

On the Power of Small-Depth Computation

Emanuele Viola¹

November 12, 2009

¹Supported by NSF grant CCF-0845003. Email: viola@ccs.neu.edu

Abstract

In this work we discuss selected topics on small-depth computation, presenting a few unpublished proofs along the way. The four chapters contain:

1. A unified treatment of the challenge of exhibiting explicit functions that have small correlation with low-degree polynomials over $\{0, 1\}$.
2. An unpublished proof that small bounded-depth circuits (AC^0) have exponentially small correlation with the parity function. The proof is due to Klivans and Vadhan; it builds upon and simplifies previous ones.
3. Valiant's simulation of log-depth linear-size circuits of fan-in 2 by sub-exponential size circuits of depth 3 and unbounded fan-in. To our knowledge, a proof of this result has never appeared in full.
4. Applebaum, Ishai, and Kushilevitz's cryptography in bounded depth.

Contents

0.1	Introduction	2
1	Polynomials over $\{0, 1\}$	4
1.1	Introduction	4
1.2	Correlation bounds	5
1.2.1	Large degree $d \gg \log n$ but noticeable correlation $\epsilon \gg 1/n$	7
1.2.2	Negligible correlation $\epsilon \ll 1/n$ but small degree $d \ll \log n$	10
1.2.3	Symmetric functions correlate well with degree $O(\sqrt{n})$	14
1.2.4	Other works	15
1.3	Pseudorandom generators vs. correlation bounds	15
1.4	Conclusion	19
2	The correlation of parity with small-depth circuits	20
2.1	Introduction	20
2.2	Stage 2: From output-majority circuits to Theorem 12	21
2.2.1	Proof of Theorem 12 assuming Theorem 14	22
2.3	Stage 1: Output-majority circuits cannot compute parity	24
2.3.1	Step 1: Sign-approximating output-majority circuits	25
2.3.2	Proof of Lemma 19	27
2.3.3	Step 2: From sign-approximating to weakly computing	28
2.3.4	Step 3	30
2.3.5	Putting the three steps together	31
3	Logarithmic depth vs. depth 3	32
3.1	Exponential lower bounds for depth 2	32
3.2	From logarithmic depth to depth 3	33
4	Cryptography in bounded depth	38
4.1	Definitions and main result	38
4.1.1	Preliminaries on randomized encodings	42
4.1.2	Encoding branching programs by degree-3 polynomials	43
4.1.3	Encoding polynomials locally	48
4.1.4	Putting the encodings together	49

0.1 Introduction

The NP-completeness of SAT is a celebrated example of the power of bounded-depth computation: the core of the argument is a depth reduction establishing that any small non-deterministic circuit – an arbitrary NP computation on an arbitrary input – can be simulated by a small non-deterministic circuit of depth 2 with unbounded fan-in – a SAT instance.

Many other examples permeate theoretical computer science. In this work we discuss a selected subset of them, and include a few unpublished proofs.

We start in Chapter 1 with considering low-degree polynomials over the fields with two elements $\{0, 1\}$ (a.k.a. $\text{GF}(2)$). Polynomials are a bounded-depth computational model: they correspond to depth-2 unbounded fan-in circuits whose output gate is a sum (in our case, modulo 2). Despite the apparent simplicity of the model, a fundamental challenge has resisted decades of attacks from researchers: exhibit explicit functions that have small correlation with low-degree polynomials. This chapter is a unified treatment of the state-of-the-art on this challenge. We discuss long-standing results and recent developments, related proof techniques, and connections with pseudorandom generators. We also suggest several research directions. Along the way, we present previously unpublished proofs of certain correlation bounds.

In Chapter 2 we consider unbounded fan-in circuits of small depth with \wedge (and), \vee (or), and \neg (not) gates, known as AC^0 . Here we present an unpublished proof of the well-known result that small AC^0 circuits have exponentially small correlation with the parity function. The proof is due to Klivans and Vadhan; it builds upon and simplifies previous ones.

In Chapter 3 we present a depth-reduction result by Valiant [Val77, Val83] whose proof to our knowledge has never appeared in full. The result is that log-depth linear-size circuits of fan-in 2 can be simulated by sub-exponential size circuits of depth 3 and unbounded fan-in (again, the gates are \wedge, \vee, \neg). Although the parameters are more contrived, this result is in the same spirit of the NP-completeness of SAT mentioned at the beginning of this introduction. The latter depth-reduction crucially exploits *non-determinism*; interestingly, we have to work harder to prove Valiant’s *deterministic* simulation.

Finally, in Chapter 4 we present the result by Applebaum, Ishai, and Kushilevitz [AIK06] that shows that, under standard complexity theoretic assumptions, many cryptographic primitives can be implemented in very restricted computational models. Specifically, one can implement those primitives by functions such that each of their output bits only depends on a constant number of input bits. In particular, each output bit can be computed by a circuit of constant size and depth.

Of course, many exciting works on small-depth computation are not covered here. Recent ones include Rossman’s lower bound [Ros08] and the pseudorandom generators for small-depth circuits by Bazzi, Razborov, and Braverman [Bra09].

Publishing note. Chapter 1 appeared in ACM SIGACT News Volume 40, Issue 1 (March 2009). The other chapters are a polished version of the notes of Lectures 4,5,6,7,10,12,13, and

14 of the author's class "Gems of Theoretical Computer Science," taught at Northeastern University in Spring 2009 [Vio09a]. I thank the audience of the class, Rajmohan Rajaraman, and Ravi Sundaram for useful feedback. I am grateful to Aldo Cassola, Dimitrios Kanoulas, Eric Miles, and Ravi Sundaram for scribing the above lectures.

Chapter 1

Polynomials over $\{0, 1\}$

1.1 Introduction

This chapter is about one of the most basic computational models: low-degree polynomials over the field $\{0, 1\} = \text{GF}(2)$. For example, the following is a polynomial of degree 2 in 3 variables

$$p(x_1, x_2, x_3) := x_1 \cdot x_2 + x_2 + x_3 + 1,$$

given by the sum of the 4 monomials x_1x_2 , x_2 , x_3 , and 1, of degree 2, 1, 1, and 0, respectively. This polynomial computes a function from $\{0, 1\}^3$ to $\{0, 1\}$, which we also denote p , by performing the arithmetic over $\{0, 1\}$. Thus the sum “+” is modulo 2 and is the same as “xor,” while the product “.” is the same as “and.” For instance, $p(1, 1, 0) = 1$. Being complexity theorists rather than algebraists, we are only interested in the function computed by a polynomial, not in the polynomial itself; therefore we need not bother with variables raised to powers bigger than 1, since for $x \in \{0, 1\}$ one has $x = x^2 = x^3$ and so on. In general, a polynomial p of degree d in n Boolean variables $x_1, \dots, x_n \in \{0, 1\}$ is a sum of monomials of degree at most d :

$$p(x_1, \dots, x_n) = \sum_{M \subseteq \{1, \dots, n\}, |M| \leq d} c_M \prod_{i \in M} x_i,$$

where $c_M \in \{0, 1\}$ and we let $\prod_{i \in \emptyset} x_i := 1$; such a polynomial p computes a function $p : \{0, 1\}^n \rightarrow \{0, 1\}$, interpreting again the sum modulo 2. We naturally measure the complexity of a polynomial by its degree d : the maximum number of variables appearing in any monomial. Since every function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ can be computed by a polynomial of degree n , specifically $f(x_1, \dots, x_n) = \sum_{a_1, \dots, a_n} f(a_1, \dots, a_n) \prod_{1 \leq i \leq n} (1 + a_i + x_i)$, we are interested in polynomials of low degree $d \ll n$.

Low-degree polynomials constitute a fundamental model of computation that arises in a variety of contexts, ranging from error-correcting codes to circuit lower bounds. As for any computational model, a first natural challenge is to exhibit explicit functions that cannot be computed in the model. This challenge is easily won: the monomial $\prod_{i=1}^d x_i$ requires degree d . A second, natural challenge has baffled researchers, and is the central topic of

this chapter. One now asks for functions that not only cannot be computed by low-degree polynomials, but do not even *correlate* with them.

1.2 Correlation bounds

We start by defining the correlation between a function f and polynomials of degree d . This quantity captures how well we can approximate f by polynomials of degree d , and is also known as the average-case hardness of f against polynomials of degree d .

Definition 1 (Correlation). *Let $f : \{0, 1\}^* \rightarrow \{0, 1\}$ be a function, n an integer, and D a distribution on $\{0, 1\}^n$. We define the correlation between f and a polynomial $p : \{0, 1\}^n \rightarrow \{0, 1\}$ with respect to D as*

$$\text{Cor}_D(f, p) := \left| \Pr_{x \sim D}[f(x) = p(x)] - \Pr_{x \sim D}[f(x) \neq p(x)] \right| = 2 \left| 1/2 - \Pr_{x \sim D}[f(x) \neq p(x)] \right| \in [0, 1].$$

We define the correlation between f and polynomials of degree d with respect to D as

$$\text{Cor}_D(f, d) := \max_p \text{Cor}_D(f, p) \in [0, 1],$$

where the maximum is over all polynomials $p : \{0, 1\}^n \rightarrow \{0, 1\}$ of degree d .

Unless specified otherwise, D is the uniform distribution and we simply write $\text{Cor}(f, d)$.

For more than two decades, researchers have sought to exhibit explicit functions that have small correlation with high-degree polynomials. We refer to this enterprise as obtaining, or proving, “correlation bounds.” A dream setting of parameters would be to exhibit a function $f \in \mathcal{P}$ such that for every n , and for D the uniform distribution over $\{0, 1\}^n$, $\text{Cor}_D(f, \epsilon \cdot n) \leq \exp(-\epsilon \cdot n)$, where $\epsilon > 0$ is an absolute constant, and $\exp(x) := 2^x$. For context, we mention that a random function satisfies such strong correlation bounds, with high probability.

The original motivation for seeking correlation bounds comes from circuit complexity, because functions with small correlation with polynomials require large constant-depth circuits of certain types, see e.g. [Raz87, Smo87, HMP⁺93, Bei93]. An additional motivation comes from pseudorandomness: as we will see, sufficiently strong correlation bounds can be used to construct pseudorandom generators [Nis91, NW94], which in turn have myriad applications. But as this article also aims to put forth, today the challenge of proving correlation bounds is interesting per se, and its status is a fundamental benchmark for our understanding of complexity theory: it is not known how to achieve the dream setting of parameters mentioned above, and in fact nobody can even achieve the following strikingly weaker setting of parameters.

Open question 1. *Is there a function $f \in \text{NP}$ such that for arbitrarily large n there is a distribution D on $\{0, 1\}^n$ with respect to which $\text{Cor}_D(f, \log_2 n) \leq 1/n$?*

Before discussing known results in the next sections, we add to the above concise motivation for tackling correlation bounds the following discussion of their relationship with other open problems.

Correlation bounds’ place in the hierarchy of open problems. We point out that a negative answer to Question 1 implies that NP has circuits of quasipolynomial size $s = n^{O(\log n)}$. This relatively standard fact can be proved via boosting [Fre95, Section 2.2] or min-max/linear-programming duality [GHR92, Section 5]. Thus, an affirmative answer to Question 1 is necessary to prove that NP does not have circuits of quasipolynomial size, a leading goal of theoretical computer science. Of course, this connection can be strengthened in various ways, for example noting that the circuits for NP given by a negative answer to Question 1 can be written on inputs of length n as a majority of $n^{O(1)}$ polynomials of degree $\log_2 n$; thus, an affirmative answer to Question 1 is necessary even to prove that NP does not have circuits of the latter type. On the other hand, Question 1 cannot easily be related to polynomial-size lower bounds such as $\text{NP} \not\subseteq \text{P/poly}$, because a polynomial of degree $\log n$ may have a quasipolynomial number of monomials.

While there are many other open questions in complexity theory, arguably Question 1 is among those having remarkable yet not dramatic consequences, and therefore should be attacked first. To illustrate, let us consider some of the major open problems in the area of unbounded-fan-in constant-depth circuits AC^0 . One such problem is to exhibit an explicit function that requires AC^0 circuits of depth d and size $s \geq \exp(n^\epsilon)$ for some $\epsilon \gg 1/d$ (current lower bounds give $\epsilon = O(1/d)$, see [Hås87]). However, via a guess-and-verify construction usually credited to [Nep70], one can show that any function $f \in \text{NL}$ has AC^0 circuits of depth d and size $\exp(n^{c/d})$ where c depends only on f . This means that a strong enough progress on this problem would separate NP from NL. Furthermore, a result by Valiant we present in Chapter 3 entails that improving the known lower bounds for AC^0 circuits of depth 3 to size $s = \exp(\Omega(n))$ would result in a super-linear size lower bound for (fan-in 2) circuits of logarithmic depth. On the other hand, even an answer to Question 1 with strong parameters is not known to have such daunting consequences, nor, if that is of concern, is known to require “radically new” ideas [BGS75, RR97, AW08].

Progress on Question 1 is also implied by a corresponding progress in number-on-forehead communication complexity. Specifically, a long-standing open question in communication complexity is to exhibit an explicit function $f : (\{0, 1\}^n)^k \rightarrow \{0, 1\}$ that cannot be computed by number-on-forehead k -party protocols exchanging $O(k)$ bits, for some $k \geq \log_2 n$ [KN97, Problem 6.21]. A lower bound on the communication required to compute some function f is usually proved by establishing that f has low correlation with k -party protocols – a technique also known as the discrepancy method, cf. [KN97, VW08]. The connection with polynomials is given by a beautiful observation of Håstad and Goldmann [HG91, Proof of Lemma 4], which implies that if f has low correlation with k -party protocols then f also has low correlation with polynomials of degree $d := k - 1$. But the converse connection is not known.

Finally, we note that polynomials over $\{0, 1\}$ constitute a simple model of algebraic computation, and so Question 1 can also be considered a basic question in algebraic complexity. In fact, an interesting special case – to which we will return in §1.2.2, §1.2.4 – is whether one can take f in Question 1 to be an explicit low-degree polynomial over $\{0, 1\}$.

After this high-level discussion, we now move to presenting the known correlation bounds.

It is a remarkable state of affairs that, while we are currently unable to make the correlation small and the degree large *simultaneously*, as required by Question 1, we can make the correlation small and the degree large *separately*. And in fact we can even achieve this for the same explicit function $f = \text{mod}_3$. We examine these two types of results in turn.

1.2.1 Large degree $d \gg \log n$ but noticeable correlation $\epsilon \gg 1/n$

Razborov [Raz87] (also in [CK02, Section 2.7.1]) proves the existence of a symmetric function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ that has correlation at most $1 - 1/n^{O(1)}$ with polynomials of degree $\Omega(\sqrt{n})$ (a function is symmetric when its value only depends on the number of input bits that are ‘1’).

Smolensky [Smo87] obtains a refined bound for the explicit function $\text{mod}_3 : \{0, 1\}^n \rightarrow \{0, 1\}$ which evaluates to 1 if and only if the number of input bits that are ‘1’ is of the form $3k + 1$ for some integer k , i.e., it is congruent to 1 modulo 3:

$$\text{mod}_3(x_1, \dots, x_n) = 1 \Leftrightarrow \sum_i x_i = 1 \pmod{3}.$$

For example, $\text{mod}_3(1, 0, 0) = \text{mod}_3(0, 1, 0) = 1 \neq \text{mod}_3(1, 0, 1)$.

Theorem 2 ([Smo87]). *For any n that is divisible by 3, and for U the uniform distribution over $\{0, 1\}^n$, $\text{Cor}_U(\text{mod}_3, \epsilon\sqrt{n}) \leq 2/3$, where $\epsilon > 0$ is an absolute constant.*

While the proof of Smolensky’s result has appeared several times, e.g. [Smo87, BS90, Bei93, AB09], we are unaware of a source that directly proves Theorem 2, and thus we include next a proof for completeness (the aforementioned sources either focus on polynomials over the field with three elements, or prove the bound for one of the three functions $\text{mod}_{i,3}(x_1, \dots, x_n) = 1 \Leftrightarrow \sum_i x_i = i \pmod{3}$ for $i = 0, 1, 2$).

Proof. The idea is to consider the set of inputs $X \subseteq \{0, 1\}^n$ where the polynomial computes the mod_3 function correctly, and use the polynomial to represent any function defined on X by a polynomial of degree $n/2 + d$. This means that the number of functions defined on X should be smaller than the number of polynomials of degree $n/2 + d$, which leads to the desired tradeoff between $|X|$ and d . To carry through this argument, one works over a field F that extends $\{0, 1\}$.

We start by noting that, since n is divisible by 3, one has

$$\sum_i x_i = 2 \pmod{3} \Leftrightarrow \sum_i 1 - x_i = 1 \pmod{3} \Leftrightarrow \text{mod}_3(1 + x_1, \dots, 1 + x_n) = 1, \quad (1.1)$$

where the sums $1 + x_i$ in the input to mod_3 are modulo 2. Let F be the field of size 4 that extends $\{0, 1\}$, which we can think of as $F = \{0, 1\}[t]/(t^2 + t + 1)$: the set of polynomials over $\{0, 1\}$ modulo the irreducible polynomial $t^2 + t + 1$. Note that $t \in F$ has order 3, since $t^2 = t + 1 \neq 1$, while $t^3 = t^2 + t = 1$. Let $h : \{1, t\} \rightarrow \{0, 1\}$ be the ‘change of domain’ linear map $h(\alpha) := (\alpha + 1)/(t + 1)$; this satisfies $h(1) = 0$ and $h(t) = 1$.

Observe that for every $y \in \{1, t\}^n$ we have, using Equation (1.1):

$$y_1 \cdots y_n = 1 + (t+1) \cdot \text{mod}_3(h(y_1), \dots, h(y_n)) + (t^2+1) \cdot \text{mod}_3(1+h(y_1), \dots, 1+h(y_n)). \quad (1.2)$$

Now fix any polynomial $p : \{0, 1\}^n \rightarrow \{0, 1\}$ and let

$$\Pr_{x \in \{0, 1\}^n} [p(x) \neq \text{mod}_3(x)] =: \delta,$$

which we aim to bound from below. Let $p' : \{1, t\}^n \rightarrow F$ be the polynomial

$$p'(y_1, \dots, y_n) := 1 + (t+1) \cdot p(h(y_1), \dots, h(y_n)) + (t^2+1) \cdot p(1+h(y_1), \dots, 1+h(y_n));$$

note p' has the same degree d of p . By the definition of p' and δ , a union bound, and Equation (1.2) we see that

$$\Pr_{y \in \{1, t\}^n} [y_1 \cdots y_n = p'(y_1, \dots, y_n)] \geq 1 - 2\delta. \quad (1.3)$$

Now let $S \subseteq \{1, t\}^n$ be the set of $y \in \{1, t\}^n$ such that $y_1 \cdots y_n = p'(y_1, \dots, y_n)$; we have just shown that $|S| \geq 2^n(1 - 2\delta)$. Any function $f : S \rightarrow F$ can be written as a polynomial over F where no variable is raised to powers bigger than 1: $f(y_1, \dots, y_n) = \sum_{a_1, \dots, a_n} f(a_1, \dots, a_n) \prod_{1 \leq i \leq n} (1 + h(y_i) + h(a_i))$. In any such polynomial we can replace any monomial M of degree $|M| > n/2$ by a polynomial of degree at most $n/2 + d$ as follows, without affecting the value on any input $y \in S$:

$$\prod_{i \in M} y_i = y_1 \cdots y_n \prod_{i \notin M} (y_i(t+1) + t) = p'(y_1, \dots, y_n) \prod_{i \notin M} (y_i(t+1) + t),$$

where the first equality is not hard to verify. Doing this for every monomial we can write $f : S \rightarrow F$ as a polynomial over F of degree $\lfloor n/2 + d \rfloor$.

The number of functions from S to F is $|F|^{|S|}$, while the number of polynomials over F of degree $\lfloor n/2 + d \rfloor$ is $|F|^{\sum_{i=0}^{\lfloor n/2 + d \rfloor} \binom{n}{i}}$. Thus

$$\log_{|F|} \#\text{functions} = |S| = 2^n(1 - 2\delta) \leq \sum_{i=0}^{\lfloor n/2 + d \rfloor} \binom{n}{i} = \log_{|F|} \#\text{polynomials}.$$

Since $d = \epsilon\sqrt{n}$, we have

$$\sum_{i=0}^{\lfloor n/2 + d \rfloor} \binom{n}{i} \leq 2^{n/2 + d} \cdot \binom{n}{\lfloor n/2 \rfloor} \leq 2^{n/2 + \epsilon\sqrt{n}} \cdot \Theta\left(\frac{2^n}{\sqrt{n}}\right) = (1/2 + \Theta(\epsilon))2^n,$$

where the second inequality follows from standard estimates on binomial coefficients. The standard estimate for even n is for example in [CT06, Lemma 17.5.1]; for odd $n = 2k + 1$ one can first note $\binom{n}{\lfloor n/2 \rfloor} = \binom{2k+1}{k} < \binom{2k+2}{k+1} = \binom{n+1}{(n+1)/2}$ and then again apply [CT06, Lemma 17.5.1]. Therefore $1 - 2\delta \leq 1/2 + \Theta(\epsilon)$ and the theorem is proved. \square

The limitation of the argument. There are two reasons why we get a poor correlation bound in the above proof of Theorem 2. The first is the union bound in (1.3), which immediately puts us in a regime where we cannot obtain subconstant correlation. This regime is unavoidable as the polynomial $p = 0$ of degree 0 has constant correlation with mod_3 with respect to the uniform distribution. (Later we will see a different distribution with respect to which mod_3 has vanishing, exponentially small correlation with polynomials of degree $\ll \log n$.) Nevertheless, let us pretend that the union bound in (1.3) is not there. This is not pointless because this step is indeed not present in related correlation bounds, which do however suffer from the second limitation we are about to discuss. The related correlation bounds are those between the parity function

$$\text{parity}(x_1, \dots, x_n) := x_1 + \dots + x_n \qquad \text{parity} : \{0, 1\}^n \rightarrow \{0, 1\}$$

and polynomials over the field with three elements, see e.g. [BS90, AB09], or between the parity function and the sign of polynomials over the integers [ABFR94]. If we assume that the union bound in (1.3) is missing, then we get $2^n(1 - \delta) \leq \sum_{i=0}^{\lfloor n/2+d \rfloor} \binom{n}{i}$. Even if $d = 1$, this only gives $1 - \delta \leq 1/2 + \Theta(1/\sqrt{n})$, which means that the correlation is $O(1/\sqrt{n})$: this argument does not give a correlation bound of the form $o(1/\sqrt{n})$. More generally, to our knowledge Question 1 is also open when replacing $1/n$ with $1/\sqrt{n}$.

Xor lemma. A striking feature of the above results ([Raz87] and Theorem 2) is that they prove non-trivial correlation bounds for polynomials of very high degree $d = n^{\Omega(1)}$. In this sense these results address the computational model which is the subject of Question 1, they “just” fail to provide a strong enough bound on the correlation. For other important computational models this would not be a problem: the extensive study of *hardness amplification* has developed many techniques to improve correlation bounds in the following sense: given an explicit function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ that has correlation ϵ with some class \mathcal{C}_n of functions on n bits, construct another explicit function $f' : \{0, 1\}^{n'} \rightarrow \{0, 1\}$, where $n' \approx n$, that has correlation $\epsilon' \ll \epsilon$ with a closely related class $\mathcal{C}_{n'}$ of functions on n' bits (see [SV08] for a comprehensive list of references to research in hardness amplification). While the following discussion holds for any hardness amplification, for concreteness we focus on the foremost: Yao’s xor lemma. Here $f' : (\{0, 1\}^n)^k \rightarrow \{0, 1\}$ is defined as the xor (or parity, or sum modulo 2) of k independent outputs of f :

$$f'(x^1, \dots, x^k) := f(x^1) + \dots + f(x^k) \in \{0, 1\}, \qquad x^i \in \{0, 1\}^n.$$

The compelling intuition is that, since functions from \mathcal{C}_n have correlation at most ϵ with f , and f' is the xor of k independent evaluations of f , the correlation should decay exponentially with k : $\epsilon' \approx \epsilon^k$. This is indeed the case if one tries to compute $f'(x^1, \dots, x^k)$ as $g_1(x^1) + \dots + g_k(x^k)$ where $g_i : \{0, 1\}^n \rightarrow \{0, 1\}, g_i \in \mathcal{C}_n, 1 \leq i \leq k$, but in general a function $g : (\{0, 1\}^n)^k \rightarrow \{0, 1\}, g \in \mathcal{C}_{n'}$, need not have this structure, making proofs of Yao’s xor lemma more subtle. If we could prove this intuition true for low-degree polynomials, we could combine this with Theorem 2 to answer affirmatively Question 1 via the function

$$f(x^1, \dots, x^k) := \text{mod}_3(x^1) + \dots + \text{mod}_3(x^k) \tag{1.4}$$

for $k = n$. Of course the obstacle is that nobody knows whether Yao's xor lemma holds for polynomials.

Open question 2. *Does Yao's xor lemma hold for polynomials of degree $d \geq \log_2 n$? For example, let $f : \{0, 1\}^n \rightarrow \{0, 1\}$ satisfy $\text{Cor}(f, n^{1/3}) \leq 1/3$, and for $n' := n^2$ define $f' : \{0, 1\}^{n'} \rightarrow \{0, 1\}$ as $f'(x^1, \dots, x^n) := f(x^1) + \dots + f(x^n)$. Is $\text{Cor}(f', \log_2 n') \leq 1/n'$?*

We now discuss why, despite the many alternative proofs of Yao's xor lemma that are available (e.g., [GNW95]), we cannot apply any of them to the computational model of low-degree polynomials. To prove that f' has correlation at most ϵ' with some class of functions, all known proofs of the lemma need (a slight modification of) the functions in the class to compute the majority function on about $1/\epsilon'$ bits. However, the majority function on $1/\epsilon'$ bits requires polynomials of degree $\Omega(1/\epsilon')$. This means that known proofs can only establish correlation bounds $\epsilon' \gg 1/n$, failing to answer Question 2. More generally, the works [Vio06a, SV08] show that computing the majority function on $1/\epsilon'$ bits is necessary for a central class of hardness amplification proofs.

An xor lemma is however known for polynomials of small degree $d \ll \log n$ [VW08] (this and the other results on polynomials in [VW08] appeared also in [Vio06b]). In general, the picture for polynomials of small degree is different, as we now describe.

1.2.2 Negligible correlation $\epsilon \ll 1/n$ but small degree $d \ll \log n$

It is easy to prove exponentially small correlation bounds for polynomials of degree 1, for example the *inner product* function $\text{IP} : \{0, 1\}^n \rightarrow \{0, 1\}$, defined for even n as

$$\text{IP}(x_1, \dots, x_n) := x_1 \cdot x_2 + x_3 \cdot x_4 + \dots + x_{n-1} \cdot x_n, \quad (1.5)$$

satisfies $\text{Cor}(\text{IP}, 1) = 2^{-n/2}$. Already obtaining exponentially small bounds for polynomials of constant degree appears to be a challenge. The first such bounds come from the famed work by Babai, Nisan, and Szegedy [BNS92] proving exponentially small correlation bounds between polynomials of degree $d := \epsilon \log n$ and, for $k := d + 1$, the *generalized inner product* function $\text{GIP}_k : \{0, 1\}^n \rightarrow \{0, 1\}$,

$$\text{GIP}_k(x_1, \dots, x_n) := \prod_{i=1}^k x_i + \prod_{i=k+1}^{2k} x_i + \dots + \prod_{i=n-k+1}^n x_i,$$

assuming for simplicity that n is a multiple of k . The intuition for this correlation bound is precisely that behind Yao's xor lemma (cf. §1.2.1): (i) any polynomial of degree d has correlation that is bounded away from 1 with any monomial of degree $k = d + 1$ in the definition of GIP, and (ii) since the monomials in the definition of GIP are on disjoint sets of variables, the correlation decays exponentially. (i) is not hard to establish formally. With some work, (ii) can also be established to obtain the following theorem.

Theorem 3 ([BNS92]). *For every n, d , $\text{Cor}(\text{GIP}_{d+1}, d) \leq \exp(-\Omega(n/4^d \cdot d))$.*

When $k \gg \log n$, GIP is almost always 0 on a uniform input, and thus GIP is not a candidate for having small correlation with respect to the uniform distribution with polynomials of degree $d \gg \log n$.

Our exposition of the results in [BNS92] differs in multiple ways from the original. First, [BNS92] does not discuss polynomials but rather number-on-forehead multiparty protocols. The results for polynomials are obtained via the observation of Håstad and Goldmann [HG91, Proof of Lemma 4] mentioned in the subsection “Correlation bounds’ place in the hierarchy of open problems” of §1.2. Second, [BNS92] presents the proof with a different language. Alternative languages have been put forth in a series of papers [CT93, Raz00, VW08], with the last one stressing the above intuition and the connections between multiparty protocols and polynomials.

Bourgain [Bou05] later proves bounds similar to those in Theorem 3 but for the mod_3 function. A minor mistake in his proof is corrected by F. Green, Roy, and Straubing [GRS05].

Theorem 4 ([Bou05, GRS05]). *For every n, d there is a distribution D on $\{0, 1\}^n$ such that $\text{Cor}_D(\text{mod}_3, d) \leq \exp(-n/c^d)$, where c is an absolute constant.*

A random sample from the distribution D in Theorem 4 is obtained as follows: toss a fair coin, if “heads” then output a uniform $x \in \{0, 1\}^n$ such that $\text{mod}_3(x) = 1$, if “tails” then output a uniform $x \in \{0, 1\}^n$ such that $\text{mod}_3(x) = 0$. The value $c = 8$ in [Bou05, GRS05] is later improved to $c = 4$ in [VW08, Cha07]. [VW08] also presents the proof in a different language.

Theorem 4 appears more than a decade after Theorem 3. However, Noam Nisan (personal communication) observes that in fact the first easily follows from the latter.

Sketch of Nisan’s proof of Theorem 4. Grolmusz’s [Gro95] extends the results in [BNS92] and shows that there is a distribution D' on $\{0, 1\}^n$ such that for $k := d + 1$ the function

$$\text{mod}_3 \wedge_k(x_1, \dots, x_n) := \text{mod}_3 \left(\prod_{i=1}^k x_i, \prod_{i=k+1}^{2k} x_i, \dots, \prod_{i=n-k+1}^n x_i \right)$$

has correlation $\exp(-n/c^d)$ with polynomials of degree d , for an absolute constant c . A proof of this can also be found in [VW08, §3.3]. An inspection of the proof reveals that, with respect to another distribution D'' on $\{0, 1\}^n$, the same bound applies to the function

$$\text{mod}_3 \text{mod}_2(x_1, \dots, x_n) := \text{mod}_3(x_1 + \dots + x_k, x_{k+1} + \dots + x_{2k}, \dots, x_{n-k+1} + \dots + x_n)$$

where we replace “and” with “parity” (the sums in the input to mod_3 are modulo 2).

Now consider the distribution D on $\{0, 1\}^{n/k}$ that D'' induces on the input to mod_3 of length n/k . (To sample from D one can sample from D'' , perform the n/k sums modulo 2, and return the string of length n/k .) Suppose that a polynomial $p(y_1, \dots, y_{n/k})$ of degree d has correlation ϵ with the mod_3 function with respect to D . Then the polynomial

$$p'(x_1, \dots, x_n) := p(x_1 + \dots + x_k, x_{k+1} + \dots + x_{2k}, \dots, x_{n-k+1} + \dots + x_n)$$

has degree d and correlation ϵ with the mod_3mod_2 function with respect to the distribution D'' on $\{0, 1\}^n$. Therefore $\epsilon \leq \exp(-n/c^d)$. \square

The mod_m functions have recently got even more attention because as shown in [GT07, LMS08] they constitute a counterexample to a conjecture independently made in [GT08] and [Sam07]. The main technical step in the counterarguments in [GT07, LMS08] is to show an upper bound on the correlation between polynomials of degree 3 and the function

$$\text{mod}_{\{4,5,6,7\},8}(x_1, \dots, x_n) := 1 \Leftrightarrow \sum_i x_i \in \{4, 5, 6, 7\} \pmod{8}.$$

The strongest bound is given by Lovett, Meshulam, and Samorodnitsky [LMS08] who prove the following theorem.

Theorem 5 ([LMS08]). *For every n , $\text{Cor}(\text{mod}_{\{4,5,6,7\},8}, 3) \leq \exp(-\epsilon \cdot n)$, where $\epsilon > 0$ is an absolute constant.*

In fact, to disprove the conjecture in [GT08, Sam07] any bound of the form

$$\text{Cor}(\text{mod}_{\{4,5,6,7\},8}, 3) \leq o(1)$$

is sufficient. Such a bound was implicit in the clever 2001 work by Alon and Beigel [AB01]. With an unexpected use of Ramsey's theorem for hypergraphs, they were the first to establish that the parity function has vanishing correlation with constant-degree polynomials over $\{0, 1, 2\}$. A slight modification of their proof gives $\text{Cor}(\text{mod}_{\{4,5,6,7\},8}, 3) \leq o(1)$, and can be found in the paper by B. Green and Tao [GT07].

It is interesting to note that the function $\text{mod}_{\{4,5,6,7\},8}$ is in fact a polynomial of degree 4 over $\{0, 1\}$, the so-called elementary symmetric polynomial of degree 4

$$s_4(x_1, \dots, x_n) := \sum_{1 \leq i_1 < i_2 < i_3 < i_4 \leq n} x_{i_1} \cdot x_{i_2} \cdot x_{i_3} \cdot x_{i_4}.$$

For suitable input lengths, elementary symmetric polynomials of higher degree d are candidates for having small correlation with polynomials of degree less than d . To our knowledge, even $d = n^{\Omega(1)}$ is a possibility.

The “squaring trick.” Many of the results in this section, and all those that apply to degree $d \approx \log n$ (Theorems 3 and 4) use a common technique which we now discuss also to highlight its limitation. The idea is to reduce the challenge of proving a correlation bound for a polynomial of degree d to that of proving related correlation bounds for polynomials of degree $d - 1$, by *squaring*. To illustrate, let $f : \{0, 1\}^n \rightarrow \{0, 1\}$ be any function and $p : \{0, 1\}^n \rightarrow \{0, 1\}$ a polynomial of degree d . Using the extremely convenient notation $e[z] := (-1)^z$, we write the correlation between f and p with respect to the uniform distribution as

$$\text{Cor}(f, p) = \left| \Pr_{x \in \{0,1\}^n} [f(x) = p(x)] - \Pr_{x \in \{0,1\}^n} [f(x) \neq p(x)] \right| = |E_{x \in \{0,1\}^n} e[f(x) + p(x)]|.$$

If we square this quantity, and use that $E_Z[g(Z)]^2 = E_{Z,Z'}[g(Z) \cdot g(Z')]$, we obtain

$$\text{Cor}(f, p)^2 = E_{x,y \in \{0,1\}^n} e[f(x) + f(y) + p(x) + p(y)].$$

Letting now $y = x + h$ we can rewrite this as

$$\text{Cor}(f, p)^2 = E_{x,h \in \{0,1\}^n} e[f(x) + f(x+h) + p(x) + p(x+h)].$$

The crucial observation is now that, for every fixed h , the polynomial $p(x) + p(x+h)$ has degree $d-1$ in x , even though $p(x)$ has degree d . For example, if $d = 2$ and $p(x) = x_1x_2 + x_3$, we have $p(x) + p(x+h) = x_1x_2 + x_3 + (x_1+h_1)(x_2+h_2) + (x_3+h_3) = x_1h_2 + h_1x_2 + h_1h_2 + h_3$, which indeed has degree $d-1 = 1$ in x . Thus we managed to reduce our task of bounding from above $\text{Cor}(f, p)$ to that of bounding from above a related quantity which involves polynomials of degree $d-1$, specifically the average over h of the correlation between the function $f(x) + f(x+h)$ and polynomials of degree $d-1$. To iterate, we apply the same trick, this time coupled with the Cauchy-Schwarz inequality $E[Z]^2 \leq E[Z^2]$:

$$\begin{aligned} \text{Cor}(f, p)^4 &= E_{x,h} e[f(x) + f(x+h) + p(x) + p(x+h)]^2 \\ &\leq E_h [E_x e[f(x) + f(x+h) + p(x) + p(x+h)]^2]. \end{aligned}$$

We can now repeat the argument in the inner expectation, further reducing the degree of the polynomial. After d repetitions, the polynomial p becomes a constant. After one more, a total of $d+1$ repetitions, the polynomial p “disappears” and we are left with a certain expectation involving f , known as the “Gowers norm” of f and introduced independently in [Gow98, Gow01] and in [AKK⁺03]:

$$\text{Cor}(f, p)^{2^{d+1}} \leq E_{x,h_1,h_2,\dots,h_{d+1}} e \left[\sum_{S \subseteq [d+1]} f \left(x + \sum_{i \in S} h_i \right) \right]. \quad (1.6)$$

For interesting functions f , the expectation in the right-hand side of (1.6) can be easily shown to be small, sometimes exponentially in n , yielding correlation bounds. For example, applying this method to the generalized inner product function gives Theorem 3, while a complex-valued generalization of the method can be applied to the mod_3 function to obtain Theorem 4. This concludes the exposition of this technique; see, e.g., [VW08] for more details.

This “squaring trick” for reducing the analysis of a polynomial of degree d to that of an expression involving polynomials of lower degree $d-1$ dates back at least to the work by Weyl at the beginning of the 20th century; for an exposition of the relevant proof by Weyl, as well as pointers to his work, the reader may consult [GR07a]. This method was introduced in computer science by Babai, Nisan, and Szegedy [BNS92], and employed later by various researchers [Gow98, Gow01, Bou05, GT08, VW08] in different contexts.

The obvious limitation of this technique is that, to bound the correlation with polynomials of degree d , it squares the correlation d times; this means that the bound on the correlation

will be $\exp(-n/2^d)$ at best: nothing for degree $d = \log_2 n$. This bound is almost achieved by [BNS92] which gives an explicit function f such that $\text{Cor}(f, d) \leq \exp(-\Omega(n/2^d \cdot d))$. The extra factor of d in the exponent arises because of the different context of multiparty protocols in [BNS92], but a similar argument, given in [VW08], establishes the following stronger bound.

Theorem 6 ([BNS92, VW08]). *There is an explicit $f \in P$ such that for every n and d , and U the uniform distribution over $\{0, 1\}^n$, $\text{Cor}_U(f, d) \leq \exp(-\Omega(n/2^d))$.*

The function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ in Theorem 6 takes as input an index $i \in \{0, 1\}^{en}$ and a seed $s \in \{0, 1\}^{(1-\epsilon)n}$, and outputs the i -th output bit of a certain pseudorandom generator on seed s [NN93] (Theorem 10 in §1.3). The natural question of whether these functions have small correlation with polynomials of degree $d \gg \log_2 n$ has been answered negatively in [VW08] building on the results in [Raz87, GV04, HV06, Hea08]: it can be shown that, for a specific implementation of the generator, the associated function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ satisfies $\text{Cor}(f, \log^c n) \geq 1 - o(1)$ for an absolute constant c . Determining how small c can be is an open problem whose solution might be informative, given that such functions are of great importance, as we will also see in §1.3.

1.2.3 Symmetric functions correlate well with degree $O(\sqrt{n})$

Many of the correlation bounds discussed in §1.2.1 and §1.2.2 are given by functions that are symmetric: their value depends only on the number of input bits that are ‘1.’ In this section we show that any symmetric function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ correlates well with polynomials of degree $O(\sqrt{n})$, matching the degree obtained in Theorem 2 up to constant factors, and excluding symmetric functions from the candidates to the dream setting of parameters $\text{Cor}(f, \epsilon \cdot n) \leq \exp(-\epsilon \cdot n)$. While there is a shortage of such candidates, we point out that techniques in hardness amplification such as [IW97] may be relevant. It also seems worthwhile to investigate whether the result in this section extends to other distributions.

Theorem 7. *For every n , every symmetric function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ satisfies $\text{Cor}(f, c\sqrt{n}) \geq 99/100$, where c is an absolute constant.*

We present below a proof of Theorem 7 that was communicated to us by Avi Wigderson and simplifies an independent argument of ours. It relies on a result by Bhatnagar, Gopalan, and Lipton, stated next, which in turn follows from well-known facts about the divisibility of binomial coefficients by 2, such as Lucas’ theorem.

Lemma 8 (Corollary 2.7 in [BGL06]). *Let $f : \{0, 1\}^n \rightarrow \{0, 1\}$ be a function such that $f(x)$ depends only on the Hamming weight of x modulo 2^ℓ . Then f is computable by a polynomial of degree $d < 2^\ell$*

In §1.2.2 we saw an example of Lemma 8 when we noticed that the function $\text{mod}_{\{4,5,6,7\},8}$ is computable by a polynomial of degree $4 < 2^3 = 8$. This polynomial was symmetric, and more generally the polynomials in Lemma 8 and Theorem 7 can be taken to be symmetric.

Proof of Theorem 7. The idea is to exhibit a polynomial of degree $O(\sqrt{n})$ that computes f on every input of Hamming weight between $n/2 - a\sqrt{n}$ and $n/2 + a\sqrt{n}$; for a suitable constant a this gives correlation 99/100 by a Chernoff bound.

Let a be a sufficiently large universal constant to be determined later, and let 2^ℓ be the smallest power of 2 bigger than $2a\sqrt{n} + 1$, thus $2^\ell \leq c\sqrt{n}$ for a constant c that depends only on a . Now take any function $f' : \{0, 1\}^n \rightarrow \{0, 1\}$ such that (i) f' agrees with f on every input of Hamming weight between $n/2 - a\sqrt{n}$ and $n/2 + a\sqrt{n}$, and (ii) the value of $f'(x)$ depends only on the Hamming weight of x modulo 2^ℓ . Such an f' exists because f is symmetric and we ensured that $2^\ell > 2a\sqrt{n} + 1$.

Applying first Lemma 8 and then a Chernoff bound (e.g., [DP09, Theorem 1.1]) for a sufficiently large constant a , we have for $d < 2^\ell \leq c\sqrt{n}$

$$\text{Cor}(f, d) \geq \text{Cor}(f, f') \geq 99/100,$$

which concludes the proof of the theorem. \square

1.2.4 Other works

There are many papers on correlation bounds we have not discussed. F. Green [Gre04, Theorem 3.10] manages to compute exactly the correlation between the parity function and quadratic ($d = 2$) polynomials over $\{0, 1, 2\}$, which is $(3/4)^{\lceil n/4 \rceil - 1}$. [Gre04] further discusses the difficulties currently preventing an extension of the result to degree $d > 2$ or to polynomials over fields different from $\{0, 1, 2\}$, while [GR07b] studies the structure of quadratic polynomials over $\{0, 1, 2\}$ that correlate with the parity function best.

The work [Vio07] gives an explicit function that, with respect to the uniform distribution over $\{0, 1\}^n$, has correlation $1/n^{\omega(1)}$ with polynomials of arbitrary degree but with at most $n^{\alpha \cdot \log n}$ monomials, for a small absolute constant $\alpha > 0$. This is obtained by combining a switching lemma with Theorem 3, a technique from [RW93]. The result does not answer Question 1 because a polynomial of degree $\log_2 n$ can have $\binom{n}{\log_2 n} \gg n^{\alpha \cdot \log n}$ monomials, and in fact the function in [Vio07] is a polynomial of degree $(0.3) \log_2 n$. For polynomials over $\{0, 1, 2\}$, the same correlation bounds hold for the parity function [Han06].

Other special classes of polynomials, for example symmetric polynomials, are studied in [CGT96, GT06, BEHL08]. We finally mention that many of the works we discussed, such as Theorems 2, 4, and 7 can be sometimes extended to polynomials modulo $m \neq 2$. For example, Theorem 4 extends to polynomials modulo m vs. the mod_q function for any relatively prime m and q , while Theorem 2 extends to polynomials modulo m vs. the mod_q function for any prime m and q not a power of m . We chose to focus on $m = 2$ because it is clean.

1.3 Pseudorandom generators vs. correlation bounds

In this section we discuss pseudorandom generators for polynomials and their connections to correlation bounds. Pseudorandom generators are fascinating algorithms that stretch short

input seeds into much longer sequences that “look random;” naturally, here we interpret “look random” as “look random to polynomials,” made formal in the next definition.

Definition 9 (Generator). *A generator G that fools polynomials of degree $d = d(n)$ with error $\epsilon = \epsilon(n)$ and seed length $s = s(n)$ is an algorithm running in time polynomial in its output length such that for every n : (i) G maps strings of length $s(n)$ to strings of length n , and (ii) for any polynomial $p : \{0, 1\}^n \rightarrow \{0, 1\}$ of degree d we have*

$$\left| \Pr_{S \in \{0,1\}^s} [p(G(S)) = 1] - \Pr_{U \in \{0,1\}^n} [p(U) = 1] \right| \leq \epsilon. \quad (1.7)$$

For brevity, we write $G : \{0, 1\}^s \rightarrow \{0, 1\}^n$ for a generator with seed length $s(n)$.

Ideally, we would like generators that fool polynomials of large degree d with small error ϵ and small seed length s . We discuss below various connections between obtaining such generators and correlation bounds, but first we point out a notable difference: while for correlation bounds we do have results for large degree $d \gg \log n$ (e.g., Theorem 2), we know of no generator that fools polynomials of degree $d \geq \log_2 n$, even with constant error ϵ .

Open question 3. *Is there a generator $G : \{0, 1\}^{n/2} \rightarrow \{0, 1\}^n$ that fools polynomials of degree $\log_2 n$ with error $1/3$?*

While the smaller the error ϵ the better, generators for constant error are already of great interest; for example, a constant-error generator that fools small circuits is enough to derandomize BPP. However, we currently seem to be no better at constructing generators that fool polynomials with constant error than generators with shrinking error, such as $1/n$.

We now discuss the relationship between generators and correlation bounds, and then present the known generators.

From pseudorandomness to correlation. It is easy to see and well-known [NW94] that a generator implies a worst-case lower bound, i.e., an explicit function that cannot be computed by (essentially) the class of functions fooled by the generator. The following simple observation, which does not seem to have appeared before [Vio09b, §3], shows that in fact a generator implies even a correlation bound. We will use it later to obtain new candidates for answering Question 1.

Observation 1. *Suppose that there is a generator $G : \{0, 1\}^{n-\log n-1} \rightarrow \{0, 1\}^n$ that fools polynomials of degree $\log_2 n$ with error $0.5/n$. Then the answer to Question 1 is affirmative.*

Proof sketch. Let D be the distribution on $\{0, 1\}^n$ where a random $x \in D$ is obtained as follows: toss a fair coin, if “heads” then let x be uniformly distributed over $\{0, 1\}^n$, if “tails” then let $x := G(S)$ for a uniformly chosen $S \in \{0, 1\}^{n-\log n-1}$. Define the function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ as $f(x) = 1$ if and only if there is $s \in \{0, 1\}^{n-\log n-1}$ such that $G(s) = x$; f is easily seen to be in NP. It is now a routine calculation to verify that any function $t : \{0, 1\}^n \rightarrow \{0, 1\}$ that satisfies $\text{Cor}_D(f, t) \geq 1/n$ has advantage at least $0.5/n$ in distinguishing the output of the generator from random. Letting t range over polynomials of degree $\log_2 n$ concludes the proof. \square

From correlation to pseudorandomness. The celebrated construction by Nisan and Wigderson [Nis91, NW94] shows that a sufficiently strong correlation bound with respect to the uniform distribution can be used to obtain a generator that fools polynomials. However, to obtain a generator $G : \{0, 1\}^s \rightarrow \{0, 1\}^n$ against polynomials of degree d , [NW94] in particular needs a function f on $m \leq n$ input bits that has correlation at most $1/n$ with polynomials of degree d . Thus, the current correlation bounds are not strong enough to obtain generators for polynomials of degree $d \geq \log_2 n$. It is a pressing open problem to determine whether alternative constructions of generators are possible, ideally based on constant correlation bounds such as Theorem 2. Here, an uncharted direction is to understand which distributions D enable one to construct generators starting from correlation bounds with respect to D .

The Nisan-Wigderson construction is however sharp enough to give non-trivial generators based on the current correlation bounds such as Theorem 3. Specifically, Luby, Veličković, and Wigderson [LVW93, Theorem 2] obtain generators for polynomials that have arbitrary degree but at most $n^{\alpha \cdot \log n}$ terms for a small absolute constant $\alpha > 0$; a different proof of this result appears in the paper [Vio07] which we already mentioned in §1.2.4. Albeit falling short of answering Question 3 (cf. §1.2.4), this generator [LVW93, Theorem 2] does fool polynomials of constant degree. However, its seed length, satisfying $n = s^{O(\log s)}$, has been superseded in this case by recent developments, which we now discuss.

Generators for degree $d \ll \log n$. Naor and Naor [NN93] construct a generator that fools polynomials of degree 1 (i.e., linear) with a seed length that is optimal up to constant factors – a result with a surprising range of applications (cf. references in [BSSVW03]).

Theorem 10 ([NN93]). *There is a generator $G : \{0, 1\}^{O(\log n)} \rightarrow \{0, 1\}^n$ that fools polynomials of degree 1 with error $1/n$.*

Later, Alon et al. [AGHP92] give three alternative constructions. A nice one is $G(a, b)_i := \langle a^i, b \rangle$ where $\langle \cdot, \cdot \rangle$ denotes inner product modulo 2, $a, b \in \{0, 1\}^\ell$ for $\ell = O(\log n)$, and a^i denotes the result of considering a as an element of the field with 2^ℓ elements, and raising it to the power i .

Recent progress by Bogdanov, Lovett, and the author [BV07, Lov08, Vio09b] has given generators for higher degree. The high-level idea in these works is simple: to fool polynomials of degree d , just sum together d generators for polynomials of degree 1.

Theorem 11 ([Vio09b]). *Let $G : \{0, 1\}^s \rightarrow \{0, 1\}^n$ be a generator that fools polynomials of degree 1 with error ϵ . Then $G_d : (\{0, 1\}^s)^d \rightarrow \{0, 1\}^n$ defined as*

$$G_d(x^1, x^2, \dots, x^d) := G(x^1) + G(x^2) + \dots + G(x^d)$$

fools polynomials of degree d with error $\epsilon_d := 16 \cdot \epsilon^{1/2^{d-1}}$, where ‘+’ denotes bit-wise xor.

In particular, the combination of Theorems 10 and 11 yields generators $G : \{0, 1\}^s \rightarrow \{0, 1\}^n$ that fool polynomials of degree d with error $\epsilon_d = 1/n$ and seed length $s = O(d \cdot 2^d \cdot \log(n))$.

Proof sketch of Theorem 11. This proof uses the notation $e[z] := (-1)^z$ and some of the techniques presented at the end of §1.2.2. First, let us rewrite Inequality (1.7) in the Definition 9 of a generator as

$$|E_{S \in \{0,1\}^s} e[p(G_d(S))] - E_{U \in \{0,1\}^n} e[p(U)]| \leq \epsilon_d/2. \quad (1.8)$$

To prove Inequality (1.8), we proceed by induction on the degree d of the polynomial $p : \{0,1\}^n \rightarrow \{0,1\}$ to be fooled. The inductive step is structured as a case analysis based on the value $\tau := \text{Cor}_U(p, 0) = |E_{U \in \{0,1\}^n} e[p(U)]|$.

If τ is large then p correlates with a constant, which is a polynomial of degree lower than d , and by induction one can prove the intuitive fact that G_{d-1} fools p . This concludes the overview of the proof in this case.

If τ is small we reason as follows. Let us denote by W the output of G_{d-1} and by Y an independent output of G , so that the output of G_d is $W + Y$. We start by an application of the Cauchy-Schwarz inequality:

$$E_{W,Y} e[p(W+Y)]^2 \leq E_W [E_Y e[p(W+Y)]]^2 = E_{W,Y,Y'} e[p(W+Y) + p(W+Y')], \quad (1.9)$$

where Y' is independent from and identically distributed to Y . Now we observe that for every fixed Y and Y' , the polynomial $p(U+Y) + p(U+Y')$ has degree $d-1$ in U , though p has degree d . Since by induction W fools polynomials of degree $d-1$ with error ϵ_{d-1} , we can replace W with the uniform distribution $U \in \{0,1\}^n$:

$$E_{W,Y,Y'} e[p(W+Y) + p(W+Y')] \leq E_{U,Y,Y'} e[p(U+Y) + p(U+Y')] + \epsilon_{d-1}. \quad (1.10)$$

At this point, a standard argument shows that

$$E_{U,Y,Y'} e[p(U+Y) + p(U+Y')] \leq E_{U,U'} e[p(U) + p(U')] + \epsilon^2 = \tau^2 + \epsilon^2. \quad (1.11)$$

Therefore, chaining Equations (1.9), (1.10), and (1.11), we have that

$$|E_{W,Y} e[p(W+Y)] - E_U e[p(U)]| \leq |E_{W,Y} e[p(W+Y)]| + \tau \leq \sqrt{\tau^2 + \epsilon^2 + \epsilon_{d-1}} + \tau.$$

This proves Inequality (1.8) for a suitable choice of ϵ_d , concluding the proof in this remaining case. \square

Observe that Theorem 11 gives nothing for polynomials of degree $d = \log_2 n$. The reason is that its proof again relies on the “squaring trick” discussed in §1.2.2. But it is still not known whether the construction in Theorem 11 fools polynomials of degree $d \geq \log_2 n$.

Open question 4. *Does the sum of $d \gg \log n$ copies of a generator $G : \{0,1\}^s \rightarrow \{0,1\}^n$ that fools polynomials of degree 1 with error $1/n$ fools polynomials of degree d with error $1/3$?*

Finally, note that Observation 1 combined with the construction in Theorem 11 gives a new candidate function for an affirmative answer to Question 1: the function that on input $x \in \{0,1\}^n$ evaluates to 1 if and only if x is the bit-wise xor of $d \gg \log n$ outputs of generators that fool polynomials of degree 1.

1.4 Conclusion

We have discussed the challenge of proving correlation bounds for polynomials. We have seen that winning this challenge is necessary for proving lower bounds such as “NP does not have quasipolynomial-size circuits,” that a great deal is known for various settings of parameters, and that there are many interesting research directions. The clock is ticking...

Acknowledgments. We are grateful to Noam Nisan for his permission to include his proof of Theorem 4, Rocco Servedio for a discussion on boosting, and Avi Wigderson for suggesting a simpler proof of Theorem 7. We also thank Frederic Green, Shachar Lovett, Rocco Servedio, and Avi Wigderson for helpful comments on a draft of this report.

Chapter 2

The correlation of parity with small-depth circuits

2.1 Introduction

In this chapter we prove that the parity function has exponentially small correlation with small bounded-depth circuits. To formally state the result, let us make its main players more concrete. First, the parity function $\text{Parity} : \{0, 1\}^n \rightarrow \{0, 1\}$ is the degree-1 polynomial over $\{0, 1\}$ defined as

$$\text{Parity}_n(x_1, x_2, \dots, x_n) := \sum_i x_i \bmod 2.$$

Second, the circuits we consider consist of input, \wedge (and), \vee (or), and \neg (not) gates, where the input gates are labeled with variables (e.g., x_i), and \wedge and \vee gates have unbounded fan-in. The *size* w of the circuit is the number of edges, and its *depth* d is the longest path from an input gate to the output gate. With a moderate blow-up, it is possible to “push down” all the \neg gates so that the circuit takes as input both the variables and their negations, but consists only of alternating layers of \wedge and \vee gates, hence the name alternating circuit (AC^0). We do so in Chapter 3, but in this one we find it more convenient to allow for \neg gates everywhere in the circuit.

Finally, we use the following definition of correlation

$$\text{Cor}(f, C) := |E_x e[f(x) + C(x)]|,$$

where $e[z] := (-1)^z$, cf. Page 13.

We can now state the main theorem of this chapter.

Theorem 12 ([Hås87]). *Every circuit $C : \{0, 1\}^n \rightarrow \{0, 1\}$ of depth d and size $w := \exp(n^{\alpha/d})$ satisfies $\text{Cor}(\text{Parity}_n, C) \leq 1/w$, where α is an absolute constant.*

The theorem is a significant strengthening of preceding results establishing that small bounded-depth circuits cannot compute parity [Ajt83, FSS84, Yao85], see also [Cai89]. The

motivation for studying correlation bounds we mentioned in Chapter 1 for polynomials extends to circuits. First, correlation bounds imply lower bounds for richer classes of circuits. Second, correlation bounds yield pseudorandom generators: Nisan [Nis91] uses Theorem 12 to construct unconditional pseudorandom generators for small bounded-depth circuits, a result that has had a large number of applications and anticipated the celebrated Nisan-Wigderson paradigm to construct generators from lower bounds [NW94].

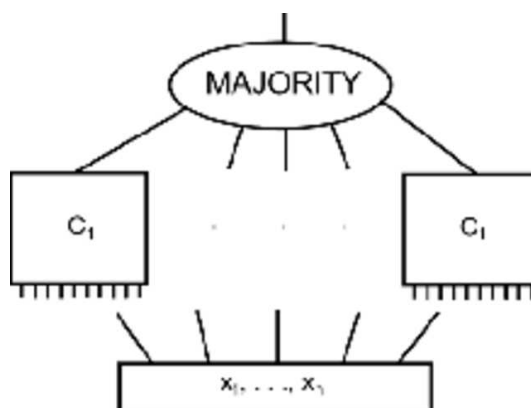
Theorem 12 was first established by Håstad [Hås87] using switching-lemma techniques. We present a simpler proof of Theorem 12 which has two stages. Stage 1 shows that the parity function cannot be computed by a special, richer class of circuits we call *output-majority*. These are small, bounded-depth circuits whose output gate computes the majority function (whereas the other gates are \wedge , \vee , and \neg as usual). This result is due to Aspnes, Beigel, Furst, and Rudich [ABFR94] and here we present its proof mostly unaltered. Stage 2 proves Theorem 12 from Stage 1. This two-stage approach was first proposed by Klivans [Kli01] who implemented Stage 2 using hardness amplification techniques. We present a simpler proof of Stage 2 that is due to Klivans and Vadhan. This proof appears here for the first time with their kind permission.

Since it is the newest part of the argument, we begin in the next §2.2 with Stage 2: we define output-majority circuits, state the result from [ABFR94] which is the outcome of Stage 1, and show how this implies Theorem 12. For completeness, in §2.3 we give the details of Stage 1.

2.2 Stage 2: From output-majority circuits to Theorem 12

We start by defining output-majority circuits.

Definition 13 (Output-majority circuits). *An output-majority circuit on n bits is a circuit of the following form:*



where x_1, \dots, x_n are the inputs, the output gate is a Majority gate, and each intermediate C_1, \dots, C_t is a circuit on n bits with only \wedge , \vee , and \neg gates.

Adding the majority gate gives extra power, as computing majority requires exponential size for bounded-depth circuits with \wedge , \vee , and \neg gates. This in fact follows from the results in this chapter, see also [Hås87].

The class of output-majority circuits is more robust than what might be apparent at first sight. Beigel [Bei94] shows that a circuit of size w and depth d with m majority gates can always be simulated by an output-majority circuit (i.e., a circuit with exactly 1 majority gate as the output gate) of depth $d' = d + O(1)$ and size $w' = \exp\left(m \cdot \log^{O(d)} w\right)$. For fixed d and $m = \text{poly log } w$, this is a quasi-polynomial blow-up in the size of the circuit which is negligible compared to the exponential lower bounds we prove in this chapter and state next:

Theorem 14 ([ABFR94]). *Let C be an output-majority circuit of depth d and size w which computes Parity on n bits. Then, $w \geq \exp\left(n^{\alpha/d}\right)$ for a universal constant α .*

In the rest of this section we prove the main Theorem 12 of this chapter assuming the above Theorem 14.

Intuition for the proof of Theorem 12 assuming Theorem 14. Imagine that you are given a coin that is biased, but you do not know on which side. That is, when tossing the coin, either the coin has probability 51% of landing heads up or probability 51% of landing tails up. Your job is to guess the side towards which the coin is biased. You would like your guess to be correct with high probability $\approx 100\%$. A solution to this problem, analyzable via standard probability theory, is to toss the coin a large number of times and decide according to the majority of the outcomes.

This is essentially what we will do. Assuming for the sake of contradiction that we have a circuit C that correlates well with parity, we proceed as follows. For a given, fixed parity instance x we construct a distribution on circuits $C_a(x)$ whose output distribution is biased towards $\text{Parity}(x)$; this is the aforementioned “biased coin.” Taking the majority of a number of independent copies of $C_a(x)$ gives an output-majority circuit that computes $\text{Parity}(x)$ correctly with overwhelming probability, using which we contradict Theorem 14.

To construct $C_a(x)$ from C we take advantage of the efficient *random self-reducibility of parity*, the ability to efficiently reduce the computation of $\text{Parity}(x)$ for any given input x to the computation of $\text{Parity}(a)$ for a randomly selected input a .

2.2.1 Proof of Theorem 12 assuming Theorem 14

Let C be a circuit of depth d and size w such that $\left|E_{x \in \{0,1\}^n} [C(x) + \text{Parity}(x)]\right| \geq 1/w$. Without loss of generality we drop the absolute value signs to have

$$E_{x \in \{0,1\}^n} [C(x) + \text{Parity}(x)] \geq 1/w$$

(if $E_x [C(x) + \text{Parity}(x)] \leq -1/w$ we can complement C at a negligible cost).

We now make use of the *random self-reducibility* of Parity. For $a \in \{0, 1\}^n$, consider the circuit C_a defined by

$$C_a(x) := C(a + x) + \text{Parity}(a).$$

First of all, notice that for any fixed a , C_a has depth $d + O(1)$ and size $w + O(1) + O(n) = O(w)$ (we use here that $w \geq n$, for else some input bit does not have wires coming out of it, and the theorem is immediate). This is because XOR'ing the output with the bit $\text{Parity}(a)$ can be done with a constant number of extra gates and wires, while XOR'ing the input by a takes constant depth and $O(n)$ gates. Now observe that for any fixed $x \in \{0, 1\}^n$,

$$\begin{aligned} E_a e [C_a(x) + \text{Parity}(x)] &= E_a e [C(a + x) + \text{Parity}(a) + \text{Parity}(x)] \\ &= E_a e [C(a + x) + \text{Parity}(a + x)] \\ &= E_a e [C(a) + \text{Parity}(a)] \geq 1/w. \end{aligned}$$

It is convenient to rewrite the above conclusion as: for any fixed $x \in \{0, 1\}^n$,

$$\Pr_a [C_a(x) = \text{Parity}(x)] \geq \frac{1}{2} + \frac{1}{2w}.$$

Now pick t independent a_1, a_2, \dots, a_t for a parameter t to be determined below, and define the output-majority circuit

$$\bar{C}_{a_1, \dots, a_t}(x) := \text{Majority}(C_{a_1}(x), \dots, C_{a_t}(x)).$$

Note that $\bar{C}_{a_1, \dots, a_t}$ has depth $d + O(1)$ and size $O(t \cdot w)$, for every choice of a_1, \dots, a_t . The next standard, concentration-of-measure lemma gives a bound on t that is $\text{poly}(w)$.

Lemma 15. *For $t := 2w^2n$, for any $x \in \{0, 1\}^n$, $\Pr_{a_1, \dots, a_t} [\bar{C}_{a_1, \dots, a_t}(x) \neq \text{Parity}(x)] < 2^{-n}$.*

Deferring the proof of the lemma for a moment, let us finish the proof of the theorem. A union bound gives us that

$$\Pr_{a_1, \dots, a_t} [\exists x : \bar{C}_{a_1, \dots, a_t}(x) \neq \text{Parity}(x)] < 2^n \cdot 2^{-n} = 1.$$

Another way of stating this is that $\Pr_{a_1, \dots, a_t} [\forall x : \bar{C}_{a_1, \dots, a_t}(x) = \text{Parity}(x)] > 0$. Thus, there is some fixed choice of a_1, \dots, a_t which results in an output-majority circuit of depth $d + O(1)$ and size $O(t \cdot w) = \text{poly}(w) \cdot n = \text{poly}(w)$ that computes Parity on every n -bit input (again we use $w > n$).

It only remains to prove Lemma 15.

Proof of Lemma 15. Fix an input $x \in \{0, 1\}^n$. Define t indicator random variables Y_1, Y_2, \dots, Y_t , where $Y_i := 1$ if and only if $C_{a_i}(x) = \text{Parity}(x)$. Let $\epsilon \in [1/(2w), 1/2]$ be the ‘‘advantage’’

that each C_{a_i} has, so that $\Pr_{a_i}[Y_i = 1] = 1/2 + \epsilon$. We have:

$$\begin{aligned}
& \Pr_{a_1, \dots, a_t} [\bar{C}_{a_1, \dots, a_t}(x) \neq \text{Parity}(x)] = \Pr [\text{Majority}(Y_1, \dots, Y_t) \neq 1] \\
&= \sum_{I \subseteq [t], |I| \geq t/2} \Pr [Y_i = 0 \ \forall i \in I] \cdot \Pr [Y_i = 1 \ \forall i \notin I] \\
&= \sum_{I \subseteq [t], |I| \geq t/2} (1/2 - \epsilon)^{|I|} \cdot (1/2 + \epsilon)^{t-|I|} \\
&\leq \sum_{I \subseteq [t], |I| \geq t/2} (1/2 - \epsilon)^{t/2} \cdot (1/2 + \epsilon)^{t/2} \quad (\text{multiply by } (\frac{1}{2} - \epsilon)^{t/2-|I|} (\frac{1}{2} + \epsilon)^{|I|-t/2} \geq 1) \\
&\leq 2^t \cdot (1/4 - \epsilon^2)^{t/2} \leq (1 - 1/w^2)^{t/2} \leq e^{-t/(2 \cdot w^2)}.
\end{aligned}$$

The latter quantity is less than 2^{-n} for $t = 2w^2 \cdot n$, concluding the proof of the lemma. \square

2.3 Stage 1: Output-majority circuits cannot compute parity

In this section we prove Theorem 14, i.e., that small output-majority circuits cannot compute parity. The proof involves a suitable approximation of output majority circuits by low-degree polynomials. Rather than working with polynomials over $\{0, 1\}$ as in Chapter 1, we now work with real-valued polynomials, whose output are arbitrary real numbers. An example of such a polynomial is

$$17x_1 - 3x_2x_3 + 4,$$

which on input $x_1 = x_2 = x_3 = 1$ has value 18.

Whereas polynomials over $\{0, 1\}$ also approximate bounded-depth circuits [Raz87], the approximation of output-majority circuits requires polynomials over the reals.

The proof of Theorem 14 can be broken up in three steps.

1. First, we show that any output-majority circuit C of depth d and size w can be *sign-approximated* by a real-valued polynomial $p : \{0, 1\}^n \rightarrow \mathcal{R}$ of degree $\log^{O(d)} w$, in the sense that for almost all inputs $x \in \{0, 1\}^n$, $\text{sign}(p(x)) = e(C(x))$. Here $\text{sign}(x) = 1/-1$ if x is positive/negative, and is undefined if $x = 0$. Also, $e(z) = e[z] = (-1)^z$.
2. Then we show that from p one can construct another real-valued polynomial p' that *weakly computes* C , in the sense that whenever $p'(x) \neq 0$ we have $\text{sign}(p'(x)) = e(C(x))$ but p' is not always 0. Here the degree will blow up to close to, but less than n . Specifically, the degree of p' will be $n - \Omega(\sqrt{n}) + \log^{O(d)} w$.
3. Finally, we show that to weakly compute the parity function on n bits, degree n is required (which is tight, since every function on n bits is computable by a polynomial of degree n).

2.3.1 Step 1: Sign-approximating output-majority circuits

In this section we prove the following result that output-majority circuits can be sign-approximated by low-degree polynomials

Theorem 16 (Sign-approximating output-majority circuits). *Let C be an output-majority circuit with n inputs, depth d , and size w . Then, for every $\epsilon > 0$ there exists a polynomial $p : \{0, 1\}^n \rightarrow \mathcal{R}$ of degree $\log^{O(d)}(w/\epsilon)$ such that*

$$\Pr_{x \in \{0,1\}^n} [\text{sign}(p(x)) = e(C(x))] \geq 1 - \epsilon.$$

A typical setting of parameters is $w = \text{poly}(n)$, $\epsilon = 0.01$, $d = O(1)$, in which case we get $\text{degree}(p) = \text{polylog}(n)$.

The proof of Theorem 16 goes by first exhibiting a stronger type of polynomial approximation for the sub-circuits of C without the majority gate, and then summing the polynomials to obtain a sign approximation of C . The stronger type of approximation simply asks for a polynomial that computes the output of the circuit for most inputs (as opposed to a polynomial that evaluates to a number whose sign matches the output of the circuit for most inputs).

Theorem 17 (Approximating circuits). *Let $C : \{0, 1\}^n \rightarrow \{0, 1\}$ be a circuit of size $w \geq n$ and depth d . Then, for every $\epsilon > 0$ there exists a polynomial $p : \{0, 1\}^n \rightarrow \mathcal{R}$ of degree $\log^{O(d)}(w/\epsilon)$ such that*

$$\Pr_{x \in \{0,1\}^n} [p(x) = C(x)] \geq 1 - \epsilon.$$

Proof of Theorem 16 assuming Theorem 17. Let C be an output-majority circuit of size w and depth d . Denote its subcircuits that feed into the output majority gate by C_1, \dots, C_t , where $t \leq w$. Apply Theorem 17 with $\epsilon' := \epsilon/t$ to get polynomials p_1, \dots, p_t such that for every $i \leq t$ we have $\Pr_x [p_i(x) = C_i(x)] \geq 1 - \epsilon'$. The degree of each p_i is $\leq \log^{O(d)}(w/\epsilon')$, since each circuit C_i has size $\leq w$ and depth $\leq d$. Then, define

$$p(x) := \sum_{i \leq t} p_i(x) - \frac{t}{2}.$$

Note that the degree of p is equal to the maximum degree of any p_i , which is $\log^{O(d)}(w/\epsilon') = \log^{O(d)}(w/\epsilon)$ (because $t \leq w$). Also note that, whenever we have $p_i(x) = C_i(x)$ for all i , then $\text{sign}(p(x)) = \text{Majority}(C_1(x), \dots, C_t(x)) = C(x)$. By a union bound, this happens with probability at least $1 - t \cdot \epsilon' = 1 - \epsilon$. \square

We now turn to the proof of Theorem 17. The plan is to replace each gate by a (randomized) polynomial and obtain the final polynomial by composing all the polynomials according to the circuit. The degree of the polynomial corresponding to each gate will be $\log^{O(1)}(w/\epsilon)$. Since composing polynomials multiplies degrees, the final degree will be (the degree at each gate) ^{d} .

We now show how to build a polynomial for every gate in the circuit. It is convenient to work over the basis \vee, \neg , which can be accomplished using De Morgan's law $\bigwedge_i x_i = \neg \bigvee_i \neg x_i$. Using this law changes our circuit slightly. This change is irrelevant for the type of quantitative bounds in this chapter. Moreover, it will be evident from the proofs that the number of \neg gates does not count towards the bounds, so this change in fact costs us nothing.

We start with the \neg gate.

Lemma 18. *There is a polynomial $p : \{0, 1\} \rightarrow \mathcal{R}$ of degree 1 such that $p(b) = \neg b$ for every $b \in \{0, 1\}$.*

Proof. $p(b) := 1 - b$. □

The next more challenging lemma deals with \vee gates. Whereas for \neg gates we exhibited a low-degree polynomial that computes the gate correctly on every input, now for \vee gates we will only show that for any distribution D there is a low-degree polynomial computing the gate correctly with high probability over the choice of an input from D . We state the lemma now and then we make some comments.

Lemma 19. *For all distributions D on $\{0, 1\}^m$, for all $\epsilon > 0$, there exists a polynomial $p : \{0, 1\}^m \rightarrow \mathcal{R}$ of degree $O(\log m \cdot \log(1/\epsilon))$ such that*

$$\Pr_{x \sim D} [\vee(x) = p(x)] \geq 1 - \epsilon.$$

We stress that the lemma holds for any distribution D . This is crucial because we need to apply the lemma to internal gates of the circuit that, over the choice of a uniform n -bit input to the circuit, may be fed inputs of length $m = \text{poly}(n)$ drawn from a distribution we do not have a grasp of. Before proving the lemma, let us see how to prove Theorem 17 by applying the above lemmas to every gate in the circuit.

Proof of Theorem 17. Given ϵ , invoke Lemma 19 for every \vee gate in the circuit with error ϵ/w and with respect to the distribution D_g that the choice of a uniform input $x \in \{0, 1\}^n$ for the circuit induces on the inputs of that gate. Also invoke Lemma 18 for every \neg gate in the circuit. Thus, for every gate g we have a polynomial p_g of degree $O(\log(w) \cdot \log w/\epsilon) = \log^{O(1)}(w/\epsilon)$ that computes g correctly with probability $1 - \epsilon/w$ with respect to the choice of an input from D_g .

Now we compose the polynomials for every gate into a single polynomial p . This composition is done according to the circuit: treat input variables to the circuit as degree-1 polynomials x_1, \dots, x_n . Then, level by level towards the output gate, for any gate g with input gates g_1, \dots, g_m substitute to the variables of p_g the polynomials p_{g_1}, \dots, p_{g_m} . When composing polynomials degrees multiply; hence the degree of p is $\left(\log^{O(1)}(w/\epsilon)\right)^{O(d)} = \log^{O(d)}(w/\epsilon)$. Finally, by a union bound we have

$$\Pr_{x \in \{0, 1\}^n} [p(x) \neq C(x)] \leq \sum_g \Pr_{y \sim D_g} [p_g(y) \neq g(y)] \leq w \cdot \epsilon/w = \epsilon.$$

□

To finish this section it only remains to prove Lemma 19. For certain distributions D the lemma is easily proved. For example if D is the uniform distribution then the constant polynomial $p = 1$ proves the lemma. In general however the arbitrariness of D in the statement of the lemma makes it somewhat challenging to establish. To handle arbitrary distributions D , the proof employs a relatively standard trick. We exhibit a distribution P on polynomials that computes the \vee gate correctly with high probability on any fixed input. This in particular means that P also computes the \vee gate correctly with high probability when the input to the gate is drawn from D . At the end, we fix the choice of the polynomial.

2.3.2 Proof of Lemma 19

We assume without loss of generality that m , the length of the input to the \vee gate, is a power of 2. For any $i = 0, \dots, \log m$, consider the random set $S_i \subseteq \{1, \dots, m\}$ obtained by including each element of $\{1, \dots, m\}$ independently with probability 2^{-i} ; in particular, $S_0 := \{1, \dots, m\}$.

Consider the associated random degree-1 polynomials

$$P_i(x) := \sum_{i \in S_i} x_i$$

for $i = 0, \dots, \log m$.

First, we claim that every polynomial always computes \vee correctly when the input is 0^m . The proof of this claim is obvious.

Claim 20. *If $x = 0^m$ then $P_i(x) = 0$ always and for all i .*

Second, we claim that on any input that is not 0^m , with high probability there is some polynomial that computes \vee correctly.

Claim 21. *For all $x \in \{0, 1\}^m$, $x \neq 0^m$, with probability at least $1/6$ over $P_0, \dots, P_{\log m}$ there is i such that $P_i(x) = 1$.*

Proof. Let $w := \sum x_i$, $1 \leq w \leq m$, be the weight of x . Let i , $0 \leq i \leq \log m$, be such that

$$2^{i-1} < w \leq 2^i. \tag{2.1}$$

We have

$$\begin{aligned} \Pr[P_i(x) = 1] &= \Pr \left[\sum_{i \in S_i} x_i = 1 \right] \\ &= w \cdot 2^{-i} (1 - 2^{-i})^{w-1} \\ &\geq \frac{1}{2} \cdot (1 - 2^{-i})^{2^i-1} \quad (\text{Using 2.1}) \\ &\geq \frac{1}{2e} \geq 1/6. \end{aligned}$$

□

We now combine the above polynomials into a single one P' as follows:

$$P'(x) := 1 - \prod_{i=0}^{\log m} (1 - P_i(x)).$$

If $x = 0^m$, then $P'(x) = 0$. If $x \neq 0^m$ then $\Pr[P'(x) = 1] \geq 1/6$ by Claim 21. Also, the degree of P' is $O(\log m)$.

We can reduce the error in P' by using the same trick. Let

$$P_\epsilon(x) := 1 - \prod_{i=0}^{4 \log 1/\epsilon} (1 - P'_i(x))$$

where $P'_i(x)$ are independent copies of P' above. If $x = 0$ then $P_\epsilon(x) = 0$. If $x \neq 0$,

$$\begin{aligned} \Pr[P_\epsilon(x) = 1] &\geq \Pr[\exists i : P'_i(x) = 1] = 1 - \Pr[\forall i, P'_i(x) \neq 1] \\ &= 1 - \Pr[P'(x) \neq 1]^{4 \log 1/\epsilon} \geq 1 - (5/6)^{4 \log 1/\epsilon} \geq 1 - \epsilon. \end{aligned}$$

Also, the degree of P_ϵ is $O(\log m \cdot \log 1/\epsilon)$.

To conclude, note that we have constructed P_ϵ such that for every x ,

$$\Pr_{P_\epsilon}[P_\epsilon(x) \neq \vee(x)] \leq \epsilon.$$

In particular,

$$\Pr_{x \sim D, P_\epsilon}[P_\epsilon(x) \neq \vee(x)] \leq \epsilon.$$

This implies that there exists a choice \hat{p} for P_ϵ such that

$$\Pr_{x \sim D}[\hat{p}(x) \neq \vee(x)] \leq \epsilon.$$

Again, the degree of \hat{p} is $O(\log m \cdot \log 1/\epsilon)$; this concludes the proof.

2.3.3 Step 2: From sign-approximating to weakly computing

In this section we show how to turn any polynomial p that sign-approximates a function f on all but an ϵ fraction of inputs, i.e. $\Pr_x[\text{sign}(p(x)) \neq e(f(x))] \leq \epsilon$, into another one p' that weakly computes f , i.e. p' is not the zero polynomial and for every x such that $p'(x) \neq 0$ we have $\text{sign}(p'(x)) = e(f(x))$.

Lemma 22. *Let $p : \{0, 1\}^n \rightarrow \mathcal{R}$ be a polynomial of degree d and $f : \{0, 1\}^n \rightarrow \{0, 1\}$ a function such that $\Pr_x[\text{sign}(p(x)) \neq e(f(x))] \leq \epsilon < 1/2$. Then, there exists a non-zero polynomial $p' : \{0, 1\}^n \rightarrow \mathcal{R}$ such that for every x where $p'(x) \neq 0$ we have $\text{sign}(p'(x)) = e(f(x))$, and the degree of p' is $\leq n - \epsilon' \sqrt{n} + d$, for a constant $\epsilon' > 0$ which depends only on ϵ .*

The idea is to construct a non-zero polynomial q that is 0 on the ϵ fraction of points on which $\text{sign}(p(x)) \neq e(f(x))$, and then define

$$p'(x) := p(x) \cdot q^2(x).$$

Since q is 0 on any x on which $\text{sign}(p(x)) \neq e(f(x))$, and we square it, p' weakly computes f : on any input where $p'(x) \neq 0$ we have $\text{sign}(p'(x)) = \text{sign}(p(x)) = e(f(x))$.

To zero-out an ϵ fraction of the points, the degree of q will need to be $n/2 - \epsilon'\sqrt{n}$. Hence the degree of q^2 will be $n - 2\epsilon'\sqrt{n}$ and that of p' will be $n - 2\epsilon'\sqrt{n}$ plus the degree of p . In a typical setting of parameters, the degree of p is polylogarithmic in n which makes the degree of p' strictly less than n , which is sufficient for our purposes.

Proof of Lemma 22. Let $S \subseteq \{0, 1\}^n$ be the set of inputs x such that $\text{sign}(p(x)) \neq e(f(x))$. By assumption, $|S| \leq \epsilon 2^n$. We define a non-zero polynomial q such that $q(x) = 0$ for every $x \in S$. Let M be the set of products of at most $n/2 - \epsilon'\sqrt{n}$ variables, for a parameter ϵ' to be chosen later. In other words, M is the set of monic, multilinear monomials of degree $\leq n/2 - \epsilon'\sqrt{n}$:

$$M := \left\{ \prod_{i \in I} x_i : I \subseteq [n], |I| \leq n/2 - \epsilon'\sqrt{n} \right\}.$$

Then, let $q(x) := \sum_{m \in M} a_m \cdot m(x)$ be the weighted sum of these monomials, for a set of weights, or coefficients, $\{a_m\}_{m \in M}$. We want to choose the coefficients in order to make q a non-zero polynomial that vanishes on S . This is equivalent to finding a non-trivial solution to a certain system of equations. Denote $S = \{s_1, \dots, s_{|S|}\}$ and $M = \{m_1, \dots, m_{|M|}\}$. Then, the system of equations we would like to solve is

$$\begin{pmatrix} m_1(s_1) & m_2(s_1) & \cdots & m_{|M|}(s_1) \\ m_1(s_2) & m_2(s_2) & \cdots & m_{|M|}(s_2) \\ \vdots & \vdots & \ddots & \vdots \\ m_1(s_{|S|}) & m_2(s_{|S|}) & \cdots & m_{|M|}(s_{|S|}) \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_{|M|} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

From linear algebra, we know that this system has a non-trivial solution if there are more variables (the $|M|$ coefficients a_i) than equations (the $|S|$ rows in the matrix). Therefore, we need to show $|M| > |S|$. Using bounds similar to those used in the proof of Theorem 2 in §1.2.1, we have:

$$\begin{aligned} |M| &= \sum_{i=0}^{\lfloor n/2 - \epsilon'\sqrt{n} \rfloor} \binom{n}{i} = \sum_{i=0}^{\lceil n/2 \rceil} \binom{n}{i} - \sum_{i=\lfloor n/2 - \epsilon'\sqrt{n} \rfloor + 1}^{\lceil n/2 \rceil} \binom{n}{i} \geq 2^{n-1} - O(\epsilon'\sqrt{n}) \binom{n}{\lfloor n/2 \rfloor} \\ &\geq 2^{n-1} - O(\epsilon' \cdot 2^n). \end{aligned}$$

By choosing ϵ' sufficiently small compared to $\epsilon < 1/2$, we have

$$|M| > 2^n (1/2 - O(\epsilon')) > 2^n \cdot \epsilon = |S|.$$

This shows the existence of the desired polynomial q .

Finally, let

$$p'(x) := p(x) \cdot q(x)^2.$$

Since q vanishes on S and q^2 is always positive, $\text{sign}(p'(x)) = e(f(x))$ whenever $p'(x) \neq 0$. Furthermore, p' is not the zero polynomial, since neither p nor q are, and the degree of p' is the degree of p plus twice the degree of q : $d + 2(n/2 - \epsilon'\sqrt{n}) = d + n - 2\epsilon'\sqrt{n}$. \square

2.3.4 Step 3

In this section we show that degree at least n is required to weakly compute the parity function.

Lemma 23. *Let $p : \{0, 1\}^n \rightarrow \mathcal{R}$ be a non-zero polynomial such that whenever $p(x) \neq 0$ we have $\text{sign}(p(x)) = e(\sum_i x_i)$. Then p has degree at least n .*

Proof. Suppose for the sake of contradiction that p has degree $n - 1$. Then, for some coefficients $\{c_I\}_{I \subseteq [n]}$, we can write

$$p(x) = \sum_{I \subseteq [n], |I| \leq n-1} c_I \prod_{i \in I} x_i.$$

Let us consider

$$E_{x \in \{0,1\}^n} \left[p(x) \cdot e \left(\sum_i x_i \right) \right].$$

On the one hand, $E_{x \in \{0,1\}^n} [p(x) \cdot e(\sum_i x_i)] > 0$, since the polynomial is not identically 0 and when it is not zero its sign is $e(\sum_i x_i)$.

On the other hand, by linearity of expectation,

$$\begin{aligned} & E_{x \in \{0,1\}^n} \left[p(x) \cdot e \left(\sum_i x_i \right) \right] \\ &= \sum_{I \subseteq [n], |I| \leq n-1} c_I \cdot E_{x \in \{0,1\}^n} \left[\prod_{i \in I} x_i \cdot e \left(\sum_i x_i \right) \right] \\ &= \sum_{I \subseteq [n], |I| \leq n-1} c_I \cdot E_{x \in \{0,1\}^n} \left[\prod_{i \in I} x_i \cdot e \left(\sum_{i \in I} x_i \right) \right] \cdot E_{x \in \{0,1\}^n} \left[e \left(\sum_{i \notin I} x_i \right) \right] \\ &= 0, \end{aligned}$$

where the last equality holds because $|I| < n$, and so the sum $\sum_{i \notin I} x_i$ contains at least one variable, which over the choice of a uniform x will be equally likely to be 0 or 1, making $e(\sum_{i \notin I} x_i)$ equally likely to be $+1$ and -1 , for an expected value of 0. \square

2.3.5 Putting the three steps together

We now put the preceding three steps together to prove that small output-majority circuits cannot compute parity, i.e., Theorem 14.

Theorem 14 ([ABFR94]). (Restated.) *Let C be an output-majority circuit of depth d and size w which computes Parity on n bits. Then, $w \geq \exp(n^{\alpha/d})$ for a universal constant α .*

Proof of Theorem 14. Let C be an output-majority circuit computing Parity : $\{0, 1\}^n \rightarrow \{0, 1\}$, and let C have size w and depth d . Apply Theorem 16 with $\epsilon := 1/3$ to obtain a polynomial $p : \{0, 1\}^n \rightarrow \mathcal{R}$ of degree $\log^{O(d)} w$ such that

$$\Pr_{x \in \{0,1\}^n} [\text{sign}(p(x)) = e(C(x)) = e(\text{Parity}(x))] \geq 2/3.$$

Apply Lemma 22 to obtain a non-zero polynomial p' of degree $\leq n - \Omega(\sqrt{n}) + \log^{O(d)} w$ such that, for any x for which $p'(x) \neq 0$, we have $\text{sign}(p'(x)) = e(\text{Parity}(x))$.

By Lemma 23, the degree of p' must be at least n . Therefore:

$$\begin{aligned} n - \Omega(\sqrt{n}) + \log^{O(d)} w &\geq n \\ \Leftrightarrow \log^{O(d)} w &\geq \Omega(\sqrt{n}), \end{aligned}$$

concluding our proof. □

Chapter 3

Logarithmic depth vs. depth 3

As is well known, researchers in complexity theory cannot yet prove strong lower bounds on the size of general circuits computing explicit functions, the best ones being of the form $c \cdot n$ for n -bit functions, where c is a constant (see, e.g., [IM02] and the references therein). Of particular importance for this chapter is the fact that this inability to prove a superlinear lower bound holds even if the circuit is restricted to have depth $O(\log n)$ (and fan-in 2).

On the other hand, researchers have been able to prove stronger, superpolynomial lower bounds on the size of circuits of bounded depth (and unbounded fan-in), some of which we saw in Chapter 2. Still, progress on these lower bounds appears slow: the size bounds obtained in the 80's essentially remain state-of-the art (see, e.g., [Hås87] and the references therein).

As anticipated in §1.2, there are links between these two fronts that indicate they are hitting the same wall. In this chapter we prove a fine such link by Valiant [Val77, Val83]. For a clean formulation, it is convenient to work with the following model of bounded-depth circuits: we allow for input gates both the variables x_1, x_2, \dots and their negations $\neg x_1, \neg x_2, \dots$, but we require that every other gate is either \wedge or \vee (this is slightly different from the model considered in Chapter 2, cf. the discussion there). With this convention, Valiant's result is that any circuit of linear size and logarithmic depth (with fan-in 2) can be simulated by a sub-exponential size circuit of depth 3 (with unbounded fan-in). Therefore, a $2^{\Omega(n)}$ bound for these depth-3 circuits implies a superlinear lower bound for log-depth circuits (with fan-in 2).

3.1 Exponential lower bounds for depth 2

To get comfortable with our circuit model and provide some motivation, we start in this section with a lower bound for depth-2 circuits of unbounded fan-in. Recall that our circuits take both variables and their negations as inputs, so a depth-2 circuit is either a DNF such as

$$(x_1 \wedge \neg x_2 \wedge x_5) \vee (\neg x_5 \wedge x_7),$$

or a CNF (swap \wedge with \vee). The *size* of a circuit is again its number of edges.

Claim 24. Any DNF for Parity : $\{0, 1\}^n \rightarrow \{0, 1\}$ has size $\geq 2^{n-1}$.

For comparison, it is easy to see that any function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ can be computed by a DNF of size $n \cdot 2^n$.

Proof. First we show that every \wedge gate (a.k.a. term) is connected to at least one of x_i and $\neg x_i$ for every i . To see this, suppose for contradiction that some \wedge gate does not depend on x_i for some i . Fix an input that makes this \wedge gate output 1 (there is such an input without loss of generality). By definition of \vee , this input also makes the whole circuit output 1. If we flip the i -th bit of this input, the value of the circuit stays the same, but the value of parity flips. This contradicts the fact that the DNF computes Parity.

Hence, every \wedge gate is connected to at least one of x_i and $\neg x_i$ for every i . This implies there is only one input that makes the gate output 1 (namely, the input on which every one of the in-wires carries 1.) On the other hand, there exist 2^{n-1} inputs that give a parity of 1. Therefore, we need 2^{n-1} \wedge gates. \square

The above claim shows a function (parity) that requires $\vee \wedge$ circuits of size $\geq 2^{n-1}$. Challenge: Prove a bound of the form $2^{0.001n}$ for $\vee \wedge \vee$ circuits. Just one more layer! The results in the next section show this would also prove a superlinear lower bound for log-depth circuits of fan-in 2.

3.2 From logarithmic depth to depth 3

We now state the main result of this chapter, that any fan-in 2 circuit of linear size and logarithmic depth can be simulated by unbounded fan-in circuits of sub-exponential size and depth 3.

Theorem 25 ([Val77, Val83]). *Let $C : \{0, 1\}^n \rightarrow \{0, 1\}$ be a circuit of size $c \cdot n$, depth $c \cdot \log n$ and fan-in 2. The function computed by C can also be computed by an unbounded fan-in circuit of size $2^{c' \cdot n / \log \log n}$ and depth 3 with inputs $x_1, x_2, \dots, x_n, \bar{x}_1, \bar{x}_2, \dots, \bar{x}_n$, where c' depends only on c .*

A statement of this result for monotone circuits appears at the end of [Val83]. The proof of this result uses graph-theoretic techniques developed in [Val77, §5]. We are unaware of a source where the proof of the result appears in full, though it is relatively straightforward to obtain it from [Val77, Val83].

The idea of the simulation is to identify $\epsilon \cdot n = o(n)$ wires to remove from C so that the resulting circuit becomes very disconnected in the sense that each of its connected components has depth $\leq \epsilon \cdot \log n$. Since the circuit has fan-in 2, the output of each component can depend on at most n^ϵ input bits, and so, given the assignment to the removed edges, the output can be computed in brute-force by a depth-2 circuit of sub-exponential size. Trying all $2^{\epsilon \cdot n} = 2^{o(n)}$ assignments to the removed edges and collapsing some gates completes the simulation. We now proceed with a formal proof and we refer to Figure 1 for an example.

A circuit can be viewed as an acyclic directed graph with nodes representing gates and directed edges representing the flow of computed values from the output of one gate to the input of the next. The graph corresponding to C in Theorem 25 is connected, but we also work with disconnected graphs.

For the depth reduction in the proof, it is convenient to think of depth as a function from nodes to integers. The next definition and simple claim formalize this.

Definition 26 (Depth). *Let $G = (V, E)$ be a directed acyclic graph. The depth of a node in G is the number of nodes in a longest directed path terminating at that node. The depth of G is the depth of a deepest node in G .*

A depth function D for G is a map $D : V \rightarrow \{1, 2, \dots, 2^k\}$ such that if $(a, b) \in E$ then $D(a) < D(b)$.

Claim 27. *A directed acyclic graph $G = (V, E)$ has depth at most 2^k if and only if there is a depth function $D : V \rightarrow \{1, 2, \dots, 2^k\}$ for G .*

Proof. \Rightarrow : if the depth is at most 2^k then setting D to be the function that maps each node to its depth is a depth function.

\Leftarrow : suppose G has a node of depth $> 2^k$. Then there is a directed path with $> 2^k$ nodes in G . No depth function with range $\{1, 2, \dots, 2^k\}$ can assign values to all nodes on that path. \square

The following is the key lemma which allows us to reduce the depth of a graph by removing few edges.

Lemma 28. *Let $G = (V, E)$ be a directed acyclic graph with w edges and depth 2^k . It is possible to remove $\leq w/k$ edges so that the depth of the resulting graph is $\leq 2^{k-1}$.*

Proof. Let $D : V \rightarrow \{1, 2, \dots, 2^k\}$ be a depth function for G . Consider the set of edges E_i for $1 \leq i \leq k$:

$$E_i := \{(a, b) \in E \mid \text{the most significant bit position where } D(a) \text{ and } D(b) \text{ differ is the } i\text{-th}\}.$$

Note that E_1, E_2, \dots, E_k is a partition of E . And since $|E| = w$, there exists an index $i, 1 \leq i \leq k$, such that $|E_i| \leq w/k$. Fix this i and remove E_i . We need to show that the depth of the resulting graph is at most 2^{k-1} . To do so we exhibit a depth function $D' : V \rightarrow \{0, 1\}^{k-1}$. Specifically, let D' be D without the i -th output bit. We claim that D' is a valid depth function for the graph $G' := (V, E \setminus E_i)$. For this, we need to show that if $(a, b) \in E \setminus E_i$ then $D'(a) < D'(b)$. Indeed, let $(a, b) \in E \setminus E_i$. Since $(a, b) \in E$, we have $D(a) < D(b)$. Now, consider the most significant bit position j where $D(a)$ and $D(b)$ differ. There are three cases to consider:

- j is more significant than i : In this case, since the j -th bit is retained, the relationship is also maintained, i.e., $D'(a) < D'(b)$;

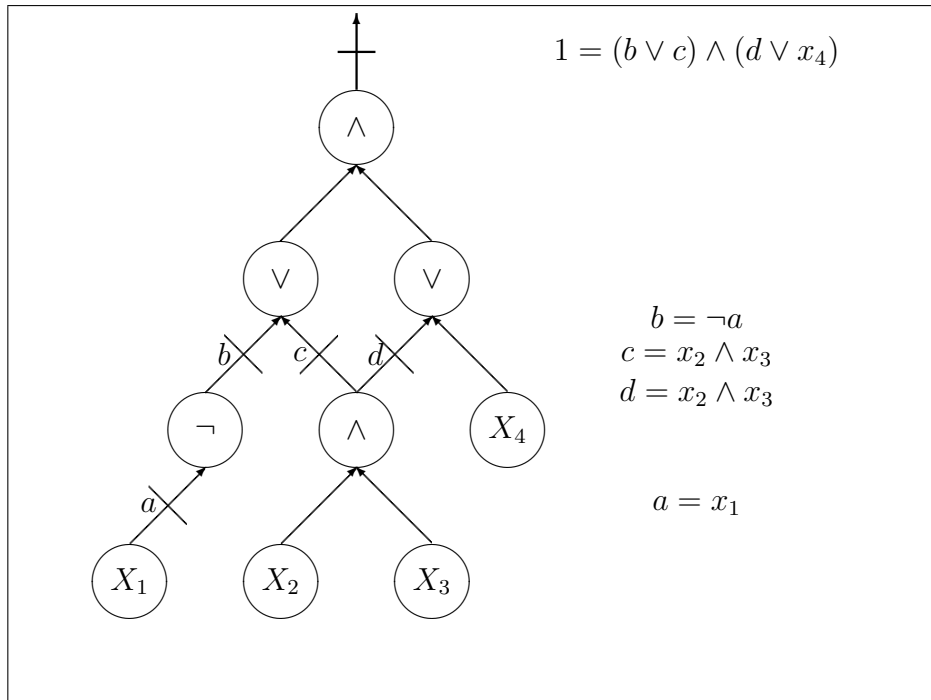


Figure 3.1: The removal of edges $a, b, c,$ and d reduces the depth. The circuit evaluates to 1 if and only if there are $a, b, c, d \in \{0, 1\}$ satisfying the corresponding equations.

- $j = i$: This case cannot occur because it would mean that the edge $(a, b) \in E_i$;
- j is less significant than i : In this case, the i -th bit of $D(a)$ and $D(b)$ is the same and so removing it maintains the relationship, i.e., $D'(a) < D'(b)$.

□

Now we prove the main theorem.

Proof of Theorem 25. For simplicity, we assume that both c and $\log n$ are powers of two. Let $2^\ell := c \cdot \log n$.

Applying the above lemma we can reduce the depth by a factor $1/2$, i.e. from 2^ℓ to $2^{\ell-1}$, by removing $\leq c \cdot n/\ell$ edges. Applying the lemma again we reduce the depth to $2^{\ell-2}$ by removing $\leq c \cdot n/(\ell - 1)$ edges. If we repeatedly apply the lemma $\log(2c)$ times the depth reduces to

$$\frac{c \log n}{2^{\log(2c)}} = \frac{\log n}{2},$$

and the total number of edges removed is at most

$$c \cdot n \left(\frac{1}{\ell} + \frac{1}{\ell-1} + \dots + \frac{1}{\ell - \log(2c) + 1} \right) \leq c \cdot n \cdot \frac{\log(2c)}{\ell - \log(2c) + 1} = c \cdot n \cdot \frac{\log(2c)}{\log \log n}.$$

For slight convenience we also remove the output edge e_{output} of the circuit; this way we can represent the output of the circuit in terms of the value of e_{output} . For a constant c' that depends only on c we remove at most

$$r := c' \cdot n / \log \log n$$

edges.

We define the depth of an edge $e = g \rightarrow g'$ as the depth of g , and the value of e on an input x as the value of the gate g on x .

For every input $x \in \{0, 1\}^n$ there exists a unique assignment h to the removed edges that corresponds to the computation of $C(x)$. Given an arbitrary assignment h and an input x we can check if h is the correct assignment by verifying if the value of every removed edge $e = g \rightarrow g'$ is correctly computed from (1) the values of the removed edges whose depth is less than that of e , and (2) the values of the input bits g is connected to. Since the depth of the component is $\leq (\log n)/2$ and the circuit has fan-in 2, at most \sqrt{n} input bits are connected to g ; we denote them by $x|_e$. Thus, for a fixed assignment h to the removed edges, the check for e can be implemented by a function $f_h^e : \{0, 1\}^{\sqrt{n}} \rightarrow \{0, 1\}$ (when fed the $\leq \sqrt{n}$ values of the input bits connected to g , i.e. $x|_e$).

Induction on depth shows:

$$C(x) = 1 \Leftrightarrow \exists \text{ assignment } h \text{ to removed edges such that } h(e_{\text{output}}) = 1 \\ \text{and } \forall \text{ removed edge } e \text{ we have } f_h^e(x|_e) = 1.$$

We now claim that the above expression for the computation $C(x)$ can be implemented with the desired resources. Since we removed $r = c' \cdot n / \log \log n$ edges, the existential quantification over all assignments to these edges can be implemented with an \vee (OR) gate with fan-in 2^r . Each function $f_h^e(x|_e)$ can be implemented via brute-force by a CNF, i.e. a depth-2 $\wedge\vee$ circuit, of size $\sqrt{n} \cdot 2^{\sqrt{n}}$. For any fixed assignment h , we can combine the output \wedge gates of these CNF to implement the check

$$\forall \text{ removed edge } e : f_h^e(x|_e) = 1$$

by a CNF of size at most

$$r \cdot \sqrt{n} \cdot 2^{\sqrt{n}}.$$

Finally, accounting for the existential quantification over the values of the r removed edges, we get a circuit of depth 3 and size

$$2^r \cdot r \cdot \sqrt{n} \cdot 2^{\sqrt{n}} = 2^{O(c' \cdot n / \log \log n)}.$$

□

We note that both the bounds in Chapter 2 and in Claim 24 are for the parity function. It is not hard to see that parity has depth-3 circuits of size $2^{O(\sqrt{n})}$ (improving the bound one obtains by noting that parity has linear-size log-depth circuits and then applying Theorem 25). So to prove stronger bounds one needs to work with a different function.

In this chapter we have seen that proving an exponential lower bound for depth-3 circuits has significant consequences for log-depth circuits. We note that a similar result can be proved for arithmetic circuits: proving a sufficiently strong, exponential lower bound for depth-4 arithmetic circuits implies a lower bound on arithmetic circuits of unrestricted depth, see [AV08] and the references therein.

Acknowledgments. We are grateful to Leslie Valiant for useful discussions.

Chapter 4

Cryptography in bounded depth

Functions that are easy to compute but hard to invert, known as *one-way functions*, are fundamental building blocks of cryptography. These functions are the computational analogue of physical actions that are easy to perform yet hard to reverse, such as breaking a glass. In this chapter we present a breakthrough result by Applebaum, Ishai, and Kushilevitz [AIK06] which shows how to “compile” many one-way functions (for example the one based on the hardness of factoring) into one-way functions that have the notable property of being *4-local*: each output bit depends on at most 4 input bits, and in particular can be computed by a circuit of constant size (and depth). Specifically, the result shows that if there are one-way functions that are computable by small branching programs, a class that includes many popular candidates, then there exist 4-local one-way functions. This result reveals a surprising power of local computation: from the point of view of one-way functions, local computation is as powerful as branching programs.

4.1 Definitions and main result

We start with a short discussion of one-way functions. The reader is referred to Goldreich’s monograph [Gol01] for a more in-depth treatment. To begin, let us define “hard to invert.”

Definition 29 (Hard to invert). *A function $f : \{0, 1\}^n \rightarrow \{0, 1\}^\ell$ is w -hard to invert if every circuit $C : \{0, 1\}^\ell \rightarrow \{0, 1\}^n$ of size w fails to invert $f(x)$ except for a $1/w$ fraction of inputs:*

$$\Pr_{x \in \{0, 1\}^n} [C(f(x)) \in f^{-1}(f(x))] \leq 1/w.$$

Typically, a “one-way function” is a function $f : \{0, 1\}^n \rightarrow \{0, 1\}^{\ell = \text{poly}(n)}$ that is $\text{poly}(n)$ -time computable and also $n^{\omega(1)}$ -hard to invert for randomized algorithms (equivalently, uniform randomized circuits). To follow the style of previous chapters, we focus here on non-uniform models of computation. This also simplifies the exposition somewhat while retaining all the main ideas.

There are a number of candidate one-way functions based on various hardness assumptions such as the hardness of factoring integers. The following is a simple candidate related

to the NP-complete problem *subset sum*, which in turn is a special case of the NP-complete problem *knapsack*. For more on this candidate the reader may consult [IN96].

Example 30 (Candidate one-way function based on the hardness of subset sum). *The function $S_n : \{0, 1\}^{n=k^2+k} \rightarrow \{0, 1\}^{\text{poly}(k)}$ is defined as*

$$S_n(x^1, x^2, \dots, x^k, y) := \left(x^1, x^2, \dots, x^k, \sum_{i:y_i=1} x^i \right),$$

where $|x^1| = |x^2| = \dots = |x^k| = |y| = k$ and y_i denotes the i -th bit of y . The function S_n is computable in time $\text{poly}(n)$ but may be w -hard to invert for some $w = n^{\omega(1)}$.

As is the case for the function S_n in Example 30, the computation of many candidate one-way functions only involves basic arithmetic. Therefore, such candidates can be implemented in restricted computational models, such as log-depth circuits with fan-in 2 (cf. [BCH86]). On the other hand, S_n cannot be computed by the more restricted constant-depth circuits with unbounded fan-in. This is because it is easy to see that computing it requires computing parity, which those circuits cannot do as proved in Chapter 2.

A first surprise comes from the work of Impagliazzo and Naor [IN96] who show that, although small constant-depth circuits with unbounded fan-in cannot compute S_n , they can generate random outputs of the function S_n . This is enough to obtain that, if S_n is one-way, then there is a closely related one-way function computable by such constant-depth circuits. This technique from [IN96] is used later in [Vio05] for depth-efficient constructions of pseudorandom generators from one-to-one one-way functions.

It is natural at this point to ask what is the minimal technology required to compute one-way functions. Among the many formalizations of this question, we specifically ask whether there are one-way functions such that each of their output bits depends only on a constant number of input bits (hence, it is computable by a circuit of constant size and depth):

Definition 31. *A function $f : \{0, 1\}^n \rightarrow \{0, 1\}^\ell$ is c -local if each output bit of f depends on c input bits.*

Historically, there have been proposals for $O(1)$ -local one-way functions [Gol00], but these were based on hardness assumptions that have received less attention than more popular ones such as the hardness of factoring or of solving subset sum. Indeed, the property of being $O(1)$ -local appears very restrictive. The function S_n in Example 30 is not $O(1)$ -local, and it also seems hard to implement the technique by Impagliazzo and Naor mentioned above using bounded locality. Nevertheless, a corollary of the main result of this chapter is that if S_n is one-way, then there is a 4-local one-way function!

Our plan is to construct one-way functions that are local starting from one-way functions that are computable by the more powerful model of small *branching programs*. This model corresponds to log-space computation, conveniently abstracting from such details as pointers to the input. It is at least as powerful (and believed to be more powerful than) log-depth fan-in 2 computation.

Definition 32 (Branching program). A branching program $G : \{0, 1\}^n \rightarrow \{0, 1\}$ of size s is a directed acyclic graph with s nodes. Exactly one node is labeled *Start* and has in-degree 0. There is at least one sink node (out-degree 0). Exactly one sink node is labeled *Accept*. Each non-sink node is labeled by a variable $x_i \in \{x_1, \dots, x_n\}$ and has two outgoing edges labeled 0 and 1 respectively.

We say that a function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ is computable by a branching program of size s if there is a branching program of size s such that, for every $x \in \{0, 1\}^n$, $f(x) = 1$ if and only if x induces in the branching program a chain of transitions that lead the start node to the node *Accept*.

We say that a non-boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}^\ell$ is computable by a branching program of size s if for $i = 1, \dots, \ell$ the i -th output bit of f is computable by a branching program of size s_i , and $\sum s_i \leq s$.

For circuits we measured size by counting edges, but for branching programs we find it more convenient to count nodes because the out-degree is bounded. We note that, like for circuits, the size of a branching program is also an upper bound on the output length of the function computed by it; we use this later.

Whereas the results in [AIK06] apply to more general types of branching programs, for simplicity here we focus on the above type.

We now state the main theorem of this chapter, which one would typically apply with $s = \text{poly}(n)$.

Theorem 33 (Branching program one-way functions \Leftrightarrow local one-way functions; [AIK06]). Suppose $f : \{0, 1\}^n \rightarrow \{0, 1\}^\ell$ is computable by a branching program of size $s \geq n$ and f is w -hard to invert. Then there is $f' : \{0, 1\}^{n'} \rightarrow \{0, 1\}^{\ell'}$ such that f' is 4-local, $n' \leq s^c$, $\ell' \leq s^c$, and f' is w' -hard to invert for $w' \geq w - s^c$, where c is a universal constant.

The above theorem shows how we can construct a 4-local function f' from f with only a polynomial loss in security.

The proof of Theorem 33 relies on a powerful intuition: the property of being hard to invert is not too tied to the specific map $x \rightarrow f(x)$, but is inherited by suitable “encodings” of f , which in turn may be computable with far less resources than those a direct implementation of f requires. Formally, we use the following notion of *randomized encoding*, that was introduced and studied in earlier work, cf. [AIK06].

Definition 34 (Randomized encoding). A randomized encoding of $f : \{0, 1\}^n \rightarrow \{0, 1\}^\ell$ with blow-up t is a function $f' : \{0, 1\}^n \times \{0, 1\}^t \rightarrow \{0, 1\}^u$, such that:

1. For every x, x', r, r' , if $f(x) \neq f(x')$ then $f'(x, r) \neq f'(x', r')$, and
2. there exists a distribution D on circuits C of size t such that, for every x , the distribution $C(f(x))$ for $C \in D$ is equal to the distribution of $f'(x, r)$ for uniform r (and so in particular, $t \geq u$).

Whereas $f'(x, r) := f(x)$ satisfies both properties in the above Definition 34, for our application we need a randomized encoding f' that is much easier to compute than f . This can be trivially obtained for each of the two properties alone, respectively using the functions $f'(x, r) := x$ and $f'(x, r) := 0$. Their combination however is not trivial to satisfy. Intuitively, it asks for a simple function f' that spreads each output $f(x)$ of f into a distribution $f'(x, r)$ that depends on $f(x)$ and only on that.

The proof of Theorem 33 goes by showing that functions f computable by small branching programs have randomized encodings f' with small blow-up and the surprising feature that they are 4-local: every output bit is a function of at most 4 input bits.

Theorem 35 (Local randomized encoding for branching programs). *Let $f : \{0, 1\}^n \rightarrow \{0, 1\}^\ell$ be computable by a branching program with $s \geq n$ nodes. Then there exists a 4-local randomized encoding $f' : \{0, 1\}^n \times \{0, 1\}^t \rightarrow \{0, 1\}^u$ of f with blow-up $t = s^c$, where c is a universal constant.*

It is instructive to prove Theorem 33 assuming the above theorem.

Proof of Theorem 33 assuming Theorem 35. The function in the conclusion of the theorem is the randomized encoding f' of f , seen as a 1-argument function. By our assumption, f' is 4-local, has input length $n' = n + t = n + \text{poly}(s) = \text{poly}(s)$, and output length $\ell' = u \leq t = \text{poly}(s)$. We need to prove that f' is hard to invert. To prove this, assume for contradiction that $A' : \{0, 1\}^{\ell'} \rightarrow \{0, 1\}^{n'}$ is a circuit of size at most $w' := w - t$ that inverts f' with probability $\geq 1/w$ (no loss in the inversion probability):

$$\Pr_{x,r} [A'(f'(x, r)) \in f'^{-1}(f'(x, r))] \geq 1/w. \quad (4.1)$$

Define a distribution on circuits $A : \{0, 1\}^\ell \rightarrow \{0, 1\}^n$:

On input z : $A(z) := A'(C(z)) \mid_{n \in \{0, 1\}^n}$, where C is random from D .

Note that the size of A is $\leq w' + t = w$. Also, we have the derivation

$$\begin{aligned} \Pr_{x,A} [A(f(x)) \in f^{-1}(f(x))] &= \Pr_{x,C} [A'(C(f(x))) \mid_{n \in f^{-1}(f(x))}] \\ &= \Pr_{x,r} [A'(f'(x, r)) \mid_{n \in f^{-1}(f(x))}] \\ &\geq \Pr_{x,r} [A'(f'(x, r)) \in f'^{-1}(f'(x, r))] \\ &\geq 1/w. \end{aligned}$$

The first equality is by definition. The second is Property 2 of Definition 34. The next inequality is (the counterpositive of) Property 1 of Definition 34: whenever the output $A'(f'(x, r)) =: (x', r')$ satisfies $f'(x, r) = f'(x', r')$ then also $f(x) = f(x')$. The final inequality is our assumption (4.1).

The above derivation shows that A inverts f with probability $\geq 1/w$. Fixing the choice of $C \in D$ in the definition of A we contradict the assumption that f is w -hard to invert. \square

Thus, to prove Theorem 33 we may prove Theorem 35, i.e., construct appropriate randomized encodings. We begin in the next section with some preliminaries and then proceed with the main construction.

4.1.1 Preliminaries on randomized encodings

In this section we state and prove a couple of lemmas that allow us break the construction of our randomized encoding in stages. Their proofs are straightforward from the definitions.

The first lemma lets us focus on boolean functions: we can construct a randomized encoding for $f : \{0, 1\}^n \rightarrow \{0, 1\}^\ell$ by concatenating randomized encodings for each output bit of f .

Lemma 36 (Concatenation of randomized encodings). *For $i = 1, \dots, \ell$, let $f_i : \{0, 1\}^n \rightarrow \{0, 1\}$ be the boolean function computing the i -th bit of $f : \{0, 1\}^n \rightarrow \{0, 1\}^\ell$. Suppose $f'_i : \{0, 1\}^n \times \{0, 1\}^t \rightarrow \{0, 1\}^u$ is a randomized encoding of f_i with blow-up t , for every i . Then $f' : \{0, 1\}^n \times \{0, 1\}^{\ell \cdot t} \rightarrow \{0, 1\}^{\ell \cdot u}$ defined as*

$$f'(x, (r_1, \dots, r_\ell)) := (f_1(x, r_1), f_2(x, r_2), \dots, f_\ell(x, r_\ell))$$

is a randomized encoding of f with blow-up $\ell \cdot t$.

Proof. Property 1: If $f(x) \neq f(x')$ then there is some $i \leq \ell$ such that $f_i(x) \neq f_i(x')$. Hence for any r_i, r'_i we have $f'_i(x, r_i) \neq f'_i(x, r'_i)$ and so for any $r, r' \in \{0, 1\}^{\ell \cdot t}$ we have $f'(x, r) \neq f'(x, r')$.

Property 2: For $1 \leq i \leq \ell$, let D_i be a distribution on circuits C_i of size t such that for any x the distribution of $C_i(f_i(x))$ for random $C_i \in D_i$ is the same as that of $f'_i(x, r_i)$ for random r_i . Define D to be the distribution on circuits C that on input $z \in \{0, 1\}^{\ell \cdot t}$ output $(C_1(z_1), C_2(z_2), \dots, C_\ell(z_\ell))$ where the circuits C_i are independently chosen from the respective distributions D_i . By the properties of D_i and the definition of f' , $C(f(x))$ is distributed like $f'(x, r)$ for random r . To conclude, note that the circuits C have size $\leq \ell \cdot t$. \square

The next lemma lets us compose randomized encodings. Jumping ahead, we will first show that small branching programs have randomized encodings where each output bit is a polynomial over $\{0, 1\}$ of degree 3. We will then push further this already striking encoding by showing that every such degree-3 polynomial has a 4-local randomized encoding.

Lemma 37 (Composition of randomized encodings). *Let $f : \{0, 1\}^n \rightarrow \{0, 1\}$ be a function with a randomized encoding $f_1 : \{0, 1\}^n \times \{0, 1\}^{t_1} \rightarrow \{0, 1\}^{u_1}$ with blow-up t_1 . Let $f_2 : \{0, 1\}^{n+t_1} \times \{0, 1\}^{t_2} \rightarrow \{0, 1\}^{u_2}$ be a randomized encoding of f_1 , seen as a one-argument function, with blow-up t_2 . Then $f' : \{0, 1\}^n \times \{0, 1\}^{t_1+t_2} \rightarrow \{0, 1\}^{u_2}$ defined as*

$$f'(x, (r_1, r_2)) := f_2((x, r_1), r_2)$$

is a randomized encoding of f with blow-up $t_1 + t_2$.

Actually, “composition” may be a bit of a misnomer, as f' is just f_2 , not the composition of f_1 and f_2 . The proof however does compose the circuits associated with f_1 and f_2 .

Proof. Property 1: Suppose $f(x) \neq f(x')$. Then for every r_1 , $f_1(x, r_1) \neq f_1(x', r_1)$. And so for every r_2 , $f_2((x, r_1), r_2) \neq f_2((x', r_1), r_2)$.

Property 2: Let $C_i \in D_i$ be random circuits corresponding to Property 2 of f_i , $i = 1, 2$. Recall that for every x , $C_1(f(x))$ is distributed like $f_1(x, r_1)$ for uniform r_1 . Hence, $C_2(C_1(f(x)))$ is distributed like $C_2(f_1(x, r_1))$, and the latter is in turn distributed like $f_2((x, r_1), r_2) = f'(x, (r_1, r_2))$ for uniform r_1, r_2 . Therefore $C_2(C_1(\cdot))$ is a distribution of circuits satisfying Property 2. Noting that these circuits have size $t_1 + t_2$ completes the proof. \square

In the next two sections we construct randomized encodings. As mentioned before, in light of Lemma 36 we focus on boolean functions. And in light of Lemma 36 we do this in two steps, first we encode branching programs using degree-3 polynomials over $\{0, 1\}$, then we encode the latter by 4-local functions.

4.1.2 Encoding branching programs by degree-3 polynomials

Let $f : \{0, 1\}^n \rightarrow \{0, 1\}$ be a function computable by a branching program of size s , and recall that this program corresponds to a directed acyclic graph on s nodes, denoted as G , with the following properties: there is a distinguished *Start* node with in-degree 0 and a distinguished *Accept* node with in-degree 0, and each non-sink node is labeled with “ x_i ,” for some $i \in [n]$, and has two outgoing edges labeled “0” and “1.”

We fix a topological ordering of the s nodes such that *Start* is the first and *Accept* the last (i.e., the s -th). Given any string $x \in \{0, 1\}^n$, define G_x to be the adjacency matrix of the graph induced by x on G 's nodes. Specifically, any node labeled x_i has only the outgoing edge labeled with the i th bit of x . Then, we can see that $f(x) = 1$ if and only if the accept node is reachable from the start node in the graph associated to G_x .

Consider the matrix $M(x) := G_x + I$, the sum of G_x and the identity matrix. With the nodes of G_x ordered according to the fixed topological order mentioned above, $M(x)$ is an upper-triangular matrix, with 1s along the diagonal, and at most one additional 1 per row (to the right of the diagonal). For example, $M(x) = G_x + I$ may look like

$$\begin{pmatrix} 1 & 0 & 0 & \cdots & 1 & \cdots & 0 \\ 0 & 1 & 0 & 1 & \cdots & 0 & 0 \\ 0 & 0 & 1 & 0 & \cdots & 1 & 0 \\ \vdots & & & \ddots & & & \vdots \\ & & & & & 1 & 0 \\ 0 & 0 & \cdots & & 0 & 1 \end{pmatrix}.$$

Then, we define $L(x)$ to be $M(x)$ with the first column and last row removed. In the above example, $L(x)$ is the $(s - 1) \times (s - 1)$ matrix

$$\begin{pmatrix} 0 & 0 & \cdots & 1 & \cdots & 0 \\ 1 & 0 & 1 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 1 & 0 \\ \vdots & & \ddots & & & \vdots \\ 0 & \cdots & & & 1 & 0 \end{pmatrix}.$$

We let

$$m := s - 1$$

and note that $L(x)$ is an $m \times m$ matrix.

Observe that the column number in the first row of $L(x)$ tells us to which node we arrive after taking one step from the node *Start*. Loosely speaking, we will now see how to continue taking steps in the branching program, by multiplying $L(x)$ by appropriate matrices. This will have the effect of moving the 1s in the matrix (on or above the diagonal) more and more towards the right, ultimately transforming $L(x)$ into a matrix which contains $f(x)$ in the upper right corner and is 0 everywhere else (except for a shifted diagonal of 1s which does not depend on x). This transformed $L(x)$ contains no information about x except for $f(x)$, and thus is intuitively suitable for a randomized encoding (cf. discussion after Definition 34).

We start by defining two families of matrices we use to transform $L(x)$. All the matrices are over $\{0, 1\}$, with addition being bit XOR.

Definition 38. R_1 is the family of upper triangular matrices with 1s along the diagonal:

$$\begin{pmatrix} 1 & * & \cdots & * \\ & 1 & \ddots & * \\ & & \ddots & \vdots \\ 0 & & & 1 & * \\ & & & & 1 \end{pmatrix},$$

where each occurrence of $*$ stands for an element of $\{0, 1\}$.

Multiplying a matrix M on the left with a matrix in R_1 corresponds to summing to each row of M a linear combination of the rows below it. For example,

$$\begin{pmatrix} 1 & a & b \\ 0 & 1 & c \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{pmatrix} = \begin{pmatrix} x_1 + ax_2 + bx_3 & y_1 + ay_2 + by_3 & z_1 + az_2 + bz_3 \\ x_2 + cx_3 & y_2 + cy_3 & z_2 + cz_3 \\ x_3 & y_3 & z_3 \end{pmatrix}.$$

Definition 39. R_2 is the family of matrices with 1s along the diagonal, and any other 1s in the last column:

$$\begin{pmatrix} 1 & 0 & \cdots & 0 & * \\ & 1 & & & * \\ & & \ddots & & \vdots \\ 0 & & & 1 & * \\ & & & & 1 \end{pmatrix}.$$

Multiplying a matrix M on the right with a matrix in R_2 corresponds to summing to the last column of M a linear combination of the other columns. For example,

$$\begin{pmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{pmatrix} \begin{pmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} x_1 & y_1 & ax_1 + by_1 + z_1 \\ x_2 & y_2 & ax_2 + by_2 + z_2 \\ x_3 & y_3 & ax_3 + by_3 + z_3 \end{pmatrix}.$$

It is important to note that both R_1 and R_2 are algebraic groups, which is not difficult to see given what operations multiplying by the matrices corresponds to.

Fact 40. Both R_1 and R_2 form groups under matrix multiplication.

We now show the promised transformation of $L(x)$ using these matrices.

Lemma 41. For all $x \in \{0, 1\}^n$, there exist matrices $r_1 \in R_1$ and $r_2 \in R_2$ such that

$$r_1 \cdot L(x) \cdot r_2 = \begin{pmatrix} 0 & \cdots & 0 & f(x) \\ 1 & & & 0 \\ & \ddots & & \vdots \\ 0 & & 1 & 0 \end{pmatrix}.$$

Proof. Let c be the column number of the 1 in the first row of $L(x)$ (numbering the columns 1 through m). Recall that this means c is the number of the node immediately following the start node in G_x . We will “push” this 1 to the right by repeatedly summing the lower rows to the first. To start, we sum row $c+1$ to row 1, which has two effects: (i) the entry at $(1, c)$ is zeroed, because we are working over $\text{GF}(2)$ and row $c+1$ has a 1 in column c ; (ii) the entry at $(1, d)$ is set to 1, where d is the node to which c points. We then sum row $d+1$ to row 1, and continue in this manner, ending when one of the following conditions holds:

- there is a 1 in entry $(1, m)$. This indicates that there is a path from the start node to the accept node in G_x , and thus that $f(x) = 1$. Or,
- the first row consists of all 0s. This indicates that a non-accepting sink was reached, and thus that $f(x) = 0$.

A key observation here is that, in either case, the upper right corner has the value $f(x)$. Also, note that this process will always terminate because G_x is acyclic.

We repeat the process for each lower row i , pushing any 1 at position $(i, j \geq i)$ to the right until the portion of the row at positions $(i, j \geq i)$ is zeroed out, except possibly for the rightmost column. When we have done this for every row, we have a matrix in the following form:

$$\begin{pmatrix} 0 & \cdots & 0 & f(x) \\ 1 & & \ddots & * \\ & \ddots & & \vdots \\ 0 & & & * \\ & & 1 & 0 \end{pmatrix}.$$

Each operation performed so far has been summing a row of $L(x)$ to a row above it; as previously mentioned, these operations can be performed by multiplying on the left by a matrix from R_1 . Composing the multiplications then, we have shown that for every $x \in \{0, 1\}^n$ there is $r_1 \in R_1$ such that $r_1 \cdot L(x)$ has the above form.

To complete the lemma we just need to zero out the last column except for the top position $(1, m)$. For this, we choose $r_2 \in R_2$ to be the matrix which, when multiplied on the right, causes the first $m - 1$ columns of $r_1 \cdot L(x)$ to be summed to the last column so as to zero out all entries below the top. Then, $r_1 \cdot L(x) \cdot r_2$ has the form given in the statement of the lemma. \square

This construction leads naturally to a randomized encoding. In its definition, for slight convenience, we identify an $m \times m$ matrix with its representation as an m^2 -bit string. This is somewhat wasteful, because the matrices $r_1 \in R_1, r_2 \in R_2$, and $r_1 \cdot L(x) \cdot r_2$ can each be specified using less than m^2 bits (for example, $m(m - 1)/2$ bits and $m - 1$ bits are needed to represent matrices in R_1 and R_2 , respectively). But this difference is negligible in our context.

Theorem 42 (Randomized encoding of branching programs by degree-3 polynomials). *Let $f : \{0, 1\}^n \rightarrow \{0, 1\}$ be computable by a branching program with $m + 1$ nodes. Then, the function $f' : \{0, 1\}^n \times \left(\{0, 1\}^{m^2} \times \{0, 1\}^{m^2}\right) \rightarrow \{0, 1\}^{m^2}$ defined by*

$$f'(x, (r_1, r_2)) := r_1 \cdot L(x) \cdot r_2$$

is a randomized encoding of f with blow-up $\text{poly}(m)$ such that each output bit of f' is computable by a degree-3 polynomial over $\{0, 1\}$.

Proof. First, the fact that each output bit of f' is computable by a degree-3 polynomial follows immediately from the definition of f' and of matrix multiplication. To verify the two properties of randomized encodings, it is convenient to define $Z(x)$, for an input x , to be the $m \times m$ identity matrix with the diagonal shifted “southwest” and $f(x)$ in the upper right

corner:

$$Z(x) := \begin{pmatrix} 0 & \cdots & 0 & f(x) \\ 1 & & \ddots & 0 \\ & & \ddots & \vdots \\ 0 & & & 1 & 0 \end{pmatrix}.$$

This is the same matrix on the right side of the equation in the statement of Lemma 41.

We verify next the two properties.

1. We need to show that if $f(x) \neq f(x')$ then $f'(x, (r_1, r_2)) \neq f'(x', (r'_1, r'_2))$ for every r_1, r_2, r'_1, r'_2 .

Fix any input x , and let \hat{r}_1, \hat{r}_2 be the matrices guaranteed by Lemma 41 such that $\hat{r}_1 \cdot L(x) \cdot \hat{r}_2 = Z(x)$. Observe that the rank of $Z(x)$, denoted $\text{Rank}(Z(x))$, equals $m - 1 + f(x)$. Also notice that each matrix from R_1 or R_2 has full rank, and recall that multiplying by a full-rank matrix does not change rank. So we see that, for every r_1, r_2 :

$$\begin{aligned} \text{Rank}(r_1 \cdot L(x) \cdot r_2) &= \text{Rank}(\hat{r}_1 \cdot L(x) \cdot \hat{r}_2) \\ &= \text{Rank}(Z(x)) \\ &= m - 1 + f(x). \end{aligned}$$

Therefore, if $f(x) \neq f(x')$ then for every r_1, r'_1, r_2, r'_2 we have $\text{Rank}(r_1 \cdot L(x) \cdot r_2) \neq \text{Rank}(r'_1 \cdot L(x') \cdot r'_2)$ and hence $f'(x, (r_1, r_2)) \neq f'(x', (r'_1, r'_2))$.

2. We need to show a distribution D on circuits of size $\text{poly}(m)$ such that for any x the distribution of $C(f(x))$ for random $C \in D$ is equal to that of $f'(x, (r_1, r_2))$ for uniformly chosen r_1, r_2 .

For randomly chosen r_1, r_2 , define $C_{r_1, r_2}(f(x))$ to be the circuit which outputs $r_1 \cdot Z(x) \cdot r_2$. A simple implementation of matrix multiplication shows that C_{r_1, r_2} has size $\text{poly}(m)$. It only remains to show that, for any fixed x , the distribution of $r_1 \cdot Z(x) \cdot r_2$ for uniform r_1, r_2 is the same as that of $r_1 \cdot L(x) \cdot r_2$. Let \hat{r}_1, \hat{r}_2 be the matrices guaranteed by Lemma 41 such that $\hat{r}_1 \cdot L(x) \cdot \hat{r}_2 = Z(x)$. Note that since R_1 is a group, the distribution of r_1 is the same as that of $r_1 \cdot \hat{r}_1$ for uniform r_1 , and similarly for r_2 . Therefore, for uniform r_1, r_2 , the distribution

$$r_1 \cdot L(x) \cdot r_2$$

is the same as

$$r_1 \cdot \hat{r}_1 \cdot L(x) \cdot \hat{r}_2 \cdot r_2$$

which is the same as

$$r_1 \cdot Z(x) \cdot r_2.$$

□

4.1.3 Encoding polynomials locally

We now show how to transform the randomized encoding from the previous section into one which is 4-local. Each bit of the randomized encoding from Theorem 42 is computable by a degree-3 polynomial. So, applying the next theorem to each bit of Theorem 42's randomized encoding gives the desired result.

Theorem 43. *Any degree-3 polynomial $f : \{0, 1\}^n \rightarrow \{0, 1\}$ over $\{0, 1\}$ has a 4-local randomized encoding with blow-up $\text{poly}(n)$.*

Proof. We write f as the sum of its degree-3 monomials:

$$f(x) = T_1(x) + \cdots + T_k(x),$$

for some parameter k crudely bound as $k \leq \text{poly}(n)$. Let r, r' be randomly-chosen bit strings with $|r| = k$, $|r'| = k - 1$. We define a randomized encoding $f' : \{0, 1\}^n \times (\{0, 1\}^k \times \{0, 1\}^{k-1}) \rightarrow \{0, 1\}^{2k}$ as

$$f'(x, (r, r')) := \begin{pmatrix} T_1(x) + r_1, & T_2(x) + r_2, & \dots, & T_{k-1}(x) + r_{k-1}, & T_k(x) + r_k, \\ r_1 + r'_1, & r_1 + r_2 + r'_2, & \dots, & r'_{k-2} + r_{k-1} + r'_{k-1}, & r'_{k-1} + r_k. \end{pmatrix}$$

Note that f' is 4-local because each $T_i(x)$ is a degree-3 monomial. We now move to verifying the two properties of randomized encodings.

For the first, observe that if we sum all the bits of $f'(x, (r, r'))$, each bit of r and r' appears exactly twice, and each $T_i(x)$ appears exactly once. Therefore, because we are working over $\{0, 1\}$ (i.e., sum is XOR), the bits of $f'(x, (r, r'))$ sum to $f(x)$ for any r, r' ; so the first property of a randomized encoding holds.

For the second property, note that for any fixed x , the first $2k - 1$ bits of $f'(x, (r, r'))$ are uniformly distributed (for uniform r, r') while the last bit is the one which makes the parity of the entire output equal to $f(x)$.

It may be worthwhile to visualize this through the following equivalences, holding for every fixed x , where we use the symbol “ \equiv ” for “have the same distribution for random

r, r' :

$$\begin{aligned}
& f'(x, (r, r')) \\
&= (T_1(x) + r_1, T_2(x) + r_2, \dots, T_{k-1}(x) + r_{k-1}, T_k(x) + r_k, \\
&\quad r_1 + r'_1, r'_1 + r_2 + r'_2, \dots, r'_{k-2} + r_{k-1} + r'_{k-1}, r'_{k-1} + r_k) \\
&\equiv (r_1, r_2, \dots, r_{k-1}, r_k, \\
&\quad r_1 + T_1(x) + r'_1, r'_1 + r_2 + T_2(x) + r'_2, \dots, r'_{k-2} + r_{k-1} + T_{k-1}(x) + r'_{k-1}, r'_{k-1} + r_k + T_k(x)) \\
&\equiv (r_1, r_2, \dots, r_{k-1}, r_k, \\
&\quad r'_1, r'_1 + r_1 + r_2 + T_1(x) + T_2(x) + r'_2, \dots, r'_{k-2} + r_{k-1} + T_{k-1}(x) + r'_{k-1}, r'_{k-1} + r_k + T_k(x)) \\
&\dots \\
&\equiv (r_1, r_2, \dots, r_{k-1}, r_k, r'_1, r'_2, \dots, r'_{k-2}, r'_{k-1}, \sum_{i \leq k} r_i + \sum_{i \leq k-1} r'_i + \sum_{i \leq k} T_i(x)) \\
&\equiv (r_1, r_2, \dots, r_{k-1}, r_k, r'_1, r'_2, \dots, r'_{k-2}, r'_{k-1}, \sum_{i \leq k} r_i + \sum_{i \leq k-1} r'_i + f(x)).
\end{aligned}$$

Thus, our distribution is over circuits C which output $2k - 1$ bits at random, and then one final bit to ensure the parity is equal to $f(x)$:

$$C_{r,r'}(f(x)) := \left(r, r', \sum_{i \leq k} r_i + \sum_{i \leq k-1} r'_i + f(x) \right).$$

This computation can be easily implemented in size $O(k) = \text{poly}(n)$. □

4.1.4 Putting the encodings together

For completeness in this section we point out how to combine the randomized encodings seen before into one.

Theorem 35 (Local randomized encoding for branching programs). (Restated.) *Let $f : \{0, 1\}^n \rightarrow \{0, 1\}^\ell$ be computable by a branching program with $s \geq n$ nodes. Then there exists a 4-local randomized encoding $f' : \{0, 1\}^n \times \{0, 1\}^t \rightarrow \{0, 1\}^u$ of f with blow-up $t = s^c$, where c is a universal constant.*

Proof. We show for every $i \leq \ell$ a 4-local randomized encoding of the i -th bit of f with blow-up $t = \text{poly}(s)$; the result then follows from Lemma 36 (note that $\ell \leq s$ by Definition 32 of branching program).

Fix any $i \leq \ell$ and for convenience let $g := f_i : \{0, 1\}^n \rightarrow \{0, 1\}$ be the function computing the i -th output bit of f . By Theorem 42 g has a randomized encoding $g' : \{0, 1\}^n \times \{0, 1\}^{t'} \rightarrow \{0, 1\}^{u'}$ with blow-up $t' = \text{poly}(s)$ such that each output bit of g' is computable by a polynomial of degree 3. In turn, by Theorem 43 each of the $u' = \text{poly}(s)$ output bits g'_j of g' , seen as a 1-argument function, has a 4-local randomized encoding $g''_j : \{0, 1\}^{n+t'} \times \{0, 1\}^{t''_j} \rightarrow \{0, 1\}^{u''_j}$ with blow-up $t''_j = \text{poly}(n + t') = \text{poly}(s)$. (Note that $u' = \text{poly}(s)$ for,

as noted earlier, the output length of a randomized encoding is always less than the blow-up parameter, since by definition the latter is also an upper bound on the size of circuits computing outputs of the randomized encoding.)

Therefore, by Lemma 36, g' has a randomized encoding g'' with blow-up $t'' = u' \cdot \text{poly}(s) = \text{poly}(s)$. Thanks to Lemma 37 we can now compose the randomized encodings g' and g'' to get a 4-local randomized encoding for $g = f_i$ with blow-up $t' + t'' = \text{poly}(s)$. \square

Finally, we note that the minimal locality of one-way functions is likely to be 3: 2-local one-way functions do not exist (inverses can be found solving 2-satisfiability problems), while 3-local one-way functions can be based on the intractability of decoding random linear binary codes, see [AIK06].

Acknowledgments. We thank Yuval Ishai for helpful comments.

Bibliography

- [AB01] Noga Alon and Richard Beigel. Lower bounds for approximations by low degree polynomials over Z_m . In *16th Annual Conference on Computational Complexity*, pages 184–187. IEEE, June 18–21 2001.
- [AB09] Sanjeev Arora and Boaz Barak. *Computational complexity*. Cambridge University Press, Cambridge, 2009. A modern approach.
- [ABFR94] James Aspnes, Richard Beigel, Merrick Furst, and Steven Rudich. The expressive power of voting polynomials. *Combinatorica*, 14(2):135–148, 1994.
- [AGHP92] Noga Alon, Oded Goldreich, Johan Håstad, and René Peralta. Simple constructions of almost k -wise independent random variables. *Random Structures & Algorithms*, 3(3):289–304, 1992.
- [AIK06] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0 . *SIAM J. Comput.*, 36(4):845–888, 2006.
- [Ajt83] Miklós Ajtai. Σ_1^1 -formulae on finite structures. *Ann. Pure Appl. Logic*, 24(1):1–48, 1983.
- [AKK⁺03] Noga Alon, Tali Kaufman, Michael Krivelevich, Simon Litsyn, and Dana Ron. Testing low-degree polynomials over $GF(2)$. In *7th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, volume 2764 of *Lecture Notes in Computer Science*, pages 188–199. Springer, 2003.
- [AV08] Manindra Agrawal and V. Vinay. Arithmetic circuits: A chasm at depth four. In *49th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 67–75, 2008.
- [AW08] Scott Aaronson and Avi Wigderson. Algebrization: a new barrier in complexity theory. In *40th Annual ACM Symposium on the Theory of Computing (STOC)*, pages 731–740, 2008.
- [BCH86] Paul Beame, Stephen A. Cook, and H. James Hoover. Log depth circuits for division and related problems. *SIAM J. Comput.*, 15(4):994–1003, 1986.

- [BEHL08] Ido Ben-Eliezer, Rani Hod, and Shachar Lovett. Random low degree polynomials are hard to approximate. Manuscript, 2008.
- [Bei93] Richard Beigel. The polynomial method in circuit complexity. In *8th Annual Structure in Complexity Theory Conference*, pages 82–95. IEEE, 1993.
- [Bei94] Richard Beigel. When do extra majority gates help? $\text{polylog}(N)$ majority gates are equivalent to one. *Comput. Complexity*, 4(4):314–324, 1994. Special issue devoted to the 4th Annual McGill Workshop on Complexity Theory.
- [BGL06] Nayantara Bhatnagar, Parikshit Gopalan, and Richard J. Lipton. Symmetric polynomials over Z_m and simultaneous communication protocols. *J. Comput. Syst. Sci.*, 72(2):252–285, 2006.
- [BGS75] Theodore Baker, John Gill, and Robert Solovay. Relativizations of the $P=?NP$ question. *SIAM J. Comput.*, 4(4):431–442, 1975.
- [BNS92] László Babai, Noam Nisan, and Mária Szegedy. Multipart protocols, pseudorandom generators for logspace, and time-space trade-offs. *J. Comput. System Sci.*, 45(2):204–232, 1992.
- [Bou05] Jean Bourgain. Estimation of certain exponential sums arising in complexity theory. *C. R. Math. Acad. Sci. Paris*, 340(9):627–631, 2005.
- [Bra09] Mark Braverman. Poly-logarithmic independence fools AC^0 circuits. In *24th Conference on Computational Complexity (CCC)*. IEEE, 2009.
- [BS90] Ravi B. Boppana and Michael Sipser. The complexity of finite functions. In *Handbook of theoretical computer science, Vol. A*, pages 757–804. Elsevier, Amsterdam, 1990.
- [BSSVW03] Eli Ben-Sasson, Madhu Sudan, Salil Vadhan, and Avi Wigderson. Randomness-efficient low degree tests and short PCPs via epsilon-biased sets. In *35th Annual Symposium on Theory of Computing (STOC)*, pages 612–621. ACM, 2003.
- [BV07] Andrej Bogdanov and Emanuele Viola. Pseudorandom bits for polynomials. In *48th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 41–51. IEEE, 2007. To appear in *SIAM Journal on Computing*.
- [Cai89] Jin-Yi Cai. With probability one, a random oracle separates pspace from the polynomial-time hierarchy. *J. Comput. Syst. Sci.*, 38(1):68–85, 1989.
- [CGT96] Jin-Yi Cai, Frederic Green, and Thomas Thierauf. On the correlation of symmetric functions. *Mathematical Systems Theory*, 29(3):245–258, 1996.
- [Cha07] Arkadev Chattopadhyay. Discrepancy and the power of bottom fan-in in depth-three circuits. In *48th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 449–458. IEEE, 2007.

- [CK02] Peter Clote and Evangelos Kranakis. *Boolean Functions and Computation Models*. Springer, 2002.
- [CT93] Fan R. K. Chung and Prasad Tetali. Communication complexity and quasi randomness. *SIAM J. Discrete Math.*, 6(1):110–123, 1993.
- [CT06] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, 2006.
- [DP09] Devdatt P. Dubhashi and Alessandro Panconesi. *Concentration of measure for the analysis of randomized algorithms*. Cambridge University Press, Cambridge, 2009.
- [Fre95] Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, September 1995.
- [FSS84] Merrick L. Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, April 1984.
- [GHR92] Mikael Goldmann, Johan Håstad, and Alexander A. Razborov. Majority gates vs. general weighted threshold gates. *Computational Complexity*, 2:277–300, 1992.
- [GNW95] Oded Goldreich, Noam Nisan, and Avi Wigderson. On Yao’s XOR lemma. Technical Report TR95–050, *Electronic Colloquium on Computational Complexity*, March 1995. www.eccc.uni-trier.de/.
- [Gol00] Oded Goldreich. Candidate one-way functions based on expander graphs. Technical report, *Electronic Colloquium on Computational Complexity*, 2000.
- [Gol01] Oded Goldreich. *Foundations of Cryptography: Volume 1, Basic Tools*. Cambridge University Press, Cambridge, 2001.
- [Gow98] Timothy Gowers. A new proof of Szemerédi’s theorem for arithmetic progressions of length four. *Geom. Funct. Anal.*, 8(3):529–551, 1998.
- [Gow01] Timothy Gowers. A new proof of Szemerédi’s theorem. *Geom. Funct. Anal.*, 11(3):465–588, 2001.
- [GR07a] Andrew Granville and Zeév Rudnick. Uniform distribution. In *Equidistribution in Number Theory, An Introduction*, volume 237 of *NATO Science Series II: Mathematics, Physics and Chemistry*, pages 1–13. Springer, 2007.
- [GR07b] Frederic Green and Amitabha Roy. Uniqueness of optimal mod 3 circuits for parity. In *Dagstuhl Seminar Proceedings, Algebraic Methods in Computational Complexity*, volume 07411, 2007.

- [Gre04] Frederic Green. The correlation between parity and quadratic polynomials mod 3. *J. Comput. System Sci.*, 69(1):28–44, 2004.
- [Gro95] Vince Grolmusz. Separating the communication complexities of MOD m and MOD p circuits. *J. Comput. System Sci.*, 51(2):307–313, 1995.
- [GRS05] Frederic Green, Amitabha Roy, and Howard Straubing. Bounds on an exponential sum arising in Boolean circuit complexity. *C. R. Math. Acad. Sci. Paris*, 341(5):279–282, 2005.
- [GT06] Anna Gál and Vladimir Trifonov. On the correlation between parity and modular polynomials. In *31st International Symposium on Mathematical Foundations of Computer Science (MFCS)*, volume 4162 of *Lecture Notes in Computer Science*, pages 387–398. Springer, 2006.
- [GT07] Ben Green and Terence Tao. The distribution of polynomials over finite fields, with applications to the Gowers norms, 2007. arXiv:0711.3191v1.
- [GT08] Ben Green and Terence Tao. An inverse theorem for the Gowers $U^3(G)$ norm. *Proceedings of the Edinburgh Mathematical Society (Series 2)*, 51(01):73–153, 2008.
- [GV04] Dan Gutfreund and Emanuele Viola. Fooling parity tests with parity gates. In *8th International Workshop on Randomization and Computation (RANDOM)*, pages 381–392. Springer, 2004.
- [Han06] Kristoffer Arnsfelt Hansen. Lower bounds for circuits with few modular gates using exponential sums. *Electronic Colloquium on Computational Complexity*, Technical Report TR06-079, 2006. www.eccc.uni-trier.de/.
- [Hås87] Johan Håstad. *Computational limitations of small-depth circuits*. MIT Press, 1987.
- [Hea08] Alexander Healy. Randomness-efficient sampling within NC^1 . *Computational Complexity*, 17(1):3–37, April 2008.
- [HG91] Johan Håstad and Mikael Goldmann. On the power of small-depth threshold circuits. *Comput. Complexity*, 1(2):113–129, 1991.
- [HMP⁺93] András Hajnal, Wolfgang Maass, Pavel Pudlák, Máriaó Szegedy, and György Turán. Threshold circuits of bounded depth. *J. Comput. System Sci.*, 46(2):129–154, 1993.
- [HV06] Alexander Healy and Emanuele Viola. Constant-depth circuits for arithmetic in finite fields of characteristic two. In *23rd International Symposium on Theoretical Aspects of Computer Science (STACS)*, pages 672–683. Springer, 2006.

- [IM02] Kazuo Iwama and Hiroki Morizumi. An explicit lower bound of $5n - o(n)$ for boolean circuits. In *Mathematical Foundations of Computer Science (MFCS)*, pages 353–364, 2002.
- [IN96] Russell Impagliazzo and Moni Naor. Efficient cryptographic schemes provably as secure as subset sum. *Journal of Cryptology*, 9(4):199–216, Fall 1996.
- [IW97] Russell Impagliazzo and Avi Wigderson. $P = BPP$ if E requires exponential circuits: Derandomizing the XOR lemma. In *29th Annual Symposium on Theory of Computing (STOC)*, pages 220–229. ACM, 1997.
- [Kli01] Adam R. Klivans. On the derandomization of constant depth circuits. In *5th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*. Springer, 2001.
- [KN97] Eyal Kushilevitz and Noam Nisan. *Communication complexity*. Cambridge University Press, Cambridge, 1997.
- [LMS08] Shachar Lovett, Roy Meshulam, and Alex Samorodnitsky. Inverse conjecture for the Gowers norm is false. In *40th Annual Symposium on the Theory of Computing (STOC)*, pages 547–556. ACM, 2008.
- [Lov08] Shachar Lovett. Unconditional pseudorandom generators for low degree polynomials. In *40th Annual Symposium on the Theory of Computing (STOC)*, pages 557–562. ACM, 2008.
- [LVW93] Michael Luby, Boban Veličković, and Avi Wigderson. Deterministic approximate counting of depth-2 circuits. In *2nd Israeli Symposium on Theoretical Computer Science (ISTCS)*, pages 18–24, 1993.
- [Nep70] Valery A. Nepomnjaščii. Rudimentary predicates and Turing calculations. *Soviet Mathematics-Doklady*, 11(6):1462–1465, 1970.
- [Nis91] Noam Nisan. Pseudorandom bits for constant depth circuits. *Combinatorica*, 11(1):63–70, 1991.
- [NN93] Joseph Naor and Moni Naor. Small-bias probability spaces: efficient constructions and applications. *SIAM J. Comput.*, 22(4):838–856, 1993.
- [NW94] Noam Nisan and Avi Wigderson. Hardness vs randomness. *J. Computer & Systems Sciences*, 49(2):149–167, 1994.
- [Raz87] Alexander Razborov. Lower bounds on the dimension of schemes of bounded depth in a complete basis containing the logical addition function. *Mat. Zametki*, 41(4):598–607, 623, 1987.

- [Raz00] Ran Raz. The BNS-Chung criterion for multi-party communication complexity. *Comput. Complexity*, 9(2):113–122, 2000.
- [Ros08] Benjamin Rossman. On the constant-depth complexity of k -clique. In *40th Annual Symposium on the Theory of Computing (STOC)*. ACM, 2008.
- [RR97] Alexander Razborov and Steven Rudich. Natural proofs. *Journal of Computer and System Sciences*, 55(1):24–35, August 1997.
- [RW93] Alexander Razborov and Avi Wigderson. $n^{\Omega(\log n)}$ lower bounds on the size of depth-3 threshold circuits with AND gates at the bottom. *Inform. Process. Lett.*, 45(6):303–307, 1993.
- [Sam07] Alex Samorodnitsky. Low-degree tests at large distances. In *39th Annual Symposium on Theory of Computing (STOC)*, pages 506–515. ACM, 2007.
- [Smo87] Roman Smolensky. Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In *19th Annual Symposium on Theory of Computing*, pages 77–82. ACM, 1987.
- [SV08] Ronen Shaltiel and Emanuele Viola. Hardness amplification proofs require majority. In *40th Annual Symposium on the Theory of Computing (STOC)*, pages 589–598. ACM, 2008.
- [Val77] Leslie G. Valiant. Graph-theoretic arguments in low-level complexity. In *6th Symposium on Mathematical Foundations of Computer Science*, volume 53 of *Lecture Notes in Computer Science*, pages 162–176. Springer, 1977.
- [Val83] L. G. Valiant. Exponential lower bounds for restricted monotone circuits. In *15th annual ACM symposium on Theory of computing (STOC)*, pages 110–117. ACM, 1983.
- [Vio05] Emanuele Viola. On constructing parallel pseudorandom generators from one-way functions. In *20th Annual Conference on Computational Complexity (CCC)*, pages 183–197. IEEE, 2005.
- [Vio06a] Emanuele Viola. *The Complexity of Hardness Amplification and Derandomization*. PhD thesis, Harvard University, 2006. www.eccc.uni-trier.de/.
- [Vio06b] Emanuele Viola. New correlation bounds for $\text{GF}(2)$ polynomials using Gowers uniformity. *Electronic Colloquium on Computational Complexity*, Technical Report TR06-097, 2006. www.eccc.uni-trier.de/.
- [Vio07] Emanuele Viola. Pseudorandom bits for constant-depth circuits with few arbitrary symmetric gates. *SIAM Journal on Computing*, 36(5):1387–1403, 2007.

- [Vio09a] Emanuele Viola. Gems of theoretical computer science, 2009. Lecture notes of the class taught at Northeastern University. Available at <http://www.ccs.neu.edu/home/viola/classes/gems-08/index.html>.
- [Vio09b] Emanuele Viola. The sum of d small-bias generators fools polynomials of degree d . *Computational Complexity*, 18(2):209–217, 2009.
- [VW08] Emanuele Viola and Avi Wigderson. Norms, XOR lemmas, and lower bounds for GF(2) polynomials and multiparty protocols. *Theory of Computing*, 4:137–168, 2008.
- [Yao85] Andrew C-C. Yao. Separating the polynomial-time hierarchy by oracles. In *Proc. 26th annual symposium on Foundations of computer science*, pages 1–10. IEEE Press, 1985.