov [Raz87], Smolensky[2] [Smo87], and Toda [Tod89, Tod91]. The point is that in each case, the polynomials not only captured the problems but also ushered in the algebraic techniques that solved the problems. A posse of papers in the next few years [All89, AH90, Sze90, Sze93, Yao90, BT91, BT94, Tar91, ABFR91, ABFR94, BRS91a, BRS91b, BRS95, Bar92, BBR92, BBR94, NS92, NS94, Pat92] codified the "polynomial method" and expanded its significance.

Although polynomials have deservedly gotten the most press and receive most attention here, there are other combinatorial objects that can do the same service of *combining* issues of complexity theory with areas of mathematics where a great many more answers are known. Space allows us only to be suggestive by introducing one notion with a more geometrical flavor that takes past work on hyperplane separations and thresholds one step further, and proving results relating it to low-level classes.

This survey covers much of the same ground as the excellent one by Beigel [Bei93], but with a different set of emphases. First, we focus more sharply on the issue of "strong" versus "weak" representations, and set up a framework for assessing the effect of both the underlying ring or field and the mode of representation on the class of languages defined. Second, we try to bring *probabilistic polynomials* into this framework, continuing the foundations laid by Tarui [Tar93], and show how tradeoffs in the theory of *error-correcting codes* impact on these polynomials. Third, we emphasize applications for the small classes inside NC^1; this also connects the geometrical notion in Section 8 to the whole. Some material is adapted from the author's joint papers [GKR$^+$95] and [NRS95], and some material is new.

2 Polynomials

Multi-variable polynomials are perhaps the simplest familiar mathematical objects that are capable of representing languages. Let R be any set

[2]This author was greatly saddened by the news of Roman Smolensky's passing. This came a week after the first draft of this article was completed. I was not able to convert this into a more in-depth study of Smolensky's papers themselves, but I hope that the coverage of results and attention given to the "problem of representations," which Smolensky highlighted in [Smo93], will do service to his memory and spur interest in the goals toward which he was working.

together with two operations $+, * : R \times R \to R$. Then any arithmetical formula in $+, *$ using variables $u_1, \ldots, u_n$ ($n \geq 0$) and elements of R defines an n-variable polynomial p *over* R, written $p \in R[u_1, \ldots, u_n]$. If $+$ is associative and commutative, has an identity $0 \in R$, and gives every element an additive inverse—and if $*$ is associative and distributes on both left and right over $+$, then $\mathcal{R} = (R, +, *)$ is a *ring*. If $*$ is also commutative and has an identity $1 \in R$, then $\mathcal{R}$ is a *commutative ring with identity*, and further if every non-0 element has a multiplicative inverse, then $\mathcal{R}$ is a *field*. The complex numbers $\mathbf{C}$, the real numbers $\mathbf{R}$, the rational numbers $\mathbf{Q}$, and the integers modulo q (denoted by $\mathbf{Z}_q$) for prime q are fields, but the integers $\mathbf{Z}$, and $\mathbf{Z}_m$ for composite $m \geq 2$, are "merely" commutative rings with identity. For any $k \geq 1$ and prime q, the *Galois field* $\mathrm{GF}(q^k)$ is defined with $R = (\mathbf{Z}_q)^k$ using vector addition and a $*$ operation whose definition does not concern us here; for more on all the above, see [Jac51]. Every finite field is isomorphic to some $\mathrm{GF}(q^k)$. $\mathrm{GF}(q)$ is the same as $\mathbf{Z}_q$, but for $k{\geq}2$, $\mathrm{GF}(q^k)$ should not be confused with $\mathbf{Z}_{q^k}$, which is not a field.

One perhaps counter-intuitive import of current research is that the more properties one adds to $\mathcal{R}$, the *weaker* the power of polynomials over $\mathcal{R}$ to represent languages. Indeed, the most recent fundamental work has been on polynomials (and generalizations of polynomials) defined over structures weaker than rings—see [BT88, BST90, MPT91, Nis91, AJ93, MV94]. However, our reasons for emphasizing rings come out in Section 4. Unless otherwise specified, languages are defined over the alphabet $\{0, 1\}$.

Definition 2.1. Given $\mathcal{R} = (R, +, *)$, let e_0 and e_1 be fixed elements of R, and let S_1 and S_0 be nonempty disjoint subsets of R. A sequence of polynomials $\{p_n : n \geq 1\}$, with each $p_n \in \mathcal{R}[u_1, \ldots, u_n]$, is said to *represent* a language L *with scheme* (e_1, e_0, S_1, S_0) if for all n and $x \in \{0, 1\}^n$,

$$\begin{aligned} x \in L &\implies p_n(x) \in S_1, \\ x \notin L &\implies p_n(x) \in S_0. \end{aligned}$$

Here $p_n(x)$ is defined by substituting, for each i ($1 \leq i \leq n$), e_0 for u_i if x_i (i.e., the ith bit of x) is a 0, and e_1 for u_i if x_i is a 1.

This definition "promises" that for all x, $p_n(x) \in S_0 \cup S_1$. When $S_0 = R \setminus S_1$, no promise is needed, and every sequence $\{p_n\}$ represents a unique language. Given e_1 and e_0, the negation of a Boolean variable u_i is expressed by $(e_1 + e_0 - u_i)$. By analogy with a *term* in a DNF Boolean formula, we call a product of factors of the form u_i or $(e_1 + e_0 - u_i)$ a *schematic term*.

The following "complexity measures" for polynomials spring to mind.

(1) *Degree:* $\deg_p(n) =$ the degree of p_n.

(2) *Size:* Here there are three main notions:

(2a) *Number of monomials:* $m_p(n) =$ the number of monomials when p_n is "multiplied out" via the distributive law.

(2b) *Number of schematic terms:* $s_p(n)$ = the minimum number of schematic terms needed to write p_n as a sum of schematic terms.

(2c) *Formula size:* $F_p(n)$ = the minimum number of $+, *$ operands in a formula for p_n.

(3) *Coefficient Size:* $C_p(n)$ = the maximum number of bits in a coefficient of a monomial of p_n.

The coefficient size comes into play for the infinite rings. Combining it with the formula size, we have a measure of the number of bits required to write p_n down. Computing the other complexity measures besides the degree, given any formula for p_n, can present difficulties. Counting the monomials and estimating coefficient size will be straightforward since our given formulas will not have tricky cancellations, but minimum number-of-schematic-terms and minimum formula size are NP-hard even in seemingly favorable cases, such as where p_n takes only polynomially many nonzero values and all of them are given. (See the corresponding problems about Boolean formulas, called MINIMUM EQUIVALENT EXPRESSION and MINIMUM DISJUNCTIVE NORMAL FORM, in [GJ79].)

In order to focus on these complexity measures as properties of the languages or functions themselves, with regard to various rings, we first try to minimize the dependence on representation scheme.

3 Representation Schemes and Language Classes

Nearly all the results in our references use one of the following six representation schemes. Sign outputs are not applicable for the finite rings. The names of (1) and (2) are adapted from Beigel's survey [Bei93], while the other four are based on nomenclature in [Bar92, Smo93, BBR94].

Definition 3.1. *Chief representation schemes for $\mathcal{R} = (R, +, *)$:*

(1) *Standard input, sign output:* $e_0 = 0$, $e_1 = 1$, $S_1 = \{ r \in R : r > 0 \}$, and $S_0 = \{ r \in R : r < 0 \}$.

(2) *Fourier input, sign output:* $e_0 = +1$, $e_1 = -1$, S_1 and S_0 as before.

(3) *Strong representation:* $e_0 = 0$, $e_1 = 1$, $S_1 = \{ 1 \}$, and $S_0 = \{ 0 \}$.

(4) *Standard nonzero representation:* $e_0 = 0$, $e_1 = 1$, and $S_1 = \{ r \in R : r \neq 0 \}$ (so zero stands for $x \notin L$, everything else for acceptance).

(5) *Weak representation:* $e_0 = 0$, $e_1 = 1$, and $S_1 = \{ 0 \}$ (equivalently, $S_1 = \{ a \}$, for any fixed $a \in R$). This is complementary to (4).

(6) *Truly weak representation:* $e_0 = 0$, $e_1 = 1$, S_1 is arbitrary and may differ for different n, and $S_0 = R \setminus S_1$.

With standard inputs as in (1), multiplication corresponds to logical AND, while with Fourier inputs as in (2), multiplication carries out XOR. Fourier inputs can also be used in place of standard inputs in (3)–(6). A major point of both these input schemes is that $x^2 = x$ holds in the former, $x^2 = 1$ in the latter. Hence the only polynomials we need to consider are *multilinear*, and the maximum degree involved is n.

The promise $p_n \neq 0$ in (1) and (2) is not important—one can meet it from the case $S_0 = R \setminus S_1$ by forming $2p_n(x) - 1$. It also makes no difference if we let a negative sign stand for true, positive for false. Hence (3) is the only one with a real promise condition, justifying Barrington's name "strong representation" for it. Taking $a = 1$ in (5) makes it clear that *all* of the other output schemes are met by polynomials obeying (3). Smolensky [Smo93] identifies (4) with Barrington's (5), but we prefer to think of (4) as loosely analogous to "NP," (5) to "coNP," and (3) to "P." Truly weak representation is equivalent to saying that we have polynomials p_n such that for all $x, y \in \{0,1\}^n$ with $x \in L$ and $y \notin L$, $p_n(x) \neq p_n(y)$. Note that no distinction between L and its complement is made in this condition.

Definition 3.2. Two representation schemes over a ring $\mathcal{R}$ are *equivalent* if for every $\{p_n\}$ representing a language L using one scheme, there exist polynomials $\{q_n\}$ that represent L using the other scheme, such that $\deg_q(n) = O(\deg_p(n))$, $F_q(n) = O(F_p(n))$, and $C_q(n) = O(nC_p(n))$.

The condition on $C_q(n)$ is just strong enough to preserve polynomial coefficient size. Now we observe that all representation schemes over finite fields are equivalent to (3), and we use the basic idea to reduce the other cases as much as possible. We need the following technical provision, which holds in many cases.

Definition 3.3. Given disjoint $S_1, S_0 \subseteq \mathcal{R}$ and disjoint $T_1, T_0 \subseteq \mathcal{R}$, say that (S_1, S_0) is *polynomially mappable* to (T_1, T_0) if there is a polynomial g in one variable over $\mathcal{R}$ such that $g(S_1) \subseteq T_1$ and $g(S_0) \subseteq T_0$. Call (S_1, S_0) and (T_1, T_0) *inter-mappable* if (T_1, T_0) is likewise mappable into (S_1, S_0).

Now suppose we want to convert a polynomial p over $\mathcal{R}$ with scheme (a_1, a_0, S_1, S_0) into a polynomial q that represents the same language with scheme (b_1, b_0, T_1, T_0), where we are given g mapping (S_1, S_0) into (T_1, T_0). If $b_1 - b_0$ has an inverse in $\mathcal{R}$, then we can use the linear formula

$$q(\vec{x}) = g(p(\frac{(a_1 - a_0)\vec{x} + a_0 b_1 - a_1 b_0}{b_1 - b_0})). \qquad (1.1)$$

To verify: if a variable x_i of q is assigned b_0, then the corresponding variable of p gets the value $((a_1 - a_0)b_0 + a_0 b_1 - a_1 b_0)/(b_1 - b_0) = a_0(b_1 - b_0)/(b_1 - b_0) = a_0$, and similarly an assignment of b_1 to an argument of q puts a_1 into the corresponding argument for the evaluation of p. This leads to a nice "robustness" theorem for fields, especially finite fields.

Proposition 3.1 *(a) Every two inter-mappable representation schemes over a field are equivalent.*

(b) All representation schemes over a finite field F are equivalent to strong representation; i.e., to (3) above.

Proof. Part (a) follows by Equation (1.1) and gives $\deg(q) \leq \deg(g)\deg(p)$. The formula size of $p(\frac{\cdots}{\cdots})$ is at most 7 times the formula size of p itself, and the composition into g gives at most another constant-factor overhead. For part (b), let a scheme (e_1, e_0, S_1, S_0) and a polynomial p be given. It suffices to find a univariate polynomial g such that $g(r) = 1$ for $r \in S_1$ and $g(r) = 0$ otherwise. That is done by

$$g(r) = 1 - [\prod_{s \in S_1}(r - s)]^{\|F\|-1},$$

since every non-zero element raised to the power of $\|F\| - 1$ gives 1. Thus we have

$$q(\vec{x}) = g(p((e_1 - e_0)\vec{x} + e_0)).$$

This yields a strong representation, and $\deg(q) \leq (\|F\| - 1)^2 \deg p$. To go from this to any other scheme (b_1, b_0, T_1, T_0), fix any $a \in T_0$ and $b \in T_1$, and define

$$q'(\vec{x}) = (b - a)q(\frac{x - b_0}{b_1 - b_0}) + a.$$

□

Now we observe that the construction in (a) using Equation (1.1) works not only in fields, but also in many other cases. It works:

- When $b_1 - b_0$ has an inverse in $\mathcal{R}$—for instance, in $\mathbf{Z}_m$ when $b_1 - b_0$ is relatively prime to m.

- When the function g can be multiplied by arbitrary powers of $b_1 - b_0$ and still map S into T and the complement of S into the complement of T. Multiplying by $(b_1 - b_0)^{\deg(p)}$ cancels all denominators in monomials of $p(\frac{\cdots}{b_1 - b_0})$.

Corollary 3.2 *(a) For sign output, all representations for the inputs are equivalent for polynomials over $\mathbf{Z}$, as well as the fields $\mathbf{Q}$ and $\mathbf{R}$. So (1) is equivalent to (2).*

(b) Fourier inputs are equivalent to standard ones for polynomials over $\mathbf{Z}_m$ when m is odd, in each of (3)–(6).

(c) When S_0 and S_1 are fixed for outputs, lower bounds on degree proved for the standard input representation apply to all other input representations.

Proof. (a) If $(b_1 - b_0)$ is positive, then $S = \{ x \in \mathbf{Z} : x > 0 \}$ is preserved under powers of $(b_1 - b_0)$, as is its complement. If $(b_1 - b_0)$ is negative, then multiply by $(b_0 - b_1)$ instead. The coefficient size stays within the bounds allowed by Definition 3.2. Part (b) holds because 2 is relatively prime to m when m is odd. For (c), any polynomials p_n representing L with a scheme (a_1, a_0, S) can be converted to $(1, 0, S)$ because then $b_1 - b_0 = 1$. $\square$

Note that we left the term-counting and monomial-counting measures out of the definition of equivalence. The above results do not preserve the latter—they can blow up to exponentially many monomials. We do not know what happens in general for schematic terms. However, Equation (1.1) does preserve the ability to wire Boolean inputs into small circuit gadgets that give the corresponding values in the ring, so that wherever "number of terms" is used in the following results, robustness does hold. Several authors use "terms" as synonymous with monomials or leave the meaning vague; we pin it down to "schematic terms" if need be. Call $\{ p_n \}$ *sparse* if the p_n can be written with polynomially many schematic terms.

To describe various kinds of *circuits* and circuit classes, we adopt and adapt the notations of Goldmann, Håstad, and Razborov [GHR92] and Maciel and Thérien [MT93, Mac95] as follows: A *stratified circuit* of *depth* d has inputs labeled $x_1, \ldots, x_n$ together with their negations $\bar{x}_1, \ldots, \bar{x}_n$, and then has d *levels*. Gates at each level receive inputs from the previous level (the inputs are level 0), and all gates at the same level have the same type. The gate types we consider are:

- AND gates (A) and OR gates (O), of unbounded fan-in;
- *"Small"* AND gates (AND_{small}), defined to have fan-in $(\log n)^{O(1)}$;
- *Mod_k gates* (Mod_k), standardly defined to output *true* iff the number of *true* inputs is zero modulo k;
- *Parity* gates (P) or (Parity), which are the same as Mod_2 gates;
- *Large Threshold* gates (LT), each of which has a threshold t and integer weights w_i associated to its 0-1 valued input lines e_i, and outputs *true* iff $\sum_i w_i e_i > t$.
- *Small Threshold* gates (T) have $t, w_i = r^{O(1)}$, where r is the fan-in.
- *Majority* gates (MAJ) have all $w_i = 1$ and $t = r/2$. We also include the negation of a MAJ gate under this heading.
- *Midbit gates* (Midbit): a Midbit gate of fan-in r returns the $\lceil \log_2 r \rceil$th bit of the number m of *true* inputs, where m is in binary notation.
- *General symmetric gates* (SYM) are any gates whose output depends only on the number of input lines that are *true*. This designation includes all of the above except T and LT gates.

The major classes defined by polynomial-size, constant-depth circuit families are AC^0, where the circuits have unbounded fan-in AND, OR, and NOT gates, ACC^0, where they may also have Mod_k gates (with k fixed for the family), and TC^0, where they may instead have LT gates. Since an LT gate can be simulated by a depth-two, n^{13}-sized gadget of MAJ gates [Hof96] (see also [GHR92]), TC^0 can also be defined via T gates or MAJ gates. Also for each $k \geq 1$, NC^k denotes the class of languages accepted by *bounded* fan-in Boolean circuit families of polynomial size and $O(\log^k n)$ depth, and $\mathrm{NC} = \cup_k \mathrm{NC}^k$. We skirt issues of *uniformity* for these classes (see the chapter by Barrington and Immerman in this volume), since uniformity goes to the background in what follows. The known inclusions are

$$\mathrm{AC}^0 \subset \mathrm{ACC}^0 \subseteq \mathrm{TC}^0 \subseteq \mathrm{NC}^1 \subseteq \mathrm{NC}^2 \subseteq \ldots \subseteq \mathrm{NC} \subseteq \mathrm{P}.$$

Only the first inclusion is known to be proper [FSS84], and even $\mathrm{ACC}^0 \neq \mathrm{NP}$ is unknown!

The stratified-circuit notation allows us to define more-refined classes than the above. For example, $\mathrm{MAJ} \circ \mathrm{A}$ stands for polynomial-sized circuits consisting of a MAJ gate connected to one layer of AND gates at the inputs, and $\mathrm{LT} \circ \mathrm{A}$ for circuits that may have a large threshold gate at the output instead. Both $\mathrm{MAJ} \circ \mathrm{A}$ and $\mathrm{LT} \circ \mathrm{A}$ denote proper subclasses of TC^0 (see [GHR92, Mac95]). Now we can relate the circuit classes to polynomials.

Theorem 3.3 (cf. [Bei93]) *(a) Let L be represented by polynomials p_n over* $\mathbf{Z}$*,* $\mathbf{Q}$*, or* $\mathbf{R}$ *having polynomial formula and coefficient size, using sign outputs. Then $L \in \mathrm{NC}^2$.*

(b) If the p_n are sparse, then $L \in \mathrm{LT} \circ \mathrm{A}$. Conversely, every language in $\mathrm{LT} \circ \mathrm{A}$ has sparse polynomials over $\mathbf{Z}$ *and* $\mathbf{R}$*, using standard inputs.*

(c) If standard inputs are used and the coefficients of the sparse p_n have polynomial magnitude (that is, have $O(\log n)$ bits), then $L \in \mathrm{MAJ} \circ \mathrm{A}$. Conversely, every language in $\mathrm{MAJ} \circ \mathrm{A}$ has sparse polynomials over $\mathbf{Z}$ *with all (non-zero) coefficients equal to 1.*

(d) If L is represented by p_n over a finite ring $\mathcal{R}$, and the p_n have polynomial formula size, then $L \in \mathrm{NC}^1$. Moreover, every $L \in \mathrm{NC}^1$ is represented by polynomials over $GF(2)$ having polynomial formula size.

(e) In (d), if the p_n are sparse, then $L \in \mathrm{ACC}^0$; also, L has $SYM \circ \mathrm{AND}_{small}$ circuits of size $2^{\mathrm{polylog}(n)}$, where again, AND_{small} means that the AND gates have $\mathrm{polylog}(n)$ fan-in.

Proof Sketch. The main point of (a) is the fact that polynomial-sized arithmetical formulas can be effectively "rebalanced" into arithmetical formulas of polynomial size and log depth (see [Spi71, Bre74, MP92]). Since the inputs e_0 and e_1 are fixed for all p_n, all intermediate values have polynomially many bits, and since each arithmetical operation is in (Boolean)

NC^1, the whole is in NC^2. In (b), the coefficient on each monomial becomes a weight on a line into a threshold gate, with $t = 1$. For (c) one can duplicate gates below the inputs and add dummy lines. The first part of (d) is clear by the reasoning in (a), and the converse follows by standard "arithmetizing" of Boolean formulas. The first part of (e) is immediate over $\mathbf{Z}_m$, and coding tricks extend it to other finite rings. The second part is due to Beigel and Tarui [BT91], and is bundled into Theorem 6.1 below. □

Cases (c) and (e) correspond to "Theorem 2" in [Bei93]. In (e), if the ring is $\mathbf{Z}_m$ and weak representation (5) in Definition 3.1 is used with $a = 0$, then the output gate becomes a Mod_k gate.

Curiously, these basic relationships with circuit classes say nothing by themselves about the *degree* measure. Degree corresponds to the *order* of a *perceptron*, as formalized and studied by Minsky and Papert [MP68]. The equivalence of perceptrons to polynomials with bounded coefficients (and with the number of monomials plus one equal to the *size* of the perceptron) is shown by Beigel [Bei93] and treated further in [Bei94a]. One remark is that an order-d perceptron of order d, size s, and weights of magnitude w can be converted into an order-d perceptron of size $2^d s$ and weight sw that has no negated inputs and no duplicate AND gates (see [Bei94b, MP68]); this corresponds to the obvious relationship between number of schematic terms and number of monomials. We do not discuss perceptrons further here. The impact of having low-degree polynomials comes out in other simulations described below. In contrast to the lack of good lower bounds for familiar machine-based complexity measures, the degree measure lends itself to tight lower and upper bounds in a number of important cases.

4 Strong Versus Weak Representation

First, we note that to every Boolean function $f(x_1, \ldots, x_n)$ we can associate a canonical polynomial σ_f, such that σ_f represents f over *any* ring $\mathcal{R}$ under strong representation. For every assignment $\vec{a} = (a_1, \ldots, a_n)$ in $\{0, 1\}^n$, let $M_{\vec{a}}(\vec{x}) = \prod_i (2a_i x_i - x_i - a_i + 1)$. This is zero except when $\vec{x} = \vec{a}$, when it is 1. Then let σ_f be the sum of $M_{\vec{a}}$ over all $\vec{a}$ such that $f(\vec{a}) = \mathit{true}$. As explained by Tarui [Tar91], because $\mathcal{R}$ is a ring and not a weaker structure, the $\mathcal{R}$-*module* $\mathcal{F}_n(\mathcal{R})$ of functions from $\{0, 1\}^n$ to $\mathcal{R}$ behaves much like a 2^n-dimensional vector space—even if $\mathcal{R}$ is not a field. In particular, the 2^n multilinear monomials form a basis for this space, so every function in $\mathcal{F}_n(\mathcal{R})$ can be written uniquely as a linear combination (with coefficients in $\mathcal{R}$) of these monomials. (Since 0 and 1 commute with every element of a ring, we do not even need $\mathcal{R}$ to be a commutative ring, and the above features hold also for Fourier inputs.) Hence σ_f is the unique strong representation of f. If we know the degree and size measures of σ_f, that's it—no strong representation can do better.

Now define Z_f to be the set of polynomials that compute f (over a given $\mathcal{R}$) under the standard nonzero representation. Proposition 3.1(b) now says that over a finite field $\mathcal{F}$, all members of Z_f have degree within a factor of $|\mathcal{F}| - 1$ of that of σ_f. Over $\mathbf{Z}_m$ with m composite, however, there can be drastic differences. Barrington [Bar92] gives this example with $m = 6$:

$$L = (0^*(10^*)^6)^*.$$

L is weakly represented over $\mathbf{Z}_6$ by the degree-one polynomials $p_n(\vec{x}) = x_1 + \ldots + x_n$. However, the unique strong representations have degree n.

The differences emerge even for the basic AND and OR functions. With standard inputs and sign output, the languages 1^* and $0^*1(0+1)^*$, standing for arbitrary fan-in AND and OR respectively, are represented by linear polynomials over $\mathbf{Z}$ and the other infinite rings; viz., OR by $x_1 + \cdots + x_n$ and AND by $x_1 + \cdots + x_n - (n-1)$. For $\mathbf{Z}_m$ the known bounds are different.

Theorem 4.1 *For polynomials over $\mathbf{Z}_m$, $m \geq 2$:*

(a) **[Tar91, BST90]** *(Beigel* **[Bei93]** *adds "folklore") Under strong representation, AND and OR require degree n.*

(b) **[BBR94]** *Under the standard nonzero representation, AND still requires degree n, but OR is representable in degree $O(n^{1/k})$, where k is the number of distinct prime factors of m. The best known lower bound on degree for OR is $\Omega(\log^{1/(k-1)} n)$* **[TB95]**.

(c) **[Smo87, BST90, BBR94]** *If m is a prime power (so $k = 1$), OR is representable in degree $\lceil n/m - 1 \rceil$, and this is best possible.*

(d) Under weak representation, (b) and (c) hold with the roles of AND and OR reversed. In particular, there is no degree-preserving simulation between the standard nonzero representation and its complement.

Proof. (a) We have $\sigma_{AND}(u_1, \ldots, u_n) = u_1 \cdots u_n$ and $\sigma_{OR} = 1 - (1 - u_1) \cdots (1 - u_n)$. Those are the unique strong representations, and each has degree n. Now in (b), any standard representation p of AND maps all of $\{0, 1\}^n \setminus 1^n$ to 0. Hence the value $p(1^n) = a$ determines the whole function—it is the monomial $ax_1 \cdots x_n$. Each of these has degree n, and by the reasoning for σ_f above, there are no others. For the other part of (b), see [Bei93] or [BBR94].

(c) For the upper bound, let $d = \lceil n/m - 1 \rceil$, and use

$$g(u) = (u_1 \cdots u_d) + (u_{d+1} \cdots u_{2d}) + \ldots + (u_{(m-2)d} \cdots u_n).$$

Then g has $m - 1$ monomials, each of degree at most d, and $g(\vec{x}) \neq 0 \iff \mathrm{OR}(x_1, \ldots, x_n)$. AND under the complementary weak representation is treated dually. For the lower bound, note that the conversion to strong representation works since $\mathbf{Z}_m$ is a field and multiplies the degree by $(m-1)$. By (a), the degree here cannot be lower than d. Part (d) follows from the definitions. $\square$

The polynomials constructed in [BBR94] to achieve the upper bound for OR in (b) are symmetric, and a matching lower bound for symmetric polynomials representing OR is proved in [BBR94]. We will see that the degree picture for AND and OR improves considerably when we go to *probabilistic* polynomials, even under strong representation. First we examine bounds for some other functions and languages.

5 Known Upper and Lower Bounds on Degree

The following results are taken from Beigel's survey [Bei93], where full proofs may be found. By the robustness results and usages established in the last section, we can be fairly brief in stating the hypotheses.

Theorem 5.1 ([MP68]) *The parity language* $0^*1(0^*10^*1)^*0^*$ *requires degree* n *over* $\mathbf{Z}$, $\mathbf{Q}$, *and* $\mathbf{R}$.

Note that the parity function $x_1 + x_2 + \cdots + x_n \pmod 2$ is a degree-one polynomial over GF(2). Over $\mathbf{Z}_m$ with $m = 2k$ one can use $kx_1 + \cdots + kx_n$ to get a degree-one representation with $S_0 = \{0\}$ and $S_1 = \{k\}$. The case of odd m is different.

Theorem 5.2 ([Smo87]) *Parity requires degree* $\Omega(n^{1/2})$ *over* $\mathbf{Z}_m$ *for any odd* $m \geq 3$.

Now, following [BBR94], define $Mod_k(x_1, \ldots, x_n)$ to be *false* if $x_1 + \cdots + x_n \equiv 0 \pmod k$, and *true* otherwise. Write $\delta(f, m)$ for the minimum degree of a standard nonzero representation of f over $\mathbf{Z}_m$, and $\Delta(f, m)$ for that of a "truly weak" representation. Recall that the minimum degree of f under weak representation (i.e., with $S_1 = \{0\}$) is the same as $\delta(\neg f, m)$.

Theorem 5.3 *(a)* **[Smo87]** *When* $m = p$ *is prime and* k *is not a power of* p, $\delta(Mod_k, p) = \Omega(n)$.

(b) **[BBR94]** *If* k *has a prime divisor that is not a divisor of* m, *then* $\delta(Mod_k, m) = n^{\Omega(1)}$ *and also* $\delta(\neg Mod_k, m) = n^{\Omega(1)}$.

(c) (see **[BBR94]***) If the set of prime divisors of* k *is contained in that of* m, *then* $\delta(Mod_k, m) = O(1)$ *and* $\delta(\neg Mod_k(m)) = O(1)$.

(d) **[Tsa93]** *If* m *is not a prime power, then* $\delta(\neg Mod_m, m) = \Omega(n)$.

(e) **[Tsa93]** *If m is not a prime power, and k has a prime divisor that does not divide m, then $\delta(Mod_k, m)$ and $\delta(\neg Mod_k, m)$ are both $\Omega(n)$.*

The results by Tsai [Tsa93] improved $n^{\Omega(1)}$ bounds in [BBR94] in the case where m is not square-free. Green [Gre95] improved the results of [BBR94, Tsa93] further by showing that under standard nonzero representation, for all k there is a constant C_k such that for all m that are relatively prime to k, $\delta(Mod_k, m) \geq C_k n$. That is, the constant in "$\delta(Mod_k, m) = \Omega(n)$" is independent of m so long as the modulus m is prime to k. This holds even if Definition 3.1(4) is made weaker by requiring only that the polynomial p is not identically zero but gives zero whenever $x \notin L$ (i.e., the Boolean function concerned, here Mod_k, is false). However, none of these bounds are known at all for the degrees $\Delta(Mod_k, m)$ under "truly weak" representation. Tsai also proved the following theorem.

Theorem 5.4 ([Tsa93]) *For any integer $m \geq 2$:*

(a) $\delta(MAJ, m) \geq n/2$.

(b) $\delta(Midbit, m) = \Omega(n^{1/2})$.

Some functions that (unlike parity and Mod_k) do belong to uniform AC^0 also require more than polylog degree over the infinite rings.

Theorem 5.5 ([MP68]) *Over **Z**, **Q**, and **R**, the Boolean function f defined by $f(x_0, \ldots, x_{4m^3-1}) = (\forall i \in [0 \ldots m-1])(\exists j \in [0 \ldots 4m^2-1])x_{i+j}$ requires degree m. Hence with $n = 4m^3$, the degree is $\Omega(n^{1/3})$.*

For representation by polynomials over **R**, it is most common to use $S_1 = \{ x \in \mathbf{R} : |x-1| \leq 1/3 \}$, and define S_0 similarly around 0. Then OR and AND cannot be done with degree $o(\sqrt{n})$ (see [Bei93]), and Paturi [Pat92] showed that the MAJ function requires degree $\Omega(n)$. Nisan and Szegedy [NS94] showed that the degree of this representation is polynomially related to that of strong representation. Namely, for every Boolean function f, every polynomial representing f in this scheme has degree at least $c(\deg(\sigma_f))^{1/8}$, where the constant $c > 0$ is independent of f. In fact, they showed that both measures are polynomially related to the decision-tree complexity of f. Similar techniques were used by Beigel [Bei94a] to show that the language

$$L = (00 + 01 + 10 + 11)^* 10^*$$

(called ODDMAXBIT in [Bei93]), which is represented over **Z** by the degree-one polynomial $\sum_{i=1}^{n}(-2)^i x_n$ with linear-sized coefficients, cannot be represented over **Z** or **Q** or **R** by low-degree polynomials with small coefficients. The exact result is that it cannot be done in degree $n^{o(1)}$ with coefficient size $n^{o(1)}$ (and $2^{n^{o(1)}}$ monomials). In particular, this language is not recognizable by perceptrons of polylog order, sub-exponential weight, and quasipolynomial size (i.e., size $2^{\text{polylog}(n)}$).

Several of the lower bounds show that all polynomials of a given low degree $d(n)$ fail to represent a given Boolean function on a large portion of inputs $x \in \{0,1\}^n$, such as a constant fraction of them. The next theorem gives an example.

Theorem 5.6 ([ABFR94]) *For all d, n, and $m \leq 2^n$, there exists a degree-d polynomial p over $\mathbf{Z}$ whose sign represents $Parity(x)$ for m-many $x \in \{0,1\}^n$, iff $m \leq \sum_{0 \leq k < (n+d+1)/2} \binom{n}{k}$.*

In particular, to compute parity correctly on $1/2 + \epsilon$ of the inputs, for fixed $\epsilon > 0$, one needs degree $\Omega(\sqrt{n})$. Now define L_k ($k \geq 2$) to be the language of strings x formed by catenating some number m of "blocks" of the form $0^{r_i}10^{k-r_i}$, such that $\sum_{i=1}^{m} r_i \neq 0$ modulo k. Using polynomials over $\mathbf{C}$ in an auxiliary role, Barrington and Straubing [BS94] obtained the following theorem.

Theorem 5.7 ([BS94]) *There exists δ depending on k such that polynomials representing L_k (by sign over $\mathbf{Z}$) on a $1-\delta$ proportion of inputs require degree $\Omega(\sqrt{n})$.*

The most basic non-approximability results stem from the following "folklore" lemma, versions of which may be found in [Bar92] and [Smo93].

Lemma 5.8 *Every polynomial $p(x_1, \ldots, x_n)$ of degree at most d over a field $\mathcal{F}$ is either constant, or takes value 0 on at least $2^{n-d(|\mathcal{F}|-1)}$ Boolean (i.e., 0-1) arguments.*

In consequence, a degree-d polynomial over $\mathbf{Z}_2$ must disagree with OR on at least $2^{n-d} - 1$ arguments, and straightforward constructions show that this bound is tight. Barrington [Bar92] proved a generalization.

Theorem 5.9 ([Bar92]) *Let p have degree d and take at most r distinct values in a field $\mathcal{F}$. Then p has value 0 on at least $2^{n-d(r-1)}$ 0-1 arguments.*

For arbitrary rings $\mathcal{R}$ in place of $\mathcal{F}$, Barrington proved that the statement of Theorem 5.9 holds if $d = 1$ or $r = 2$, and that the weaker Lemma 5.8 holds for all d in $\mathbf{Z}_{p^k}$, for any prime p and all k [Bar92]. However, an example credited to Applegate, Aspnes, and Rudich in [Bar92] shows that the statement fails for $\mathcal{R} = \mathbf{Z}_6$ with $d = 3$ and $n = 27$: Let

$$p(\vec{x}) = s_3(\vec{x}) + 5s_2(\vec{x}) + 3s_1(\vec{x}),$$

where s_i stands for the mod-6 sum of all monomials of degree i. This polynomial is a standard nonzero representation of OR in $\mathbf{Z}_6$, and meets the prescribed bounds from Theorem 4.1(b). For a full explanation of the failure, see [BBR94].

Smolensky [Smo93] used *Hilbert functions* to prove several other non-approximability results in fields of finite characteristic.

Theorem 5.10 ([Smo93]) *Using asymptotic notation that depends only on the characteristic c and not on the size of a field $\mathcal{F}$, and using standard non-zero representation:*

(a) Every polynomial of degree $o(n^{1/2})$ differs from MAJ on at least $2^{n-1} - o(2^n)$ Boolean arguments.

(b) If $c \neq 2$, then every polynomial of degree $o(n^{1/2})$ differs from Parity on at least $2^{n-1} - o(2^n)$ Boolean arguments.

(c) If q is prime and $c \neq q$, then every polynomial of degree $o(n^{1/2})$ differs from $\neg Mod_q$ on at least $(1/q)2^n - o(2^n)$ Boolean arguments.

6 Polynomials For Closure Properties

Polynomials have also been used to prove relationships among complexity classes. Instead of n variables standing for bits in an input string, the polynomials used here may have just one or two variables standing for numerical quantities used in defining the classes. The first striking application of this kind was given by Toda [Tod91] in proving that the polynomial hierarchy is contained in $\mathrm{P}^{\#\mathrm{P}}$. He constructed single-variable polynomials P_d over $\mathbf{Z}$ that have the following *modulus-amplifying* property for all integers $k \geq 1$ and $x \geq 0$:

$$x \equiv 0 \pmod{k} \implies P_d(x) \equiv 0 \pmod{k^d}, \tag{1.2}$$

$$x \equiv -1 \pmod{k} \implies P_d(x) \equiv -1 \pmod{k^d}. \tag{1.3}$$

Toda used $P_2(x) = 3x^4 + 4x^3$ and inductively defined $P_{2d}(x) = P_2(P_d(x))$ for $d \geq 2$, using only moduli a power of 2. Yao [Yao90] improved the degree and showed that ACC^0 circuits can be simulated by probabilistic SYM∘AND circuits of quasipolynomial (i.e., $2^{\mathrm{polylog}(n)}$ size, where the ANDs have polylog fan-in. Beigel and Tarui [BT91] made Yao's circuits deterministic without increasing their size, and constructed the following polynomials P_d of optimal degree $2d-1$:

$$P_d(x) = 1 - (1-x)^d\left(\sum_{j=0}^{d-1}\binom{d+j-1}{j}x^j\right). \tag{1.4}$$

(These satisfy $x \equiv +1 \pmod{k} \implies P_d(x) \equiv +1 \pmod{k^d}$ in place of (1.3), but it is easy to convert between these conditions, and this is the one we use below.) Finally, Green, Köbler, and Torán [GKT92], following on from observations about Toda's theorem in [RS92], replaced the arbitrary SYM gate in these results by a Midbit gate, and obtained the following theorem.

Theorem 6.1 ([GKT92, GKR$^+$95]) *Every language in Midbit* $\circ$ ACC^0 *has Midbit* $\circ$ AND$_{small}$ *circuits of quasipolynomial size.*

Proof Sketch. Let $L \in$ *Midbit* $\circ$ ACC^0. The first idea is that all AND and OR gates in the ACC^0 part of the circuits defining L can be replaced by probabilistic $Mod_m \circ$ AND$_{small}$ sub-circuits, where as before, AND$_{small}$ means one level of AND gates of polylog fan-in. Since only polylog-many random bits are needed to do this (see the next section), this part can be simulated by taking a sum of quasipolynomially-many deterministic $Mod_m \circ$AND$_{small}$ circuits. By a lemma in [AH90, BT91], all the small ANDs can be interchanged with Mod_m gates below them and pushed into one layer of small ANDs at the inputs. Then the "pure ACC^0" part of the circuits between the Midbit-of-sum and the small ANDs can be written in stratified form where each level uses Mod_k gates for some prime k [BT91]. It remains to show that the Midbit gate can "swallow up" a sum of Mod_k circuits, yielding a Midbit-of-small-ANDs. Pushing the small ANDs beyond the next layer of Mod_k gates (generally a different k) toward the inputs (as before) leaves a *Midbit*-of-sum-of-Mod_k again, and the process is repeated until all the Mod_k gates are gone. We give the key lemma for the *Midbit*-of-sum-of-Mod_k part in full since its proof shows the use of the Toda polynomials.

Lemma 6.2 *Let k be prime and let $\{ b_n \}$ be a family of functions such that there exists a polynomial r where for each n, b_n is of the form*

$$b_n(x_1, ..., x_n) = \sum_{i=1}^{w} c_i(x_1, ..., x_n),$$

where each c_i is a $Mod_k \circ$ AND$_{small}$ circuit and $w \leq 2^{r(\log n)}$. Then for any polynomial t there are polynomials p and q and a family of polynomials $\{ h_n \}$ of degree $p(\log n)$ such that for each n,

$$b_n(x_1, ..., x_n) \equiv (h_n(x_1, ..., x_n) \textbf{ div } 2^{q(\log n)}) \pmod{2^{t(\log n)}}.$$

Proof. To simplify notation, let p, p', q, r, s, and t denote $p(\log n)$, $p'(\log n)$, $q(\log n)$, $r(\log n)$, $s(\log n)$, and $t(\log n)$, respectively. Each $Mod_k \circ$AND$_{small}$ circuit c_i outputs 1 if and only if a certain sum σ_i of the AND-gates is nonzero mod k. Now each σ_i can be regarded as a polynomial in variables $(x_1, \ldots, x_n)$ over $\mathbf{Z}_k$ of degree equal to the fan-in of the small ANDs, and since k is prime, we may arrange via Lemma 3.1 that $\sigma_i(\vec{x})$ is always 0 or 1 (mod k). Now using the "Toda polynomials" P_d in (1.4) above, it follows that

$$b_n(x) = \sum_{i=1}^{w} \left[P_d(\sigma_i) \bmod k^d \right].$$

We choose $d = p'(\log n)$ where p' is a polynomial such that $k^{p'} > 2^{r+t+2}$. Then $b_n(x) \leq 2^r < k^{p'}$. Now the outer sum in the equation above for b_n is less than $k^{p'}$, so the "mod" can be moved outside; i.e.,

$$b_n(x) \equiv \left[\sum_{i=1}^{w} P_{p'}(\sigma_i)\right] \pmod{k^{p'}}.$$

Writing $f_n(x) = \sum_{i=1}^{w} P_{p'}(\sigma_i)$, we have

$$f_n(x) = a_n(x)k^{p'} + b_n(x)$$

for some $a_n(x)$. Note that for some polynomial s, $f_n(x) < 2^s$. Also note that since σ_i is a polynomial of polylog degree, there is some polynomial p such that f_n is a polynomial of degree $p(\log n)$ in the variables $x_1, ..., x_n$. Define the degree $p(\log n)$ polynomial h_n as follows:

$$h_n(x) = i(n)\left\lceil 2^q/k^{p'}\right\rceil f_n(x) + 2^q f_n(x),$$

where $i(n) \equiv -k^{p'} \pmod{2^t}$ and q is a polynomial such that $q \geq s+t+2$. Then $\lceil 2^q/k^{p'}\rceil f_n(x) = a_n(x)2^q + b'_n(x)$, where $b'_n(x) < 2^{q-t-1}$. Hence

$$h_n(x) \equiv 2^q b_n(x) + i(n)b'_n(x) \pmod{2^{q+t}},$$

where $i(n)b'_n(x) < 2^{q-1}$. This completes the proof of Lemma 6.2 and the sketch of Theorem 6.1. □

The class MP (also called MidbitP) introduced in [RS92, GKR$^+$95] was motivated to find the sharpest upper bound for the polynomial hierarchy in Toda's theorem. A language L belongs to MP if there exists a polynomial-time NTM N such that for all strings x, $x \in L \iff$ the middle bit of the standard binary representation of $\#acc_N(x)$ is a "1." Here $\#acc_N(x)$ stands for the number of accepting computations of N on input x, while $\mathrm{Gap}_N(x)$ (see [FFK91]) stands for $\#acc_N(x)$ minus the number of non-accepting computations. A useful equivalent definition of MP is obtained by combining observations in [GKR$^+$95] and [FFL93]. Say that an integer r is "top modulo 2^k" if ($r \bmod 2^k$) belongs to $[2^{k-1} \dots 2^k - 1]$. Then $L \in \mathrm{MP}$ iff there exist N and a polynomial-time computable function g such that for all x,

$$x \in L \iff \mathrm{Gap}_N(x) \text{ is top modulo } 2^{g(x)}. \tag{1.5}$$

This compares well with the standard definition of PP as the class of languages L such that for some N and all x,

$$x \in L \iff \mathrm{Gap}_N(x) > 0. \tag{1.6}$$

Both PP and MP are closed under complements. The closure of PP under intersection follows via (1.6) from the existence of an integer-valued function h such that for all polynomial-time NTMs N_1 and N_2 and all x,

$$h(\mathrm{Gap}_{N_1}(x), \mathrm{Gap}_{N_2}(x), x) > 0 \iff \mathrm{Gap}_{N_1}(x) > 0 \wedge \mathrm{Gap}_{N_2}(x) > 0, \tag{1.7}$$

and such that there is a polynomial-time NTM N_3 whose gap function $\mathrm{Gap}_{N_3}(x)$ equals the left-hand side of (1.7). One would like to have a polynomial A such that for all integers r and s, $A(r,s) > 0 \iff r > 0 \wedge s > 0$, but no such A exists [MP68]. However, we only need this to hold for those r and s that can possibly arise as values of $\mathrm{Gap}_{N_1}(x)$ and $\mathrm{Gap}_{N_2}(x)$. For some k depending only on N_1 and N_2, these must be in the range $[-2^m \dots 2^m]$, where $m = |x|^k$. The following polynomials A_m fitting this bill were found by Beigel, Reingold, and Spielman [BRS91a, BRS95], and were based on one-variable rational functions (i.e., quotients of two polynomials) that compute *sign*(x) on similar ranges found by Newman [New64].

$$\begin{aligned} A_m(r,s) &:= \frac{1}{4}(P_m(r)+P_m(-r))(P_m(s)+P_m(-s)) \\ &\quad -P_m(r)(P_m(s)+P_m(-s)) - P_m(s)(P_m(r)+P_m(-r)), \\ &= \frac{1}{4}(3P_m(r)-P_m(-r))(3P_m(s)-P_m(-s)) \;-\; 4P_m(r)P_m(s) \end{aligned}$$

where

$$P_m(r) = (r-1)\prod_{i=1}^{m}(r-2^i)^2.$$

For more details, see [BRS95]. Unlike the Toda polynomials, the coefficients of A_m do not belong to $\mathbf{Z}$. However, all values on integral arguments belong to $\mathbf{Z}$, so we call A_m *integer-valued*, and the degree of A_m is polynomial in m. As shown in [BRS95], this suffices for constructing the required polynomial-time NTM N_3. Fortnow and Reingold extended this construction to show that PP is closed under polynomial-time truth-table reductions [FR91]. Ogihara [Ogi95] has recently extended these polynomials to show that the log-space analogue PL of PP is self-low, i.e., that $\mathrm{PL}^{\mathrm{PL}} = \mathrm{PL}$.

Now let us turn attention to the problem of whether MP is closed under intersection. This time, what we want is a polynomial bound on degree in the following statement.

> In terms of k, what is the minimum degree of an integer-valued polynomial $p(x,y)$ such that for some polynomial t and all x and y, $p(x,y)$ is top modulo $2^{t(k)} \iff$ both x and y are top modulo 2^k?

Note that p may have rational coefficients so long as it is integer-valued. The simplest polynomial we know that satisfies this congruence relation (with $t(k) = k$) is $p(x,y) = \binom{x}{2^{k-1}}\binom{y}{2^{k-1}}2^{k-1}$, which has degree 2^k. M. Coster and A. Odlyzko [personal communication, 3/91] found solutions with degree $O(\phi^k)$, where ϕ is the golden ratio $1.618\ldots$, and with coefficients likewise appreciably smaller than the above. If such p can be found with degree polynomial in k, then p can be written as a polynomial-sized sum of small binomial coefficients in x and y, which can then be used in building the polynomial-time NTM needed to show MP closed under intersection.

A related question is: What is the minimum degree required to achieve, with all quantities defined modulo m,

$$p(x,y) = 0 \iff (x = 0 \ \wedge \ y = 0)?$$

With integer coefficients, this is possible iff m is square-free—and then p can have degree 2. For the case $m = 2^k$ (and rational coefficients), D.A.M. Barrington [personal communication, 11/95] gives an argument that makes an $\Omega(\sqrt{m}) = \Omega(2^{k/2})$ lower bound on degree highly plausible, for both this and the "top mod 2^k" problem with $t(k) = k$. The main idea is that there is a unique way to write $p(x,y)$ in the form $\sum_{i,j} a_{i,j}\binom{x}{i}\binom{y}{j}$ with integral $a_{i,j}$. A well-known fact is that $\binom{2^k}{i}$ is divisible by $2^k/\mathit{ord}_2(i)$, where $\mathit{ord}_2(i)$ stands for the largest power of 2 dividing i. With $x = y = m/2$, all the terms with both $\mathit{ord}_2(i)$ and $\mathit{ord}_2(j)$ at most $\sqrt{m}/2$ are divisible by m. Thus in particular, all terms with $1 \leq i,j \leq \sqrt{m}/2$ disappear in the congruence mod m. The only low-degree terms that can squeak through this analysis are those with $i = 0$ or $j = 0$, and these give us single-variable polynomials (with zero constant term) $q(x)$ and $r(y)$ such that $p(m/2, m/2)$ is congruent to $p(0,0) + q(m/2) + r(m/2)$ modulo m. If the q and r terms can be made to "go away," we have the desired contradiction.

Note that this would not contradict the above upper bound since it amounts to $\Omega(1.414\ldots^k)$. The tantalizing aspect, however, is that even this argument has no effect when $t(k) \geq 2k$. It may yet be possible to build two-variable "modulus-shifting" polynomials to meet the above requirement for closure of MP under intersection, and a direct and efficient-enough construction might collapse some counting classes between $\mathrm{PP}^{\oplus\mathrm{P}}$ and P^{PP}, as discussed at the end of Section 3 of [GKR+95].

7 Probabilistic Polynomials

A *probabilistic polynomial* in n variables over a ring $\mathcal{R}$ is formally defined, following Tarui [Tar93], as a mapping σ from a *sample space* U to the set $\mathcal{R}[x_1,\ldots,x_n]$ of polynomials in variables $x_1,\ldots,x_n$ with coefficients in $\mathcal{R}$. The *degree* of σ is defined to be the maximum, over all $j \in U$, of the degree of the polynomial σ_j. We write $\Pr_{j\in U}[\ldots]$ to indicate sampling according to the uniform distribution on U.

Definition 7.1. A probabilistic polynomial σ *represents* a language L *within error* δ, using scheme (e_1, e_0, S_1, S_0), if for all n and $x \in \{0,1\}^n$,

$$\begin{aligned} x \in L &\implies \Pr_{j\in U}[\sigma_j(x) \in S_1] \geq 1 - \delta, \\ x \notin L &\implies \Pr_{j\in U}[\sigma_j(x) \in S_0] \geq 1 - \delta. \end{aligned}$$

Here we will use the standard $(1, 0)$ input representation, and consider both strong and weak representation for outputs. We will also consider cases where $\mathcal{R}$ is the field $\mathrm{GF}(2^k)$ and k may increase with n, so that the overhead in Proposition 3.1 becomes a factor. However, we will then convert from weak representations over $\mathrm{GF}(2^k)$ to representations over GF(2), where strong and weak are the same. Over GF(2), schematic terms are products of x_i and $(1 - x_i)$, and we simply call them "terms." The following two results are well-known.

Proposition 7.1 (see [Tar93]) *Let σ be a probabilistic polynomial in n variables over $\mathbf{Z}_m$ (with standard nonzero representation) that has degree d and sample space $\{0, 1\}^r$, and such that for each j, σ_j is written with at most c terms. Then there is an equivalent $Mod_m \circ \mathrm{AND}$ circuit C with n "actual inputs" and r "random inputs," such that the AND layer has at most $c2^r$ gates, each of fan-in at most $d + r$.*

Proof. Each possible value of j can be regarded as a binary string of length r. For each j and each term in σ_j involving variables $x_{i_1}, \ldots, x_{i_d}$, assign an AND gate with $d + r$ wires to the inputs. The first d wires go to the actual inputs or their complements that appear in the term, and the other r wires go to the random inputs or their complements, each according to whether the corresponding bit of j is 1 or 0. Then this AND gate evaluates to 1 iff the values of the random inputs are precisely j, and the corresponding clause in σ_j contributes 1. The Mod_m of all the AND gates for this j is the same as the value of $\sigma_j(x)$ modulo m. All AND gates for other values of j output zero, so connecting everything to a single Mod_m gate yields the correct values for all j and x. □

The circuits in turn can be regarded as polynomials over $\mathbf{Z}_m$ that have n "actual arguments" and r "random arguments." This shows the essential equivalence between Tarui's convenient formalism using distributions and the older notion of a single polynomial with "probabilistic arguments."

For the case $m = 2$, there is a deterministic simulation that is more efficient than Theorem 6.1 given above.

Proposition 7.2 ([All89, AH90]) *If the probabilistic polynomial σ represents f over GF(2) with success probability $> 1/2$, then the probabilistic circuit C simulating σ in Proposition 7.1 can be converted to a deterministic depth-three $MAJ \circ MAJ \circ \mathrm{AND}$ circuit C' that has $c2^r$ AND gates, each of fan-in d, and $c2^{2r}$ MAJ gates in the second layer.*

Proof. This follows from the simulation of a MAJ of u-many Parity gates, where each Parity has fan-in at most m (an even number), by a depth-2 circuit comprised of $um + 1$ MAJ gates [All89, AH90]. □

Håstad and Goldmann [HG91] proved that any depth-3 circuit of this kind (even with the MAJORITY gates replaced by arbitrary unweighted threshold gates) that computes the GF(2) polynomial $\sum_{i=1}^{n} \prod_{j=1}^{d} x_{ij}$ must have size at least $2^{\Omega(n/(d+1)4^{d+1})}$, which translates to $2^{n^{\Omega(1)}}$ if $d \leq (1/3)\log n$. Razborov and Wigderson [RW93] showed that any such circuit that computes the GF(2) polynomial $\sum_{i=1}^{n} \prod_{j=1}^{\log n} \sum_{k=1}^{n} x_{ijk}$ must have $n^{\Omega(\log n)}$ size, regardless of the bottom fan-in d. This still leaves open the possibility of achieving polynomial size in the depth-3 construction for functions in (uniform) AC^0, or for achieving polynomial size with a higher constant depth (cf. [HHK91]).

Now we look concretely at polynomials for OR. First note that under the sign output representation, OR is trivially represented over $\mathbf{Z}$ by the degree-one polynomial $\sum_{i=1}^{n} x_i$. Under strong representation, however, the degree jumps all the way to n, over $\mathbf{Z}_m$ as well as $\mathbf{Z}$. For probabilistic polynomials, however, strong representation is much less costly.

Theorem 7.3 ([ABFR94]) *OR is strongly represented over* $\mathbf{Z}$ *within error* ϵ *by probabilistic polynomials of degree* $O(\log(1/\epsilon)\log n)$.

Proof. We vary somewhat from the proof in [ABFR94]: Let $\ell = \lceil \log_2 n \rceil$. For each k, $0 \leq k \leq \ell$, let $R_k \subseteq \{1, \ldots, n\}$ be randomly selected by independently placing $i \in R_k$ with probability $1/2^k$. This gives us for each k a randomly-selected polynomial $\rho_k(x_1, \ldots, x_n) = \sum_{i \in R_k} x_i$. Finally define

$$\sigma(x_1, \ldots, x_n) = 1 - \prod_{k=0}^{\ell} (1 - \rho_k(x_1, \ldots, x_n)). \tag{1.8}$$

Now when all x_i are 0, with probability 1 this polynomial gives 0. When the set S of x_i that are 1 is nonempty, an easy analysis of the two values of k that straddle $\log_2 |S|$ shows that with probability at least 1/4, some ρ_k takes value 1, so the product is zero, so σ takes value 1. Finally, to amplify the 1/4 to $1 - \epsilon$, replace the product in (1.8) by a product of $\lceil \log_2(4/\epsilon) \rceil$ independent copies of $\prod_k (1 - \rho_k(x_1, \ldots, x_n))$. $\square$

This uses $O(n \log(1/\epsilon))$ random bits. As is well known, one only needs the random variables defining the sets R_k to be pairwise-independent, and a standard universal hashing construction needs only $O(\log n)$ random bits for the success probability 1/4, hence $O(\log n \log(1/\epsilon))$ random bits overall. This construction works for probabilistic strong representation over $\mathbf{Z}_m$ as well as $\mathbf{Z}$.

The question we ask is: Can one do better in the degree and random-bits measures? We show that the answer is *yes* for polynomials over any finite field, including $\mathbf{Z}_p$ with p prime, by a construction involving error-correcting codes. Such codes were used by Tarui [Tar93] in non-constructive arguments about probabilistic polynomials, and by Naor and Naor [NN93] in other contexts. Here we emphasize the way probabilistic polynomials are constructed from the codes. For sake of comparison, we first show how the basic "parity trick" of Naor and Naor [NN93] can be applied to eliminate the product over k in (1.8): Using $\ell + 1$ more random bits $b_0, \ldots, b_\ell$, define

$$\sigma(x_1, \ldots, x_n) = \sum_{k=0}^{\ell} b_k \rho_k(x_1, \ldots, x_n)) \pmod 2.$$

Then $\sigma(\vec{0}) = 0$ with probability one. For all arguments $\vec{x} \neq 0$, think of some ρ_k that gives value 1. The bit b_k alters the overall sum by 1 depending on its value. Hence with probability at least 1/8, $\sigma(\vec{x}) = 1 \pmod 2$. Thus we have a degree-one probabilistic polynomial achieving constant success probability for OR. This amplifies by repeated trials to degree $O(\log(1/\epsilon))$ for success probability $1 - \epsilon$, using $O(\log n)\log(1/\epsilon)$ random bits. Hence this saves an $O(\log n)$ factor on degree compared to Theorem 7.3, although we have strong representation over $\mathbf{Z}_2$ instead of over $\mathbf{Z}$.

Open Problem 1. Can $\vee$ be strongly represented within error ϵ over $\mathbf{Z}$ by probabilistic polynomials of degree $\log(1/\epsilon) \cdot o(\log n)$?

This relates to whether the randomized reduction from SAT to Unique SAT by Valiant and Vazirani [VV86] can be made as efficient as the reduction to Parity SAT as optimized in [NRS95] (see also [NN93, Gup93]).

The construction via error-correcting codes that we present next achieves slightly better constants in the bounds compared to the above. The success probability for degree-one representation over $\mathbf{Z}_2$ is improved from 1/8 to $1/2 - \eta$, where η has a minimal effect on the other bounds. The number of random bits beats the bound of $2 \log n$ needed for pairwise independence, i.e., needed to construct a family of universal$_2$ hash functions from n bits to n bits.

7.1 Error-Correcting Codes

In this subsection, let Σ be an alphabet whose cardinality is a prime power $q = p^k$. The *Hamming distance* $d_H(x, y)$ of two strings $x, y \in \Sigma^*$ of equal length is defined to be the number of positions in which x and y differ. For any $N, d \geq 1$, an (N, d) *code* over Σ is a set $C \subseteq \Sigma^N$ such that for all distinct $x, y \in C$, $d_H(x, y) \geq d$. Elements of C are called *codewords*. We identify Σ with the finite field $\mathcal{F} = \mathrm{GF}(p^k)$, and then Σ^N can be regarded as a vector space of dimension N over $\mathcal{F}$.

Definition 7.2. A *linear code* over $\mathcal{F}$ with *parameters* $[N, K, d]$ is an (N, d) code C that forms a vector subspace of Σ^N of dimension K over $\mathcal{F}$. Two other important parameters are the *rate* $R = K/N$, and the *density* $\delta = d/N$.

The use of $[\ldots]$ to distinguish linear codes is standard in coding theory. Where intent is clear we write $[N,K,\delta]$ in place of $[N,K,d]$. Given any K-dimensional vector subspace $C \subseteq \Sigma^N$, denote by d_C the maximum d such that C is an $[N,K,d]$ code, and write $\delta_C = d_C/N$. Thus d_C equals $\min\{\, d_H(x,y) : x,y \in C, x \neq y \,\}$, and so is called the *minimum distance* of the code C. The *weight* $wt(x)$ of a string $x \in \Sigma^N$ is the number of nonzero entries, which is the same as $d_H(x,0)$. A well-known fact is that in a linear code C, the minimum distance is equal to the minimum weight of a non-zero codeword. This is because for all $x,y \in C$, $x-y$ is also in C. Thus the density δ_C gives the minimum proportion of non-0 entries in any non-zero codeword. Where intent is clear we write just d and δ for d_C and δ_C.

Definition 7.3. A *generator matrix* G for an $[N,K,d]$ code C is a $K \times N$ matrix over $\mathcal{F}$ whose rows $G(i,\cdot)$, $1 \leq i \leq K$, form a basis for C.

Now we indicate how we intend to make N and K scale with our input lengths n.

Definition 7.4. Let $[C_n]_{n=1}^{\infty}$ be a sequence of $[N_n,K_n,d_n]$ codes over $\mathcal{F}$. Then the C_n are said to be *small codes* if $N_n = n^{O(1)}$, and *large codes* if they are not small and $N_n = 2^{n^{O(1)}}$.

For probabilistic polynomials we use small codes, with $K_n = n$:

Proposition 7.4 *Let G generate an $[N,n,\delta]$ code over $\mathcal{F}$. Then the probabilistic polynomial σ with sample space the columns of G, defined by*

$$\sigma_j(x_1,\ldots,x_n) = \sum_{i=1}^{n} G(i,j)x_i, \tag{1.9}$$

represents $OR(x_1,\ldots,x_n)$ over $\mathcal{F}$ with success probability at least δ.

Proof. If all x_i are 0, then for all j, $\sigma_j(x_1,\ldots,x_n) = 0$, so this probabilistic polynomial gives one-sided error. Now let $S = \{\, i : x_i = 1 \,\}$ be nonempty. To S there corresponds the unique codeword $w_S = \sum_{i=1}^{n} G(i,\cdot)$. Since w_S is nonzero, with probability at least δ over j sampled uniformly from $\{\, 1,\ldots,N \,\}$, $w_S(j) \neq 0$. And $w_S(j) = \sigma_j(x_1,\ldots,x_n)$. $\square$

Note that σ has the columns of G as its sample space and is linear. Also, $1-\sigma$ represents NOR, $\sigma(1-x_1,\ldots,1-x_n)$ represents NAND, and $1-\sigma(1-x_1,\ldots,1-x_n)$ represents AND. These probabilistic polynomials are also linear with constant success probability. The number r of random bits used is $\lceil \log_2 N \rceil$. Expressed in terms of the rate $R = K/N$, with $K = n$, $r = \log n + \log(1/R)$. Thus if the rate is constant, $r = \log n + O(1)$, while if N is polynomial in K, $r = O(\log n)$. This motivates the next definition, part (a) of which is standard in coding theory.

Definition 7.5. (a) A sequence $[C_n]_{n=1}^{\infty}$ of codes over $\mathcal{F}$ is *asymptotically good* if there are constants $R, \delta > 0$ such that $(\forall^{\infty} n) R_n \geq R \wedge \delta_n \geq \delta$.

(b) The sequence is *almost-good* if $\delta > 0$ exists giving $(\forall^{\infty} n)\delta_n \geq \delta$, and the lengths N_n are polynomial in K_n.

The emphasis in Proposition 7.4 is on the density, rate, and the complexity of computing individual entries $G(i,j)$. This stands apart from the main application of coding theory, which is to take a plaintext message w of length K, *encode* w into the codeword $x = wG$, transmit x over a channel that may alter it to x', and have the receiver *decode* x' to recover w. So long as $d_H(x,x') < d/2$, the "nearest codeword" decoding algorithm given x' will find x and then w. The higher one can make δ and R, the more erroneous symbols one can correct and the less the transmission overhead. In recent breakthrough work, Spielman [Spi95] constructed asymptotically good codes with $K_n = N_n = O(n)$ that give encoding and decoding in linear time. However, we do not know whether the computation of entries of his $G_n(i,j)$ can be done in (uniform) AC^0. The codes used by Sudan [Sud92] to streamline the "PCP" results of [ALM$^+$92] are almost-good, and put $G(i,j)$ into AC^0 when used as small codes. They suffice for the next result.

Theorem 7.5 *There are* AC^0*-uniform linear probabilistic polynomials that represent OR over GF*(2) *with constant success probability using* $O(\log n)$ *random bits.*

Proof. We scale down the main theorem in section 3 of [NRS95] from "large codes" to "small codes" with $K = n$ as follows: Let $h = \lceil \log_2 K \rceil$, $\ell = \lceil \log_2 h \rceil$, and $m = \lceil h/\ell \rceil$. That is, we identify $\{1, ..., K\}$ with (a subset of) $\{0,1\}^h$ for the row labels, and break each $i \in \{0,1\}^h$ into m strings $i_1 \cdots i_m$, where each has length ℓ. Here we may suppose that the last one, i_m, is padded out to length ℓ with 0s. The first idea is that each such i corresponds to a monomial in m formal variables $z_1, \ldots, z_m$, namely $z_1^{i_1} \cdots z_m^{i_m}$, where now $i_1, \ldots, i_m$ are regarded as numbers between 0 and $h-1$. Note that all such monomials are distinct and have total degree D less than mh.

Now let $\eta > 0$ be arbitrary, let $s = \lceil \log_2(mh/\eta) \rceil$, and let $\mathcal{F}$ be the finite field $\mathrm{GF}(2^s)$. The column space of our matrix G over GF(2) is given by

$$J = \{ (a_1, \ldots, a_m, v) : a_1, \ldots, a_m, v \in \mathcal{F} \},$$

which is in 1-1 correspondence with strings j of length $(m+1)s$. Then for all rows i and columns $j = (a_1, \ldots, a_m, v)$ we define:

$$G(i,j) = (a_1^{i_1} \cdots a_m^{i_m}) \bullet v, \tag{1.10}$$

where the powers and products are over $\mathcal{F}$, but $\bullet$ stands for the dot-product of two binary strings, which brings the final result down into GF(2). Since the field elements in (1.10) are binary strings of length only $2\log\log n + O(1)$, and $m < \log n$, precomputed tables yield uniform AC^0 circuits (and we suspect the whole is in DLOGTIME as treated in [BIS90]). The number of random bits to generate j is $(m+1)s \simeq ms =$

$$\frac{h}{\log h}\log\left(\frac{h^2}{\eta\log h}\right) = 2h - \frac{h\log\log h}{\log h} + \frac{h}{\log h}\log(1/\eta) < 2\log n.$$

Hence the codes are almost-good, with length N a tad below n^2.

We claim that G generates a code of the required dimension and density. Since the distinct monomials are linearly independent in $\mathcal{F}[z_1,\ldots,z_m]$, they generate a space of dimension K over $\mathcal{F}$. For the density we use the key lemma from the aforementioned "PCP" papers, often ascribed to Schwartz [Sch80] but anticipated by Zippel [Zip79]: For every two distinct polynomials p and q of total degree at most D over a field $\mathcal{F}$, and every $I \subseteq F$,

$$|\{\,\vec{a}\in\mathcal{F}^m : p(\vec{a}) = q(\vec{a})\,\}| \le D|I|^{m-1}.$$

With $I = \mathcal{F}$, it follows that every nonzero polynomial p in our space takes on at least $|\mathcal{F}|^m - D|\mathcal{F}|^{m-1}$ nonzero values. Dividing by $|\mathcal{F}|^m$ says that the proportion of nonzero values is at least $1 - D/|\mathcal{F}| = 1 - mh/2^s = 1-\eta$. Now consider the codeword w_p corresponding to p, and. consider a nonzero value $p(a_1,\ldots,a_m) = u$. This corresponds to a range of 2^s-many columns indexed by $(a_1,\ldots,a_m,v)$ over all $v\in\mathcal{F}$. Since $u\neq 0$, exactly half of those v give $u\bullet v = 1$. Hence the density of the codeword w_p is at least $(1-\eta)/2$, and this fulfills the claim made about the code. (Technically, G is the "concatenation" of the code over $\mathcal{F}$ with the so-called binary "Hadamard code" defined by the dot-product function over GF(2).)

Finally, Proposition 7.4 gives us the desired linear probabilistic polynomials, for AND, NOR, NAND as well as OR, with constant one-sided error arbitrarily close to 1/2, and with polynomial sample-space size. $\square$

We do not know of a sequence of *good* small codes that is AC^0-uniform. B.-Z. Shen [She93] shows how to construct asymptotically good binary codes by an algebraic technique that (in an analogous situation) chops many columns out of J without reducing the density of the code, but we do not know how uniform the "chops" are.

In the case of large codes with $K = 2^n$, the computation of $G(i,j)$ in (1.10) involves field elements of size $O(\log n)$. Since both the sequences $(a_1,\ldots,a_m)$ and $(i_1,\ldots,i_m)$ can be read left-to-right, the entire computation can be done in one-way log-space. Thus NL random-logspace reduces to $\oplus$L, and this immediately implies Wigderson's theorem that NL/*poly* $\subseteq$ $\oplus$L/*poly* [Wig94]. We would like to know whether this computation can be done in TC^0.

Matters become more complex when the error tolerance ϵ is not constant but shrinks rapidly with n. A example application is simulating AC^0 circuits of depth b and size n^c within a target error e. We may suppose that each gate is a NAND gate of fan-in at most n. The idea is to substitute "the same" probabilistic polynomial σ of degree d and error ϵ_n for each gate. Composing these polynomials then yields a single probabilistic polynomial τ of degree d^b in the input variables $x_1, \ldots, x_n$ of the circuit that computes it with error at most $\epsilon_n n^c$. This works even though the "errors at each gate" are not independent; note that τ has the same sample space as σ. Thus we wish to arrange $\epsilon_n \leq e/n^c$.

We could do $O(\log(1/\epsilon_n))$ independent trials in this example, thus making $d \simeq c \log n$ and using $r(n) = O(\log n \log(1/\epsilon_n)) = O(\log^2 n)$ random bits. Plugging this in to the construction in Theorem 7.5 improves the original degree bounds of Beigel, Reingold, and Spielman [BRS91a, BRS95], and stays slightly ahead of the result of Gupta [Gup93], but still falls well short of the optimal bounds shown to be achievable by the *non-constructive* argument of Tarui [Tar93].

However, there is a very interesting possibility of achieving better bounds in a uniform manner by using larger fields $\mathcal{F}$. It is known that whenever $\delta < 1 - 1/|\mathcal{F}|$, sequences of good codes over $\mathcal{F}$ with $\delta_n \geq \delta$ exist. If we have $1/2^k \leq \epsilon = 1 - \delta$, then the argument of Proposition 7.4 immediately gives us a linear probabilistic polynomial τ over $\mathrm{GF}(2^k)$ that computes OR using $r(n) = \log n + O(1)$ random bits. The following then gives an alternative way to obtain a small-degree σ over GF(2) for OR. Let or_k be the unique polynomial that represents (deterministically!) k-bit OR over GF(2).

Proposition 7.6 *Let G be an $n \times N$ generator matrix for a code of density δ over $GF(2^k)$, and let G' be the straightforward way of regarding G as an $n \times kN$ matrix over $GF(2)$. Then the probabilistic polynomial σ defined by*

$$\sigma_j(x_1, \ldots, x_n) = or_k \left(\sum_{i=1}^{n} G'(i, kj - k + 1)x_i, \ldots, \sum_{i=1}^{n} G'(i, kj)x_i \right) \quad (1.11)$$

represents $OR(x_1, \ldots, x_n)$ over $GF(2)$ with error at most δ, and has degree k and sample space $\{1, \ldots, N\}$.

Note that σ has the same sample space as τ. There is a way to regard G as a $kn \times kN$ matrix G'' over GF(2), but the difference between G' and G'' seems not to matter much.

Now we want to ask: What happens to "good" codes over $\mathrm{GF}(2^k)$ when k scales upward with n? What must at least happen to the "$+O(1)$" in $r(n)$ above is shown by a coding-theory bound called the *Singleton bound*: in an $[N, K, d]$ code (over any field), $K + d \leq N + 1$. Written another way with $\epsilon = 1 - d/N$, the bound becomes $N \geq (K - 1)/\epsilon$, and putting $K = n$, this says that the number of random bits used must satisfy

$$r(n) \geq \log_2(n-1) + \log_2(1/\epsilon).$$

This is much better than the $r(n) = \log(n)\log(1/\epsilon)$ from repeated trials. There *are* codes that meet this bound—such codes are called *maximum distance separable* (MDS). The question now becomes: Can we construct MDS (or nearly MDS) codes of high density over $\mathrm{GF}(2^k)$? This and the question of how the densities δ_k and rates R_k for good codes in Definition 7.5 may scale with k lead into deep areas of coding theory, for which [MS77] is a standard reference. The issues here about working with larger fields of the same characteristic (here, $\mathrm{GF}(2^k)$ versus $\mathrm{GF}(2)$) seem connected to similar issues in Smolensky's paper [Smo93] (see Theorem 5.10 cited above). We believe that there is room in the theory of error-correcting codes for new discoveries that will add to our knowledge about complexity classes.

8 Other Combinatorial Structures

If we blur the distinction between a language L and its complement $\sim L$, we can regard L as a "two-coloring" of Σ^*. We seek connections to a powerful body of mathematics that is bound up with generalizations of a familiar theorem about two-colorings of the plane:

> Any map formed by simple closed curves and infinite straight lines in the plane can be colored with two colors, so long as all intersections are "general"—meaning that for every pair of curves or lines, the intersections (if any) between them form a collection of isolated crossing points.

Versions of this theorem extend to higher dimensions. To use them, we need to identify Σ^* with a subset of real space. We decide to take $\Sigma = \{0, 1\}$ and identify Σ^n with the vertices of the unit cube in $\mathbf{R}^n$, for each n.

The generalization of "infinite straight line" to $\mathbf{R}^n$ is a *hyperplane*, meaning an affine translation of an $(n-1)$-dimensional subspace of $\mathbf{R}^n$. Every hyperplane is defined by an equation of the form $\sum_{i=1}^{n} x_i w_i = t$. The (upper) *open half space* associated to the hyperplane consists of points $\vec{x}$ satisfying $\sum_{i=1}^{n} x_i w_i > t$. Note that this inequality defines a threshold gate. A *polytope* is an intersection of open half-spaces that defines a bounded nonempty region of space, together with its *surface* consisting of those points belonging to the hyperplanes that are added in forming the topological closure of this region. (This wording makes all polytopes "full-dimensional.") Every polytope is convex. Familiar examples of polytopes in $\mathbf{R}^3$ are the tetrahedron, cube, octahedron, and prism, but not a cylinder, cone, or sphere.

For simplicity, we will not work with the theories of algebraic curves and algebraic topology that provide full generalizations of simple closed curves in the plane, but will confine attention to polytopes. Say a collection of polytopes is in "general position" if no polytope has a hyperplane that goes through a vertex of the unit n-cube, and no two polytopes share a hyperplane nor any lower-dimensional facet. (See [MP68] for more on this.) Then we have an analogue of the above two-coloring theorem:

> Every finite collection of polytopes in general position defines a two-coloring of $\mathbf{R}^n$, and every vertex of the unit n-cube has a well-defined color.

Definition 8.1. A language L belongs to *PALT* if there are polynomial-sized collections $\mathcal{P}_n$ of polytopes in general position, with each member of $\mathcal{P}_n$ defined by polynomially-many half-spaces, such that for all n, all points in $L^{=n}$ have one color, and all points in $\sim L^{=n}$ have the other color.

By definition, PALT is closed under complements. If we want to distinguish "the language of $\{\mathcal{P}_n\}$" from its complement, we may exploit the fact that the partition of $\mathbf{R}^n$ by $\mathcal{P}_n$ has only one infinite region, and define $L(\mathcal{P}_n)$ by those points on the unit cube that have the same color as this region. We can always complement L by adding one polytope that encloses the unit cube.

Now we can characterize this idea in terms of the circuit classes defined at the end of Section 3.

Proposition 8.1 *A language L belongs to* PALT *iff L is recognized by polynomial-sized Parity $\circ$ AND $\circ$ LT circuits.*

Proof. Let L in PALT, and let $\mathcal{P}_n$ define $L^{=n}$. Each polytope is definable by an AND of LT gates. By a standard lemma in [MP68], each LT gate can be replaced by an LT gate that gives the same outputs on the vertices of the unit cube in $\mathbf{R}^n$, such that the weights and threshold for the latter gate all have $O(n \log n)$ bits. Hence the circuits obtained by attaching a single parity gate to all the AND-of-LTs are polynomial-size P $\circ$ A $\circ$ LT circuits. Now we claim that two vertices x and y of the unit cube have different colors iff the number of polytopes $P \in \mathcal{P}_n$ such that exactly one of x, y belongs to the interior of P is odd. This claim implies that the P $\circ$ A $\circ$ LT circuit computes the same language as the coloring. To verify the claim, consider the straight line segment ℓ from x to y. Now all intersection points of ℓ with the surfaces of polytopes are isolated, for if ℓ were to lie along the surface of a polytope, then some hyperplane involved would go through x and y, in violation of general positioning. Two different polytopes may intersect ℓ at a given point, but each intersection, counting this kind of multiplicity, represents a color change. Hence the total number of intersections, counting multiplicities, is odd. Since a polytope that has both or neither of x, y in its interior contributes 0 or 2 to this total, the claim is proved.

Going the other way, given a P $\circ$ A $\circ$ LT circuit, each AND gate defines an intersection of open half spaces. By adding some LT gates that output *true* for all assignments in $\{0, 1\}^n$, we can make this intersection into an equivalent polytope. The lemma from [MP68] can also be used to tweak the polytopes into general position without changing any values on the unit cube. □

Thus PALT is a small-depth, polynomial-size circuit class. It seems to lie "just above" the polynomial-size circuit classes for which strong lower bounds are currently known, such as those treated in [HG91, GHR92, MT93, Mac95]. All of the latter classes are contained in polynomial-size TC^0 depth-3. Much attention has focused on the problem of whether $\mathrm{NC}^1 = \mathrm{TC}^0_3$. We find it just conceivable that NC^1 might equal PALT. Since PALT $\subseteq \mathrm{TC}^0_4$ by the obvious simulation of AND gates by MAJ gates and the known simulation of Parity by two levels of MAJ [CSV84], this would imply $\mathrm{NC}^1 = \mathrm{TC}^0_4$. However, we do not know whether PALT $\subseteq \mathrm{TC}^0_3$. The best depth simulation we know was furnished by Alexis Maciel and refined in personal communications with Maciel and Mikael Goldmann.

Theorem 8.2 *For any $m > 0$, $Mod_m \circ \mathrm{AND} \circ LT$ has quasipolynomial-sized circuits consisting of a Midbit gate connected to one layer of MAJ gates at the inputs. In particular,* PALT $\subseteq q\mathrm{TC}^0_3$*, where the "q" indicates quasipolynomial size.*

Proof. As shown in [CSV84], by using iterated addition to simulate the weights in an LT gate, $LT \subseteq \mathrm{AC}^0 \circ MAJ$. Thus

$$Mod_m \circ \mathrm{AND} \circ LT \subseteq Mod_m \circ \mathrm{AC}^0 \circ MAJ.$$

Now by Theorem 6.1, $Mod_m \circ \mathrm{AC}^0 \subseteq q(Midbit \circ \mathrm{AND}_{small})$, where all AND gates in the corresponding level are *small*, i.e., have polylog fan-in. Now each MAJ gate involved has fan-in at most $r = n^{O(1)}$. We want to simulate each small-AND of MAJ by a single SYM gate of quasipolynomial fan-in. Theorem 3.6 of [Mac95], which is a slight extension of the relevant special case of results in [Bei94b] and [HHK91], does this by brute-force coding of all $(r+1)^{\mathrm{polylog}(n)}$ possible vectors of sums of the input bits to the polylog(n)-many MAJ gates, with a different integer for each vector. The coding produces a function of these integers, which becomes a symmetric function of $n^{\mathrm{polylog}(n)}$-many input lines. The codings used in [Bei94b, Mac95] do not produce a threshold function of these integers, so they do not yield a single MAJ gate. However, because all of the MAJ and small-AND gates above can be normalized to have the same fan-in via "dummy inputs," the number k of values on which each new SYM gate outputs *true* can be the same for all SYM gates in the circuit. Thus

$$Mod_m \circ \mathrm{AND} \circ LT \subseteq q(Midbit \circ SYM).$$

By a lemma of Hájnal et al. [HMP$^+$87], every symmetric 0-1 valued function h in m "own" variables can be written as

$$h(z_1, \ldots, z_m) = g(M_1(z_1, \ldots, z_m), \ldots, M_{2k}(z_1, \ldots, z_m)),$$

where (1) $i_1 < i_2 < \ldots < i_k$ are the k values of $z_1 + \cdots + z_m$ on which h outputs 1, (2) for each j, $1 \leq j \leq k$, $M_{2j}(z_1, \ldots, z_m) = 1 \iff z_1 + \cdots + z_m \geq i_j$ and $M_{2j-1}(z_1, \ldots, z_m) = 1 \iff z_1 + \cdots + z_m \leq i_j$, and (3) g is the linear $2k$-variable integer polynomial $g(r_1, \ldots, r_{2k}) = r_1 + \cdots + r_{2k} - k$. Each of the M_j functions is computable by a single MAJ gate using extra "dummy" inputs (and using the fact that negated inputs are available), so we can abbreviate this as $h = g \circ MAJ$. So the whole circuit is now a quasipolynomial-size $Midbit \circ g \circ MAJ$.

Now finally we claim that since g is *linear*, the $Midbit \circ g$ portion of the circuit can be replaced by a single *Midbit* gate. This is because the original single Midbit gate M outputs the middle bit of the binary sum of its inputs, and each input line $g(r_1, \ldots, r_{2k})$ is itself a sum, minus k. Hence we can gather all the inputs to all g's into a "positive section" of inputs to a new gate M', and all of the "$-k$"s into a "negative section," so that the new M' outputs the middle bit of the *gap* between the number of inputs in its positive section that are on and the number of inputs in its negative section, all of which are on. Here we are helped for ease of verification by the fact that we have made k the same for each of the quasipolynomially-many inputs $g_1, \ldots, g_q$ to the original M, so that the new M' outputs the middle bit of

$$\left(\sum_{i=1}^{q} \sum_{j=1}^{2k} r_{j,i} \right) - qk.$$

Analogous to the way that the class MidbitP is robust under changing the definition to the middle bit of a GapP function, this M' can be transformed into a Midbit gate (with all its input lines treated "positively"). This leaves us with a quasipolynomial-size $Midbit \circ MAJ$ circuit. Finally, applying the same lemma from [HMP$^+$87] to the symmetric gate M' produces a quasipolynomial-size circuit of the form $g' \circ MAJ \circ MAJ$, which is clearly in $q\mathrm{TC}^0_3$. □

PALT seems to be worthy of further research. Some open questions: Is it contained in polynomial-size TC^0 depth-3? Does it contain (uniform) AC^0? We note that given collections $\mathcal{P}_1$ defining L_1 and $\mathcal{P}_2$ defining L_2, one can construct a collection $\mathcal{P}_3$ defining $L_1 \cap L_2$ by taking all pairwise intersections of polytopes in $\mathcal{P}_1$ and $\mathcal{P}_2$. However, the size blowup in the collections by this method is too great to answer these questions. The real technical matter of interest seems to be what kind of projections can be done from $\mathbf{R}^n$ to $\mathbf{R}^m$ with $m < n$.

A simpler class PLT can be defined in terms of two-colorings obtained from collections of hyperplanes alone. Similar to Proposition 8.1, these are equivalent to polynomial-size Parity $\circ$ LT circuits, which are a proper subclass of TC^0 depth-3. The class of languages defined by a single hyperplane in $\mathbf{R}^n$ (for each n) is called LT_1 by Agrawal and Arvind [AA95], who show that if NP polynomial-time bounded truth-table reduces to LT_1, then NP = P. Clearly PALT polynomial-time truth-table reduces to LT_1, but this reduction is neither bounded nor conjunctive nor disjunctive, so even the stronger results in [AA95] seem not to apply, and we do not know how PALT relates to P and NP. A truth-table reduction to LT_1 is essentially the same as a polynomial-size *linear decision tree*, as studied by Björner, Lovász, and Yao [BLY92]. The exponential size lower bounds in [BLY92] and [Yao94] are for decision problems about arbitrary points in $\mathbf{R}^n$, however, and we do not know whether they carry over to problems restricted to points on vertices of the unit cube. Still, there seems to be much promise that geometrical methods of the kind used in [AA95, BLY92, Yao94] can be brought to bear on Boolean complexity via this route.

Acknowledgments

Alexis Maciel furnished the initial version of Theorem 8.2, during the 1995 Montreal-McGill Workshop on Computational Complexity in Barbados, March 1995. I also thank Maciel and Mikael Goldmann for subsequent discussion of this theorem and PALT in general. Richard Beigel provided me an updated draft of his survey [Bei93]. David Mix Barrington gave me helpful discussions and clarifications of much of the material, and the anonymous referee did the same. Barrington and Andrew Odlyzko (the latter for Mattijas Coster) sent interesting personal communications on the open problems in Section 6. Finally, I thank Eric Allender for bringing some useful results and matters to my attention.

9 References

[AA95] M. Agrawal and V. Arvind. Reductions of self-reducible sets to depth-1 weighted threshold circuit classes, and sparse sets. In *Proc. 10th Annual IEEE Conference on Structure in Complexity Theory*, pages 264–276, 1995.

[ABFR91] J. Aspnes, R. Beigel, M. Furst, and S. Rudich. The expressive power of voting polynomials. In *Proc. 33rd Annual IEEE Symposium on Foundations of Computer Science*, pages 402–409, 1991.

[ABFR94] J. Aspnes, R. Beigel, M. Furst, and S. Rudich. The expressive power of voting polynomials. *Combinatorica*, 14:1–14, 1994.

[AF90] M. Ajtai and R. Fagin. Reachability is harder for directed than for undirected finite graphs. *J. Symb. Logic*, 55:113–150, 1990.

[AH90] E. Allender and U. Hertrampf. On the power of uniform families of constant-depth circuits. In *Proc. 15th International Symposium on Mathematical Foundations of Computer Science*, volume 452 of *Lect. Notes in Comp. Sci.*, pages 158–164. Springer Verlag, 1990.

[AJ93] E. Allender and J. Jiao. Depth reduction for noncommutative arithmetic circuits (extended abstract). In *Proc. 25th Annual ACM Symposium on the Theory of Computing*, pages 515–522, 1993.

[All89] E. Allender. A note on the power of threshold circuits. In *Proc. 30th Annual IEEE Symposium on Foundations of Computer Science*, pages 580–584, 1989.

[ALM+92] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. In *Proc. 33rd Annual IEEE Symposium on Foundations of Computer Science*, pages 14–23, 1992.

[Bar92] D. Mix Barrington. Some problems involving Razborov-Smolensky polynomials. In M. Paterson, editor, *Boolean Function Complexity*, volume 169 of *LMS Lecture Note Series*, pages 109–128. London Math. Soc., 1992. Proceedings of an LMS Symposium in Durham, July 1990.

[BBR92] D. Mix Barrington, R. Beigel, and S. Rudich. Representing Boolean functions as polynomials modulo composite numbers. In *Proc. 24th Annual ACM Symposium on the Theory of Computing*, pages 455–461, 1992.

[BBR94] D. Mix Barrington, R. Beigel, and S. Rudich. Representing Boolean functions as polynomials modulo composite numbers. *Computational Complexity*, 4:367–382, 1994.

[Bei93] R. Beigel. The polynomial method in circuit complexity. In *Proc. 8th Annual IEEE Conference on Structure in Complexity Theory*, pages 82–95, 1993. Revised version, 1995.

[Bei94a] R. Beigel. Perceptrons, PP, and the polynomial hierarchy. *Computational Complexity*, 4:339–349, 1994.

[Bei94b] R. Beigel. When do extra majority gates help? polylog(n) majority gates are equivalent to one. *Computational Complexity*, 4:314–324, 1994.

[BIS90] D. Mix Barrington, N. Immerman, and H. Straubing. On uniformity within NC^1. *J. Comp. Sys. Sci.*, 41:274–306, 1990.

[BLY92] A. Björner, L. Lovász, and A. Yao. Linear decision trees: volume estimates and topological bounds. In *Proc. 24th Annual ACM Symposium on the Theory of Computing*, pages 170–177, 1992.

[Bre74] R. Brent. The parallel evaluation of general arithmetic expressions. *J. Assn. Comp. Mach.*, 21:201–206, 1974.

[BRS91a] R. Beigel, N. Reingold, and D. Spielman. The perceptron strikes back. In *Proc. 6th Annual IEEE Conference on Structure in Complexity Theory*, pages 286–291, 1991.

[BRS91b] R. Beigel, N. Reingold, and D. Spielman. PP is closed under intersection. In *Proc. 23rd Annual ACM Symposium on the Theory of Computing*, pages 1–9, 1991.

[BRS95] R. Beigel, N. Reingold, and D. Spielman. PP is closed under intersection. *J. Comp. Sys. Sci.*, 50:191–202, 1995.

[BS94] D. Mix Barrington and H. Straubing. Complex polynomials and circuit lower bounds for modular counting. *Computational Complexity*, 4:325–338, 1994.

[BST90] D. Mix Barrington, H. Straubing, and D. Thérien. Non-uniform automata over groups. *Inform. and Comp.*, 89:109–132, 1990.

[BT88] D. Mix Barrington and D. Thérien. Finite monoids and the fine structure of NC^1. *J. Assn. Comp. Mach.*, 35:941–952, 1988.

[BT91] R. Beigel and J. Tarui. On ACC. In *Proc. 32nd Annual IEEE Symposium on Foundations of Computer Science*, pages 783–792, 1991.

[BT94] R. Beigel and J. Tarui. On ACC. *Computational Complexity*, 4:350–366, 1994.

[CSV84] A. Chandra, L. Stockmeyer, and U. Vishkin. Constant-depth reducibility. *SIAM J. Comput.*, 13:423–439, 1984.

[FFK91] S. Fenner, L. Fortnow, and S. Kurtz. Gap-definable counting classes. In *Proc. 6th Annual IEEE Conference on Structure in Complexity Theory*, pages 30–42, 1991.

[FFL93] S. Fenner, L. Fortnow, and L. Li. Gap-definability as a closure property. In *Proc. 10th Annual Symposium on Theoretical Aspects of Computer Science*, volume 665 of *Lect. Notes in Comp. Sci.*, pages 484–493. Springer Verlag, 1993.

[FR91] L. Fortnow and N. Reingold. PP is closed under truth-table reductions. In *Proc. 6th Annual IEEE Conference on Structure in Complexity Theory*, pages 13–15, 1991.

[FSS84] M. Furst, J. Saxe, and M. Sipser. Parity, circuits, and the polynomial-time hierarchy. *Math. Sys. Thy.*, 17:13–27, 1984.

[FSV93] R. Fagin, L. Stockmeyer, and M. Vardi. On monadic NP vs. monadic co-NP. In *Proc. 8th Annual IEEE Conference on Structure in Complexity Theory*, pages 19–30, 1993.

[GHR92] M. Goldmann, J. Håstad, and A. Razborov. Majority gates vs. general weighted threshold gates. *Computational Complexity*, 2:277–300, 1992.

[GJ79] M. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, 1979.

[GKR$^+$95] F. Green, J. Köbler, K. Regan, T. Schwentick, and J. Torán. The power of the middle bit of a #P function. *J. Comp. Sys. Sci.*, 50:456–467, 1995.

[GKT92] F. Green, J. Köbler, and J. Torán. The power of the middle bit. In *Proc. 7th Annual IEEE Conference on Structure in Complexity Theory*, pages 111–117, 1992.

[Gre95] F. Green. Lower bounds for depth-three circuits with equals and mod-gates. In *Proc. 12th Annual Symposium on Theoretical Aspects of Computer Science*, volume 900 of *Lect. Notes in Comp. Sci.* Springer Verlag, 1995.

[Gup93] S. Gupta. On isolating an odd number of elements and its applications to complexity theory. Technical Report OSU-CISRC-6/93-TR24, Dept. of Comp. Sci., Ohio State University, 1993.

[HG91] J. Håstad and M. Goldmann. On the power of small-depth threshold circuits. *Computational Complexity*, 1:113–129, 1991.

[HHK91] T. Hofmeister, W. Hohberg, and S. Köhling. Some notes on threshold circuits and multiplication in depth 4. In *Proc. 8th International Conference on Fundamentals of Computation Theory*, pages 230–239, 1991.

[HMP$^+$87] A. Hajnal, W. Maass, P. Pudlák, M. Szegedy, and G. Turán. Threshold circuits of bounded depth. In *Proc. 28th Annual IEEE Symposium on Foundations of Computer Science*, pages 99–110, 1987.

[Hof96] T. Hofmeister. A note on the simulation of exponential threshold weights. In *Proc. 2nd International Computing and Combinatorics Conference (COCOON'96)*, volume 1090 of *Lect. Notes in Comp. Sci.*, pages 136–141. Springer Verlag, 1996.

[Jac51] N. Jacobson. *Lectures in Abstract Algebra, Vols. 1–3.* Van Nostrand, 1951.

[Mac95] A. Maciel. *Threshold Circuits of Small Majority-Depth.* PhD thesis, McGill University, School of Computer Science, 1995.

[MP68] M. Minsky and S. Papert. *Perceptrons.* MIT Press, 1968. Revised and expanded in 1988.

[MP92] D. Muller and F. Preparata. Parallel restructuring and evaluation of expressions. *J. Comp. Sys. Sci.*, 44:43–62, 1992.

[MPT91] P. McKenzie, P. Péladeau, and D. Thérien. NC^1: The automata-theoretic viewpoint. *Computational Complexity*, 1:330–359, 1991.

[MS77] F. MacWilliams and N. Sloane. *The Theory of Error-Correcting Codes.* North-Holland, Amsterdam, 1977.

[MT93] A. Maciel and D. Thérien. Threshold circuits for iterated mutliplication: using AC^0 for free. In *Proc. 10th Annual Symposium on Theoretical Aspects of Computer Science*, volume 665 of *Lect. Notes in Comp. Sci.*, pages 545–554. Springer Verlag, 1993.

[MV94] M. Mahajan and V. Vinay. Non-commutative computation, depth reduction, and skew circuits. In *Proc. 14th Annual Conference on Foundations of Software Technology and Theoretical Computer Science*, volume 880 of *Lect. Notes in Comp. Sci.* Springer Verlag, 1994.

[New64] D. Newman. Rational approximation to $|x|$. *Michigan Mathematical Journal*, 11:11–14, 1964.

[Nis91] N. Nisan. Lower bounds for non-commutative computation: extended abstract. In *Proc. 23rd Annual ACM Symposium on the Theory of Computing*, pages 410–418, 1991.

[NN93] J. Naor and M. Naor. Small-bias probability spaces: efficient constructions and applications. *SIAM J. Comput.*, 22:838–856, 1993.

[NRS95] A. Naik, K. Regan, and D. Sivakumar. On quasilinear time complexity theory. *Theor. Comp. Sci.*, 148:325–349, 1995.

[NS92] N. Nisan and M. Szegedy. On the degree of Boolean functions as real polynomials. In *Proc. 24th Annual ACM Symposium on the Theory of Computing*, pages 462–467, 1992.

[NS94] N. Nisan and M. Szegedy. On the degree of Boolean functions as real polynomials. *Computational Complexity*, 4:301–313, 1994.

[Ogi95] M. Ogihara. The PL hierarchy collapses. Technical Report UR CS TR 587, Department of Computer Science, University of Rochester, June 1995.

[Pat92] R. Paturi. On the degree of polynomials that approximate symmetric Boolean functions. In *Proc. 24th Annual ACM Symposium on the Theory of Computing*, pages 468–474, 1992.

[Raz87] A. Razborov. Lower bounds for the size of circuits of bounded depth with basis $\{\wedge, \oplus\}$. *Mathematical Notes, (formerly of the Academy of Natural Sciences of the USSR)*, 41:333–338, 1987.

[RS92] K. Regan and T. Schwentick. On the power of one bit of a #p function. In *Proc. 4th Annual Italian Conference on Theoretical Computer Science*, pages 317–329. World Scientific, Singapore, 1992.

[RW93] A. Razborov and A. Wigderson. $n^{\Omega(\log n)}$ lower bounds on the size of depth-3 threshold circuits with AND gates at the bottom. *Inf. Proc. Lett.*, 45:303–307, 1993.

[Sch80] J.T. Schwartz. Fast probabilistic algorithms for polynomial identities. *J. Assn. Comp. Mach.*, 27:701–717, 1980.

[Sch94] T. Schwentick. Graph connectivity and monadic NP. In *Proc. 35th Annual IEEE Symposium on Foundations of Computer Science*, pages 614–622, 1994.

[She93] B.-Z. Shen. A Justesen construction of binary concatenated codes than asymptotically meet the Zyablov bound for low rate. *IEEE Transactions on Information Theory*, 39(1):239–242, January 1993.

[Smo87] R. Smolensky. Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In *Proc. 19th Annual ACM Symposium on the Theory of Computing*, pages 77–82, 1987.

[Smo93] R. Smolensky. On representations by low-degree polynomials. In *Proc. 34th Annual IEEE Symposium on Foundations of Computer Science*, pages 130–138, 1993.

[Spi71] M. Spira. On time-hardware complexity tradeoffs for Boolean functions. In *Proceedings of the Fourth International Symposium on Systems Sciences*, pages 525–527, North Hollywood, California, 1971. Western Periodicals.

[Spi95] D. Spielman. Linear-time encodable and decodable error-correcting codes. In *Proc. 27th Annual ACM Symposium on the Theory of Computing*, pages 388–397, 1995.

[Sud92] M. Sudan. *Efficient checking of polynomials and proofs and the hardness of approximation problems.* PhD thesis, University of California, Berkeley, 1992.

[Sze90] M. Szegedy. Functions with bounded symmetric communication complexity and circuits with mod m gates. In *Proc. 22nd Annual ACM Symposium on the Theory of Computing*, pages 278–286, 1990.

[Sze93] M. Szegedy. Functions with bounded symmetric communication complexity, programs over commutative monoids, and ACC. *J. Comp. Sys. Sci.*, 47:405–423, 1993.

[Tar91] J. Tarui. Randomized polynomials, threshold circuits, and the polynomial hierarchy. In *Proc. 8th Annual Symposium on Theoretical Aspects of Computer Science*, volume 480 of *Lect. Notes in Comp. Sci.*, pages 238–250. Springer Verlag, 1991.

[Tar93] J. Tarui. Probabilistic polynomials, AC^0 functions, and the polynomial-time hierarchy. *Theor. Comp. Sci.*, 113:167–183, 1993.

[TB95] G. Tardos and D. Mix Barrington. A lower bound on the mod 6 degree of the OR function. In *Proceedings of the Third Israel Symposium on the Theory of Computing and Systems (ISTCS'95), Tel Aviv, Israel*, pages 52–56. IEEE Computer Society Press, 1995.

[Tod89] S. Toda. On the computational power of PP and $\oplus$P. In *Proc. 30th Annual IEEE Symposium on Foundations of Computer Science*, pages 514–519, 1989.

[Tod91] S. Toda. PP is as hard as the polynomial-time hierarchy. *SIAM J. Comput.*, 20:865–877, 1991.

[Tsa93] S.-C. Tsai. Lower bounds on representing Boolean functions as polynomials in Z_m. In *Proc. 8th Annual IEEE Conference on Structure in Complexity Theory*, pages 96–101, 1993.

[VV86] L. Valiant and V. Vazirani. NP is as easy as detecting unique solutions. *Theor. Comp. Sci.*, 47:85–93, 1986.

[Wig94] A. Wigderson. NL/*poly* $\subseteq$ $\oplus$L/*poly*. In *Proc. 9th Annual IEEE Conference on Structure in Complexity Theory*, pages 59–62, 1994.

[Yao90] A. Yao. On ACC and threshold circuits. In *Proc. 31st Annual IEEE Symposium on Foundations of Computer Science*, pages 619–627, 1990.

[Yao94] A. Yao. Decision tree complexity and Betti numbers. In *Proc. 26th Annual ACM Symposium on the Theory of Computing*, pages 615–624, 1994.

[Zip79] R. Zippel. Probabilistic algorithms for sparse polynomials. In Edward W. Ng, editor, *Proceedings of the International Symposium on Symbolic and Algebraic Manipulation (EUROSAM '79)*, volume 72 of *Lect. Notes in Comp. Sci.*, pages 216–226, Marseilles, June 1979. Springer Verlag.

Index