The Complexity of Hardness Amplification and Derandomization

A thesis presented
by

Emanuele Viola

to
The Division of Engineering and Applied Sciences
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy
in the subject of

Computer Science

Harvard University
Cambridge, Massachusetts
May 2006

Thesis advisor
Salil P. Vadhan

Author
Emanuele Viola

The Complexity of Hardness Amplification and Derandomization

Abstract

This thesis studies the interplay between randomness and computation. We investigate this interplay from the perspectives of *hardness amplification* and *derandomization.*

Hardness amplification is the task of taking a function that is hard to compute on some input or on some fraction of inputs, and producing a new function that is very hard on average, i.e. hard to compute on a fraction of inputs that is as large as possible. Hardness amplification is an important step toward understanding average-case hardness, and is also motivated by modern cryptography, which for the most part relies on the existence of a very average-case hard function in NP. Our results in this area include the following:

- We show that if NP contains a function that is hard to compute on a constant fraction of inputs then NP contains a function that is hard to compute on a fraction of inputs that is exponentially close to one-half, as opposed to polynomially close to one-half in previous work.

- We show that there is no black-box construction of an average-case hard function in NP starting from a worst-case hard function.

Derandomization studies the possibility of removing randomness from probabilistic algorithms. Its study is key to understanding the power of randomness in computation, and has recently led to several algorithmic breakthroughs. Our contributions to this area include the following:

- We construct a new pseudorandom generator that stretches a random seed into a much longer sequence that looks random to any small constant-depth circuit with a few arbitrary symmetric gates, such as Parity or Majority.

- We show that any black-box simulation of randomized polynomial time in the second level of the polynomial-time hierarchy must incur a quadratic slow-down in the running time, which matches the running time of known simulations. We also exhibit a quasilinear-time simulation at the third level of the polynomial-time hierarchy.

Contents

Bibliographic Note

Chapter 2 is based on the paper "Using Nondeterminism to Amplify Hardness" which is joint work with Alexander Healy and Salil Vadhan and appeared in *SIAM Journal on Computing* [HVV]. A preliminary version of this paper appeared in the proceedings of the 2004 annual ACM Symposium on the Theory of Computing.

Chapter 3 is based on the papers "The Complexity of Constructing Pseudorandom Generators from Hard Functions" and "On Constructing Parallel Pseudorandom Generators from One-Way Functions."

The paper "The Complexity of Constructing Pseudorandom Generators from Hard Functions" appeared in the journal *Computational Complexity* [Vio1]. A preliminary version of this paper appeared in the proceedings of the 2003 annual IEEE Conference on Computational Complexity.

The paper "On Constructing Parallel Pseudorandom Generators from One-Way Functions" appeared in the proceedings of the 2005 annual IEEE Conference on Computational Complexity [Vio2].

Chapter 4 is based on the paper "Pseudorandom Bits for Constant-Depth Circuits with Few Arbitrary Symmetric Gates" which has been accepted for publication in *SIAM Journal on Computing*. A preliminary version of this paper appeared in the proceedings of the 2005 annual IEEE Conference on Computational Complexity [Vio3].

Acknowledgments

I am deeply grateful to Salil Vadhan for being a better advisor than what I could have ever imagined. Salil has always been generous with his time and help. At each of our meetings, I felt like he was lending me his mind with all its sharpness of thought, for as long as I needed it. Salil's research and undaunted pursuit of clarity have been of great inspiration to me; his passionate and endless encouragement prompted me to work on this thesis. Although he officially co-authored only the results described in Chapter 2, every chapter in this thesis owes very much to Salil.

Thanks to Alex Healy, who collaborated with me on the research described in Chapter 2 and on [HV], and to Dan Gutfreund, who collaborated with me on [GV], for sharing my passion for complexity theory and for their friendship.

During this research I was very lucky to interact with several researchers that made my graduate experience memorable and productive. I am particularly grateful to Oded Goldreich for taking much time to tell me some of his views on doing research, to Luca Trevisan for the opportunity to visit his group at Berkeley University, to Harry Buhrman for the opportunity to visit his group at CWI in Amsterdam, and to the respective organizers for inviting me to the Fall Central Section Meeting of the American Mathematical Society (University of Nebraska-Lincoln, October 2005), and to the Dagstuhl seminar "Complexity of Boolean Functions" (Schloss Dagstuhl, March 2006).

I would like to thank the following people for inspiring conversations: Eric Allender, Andrej Bogdanov, Prahladh Harsha, Valentine Kabanets, Adam Klivans, Troy Lee, Ryan O'Donnell, Omer Reingold, Rocco Servedio, Ronen Shaltiel, Amir Shpilka, Madhu Sudan, Chris Umans, Dieter van Melkebeek, Hoeteck Wee, Avi Wigderson, David Zuckerman, and those that I forgot. I also would like to thank Michael Mitzenmacher, Michael Rabin, and Leslie Valiant for serving on my thesis committee.

Studying at Harvard University was very pleasant. I would like to thank all the wonderful teachers I had, the academic staff, and my officemates in Maxwell Dworkin, especially Johnny Chen, Vitaly Feldman, and Minh-Huyen Nguyen.

I also would like to express my gratitude to my undergraduate teachers at University of Rome "La Sapienza" for their contribution to my education.

This thesis is dedicated to those who are closest to me, whose affection and understanding accompany me through my life: my parents, Paola end Eugenio, my sister, Alessandra, and my twin soul, Jasmina.

This research was supported by NSF grant CCR-0133096, US-Israel BSF grant 2002246, and ONR grant N-00014-04-1-0478.

Chapter 1

Introduction

Computational complexity theory is the theory that investigates the amount of resources required to solve computational problems. For example, one might be interested in how much time it takes to find a factorization of a given integer number. Within this theory, the present thesis focuses mostly on the interplay between randomness and computation.

There are at least two ways in which randomness plays an important role in computation. The first is in the choice of the problem instance. Going back to our example of factorization, we can ask how much time it takes to factor an integer number *that is chosen at random*. Besides their theoretical interest, such questions are of practical relevance; for example, the security of most cryptography currently in use relies on the assumption that factoring randomly chosen integer numbers requires an infeasible amount of time. The investigation of these questions belongs to the area of *average-case complexity* and leads to the study of *hardness amplification* (discussed below in Section 1.1), which is the task of constructing problems that are as hard on average as possible.

The second way in which randomness can play a role in computation is in allowing our algorithms to be randomized. For example, does it become easier to factor a given integer number if we are given access to a source of randomness? While there are several computational tasks that we only know how to perform efficiently using randomness, researchers have found striking evidence that, in fact, it is always possible to remove randomness from probabilistic algorithms to obtain efficient deterministic counterparts. The study of the possibility of removing randomness from computation belongs to the field of *derandomization* (discussed below in Section 1.2), and has led to several algorithmic breakthroughs, such as the existence of efficient deterministic algorithms to decide whether a given number is prime [AKS].

As we will see, hardness amplification and derandomization are closely related. We now proceed to describe these two areas in more detail and highlight our contributions.

1.1 Hardness amplification

Average-case complexity is the study of the computational complexity of functions over random choice of the input. This is a fundamental topic in complexity theory, the study of which has at least two distinct motivations. First, it may provide more meaningful explanations than worst-case complexity about the intractability of problem instances actually encountered in practice. Second, it provides us with methods to generate hard instances, allowing us to harness intractability for 'useful' ends such as derandomization and cryptography. In fact, most modern cryptography requires the existence of a function in NP that is very *average-case* hard, i.e. hard to compute over random choice of the input.

The ultimate goal of this research is to understand whether important complexity classes, such as NP, contain functions that are *average-case* hard. Many researchers believe that this is indeed the case, but, just like for the "$\mathsf{P} \stackrel{?}{=} \mathsf{NP}$" question, the answer is still unknown and, some people argue, very far.

A more modest goal is that of *hardness amplification.* Hardness amplification seeks connections between the existence of average-case hard functions and other "seemingly weaker" assumptions, such as the existence of "mildly" average-case hard functions, or the existence of functions that are hard in the worst-case. An important step toward understanding the possibility and implications of the existence of average-case hard functions is the *hardness amplification problem*, which is the problem of converting a given function that is "somewhat hard" into a new function that is "much harder" on average. More precisely, we define the hardness of a function as the fraction of inputs on which it is hard to compute. The hardness amplification problem is then the problem of taking a function that is hard to compute on some input or on some fraction of inputs, and converting it into a new function that is hard to compute on a fraction of inputs that is as large as possible. In this conversion process, we would also like the new function to be not much more "complex" than the original function, which is captured by fixing a complexity class C (e.g., $\mathsf{C} = \mathsf{NP}$) and asking that the new function belongs to C whenever the original function does. We refer to this as *hardness amplification within* C.

We distinguish between two types of hardness amplification problems, according to the hardness of the initial function.

Amplifying noticeable hardness: The first type corresponds to starting with a function that is "noticeably" or "mildly" hard on average. For example, we can think of a function that any efficient algorithm fails to compute on a constant fraction of inputs. From this function, we would like to construct a new function that is as hard on average as possible. How hard can we hope this new function to be? In this thesis we will mostly focus on boolean functions, i.e. functions with output in $\{0, 1\}$. Such functions are either 0 or 1 for at least half of their inputs, and therefore they can trivially be computed on half of their inputs. Our goal will be getting as close to hardness one-half as possible. Specifically, we will measure the distance from one-half as a function of the input length, and we will want this distance to vanish as quickly as possible.

In this thesis we show how to amplify hardness within NP with nearly optimal parameters. Specifically, we obtain hardness exponentially close to one-half, thereby improving on a previous result by O'Donnell [O'D] which achieves hardness polynomially close to one-half.

Amplifying worst-case hardness: The second type of the hardness amplification problem corresponds to starting with a function that is *worst-case hard*, i.e. hard to compute on *some* input, as opposed to a noticeable fraction of inputs in the previous case. We would like to construct a new function that is hard to compute on a noticeable fraction of inputs. In this case, we speak of *worst-case to average-case hardness amplification.* This is the setting that corresponds to relating average-case complexity questions to "standard" worst-case questions, such as: "does NP have polynomial-size circuits?" Worst-case to average-case hardness amplification has been accomplished for a number of "high" complexity classes such as $\mathsf{P}^{\#\mathsf{P}}$ and EXP (e.g. [Lip, BF, BFL, FL, CPS, STV, TV]); however, it remains a major open question for NP or even the polynomial-time hierarchy PH.

In this thesis we explain our current inability to exhibit a worst-case to average-case hardness amplification within NP by showing that a "large class" of techniques cannot yield such a hardness amplification. These techniques are known as "black-box" or "relativizing," and encompass most techniques actually used in hardness amplification, as we discuss below in Section 1.3.

1.2 Derandomization

Understanding the power of probabilistic computation is a central problem in theoretical computer science. There seems to be a strong belief among complexity researchers that randomness does not buy much in computation. Specifically, it is believed that it is always possible to *derandomize* probabilistic algorithms, that is, drastically reduce the amount of random bits they employ. Such a derandomized probabilistic algorithm can then be simulated deterministically with only a small overhead in resources by going over all the possibilities for its few random bits. Some researchers believe that derandomization can be pushed as far as obtaining that the class of problems solvable in probabilistic polynomial time equals the class of problems solvable in deterministic polynomial time, i.e., $\mathsf{BPP} = \mathsf{P}$.

A long line of exciting research (e.g., [NW, Imp1, IW1, ISW, STV, SU1, Uma2]) shows that derandomization is in fact possible given some seemingly unrelated complexity assumptions, for example the existence of explicit worst-case hard functions.

Such derandomization is obtained through the construction of fascinating objects called pseudorandom generators, which we now discuss.

Pseudorandom Generators (PRGs): Rigorously introduced by Blum and Micali [BM] and Yao [Yao1], a pseudorandom generator (PRG) is an efficient deterministic algorithm that stretches a randomly chosen seed into a much longer sequence, which nevertheless *fools* any efficient algorithm, in the sense that the algorithm cannot distinguish it from truly random. A PRG can be used to derandomize a probabilistic algorithm by replacing the random bits of that algorithm with the output of the PRG. Since no efficient algorithm can distinguish the output of the PRG from truly random, the derandomized algorithm will behave just like the original probabilistic algorithm.

Nisan and Wigderson, in a breakthrough result [NW], show how to construct a PRG starting from a very *average-case hard* function. Informally, their PRG is obtained by outputting many evaluations of the average-case hard function on "nearly disjoint" inputs. Because the *average-case* hardness assumption required for the Nisan-Wigderson generator seemed very strong, much research (e.g. [BFNW, Imp1, IW1, STV]) was devoted to constructing average-case hard functions from weaker hardness assumptions, i.e. to hardness amplification (see Section 1.1).

PRGs that fool restricted classes of algorithms: While we do not know of PRGs that fool *general* efficient algorithms, we do know of unconditional PRGs that fool *restricted* classes of algorithms. A notable example is the class of algorithms implemented by circuits of *constant depth*: a remarkable result by Nisan [Nis1] shows a PRG that fools any small constant-depth circuit. Perhaps surprisingly, PRGs that fool restricted classes of algorithms, such as Nisan's, have found many applications that go well beyond the derandomization of the restricted classes. For example, in this thesis we develop new applications of such PRGs to the field of hardness amplification: several of our results on hardness amplification mentioned in Section 1.1 rely on Nisan's PRG for constant-depth circuits.

In light of their usefulness, it is natural to try to construct PRGs that fool as general classes of algorithms as possible. However, it has proved to be very challenging to construct PRGs that fool even very restricted classes of algorithms. Nisan's remarkable PRG for constant-depth circuits is one of the very few unconditional PRGs we have. While this PRG has many applications, its use is limited by the fact that it only fools a relatively weak class of algorithms: it is known that constant-depth circuits cannot even compute "simple" functions like Parity or Majority.

In this thesis we construct a new PRG that fools a class of algorithms richer than Nisan's, namely constant-depth circuits that are equipped with few "arbitrary symmetric" gates, i.e. gates computing arbitrary symmetric functions, such as Parity or Majority. As a corollary, we obtain that probabilistic constant-depth circuits with few arbitrary symmetric gates can be simulated deterministically in subexponential time. This is the richest probabilistic circuit class known to admit a subexponential derandomization.

BPP vs. PH: For unrestricted models of probabilistic polynomial-time computation (BPP), we do not know of any deterministic subexponential-time simulation. However, we can prove several interesting and useful relationships between probabilistic polynomial time (BPP) and the polynomial-time hierarchy (PH), which is a generalization of NP allowing a constant number of alternating nondeterministic quantifiers, $\exists$ and $\forall$, rather than just $\exists$ as in NP. Even though it is not known whether $\mathsf{BPP} \subseteq \mathsf{NP}$, it is known that probabilistic polynomial time is in the second level of the polynomial-time hierarchy, i.e. $\mathsf{BPP} \subseteq \Sigma_2^{\mathsf{P}} = \mathsf{NP}^{\mathsf{NP}}$, which was proven in '83 by Sipser and Gács [Sip], and independently by Lautemann [Lau]. (One can think of Σ_2^{P} as what can be computed efficiently using two quantifiers $\exists \cdot \forall$.)

In this thesis we investigate the relationship between BPP and Σ_2^{P}. We observe that the Siper-Gács-Lautemann simulation of BPP in Σ_2^{P} incurs a quadratic slow-down in the running time. Specifically, their simulation shows that any probabilistic algorithm running in time t can be simulated in Σ_2^{P} in roughly quadratic time t^2. The question of whether this quadratic slow-down is necessary seems very natural, given that "$\mathsf{BPP} \subseteq \Sigma_2^{\mathsf{P}}$" is essentially the only non-trivial upper bound that we have on the power of probabilistic time. Moreover, this question arises when studying computational limitations of probabilistic machines, as we will see in later chapters. We prove that, for simulating BPP in Σ_2^{P}, a quadratic slow-down in the running time is in fact necessary for black-box techniques (discussed below in Section 1.3), such as those used in the simulations by Siper, Gács, and Lautemann, as well as any other known simulation. However, we show that the quadratic slow-down disappears at the third level of the polynomial-time hierarchy, where a *quasi-linear* time simulation is possible.

1.3 Black-box techniques

Part of this thesis studies relationships between complexity assumptions. That is, we study whether a certain complexity assumption implies another complexity assumption, even though we do not know whether either one of these assumptions actually holds. For example, the hardness amplification problem, introduced in Section 1.1, is the problem of establishing that the existence of a "somewhat hard" function (assumption A) implies the existence of a "very hard on average" function (assumption B).

In some cases, researchers have been able to prove such an implication. For example, it is shown in [BFNW] that if EXP contains a worst-case hard function (assumption A) then EXP contains a very average-case hard function (assumption

B).

In other important cases, researchers have been unable to prove such an implication. For example, they have been unable to show a worst-case vs. average-case connection for NP, i.e. to show that the existence of a worst-case hard function in NP (assumption A) implies the existence of a function in NP that is noticeably hard on average (assumption B). In light of this, it is natural to ask whether such an implication cannot be proved. One must be careful in formalizing such questions. If it is the case, as many researchers believe, that NP does contain a function that is noticeably hard on average, the implication "assumption A implies assumption B" is true in a trivial logical sense. Paraphrasing [RTV], *the question is whether the techniques that we typically use to prove implications of worst-case hard functions in* NP *have some inherent limitation that prevents us from deriving the existence of a function in* NP *that is noticeably hard on average.*

To make formal sense of the above question one needs a model. In this thesis we focus on the *black-box model*, essentially introduced in the work of Baker et al. [BGS], which is also known as the relativizing model (cf. [IR, RTV]). Roughly speaking, this model captures proof techniques that are information-theoretic, in the sense that they do not exploit computational properties of the objects involved but only use them as "black-box" in an input/output fashion. Continuing with our example, a black-box hardness amplification would transform *any* given function f into a harder function, regardless of whether f belongs to NP or not, provided that we allow input/output access to f.

There exist relatively few non-black-box techniques. In fact, there seem to be some areas of complexity theory, such as the study of the relationship between BPP and PH (see Section 1.2), where all known results are proven via black-box techniques. Therefore, in certain contexts, a negative result in the black-box model about an implication between two complexity assumptions is interpreted as meaning that proving such an implication, if it is true at all, would require a significant departure from current techniques.

An additional motivation for studying problems in the black-box model is that they often correspond to interesting problems in other fields. For example, black-box hardness amplification can be seen as a task in *coding theory*, and this connection has fostered a fruitful exchange of ideas between complexity theory and coding theory (cf. [Tre3]). We will exploit this connection several times in later chapters.

1.4 Results and structure of this thesis

Chapter 2 – Hardness Amplification within NP. We study hardness amplification from noticeable average-case hardness. We prove that if NP contains a balanced function that is hard to compute on a noticeable fraction of the inputs then NP contains a function that is hard to compute on a fraction of inputs exponentially close to one-half. (A function is balanced if it evaluates to 1 on half of the inputs.) This improves on the previous results of O'Donnell [O'D], who, under the same assumption, obtained a function that is hard to compute on a fraction of inputs polynomially close to one-half (as opposed to exponentially in our result).

We also prove an impossibility result demonstrating that the assumption that the original function is balanced is necessary for black-box techniques (such as ours).

Chapter 3 – Worst-Case Hardness Amplification. We study hardness amplification from worst-case hardness. We show that there is no black-box worst-case to average-case hardness amplification within NP, or even within the polynomial-time hierarchy PH. We exhibit two versions of this result, addressing different notions of "black-box." Our results complement the previous ones (e.g., [BFNW, STV]), which show that such worst-case hardness amplifications are possible in complexity classes above PH, such as EXP.

Chapter 4 – Pseudorandom Bits for Constant-Depth Circuits with Few Arbitrary Symmetric Gates. We construct a new PRG that fools any small constant-depth circuit that is equipped with few special gates computing arbitrary symmetric functions, such as Parity or Majority. This improves on a generator by Luby, Velickovic, and Wigderson [LVW] that only fools circuits of depth 2 with one arbitrary symmetric gate at the top. Our generator also fools a richer class of circuits than Nisan's generator for constant-depth circuits [Nis1]. In particular, we conclude that every function computable by small *probabilistic* constant-depth circuits with few arbitrary symmetric gates can be simulated *deterministically* in subexponential time. This is the richest probabilistic circuit class known to admit a subexponential derandomization.

Chapter 5 – Probabilistic Time versus Alternating Time. We study the relationship between probabilistic polynomial time, BPP, and the polynomial-time hierarchy, PH. We prove that, for simulating BPP in Σ_2^{P}, a quadratic slow-down in the running time is necessary for black-box techniques. This matches the running time of the known Σ_2^{P} simulations [Sip, Lau], which are black-box. We also show that the quadratic slow-down disappears at the third level of the polynomial-time hierarchy, where a *quasi-linear* time simulation is possible. Using this latter time-efficient simulation we prove new lower bounds for computing certain explicit functions on probabilistic Turing machines.

Chapter 6 – The Complexity of Decoding. We remark that none of the known hardness amplification results can be applied to the computational models for which we actually can establish the existence of hard functions (i.e. prove lower bounds).

We conjecture that the most restricted circuit class to which black-box hardness amplification can be applied is the class of constant-depth circuits with Majority gates, i.e., TC^0 circuits. This is problematic because no explicit function is known to require super-polynomial size TC^0 circuits. We prove special cases of our conjecture for settings of parameters that are achieved by known hardness amplification results.

Chapter 2

Hardness Amplification within NP

2.1 Introduction

In this chapter we study hardness amplification within NP, starting from functions that are at least $1/n^{O(1)}$-hard. Let us recall (from Section 1.1) the definition of "hard."

Definition 2.1. *For $\delta \in [0, 1/2]$, a function $f : \{0,1\}^n \to \{0,1\}$ is* δ-hard for size s *if every circuit of size s fails to compute f on at least a δ fraction of inputs.*

Recall that the maximum value of the hardness parameter δ is $1/2$ because f is boolean (and so can trivially be computed with error probability at most $1/2$). The *hardness amplification problem* is to convert a function f that is δ-hard for size s into a function f' that is $(1/2 - \epsilon)$-hard for size polynomially related to s. In this chapter, $\delta = 1/\operatorname{poly}(n)$ and the aim is to make $\epsilon = \epsilon(n)$ vanish as quickly as possible.

On the definition of hardness: The above notion of hardness (Definition 2.1) is fairly standard (e.g. in the literature on hardness amplification starting from [NW]), but we remark that it differs from Levin's original notion of average-case complexity [Lev] (cf. Impagliazzo's survey [Imp2]). The main difference is that Levin's formulation corresponds to algorithms that always either give the correct answer or say "do not know," whereas we consider even "heuristic" algorithms that can make arbitrary errors.

The standard approach to hardness amplification employs Yao's Xor Lemma [Yao1] (see [GNW]): Given a "mildly" hard-on-average function $f : \{0,1\}^n \to \{0,1\}$, we define $f' : \{0,1\}^{n \cdot k} \to \{0,1\}$ by

$$f'(x_1, \ldots, x_k) := f(x_1) \oplus f(x_2) \oplus \cdots \oplus f(x_k).$$

The Xor Lemma says that the hardness of f' approaches $1/2$ exponentially fast with k. More precisely:

Theorem 2.2 (Yao's Xor Lemma). *If f is δ-hard for size $s(n) \geq n^{\omega(1)}$ and $k \leq \mathrm{poly}(n)$, then f' is $(1/2 - 1/2^{\Omega(\delta k)} - 1/s')$-hard for size $s'(n \cdot k) = s(n)^{\Omega(1)}$.*

In particular, taking $k = \Theta(n/\delta)$, the amplified hardness is dominated by the $1/s'$ term. That is, we can amplify to hardness $(1/2 - \epsilon)$, where ϵ is polynomially related to the (reciprocal of the) circuit size for which f was hard. (Note, however, that we should measure $\epsilon = \epsilon(n')$ as a function of the new input length $n' = n \cdot k$, so when $k = n$, the hardness is actually $1/2 - 1/s(\sqrt{n'})^{\Omega(1)}$.)

However, if we are interested in hardness amplification within NP (i.e. $f, f' \in$ NP), we cannot use the Xor lemma; it does not ensure that f' is in NP when f is in NP. Hardness amplification within NP was first addressed in a recent paper of O'Donnell [O'D], which is the starting point for our work.

2.1.1 O'Donnell's hardness amplification

To ensure that the new function f' is in NP when f is in NP, O'Donnell [O'D] was led to study constructions of the form

$$f'(x_1, \ldots, x_k) := C(f(x_1), f(x_2), \ldots, f(x_k)), \tag{2.1}$$

where C is an efficiently computable *monotone* function. The monotonicity of C ensures that f' is in NP when f is in NP. But we are left with the task of choosing such a function C and proving that it indeed amplifies hardness.

Remarkably, O'Donnell was able to precisely characterize the amplification properties of Construction 2.1 in terms of a combinatorial property of the combining function C, called its *expected bias*. (The actual definition is not needed for this discussion, but can be found in Section 2.3.) By finding a monotone combining function in which this expected bias is small, he obtained the first positive result on hardness amplification in NP:

Theorem 2.3 (O'Donnell's Theorem [O'D]). *If* NP *contains a balanced function that is $1/\mathrm{poly}(n)$-hard for polynomial-size circuits, then* NP *contains a function that is $(1/2 - 1/n^{1/2-\alpha})$-hard for polynomial-size circuits (where α is an arbitrarily small positive constant).*

However, the amplification provided by O'Donnell's theorem is not as strong as what the Xor Lemma gives. It is limited to $1/2 - 1/\sqrt{n}$, regardless of the circuit size s for which the original function is hard, even if s is exponentially large. The Xor Lemma, on the other hand, amplifies to $1/2 - 1/s^{\Omega(1)}$. O'Donnell showed that this difference is inherent — no construction of the form (2.1) with a monotone combining function C can always amplify hardness to better than $1/2 - 1/n$. (The gap between O'Donnell's positive result of $1/2 - 1/\sqrt{n}$ and his negative result of $1/2 - 1/n$ is not significant for what follows, and it will be subsumed by our improvements.)

2.1.2 Our result

In this chapter, we amplify hardness within NP beyond the $1/2 - 1/n$ barrier:

Theorem 2.4 (Main). *If* NP *contains a balanced function that is* $1/\operatorname{poly}(n)$*-hard for circuits of size* $s(n)$*, then* NP *contains a function that is* $(1/2 - 1/s'(n))$*-hard for circuits of size* $s'(n)$ *for some function*[1] $s'(n) = s(\sqrt{n})^{\Omega(1)}$. *In particular,*

1. *If* $s(n) = n^{\omega(1)}$, *we amplify to hardness* $1/2 - 1/n^{\omega(1)}$.
2. *If* $s(n) = 2^{n^{\Omega(1)}}$, *we amplify to hardness* $1/2 - 1/2^{n^{\Omega(1)}}$.
3. *If* $s(n) = 2^{\Omega(n)}$, *we amplify to hardness* $1/2 - 1/2^{\Omega(\sqrt{n})}$.

Items 1–3 match the parameters of the Yao's Xor Lemma. However, subsequent "derandomizations" of the Xor Lemma [Imp1, IW1] actually amplify up to $1/2 - 1/2^{\Omega(n)}$ rather than just $1/2 - 1/2^{\Omega(\sqrt{n})}$ in the case $s(n) = 2^{\Omega(n)}$. This gap is *not* inherent in our approach and, as mentioned below, would be eliminated given a corresponding improvement in one of the tools we employ.

Of course, our construction cannot be of the form in Construction (2.1). Below we describe our two main points of departure.

2.1.3 Techniques

To explain how we bypass it, we first look more closely at the source of the $1/2 - 1/n$ barrier. The actual barrier is $1/2 - 1/k$, where k is the input length of the monotone combining function C. Since in Construction (2.1), f' has input length $n' = n \cdot k \geq k$, it follows that we cannot amplify beyond $1/2 - 1/n'$.

Derandomization. Given the above, our first idea is to break the link between the input length of f' and the input length of the combining function C. We do this by *derandomizing* O'Donnell's construction. That is, the inputs $x_1, \ldots, x_k$ are no longer taken independently (as in Construction (2.1)), but are generated pseudorandomly from a short seed of length $n' \ll k$, which becomes the actual input to f'. Our method for generating the x_i's is based on combinatorial designs (as in the Nisan–Wigderson generator [NW]) and Nisan's pseudorandom generator for space-bounded computation [Nis2]; it reduces the input length of f' from $n \cdot k$ to $n' = O(n^2 + \log^2 k)$. We stress that this derandomization is unconditional, i.e. requires no additional complexity assumption. We also remark that it is the quadratic seed length of Nisan's generator that limits our amplification to $1/2 - 1/2^{\Omega(\sqrt{n})}$ rather than $1/2 - 1/2^{\Omega(n)}$ in

[1] In the rest of the chapter, we more compactly write "for circuits of size $s'(n) = s(\sqrt{n})^{\Omega(1)}$," where the $\Omega(1)$ is to be interpreted as a fixed constant (possibly depending on previously quantified constants).

Part 3 of our Main Theorem, and thus any improvement in Nisan's generator would yield a corresponding improvement in our result.

Similar derandomizations have previously been achieved for Yao's Xor Lemma by Impagliazzo [Imp1] and Impagliazzo and Wigderson [IW1]. The analysis of such derandomizations is typically tailored to a particular proof, and indeed both [Imp1, IW1] gave new proofs of the Xor Lemma for that purpose. In our case, we do not know how to derandomize O'Donnell's original proof, but instead manage to derandomize a different proof due to Trevisan [Tre2].

Our derandomization allows for k to be larger than the input length of f', and hence we can go beyond the $1/2-1/n'$ barrier. Indeed, by taking k to be a sufficiently large polynomial, we amplify to $1/2-1/(n')^c$ for any constant c.

Using nondeterminism. To amplify further, it is tempting to take k superpolynomial in the input length of f'. But then we run into a different problem: how do we ensure that f' is in NP? The natural algorithm for f' requires running the algorithm for f on k inputs.

To overcome this difficulty, we observe that we need only give an efficient *nondeterministic* algorithm for f'. Each nondeterministic path may involve only polynomially many evaluations of f while the global outcome $f'(x)$ depends on exponentially many evaluations. To implement this idea, we exploit the specific structure of the combining function C. Namely, we (like O'Donnell) use the Tribes function of Ben-Or and Linial [BL], which is a monotone DNF with clauses of size $O(\log k)$. Thus, the nondeterministic algorithm for f' can simply guess a satisfied clause and (nondeterministically) evaluate f on the $O(\log k)$ corresponding inputs.

2.1.4 Other results

We also present the following complementary negative results for black-box hardness amplifications:

- We show that the assumption that the original hard function is balanced is necessary, in the sense that no monotone "black-box" hardness amplification can amplify unbalanced functions of unknown bias (or even improve their bias).[2]

- We show that our use of nondeterminism is necessary, in the sense that any "black-box" hardness amplification in which each evaluation of f' is a monotone function of at most k evaluations of f can amplify hardness to at most $1/2-1/k$.

Informally, a "black-box" hardness amplification is one in which the construction of the amplified function f' from f only utilizes f as an oracle and is well-defined

[2] We note that there do exist balanced NP-complete problems, as observed by Barak [For], but this has no direct implication for us because we are studying the average-case complexity of NP.

for any function f (regardless of whether or not it is in NP). Moreover, the correctness of the construction is proved by a generic reduction that converts any *oracle* A (regardless of its circuit size) that computes f' well on average (e.g., with probability $1/2+\epsilon$ over random choice of input) into one that computes f much better on average (e.g., with probability $1-o(1)$ over random input). (A formal definition is given in Section 2.7.1.) We note that most results on hardness amplification against circuits, including ours, are black-box (though there have been some recent results using non-black-box techniques in hardness amplification against *uniform* algorithms; see [IW2, TV]).

Our framework also gives a new proof of the hardness amplification by Impagliazzo and Wigderson [IW1]. Our proof is simpler and in particular it does not employ the Goldreich–Levin [GL] step.

2.1.5 Organization

The rest of this chapter is organized as follows. In Section 2.2, we discuss some preliminaries. In Section 2.3, we review existing results on hardness amplification in NP. In Section 2.4, we present our main results and new techniques. In Section 2.5 we treat the details of the proof of our main theorem. In Section 2.6 we show how we could amplify to $1/2-1/2^{\Omega(n)}$ given an improvement in the pseudorandom generator we use, and we also give a new proof of the hardness amplification by Impagliazzo and Wigderson [IW1]. In Section 2.7 we discuss some limitations of monotone hardness amplification; in particular we show a sense in which the hypothesis that the starting function be balanced is necessary, and also that the use of nondeterminism is necessary.

2.2 Preliminaries

We denote the uniform distribution on $\{0,1\}^n$ by U_n. If U_n occurs more than once in the same expression, it is understood that these all represent the same random variable; for example, $U_n \cdot f(U_n)$ denotes the random variable obtained by choosing X uniformly at random in $\{0,1\}^n$ and outputting $X \cdot f(X)$ (where $\cdot$ means concatenation).

Definition 2.5. *Let X and Y be two random variables taking values over the same set S. Then the* statistical difference *between X and Y, is*

$$\Delta(X,Y) := \max_{T\subseteq S}\Big|\Pr[X\in T]-\Pr[Y\in T]\Big|.$$

We view probabilistic functions as functions of two inputs, e.g. $h(x;r)$, the first being the input to the function and the second being the randomness. (Deterministic functions may be thought of as probabilistic functions that ignore the randomness.)

For notational convenience, we will often omit the second input to a probabilistic function, e.g. writing $h(x)$ instead of $h(x;r)$, in which case we view $h(x)$ as the random variable $h(x;U_{|r|})$.

Definition 2.6. *The* bias *of a* 0-1 *random variable* X *is*

$$Bias\,[X] := \Big|\Pr[X=0]-\Pr[X=1]\Big| = 2\cdot\Delta(X,U_1).$$

Analogously, the bias *of a probabilistic function* $f:\{0,1\}^n\to\{0,1\}$ *is*

$$Bias\,[f] := \Big|\Pr[f(U_n)=0]-\Pr[f(U_n)=1]\Big|,$$

where the probabilities are taken over both the input chosen according to U_n *and the coin tosses of* f. *We say that* f *is* balanced *when* $Bias\,[f]=0$.

Note that the bias of a random variable is a quantity between 0 and 1.

We say that the random variables X and Y are *ϵ-indistinguishable for size s* if for every circuit C of size s,

$$\Big|\Pr_X[C(X)=1]-\Pr_Y[C(Y)=1]\Big| \le \epsilon.$$

We will routinely use the following connection between hardness and indistinguishability.

Lemma 2.7 ([Yao1])**.** *Let* $h:\{0,1\}^n\to\{0,1\}$ *be any probabilistic function. If the distributions* $U_n\cdot h(U_n)$ *and* $U_n\cdot U_1$ *are* ϵ*-indistinguishable for size* s *then* h *is* $(1/2-\epsilon)$*-hard for size* $s-O(1)$. *Conversely, if* h *is* $(1/2-\epsilon)$*-hard for size* s *then the distributions* $U_n\cdot h(U_n)$ *and* $U_n\cdot U_1$ *are* ϵ*-indistinguishable for size* $s-O(1)$.

Finally, whenever we amplify the hardness of a function $f:\{0,1\}^n\to\{0,1\}$ that is hard for circuits of size $s(n)$, we assume that $s(n)$ is well-behaved in the sense that it is computable in time poly(n) and $s(cn)=s(n)^{O(1)}$, for all constants $c>0$. Most natural functions smaller than 2^n, such as $n^k, 2^{\log^k n}, 2^{n^\epsilon}, 2^{\epsilon n}$, are well-behaved in this sense.

2.3 Overview of previous hardness amplification in NP

In this section we review the essential components of existing results on hardness amplification in NP. We then discuss the limitations of these techniques. By the end of this section, we will have sketched the main result of O'Donnell [O'D], following the approach of Trevisan [Tre2]. We outline this result in a way that will facilitate the presentation of our results in subsequent sections.

Let $f : \{0,1\}^n \to \{0,1\}$ be an average-case hard function, and let $C : \{0,1\}^k \to \{0,1\}$ be any function. In [O'D], O'Donnell studies the hardness of functions of the form

$$C \circ f^{\otimes k} : (\{0,1\}^n)^k \to \{0,1\}$$

where $f^{\otimes k}(x_1,\ldots,x_k) := (f(x_1),\ldots,f(x_k))$, and $\circ$ denotes composition. That is,

$$(C \circ f^{\otimes k})(x_1,\ldots,x_k) := C(f(x_1),\ldots,f(x_k)).$$

In order to ensure that $C \circ f^{\otimes k} \in \mathsf{NP}$ whenever $f \in \mathsf{NP}$, O'Donnell chooses C to be a polynomial-time computable *monotone* function. (Indeed, it is not hard to see that a monotone combination of NP functions is itself in NP.)

O'Donnell characterizes the hardness of $C \circ f^{\otimes k}$ in terms of a combinatorial property of the combining function C, called its *expected bias* (which we define later). We will now review the key steps in establishing this characterization and O'Donnell's final amplification theorem.

Step 1: Impagliazzo's hardcore sets. An important tool for establishing this connection is the hardcore set lemma of Impagliazzo [Imp1], which allows us to pass from computational hardness to information-theoretic hardness.

Definition 2.8. *We say that a (probabilistic) function $g : \{0,1\}^n \to \{0,1\}$ is δ-*random *if g is balanced and there exists a subset $H \subseteq \{0,1\}^n$ with $|H| = 2\delta 2^n$ such that $g(x) = U_1$ (i.e. a coin flip) for $x \in H$ and $g(x)$ is deterministic for $x \notin H$.*

Thus, a δ-random function has a set of relative size 2δ on which it is information-theoretically unpredictable. Note that in the above definition we require g to be balanced. This will be convenient when dealing with functions f that are balanced.

The following version of Impagliazzo hardcore set lemma says that any balanced δ-hard function $f : \{0,1\}^n \to \{0,1\}$ has a hardcore set $H \subseteq \{0,1\}^n$ of density $\approx 2\delta$ such that f is very hard-on-average on H. Thus, f looks like a δ-random function to small circuits (cf. Lemma 2.7). (Following subsequent works, our formulation of Impagliazzo's lemma differs from the original one in several respects.)

Lemma 2.9 ([Imp1, KS, STV, O'D])**.** *For any function $f : \{0,1\}^n \to \{0,1\}$ that is balanced and δ-hard for size s, there exists a δ'-random function $g : \{0,1\}^n \to \{0,1\}$ such that $X \cdot f(X)$ and $X \cdot g(X)$ are ϵ-indistinguishable for size $\Omega(s\epsilon^2/\log(1/\delta))$, with $\delta/2 \le \delta' \le \delta$, where $X \equiv U_n$.*

In particular, by a standard hybrid argument (see, e.g., [Gol3]),

$$X_1 \cdots X_k \cdot f(X_1) \cdots f(X_k) \text{ and } X_1 \cdots X_k \cdot g(X_1) \cdots g(X_k)$$

are $k\epsilon$-indistinguishable for size $\Omega(s\epsilon^2/\log(1/\delta))$, where the X_i's are uniform and independent.

Step 2: expected bias. By the above, proving the computational hardness of $C \circ f^{\otimes k}$ reduces to calculating the information-theoretic hardness of $C \circ g^{\otimes k}$ for some δ'-random g. It turns out that information-theoretic hardness can be characterized by the following quantity.

Definition 2.10. *Let $h : \{0,1\}^n \to \{0,1\}$ be any probabilistic function. We define the* expected bias *of h by*

$$\mathrm{ExpBias}\,[h] := \mathop{\mathrm{E}}_{x \leftarrow U_n}\big[\mathit{Bias}\,[h(x)]\,\big],$$

where $\mathit{Bias}\,[h(x)]$ is taken over the coin tosses of h.

It turns out that for any function $C : \{0,1\}^k \to \{0,1\}$ and any δ-random g, the quantity $\mathrm{ExpBias}\left[C \circ g^{\otimes k}\right]$ does not depend on the particular choice of the δ-random function g; indeed, it turns out to equal the quantity that O'Donnell [O'D] calls the "expected bias of C with respect to noise 2δ" and denotes by $\mathrm{ExpBias}_{2\delta}(C)$ in [O'D]. However, the more general notation we use will be useful in presenting our improvements.

The next lemma shows that information-theoretic hardness is equivalent to expected bias.

Lemma 2.11. *For any probabilistic function $h : \{0,1\}^n \to \{0,1\}$,*

$$\Delta(U_n \cdot h(U_n), U_n \cdot U_1) = \frac{1}{2}\,\mathrm{ExpBias}\,[h]\,.$$

Proof.

$$\begin{aligned}\Delta(U_n \cdot h(U_n), U_n \cdot U_1) &= \mathop{\mathrm{E}}_{x \leftarrow U_n}\left[\Delta(h(x), U_1)\right]\\ &= \mathop{\mathrm{E}}_{x \leftarrow U_n}\left[\mathit{Bias}\,[h(x)]\,/2\right] = \mathrm{ExpBias}\,[h]\,/2.\end{aligned}$$

□

In particular, for any circuit C (regardless of its size) we have

$$\Big|\Pr[C(U_n \cdot h(U_n)) = 1] - \Pr[C(U_n \cdot U_1) = 1]\Big| \le \mathrm{ExpBias}\,[h]\,/2,$$

and thus by Lemma 2.7, h is $(1/2 - \mathrm{ExpBias}\,[h]\,/2)$-hard for circuits of any size.

Now we characterize the hardness of $C \circ f^{\otimes k}$ in terms of expected bias. Specifically, by taking, say, $\epsilon = 1/s^{1/3}$ in Lemma 2.9 and using Lemmas 2.7 and 2.11, one can prove the following (we defer the details until the proof of the more general Lemma 2.21).

Lemma 2.12 ([O'D]). *Let $f : \{0,1\}^n \to \{0,1\}$ be balanced and δ-hard for size s, and let $C : \{0,1\}^k \to \{0,1\}$ be any function. Then there exists a δ'-random function $g : \{0,1\}^n \to \{0,1\}$, with $\delta/2 \leq \delta' \leq \delta$, such that $C \circ f^{\otimes k} : (\{0,1\}^n)^k \to \{0,1\}$ has hardness*

$$\frac{1}{2} - \frac{\mathrm{ExpBias}\left[C \circ g^{\otimes k}\right]}{2} - \frac{k}{s^{1/3}}$$

for circuits of size $\Omega\left(s^{1/3}/\log(1/\delta)\right) - \mathrm{size}(C)$, where $\mathrm{size}(C)$ denotes the size of a smallest circuit computing C.

What makes this lemma so useful is that, as noted above, the quantity $\mathrm{ExpBias}\left[C \circ g^{\otimes k}\right]$ is independent of the choice of the δ-random function g (using the fact that g is balanced, by definition of δ-random); hence the hardness of $C \circ f^{\otimes k}$ depends only on the combining function C and the hardness parameter δ. Thus, understanding the hardness of $C \circ f^{\otimes k}$ is reduced to analyzing a combinatorial property of the combining function C.

Step 3: noise stability. Unfortunately, it is often difficult to analyze the expected bias directly. However, the expected bias is closely related to the *noise stability*, a quantity that is more amenable to analysis and well-studied (see, e.g., [O'D], [MO]). The noise stability of a function is (up to normalization) the probability that the value of the function is the same on two correlated inputs x and $x + \eta$, where x is a random input and η a random vector of noise.

Definition 2.13. *The noise stability of C with respect to noise δ, denoted $\mathrm{NoiseStab}_\delta[C]$, is defined by*

$$\mathrm{NoiseStab}_\delta[C] := 2 \cdot \Pr_{x,\eta}[C(x) = C(x \oplus \eta)] - 1,$$

where x is random, η is a vector whose bits are independently one with probability δ and $\oplus$ denotes bitwise Xor.

The following lemma from [O'D] bounds the expected bias of $C \circ g^{\otimes k}$ (and hence the hardness in Lemma 2.12) in terms of the noise stability of C.

Lemma 2.14. *Let $g : \{0,1\}^n \to \{0,1\}$ be δ-random. Then*

$$\mathrm{ExpBias}\left[C \circ g^{\otimes k}\right] \leq \sqrt{\mathrm{NoiseStab}_\delta[C]}.$$

Combining this with Lemma 2.12, we find that the hardness of $C \circ f^{\otimes k}$ is at least (roughly) $1/2 - \sqrt{\mathrm{NoiseStab}_\delta[C]}/2$. The next step is to exhibit a combining function C with a small noise stability (to ensure that the hardness of $C \circ f^{\otimes k}$ is as close to $1/2$ as possible). The following is shown in [O'D].

Lemma 2.15 ([O'D])**.** *For all $\delta > 0$, there exists a $k = \mathrm{poly}(1/\delta)$ and a polynomial-time computable monotone function $C : \{0,1\}^k \to \{0,1\}$ with $\mathrm{NoiseStab}_\delta[C] \leq 1/k^{\Omega(1)}$.*

Finally, by combining Lemmas 2.12, 2.14 and 2.15, we obtain the following weaker version of O'Donnell's hardness amplification within NP. (While in the introduction we mentioned a stronger version of O'Donnell's result, that amplifies up to hardness $1/2 - 1/m^{1/2-\alpha}$ for every constant $\alpha > 0$, the following version will suffice as a starting point for our work. The loss in the amplification in this version comes from the fact that we did not specify the constants in Lemma 2.15.)

Theorem 2.16 ([O'D])**.** *If there is a balanced function $f : \{0,1\}^n \to \{0,1\}$ in* NP *that is $1/\mathrm{poly}(n)$-hard for size $s(n)$, then there is a function $f' : \{0,1\}^m \to \{0,1\}$ in* NP *that is $(1/2 - 1/m^{\Omega(1)})$-hard for size $s(m^{\Omega(1)})^{\Omega(1)}$.*

Limitations of direct product constructions. O'Donnell also showed that Theorem 2.16 is essentially the best result that one can obtain using the techniques that we have described thus far. He showed that for all monotone combining functions C there is a δ-hard function f such that the hardness of $C \circ f^{\otimes k}$ is not much better than $1/2 - \mathrm{NoiseStab}_\delta[C]/2$ (assuming that C is easily computable). This is problematic because the noise stability of monotone functions cannot become too small. Specifically, by combining a result from [KKL] with a Fourier characterization of noise stability, O'Donnell [O'D] proves the following theorem.

Theorem 2.17 ([KKL, O'D])**.** *For every monotone function $C : \{0,1\}^k \to \{0,1\}$ and every $\delta > 0$,*

$$\mathrm{NoiseStab}_\delta[C] \geq (1 - 2\delta) \cdot \Omega\left(\frac{\log^2 k}{k}\right).$$

Therefore, for any monotone $C : \{0,1\}^k \to \{0,1\}$ there is a δ-hard f such that $C \circ f^{\otimes k}$ does *not* have hardness $1/2 - \mathrm{NoiseStab}_\delta[C]/2 \leq 1/2 - \Omega(1/k)$. Since $C \circ f^{\otimes k}$ takes inputs of length $m = n \cdot k \geq k$, this implies that we must employ a new technique to amplify beyond hardness $1/2 - \Omega(1/m)$.

2.4 Main Theorem and overview

In this chapter, we obtain the following improvement upon Theorem 2.16.

Theorem 2.18 (Main Theorem, restated)**.** *If there is a balanced function $f : \{0,1\}^n \to \{0,1\}$ in* NP *that is $1/\mathrm{poly}(n)$-hard for size $s(n)$, then there is a function $f' : \{0,1\}^m \to \{0,1\}$ in* NP *that is $(1/2 - 1/s(\sqrt{m})^{\Omega(1)})$-hard for size $s(\sqrt{m})^{\Omega(1)}$.*[3]

[3]A comment is in order about the input lengths for which f' is hard. As it turns out, the hardness

We also show that the assumption that we start with a *balanced* function f is essential for a large class of hardness amplifications. Specifically, we show (Section 2.7.1) that no monotone black-box hardness amplification can amplify the hardness of functions whose bias is unknown. Most hardness amplifications, including the one in this chapter, are black-box.

We now elaborate on the two main techniques that allow us to prove Theorem 2.18. As explained in the introduction, these two techniques are derandomization and nondeterminism.

2.4.1 Derandomization

As in the previous section, let $f : \{0,1\}^n \to \{0,1\}$ be our hard function and let $C : \{0,1\}^k \to \{0,1\}$ be a (monotone) combining function.

We will derandomize O'Donnell's construction using an appropriately "pseudorandom" generator.

Definition 2.19. *A* generator *is a function $G : \{0,1\}^l \to (\{0,1\}^n)^k$. We call l the* seed length *of G, and we often write $G(\sigma) = X_1 \cdots X_k$, with each $X_i \in \{0,1\}^n$. G is* explicitly computable *if given σ and $1 \le i \le k$, we can compute X_i in time* $\mathrm{poly}(l, \log k)$*, where $G(\sigma) = X_1 \cdots X_k$.*

Instead of using the function $C \circ f^{\otimes k} : (\{0,1\}^n)^k \to \{0,1\}$, we take a generator $G : \{0,1\}^l \to (\{0,1\}^n)^k$ (where $l \ll nk$) and use $(C \circ f^{\otimes k}) \circ G : \{0,1\}^l \to \{0,1\}$, i.e.,

$$(C \circ f^{\otimes k}) \circ G(\sigma) = C\big(f(X_1), \ldots, f(X_k)\big),$$

where $(X_1, \ldots, X_k) \in (\{0,1\}^n)^k$ is the output of $G(\sigma)$. This reduces the input length of the function to l. Therefore, if G is a "good" pseudorandom generator we would expect $(C \circ f^{\otimes k}) \circ G$ to be harder (with respect to its input length) than $C \circ f^{\otimes k}$. We will show that this is indeed the case, provided the generator G satisfies the following requirements:

1. **G is indistinguishability-preserving:** Analogously to Lemma 2.12, the generator G should be such that the computational hardness of $(C \circ f^{\otimes k}) \circ G$ is at least the information-theoretic hardness of $(C \circ g^{\otimes k}) \circ G$ for some δ-random function g – that is, at least $1/2 - \mathrm{ExpBias}\big[(C \circ g^{\otimes k}) \circ G\big]$. We will see that this can be achieved provided that G is *indistinguishability-preserving*; that is (analogously to the last part of Lemma 2.9),

$$\sigma \cdot f(X_1) \cdots f(X_k) \text{ and } \sigma \cdot g(X_1) \cdots g(X_k)$$

of f' on inputs of length m is related to the hardness of the original function f on inputs of length $\Theta(\sqrt{m})$. Thus if f is hard for all sufficiently large input lengths, then so is f'. Alternatively, if f is only hard infinitely often, then we may still conclude that f' is hard infinitely often.

should be indistinguishable, for some δ-random g, when $\sigma \stackrel{\mathrm{R}}{\leftarrow} \{0,1\}^l$ and $(X_1, \ldots, X_k) \in (\{0,1\}^n)^k$ is the output of G on input σ.

2. **G fools the expected bias:** G should be such that for any δ-random g, $\mathrm{ExpBias}\left[(C \circ g^{\otimes k}) \circ G\right]$ is approximately $\mathrm{ExpBias}\left[C \circ g^{\otimes k}\right]$, and thus, by Lemma 2.14:
$$\mathrm{ExpBias}\left[(C \circ g^{\otimes k}) \circ G\right] \leq \sqrt{\mathrm{NoiseStab}_\delta[C] + \epsilon}, \tag{2.2}$$
for a suitably small ϵ. Actually, we will not show that G fools the expected bias directly but instead will work with a related quantity (the expected collision probability), which will still suffice to show Inequality (2.2).

Informally, the effect of the two above requirements on the generator G is that the hardness of $(C \circ f^{\otimes k}) \circ G$ is roughly the hardness of $C \circ f^{\otimes k}$, while the input length is dramatically reduced from nk to l (the seed length of G). More precisely, as illustrated in Figure 1, the first requirement allows us to relate the hardness of $(C \circ f^{\otimes k}) \circ G$ to the information-theoretic hardness of $(C \circ g^{\otimes k}) \circ G$ (where g is a δ-random function); the second allows us to relate this information-theoretic hardness to the noise stability of the combining function C. In particular, if we employ the combining function from Lemma 2.15, we obtain hardness $1/2 - 1/k^{\Omega(1)}$. Thus, by choosing $k \gg l$, we bypass the barrier discussed at the end of the previous section.

Now we briefly describe how the above requirements on G are met. The first requirement is achieved through a generator that outputs *combinatorial designs*. This construction is essentially from Nisan and Wigderson [Nis1, NW] and has been used in many places, e.g. [IW1, STV].

The second requirement is achieved as follows. We show that if G is pseudorandom against space-bounded algorithms and the combining function C is computable in small space (with one-way access to its input), then Inequality (2.2) holds. We then use Nisan's *unconditional* pseudorandom generator against space-bounded algorithms [Nis2], and show that combining functions with low noise stability can in fact be computed in small space.[4] Note that we only use the pseudorandomness of the generator G to relate the expected bias with respect to G to a combinatorial property of the combining function C. In particular, it is *not* used to fool the circuits trying to compute the hard function. This is what allows us to use an unconditional generator against a relatively weak model of computation.

Our final generator, Γ, is the generator obtained by Xor'ing a generator that is indistinguishability-preserving and a generator that fools the expected bias, yielding a generator that has both properties. The approach of Xor'ing two generators in this way appeared in [IW1], and was subsequently used in [STV].

[4] The same approach also works using the unconditional pseudorandom generator against constant-depth circuits of [Nis1] and showing that the combining function is computable by a small constant-depth circuit; however, the space generator gives us slightly better parameters.

Table 2.1: Hardness amplification within NP.

Functions : $\{0,1\}^n \to \{0,1\}$		
Amplification up to	**Technique**	**Reference**
$1/2 - 1/\sqrt{n}$	Direct Product	[O'D]
$1/2 - 1/n^c$, for *every* c	Derandomized Direct Product	Theorem 2.27
$1/2 - 1/2^{\Omega(\sqrt{n})}$	Derandomized Direct Product & Nondeterminism	Theorem 2.32

2.4.2 Using nondeterminism

The derandomization described above gives hardness amplification up to $1/2 - 1/n^c$ *for any* constant c. This already improves upon the best previous result, namely Theorem 2.16. However, to go beyond this new techniques are required. The problem is that if we want C to be computable in time $\mathrm{poly}(n)$, we must take $k = \mathrm{poly}(n)$ and thus we amplify to at most $1/2 - 1/k = 1/2 - 1/\mathrm{poly}(n)$.

We solve this problem by taking full advantage of the power of NP, namely nondeterminism. This allows us to use a function $C : \{0,1\}^k \to \{0,1\}$ which is computable in *nondeterministic* time $\mathrm{poly}(n, \log(k))$; thus, the amplified function will still be in NP *for k as large as 2^n*.

Conversely, in Section 2.7.2 we show that any non-adaptive monotone black-box hardness amplification that amplifies to hardness $1/2 - 1/n^{\omega(1)}$ cannot be computed in P, i.e. the use of nondeterminism is essential.

We proceed by discussing the details of the derandomization (Sections 2.5.1, 2.5.2 and 2.5.3) and the use of nondeterminism (Section 2.5.4). The results obtained in these sections are summarized in Table 1. For clarity of exposition, we focus on the case where the original hard function f is balanced and is 1/3-hard. Hardness amplification from hardness $1/\mathrm{poly}(n)$ is discussed in Section 2.5.5, and hardness amplification of unbalanced functions is discussed in Section 2.7.1.

2.5 Proof of Main Theorem

In this section we prove our main theorem (i.e., Theorem 2.18).

2.5.1 Preserving indistinguishability

The main result in this subsection is that if G is pseudorandom in an appropriate sense, then the hardness of $(C \circ f^{\otimes k}) \circ G$ is roughly

$$1/2 - \text{ExpBias}\left[(C \circ g^{\otimes k}) \circ G\right]$$

for some δ-random function g. As we noted in the previous section, it will be sufficient for G to be *indistinguishability-preserving*. We give the definition of indistinguishability-preserving and then our main result.

Definition 2.20. *A generator* $G : \{0,1\}^l \to (\{0,1\}^n)^k$ *is said to be* indistinguishability-preserving for size t *if for all (possibly probabilistic) functions* $f_1, \ldots, f_k$, $g_1, \ldots, g_k$ *the following holds:*

If for every $i, 1 \le i \le k$ *the distributions*

$$U_n \cdot f_i(U_n) \text{ and } U_n \cdot g_i(U_n)$$

are ϵ*-indistinguishable for size* s*, then*

$$\sigma \cdot f_1(X_1) \cdots f_k(X_k) \text{ and } \sigma \cdot g_1(X_1) \cdots g_k(X_k)$$

are $k\epsilon$*-indistinguishable for size* $s - t$*, where* σ *is a random seed of length* l *and* $X_1 \cdots X_k$ *is the output of* $G(\sigma)$*.*

The fact that in the above definition we consider k f_i's and k g_i's implies that an $indistinguishability-preserving$ generator stays $indistinguishability-preserving$ when Xor'ed with any other generator (cf. the proof of Item 1 in Lemma 2.31). We will use this property in the proof of our main result.

Lemma 2.21. *Let* $f : \{0,1\}^n \to \{0,1\}$ *be* δ*-hard for size* s*, let* $G : \{0,1\}^l \to (\{0,1\}^n)^k$ *be a generator that is indistinguishability-preserving for size* t *and let* $C : \{0,1\}^k \to \{0,1\}$ *be any function. Then there exists a* δ'*-random* g*, with* $\delta/2 \le \delta' \le \delta$ *such that the function* $(C \circ f^{\otimes k}) \circ G : \{0,1\}^l \to \{0,1\}$ *has hardness*

$$\frac{1}{2} - \frac{\text{ExpBias}\left[(C \circ g^{\otimes k}) \circ G\right]}{2} - \frac{k}{s^{1/3}}$$

for circuits of size $\Omega\left(s^{1/3}/\log(1/\delta)\right) - t - \text{size}(C)$ *where* $\text{size}(C)$ *denotes the size of a smallest circuit computing* C*.*

Proof. By Lemma 2.9, there exists a δ'-random function g with $\delta/2 \le \delta' \le \delta$, such that $U_n \cdot f(U_n)$ and $U_n \cdot g(U_n)$ are ϵ-indistinguishable for size $\Omega(s\epsilon^2/\log(1/\delta))$. Since G is an indistinguishability-preserving for size t by assumption, this implies that

$$\sigma \cdot f(X_1) \cdots f(X_k) \text{ and } \sigma \cdot g(X_1) \cdots g(X_k)$$

are $k\epsilon$-indistinguishable for size $\Omega(s\epsilon^2/\log(1/\delta)) - t$, where here and below σ denotes a uniform random seed in $\{0,1\}^l$ and $X_1 \cdots X_k$ will denote the output of $G(\sigma)$. This in turn implies that

$$\sigma \cdot C(f(X_1)\cdots f(X_k)) \text{ and } \sigma \cdot C(g(X_1)\cdots g(X_k))$$

(i.e., $\sigma \cdot (C \circ f^{\otimes k}) \circ G(\sigma)$ and $\sigma \cdot (C \circ g^{\otimes k}) \circ G(\sigma)$) are $k\epsilon$-indistinguishable for size $\Omega(s\epsilon^2/\log(1/\delta)) - t - \text{size}(C)$. By Lemma 2.11,

$$\sigma \cdot (C \circ g^{\otimes k}) \circ G \text{ and } \sigma \cdot U_1$$

are (ExpBias $\left[(C \circ g^{\otimes k}) \circ G\right]/2$)-indistinguishable for any size. Therefore, we have that

$$\sigma \cdot (C \circ f^{\otimes k}) \circ G \text{ and } \sigma \cdot U_1$$

are (ExpBias $\left[(C \circ g^{\otimes k}) \circ G\right]/2 + k\epsilon$)-indistinguishable for size $\Omega(s\epsilon^2/\log(1/\delta)) - t - \text{size}(C)$. The result follows by setting $\epsilon = 1/s^{1/3}$ and applying Lemma 2.7. □

In particular, we note that the *identity generator* $G : \{0,1\}^{nk} \rightarrow (\{0,1\}^n)^k$, i.e. $G(x) = x$, is indistinguishability-preserving for size 0 (by a hybrid argument, see, e.g., [Gol3]), and thus Lemma 2.12 is a corollary of Lemma 2.21. However, the identity generator has seed-length nk and is therefore a very poor pseudorandom generator. Fortunately, there are indistinguishability-preserving pseudorandom generators with much shorter seeds which will allow us to use Lemma 2.21 to obtain much stronger hardness amplifications.

Lemma 2.22. *There is a constant c such that for every $n \geq 2$ and every $k = k(n)$ there is an explicitly computable generator $NW_k : \{0,1\}^l \rightarrow (\{0,1\}^n)^k$ with seed length $l = c \cdot n^2$ that is indistinguishability-preserving for size k^2.*

Proof. The generator is the main component of the generator by Nisan and Nisan and Wigderson [Nis1, NW], and is based on combinatorial designs. Specifically, we let $S_1, \ldots, S_k \subseteq [l]$ be an explicit family of sets such that $|S_i| = n$ for all i, and $|S_i \cap S_j| \leq \log k$ for all $i \neq j$. Nisan [Nis1] gives an explicit construction of such sets with $l = O(n^2)$. Then the generator $NW_k : \{0,1\}^l \rightarrow (\{0,1\}^n)^k$ is defined by

$$NW_k(\sigma) := (\sigma|_{S_1}, \ldots, \sigma|_{S_k}),$$

where $\sigma|_{S_i} \in \{0,1\}^n$ denotes the projection of σ onto the coordinates indexed by the set S_i.

The proof that this generator is indistinguishability preserving for size k^2 follows the arguments in [NW, STV]. For completeness, we sketch the proof here. Suppose that we have a circuit C of size $s - k^2$ distinguishing the distributions $\sigma \cdot f_1(\sigma|_{S_1}) \cdots f_k(\sigma|_{S_k})$ and $\sigma \cdot g_1(\sigma|_{S_1}) \cdots g_k(\sigma|_{S_k})$ with advantage greater than $k \cdot \epsilon$. For $i = 0, \ldots, k$, let H_i be the hybrid distribution

$$H_i = \sigma \cdot g_1(\sigma|_{S_1}) \cdots g_i(\sigma|_{S_i}) \cdot f_{i+1}(\sigma|_{S_{i+1}}) \cdots f_k(\sigma|_{S_k}).$$

Then there must exist an $i \in \{0, \ldots, k\}$ such that C distinguishes H_i from H_{i+1} with advantage greater than ϵ. The only difference between H_i and H_{i+1} is that H_i has the component $f_{i+1}(\sigma|_{S_{i+1}})$ while H_{i+1} has $g_{i+1}(\sigma|_{S_{i+1}})$. By averaging, we may fix all the bits of σ outside of S_{i+1} (as well as the randomness of f_j, g_j for $j \neq i+1$ if they are probabilistic functions) while preserving the advantage of C. Thus C distinguishes between two distributions of the form

$$\tau \cdot h_1(\tau) h_2(\tau) \cdots h_i(\tau) f_{i+1}(\tau) h_{i+2}(\tau) \cdots h_k(\tau),$$

and

$$\tau \cdot h_1(\tau) h_2(\tau) \cdots h_i(\tau) g_{i+1}(\tau) h_{i+2}(\tau) \cdots h_k(\tau),$$

where τ is uniform in $\{0,1\}^n$ and each h_j is a function of at most $|S_j \cap S_{i+1}|$ bits of τ. Then each h_j can be computed by a circuit of size smaller than $2^{|S_j \cap S_{i+1}|} \leq k$. Combining these $k-1$ circuits with C, we get a distinguisher between $\tau \cdot f_{i+1}(\tau)$ and $\tau \cdot g_{i+1}(\tau)$ of size $|C| + (k-1) \cdot k < s$ and advantage greater than ϵ. $\square$

2.5.2 Fooling the expected bias

In this subsection we prove a derandomized version of Lemma 2.14. Informally, we show that if C is computable in a restricted model of computation and G "fools" that restricted model of computation, then for any δ-random function g:

$$\mathrm{ExpBias}\left[(C \circ g^{\otimes k}) \circ G\right] \leq \sqrt{\mathrm{NoiseStab}_\delta[C] + \epsilon}.$$

The restricted model of computation we consider is that of nonuniform space-bounded algorithms which make one pass through the input, reading it in blocks of length n. These are formally modeled by the following kind of branching programs.

Definition 2.23. *A (probabilistic, read-once, oblivious)* branching program *of* size s *with* block-size n *is a finite state machine with s states, over the alphabet $\{0,1\}^n$ (with a fixed start state, and an arbitrary number of accepting states). Each edge is labelled with a symbol in $\{0,1\}^n$. For every state a and symbol $\alpha \in \{0,1\}^n$, the edges leaving a and labelled with α are assigned a probability distribution. Then computation proceeds as follows. The input is read sequentially, one block of n bits at a time. If the machine is in state a and it reads α, then it chooses an edge leaving a and labelled with α according to its probability, and moves along it.*

Now we formally define pseudorandom generators against branching programs.

Definition 2.24. *A generator $G : \{0,1\}^l \to (\{0,1\}^n)^k$ is* ϵ-pseudorandom against branching programs of size s and block-size n *if for every branching program* B *of size s and block-size n:*

$$\left|\Pr[\mathrm{B}(G(U_l)) = 1] - \Pr[\mathrm{B}(U_{nk}) = 1]\right| \leq \epsilon.$$

In [Nis2], Nisan builds an unconditional pseudorandom generator against branching programs. Its parameters (specialized for our purposes) are given in the following theorem.

Theorem 2.25 ([Nis2]). *For every n and $k \leq 2^n$, there exists a generator*

$$N_k : \{0,1\}^l \to (\{0,1\}^n)^k$$

such that:

- N_k *is* 2^{-n}*-pseudorandom against branching programs of size* 2^n *and block-size* n.
- N_k *has seed length* $l = O(n \log k)$.
- N_k *is explicitly computable.*

Note that Nisan [Nis2] does not mention *probabilistic* branching programs. However, if there is a probabilistic branching program distinguishing the output of the generator from uniform, then by a fixing of the coin tosses of the branching program there is a *determinisitic* branching program that distinguishes the output of the generator from uniform.

We now state the derandomized version of Lemma 2.14.

Lemma 2.26. *Let*

- $g : \{0,1\}^n \to \{0,1\}$ *be a* δ*-random function,*
- $C : \{0,1\}^k \to \{0,1\}$ *be computable by a branching program of size* t *and block-size* 1,
- $G : \{0,1\}^l \to (\{0,1\}^n)^k$ *be* $\epsilon/2$*-pseudorandom against branching programs of size* t^2 *and block-size* n.

Then $\mathrm{ExpBias}\left[(C \circ g^{\otimes k}) \circ G\right] \leq \sqrt{\mathrm{NoiseStab}_\delta[C] + \epsilon}$.

Proof. We will not show that G fools the expected bias, but rather the following related quantity. For a probabilistic boolean function $h(x;r)$ we define its (normalized) *expected collision probability* as

$$\mathrm{ExpCP}[h] := \mathop{\mathrm{E}}_x\big[2 \cdot \Pr_{r,r'}[h(x;r) = h(x;r')] - 1\big].$$

The same reasoning that proves Lemma 2.14 also shows that for every probabilistic boolean function h:

$$\mathrm{ExpBias}\,[h] \leq \sqrt{\mathrm{ExpCP}[h]}. \tag{2.3}$$

More specifically, Inequality (2.3) holds because

$$\begin{aligned}
\mathrm{ExpBias}\,[h] &= \mathop{\mathrm{E}}_{x\leftarrow U_n}\left[Bias\,[h(x)]\,\right] \\
&\leq \sqrt{\mathop{\mathrm{E}}_{x\leftarrow U_n}\left[Bias\,[h(x)]^2\,\right]} \qquad \text{(by Cauchy-Schwartz)} \\
&= \sqrt{\mathrm{ExpCP}[h]}.
\end{aligned}$$

Let $h(x;r) : (\{0,1\}^n)^k \to \{0,1\}$ be the probabilistic function $C \circ g^{\otimes k}$. Even though h is defined in terms of g, it turns out that its expected collision probability is the same for all δ-random functions g. Specifically, for $x = (x_1, \ldots, x_k)$, the only dependence of the collision probability $\Pr_{r,r'}[h(x;r) = h(x;r')]$ on x_i comes from whether $g(x_i)$ is a coin flip (which occurs with probability δ over the choice of x_i), $g(x_i) = 1$ (which occurs with probability $(1-\delta)/2$), or $g(x_i) = 0$ (which occurs with probability $(1-\delta)/2$). In the case where $g(x_i)$ is a coin flip, then the i'th bits of the two inputs fed to C (i.e. $g(x_i;r)$ and $g(x_i;r')$) are random and independent, and otherwise they are equal and fixed (according to $g(x_i)$). It can be verified that this corresponds precisely to the definition of noise stability, so we have:

$$\mathrm{ExpCP}[h] = \mathrm{NoiseStab}_\delta[C]. \tag{2.4}$$

Now we construct a probabilistic branching program $M : (\{0,1\}^n)^k \to \{0,1\}$ of size t^2 and block-size n such that for every $x \in (\{0,1\}^n)^k$:

$$\Pr[M(x) = 1] = \Pr_{r,r'}[h(x;r) = h(x;r')].$$

To do this, we first note that, using the branching program for C, we can build a probabilistic branching program of size t with block-size n which computes $C \circ g^{\otimes k}$: The states of the branching program are the same as those of the branching program for C, and we define the transitions as follows. Upon reading symbol $\alpha \in \{0,1\}^n$ in state s, if $g(\alpha) = 0$ (resp. $g(\alpha) = 1$), we deterministically go to the state given by the 0-transition (resp., 1-transition) of C from state s, and if $g(\alpha)$ is a coin flip, then we put equal probability on these two transitions.

Then, to obtain M, run two *independent* copies of this branching program (i.e., using independent choices for the probabilistic state transitions) and accept if and only if both of them accept or both of them reject. Now,

$$\begin{aligned}
&\left|\mathrm{ExpCP}[(C \circ g^{\otimes k}) \circ G] - \mathrm{NoiseStab}_\delta[C]\right| \\
&= \left|\mathrm{ExpCP}[(C \circ g^{\otimes k}) \circ G] - \mathrm{ExpCP}[C \circ g^{\otimes k}]\right| \quad \text{(by (2.4))} \\
&= 2 \cdot \left|\Pr[M \circ G(U_l) = 1] - \Pr[M(U_{n\cdot k}) = 1]\right| \\
&\leq \epsilon. \quad \text{(by pseudorandomness of } G\text{)}
\end{aligned}$$

The lemma follows combining this with Equation (2.3). $\square$

2.5.3 Amplification up to $1/2 - 1/\operatorname{poly}$

In this subsection we sketch our hardness amplification up to $1/2-1/n^c$, for every c:

Theorem 2.27. *If there is a balanced function $f : \{0,1\}^n \to \{0,1\}$ in* NP *that is $(1/3)$-hard for size $s(n) \geq n^{\omega(1)}$, then for every $c > 0$ there is a function $f' : \{0,1\}^m \to \{0,1\}$ in* NP *that is $(1/2 - 1/m^c)$-hard for size $(s(\sqrt{m}))^{\Omega(1)}$.*

To amplify we use the Tribes function of Ben-Or and Linial [BL], a monotone read-once DNF.

Definition 2.28. *The* Tribes *function on k bits is:*

$$\mathrm{Tribes}_k(x_1, \ldots, x_k) :=$$

$$(x_1 \wedge \ldots \wedge x_b) \vee (x_{b+1} \wedge \ldots \wedge x_{2b}) \vee \ldots \vee (x_{k-b+1} \wedge \ldots \wedge x_k)$$

where there are k/b clauses each of size b, and b is the largest integer such that $(1 - 2^{-b})^{k/b} \geq 1/2$. Note that this makes $b = O(\log k)$.

The Tribes DNF has very low noise stability when perturbed with constant noise.

Lemma 2.29 ([O'D, MO])**.** *For every constant $\delta > 0$,*

$$\mathrm{NoiseStab}_\delta[\mathrm{Tribes}_k] \leq \frac{1}{k^{\Omega(1)}}.$$

A key step in our result is that Tribes_k is easily computable by a branching program of size $O(k)$, and therefore we can use Lemma 2.26 to fool its expected bias.

We now define the generator we will use in our derandomized direct product construction.

Definition 2.30. *Given n and $k \leq 2^n$, define the generator $\Gamma_k : \{0,1\}^m \to (\{0,1\}^n)^k$ as follows:*

$$\Gamma_k(x, y) := NW_k(x) \oplus N_k(y),$$

where $\oplus$ denotes bitwise Xor.

We recall the properties of Γ we are interested in:

Lemma 2.31. *The following hold:*

1. *Γ_k is indistinguishability-preserving for size k^2.*
2. *Γ_k is 2^{-n}-pseudorandom against branching programs of size 2^n and block-size n.*
3. *Γ_k has seed length $m = O(n^2)$.*

4. Γ_k is explicitly computable (see Definition 2.19 for the definition of explicit).

Proof. Item (1) follows from Lemma 2.22 and the fact that an indistinguishability-preserving generator Xor'ed with any fixed string is still indistinguishability-preserving. More specifically, suppose, for the sake of contradiction, that $\Gamma_k(x, y) = NW_k(x) \oplus N_k(y)$ is not indistinguishability-preserving. Then there are functions $f_1, \ldots, f_k$ and $g_1, \ldots, g_k$ such that for every i the distributions $U_n \cdot f_i(U_n)$ and $U_n \cdot g_i(U_n)$ are indistinguishable, yet the distributions

$$(x, y) \cdot f_1(NW_1(x) \oplus N_1(y)) \cdots f_k(NW_k(x) \oplus N_k(y))$$

and

$$(x, y) \cdot g_1(NW_1(x) \oplus N_1(y)) \cdots g_k(NW_k(x) \oplus N_k(y))$$

are distinguishable (for random x, y). Then, by averaging, they are distinguishable for some fixed value of $y = \tilde{y}$. Thus, we obtain that

$$x \cdot f'_1(NW_1(x)) \cdots f'_k(NW_k(x))$$

and

$$x \cdot g'_1(NW_1(x)) \cdots g'_k(NW_k(x))$$

are distinguishable (for random x), where $f'_i(z) = f_i(z \oplus N_i(\tilde{y}))$, $g'_i(z) = g_i(z \oplus N_i(\tilde{y}))$. (Note that we hardwire the fixed part of the seed $\tilde{y}$ in the distinguisher.) Now observe that the indistinguishability of $U_n \cdot f_i(U_n)$ and $U_n \cdot g_i(U_n)$ implies the indistinguishability of $U_n \cdot f'_i(U_n)$ and $U_n \cdot g'_i(U_n)$, because the mapping $T(u \cdot v) = (u \oplus N_i(\tilde{y})) \cdot v$ transforms the latter pair of distributions to the former. (There is no loss in the circuit size assuming that circuits have input gates for both the input variables and their negations.) But this is a contradiction because NW is indistinguishability-preserving.

Item (2) follows from Theorem 2.25 and the fact that Xor'ing with any fixed string (in particular, $NW_k(x)$ for any x) preserves pseudorandomness against branching programs.

Item (3) is an immediate consequence of the seed lengths of NW_k (Lemma 2.22) and N_k (Theorem 2.25).

Item (4) follows from the fact that NW_k is explicit (Lemma 2.22) and N_k is explicit (Theorem 2.25). $\square$

Proof of Theorem 2.27. Given $f : \{0,1\}^n \to \{0,1\}$ that is δ-hard for size $s(n)$ (for $\delta = 1/3$) and a constant c, let $k = n^{c'}$ for $c' = O(c)$ to be determined later. Consider the function $f' : \{0,1\}^m \to \{0,1\}$ defined by

$$f' := (\mathrm{Tribes}_k \circ f^{\otimes k}) \circ \Gamma_k.$$

Note that $f' \in \mathsf{NP}$ since $f \in \mathsf{NP}$, Tribes is monotone and both Γ and Tribes are efficiently computable.

We now analyze the hardness of f'. Since Γ_k is indistinguishability-preserving for size k^2 by Lemma 2.31, Lemma 2.21 implies that there is a δ'-random function g (for $\delta/2 \leq \delta' \leq \delta$) such that f' has hardness

$$\frac{1}{2} - \frac{\mathrm{ExpBias}\left[(\mathrm{Tribes}_k \circ g^{\otimes k}) \circ \Gamma_k\right]}{2} - \frac{k}{s(n)^{1/3}} \tag{2.5}$$

for circuits of size $\Omega\left(s(n)^{1/3}\right) - k^2 - \mathrm{size}(\mathrm{Tribes}_k)$. Next we bound the hardness. By Lemma 2.31, we know that Γ_k is 2^{-n}-pseudorandom against branching programs of size 2^n and block-size n. In particular, since $k = \mathrm{poly}(n)$, Γ_k is $1/k$-pseudorandom against branching programs of size $9k$ and block-size n. Since, as we noted before, Tribes_k is easily computable by a branching program of size $O(k)$, we can apply Lemma 2.26 (noting that $O(k)^2 = \mathrm{poly}(n) \ll 2^n$) in order to bound $\mathrm{ExpBias}\left[(\mathrm{Tribes}_k \circ g^{\otimes k}) \circ \Gamma_k\right]$ by $\sqrt{\mathrm{NoiseStab}_{\delta'}[\mathrm{Tribes}_k] + 2/k}$. And the noise stability inside the square root is at most $1/k^{\Omega(1)}$ by Lemma 2.29. Since $k = \mathrm{poly}(n)$ and $s(n) = n^{\omega(1)}$, the $k/s^{1/3}$ term in the hardness (2.5) is negligible and we obtain hardness at least $1/2 - 1/k^{\Omega(1)}$.

We now bound the circuit size: Since Tribes_k is computable by circuits of size $O(k)$, and $s(n) = n^{\omega(1)}$, the size is at least $s(n)^{\Omega(1)}$.

Now note that f' has input length $m = m(n) = O(n^2)$ by Lemma 2.31. Strictly speaking, we have only defined f' for certain input lengths; however, it is easy to extend the function to every input length, by simply defining $f'(x) := f'(x')$ where x' consists of the first $m(n)$ bits of x and n is the largest integer such that $m(n) \leq |x|$. It is easy to check that f' still has hardness $1/2 - 1/k^{\Omega(1)} = 1/2 - 1/n^{\Omega(c')}$. The result then follows by an appropriate choice of $c' = O(c)$. □

2.5.4 Using nondeterminism

In this subsection we discuss how to use nondeterminism to get the following theorem.

Theorem 2.32. *If there is a balanced function $f : \{0,1\}^n \to \{0,1\}$ in* NP *that is $(1/3)$-hard for size $s(n)$, then there is a function $f' : \{0,1\}^m \to \{0,1\}$ in* NP *that is $(1/2 - 1/s(\sqrt{m})^{\Omega(1)})$-hard for size $s(\sqrt{m})^{\Omega(1)}$.*

Our main observation is that Tribes_k is a DNF with clause size $O(\log k)$, and therefore it is computable in nondeterministic time $\mathrm{poly}(n)$ *even when k is superpolynomial in n*:

Lemma 2.33. *Let $f : \{0,1\}^n \to \{0,1\}$ be in* NP, *and let $G_k : \{0,1\}^l \to (\{0,1\}^n)^k$ be any explicitly computable generator (see Definition 2.19) with $l \geq n$. Then the function $f' := (\mathrm{Tribes}_k \circ f^{\otimes k}) \circ G_k$ is computable in* NP *for every $k = k(n) \leq 2^n$.*

Proof. We compute $f'(\sigma)$ nondeterministically as follows: Guess a clause $v_i \wedge v_{i+1} \wedge \cdots \wedge v_j$ in Tribes_k. Accept if for every h s.t. $i \leq h \leq j$ we have $f(X_h) = 1$, where $G(\sigma) = (X_1, \ldots, X_k)$ and the values $f(X_h)$ are computed using the NP algorithm for f.

It can be verified that this algorithm has an accepting computation path on input σ iff $f'(\sigma) = 1$. Note that the clauses have size logarithmic in k, which is polynomial in n. Moreover, G is explicitly computable. The result follows. $\square$

Now the proof of Theorem 2.32 proceeds along the same lines as the proof of Theorem 2.27, setting $k := s(n)^{\Omega(1)}$.

2.5.5 Amplifying from hardness $1/\operatorname{poly}$

Our amplification from hardness $\Omega(1)$ to $1/2 - \epsilon$ (Theorem 2.27) can be combined with O'Donnell's amplification from hardness $1/\operatorname{poly}$ to hardness $\Omega(1)$ to obtain an amplification from $1/\operatorname{poly}$ to $1/2 - \epsilon$. However, since O'Donnell's construction blows up the input length polynomially, we would only obtain $\epsilon = 1/s(n^{\Omega(1)})$ (where the hidden constant depends on the initial polynomial hardness) rather than $\epsilon = 1/s(\sqrt{n})^{\Omega(1)}$ (as in Theorem 2.27). Thus we show here how to amplify directly from $1/\operatorname{poly}$ to $1/2 - \epsilon$ using our approach. For this we need a combining function C that is more involved than the Tribes function. The properties of C that are needed in the proof of Theorem 2.18 are captured by the following lemma.

Lemma 2.34. *For every $\delta(n) = 1/n^{O(1)}$, there is a sequence of functions $C_k : \{0,1\}^k \to \{0,1\}$, such that for every $k = k(n)$ with $n^{\omega(1)} \leq k \leq 2^n$, the following hold:*

1. $\mathrm{NoiseStab}_\delta[C_k] \leq 1/k^{\Omega(1)}$.
2. *For every $f : \{0,1\}^n \to \{0,1\}$ in NP and every explicitly computable generator (see Definition 2.19) $G_k : \{0,1\}^l \to (\{0,1\}^n)^k$ with $l \geq n$, the function $(C_k \circ f^{\otimes k}) \circ G_k$ is in NP.*
3. *C_k can be computed by a branching program of size $\mathrm{poly}(n) \cdot k$, and also by a circuit of size $\mathrm{poly}(n) \cdot k$.*

Before proving Lemma 2.34, let us see how it can be used to prove our main theorem.

Theorem 2.35 (Thm. 2.18, restated)**.** *If there is a balanced function $f : \{0,1\}^n \to \{0,1\}$ in NP that is $1/\operatorname{poly}(n)$-hard for size $s(n)$, then there is a function $f' : \{0,1\}^m \to \{0,1\}$ in NP that is $(1/2 - 1/s(\sqrt{m})^{\Omega(1)})$-hard for size $s(\sqrt{m})^{\Omega(1)}$.*

Proof. Let $f : \{0,1\}^n \to \{0,1\}$ be a balanced function in NP that is $\delta = \delta(n)$-hard for size $s(n)$, where $\delta \geq 1/n^{O(1)}$. Let $k = k(n) := s(n)^{1/7}$ and let C_k be the function guaranteed by Lemma 2.34. Let Γ_k be the the generator from Definition 2.30. Consider the function $f' : \{0,1\}^m \to \{0,1\}$ defined by $f' := (C_k \circ f^{\otimes k}) \circ \Gamma_k$. Note that $f' \in$ NP by Item 2 in Lemma 2.34.

We now analyze the hardness of f'. Since Γ_k is indistinguishability-preserving for size k^2 (by Lemma 2.31), Lemma 2.21 implies that there is a δ'-random function g (for $\delta/2 \leq \delta' \leq \delta$) such that f' has hardness

$$\alpha(m) = \frac{1}{2} - \frac{\mathrm{ExpBias}\left[(C_k \circ g^{\otimes k}) \circ \Gamma_k\right]}{2} - \frac{k}{s(n)^{1/3}} \tag{2.6}$$

for circuits of size

$$s'(m) = \Omega\left(\frac{s(n)^{1/3}}{\log(1/\delta)}\right) - k^2 - \mathrm{size}(C_k).$$

We first bound the hardness $\alpha(m)$. By Lemma 2.31, we know that Γ_k is 2^{-n}-pseudorandom against branching programs of size 2^n and block-size n. Since the branching program for computing C_k has size $\mathrm{poly}(n) \cdot k$, and $(\mathrm{poly}(n) \cdot k)^2 \ll 2^n$ (by our choice of $k(n)$), we may apply Lemma 2.26 in order to bound $\mathrm{ExpBias}\left[(C_k \circ g^{\otimes k}) \circ \Gamma_k\right]$ by $\sqrt{\mathrm{NoiseStab}_{\delta'}[C_k] + 2/2^n}$. This noise stability is at most $1/k^{\Omega(1)}$ by Item 1 in Lemma 2.34. Using the fact that $k = s(n)^{1/7}$, we have

$$\alpha(m) \geq \frac{1}{2} - \frac{\sqrt{1/k^{\Omega(1)} - 2/2^n}}{2} - \frac{k}{s(n)^{1/3}} = \frac{1}{2} - \frac{1}{s(n)^{\Omega(1)}}.$$

We now bound the circuit size $s'(m)$. Since C_k is computable by a circuit of size $\mathrm{poly}(n) \cdot k$ (by Item 3 in Lemma 2.34) and $\log(1/\delta) = O(\log n)$ and $s(n) = n^{\omega(1)}$, we have

$$s'(m) = \Omega\left(\frac{s(n)^{1/3}}{\log n}\right) - s(n)^{2/7} - \mathrm{poly}(n) = s(n)^{\Omega(1)}.$$

To conclude, we note that f' has input length $m = O(n^2)$ by Lemma 2.31, so $s(n) = s(\Omega(\sqrt{m})) = s(\sqrt{m})^{\Omega(1)}$, and we indeed obtain hardness $\alpha(m) = 1/2 - 1/s(\sqrt{m})^{\Omega(1)}$ for size $s'(m) = s(\sqrt{m})^{\Omega(1)}$. Strictly speaking, we have only defined f' for certain input lengths; however, it is easy to extend the function to every input length, cf. the end of the proof of Theorem 2.27. $\square$

The rest of this subsection is devoted to the proof of Lemma 2.34. Recall that amplification from hardness $\Omega(1)$ (Theorem 2.27) relies on the fact that the Tribes DNF has low noise stability with respect to noise parameter $\delta = \Omega(1)$ (i.e., Lemma 2.29). Similarly, to amplify from hardness $1/\mathrm{poly}(n)$ we need to employ a combining function that has low noise stability with respect to noise $1/\mathrm{poly}(n)$. To this end, following [O'D], we employ the recursive-majorities function, RMaj_r. Let Maj denote the majority function.

Definition 2.36. *The* RMaj_r *function on* 3^r *bits is defined recursively by:*

$$\begin{aligned}\mathrm{RMaj}_1(x_1,x_2,x_3) &:= \mathrm{Maj}(x_1,x_2,x_3)\\ \mathrm{RMaj}_r(x_1,\ldots,x_{3^r}) &:= \mathrm{RMaj}_{r-1}\left(\mathrm{Maj}(x_1,x_2,x_3),\ldots,\mathrm{Maj}(x_{3^r-2},x_{3^r-1},x_{3^r})\right)\end{aligned}$$

The following lemma quantifies the noise stability of RMaj_r.

Lemma 2.37 ([O'D], Proposition 11)**.** *There is a constant* c *such that for every* $\delta > 0$ *and every* $r \geq c \cdot \log(1/\delta)$*, we have*

$$\mathrm{NoiseStab}_\delta[\mathrm{RMaj}_r] \leq \frac{1}{4}.$$

Note that if $r = O(\log n)$ then RMaj_r is a function of $3^r = \mathrm{poly}(n)$ bits.

However, when $r = O(\log n)$, RMaj_r does not have sufficiently low noise stability to be used on its own. For this reason, we will combine RMaj with Tribes. (The same combination of RMaj and Tribes is employed by O'Donnell [O'D], albeit for a different setting of parameters.)

Proof of Lemma 2.34. Given n and $\delta = \delta(n) \geq 1/n^{O(1)}$, let $r := c \cdot \log(1/\delta)$ for a constant c to be chosen later. Assume, without loss of generality, that r and $k/3^r$ are integers. The function $C_k : \{0,1\}^k \to \{0,1\}$ is defined as follows

$$C_k := \mathrm{Tribes}_{k/3^r} \circ \mathrm{RMaj}_r^{\otimes k/3^r}.$$

We now prove that C_k satisfies the required properties.

Item (1): We will use the following result from [O'D].

Lemma 2.38 ([O'D], Proposition 8)**.** *If* h *is a balanced boolean function and* $\varphi : \{0,1\}^\ell \to \{0,1\}$ *is any boolean function, then*

$$\mathrm{NoiseStab}_\delta[\varphi \circ h^{\otimes \ell}] = \mathrm{NoiseStab}_{\frac{1}{2}-\frac{\mathrm{NoiseStab}_\delta[h]}{2}}[\varphi].$$

Letting c be a sufficiently large constant (recall that $r = c \cdot \log(1/\delta)$), by Lemma 2.37 we have that $\mathrm{NoiseStab}_\delta[\mathrm{RMaj}_r]/2 \geq 1/2 - 1/8 \geq 3/8$. Now note that RMaj_r is balanced because taking the bitwise complement of an input x also negates the value of $\mathrm{RMaj}_r(x)$. Hence, by Lemma 2.38,

$$\begin{aligned}\mathrm{NoiseStab}_\delta[\mathrm{Tribes}_{k/3^r} \circ \mathrm{RMaj}_r^{\otimes k/3^r}] &= \mathrm{NoiseStab}_{3/8}[\mathrm{Tribes}_{k/3^r}]\\ &\leq \frac{1}{(k/3^r)^{\Omega(1)}} = \frac{1}{k^{\Omega(1)}},\end{aligned}$$

where the last two equalities use Lemma 2.29 and the fact that $k = n^{\omega(1)}$ and $r = O(\log n)$.

The proof of Item (2) is similar to the proof of Lemma 2.33. To compute

$$(\mathrm{Tribes}_{k/3^r} \circ \mathrm{RMaj}_r^{\otimes k/3^r} \circ f^{\otimes k}) \circ G_k,$$

we guess a clause of the $\mathrm{Tribes}_{k/3^r}$ and verify that all the RMaj_r evaluations feeding into it are satisfied (using the NP algorithm for f). The only additional observation is that each of the recursive majorities depends only on $3^r = \mathrm{poly}(n)$ bits of the input, and hence can be computed in time polynomial in n.

Item (3): As noted earlier, $\mathrm{Tribes}_{k/3^r}$ is easily computable by a branching program of size $O(k)$. RMaj_r, on the other hand, can be computed by a branching program of size $\mathrm{poly}(n)$. Indeed, $\mathrm{Maj}(x_1, x_2, x_3)$ is clearly computable by a branching program of constant size c, and therefore $\mathrm{RMaj}_r = \mathrm{RMaj}_{r-1}\left(\mathrm{Maj}(x_1, x_2, x_3), \ldots, \mathrm{Maj}(x_{3^r-2}, x_{3^r-1}, x_{3^r})\right)$ can be computed by a branching program whose size is at most c times the size of RMaj_{r-1}. By induction it follows that RMaj_r can be computed in size $c^r = \mathrm{poly}(n)$.

By composing the branching program of size $O(k)$ for Tribes with the branching program of size $\mathrm{poly}(n)$ for RMaj, we can compute C_k by a branching program of size $\mathrm{poly}(n) \cdot k$. □

2.6 Extensions

2.6.1 On the possibility of amplifying hardness up to $1/2 - 1/2^{\Omega(n)}$

Even when starting from a function that is δ-hard for size $2^{\Omega(n)}$, our results (Theorem 2.18) only amplify hardness up to $1/2 - 1/2^{\Omega(\sqrt{n})}$ (rather than $1/2 - 1/2^{\Omega(n)}$). In this section we discuss the possibility of amplifying hardness in NP up to $1/2 - 1/2^{\Omega(n)}$, when starting with a function that is δ-hard for size $2^{\Omega(n)}$. The problem is that the seed length of our generator in Lemma 2.31 is *quadratic* in n, rather than linear. To amplify hardness up $1/2 - 1/2^{\Omega(n)}$ we need a generator (for every $k = 2^{\Omega(n)}$) with the same properties of the one in Lemma 2.31, but with linear seed length.

Recall our generator is the Xor of an indistinguishability-preserving generator and a generator that is pseudorandom against branching programs. While it is an open problem to exhibit a generator with linear seed length that is pseudorandom against branching programs, an indistinguishability-preserving generator with linear seed length is given by the following lemma.

Lemma 2.39. *For every constant γ, $0 < \gamma < 1$, there is a constant c such that for every n there is an explicitly computable generator $NW'_{2^{n/c}} : \{0,1\}^l \to (\{0,1\}^n)^{2^{n/c}}$ with seed length $l = c \cdot n$ that is indistinguishability-preserving for size $2^{\gamma \cdot n}$.*

The generator is due to Nisan and Wigderson [NW] and Impagliazzo and Wigderson [IW1]. The approach is the same as for the generator used in Lemma 2.22, except we now require a design consisting of $2^{\Omega(n)}$ sets of size n in a universe of size $O(n)$, with pairwise intersections of size at most $\gamma n/2$. An explicit construction of such a design is given in [GV].[5]

Theorem 2.40. *Suppose that there exists an explicit generator $N'_{2^n} : \{0,1\}^l \to (\{0,1\}^n)^{2^n}$ that is 2^{-n}-pseudorandom against branching programs of size 2^n and block-size n and that has seed length $l = O(n)$. Then the following holds: If there is a balanced function $f : \{0,1\}^n \to \{0,1\}$ in* NP *that is $1/\mathrm{poly}(n)$-hard for size $2^{\Omega(n)}$, then there is a function $f' : \{0,1\}^m \to \{0,1\}$ in* NP *that is $(1/2 - 1/2^{\Omega(n)})$-hard for size $2^{\Omega(n)}$.*

A slightly more careful analysis shows that all the branching programs considered in our constructions have *width* much smaller than their size, where the width of a branching program is the maximum, over i, of the number of states that are reachable after reading i symbols. Specifically, the branching program for Tribes has constant width, and the one for the function in Lemma 2.34, i.e. Tribes composed with RMaj, has width $\mathrm{poly}(n)$ independent of k. Thus to prove the conclusion in the statement of Theorem 2.40 it would suffice to exhibit an explicit pseudorandom generator that fools such restricted branching programs.

For amplifying from constant hardness, it suffices to instead have a generator fooling *constant-depth circuits* of size 2^n with seed length $O(n)$. (The generator of Nisan [Nis1] has seed length $\mathrm{poly}(n)$.) The reason is that our proof that PRGs versus branching programs "fool" the expected bias also works for PRGs versus constant-depth circuits, provided that the combining function is computable in constant depth. The Tribes function is depth 2 by definition (but the recursive majorities RMaj is not constant-depth, and hence this would only amplify from constant hardness).

More generally, we only need, for every constant $\gamma > 0$, a generator $G : \{0,1\}^{O(n)} \to (\{0,1\}^n)^k$ where $k = 2^{\gamma n}$ such that for every δ-random function g,

$$\mathrm{ExpBias}\left[(C_k \circ g^{\otimes k}) \circ G\right] = 2^{-\Omega(n)},$$

where, for example, $C_k = \mathrm{Tribes}_k$ (when δ is constant). As in the proof of Lemma 2.14, in proving such a statement, it may be convenient to work instead with the (polynomially related) expected collision probability. An important property of $C_k = \mathrm{Tribes}_k$ we used in bounding the expected bias with respect to G is that it gives expected bias $2^{-\Omega(n)}$ if G is replaced with a truly random generator (i.e. using seed length $n \cdot k$) and δ is constant. One might try to use a different monotone combining function with this property, provided it can also be evaluated in nondeterministic time $\mathrm{poly}(n)$.

[5] Alternatively, we can use the randomized algorithm described in [IW1] that computes such sets $S_1, \ldots, S_M$ with probability exponentially close to 1 using $O(n)$ random bits. This is sufficient for constructing an indistinguishability-preserving generator.

2.6.2 Impagliazzo and Wigderson's hardness amplification

Our framework gives a new proof of the hardness amplification by Impagliazzo and Wigderson [IW1]. Impagliazzo and Wigderson [IW1] show that if $\mathsf{E} := \mathsf{Time}\left(2^{\mathsf{O(n)}}\right)$ contains a function $f : \{0,1\}^n \to \{0,1\}$ that requires (in the worst-case) circuits of size $2^{\Omega(n)}$, then E contains a function f' that is $\left(1/2 - 1/2^{\Omega(n)}\right)$-hard for circuits of size $2^{\Omega(n)}$. The main contribution in [IW1] is amplification from constant hardness, e.g. $1/3$, to $\left(1/2 - 1/2^{\Omega(n)}\right)$ (amplification from worst-case hardness and constant hardness was essentially already established in [BFNW, Imp1]). The improvement over the standard Yao Xor Lemma is that the input length of the amplified function increases only by a constant factor. In this section, we sketch a simple proof of this result using the framework developed in earlier sections. While other, more recent hardness amplifications achieving the same result for E are known [STV, SU2, Uma1], the original one by Impagliazzo and Wigderson is still interesting because it can be implemented in "low" complexity classes, such as the polynomial-time hierarchy, while the others cannot. This is due to the fact that they actually amplify from worst-case hardness and will be proved in Chapter 3.

The construction of [IW1] uses an *expander-walk* generator $W_k : \{0,1\}^l \to (\{0,1\}^n)^k$, which uses its seed of length $l = n + O(k)$ to do a random walk of length k (started at a random vertex) in a constant-degree expander graph on 2^n vertices. More background on such generators can be found in [Gol2, Sec 3.6.3]. The construction of [IW1] Xor's the expander-walk generator with the (first k outputs of the) indistinguishability-preserving generator from Lemma 2.39:

Definition 2.41. *Let $k = c \cdot n$ for a constant $c > 1$. Let $NW''_k : \{0,1\}^{O(n)} \to (\{0,1\}^n)^k$ be a generator that is indistinguishability-preserving for size $2^{n/c}$ as given by Lemma 2.39. The generator $IW_k : \{0,1\}^l \to (\{0,1\}^n)^k$ is defined as*

$$IW_k(x, y) := NW''_k(x) \oplus W_k(y).$$

The seed length of IW_k is $l = O(n)$.

Given a function f that is $1/3$-hard for size $s = 2^{\Omega(n)}$, the Impagliazzo–Wigderson amplification defines

$$f' := (\mathsf{Xor} \circ f^{\otimes k}) \circ IW_k : \{0,1\}^{O(n)} \to \{0,1\},$$

where $k = c \cdot n$ for a constant c that depends on the hidden constant in the $s = 2^{\Omega(n)}$. They prove the following about this construction.

Theorem 2.42 ([IW1])**.** *If there is a function $f : \{0,1\}^n \to \{0,1\}$ in E that is $1/3$-hard for size $2^{\Omega(n)}$, then there is a function $f' : \{0,1\}^m \to \{0,1\}$ in E that is $(1/2 - 2^{-\Omega(m)})$-hard for size $2^{\Omega(m)}$.*

Proof. By Theorem 2.21 there exists a δ'-random function $g : \{0,1\}^n \to \{0,1\}$, where δ' is a constant, such that the hardness of $f' : \{0,1\}^{O(n)} \to \{0,1\}$ is $1/2 - \text{ExpBias}\left[(\mathsf{Xor} \circ g^{\otimes k}) \circ IW_k\right] - 2^{-\Omega(n)}$ for circuits of size $2^{\Omega(n)}$.

We now bound the hardness. Whenever some $IW_i(x)$ falls in the set of inputs of density $2 \cdot \delta'$ where the output of g is a coin flip, the bias of $(\mathsf{Xor} \circ g^{\otimes k}) \circ IW_k$ is 0. Therefore

$$\text{ExpBias}\left[(\mathsf{Xor} \circ g^{\otimes k}) \circ IW_k\right] \leq \Pr_x[\forall i : IW_i(x) \notin H] \leq 2^{-\Omega(n)},$$

where in the last inequality we use standard hitting properties of expander walks (see e.g. [Gol1] for a proof), and take c to be a sufficiently large constant. □

2.7 Limitations of monotone hardness amplification

2.7.1 On the hypothesis that f is balanced

The hardness amplification results in the previous sections start from balanced functions. In this section we study this hypothesis. Our main finding is that, while this hypothesis is not necessary for hardness amplification within NP/poly (i.e., non-deterministic polynomial size circuits), it is likely to be necessary for hardness amplification within NP.

To see that this hypothesis is not necessary for amplification within NP/poly, note that if the quantity $\Pr_x[f(x) = 1]$ of the original hard function $f : \{0,1\}^n \to \{0,1\}$ is known, then we can easily pad f to obtain a balanced function $\bar{f} : \{0,1\}^{n+1} \to \{0,1\}$:

$$\bar{f}(x,p) := \begin{cases} f(x) & \text{if } p = 0 \\ 0 & \text{if } p = 1 \text{ and } x \leq \Pr_x[f(x) = 1] \cdot 2^n \\ 1 & \text{if } p = 1 \text{ and } x > \Pr_x[f(x) = 1] \cdot 2^n \end{cases}$$

It is easy to see that $\bar{f}$ is $1/\operatorname{poly}(n)$-hard if f is. Since a circuit can (non-uniformly) know $\Pr_x[f(x) = 1]$, the following hardness amplification within NP/poly is a corollary to the proof of Theorem 2.18.

Corollary 2.43. *If there is a function $f : \{0,1\}^n \to \{0,1\}$ in* NP/poly *that is $1/\operatorname{poly}(n)$-hard for size $s(n)$, then there is a function $f' : \{0,1\}^m \to \{0,1\}$ in* NP/poly *that is $(1/2 - 1/s(\sqrt{m})^{\Omega(1)})$-hard for size $s(\sqrt{m})^{\Omega(1)}$.*

Now we return to hardness amplification within NP. First we note that, in our results, to amplify the hardness of $f : \{0,1\}^n \to \{0,1\}$ up to $1/2 - \epsilon$ it is only necessary that $Bias\,[f] \leq \epsilon^c$ for some universal constant c. The argument is standard and can be found, for example, in [Tre2].

Combining this observation with the above padding technique, O'Donnell constructs several candidate hard functions, one for each "guess" of the bias of the original hard function. He then combines them in a single function using a different input length for each candidate; this gives a function that is very hard on average for infinitely many input lengths. However, this approach, even in conjunction with derandomization and nondeterminism, cannot give better hardness than $1/2 - 1/n$. (Roughly speaking, if we want to amplify to $1/2 - \epsilon$, then we will have at least $1/\epsilon$ different candidates and thus the "hard" candidate may have input length $n \geq 1/\epsilon$, which means $1/2 - \epsilon \leq 1/2 - 1/n$.)

To what extent can we amplify the hardness of functions whose bias is *unknown*? Non-monotone hardness amplifications, such as Yao's Xor Lemma, work regardless of the bias of the original hard function. However, in the rest of this section we show that, for hardness amplifications that are *monotone* and *black-box*, this is impossible. In particular, we show that black-box monotone hardness amplifications cannot amplify the hardness beyond the bias of the original function.

We now formalize the notion of black-box monotone hardness amplification and then state our negative result.

Definition 2.44. *An oracle algorithm* $Amp : \{0,1\}^l \to \{0,1\}$ *is a* black-box β-bias $[\delta \mapsto (1/2-\epsilon)]$-hardness amplification for length n and size s *if for every* $f : \{0,1\}^n \to \{0,1\}$ *such that* $Bias\,[f] \leq \beta$ *and for every* $A : \{0,1\}^l \to \{0,1\}$ *such that*

$$\Pr[A(U_l) \neq Amp^f(U_l)] \leq 1/2 - \epsilon,$$

there is an oracle circuit C *of size at most* s *such that*

$$\Pr[C^A(U_n) \neq f(U_n)] \leq \delta.$$

Amp is monotone *if for every* x, $Amp^f(x)$ *is a monotone function of the truth table of* f.

Note that if Amp is as in Definition 2.44 and if f is δ-hard for size s' and $Bias\,[f] \leq \beta$, then Amp^f is $(1/2-\epsilon)$-hard for size s'/s: if there were a circuit A of size s'/s computing Amp^f with error probability at most $1/2-\epsilon$, then C^A would be a circuit of size $s \cdot (s'/s) = s'$ computing f with error probability at most δ, contradicting the hardness of f. The term "black box" refers to the fact that the definition requires this to hold for *every* f and A, regardless of whether or not f is in NP and A is a small circuit.

The following theorem shows that any *monotone* black-box hardness amplification up to $1/2 - \epsilon$ must start from functions of bias $\beta \approx \epsilon$.

Theorem 2.45. *For any constant* $\gamma > 0$, *if Amp is a monotone black-box* β*-bias* $[\delta \mapsto (1/2-\epsilon)]$*-hardness amplification for length* n *and size* $s \leq 2^{n/3}$ *such that* $1/2 - 4\epsilon > \delta + \gamma$, *then* $\beta \leq 8\epsilon + O(2^{-n})$.

The main ideas for proving this bound are the following: first we show that the above kind of hardness amplification satisfies certain coding-like properties. Roughly, *Amp* can be seen as a kind of list-decodable code where the distance property is guaranteed only for δ-distant messages with bias at most β (cf. [Tre2]). Then we show that monotone functions fail to satisfy these properties. The limitation we prove on monotone functions relies on the following corollary to the Kruskal-Katona theorem (see [And], Theorem 7.3.1).

Lemma 2.46. *Let $\mathcal{S} = \{S_1, \ldots, S_m\}$ be a collection of m subsets $S_i \subseteq \{1, \ldots, N\}$, where $|S_i| = t$. If $m \geq \binom{N-1}{t} = \left(1 - \frac{t}{N}\right)\binom{N}{t}$ then for every integer $t' < t$*

$$|\{S : |S| = t', S \subseteq S_i \text{ for some } i\}| \geq \binom{N-1}{t'} = \left(1 - \frac{t'}{N}\right)\binom{N}{t'}.$$

Let $\mathcal{F}_p$ be the uniform distribution on functions f whose truth-tables have relative Hamming weight exactly p, i.e. $\Pr_x[f(x) = 1] = p$. We use the above Lemma 2.46 to prove the following fact.

Lemma 2.47. *Let $A : \{0,1\}^l \to \{0,1\}$ be a an oracle function such that for every $x \in \{0,1\}^l$, $A^f(x)$ is a* monotone *function of the truth-table of $f : \{0,1\}^n \to \{0,1\}$. (For example, any monotone black-box hardness amplification Amp satisfies this condition.) Let τ be an integer multiple of $1/2^n$. Then there is $p \in \{1/2 - \tau, 1/2 + \tau\}$ such that:*

$$\mathrm{E}_{U_l}\left[\mathrm{Bias}_{F \leftarrow \mathcal{F}_p}[A^F(U_l)]\right] \geq \tau.$$

Proof. We show that for every fixed $x \in \{0,1\}^l$ there is a $p \in \{1/2 - \tau, 1/2 + \tau\}$ such that $\mathrm{Bias}_{F \leftarrow \mathcal{F}_p}\left[A^F(x)\right] \geq 2\tau$. The theorem then follows easily. Fix x. For every p define S_p as the set of functions f in $\mathcal{F}_p$ such that $A^f(x) = 0$. Note that

$$\mathrm{Bias}_{F \leftarrow \mathcal{F}_p}\left[A^F(x)\right] = \left|1 - \frac{2|S_p|}{|\mathcal{F}_p|}\right|.$$

If $|S_{1/2+\tau}| < (1/2 - \tau) \cdot |\mathcal{F}_{1/2+\tau}|$ then $\mathrm{Bias}_{F \leftarrow \mathcal{F}_{1/2+\tau}}\left[A^F(x)\right] > 2\tau$ and we are done. Otherwise,

$$|S_{1/2+\tau}| \geq (1/2 - \tau) \cdot |\mathcal{F}_{1/2+\tau}| = (1 - (1/2 + \tau)) \cdot \binom{2^n}{(1/2 + \tau)2^n}.$$

View the elements $f \in S_p$ as subsets of $\{1, \ldots, 2^n\}$ of size exactly $p2^n$ in the natural way; i.e. the subset associated with f is the set of inputs x such that $f(x) = 1$. Note that if $f' \subseteq f \subseteq \{1, \ldots, 2^n\}$ and $A^f(x) = 0$ then by the monotonicity of A, $A^{f'}(x) = 0$. Therefore, by Lemma 2.46,

$$|S_{1/2-\tau}| \geq (1 - (1/2 - \tau)) \cdot \binom{2^n}{(1/2 - \tau)2^n} = (1/2 + \tau) \cdot |\mathcal{F}_{1/2-\tau}|,$$

and so $\mathrm{Bias}_{F \leftarrow \mathcal{F}_{1/2-\tau}}\left[A^F(x)\right] \geq 2\tau$. $\square$

The following lemma captures the coding-like properties of monotone, black-box hardness amplifications — it shows that it is very unlikely that Amp^f for a "random f" will land in any fixed Hamming ball of radius $1/2-\epsilon$. For two functions f_1, f_2, let $Dist$ denote the relative Hamming distance of their truth tables, i.e. $Dist(f_1, f_2) := \Pr_x[f_1(x) \neq f_2(x)]$.

Lemma 2.48. *Let $\gamma > 0$ be any fixed constant. Let Amp be a monotone black-box 8ϵ-bias $[\delta \mapsto (1/2-\epsilon)]$-hardness amplification for length n and size $s \leq 2^{n/3}$, where $1/2 - 4\epsilon > \delta + \gamma$, and let $\epsilon > 0$ be an integer multiple of $1/2^n$. Then for both p in $\{1/2-4\epsilon, 1/2+4\epsilon\}$ and every function G:*

$$\Pr_{F \leftarrow \mathcal{F}_p}[Dist(G, Amp^F) \leq 1/2 - \epsilon] \leq 2^{-\Omega(2^n)}.$$

Proof. Let $N := 2^n$. For every function f of bias at most 8ϵ such that $Dist(G, Amp^f) \leq 1/2 - \epsilon$, there must exist a circuit of size s, with oracle access to G, that computes f with error at most δ. Therefore, since there are $2^{O(s \log s)}$ circuits of size s and no more than $2^{H(\delta)N}$ functions that are at distance at most δ from f, there are at most $2^{O(s\log s)}2^{H(\delta)N}$ such functions. Thus, when we restrict our attention to the $\binom{N}{pN}$ functions in $\mathcal{F}_p$ (for $p \in \{1/2-4\epsilon, 1/2+4\epsilon\}$), we have:

$$\begin{aligned}
\Pr_{F \leftarrow \mathcal{F}_p}[Dist(G, Amp^F) \leq 1/2 - \epsilon] &\leq \frac{2^{O(s\log s)} \cdot 2^{H(\delta)N}}{\binom{N}{pN}} \\
&\leq 2^{O(s\log s)} \cdot (N+1) \cdot 2^{(H(\delta)-H(p))N} \\
&\leq 2^{O(s\log s)} \cdot (N+1) \cdot 2^{(H(\delta)-H(1/2-4\epsilon))N} \\
&\leq 2^{-\Omega(N)},
\end{aligned}$$

where the second inequality follows from the fact that $\binom{N}{pN} \geq 2^{H(p)N}/(N+1)$,[6] and the last inequality uses the fact that $1/2 - 4\epsilon > \delta + \gamma$ implies $H(1/2-4\epsilon) > H(\delta) + \Omega(1)$. □

Proof of Theorem 2.45. We assume that ϵ is a positive integer multiple of $1/2^n$ and show that $\beta \leq 8\epsilon$. The theorem then follows for general ϵ by rounding it up to the next integer multiple of $1/2^n$. We show that $\beta = 8\epsilon$ is impossible, and therefore that $\beta < 8\epsilon$ (because any β'-bias hardness amplification is clearly also a β-bias hardness amplification for every $\beta \leq \beta'$).

Suppose, for the sake of contradiction, that $\beta = 8\epsilon$.

By Lemma 2.47, we may choose $p \in \{1/2-4\epsilon, 1/2+4\epsilon\}$ such that

$$\mathrm{E}_{U_l}\left[\mathrm{Bias}_{F \leftarrow \mathcal{F}_p}[Amp^F(U_l)]\right] \geq 4\epsilon.$$

[6] Actually, $\binom{N}{pN}$ is asymptotically $\Theta\left(2^{H(p)N}/\sqrt{N}\right)$, but for our purpose the easier estimate stated suffices.

Define the function $G(x) := \mathrm{Maj}_{F \leftarrow \mathcal{F}_p} Amp^F(x)$, and consider

$$\Pr_{U_l, F \leftarrow \mathcal{F}_p}[Amp^F(U_l) \neq G(U_l)]. \tag{2.7}$$

We have:

$$\begin{aligned}
&\Pr_{U_l, F \leftarrow \mathcal{F}_p}[Amp^F(U_l) \neq G(U_l)] \\
&= \mathrm{E}_{U_l}\left[\Pr_{F \leftarrow \mathcal{F}_p}[Amp^F(U_l) \neq G(U_l)]\right] \\
&= \mathrm{E}_{U_l}\left[\frac{1}{2} - \frac{\mathrm{Bias}_{F \leftarrow \mathcal{F}_p}[Amp^F(U_l)]}{2}\right] \qquad \text{(by def. of bias and } G\text{)} \\
&= \frac{1}{2} - \frac{\mathrm{E}_{U_l}\left[\mathrm{Bias}_{F \leftarrow \mathcal{F}_p}[Amp^F(U_l)]\right]}{2} \\
&\leq \frac{1}{2} - 2\epsilon. \qquad \text{(by the choice of } p\text{)}
\end{aligned}$$

On the other hand, Lemma 2.48 implies that quantity (2.7) is at least $1/2 - \epsilon - 2^{\Omega(2^n)}$ (note that the hypothesis of the lemma is satisfied by our assumption that $1/2 - 4\epsilon \geq \delta + \gamma$).

Combining the two bounds, we have that

$$1/2 - 2\epsilon \geq \Pr_{U_l, F \leftarrow \mathcal{F}_p}[Amp^F(U_l) \neq G(U_l)] \geq 1/2 - \epsilon - 2^{-\Omega(2^n)},$$

which is a contradiction for sufficiently large n (by the assumption that ϵ is a positive multiple of $1/2^n$). □

2.7.2 Nondeterminism is necessary

In this subsection we show that *deterministic*, monotone, non-adaptive black-box hardness amplifications cannot amplify hardness beyond $1/2 - 1/\mathrm{poly}(n)$. Thus, the use of *nondeterminism* in our results (Section 2.5.4) seems necessary. Note that most hardness amplifications, including the one in this chapter, are black-box and non-adaptive.

O'Donnell [O'D] proves that any monotone "direct product construction" (i.e. $f'(x_1, \ldots, x_k) = C(f(x_1), \ldots, f(x_k))$, as in Equation 2.1) cannot amplify to hardness better than $1/2 - 1/n$, assuming that the amplification works for all functions f (not necessarily in NP). We relax the assumption that the hardness amplification is a direct product construction (allowing any monotone nonadaptive oracle algorithm $f' = Amp^f$). On the other hand, we require that the reduction proving its correctness is also black-box (as formalized in Definition 2.44).

We prove our bound even for hardness amplifications that amplify only balanced functions (i.e. $\beta = 0$ in Definition 2.44).

Theorem 2.49. *For every constant $\delta < 1/2$, if Amp is a black-box 0-bias $[\delta \mapsto (1/2-\epsilon)]$-hardness amplification for length n and size $s \le 2^{n/3}$ such that for every x, $Amp^f(x)$ is a monotone function of $k \le 2^{n/3}$ values of f, then*

$$\epsilon \ge \Omega\left(\frac{\log^2 k}{k}\right).$$

The following lemma is similar to Lemma 2.48. The only difference is in considering functions F at distance η from f; this will correspond to perturbing the monotone amplification-function with noise having parameter η.

Lemma 2.50. *Let Amp be as in Definition 2.44 with $\beta = 0$ and $s \le 2^{n/3}$. Then for any constant $\delta < 1/2$ there is a constant $\eta < 1/2$ such that, for sufficiently large n, the following holds: If $f : \{0,1\}^n \to \{0,1\}$ is any fixed balanced function and $F : \{0,1\}^n \to \{0,1\}$ is a random balanced function such that $Dist(f,F) = \eta$, then*

$$\Pr_F[Dist(Amp^f, Amp^F) \le 1/2 - \epsilon] \le \epsilon.$$

Proof. Let $N := 2^n$. It is easy to see that F is uniform on a set of size $\binom{N/2}{\eta N/2}^2$. The rest of the proof is like the proof of Lemma 2.48:

$$\begin{aligned}
\Pr[Dist(Amp^f, Amp^F) \le 1/2 - \epsilon] &\le \frac{2^{O(s\log s)} 2^{H(\delta)N}}{\binom{N/2}{\eta N/2}^2} \\
&\le 2^{O(s\log s)} \cdot (N/2+1)^2 \cdot 2^{(H(\delta)-H(\eta))N} \\
&\le \epsilon,
\end{aligned}$$

where the last inequality holds for a suitable choice of $\eta < 1/2$, using the fact that $\delta < 1/2$ is a constant and that $s \le 2^{n/3}$. □

Proof of Theorem 2.49. Let η be the constant in Lemma 2.50. The idea is to consider

$$\Pr_{U_l, F, F'}[Amp^F(U_l) \ne Amp^{F'}(U_l)], \tag{2.8}$$

where F is a random balanced function and F' is a random balanced function such that $Dist(F, F') = \eta$.

By the above lemma, the probability (2.8) is at least $1/2 - 2\epsilon$.

On the other hand, for every fixed x, $Amp^F(x)$ is a monotone function depending only on k bits of the truth-table of the function F. Since k is small compared to 2^n, the distribution (F, F') induces on the input of $Amp^F(x)$ a distribution very close to $(U_k, U_k \oplus \mu)$, where μ is a noise vector with parameter η. Specifically, it can be verified that the statistical difference between these two distributions is at most $O(k^2/(\eta 2^n))$. Because this value is dominated by $\log^2 k/k$ when $k \le 2^{n/3}$, and because $Amp^F(x)$

is a *monotone* function of k bits, we may apply Theorem 2.17 to conclude that the probability (2.8) is at most $1/2 - O(\log^2 k/k)$.

Combining the two bounds, we have that $1/2 - O(\log^2 k/k) \geq 1/2 - 2\epsilon$ and the results follows. $\square$

Chapter 3

Worst-case hardness amplification

3.1 Introduction

In the previous Chapter 2 we studied hardness amplification of functions $f : \{0,1\}^n \to \{0,1\}$ from hardness $\delta = 1/\operatorname{poly}(n)$ to hardness close to 1/2. In particular, we showed that this hardness amplification can be accomplished within NP with nearly optimal parameters (when starting from balanced functions). We did not discuss amplification from hardness $\delta = 1/n^{\omega(1)}$. This is the topic of this chapter.

A celebrated result, usually credited to Babai et al. [BFNW], shows that it is possible to amplify worst-case hardness (i.e., $\delta = 2^{-n}$) within $\mathsf{E} := \mathsf{Time}\left(2^{\mathsf{O(n)}}\right)$.

Theorem 3.1 ([BFNW])**.** *If there is a function $f : \{0,1\}^n \to \{0,1\}$ in E that is worst-case hard for circuits of size $s(n) = n^{\omega(1)}$, then there is a function $f' : \{0,1\}^{O(n)} \to \{0,1\}$ in E that is $(1/\operatorname{poly}(n))$-hard for for circuits of size $s'(n) = n^{\omega(1)}$.*

The idea in the proof of Theorem 3.1 is to take f' to be a small degree, multi-variate polynomial extension of f. The random self-reducibility of low-degree polynomials then implies that f' has the required hardness.

Can we prove a similar result for lower complexity classes, such as NP? As pointed out in the Introduction in Section 1.1, establishing a connection between worst-case hardness and average-case hardness for NP is a central open problem in complexity theory and cryptography.

In this chapter we prove two results that show that such a connection for NP *cannot* be proved using black-box techniques. More generally, we prove that black-box techniques cannot prove such a connection within the polynomial-time hierarchy $\mathsf{PH} \supseteq \mathsf{NP}$.

While we present our negative results only for worst-case hardness amplification, i.e. $\delta = 2^{-n}$, they can be generalized to any hardness $\delta = 1/n^{\omega(1)}$. (The generalization of our first result, Theorem 3.4, is addressed in [LTW].) We fix $\delta = 2^{-n}$ for simplicity of exposition and because this seems to be the most interesting setting of parameters.

3.1.1 Organization

This chapter is organized as follows. In Section 3.2 we discuss some preliminaries. In Section 3.3 we prove that there is no black-box worst-case hardness amplification within PH. In Section 3.4 we consider a more relaxed model for black-box hardness amplification, i.e. *mildly* black-box hardness amplification, and, speaking informally, we show that even these do not exist within PH.

3.2 Preliminaries

The two main results in this chapter rely on a connection between alternating time and constant-depth circuits which originated in the seminal paper of Furst et al. [FSS] (see also [Hås]). Informally, the connection is that any oracle computation performed in alternating time with d quantifiers can be carried out by an exponential-size constant-depth circuit of depth $d+1$ that takes the truth-table of the oracle as part of the input. We now formally describe this connection. First, let us briefly recall some standard definitions of alternating machines and constant-depth circuits.

An *alternating machine* is a non-deterministic machine with two types of configurations, $\exists$ and $\forall$. An $\exists$ configuration is accepting if *at least one* of the configurations reachable in one step from it are accepting. A $\forall$ configuration is accepting if *all* of the configurations reachable in one step from it are accepting. We say that the machine uses $d-1$ alternations (or d quantifiers) if $d-1$ is the maximum number of times the machine switches between the above two types of configurations over all computation paths (see [Pap]). *Constant-depth circuits* consist of And and Or gates with unbounded fan-in. Circuits take both input variables and their negations as input. The *depth* of a circuit is the longest path from any input to the output. The *size* of a circuit is the number of gates in it.

We have the following result.

Theorem 3.2 ([FSS]). *Let $M^f : \{0,1\}^{n'} \to \{0,1\}$ be an oracle machine, where $f : \{0,1\}^n \to \{0,1\}$. Suppose that M uses $d-1$ alternations and runs in time t for every $x \in \{0,1\}^{n'}$ and every oracle f.*

Then, for every $x \in \{0,1\}^{n'}$, there is a constant-depth circuit C_x of depth $d+1$ and size $2^{O(d \cdot t)}$ that given the truth-table of f computes $M^f(x)$.

For completeness, we include a proof of Theorem 3.2 in Appendix B.

3.3 No black-box hardness amplification within PH

In this section we prove that there is no black-box worst-case hardness amplification within the polynomial-time hierarchy PH (and thus in particular within NP).

Overview of techniques: The outline of our proof is as follows. First we show that every black-box hardness amplification gives rise to a 'good' *list-decodable code*, where messages are the truth-tables of the worst-case hard functions and codewords are the truth-tables of the amplified, average-case hard functions. By the connection between alternating computation and constant-depth circuits reviewed in Section 3.2, this code (i.e., the map from messages to codewords) can be computed by a constant-depth circuit. We then show that constant-depth circuits cannot compute 'good' list-decodable codes, which proves the result.

To show that constant-depth circuits cannot compute 'good' list-decodable codes, we use the notion of *noise sensitivity*, which is a measure of how likely the output of a function is to change when the input is perturbed with random noise. (Noise sensitivity was used throughout Chapter 2). We show that 'good' list-decodable codes are very sensitive to noise. Since constant-depth circuits are not, we get our result.

We remark that the above approach is very similar to the one used in Chapter 2 to prove limitations of monotone black-box hardness amplifications (sections 2.7.1 and 2.7.2), though in fact the results in this section preceded those in Chapter 2.

We now proceed to turn the above sketch into a formal proof. First, let us formalize our notion of black-box hardness amplification.

Definition 3.3. *An oracle algorithm* $Amp : \{0,1\}^{n'} \to \{0,1\}$ *is a* $(2^{-n} \to \delta)$-black-box hardness amplification for size s and length n *if for every* $f : \{0,1\}^n \to \{0,1\}$ *and for every* $A : \{0,1\}^{n'} \to \{0,1\}$ *such that*

$$\Pr[A(U_{n'}) \neq Amp^f(U_{n'})] \leq \delta,$$

there is an oracle circuit C of size at most s such that $C^A(x) = f(x)$ for every $x \in \{0,1\}^n$.

The idea behind this definition is that if Amp is a $(2^{-n} \to \delta)$-black-box hardness amplification for size s and length n, then for every function $f : \{0,1\}^n \to \{0,1\}$ that is worst-case hard for circuits of size s', Amp^f is δ-hard for circuits of size s'/s.

In the notation of Definition 3.3, the hardness amplification in Theorem 3.1 is a $(2^{-n} \to 1/\operatorname{poly}(n))$-black-box hardness amplification for size $\operatorname{poly}(n)$ and length n. This hardness amplification is computable in time exponential in the input length n of the worst-case hard function f. In this section we suggest that this running time cannot be significantly improved, and in particular we prove the following negative result about black-box worst-case hardness amplification.

Theorem 3.4 (No worst-case hardness amplification within PH). *Suppose Amp is a $(2^{-n} \to \delta)$ black-box hardness amplification for length n and size s, and suppose that Amp uses $d-1$ alternations and runs in time $t \geq d$. Then:*

$$t \geq \left(\frac{2^n \delta}{s \log s}\right)^{\Omega(1/d)}.$$

In particular, for any constants $c > 0, \epsilon < 1$, any $(2^{-n} \to 1/n^c)$-black-box worst-case hardness amplification for length n and size $2^{\epsilon \cdot n}$ that uses $d-1$ alternations must run in time $2^{\Omega(n/d)}$.

We note that, if we start with a worst-case hard function $f : \{0,1\}^n \to \{0,1\}$, Theorem 3.4 says that black-box techniques cannot construct a $(1/\operatorname{poly}(n))$-hard function in PH. Specifically, for the amplified function Amp^f to be in PH we would need Amp to make only a constant number of alternations (i.e., $d = O(1)$) and run in time $t = \operatorname{poly}(n')$. Theorem 3.4 then implies that $n' \geq 2^{\Omega(n)}$. Since the hardness s is measured in terms of the input length of the original function f, this gives nothing when starting from a lower bound for circuits of size s whenever $s = 2^{o(n)}$. We elaborate more on the tightness of Theorem 3.4 in Section 3.3.1.

A technical remark on the input length of ***Amp*****:** It should be noted that in previous hardness amplifications (e.g., Theorem 3.1) the input length increases only by a constant factor, i.e. $Amp^f : \{0,1\}^{O(n)} \to \{0,1\}$. Our negative result applies regardless of this. More specifically, our negative result is only expressed in terms of the running time of the hardness amplification, rather than its input length. Jumping ahead, this will correspond to a negative result (Theorem 3.7) for computing codes that is independent from the block length of the code.

We now proceed with the proof of Theorem 3.4. First, let us give the definition of list-decodable codes:

Definition 3.5 (List-decodable code)**.** *A code $E : \{0,1\}^N \to \{0,1\}^{N'}$ is (δ,ρ)-*list-decodable *if for every $r \in \{0,1\}^{N'}$ we have:*

$$\left|\left\{x \in \{0,1\}^N : \Delta(r, E(x)) \leq \delta\right\}\right| \leq \rho$$

where Δ is the relative Hamming distance: $\Delta(r,y) := \Pr_i[r_i \neq y_i]$. We refer to $x \in \{0,1\}^n$ as messages *and to $E(x), x \in \{0,1\}^n$, as* codewords*.*

Let Amp be a black-box worst-case hardness amplification. The following lemma, pointed out in [TV] and also implicit in [Tre1, STV], states that if we consider the truth table of a function f as a message and the truth table of Amp^f as a codeword, then Amp can be seen as an encoding algorithm.

Lemma 3.6 (Black-box hardness amplification gives rise to a list-decodable code)**.** *Let $Amp : \{0,1\}^{n'} \to \{0,1\}$ be a $(2^{-n} \to \delta)$-black-box hardness amplification for size s and length n. Then $E : \{0,1\}^{2^n} \to \{0,1\}^{2^{n'}}$ defined as $Enc(f) := Amp^f$ is $(\delta, 2^{O(s \cdot \log s)})$-list-decodable.*

Proof. Consider $A \in \{0,1\}^{\bar{n}}$. By definition of hardness amplification, for every f such that $\Pr_{x \in \{0,1\}^{n'}}[A(x) \neq Amp^f(x)] < \delta$, there is an oracle circuit C of size at most s such that $C^A(x) = f(x)$ for every x. Therefore the number of such codewords is bounded by the number of oracle circuits. Noting that there are at most $2^{O(s\cdot\log s)}$ oracle circuits of size at most s, and that $\Pr_{x\in\{0,1\}^{\bar{n}}}[A(x) \neq Amp^f(x)] = \Delta(A, Amp^f)$, completes the proof. □

The following theorem states that constant-depth circuits cannot compute 'good' list-decodable codes.

Theorem 3.7. *Let $E : \{0,1\}^N \to \{0,1\}^{N'}$ be a $(\delta, 2^m)$-list-decodable code, with $m \leq N/2$. If each bit of E can be computed by a circuit of size g and depth d, then*

$$\log^{d-1} g \geq \Omega\left(\frac{N\delta}{m}\right).$$

Before proving the above Theorem 3.7, we note that, using the connection between list-decodable codes and black-box hardness amplification (Lemma 3.6), it implies our main Theorem 3.4.

Proof of Theorem 3.4. By Theorem 3.2, for every fixed $x \in \{0,1\}^n$ the oracle algorithm $Amp^f(x)$ can be turned into an equivalent circuit of depth $d+1$ and size $2^{O(d\cdot t)}$, where we view the truth-table of the oracle f as the input. The result then follows from Lemma 3.6 and Theorem 3.7. □

To prove Theorem 3.7 we make use of the following fact about low noise sensitivity of constant-depth circuits, which we deduce combining results in [KKL, Bop, O'D], and ultimately relies on Håstad's switching lemma [Hås]. We denote bitwise Xor by $\oplus$.

Lemma 3.8. *Let $C : \{0,1\}^n \to \{0,1\}$ be a circuit of size g and depth d. Let $X \in \{0,1\}^n$ be a random input, and $\Psi \in \{0,1\}^n$ a random noise vector where each bit is '1' independently with probability $\delta < 1/2$. Then:*

$$\Pr_{X,\Psi}\Big[C(X) \neq C(X \oplus \Psi)\Big] \leq O(\delta \log^{d-1} g).$$

The proof of Lemma 3.8 requires a detour into Fourier analysis and therefore we defer it to Section 3.5.

Proof of Theorem 3.7. Let $C_i(x)$ denote the i-th output bit of $C(x)$. Let $\Psi \in \{0,1\}^N$ be a random noise vector where each bit is '1' independently with probability m/N. Let X be chosen at random in $\{0,1\}^N$.

The idea in the proof is to consider the quantity

$$\Pr_{i,X,\Psi}\Big[C_i(X) \neq C_i(X \oplus \Psi)\Big]$$

and to bound it using (1) the assumption that C is $(\delta, 2^m)$-list-decodable and (2) the low noise sensitivity of constant-depth circuits (Lemma 3.8).

For every fixed $x \in \{0,1\}^N$, the list-decodability assumption tells us that there are at most 2^m messages whose codewords are at distance at most δ from $C(x)$. Fix any such message $a \in \{0,1\}^N$. The probability that $x \oplus \Psi$ is equal to this message a is

$$\Pr_{\Psi}[x \oplus \Psi = a] \leq \Pr_{\Psi}[x \oplus \Psi = x] \leq (1 - m/N)^N \leq e^{-m}, \tag{3.1}$$

where the first inequality holds because $m/n \leq 1/2$.

Therefore, by a union bound:

$$\Pr_{X,\Psi}\Big[\Delta(C(X), C(X \oplus \Psi)) \leq \delta\Big] \leq 2^m \cdot e^{-m} = 2^{-\Omega(m)}.$$

Therefore:

$$\begin{aligned}
&\Pr_{i,X,\Psi}\Big[C_i(X) \neq C_i(X \oplus \Psi)\Big] \\
\geq\ & \Pr_{i,X,\Psi}\Big[C_i(X) \neq C_i(X \oplus \Psi)\Big|\Delta(C(X), C(X \oplus \Psi)) > \delta\Big] \\
& \cdot \Pr_{X,\Psi}\Big[\Delta(C(X), C(X \oplus \Psi)) > \delta\Big] \\
\geq\ & \delta \cdot \big(1 - 2^{\Omega(m)}\big) \geq \delta/2. && (3.2)
\end{aligned}$$

On the other hand, by Lemma 3.8 we have

$$\Pr_{i,X,\Psi}\Big[C_i(X) \neq C_i(X \oplus \Psi)\Big] \leq m \cdot \frac{O(\log^{d-1} g)}{N}. \tag{3.3}$$

The theorem follows by putting together the bounds (3.2) and (3.3). □

3.3.1 Tightness

In this section we discuss in what sense Theorem 3.4 is tight. Recall that it established the following tradeoff for a $(2^{-n} \to 1/n^{O(1)})$ black-box hardness amplification for length n and size s computable with $d-1$ alternations in time $t \geq d$:

$$t \geq \left(\frac{2^n}{n \cdot s \cdot \log s}\right)^{\Omega(1/d)}.$$

Suppose we have a black-box worst-case to average-case hardness amplification that achieves $n' = \mathrm{poly}(n)$, as achieved by Theorem 3.1. If we insist that *Amp* is computable in the polynomial-time hierarchy PH, i.e. that runs in time $t = \mathrm{poly}(n') = \mathrm{poly}(n)$ with a constant number of alternations, then the above bound implies that $s \geq 2^n/n^{O(1)}$, which means that the function $f : \{0,1\}^n \to \{0,1\}$ we start with must be hard for circuits of size at least $s = 2^n/n^{O(1)}$. However, the next easy proposition shows that for such big sizes average-case and worst-case hardness *are equivalent*! Consequently, under such an assumption no worst-case to average-case hardness amplification is needed.

Proposition 3.9. *If $f : \{0,1\}^n \to \{0,1\}$ is worst-case hard for circuits of size $(2^n/n^c)$ then f is $(1/n^{c+O(1)})$-hard for circuits of size $(2^n/n^{c+O(1)})$.*

Proof. Suppose not: let C be a circuit of size at most $p \cdot 2^n$ such that

$$\Pr[C(U_n) \neq f(U_n)] < p.$$

Then there are at most $p \cdot 2^n$ inputs x such that $C(x) \neq f(x)$. By hardwiring all these inputs in the circuit, we can construct a circuit C' of size at most $\mathrm{poly}(n) \cdot p \cdot 2^n$ such that, given x, decides whether $C(x) \neq f(x)$.

Combining C and C' with a Xor we obtain a circuit of size $p \cdot 2^n + \mathrm{poly}(n) \cdot p \cdot 2^n$ computing f everywhere. We reach a contradiction for $p = 1/n^{c+O(1)}$. □

3.4 No mildly black-box hardness amplification in PH

In the previous Section 3.3 we proved a negative result regarding black-box hardness amplification. As pointed out in Section 1.3, this covers all known hardness amplifications against circuits. However, there are instances in complexity theory and cryptography when non-black-box techniques have been successfully used, and thus it is natural to wonder if non-black-techniques could be used for worst-case hardness amplification, or if one can prove meaningful negative results about more relaxed notions of "black-box." In this section we present a result of the latter kind. Let us first discuss a relaxation of the notion of black-box hardness amplification.

Mildly black-box hardness amplification: Our previous notions of black-box hardness amplification (definitions 3.3 and 2.44) was black-box *both* in the use of the worst-case hard function f *and* in the 'proof of correctness.' Namely it asked for an 'efficient' algorithm Amp such that for *every* function f and *every* adversary A, if A computes Amp^f well on average then there is a small circuit D such that D^A computes f in the worst-case. Recall from Section 3.3 that the rationale behind this definition is that if f is worst-case hard then D^A cannot be a small circuit. However, since D *is* a small circuit, we have that A cannot be a small circuit, and hence Amp^f is average-case hard. A natural relaxation of this notion is to drop the requirement that Amp is black-box in *both* the worst-case hard function and the 'proof of correctness.'

Bogdanov and Trevisan [BT] prove a negative result where Amp is black-box only in the proof of correctness. Specifically, building on [FF], they show that every hardness amplification within NP such that its proof of correctness is black-box and D makes non-adaptive queries to A implies a collapse of the polynomial-time hierarchy PH, and therefore such a hardness amplification is unlikely to exist. Their result is powerful in that it allows for the hardness amplification to depend on the particular worst-case hard function in NP (e.g., SAT). However, it only applies when the proof of correctness is non-adaptive.

In this section we present a negative result on hardness amplifications that is black-box only in the use of f. Let us pause a moment to reflect on what kind of negative result one can expect. If we put no requirements on the proof of correctness, we are simply studying hardness amplifications Amp such that Amp^f is average-case hard whenever f is worst-case hard. But if PH contains average-case hard functions than such hardness amplifications do exist: simply ignore the oracle and compute the average-case hard function known to exist in PH! Thus, if we believe, as most researchers do, that PH contains average-case hard functions, we cannot rule out such pathological hardness amplifications. The best we can hope to show is that such pathological hardness amplifications are *the only possibility.* That is exactly what we show: we show that exhibiting a hardness amplification that is black-box only in the use of f is equivalent to exhibiting an average-case hard function in PH.

Theorem 3.10. *Let $s(n)$ be a function such that $s(n) = 2^n/n^{\omega(1)}$. Suppose that there is a constant a and an oracle algorithm Amp such that for every $f : \{0,1\}^n \to \{0,1\}$ that is worst-case hard for size $s(n)$, $Amp^f : \{0,1\}^{n^a} \to \{0,1\}$ is .3-hard for size $s'(n)$. Suppose moreover that Amp runs in polynomial time and uses a constant number of alternations.*

Then there is a constant b and a function $f' \in \mathsf{PH}$ such that $f' : \{0,1\}^{n^b} \to \{0,1\}$ is .1-hard for size $s'(n)$.

Before discussing the ideas behind the proof of Theorem 3.10 we make some remarks.

Remarks on Theorem 3.10: (1) We required (in the statement of the theorem) that for every c and for sufficiently large n, $s(n) \leq 2^n/n^c$. This is because for $s \geq 2^n/n^c$ for some fixed constant c, the oracle is already $1/\operatorname{poly}(n)$-hard by Proposition 3.9. In this case, one can construct a function in PH that is .3-hard by using, say, the Xor Lemma (see Section 2.1). (2) Similar results hold for hardness amplifications $Amp^f : \{0,1\}^{l(n)} \to \{0,1\}$ running in time $t(n)$ with a constant number of alternations, for a 'wide' range of parameters $l(n), t(n)$. Here we set $l(n) = \operatorname{poly}(n)$ and $t(n) = \operatorname{poly}(n)$ for simplicity of exposition.

Techniques: We now sketch the main ideas in the proof of Theorem 3.10. The idea is to choose the oracle function f at random from a suitable distribution of functions $\tilde{F}$ such that (1) $\tilde{F}$ is worst-case hard with high probability, but (2) $Amp^{\tilde{F}}$ makes 'little' use of the oracle $\tilde{F}$. By (1) and the assumption of the theorem we have that $Amp^{\tilde{F}}$ is average-case hard, but by (2) we can dispense with the oracle and construct an average-case hard function f' from scratch, thus proving the theorem.

We now explain how we construct the distribution $\tilde{F}$. The idea is to fix some of the bits in the truth table of $\tilde{F}$, and choose the other at random. The bits to be fixed are chosen applying a *random restriction* to the truth table of $\tilde{F}$. Now, (1) follows by the fact that we only fix few bits, while (2) follows by the fact that the oracle PH computation can be simulated by constant-depth circuits (Theorem 3.2) and such circuits have low noise sensitivity (Lemma 3.8). More precisely, (2) follows by the fact that a constant-depth circuit becomes very biased after we apply a restriction to it, and so the choice of the other bits does not affect its outcome too much.

An idea now would be to include the fixing of the bits in the input to the function h, but the problem is that to describe this fixing of bits requires length of the order of the truth table of the oracle f, i.e. 2^n, while we need the input length of h to be polynomial in n (since the circuit size s' in Theorem 3.10 is relative to the input length of f). To overcome this problem, we *derandomize* the random restrictions. I.e., we create a pseudorandom distribution on restrictions that can be generated using only $\mathrm{poly}(n)$ random bits, yet still with high probability satisfies (1) and (2). Now, the function h takes σ as part of the input, where σ is of size $\mathrm{poly}(n)$ and is used to generate a pseudorandom restriction. The input length of h is polynomial in n and the theorem can be proved. This pseudorandom distribution on restrictions is obtained using Nisan's unconditional pseudorandom generator against constant-depth circuits [Nis1]. The challenges of course are showing that after this derandomization (1) and (2) still hold. In particular, for (2) we show that a constant depth circuit becomes very biased even after applying a pseudorandom restriction. We remark that the idea of using Nisan's generator to derandomize restrictions already appeared in [CSS] in the context of resource-bounded measure theory. Our use of it in hardness amplification and our technique based on noise sensitivity are new.

Before presenting the formal proof of Theorem 3.10, we need to discuss some notation about restrictions.

Restrictions: A *restriction* ρ on t bits is an element of $\{0,1,*\}^t$, where we think of the *'s as values yet to be chosen. For $x \in \{0,1\}^t$ we denote by $x^\rho \in \{0,1\}^t$ the string obtained from ρ by substituting the *'s with the corresponding bits of x. Note x^ρ only depends on the bits of x corresponding to * in ρ. We consider restrictions on 2^n bits, which will be convenient to view as functions $\rho : \{0,1\}^n \to \{0,1,*\}$. When $\rho : \{0,1\}^n \to \{0,1,*\}$ we think of $\rho(i)$ as a partial assignment to the output $f(i)$ of some function $f : \{0,1\}^n \to \{0,1\}$. The following is a key definition: for a function $f : \{0,1\}^n \to \{0,1\}$ we denote by $f^\rho : \{0,1\}^n \to \{0,1\}$ the function defined by

$$f^\rho(x) := f(x)^{\rho(x)} = \begin{cases} f(x) & \text{if } \rho(x) = *, \\ \rho(x) & \text{otherwise.} \end{cases}$$

For a fixed restriction ρ, a key random variable we will look at is F^ρ, where $F : \{0,1\}^n \to \{0,1\}$ is a uniform random function. Note that F^ρ can be seen as the distribution on functions whose truth table is obtained starting from the truth table of ρ and replacing each * with a uniform and independent random bit. The standard [FSS] distribution on restrictions R_δ is the one where each symbol in the restriction is independently * with probability δ and otherwise it is a uniform and independent random bit. When we say that $\rho : \{0,1\}^n \to \{0,1,*\}$ is random in R_δ we mean that each of the 2^n symbols in the truth table of ρ is independently * with probability δ and otherwise it is a uniform and independent random bit.

We would like to point out some differences between our notation for restrictions (above) and the notation more commonly used in literature (e.g. [FSS, Hås]) and also used in Chapter 4. The notation commonly used in literature is the following: for a function $f : \{0,1\}^n \to \{0,1\}$ and a restriction $\rho : \{1,2,\ldots,n\} \to \{0,1,*\}$ with s stars (i.e. exactly s distinct indexes i such that $\rho(i) = *$) one denotes by $f|_\rho : \{0,1\}^s \to \{0,1\}$ the function obtained by 'restricting' f on the s bits mapped to $*$ by ρ, where the other bits are fixed as prescribed by ρ. Thus, in the standard notation one restricts the *input* of the function, while in our notation we restrict the *truth-table* of the function. Our notation makes sense in our setting because we will use restrictions to argue about an exponential-size constant-depth circuit that computes Amp^f when given the truth-table of f as input. To avoid confusion we never use the notation $f|_\rho$ in this chapter.

3.4.1 The proof

To prove Theorem 3.10 we build a certain pseudorandom distribution on restrictions. This is the main technical lemma of this section.

Lemma 3.11. *For every constant c, there is a distribution $\tilde{R}^c$ on restrictions $\rho : \{0,1\}^n \to \{0,1,*\}$ such that:*

(1) Every $\rho \in \tilde{R}^c$ is described by $\mathrm{poly}(n)$ *bits σ. We denote by ρ_σ the restriction described by σ. For random σ, we have that ρ_σ is random in $\tilde{R}^c$. There is a polynomial time algorithm such that given σ and $x \in \{0,1\}^n$, computes $\rho_\sigma(x)$.*

(2) For every circuit C of size 2^{n^c} and depth c on $N := 2^n$ bits, with probability $1 - o(1)$ over $\rho_\sigma \leftarrow \tilde{R}^c$,

$$\mathrm{Bias}_{U_N}[C(U_N{}^{\rho_\sigma})] \geq 1 - o(1).$$

(Where the Bias *of a $0-1$ random variable X is* $|\Pr[X=0] - \Pr[X=1]|$*.)*

(3) There is a constant d such that with probability $1 - o(1)$ over $\rho_\sigma \leftarrow \tilde{R}^c$, ρ_σ has at least $2^n/n^d$ $*$'s.

We now assume the above Lemma and prove Theorem 3.10.

Proof of Theorem 3.10. By Theorem 3.2, the oracle algorithm *Amp* can be turned into an exponential size constant-depth circuit whose input is the truth table of the oracle f. In particular, let c be such that, for every x, $Amp^f(x)$ has depth c and size 2^{n^c} when turned into a constant depth circuit whose input is the truth table of f. Consider the distribution on restrictions $\tilde{R}^c$ whose existence is guaranteed by Lemma 3.11, let F denote a uniform random function $F : \{0,1\}^n \to \{0,1\}$. Consider $Amp^{F^{\rho_\sigma}}$, where $\rho_\sigma \leftarrow \tilde{R}^c$. We need a couple of lemmas.

Lemma 3.12. *With probability* $1 - o(1)$ *over* $\rho_\sigma \leftarrow \tilde{R}^c$ *and* F, $F^{\rho_\sigma} : \{0,1\}^n \to \{0,1\}$ *is worst-case hard for size* $s(n)$. *In particular, with probability* $1 - o(1)$ *over* $\rho_\sigma \leftarrow \tilde{R}^c$ *and* F, $Amp^{F^{\rho_\sigma}} : \{0,1\}^{n^a} \to \{0,1\}$ *is .3-hard for size* $s'(n)$.

Lemma 3.13. *There is a* PH *machine* A' *such that given* σ *and an input* x *computes the most likely value of* $Amp^{F^{\rho_\sigma}}(x)$ *over the choice of* F, *whenever* $\text{Bias}_F[Amp^{F^{\rho_\sigma}}(x)] \geq .2$. *I.e., if* $\Pr_F[Amp^{F^{\rho_\sigma}}(x) = 1] \geq .6$ *then* $A'(\sigma, x) = 1$, *and if* $\Pr_F[Amp^{F^{\rho_\sigma}}(x) = 0] \geq .6$ *then* $A'(\sigma, x) = 0$.

Now for the proof of the theorem. By Lemma 3.11, for every x, w.p. $1 - o(1)$ over σ, $\text{Bias}_F[Amp^{F^{\rho_\sigma}}(x)] \geq 1 - o(1)$. This holds because, for fixed x, $Amp^f(x)$ is a constant depth function of the truth table of f. Let A' be the oracle PH machine in Lemma 3.13. By Lemma 3.13 we have:

$$\Pr_{x,\sigma,F}[A'(\sigma, x) \neq Amp^{F^{\rho_\sigma}}(x)] \leq o(1) + o(1) = o(1). \tag{3.4}$$

Thus there is a function $\eta(n) = o(1)$ such that

$$\Pr_{\sigma,F}[\Delta(A'(\sigma, .), Amp^{F^{\rho_\sigma}}) \geq \eta(n)] \leq \eta(n). \tag{3.5}$$

Where $\Delta(f, f')$ denotes the relative Hamming distance of the truth tables of $f, f' : \{0,1\}^n \to \{0,1\}$. Thus:

$$\begin{aligned}
&\Pr_{\sigma,F}[A'(\sigma, .) \text{is not .2-hard for size } s'(n)] \\
\leq\ & \Pr_{\sigma,F}\left[\Delta(A'(\sigma, .), Amp^{F^{\rho_\sigma}}) > .1 \text{ or } Amp^{F^{\rho_\sigma}} \text{ is not .3-hard for size } s'(n)\right] \\
\leq\ & o(1) + o(1) \quad \text{(By Inequality (3.5) and Lemma 3.12)} \\
\leq\ & o(1),
\end{aligned}$$

where we use the fact that if $A'(\sigma, .)$ is at relative Hamming distance at most .1 from $Amp^{F^{\rho_\sigma}}$ that is .3-hard, then $A'(\sigma, .)$ must be .2-hard. This holds because otherwise the circuit computing $A'(\sigma, .)$ with error less than .2 would also compute $Amp^{F^{\rho_\sigma}}$ with error less than $.2 + .1 = .3$.

Now, since with high probability over σ we have that $A'(\sigma, .)$ is .2-hard for size $s'(n)$, we have that every circuit of size $s'(n)$ fails to compute A' on at least a $.2 \cdot (1 - o(1)) > .1$ fraction of inputs, and thus A' is .1-hard for size $s'(n)$.

To finish the proof note that A' is in PH and that it has input length n^b for some constant b. □

We now discuss Lemmas 3.12 and 3.13.

Proof of Lemma 3.12. This is a simple counting argument. By Lemma 3.11 there is a constant d such that ρ has at least $2^n/n^d$ $*$'s with high probability. Whenever this happens, F^ρ is uniform (over the choice of F) on a set of $2^{2^n/n^d}$ functions. There are at most $2^{O(S\cdot\log S)}$ circuits of size S. By assumption, for sufficiently large n, $S(n) \le 2^n/n^{d+2}$. This means in particular that for sufficiently large n, $O(S\cdot\log S) \le (2^n/n^d)/2$. Therefore, for sufficiently large n, the probability (over F) that F^ρ is not worst-case hard for size S is at most $2^{O(S(n)\cdot\log S(n))-2^n/n^d} \le 2^{2^n/(2\cdot n^d)} = o(1)$. □

Lemma 3.13 was essentially proved by Nisan in [Nis1] (see also [NW]) (In [Nis1, NW] the lemma is stated as "almost-PH=PH"). The idea is to use Nisan's generator (described below in Theorem 3.14) to replace the random oracle F with a pseudorandom oracle that is generated using only $\mathrm{poly}(n)$ random bits (as opposed to 2^n). These $\mathrm{poly}(n)$ random bits are then replaced by a constant number of alternations just like in the proof that $\mathsf{BPP} \subseteq \mathsf{PH}$ (which is discussed in detail in Chapter 5).

3.4.2 Pseudorandom restrictions

In this section we prove Lemma 3.11. A key tool is Nisan's pseudorandom generator against constant depth circuits.

Theorem 3.14 ([Nis1]). *For every constant c and every n there is a generator Nis : $\{0,1\}^{\mathrm{poly}\log n} \to \{0,1\}^n$ such that (1) given x and $i \le n$ we can compute the i-th bit of $Nis(x)$ in time* $\mathrm{poly}\log(n)$ *and (2) Nis is* $1/n$-pseudorandom *for circuits of size n and depth c. That is, for every circuit $C : \{0,1\}^n \to \{0,1\}$ of size n and depth c:*

$$\Big|\Pr[C(Nis(U_{\mathrm{poly}\log n})) = 1] - \Pr[C(U_n) = 1]\Big| \le 1/n.$$

Proof of Lemma 3.11. Let $\delta := 1/n^{c^2}$. First, let us see that R_δ satisfies Items (2) and (3) in Lemma 3.11. Using our previous result on noise-sensitivity of constant-depth circuits (Lemma 3.8), we have that for every circuit $C : \{0,1\}^t \to \{0,1\}$ of size 2^{n^c} and depth c on t bits:

$$\Pr_{\rho\in R_\delta, U_t, U'_t}[C(U_t{}^\rho) \ne C(U'_t{}^\rho)] = \Pr_{U_t,\Psi}[C(U_t) \ne C(U_t \oplus \Psi)] \le O(\delta\cdot\log^{c-1} 2^{n^c}) = o(1), \tag{3.6}$$

where $\Psi \in \{0,1\}^t$ is a random string where each bit is '1' independently with probability $\delta/2$. The above Equation 3.6 in turn implies that with probability $1-o(1)$ over R_δ, $\mathrm{Bias}_{U_t}[C(U_t{}^\rho)] \ge 1-o(1)$ (see below). (A similar connection between collision probability and bias was also used in Section 2.5.2.) Moreover by a Chernoff bound (see Appendix A)the fraction of $*$'s in $\rho \in R_\delta$ will be concentrated around δ.

So R_δ satisfies Items (2) and (3) in Lemma 3.11. But the problem is that R_δ requires at least 2^n bits to be generated, while we aim to a distribution on restrictions which can be generated with $\mathrm{poly}\, n$ bits. To this aim we *derandomize* R_δ.

Let W be a canonical circuit that given $I := O(2^n \log(1/\delta))$ random bits generates R_δ (say we use blocks of $O(\log(1/\delta))$ bits to put a $*$ with probability δ). It is easy to see that there is such a circuit W of size $\mathrm{poly}(2^n)$ and depth $O(1)$.

We now define $\tilde{R}^c$. For a constant c' to be determined later, consider the generator $Nis : \{0,1\}^{\log^d I} \to \{0,1\}^I$ which is $\left(2^{n^{c'}}\right)$-pseudorandom against depth c', where d is a constant that depends on c', as given by Theorem 3.14. Then

$$\tilde{R}^c := W(Nis(U_{\log^d I})).$$

We now prove that $\tilde{R}^c$ has the required properties.

(1) Follows immediately from Theorem 3.14.

(2) First we show that, even under pseudorandom restrictions, circuits of size 2^{n^c} and depth c still have low noise sensitivity. The claim about the bias then easily follows. To show this we use an approach similar to one used in the previous Chapter 2. Specifically, we let $N := 2^n$ and we note that the noise sensitivity of a constant-depth circuit C equals the acceptance probability of another (slightly bigger) constant-depth circuit C' defined as follows: Given a restriction ρ, C' tosses coins for U_N and U'_N and answers 1 if and only if $C({U_N}^\rho) \neq C({U'_N}^\rho)$. It is easy to see that such a C' can be implemented in constant depth. Combining C' with our constant depth circuit W that given random bits generates a random restriction in R_δ we obtain another constant depth circuit $C'' := C' \circ W$. Now, the acceptance probability of C'' over a truly random input is the noise sensitivity of C with respect to R_δ, while the acceptance probability of C'' over a pseudorandom input generated using Nisan's PRG Nis is the noise sensitivity of C with respect to $\tilde{R}^c$. Therefore, since C'' cannot distinguish the output of Nisan's PRG from truly random, we deduce that the noise sensitivity of C with respect to $\tilde{R}^c$ is close to the noise sensitivity of C with respect to R_δ.

Therefore, choosing a sufficiently large constant c' in the definition of $\tilde{R}^c$ we have that, for every $C : \{0,1\}^N \to \{0,1\}$ of size 2^{n^c} and depth c:

$$\begin{aligned}
&\Pr_{\rho_\sigma \in \tilde{R}^c, U_N, U'_N}[C({U_N}^{\rho_\sigma}) \neq C({U'_N}^{\rho_\sigma})] \\
\leq \; & \Pr_{\rho \in R_\delta, U_N, U'_N}[C({U_N}^{\rho}) \neq C({U'_N}^{\rho})] + o(1) \qquad \text{(by pseudorandomness)} \\
\leq \; & o(1) \qquad \text{(by Equation (3.6))}
\end{aligned}$$

We now deduce a claim about the bias. By above there is a function $\eta(n) = o(1)$ such that

$$\Pr_{\rho_\sigma \in \tilde{R}^c}\left[\Pr_{U_N, U'_N}[C({U_N}^{\rho_\sigma}) \neq C({U'_N}^{\rho_\sigma})] \leq \eta(n)\right] \geq 1 - \eta(n).$$

Noticing that $\Pr_{U_N, U'_N}[C({U_N}^{\rho_\sigma}) \neq C({U'_N}^{\rho_\sigma})] \leq o(1)$ implies that $\mathrm{Bias}_{U_N}[C({U_N}^{\rho_\sigma})] = 1 - o(1)$ concludes the proof of this item.

(3) Ajtai [Ajt1] shows the following:

Lemma 3.15 ([Ajt1]). *For every i there is a circuit C of size* $\mathrm{poly}(2^n)$ *and depth* $O(1)$ *such that, given u and a bit string of length 2^n:*

If the bit string has more than $u + 2^n/n^i$ occurrences of '1' then C outputs 1.

If the bit string has fewer than $u - 2^n/n^i$ occurrences of '1' then C outputs 0.

(In Theorem 5.2 we show that this can be done by uniform polynomial-size depth-3 circuits for the special setting of parameters where $u = 2^n/2$ and we have the constant 6 instead of n^i.)

We expect a random $\rho \in R_\delta$ to have $\delta 2^n$ $*$'s. By a Chernoff bound (see Appendix A) the probability that it has less than $(\delta 2^n)/2$ is $o(1)$. Since $\delta = 1/n^{c^2} = 1/\mathrm{poly}(n)$, by Lemma 3.15 there is a constant depth circuit of size $\mathrm{poly}(2^n)$ that can distinguish the cases, say, "more than $\delta/2$ fraction of $*$'s" and "less than $\delta/3$ fraction of $*$'s" (setting $u := (\delta 2^n)/2.5$ in Lemma 3.15). Therefore, choosing a sufficiently large constant c' in the definition of $\tilde{R}^c$, by pseudorandomness, we have that $\rho_\sigma \in \tilde{R}^c$ has at least $(\delta 2^n)/3$ $*$'s with high probability. □

3.5 Noise sensitivity of constant-depth circuits

In this section we prove Theorem 3.8. Recall that $\oplus$ denotes bitwise Xor.

Theorem (3.8, restated). *Let $C : \{0,1\}^n \to \{0,1\}$ be a circuit of size g and depth d. Let $X \in \{0,1\}^n$ be a random input and let $\Psi \in \{0,1\}^n$ be a random noise vector where each bit is '1' independently with probability $\delta < 1/2$. Then:*

$$\Pr_{X,\Psi}\Big[C(X) \neq C(X \oplus \Psi)\Big] \leq O(\delta \log^{d-1} g).$$

Although it is well known that constant-depth circuits have small noise sensitivity, the bound we need is not stated in the literature, and to prove it we need to introduce the Fourier machinery and then combine several results.

We now set up the usual Fourier machinery, see e.g. [LMN] for details. Whenever we discuss Fourier coefficients, we will use $\{+1,-1\}$ instead of $\{0,1\}$. When working over $\{+1,-1\}$ we denote also by $\oplus$ bitwise multiplication. Let $f : \{+1,-1\}^n \to \{+1,-1\}$ be any boolean function. Then f has a unique representation as a multilinear polynomial in $x_1,\ldots,x_n$ of total degree at most n. The S-th Fourier coefficient of f, denoted $\hat{f}(S)$, is the coefficient of the monomial $\prod_{i\in S} x_i$ in this polynomial. We also have, by Parseval's identity:

$$\sum_{S\subseteq[n]} \hat{f}(S)^2 = 1,$$

where $[n] := \{1,\ldots,n\}$.

The first result we need is a characterization of the noise sensitivity of a function in terms of its Fourier coefficients. Such a characterization is given by [O'D].

Lemma 3.16 ([O'D]). *Let $f : \{+1,-1\}^n \to \{+1,-1\}$ be any boolean function. Let $X \in \{+1,-1\}^n$ be a random input and let $\Psi \in \{+1,-1\}^n$ be a random noise vector where each bit is '1' independently with probability $\delta < 1/2$. Then:*

$$\Pr_{X,\Psi}[f(X) \neq f(X \oplus \Psi)] = \frac{1}{2} - \frac{1}{2} \sum_{S \subseteq [n]} (1-2\delta)^{|S|} \hat{f}(S)^2.$$

The second result we need is a bound on the Fourier coefficients of functions computed by constant-depth circuits. [Bop] gives a tight bound on the *average sensitivity* of constant-depth circuits. Combining this bound with the characterization of average sensitivity in terms of Fourier coefficients given by [KKL] (see also [LMN]), we obtain the following bound.

Lemma 3.17 ([KKL, Bop]). *Let f be computable by a circuit of size g and depth d. Then*

$$\sum_{S \subseteq [n]} |S| \hat{f}(S)^2 \leq O(\log^{d-1} g).$$

We can now prove Lemma 3.8.

Proof of Lemma 3.8. Let $f : \{+1,-1\}^n \to \{+1,-1\}$ be the function computed by C. We have:

$$\begin{aligned}
&\Pr_{X,\Psi}[C(X) \neq C(X \oplus \Psi)] \\
= \;& \frac{1}{2} - \frac{1}{2} \sum_{S \subseteq [n]} (1-2\delta)^{|S|} \hat{f}(S)^2 \qquad \text{(by Lemma 3.16)} \\
\leq \;& \frac{1}{2} - \frac{1}{2} \sum_{S \subseteq [n]} (1-2\delta|S|) \hat{f}(S)^2 \qquad \text{(by Bernoulli's inequality)} \\
= \;& \frac{1}{2} \sum_{S \subseteq [n]} 2\delta |S| \hat{f}(S)^2 \qquad \text{(by Parseval's identity)} \\
\leq \;& O(\delta \log^{d-1} g) \qquad \text{(by Lemma 3.17)}
\end{aligned}$$

which proves the lemma. $\square$

Chapter 4

Pseudorandom Bits for Constant-Depth Circuits with Sym Gates

A major consequence of the existence of average-case hard functions is that it allows us to build pseudorandom generators and consequently derandomize probabilistic algorithms (see Section 1.2). In this chapter we construct a function that is $\big(1/2 - 1/n^{\omega(1)}\big)$-hard for small constant-depth circuits with few arbitrary symmetric gates. We then apply it to construct a new pseudorandom generator and obtain a subexponential derandomization of the class of functions computable by small probabilistic constant-depth circuits with few arbitrary symmetric gates.

4.1 Introduction

A *pseudorandom generator* $G : \{0,1\}^l \to \{0,1\}^m$ is an efficient procedure that stretches l input bits into $m \gg l$ output bits such that the output distribution of the generator *fools* small circuits. That is, for every circuit C of size m we have

$$\left| \Pr_{x \in \{0,1\}^l}[C(G(x)) = 1] - \Pr_{x \in \{0,1\}^m}[C(x) = 1] \right| \leq \frac{1}{m}.$$

Pseudorandom generators have found a striking variety of applications in complexity theory, most notably to *derandomize* probabilistic algorithms.

Starting with the seminal work of Nisan and Wigderson [NW], a series of results (e.g. [BFNW, STV, SU1, Uma1]) show how to construct pseudorandom generators starting from an explicit function that requires circuits of superpolynomial size. However, no such function is known to exist.

On the other hand, pseudorandom generators that fool *restricted* kinds of circuits, such as *constant-depth* circuits with unbounded fan-in, are already very interesting.

They also have a large variety of applications (see, e.g., sections 2.4.1 and 3.4.2) and are central to understanding the power of randomness in restricted classes of algorithms. While there has been exciting progress in constructing explicit functions that require superpolynomial size constant-depth circuits with certain kinds of gates (e.g. [Hås, Raz, Smo, HG, RW, HM]), no explicit function is known to require superpolynomial size constant-depth circuits with Majority gates (cf. [RR]). This is an obstacle to construct pseudorandom generators, as most constructions need such a function. This need is due to the fact that the reductions in the proofs of correctness of these constructions use (a polynomial number of) Majority gates, as discussed in greater detail in Chapter 6.

But when starting from an *average-case* hard function, the reduction in the proof of correctness of the Nisan-Wigderson construction [NW] does not require Majority gates (here by average-case hard we mean a function $f : \{0,1\}^n \to \{0,1\}$ that is $\big(1/2 - 1/n^{\omega(1)}\big)$-hard). Thus, one can plug average-case lower bounds into the Nisan-Wigderson construction to get a generator that fools small constant-depth circuits. This approach is used in a celebrated work by Nisan [Nis1] (that actually predates the more general construction in [NW]) where he exhibits a generator $G : \{0,1\}^l \to \{0,1\}^{2^{l^{\Omega(1)}}}$ that fools small AC^0 circuits (i.e. constant-depth circuits with And and Or gates). This generator is based on the fact that Parity is very average-case hard for small AC^0 circuits [Hås].

Subsequently, Luby, Velickovic, and Wigderson (Theorem 2 in [LVW]) build a generator $G : \{0,1\}^l \to \{0,1\}^{l^{\Omega(\log l)}}$ that fools small Sym ∘ And circuits, i.e. depth 2 circuits with one *arbitrary symmetric gate* (Sym) at the top and And gates at the bottom. By arbitrary symmetric gate we mean a gate that computes an arbitrary function whose value depends only on the number of input bits being 1, important examples being Parity and Majority. This generator is based on the fact that the 'generalized inner product' function is average-case hard for small Sym ∘ And circuits with small bottom fan-in [BNS, HG].

The above two generators ([Nis1] and Theorem 2 in [LVW]) fool two incomparable classes of circuits (i.e. small AC^0 circuits and small Sym ∘ And circuits). In this chapter we exhibit a generator that fools a class of circuits strictly richer than both of them, namely small constant-depth circuits with few arbitrary symmetric gates.

4.1.1 Our Results

In this chapter we exhibit the following generator.

Theorem 4.1. *For every constant d there is a constant $\epsilon > 0$ such that for every l there is a generator $G : \{0,1\}^l \to \{0,1\}^m$, where $m = m(l) := l^{\epsilon \log l}$, such that for every circuit C of size m and depth d with $\log m(l)$ arbitrary symmetric gates, we* *have:*

$$\left|\Pr_{x\in\{0,1\}^m}[C(x)=1]-\Pr_{x\in\{0,1\}^l}[C(G(x))=1]\right|\leq\frac{1}{m},$$

and given $x \in \{0,1\}^l, i \leq m$, *we can compute the* i*-th output bit of* $G(x)$ *in time* poly(l).

The generator in Theorem 4.1 improves on the generator by Luby, Velickovic, and Wigderson (Theorem 2 in [LVW]) that achieves the same stretch (up to a different constant ϵ) but only fools circuits of depth 2 (as opposed to any constant depth) with one symmetric gate at the top. (We elaborate more on the difference between the two generators in Section 4.6.) The generator in Theorem 4.1 also fools a strictly richer class of circuits than Nisan's generator that fools constant depth circuits [Nis1]. (However, Nisan's generator has a much bigger stretch: it stretches l bits to $2^{l^{\Omega(1)}}$ bits.)

As a standard consequence of Theorem 4.1 we obtain the following subexponential derandomization of probabilistic constant depth circuits with a constant number of arbitrary symmetric gates. This is the richest probabilistic circuit class known to admit a subexponential derandomization. (See Section 1.2 or, e.g., [NW] for the connection between generators and derandomization.)

Corollary 4.2. *Let a function* f *be computed by a uniform family of probabilistic* poly(n)*-size constant depth circuits with* $O(\log n)$ *arbitrary symmetric gates. Then* f *can be computed in deterministic time* $exp(2^{O(\sqrt{\log n})}) = 2^{n^{o(1)}}$.

4.1.2 Techniques

The generator in Theorem 4.1 is obtained by plugging into the Nisan-Wigderson pseudorandom generator construction [NW] a function that is very hard on average for 'small' constant-depth circuits with 'few' arbitrary symmetric gates (cf. Theorem 4.3 below). Here a simple and crucial observation is that the reduction in the proof of correctness of the Nisan-Wigderson generator (essentially) does not increase the number of arbitrary symmetric gates.

Given our average-case hardness result (Theorem 4.3), the construction of our generator is simpler than the construction of the (weaker) generator by Luby, Velickovic, and Wigderson (Theorem 2 in [LVW]) that uses more involved combinatorial arguments than those in [NW]. These more involved combinatorial arguments were probably used because the generator in [LVW] builds on a function that is hard on average for circuits of depth 2 (as opposed to any constant depth), and thus one cannot use directly the Nisan-Wigderson construction [NW] since the reduction in its proof of correctness increases the depth by 1.

We now state our average-case hardness result.

Theorem 4.3. *There is a function* $f : \{0,1\}^* \to \{0,1\}$ *computable in polynomial time such that for every constant* d *there is a constant* $\epsilon > 0$ *such that for every* n

and every circuit C of size $n^{\epsilon \cdot \log n}$, depth d and with $\epsilon \log^2 n$ arbitrary symmetric gates, the following holds:

$$\Pr_{x \in \{0,1\}^n}[C(x) \neq f(x)] \geq 1/2 - 1/n^{\epsilon \cdot \log n}.$$

We now explain the techniques involved in proving Theorem 4.3. To simplify the discussion we first focus on how to prove an average-case hardness result for 'small' constant-depth circuits with *one* arbitrary symmetric gate at the top, i.e. 'small' $\mathsf{Sym} \circ \mathsf{AC}^0$ circuits (Theorem 4.4). The extension to circuits with more arbitrary symmetric gates is deferred to the paragraph "Circuits with more Arbitrary Symmetric Gates" below. We obtain our average-case hardness result for 'small' $\mathsf{Sym} \circ \mathsf{AC}^0$ circuits through a modification of previous lower bounds. We now discuss these previous lower bounds, then we discuss why they are not sufficient for our purposes, and then we sketch the proof of our average-case hardness result for 'small' $\mathsf{Sym} \circ \mathsf{AC}^0$ circuits.

Previous lower bounds: Babai, Nisan and Szegedy [BNS] prove that the "generalized inner product" function (i.e., $GIP_{n,s}(x) := \bigoplus_{i \leq n} \bigwedge_{j \leq s} x_{i,j}$) is very hard *on average* for multiparty communication complexity protocols among 'few' parties that communicate 'little'.

Håstad and Goldmann [HG] notice that any function computed by a 'small' depth 2 circuit with an arbitrary symmetric gate of unbounded fan-in at the top and (arbitrary) gates of 'small' fan-in at the bottom can be computed by a multiparty communication complexity protocol among 'few' parties communicating 'little'. Thus, by the above result [BNS], they obtain that GIP is average-case hard for that kind of circuits. Now, by the so-called "ϵ-discriminator lemma"[1] of Hajnal et. al. [HMP$^+$] they conclude that GIP cannot be computed, in the *worst-case*, by 'small' depth 3 circuits with one majority gate of unbounded fan-in at the top, arbitrary symmetric gates of unbounded fan-in in the middle, and (arbitrary) gates of 'small' fan-in at the bottom.

Razborov and Wigderson [RW] eliminate the constrain on the bottom fan-in: they exhibit a new function RW that cannot be computed, in the worst-case, by 'small' depth 3 circuits with one majority gate at the top, symmetric gates in the middle, and And gates at the bottom, where all the gates have unbounded fan-in ($\mathsf{Maj} \circ \mathsf{Sym} \circ \mathsf{And}$ circuits). Their function RW is obtained from GIP by replacing each input variable with a parity function, i.e. $RW(x) := \bigoplus_{i \leq n} \bigwedge_{j \leq \log n} \bigoplus_{k \leq n} x_{i,j,k}$.

To explain their argument we introduce *restrictions* [FSS]. A restriction on m variables $x_1, x_2, \ldots, x_m$ is a map $\rho : \{x_1, x_2, \ldots, x_m\} \to \{0, 1, *\}$. For a circuit C we denote by $C|_\rho$ the circuit we get by doing the substitutions prescribed by ρ, followed

[1] This lemma states that if a function is computed by a 'small' circuit with a Majority gate at the top, then some input circuit to the Majority gate computes the function 'well' on average.

by all obvious cancellations made possible by applying ρ. The input variables of $C|_\rho$ are the variables which were given the value * by ρ.

The argument in [RW] goes as follows: suppose that RW is computable by a 'small' $\mathsf{Maj} \circ \mathsf{Sym} \circ \mathsf{And}$ circuit C. Then there is a restriction ρ that accomplishes simultaneously two things: (1) $C|_\rho$ has 'small' bottom fan-in and (2) $C|_\rho$ is still computing GIP as a subfunction. Note that, by definition of RW and by the nature of parity, (2) happens whenever for every i, j there is k such that $\rho(x_{i,j,k}) = *$. But (1) and (2) contradict the above result by Håstad and Goldmann.

Finally, Hansen and Miltersen [HM] observed that RW actually cannot be computed by 'small' circuits of *any* constant depth with one majority gate at the top, and one layer of arbitrary symmetric gates immediately below it, where all the gates have unbounded fan-in ($\mathsf{Maj} \circ \mathsf{Sym} \circ \mathsf{AC}^0$ circuits). The argument in [HM] goes as follows: suppose that RW is computable by a 'small' $\mathsf{Maj} \circ \mathsf{Sym} \circ \mathsf{AC}^0$ circuit C. Then there is a restriction ρ that accomplishes simultaneously two things: (1') $C|_\rho$ is equivalent to a 'small' $\mathsf{Maj} \circ \mathsf{Sym} \circ \mathsf{And}$ circuit and (2') $C|_\rho$ is still computing RW on an input of polynomially related size. (1') is obtained through Håstad's switching lemma [Hås], and for (2') they show that for every i, j there are 'many' k's such that $\rho(x_{i,j,k}) = *$. But (1') and (2') contradict the above result by Razborov and Wigderson.

Why previous lower bounds are not sufficient to our purposes: The main problem with these previous lower bounds is that they only give a function that is *worst-case* hard for $\mathsf{Sym} \circ \mathsf{AC}^0$ circuits, while as explained before we need a function that is *average-case* hard. In fact, the choice of parameters in the definition of RW implies that $\Pr_x[RW(x) = 0] = 1/2 + \Omega(1)$, and thus RW cannot be average-case hard (since the constant size circuit that always outputs '0' computes the function fairly well on average). Moreover the choice of parameters for the restrictions in [RW] does not guarantee that the reduction holds with high probability, which is needed to establish average-case hardness.

We would like to stress that, even though in earlier chapters we saw several hardness amplification results, none of the known hardness amplifications can be applied to construct an average-case hard function for small $\mathsf{Sym} \circ \mathsf{AC}^0$ circuits starting from a worst-case hard function for small $\mathsf{Sym} \circ \mathsf{AC}^0$ circuits. The reason for this is investigated in Chapter 6.

Proof sketch of our average-case hardness result for $\mathsf{Sym} \circ \mathsf{AC}^0$ circuits: We define a function f (similar to RW, but with a different choice of parameters), and we show that f is average-case hard for $\mathsf{Sym} \circ \mathsf{AC}^0$ circuits. Our argument simplifies the previous ones and goes as follows: Suppose that C is a small $\mathsf{Sym} \circ \mathsf{AC}^0$ circuit computing f. We argue that, with high probability $\left(1 - n^{-\Omega(\log n)}\right)$ over the choice of a random restriction ρ, both the following two events happen:

- Event $E_1 :=$ the function computed by $C|_\rho$ is computable by a multiparty communication complexity protocol among 'few' parties communicating 'little'.

- Event $E_2 := C|_\rho$ is computing GIP as a subfunction.

To show E_1 we use Håstad's switching lemma to argue that with high probability over ρ, $C|_\rho$ is equivalent to a 'small' depth-2 circuit with a symmetric gate at the top (of unbounded fan-in) and And gates of 'small' fan-in at the bottom, and then use Håstad and Goldmann's connection [HG] between these circuits and multiparty communication complexity protocols (cf. paragraph "Previous Lower Bounds"). Now, when ρ satisfies both E_1 and E_2 we have that $\Pr_y[C|_\rho(y) \neq GIP(y)] \geq 1/2 - n^{-\Omega(\log n)}$ by the multiparty communication complexity lower bound by Babai, Nisan and Szegedy [BNS]. Since we can think of a random input x as being generated by first choosing a random restriction ρ and then a random input y for the *'s of ρ (so that $C(x) = C|_\rho(y)$), we have that

$$\begin{aligned}
&\Pr_x[C(x) \neq f(x)] \\
&\quad \geq \Pr_y\left[C|_\rho(y) \neq GIP(y) \,\middle|\, \rho \text{ satisfies } E_1 \text{ and } E_2\right] \cdot \Pr_\rho\left[\rho \text{ satisfies } E_1 \text{ and } E_2\right] \\
&\quad \geq \left(1/2 - n^{-\Omega(\log n)}\right) \cdot \left(1 - n^{-\Omega(\log n)}\right) \\
&\quad = 1/2 - n^{-\Omega(\log n)}.
\end{aligned}$$

We show that the above argument goes through for $\mathsf{Sym} \circ \mathsf{AC}^0$ circuits C of size $n^{\Omega(\log n)}$ and this proves our average-case hardness result for $\mathsf{Sym} \circ \mathsf{AC}^0$ circuits.

Circuits with more arbitrary symmetric gates: Before discussing how to extend our techniques to get an average-case hardness result for 'small' constant-depth circuits with $\epsilon \log^2 n$ arbitrary symmetric gates, we would like to mention two other approaches that give weaker bounds. Beigel (Theorem 5.1 in [Bei2]) shows that for every circuit of size s and depth d with σ arbitrary symmetric gates there is another circuit of size $s^{2^\sigma+1}$ and depth $d+1$ with *one* arbitrary symmetric gate at the top computing the same function. Combining this with our average-case hardness result for $\mathsf{Sym} \circ \mathsf{AC}^0$ circuits one obtains an average-case hardness result for constant-depth circuits of size $n^{\epsilon \cdot \log n}$ with a constant number of arbitrary symmetric gates. But this approach gives weaker bounds (than $n^{\Omega(\log n)}$) if the circuits have $\sigma = \omega(1)$ arbitrary symmetric gates; and it gives nothing at all if the circuits have $\sigma = \log\log n$ arbitrary symmetric gates.

Chattopadhyay and Hansen [CH] prove a *worst-case* hardness result for constant-depth circuits of size $n^{\epsilon \cdot \log n}$ with $\epsilon \log^2 n$ arbitrary symmetric gates. They obtain this result independently from ours. Subsequently to our results for $\mathsf{Sym} \circ \mathsf{AC}^0$ circuits, they also prove an *average-case* hardness result for constant-depth circuits of size $n^{\epsilon \cdot \log n}$ with fewer arbitrary symmetric gates, namely $\epsilon \log n$.

Inspired by the work of Chattopadhyay and Hansen, we prove an average-case hardness result for constant-depth circuits of size $n^{\epsilon \cdot \log n}$ with $\epsilon \log^2 n$ arbitrary symmetric gates (Theorem 4.3). The proof of our result has the same structure of our result for $\mathsf{Sym} \circ \mathsf{AC}^0$ circuits discussed in the previous paragraph. The only difference is proving that, if C is a 'small' constant-depth circuit with $\epsilon \log^2 n$ arbitrary symmetric gates, then with high probability over a random restriction ρ the function computed by $C|_\rho$ is computable by a multiparty communication complexity protocol P among 'few' parties communicating 'little' (cf. event E_1 in the previous paragraph). The idea is to let the protocol P compute the outputs of each arbitrary symmetric gate in order. Specifically, first fix a topological order of the arbitrary symmetric gates (the simple order induced by reading the gates level by level from the inputs to the output node will do). Now consider the $\mathsf{Sym} \circ \mathsf{AC}^0$ subcircuit C_1 whose root is the first arbitrary symmetric gate in this order. We know that with high probability over the restriction ρ, the function computed by $C_1|_\rho$ is computable by a multiparty communication complexity protocol P_1 exchanging 'few' bits (cf. event E_1 in the previous paragraph). Our protocol P first simulates P_1 to determine the output b_1 of $C_1|_\rho$. Then it considers the $\mathsf{Sym} \circ \mathsf{AC}^0$ circuit C_2 whose root is the second arbitrary symmetric gate, and where the first arbitrary symmetric gate is replaced with the constant b_1. Again, we argue that the function computed by $C_2|_\rho$ is computable by a multiparty communication complexity protocol P_2 exchanging 'few' bits. Our protocol P now simulates P_2 to determine the output b_2 of $C_2|_\rho$. We continue in this way until all the arbitrary symmetric gates are computed. Assuming w.l.o.g. that the output gate of the circuit is included in the arbitrary symmetric gates, the protocol P computes $C|_\rho$.

4.1.3 Organization

This chapter is organized as follows. In Section 4.2 we fix some notation. In Section 4.3 we show how our average-case hardness result (Theorem 4.3) implies our generator (Theorem 4.1). In Section 4.4 we prove our average-case hardness result for $\mathsf{Sym} \circ \mathsf{AC}^0$ circuits. In Section 4.5 we extend this to our average-case hardness result for constant-depth circuits with few arbitrary symmetric gates, thus proving Theorem 4.3. In Section 4.6 we elaborate on why our generator improves on the generator by Luby, Velickovic, and Wigderson (Theorem 2 in [LVW]). In Section 4.7 we discuss some open problems.

4.2 Preliminaries

An *arbitrary symmetric gate* (Sym) is a gate that computes an arbitrary symmetric function, i.e. a function whose value depends only on the number of input bits being 1 (e.g. Parity, Majority). We use standard definitions of constant depth circuits, which we now briefly recall. Constant depth circuits consist of And, Or and possibly other gates (e.g. one arbitrary symmetric gates). It is intended that all gates whose type is not specified are either And or Or, and that And and Or gates are not counted towards arbitrary symmetric gates. All circuit gates, unless specified otherwise, have unbounded fan-in. Circuits take both input variables and their negations as input. *Bottom* gates are the one adjacent to the input bits. The *top* gate is the output gate. Levels are numbered from the bottom. So the input bits are at level 0, the bottom gates at level 1 and so on. Gates at level i are connected to gates at levels $i-1$ and $i+1$ only. The *depth* of a circuit is the longest path from any input to the output. The *size* of a circuit is the number of gates in it. Multiple edges between pairs of nodes in the circuit are not allowed (otherwise an arbitrary symmetric gate can compute any function; this convention is standard in the literature, e.g. [HG]).

4.3 From average-case hardness to pseudorandomness

In this section we show how our average-case hardness result (Theorem 4.3) implies our generator (Theorem 4.1). We restate the theorems for the reader's convenience.

Theorem (4.1, restated)**.** *For every constant d there is a constant $\epsilon > 0$ such that for every l there is a generator $G : \{0,1\}^l \to \{0,1\}^m$, where $m = m(l) := l^{\epsilon \log l}$, such that for every circuit C of size m and depth d with $\log m(l)$ arbitrary symmetric gates, we have:*

$$\left| \Pr_{x\in\{0,1\}^m}[C(x) = 1] - \Pr_{x\in\{0,1\}^l}[C(G(x)) = 1] \right| \le \frac{1}{m},$$

and given $x \in \{0,1\}^l, i \le m$, we can compute the i-th output bit of $G(x)$ in time $\mathrm{poly}(l)$.

Theorem (4.3, restated)**.** *There is a function $f : \{0,1\}^* \to \{0,1\}$ computable in polynomial time such that for every constant d there is a constant $\epsilon > 0$ such that for every n and every circuit C of size $n^{\epsilon\cdot\log n}$, depth d and with $\epsilon \log^2 n$ arbitrary symmetric gates, the following holds:*

$$\Pr_{x\in\{0,1\}^n}[C(x) \ne f(x)] \ge 1/2 - 1/n^{\epsilon\cdot\log n}.$$

Proof of Theorem 4.1, assuming Theorem 4.3. The generator is obtained by plugging the function from Theorem 4.3 into Nisan-Wigderson's pseudorandom generator construction [NW]. Specifically, they show how given a function $f : \{0,1\}^{\sqrt{l/2}} \to \{0,1\}$ and a parameter m (which we set to be $m(l) := l^{\epsilon\cdot\log l}$) to construct a generator $G : \{0,1\}^l \to \{0,1\}^m$ such that every circuit C for which

$$\left| \Pr_{x\in\{0,1\}^m}[C(x) = 1] - \Pr_{x\in\{0,1\}^l}[C(G(x)) = 1] \right| > 1/m$$

can be transformed into another circuit C' of size $|C| + \text{poly}(m)$ that computes the function f correctly with probability (over random input) greater than $1/2 + 1/m^2 = 1/2 + 1/l^{2\epsilon \log l}$.

As observed in [Nis1, NW], C' is simply C with one more layer of And (or Or) gates at the bottom, and possibly negating the output. Adding one layer of And (or Or) gates at the bottom clearly does not increase the number of arbitrary symmetric gates in C, and we can think of negating the output by, say, including the top gate in the arbitrary symmetric gates and complementing it. Thus, if C is a circuit of size $m = m(l) = l^{\epsilon \log l}$ of depth d with $\log m(l) = \epsilon \log^2 l$ arbitrary symmetric gates we obtain another circuit C' of size $l^{O(\epsilon \log l)}$ of depth $d + 1$ with $1 + \epsilon \log^2 l$ arbitrary symmetric gates that computes $f : \{0,1\}^{\sqrt{l/2}} \to \{0,1\}$ with probability greater than $1/2 + 1/l^{2\epsilon \log l}$. This contradicts Theorem 4.3 for sufficiently small ϵ.

The complexity of the generator follows from the arguments in [Nis1, NW] and the fact that f is computable in time $\text{poly}(l)$. □

4.4 Average-case hardness for $\mathsf{Sym} \circ \mathsf{AC}^0$ circuits

In this section we prove our average-case hardness result for 'small' constant-depth circuits with one arbitrary symmetric gate at the top.

Theorem 4.4. *There is a function $f : \{0,1\}^* \to \{0,1\}$ computable in polynomial time such that for every constant d there is a constant $\epsilon > 0$ such that for every n and every circuit C of size $n^{\epsilon \cdot \log n}$, depth d, with 1 arbitrary symmetric gate at the top, the following holds:*

$$\Pr_{x \in \{0,1\}^n}[C(x) \neq f(x)] \geq 1/2 - 1/n^{\epsilon \cdot \log n}.$$

In the rest of this section we prove Theorem 4.4. In the proof we use two results which we describe in the following two subsections. The first is a version of Håstad's switching lemma [Hås] due to Beame [Bea], and the second is the multiparty communication complexity lower bound for *GIP* by Babai, Nisan and Szegedy [BNS].

4.4.1 Switching Lemma

We now describe the switching lemma we use in the proof of Theorem 4.4. As in [HM], the crucial property that we need is that the DNF obtained after applying the restriction is such that all the terms are mutually contradictory, i.e. no input satisfies more than one term. This allows us to merge the top Or gate of the DNF in the symmetric gate at the top (cf. Fact 4.6). The fact that this property holds for Håstad's switching lemma was already noted by Boppana and Håstad in [Hås] (inside the proof of Lemma 8.3). However, there does not seem to be a full proof of this fact in the literature. For this reason we use a slightly different version of the Håstad's switching lemma, due to Beame [Bea].

A *restriction* on m variables $x_1, x_2, \ldots, x_m$ is a map $\rho : \{x_1, x_2, \ldots, x_m\} \to \{0, 1, *\}$. For a function $f : \{0,1\}^m \to \{0,1\}$ we denote by $f|_\rho$ the function we get by doing the substitutions prescribed by ρ. $f|_\rho$ will be a function of the variables that were given the value * by ρ. Similar conventions hold for circuits. If ρ and ρ' are restrictions, and ρ' is defined on the variables mapped to * by ρ we write $\rho\rho'$ for the restriction obtained by combining ρ and ρ', so that $f|_{\rho\rho'} = (f|_\rho)|_{\rho'}$. Let $R_m^{\delta \cdot m}$ denote the uniform distribution on restrictions on m variables assigning exactly δm variables to $*$, and assigning random values to the others.

A *decision tree* on m variables is a labelled binary tree where edges and leaves are labelled with 0 or 1, and internal nodes with variables. A decision tree computes a function in the intuitive way, starting at the root and following the path according to the values of the input variables, and outputting the value at the reached leaf.

Lemma 4.5 ([Bea]). *Let φ be a* DNF *or a* CNF *formula in m variables with bottom fan-in at most r. For every $s \geq 0, p < 1/7$, the probability over $\rho \in R_m^{p \cdot m}$ that the function computed by $\varphi|_\rho$ is not computable by a decision tree of height strictly less than s is less than $(7pr)^s$.*

We will use Lemma 4.5 in combination with the following fact.

Fact 4.6. *Let f be a symmetric function of s decision trees of height h. Then f is computable by a depth 2 circuit of size $s \cdot 2^h + 1$ with a symmetric gate of unbounded fan-in at the top and* And *gates of fan-in h at the bottom.*

Proof. Write each decision tree as a DNF with bottom fan-in h, where each term corresponds to a path leading to 1. The number of terms in each DNF is at most 2^h, i.e. at most the number of paths in a decision tree of height h. Because every input to a decision tree follows a unique path, each DNF we construct has the property that every input satisfies at most one term. Thus we can merge the top Or gate of all these DNF's with the top symmetric gate of the circuit. Specifically, if the original symmetric gate was $\psi(x_1, x_2, \ldots, x_s) = g(\sum_{i \leq s} x_i)$ for some arbitrary function $g : [s] \to \{0,1\}$, the new symmetric gate is simply $\psi'(x_1, x_2, \ldots, x_{s \cdot 2^h}) := g(\sum_{i \leq s \cdot 2^h} x_i)$. □

4.4.2 Multiparty communication complexity

In this section we describe some results on communication complexity that will be used in the proof of our main results. The model of interest is the *multiparty communication complexity model*. In this model there are s parties, each having unlimited computational power, who wish to collaboratively compute a certain function. The input bits to the function are partitioned in s blocks, and the i-th party knows all the input bits except those corresponding to the i-th block in the partition. The communication between the parties is by "writing on a blackboard" (broadcast): any bit sent by any party is seen by all the others. The parties exchange messages according to a fixed protocol. The measure of interest is the number of bits exchanged by the parties. We refer the reader to the book by Kushilevitz and Nisan [KN] for background on this model.

Babi, Nisan and Szegedy [BNS] prove a multiparty communication complexity lower bound for the *generalized inner product* function $GIP_{n,s} : \{0,1\}^{n \cdot s} \to \{0,1\}$, which is defined as follows:

$$GIP_{n,s}(x) := \bigoplus_{i=1}^{n} \bigwedge_{j=1}^{s} x_{i,j}.$$

Lemma 4.7 ([BNS])**.** *There is a partition of the inputs to $GIP_{n,s}$ in s blocks such that the following holds: Let P be a s-party communication complexity protocol exchanging at most $.1 \cdot (n/4^s - \log(1/\gamma))$ bits of communication, then*

$$\Pr_{x \in \{0,1\}^{n \cdot (.3 \log n)}} \left[P(x) \neq GIP_{n,.3 \log n}(x) \right] \geq 1/2 - \gamma.$$

Håstad and Goldmann [HG] show that the function computed by a 'small' $\mathsf{Sym} \circ \mathsf{And}$ circuit with 'small' bottom fan-in can be computed by a multiparty communication complexity protocol among 'few' parties exchanging 'few' bits.

Lemma 4.8 ([HG])**.** *Let C be a depth-2 circuit of size s with an arbitrary symmetric gate (of unbounded fan-in) at the top, and* And *gates of fan-in strictly less than s at the bottom. Then the function computed by C can be computed (under any partition of the input) by a s-party communication complexity protocol exchanging $1 + s \log s$ bits.*

The idea in Lemma 4.8 is that since each bottom And gate has fan-in strictly less than s then, for any partition of the input in s blocks, the input bits to each And can lie in at most $s - 1$ distinct blocks. Therefore we can assign each And gate to some party that knows all the input bits necessary to compute it. Now each party broadcasts the number of And gates assigned to him that evaluate to 1, which takes at most $\log s$ bits. Since the top gate is symmetric this information is sufficient to compute the output of the circuit.

Our next lemma combines the above observation by Håstad and Goldmann with the "switching lemma" results from the previous section to argue the following: for every small $\mathsf{Sym} \circ \mathsf{AC}^0$ circuit, with high probability over a suitable restriction ρ, the function computed by $C|_\rho$ can be computed by a multiparty communication complexity protocol among 'few' parties exchanging 'few' bits.

Lemma 4.9. *For every constant d there is a constant $\epsilon > 0$ such that the following holds. Let $C : \{0,1\}^n \to \{0,1\}$ be a circuit of size $n^{\epsilon \cdot \log n}$, depth d, with 1 arbitrary symmetric gate at the top. Let ρ be a random restriction on the n input variables* *that assigns * to a subset of the variables of relative size* $1/n^{.1}$ *, i.e. let* $\rho \in R_n^{n/n^{.1}}$. *Then with probability at least* $1 - n^{-\Omega(\log n)}$ *over* ρ*, the function computed by* $C|_\rho$ *is computable (under any partition of the input) by a* $.3\log n$*-party communication complexity protocol exchanging* $\log^3 n$ *bits of communication.*

Proof. The proof amounts to a combination of the previous lemmas for some specific setting of parameters.

Claim 4.10. *With probability* $1 - n^{-\Omega(\log n)}$ *over* $\rho \in R_n^{n/n^{.1}}$*, the function computed by* $C|_\rho$ *is computable by a depth-2 circuit of size* $|C| \cdot 2^{.3\log n}$ *with a symmetric gate (of unbounded fan-in) at the top and* And *gates of fan-in strictly less than* $.3\log n$ *at the bottom.*

The lemma follows by the above claim using Lemma 4.8, which implies that the function computed by a depth-2 circuit of size $s = |C| \cdot 2^{.3\log n} \leq n^{\log n}$ with a symmetric gate (of unbounded fan-in) at the top and And gates of fan-in strictly less than $.3\log n$ at the bottom is computable by a $.3\log n$-party communication complexity protocol exchanging $1 + (.3\log n)\log s \leq \log^3 n$ bits.

We now prove Claim 4.10. Similar calculations have already been done elsewhere (e.g., Lemma 2 in [LMN]). However, we have not found the exact claim we need in the literature.

Proof of Claim 4.10. We see the restriction ρ as $d-1$ successive applications of restrictions $\rho_1, \rho_2, \ldots, \rho_{d-1}$ each mapping to * a subset of variables of relative size $1/n^\alpha$ of the (remaining) variables. Taking $\alpha = .1/(d-1)$ we have that, after applying all $d-1$ restrictions, the total number of variables mapped to * is $n \cdot (1/n^\alpha)^{d-1} = n/n^{.1}$, and so this distribution on restrictions is exactly $R_n^{n/n^{.1}}$.

For every $i \in [d-1]$ let DT_i be the event that, after applying the first i restrictions $\rho_1, \rho_2, \ldots, \rho_i$, the function computed by every gate at level i is computable by a decision tree of height strictly less than $.3\log n$. We now bound $\Pr_\rho[\text{ not } DT_{d-1}]$. Note that it is at most

$$\Pr_{\rho_1}[\text{ not } DT_1] + \Pr_{\rho_1,\rho_2}[\text{ not } DT_2|DT_1] + \ldots + \Pr_{\rho_1,\rho_2,\ldots,\rho_{d-1}}[\text{ not } DT_{d-1}|DT_{d-2}].$$

We now bound each term. Fix any $i \leq d-1$ and consider $\Pr_{\rho_1,\rho_2,\ldots,\rho_i}[\text{ not } DT_i|DT_{i-1}]$ (if $i=1$, think of the input variables as functions computed by decision trees of depth 1, and define $DT_0 :=$ TRUE). Fix any gate φ at level i. Without loss of generality assume φ is an Or gate (otherwise we can consider its negation, apply the same reasoning, and then negate again). Since we are conditioning over DT_{i-1}, all the functions computed by gates at level $i-1$ can be computed by decision trees of height (strictly) less than $.3\log n$. Write each such function as a DNF with terms of size at most $.3\log n$ (where each term corresponds to a path in the decision tree leading to '1'). Merging the top Or gates of all these DNF's with φ

we see that, given DT_{i-1}, the function computed by φ is a DNF with terms of size at most $r = .3 \log n$. By Lemma 4.5 the probability over the choice of the i-th restriction ρ_i that the function computed by $\varphi|_{\rho_1 \rho_2 \cdots \rho_i}$ cannot be computed by a decision tree of depth strictly less than $s = .3 \log n$ is at most

$$(7pr)^s = (7 \cdot (1/n^\alpha) \cdot (.3 \log n))^{.3 \log n} = n^{-\Omega(\log n)}.$$

Thus by a union bound we have that

$$\Pr_{\rho_1, \rho_2, \ldots, \rho_i}[\text{ not } DT_i | DT_{i-1}]$$

is at most $n^{-\Omega(\log n)}$ times the number of gates at level i. Therefore, if the circuit C has size $n^{\epsilon \log n}$ for sufficiently small ϵ we have

$$\Pr_\rho[\text{ not } DT_{d-1}] \leq n^{-\Omega(\log n)} \cdot |C| = n^{-\Omega(\log n)}.$$

We have shown that with probability $1 - n^{-\Omega(\log n)}$ (over ρ) the function computed by $C|_\rho$ is computable by a symmetric function of $|C|$ decision trees of height strictly less than $.3 \log n$. By Fact 4.6 we can write each decision tree as a DNF and merge the top Or gates of these DNF's into the top symmetric gate of C, thus proving the claim. □

□

4.4.3 Proof of Theorem 4.4

We now prove Theorem 4.4. We restate the theorem for the reader's convenience.

Theorem (4.4, restated)**.** *There is a function $f : \{0,1\}^* \to \{0,1\}$ computable in polynomial time such that for every constant d there is a constant $\epsilon > 0$ such that for every n and every circuit C of size $n^{\epsilon \cdot \log n}$, depth d, with 1 arbitrary symmetric gate at the top, the following holds:*

$$\Pr_{x \in \{0,1\}^n}[C(x) \neq f(x)] \geq 1/2 - 1/n^{\epsilon \cdot \log n}.$$

Proof of Theorem 4.4. Similarly to [RW], we consider the function obtained by attaching Parity gates on n bits at the bottom of $GIP_{n,.3 \log n}$. That is, let $f_n : \{0,1\}^{n^2(.3 \log n)} \to \{0,1\}$ be defined as

$$f_n(x) := \bigoplus_{i=1}^{n} \bigwedge_{j=1}^{.3 \log n} \bigoplus_{k=1}^{n} x_{i,j,k}.$$

We will prove Theorem 4.3 with f_n as hard function. While f_n is a function on $m = m(n) := n^2(.3 \log n)$ bits, it will be convenient to parameterize it by n. Since we will prove $n^{\Omega(\log n)}$ lower bounds for f_n and the input length of f_n is $m = \mathrm{poly}(n)$, we also obtain $m^{\Omega(\log m)}$ lower bounds for f_n (for a different hidden constant in the $\Omega(\cdot)$).

It is easy to see that f_n is computable in polynomial time.

Let $C : \{0,1\}^m \to \{0,1\}$ be a circuit as in the statement of Theorem 4.4, for a sufficiently small constant ϵ. Let ρ be a random restriction on the m input variables that assigns * to a subset of the variables of relative size $1/m^{.1}$, i.e. let $\rho \in R_m^{m/m^{.1}}$.

Consider the following two events.

- Event $E_1 :=$ the function computed by $C|_\rho$ is computable (under any partition of the input) by a $.3 \log n$-party communication complexity protocol exchanging $n^{.2}$ bits.

- Event $E_2 :=$ for every $i \in [n], j \in [.3 \log n]$ there is $k \in [n]$ such that $\rho(x_{i,j,k}) = *$. (In other words, for each of the $n \cdot (.3 \log n)$ bottom parity functions of f_n, ρ maps some of its input variable to *.)

Claim 4.11. $\Pr_{\rho \in R_m^{m/m^{.1}}}[E_1 \wedge E_2] \geq 1 - n^{-\Omega(\log n)}$.

Before proving Claim 4.11 let us see how we can use it to prove Theorem 4.4. Suppose that some $\rho \in R_m^{m/m^{.1}}$ satisfies both E_1 and E_2. Then

$$\Pr_{y \in \{0,1\}^{m/m^{.1}}}[C|_\rho(y) \neq f_n|_\rho(y)] \geq 1/2 - n^{-\Omega(\log n)}. \tag{4.1}$$

This holds by Lemma 4.7. Specifically, fix any restriction ρ' taken on the variables mapped to * by ρ, such that for every $i \in [n], j \in [.3 \log n]$ there is *exactly one* $k \in [n]$ such that $\rho\rho'(x_{i,j,k}) = *$. We then have that $f_n|_{\rho\rho'}$ equals $GIP_{n,.3\log n}$ (up to possibly negating some input variables). If the function computed by $C|_\rho$ is computable by a s-party communication complexity protocol exchanging $n^{.2}$ bits then clearly the same holds for the function computed by $C|_{\rho\rho'}$. Therefore by the multiparty communication complexity lower bound for *GIP* (Lemma 4.7) we obtain (noticing that for $s = .3 \log n, \gamma = 2^{-n^{.3}}$ we have $.1 \cdot (n/4^s - \log(1/\gamma)) = \Omega(n^{.4} - n^{.3}) > n^{.2}$):

$$\Pr_{z \in \{0,1\}^{n(.3\log n)}}[C|_{\rho\rho'}(z) \neq f_n|_{\rho\rho'}(y)] \geq 1/2 - 1/2^{n^{\Omega(1)}} \geq 1/2 - n^{-\Omega(\log n)}.$$

Equation 4.1 follows noticing that we can think of a random y as choosing first a random ρ' as above and then a random $z \in \{0,1\}^{n(.3\log n)}$ for the *'s of ρ' (so that $C|_\rho(y) = C|_{\rho\rho'}(z)$).

Thus we have:

$$\begin{aligned}
&\Pr_x[C(x) \neq f_n(x)] \\
&= \Pr_{\rho \in R_m^{m/m^{.1}}, y \in \{0,1\}^{m/m^{.1}}}[C|_\rho(y) \neq f_n|_\rho(y)] \\
&\geq \Pr_{\rho \in R_m^{m/m^{.1}}, y \in \{0,1\}^{m/m^{.1}}}[C|_\rho(x) \neq f_n|_\rho(x) | E_1 \wedge E_2] \cdot \Pr[E_1 \wedge E_2] \\
&\geq \left(1/2 - n^{-\Omega(\log n)}\right) \cdot \left(1 - n^{-\Omega(\log n)}\right) \qquad \text{(by Equation 4.1 and Claim 4.11)} \\
&= 1/2 - n^{-\Omega(\log n)},
\end{aligned}$$

which proves Theorem 4.3.

It is only left to prove Claim 4.11.

Proof of Claim 4.11. We show that E_1 and E_2 each do not happen with probability at most $n^{-\Omega(\log n)}$.

The bound on $\Pr_\rho[\text{not } E_1]$ is given by Lemma 4.9. (The direct application of Lemma 4.9 gives communication complexity $\operatorname{poly}\log(n) \ll n^{.2}$ for circuits of size $m^{\epsilon \log m} \geq n^{\epsilon \log n}$).

We now bound $\Pr_\rho[\text{not } E_2]$. Fix $i \in [n], j \in [.3 \log n]$. The probability that for every $k \in [n]$ we have $\rho(x_{i,j,k}) \neq *$ is the probability that a random subset $A \subseteq [m]$ of size $m/m^{.1} = m^{.9}$ does not intersect a fixed subset $B \subseteq [m]$ of size n. This probability is at most the probability that $m^{.9}$ *independent* random elements uniformly distributed in $[m]$ all fall outside B (to see this, think of choosing the random subset A one element at the time, and note that when an element falls outside B it is more likely for the next element to fall inside B). This latter probability is

$$\left(1 - \frac{n}{m}\right)^{m^{.9}} \leq exp(-m^{.9} n/m) \leq exp(-m^{\Omega(1)}) \ll n^{-\Omega(\log n)}$$

where we used that $m = n^2 \cdot (.3 \log n)$. By a union bound we have

$$\Pr[\text{not } E_2] \leq n \cdot (.3 \log n) \cdot n^{-\Omega(\log n)} = n^{-\Omega(\log n)}.$$

□

□

We point out that Theorem 4.4 is tight for the particular choice of

$$f_n(x) = \bigoplus_{i=1}^{n} \bigwedge_{j=1}^{.3 \log n} \bigoplus_{k=1}^{n} x_{i,j,k}.$$

Namely, f_n is computable by Parity $\circ$ And circuits of size $n^{O(\log n)}$. This can be seen by writing the function computed by each And as a Parity of $n^{O(\log n)}$ And's (cf. [RW]).

4.5 Fooling circuits with more arbitrary symmetric gates

In this section we prove our average-case hardness result for constant-depth circuits of size $n^{\epsilon \log n}$ with $\epsilon \log^2 n$ arbitrary symmetric gates (Theorem 4.3). The proof has the same structure as the proof of our average-case hardness result for circuits with *one* arbitrary symmetric gate (Theorem 4.4). The only difference is that now we want to argue that event E_1 happens with high probability even for circuits with $\epsilon \log^2 n$ arbitrary symmetric gates, i.e. we want to show that with high probability over the restriction ρ, the function computed by $C|_\rho$ is computable by a multiparty communication complexity protocol among 'few' parties exchanging 'few' bits. Thus the proof of Theorem 4.3 follows from the next lemma.

Lemma 4.12. *For every constant d there is a constant $\epsilon > 0$ such that the following holds. Let $C : \{0,1\}^n \to \{0,1\}$ be a circuit of size $n^{\epsilon \cdot \log n}$, depth d, with $\epsilon \log^2 n$ arbitrary symmetric gates. Let ρ be a random restriction on the n input variables that assigns * to a subset of the variables of relative size $1/n^{.1}$, i.e. let $\rho \in R_n^{n/n^{.1}}$. Then with probability at least $1 - n^{-\Omega(\log n)}$ over ρ, the function computed by $C|_\rho$ is computable (under any partition of the input) by a $.3 \log n$-party communication complexity protocol exchanging $\log^5 n$ bits of communication.*

Proof. Assume without loss of generality that the output gate of the circuit C is included in the arbitrary symmetric gates. Fix a topological order of the arbitrary symmetric gates (the simple order induced by reading the gates level by level from the inputs to the output node will do). For every $i \in \{1, \ldots, \epsilon \log^2 n\}, z \in \{0,1\}^{i-1}$, define $C_{i,z}$ as the subcircuit of C whose output gate is the i-th arbitrary symmetric gate but where the previous arbitrary symmetric gates are replaced with z (i.e., the j-th gate is replaced with the j-th bit in z). Note $C_{i,z}$ is a $\mathsf{Sym} \circ \mathsf{AC}^0$ circuit.

Claim 4.13. *For a sufficiently small constant $\epsilon > 0$, with probability $1 - n^{-\Omega(\log n)}$ over $\rho \in R_n^{n/n^{.1}}$ we have that for every $i \in \{1, \ldots, \epsilon \log^2 n\}$ and $z \in \{0,1\}^{i-1}$ the function computed by $C_{i,z}|_\rho$ is computable (under any partition of the input) by a $.3 \log n$-party communication complexity protocol $P_{i,z}$ exchanging $\log^3 n$ bits of communication.*

Proof. The claim follows by noting that the number of $\mathsf{Sym} \circ \mathsf{AC}^0$ circuits $C_{i,z}$ is at most

$$\left(\epsilon \log^2 n\right) \cdot 2^{\epsilon \log^2 n} \leq n^{1+\epsilon \log n}$$

and then using a union bound and Lemma 4.9, which states that for each fixed circuit $C_{i,z}$, with probability $1 - n^{-\Omega(\log n)}$ over ρ, the function computed by $C_{i,z}|_\rho$ is computable by a $.3 \log n$-party communication complexity protocol exchanging $\log^3 n$ bits. □

The lemma follows by noting that whenever ρ satisfies the conclusion of the above claim we have (under any partition of the input bits) the following $.3\log n$-party communication complexity protocol P for $C|_\rho$: On input x compute $C|_\rho(x)$ as follows. Simulate P_1 to compute $b_1 = C_1|_\rho(x)$. Then simulate P_{2,b_1} to compute $b_2 = C_{2,b_1}|_\rho(x)$. Then simulate $P_{3,b_1\circ b_2}$ to compute $b_3 = C_{3,b_1\circ b_2}|_\rho(x)$. Continue in this way until $C_{\epsilon\log^2 n,z}(x) = C|_\rho(x)$ (this last equality is easy to verify).

Since each protocol $P_{i,z}$ exchanges at most $\log^3 n$ bits of communication, and we simulate $\epsilon\log^2 n$ of these protocols, the total number of bits exchanged by the protocol P is at most $\log^5 n$. □

It is perhaps interesting to note that, unlike the corresponding protocol in the proof of Theorem 4.4, the protocol in the above lemma is not simultaneous, i.e. the bits sent by a party in general depend on the bits previously sent by other parties (cf. [KN] for background on simultaneous protocols). Thus in our proof we are taking advantage of the fact that the lower bound for *GIP* (Lemma 4.7) holds even for non-simultaneous protocols. We do not know how to prove the same result starting from a multiparty communication complexity lower bound for simultaneous protocols.

4.6 Our PRG vs. Luby, Velickovic, and Wigderson's

In this section we elaborate on why our generator (Theorem 4.1) improves on the generator by Luby, Velickovic, and Wigderson (Theorem 2 in [LVW]). Recall that the generator in [LVW] fools 'small' depth 2 circuits with one arbitrary symmetric gate at the top (Sym ∘ And circuits). On the other hand our generator fools 'small' circuits of any constant depth with 'few' arbitrary symmetric gates.

We note that there are several results (e.g. [Raz, Smo, All, Yao2, BRS, BT]) showing that 'small' circuits in certain 'rich' constant-depth circuit classes can be converted into 'not-too-big' Sym ∘ And circuits. Thus one may wonder whether we can use these results to deduce that the generator in [LVW] is already powerful enough to give our main result (Theorem 4.1), i.e. whether it can fool 'small' constant-depth circuits with 'few' arbitrary symmetric gates.

The problem with this idea is that *in all these conversion results the blow-up in the circuit size is bigger than the saving of the generator*. More specifically, these conversion results show how to convert, say, a AC^0 circuit of size s into a Sym ∘ And circuit of size *quasi*-polynomial, i.e. $s^{\log^{O(1)} s}$, where the constant in the $O(1)$ depends on the depth of the original circuit. However, to fool a circuit of size $s^{\log^{O(1)} s}$, the generator in [LVW] needs a seed of length at least s, and therefore it is of no use in this particular setting.

It seems natural to ask whether the known conversion results are the best possible, i.e. if the quasi-polynomial blow-up is inherent in the conversion. There are works (e.g. [BRS, RW]) suggesting that this is indeed the case. We give another result of this flavor.

Specifically, we show how to modify the lower bound in Theorem 4.4 to get a function computable by polynomial size $\mathsf{Parity} \circ \mathsf{AC}^0$ circuits that is average-case hard for superpolynomial size $\mathsf{Sym} \circ \mathsf{And}$ circuits. The idea is to change the fan-in of the bottom parities of f so that they are computable by polynomial size AC^0 circuits (specifically we change their fan-in from n to $\log^3 n$). While our lower bound is only 'slightly' superpolynomial (i.e. $n^{\Omega(\log\log n)}$), it shows that the parameters of our generator (Theorem 4.1) *cannot* be obtained combining a conversion result with Theorem 2 in [LVW], even if we only want to fool $\mathsf{Parity} \circ \mathsf{AC}^0$ circuits.

Theorem 4.14. *There is a function $f : \{0,1\}^* \to \{0,1\}$ computable by uniform polynomial size* $\mathsf{Parity} \circ \mathsf{AC}^0$ *circuits and a constant $\epsilon > 0$ such that for every n and every* $\mathsf{Sym} \circ \mathsf{And}$ *circuit C of size $n^{\epsilon \cdot \log\log n}$, the following holds:*

$$\Pr_{x \in \{0,1\}^n}[C(x) \neq f(x)] \geq 1/2 - 1/n^{\epsilon \cdot \log\log n}.$$

Proof of Theorem 4.14. The proof follows closely the proof of Theorem 4.4. Let

$$g_n : \{0,1\}^{n(.3 \log n) \log^3 n} \to \{0,1\}$$

be defined as

$$g_n(x) := \bigoplus_{i=1}^{n} \bigwedge_{j=1}^{.3 \log n} \bigoplus_{k=1}^{\log^3 n} x_{i,j,k}.$$

We will prove Theorem 4.14 with g_n as hard function. While g_n is a function on $m = m(n) := n \cdot (.3 \log n) \cdot (\log^3 n)$ bits, it will be convenient to parameterize it by n. Since we will prove $n^{\Omega(\log\log n)}$ lower bounds for g_n and the input length of g_n is $m = n \cdot \mathrm{poly}\log n$, we also obtain $m^{\Omega(\log\log m)}$ lower bounds for g_n (for a different hidden constant in the $\Omega(\cdot)$).

Note that g_n is computable by a (uniform) polynomial size circuit of depth 5 with one Parity gate at the top. To see this note that each of the bottom parities in the definition of g_n is only on $\log^3 n$ bits, and therefore it can be computed by a (uniform) circuit of size $\mathrm{poly}(n)$ and depth 4 (see e.g. [Hås], Theorem 2.2).

Let $C : \{0,1\}^m \to \{0,1\}$ be a circuit as in the statement of Theorem 4.14, for a sufficiently small constant ϵ. Let ρ be a random restriction on the m input variables that assigns * to a subset of the variables of relative size $1/\log(n)$, i.e. let $\rho \in R_m^{m/\log(n)}$.

Consider the following two events.

- Event $E_1' :=$ the function computed by $C|_\rho$ is computable (under any partition of the input) by a $.3 \log n$-party communication complexity protocol exchanging $n^{.2}$ bits.

- Event $E'_2 :=$ for every $i \in [n], j \in [.3 \log n]$ there is $k \in [\log^3 n]$ such that $\rho(x_{i,j,k}) = *$. (In other words, for each of the $n \cdot (.3 \log n)$ bottom parity functions of f_n, ρ maps some of its input variable to *.)

As before, Theorem 4.14 follows from the next claim (cf. the proof of Theorem 4.4).

Claim 4.15. $\Pr_{\rho \in R_m^{m/\log(n)}}[E'_1 \wedge E'_2] \geq 1 - n^{-\Omega(\log \log n)}$.

Proof. We show that E'_1 and E'_2 each do not happen with probability at most $n^{-\Omega(\log \log n)}$.

We now bound $\Pr_\rho[\text{not } E'_1]$. Analogously to the proof of Lemma 4.9, the main step is proving the following claim: with high probability $\left(1 - n^{-\Omega(\log \log n)}\right)$ over $\rho \in R_m^{m/\log(n)}$, the function computed by $C|_\rho$ is computable by a depth 2 circuit of size $|C| \cdot 2^{.3 \log n} = n^{\epsilon \cdot \log \log n} \cdot 2^{.3 \log n}$ with a single symmetric gate (of unbounded fan-in) at the top and And gates of fan-in strictly less than $.3 \log n$ at the bottom.

While this probability can be bound directly, similarly to what is done in [RW], it seems simpler to use again the Switching Lemma. Fix a bottom And gate φ of C, and think of the input variables to φ as clauses of size $r = 1$. By Lemma 4.5 the probability over the choice of ρ that the function computed by $\varphi|_\rho$ cannot be computed by a decision tree of depth strictly less than $s = .3 \log n$ is at most

$$(7pr)^s = (7 \cdot (1/\log(n)) \cdot 1)^{.3 \log n} = n^{-\Omega(\log \log n)}.$$

Therefore if the circuit C has size $n^{\epsilon \log \log n}$ for sufficiently small ϵ we have, by a union bound, that with probability $1 - n^{-\Omega(\log \log n)}$ (over ρ) the function computed by $C|_\rho$ is computable by a symmetric function of $|C|$ decision trees of height strictly less than $.3 \log n$. By Fact 4.6 we can write each decision tree as a DNF and merge the top Or gates of these DNF's into the top symmetric gate of C, and thus E'_1 holds.

We now bound $\Pr_\rho[\text{not } E'_2]$. Fix $i \in [n], j \in [.3 \log n]$. The probability that for every $k \in [\log^3 n]$ we have $\rho(x_{i,j,k}) \neq *$ is is the probability that a random subset $A \subseteq [m]$ of size $m/\log(n)$ does not intersect a fixed subset $B \subseteq [m]$ of size $\log^3 n$. This probability is at most the probability that $m/\log(n)$ *independent* random elements uniformly distributed in $[m]$ all fall outside B (to see this, think of choosing the random subset A one element at the time, and note that when an element falls outside B it is more likely for the next element to fall inside B). This latter probability is

$$\left(1 - \frac{\log^3 n}{m}\right)^{m/\log(n)} \leq exp(-\Omega(\log^2 n)) \ll n^{-\Omega(\log \log n)}.$$

By a union bound we have

$$\Pr[\text{not } E'_2] \leq n \cdot (.3 \log n) \cdot n^{-\Omega(\log \log n)} = n^{-\Omega(\log \log n)}.$$

□

□

4.7 Open problems

Can the techniques in this chapter be used to prove (average-case) hardness results for constant-depth circuits with $\omega(\log^2 n)$ arbitrary symmetric gates? Such a hardness result would follow from a positive answer to the following open question: Let C be constant-depth circuit of size $n^{\epsilon \log n}$ with $\omega(\log^2 n)$ arbitrary symmetric gates, and let ρ be a restriction as in the statement of Lemma 4.9. Is it true that with high probability over ρ the function computed by $C|_\rho$ is computable by a $.9 \log n$-party communication complexity protocol exchanging $n^{.9}$ bits?

Chapter 5

Probabilistic Time versus Alternating Time

We study the power of probabilistic time. In particular, we study the relationship relationship between probabilistic time, $\mathsf{BPTime}(\mathsf{t})$, and alternating time, $\Sigma_{\mathsf{O}(1)}\mathsf{Time}(\mathsf{t})$. This relationship is closely related to the circuit complexity of *Approximate Majority*, which is the problem of computing Majority of a given bit string whose fraction of 1's is bounded away from 1/2 (by a constant).

5.1 Introduction

Understanding the power of probabilistic computation is a central problem in theoretical computer science, and one for which little is known. Essentially, the only non-trivial upper bound that we have on the power of probabilistic computation is the result that probabilistic polynomial time is in the second level of the polynomial-time hierarchy, i.e. $\mathsf{BPP} \subseteq \Sigma_2^P$, which was proven in '83 by Sipser and Gács [Sip], and independently by Lautemann [Lau].[1] The results in [Sip, Lau] actually show that probabilistic time $t = t(n)$ with error 1/3 can be simulated deterministically, using one alternation, with a quadratic blow-up in the running time, i.e. $\mathsf{BPTime}(\mathsf{t}) \subseteq \Sigma_2\mathsf{Time}(\mathsf{t}^2 \cdot \mathrm{poly}\log \mathsf{t})$. To the knowledge of the author, there has been no result on whether this quadratic blow-up is necessary. The question of whether the above quadratic blow-up is necessary is closely related to the circuit complexity of *Approximate Majority*, which is the problem of computing Majority of a given bit string whose fraction of 1's is bounded away from 1/2 (by a constant). In addition to its implications for probabilistic time, the complexity of Approximate Majority is a per se interesting problem which has been widely studied (e.g.,

[1] It is actually known that $\mathsf{BPP} \subseteq \mathsf{MA} \subseteq S_2^P \subseteq \Sigma_2^P$ [Can, RS]. See [GZ] for discussion of these inclusions. These strengthenings are not directly relevant to our work which, unlike [Can, RS, GZ], focuses on the running time of the simulation.

[Ajt1, ABO, Sto, Ajt2, CR]). *In this chapter we prove new results on the circuit complexity of Approximate Majority, and we apply these results to obtain new relationships between probabilistic time and alternating time.*

First, we prove that $\left(2^{n^{.1}}\right)$-size depth-3 circuits for Approximate Majority on n bits have bottom fan-in $\Omega(\log n)$. As a corollary, we obtain that $\mathsf{BPTime}(\mathsf{t}) \not\subseteq \Sigma_2\mathsf{Time}(\mathsf{o}(\mathsf{t}^2))$ with respect to some oracle. This shows that the above mentioned quadratic blow-up in the running time of Σ_2 simulations of $\mathsf{BPTime}(\mathsf{t})$ is in fact necessary for relativizing techniques (such as those in [Sip, Lau]).

The above result naturally raises the question of whether the quadratic blow-up in the running time of the simulation can be avoided at some higher level of the polynomial-time hierarchy. An involved result by Ajtai [Ajt2] implies that this is indeed possible at *some* level, namely that $\mathsf{BPTime}(\mathsf{t}) \subseteq \Sigma_c\mathsf{Time}(\mathsf{t})$ for some constant c. Ajtai does not bound the constant c, and an analysis of his proof only gives a large constant $c \gg 3$. In this chapter, we show that there is a quasilinear-time simulation at level $c = 3$, i.e. we show that $\mathsf{BPTime}(\mathsf{t}) \subseteq \Sigma_3\mathsf{Time}(\mathsf{t} \cdot \mathrm{poly}\log \mathsf{t})$. Our techniques relativize and thus $c = 3$ is optimal for them, as implied by our oracle result stated in the preceding paragraph. Again, the complexity of Approximate Majority is closely related to our result that $\mathsf{BPTime}(\mathsf{t}) \subseteq \Sigma_3\mathsf{Time}(\mathsf{t} \cdot \mathrm{poly}\log \mathsf{t})$. With a slight modification of the techniques used in deriving this result, we also obtain the following: Approximate Majority is computable by *uniform* polynomial-size circuits of depth 3. Prior to our work, the only known polynomial-size depth-3 circuits for Approximate Majority were *non-uniform* [Ajt1]. Tables 5.1 and 5.2 summarize the results discussed so far.

The study of the relationship between probabilistic time and alternating time is also motivated by the challenge of proving lower bounds on the running time of probabilistic algorithms for some 'natural' problem. Of course, it is unknown how to prove superlinear time lower bounds on general computational models (such as multi-tape Turing machines). This is already true for deterministic computation, and probabilistic computation only makes the challenge harder. However, there has been progress in proving lower bounds on restricted models of computation. Much of this progress crucially relies on clever simulations of these restricted models by alternating-time computations. Using our above result that $\mathsf{BPTime}(\mathsf{t}) \subseteq \Sigma_3\mathsf{Time}(\mathsf{t} \cdot \mathrm{poly}\log \mathsf{t})$, among other ideas, we prove new lower bounds for solving $\mathrm{QSAT}_3 \in \Sigma_3\mathsf{Time}(\mathsf{n} \cdot \mathrm{poly}\log \mathsf{n})$ on probabilistic computational models. As it will be apparent from the proofs, our negative result that $\mathsf{BPTime}(\mathsf{t}) \not\subseteq \Sigma_2\mathsf{Time}(\mathsf{o}(\mathsf{t}^2))$ with respect to some oracle shows that substantially different techniques are required to prove similar lower bounds for computing $\mathrm{SAT} = \mathrm{QSAT}_1$ or QSAT_2 (as opposed to QSAT_3 in our results).

We now describe our lower bounds in more detail. The first model we consider is that of probabilistic random-access Turing machines using little space, say at most $n^{.9}$. The works by Beame et al. [BSSV], Allender et al. [AKR$^+$], and Diehl et al. [DvM] prove lower bounds on this model; in particular, Allender et al. [AKR$^+$] and Diehl

Table 5.1: Results on the complexity of computing Approximate Majority on n bits.

Previous Results		
Complexity of Approximate Majority	**Uniformity**	**Reference**
Computable by depth-3 poly(n)-size circuits	non-uniform	[Ajt1]
Computable by depth-$O(1)$ poly(n)-size circuits	Dlogtime-uniform	[Ajt2]
Our Results		
Complexity of Approximate Majority	**Uniformity**	**Reference**
Not computable by depth-3 $2^{n^{.1}}$-size circuits with bottom fan-in $(\log n)/2$	non-uniform	Th. 5.1
Computable by depth-3 poly(n)-size circuits	P-uniform	Th. 5.2

and van Melkebeek [DvM] prove time lower bounds $t = n^{1+\Omega(1)}$ on this model, but their results hold only for machines with *one-way* access to the random bits. We prove a $t = n^{1+\Omega(1)}$ lower bound for solving $\mathrm{QSAT}_3 \in \Sigma_3\mathsf{Time}\,(\mathsf{n}\cdot \mathrm{poly}\log \mathsf{n})$ on machines with *two-way* (sequential) access to the random bits. To the best of our knowledge, this is the first lower bound of the form $t = n^{1+\Omega(1)}$ on a probabilistic model with random access to the input and two-way access to the random bits. (We elaborate on this model in Section 5.1.3.)

We then consider the model of Turing machines with two tapes, i.e. the read-only random-access input tape and one sequential-access work tape with no space restrictions. Maass and Schorr [MS] prove a lower bound of the form $t \geq n^{1.22}$ for simulating $\Sigma_1\mathsf{Time}\,(\mathsf{n})$ on this model. This bound was independently rediscovered in [vMR] where it is also shown that the same bound holds for solving $\mathrm{SAT} \in \Sigma_1\mathsf{Time}\,(\mathsf{n}\cdot \mathrm{poly}\log \mathsf{n})$. While the model in [MS, vMR] is deterministic, we prove that a bound of the form $t = n^{1+\Omega(1)}$ holds, for solving $\mathrm{QSAT}_3 \in \Sigma_3\mathsf{Time}\,(\mathsf{n}\cdot \mathrm{poly}\log \mathsf{n})$, even if the work tape is initialized with random bits and the machine allowed to err with small probability. To the best of our knowledge, no lower bound was previously known on this randomized model.

5.1.1 Our results on the complexity of Approximate Majority

We now describe our results regarding the complexity of *Approximate Majority*. Our main results are summarized and compared to previous work in Table 5.1.

Approximate Majority is a *promise* problem [ESY] where the task is computing Majority of a given bit string that is promised to have either at least a 2/3 fraction of bits set to 1, or at most a 1/3 fraction of bits set to 1. In this chapter we prove the following new lower bound on the *bottom fan-in* of depth-3 (unbounded fan-in)

Table 5.2: Results on simulating probabilistic time by alternating time.

Previous Results		
Inclusion	**Oracle**	**Reference**
$\mathsf{BPTime}(\mathsf{t}) \subseteq \Sigma_2\mathsf{Time}(\mathsf{t}^2 \cdot \mathrm{poly}\log \mathsf{t})$	Holds for every oracle	[Sip, Lau]
$\mathsf{BPTime}(\mathsf{t}) \subseteq \Sigma_{O(1)}\mathsf{Time}(\mathsf{t})$	Holds for every oracle	[Ajt2]
Our Results		
Inclusion	**Oracle**	**Reference**
$\mathsf{BPTime}(\mathsf{t}) \subseteq \Sigma_3\mathsf{Time}(\mathsf{t} \cdot \mathrm{poly}\log \mathsf{t})$	Holds for every oracle	Th. 5.3
$\mathsf{BPTime}(\mathsf{t}) \not\subseteq \Sigma_2\mathsf{Time}(\mathsf{o}(\mathsf{t}^2))$	Holds for some oracle	Th. 5.5

circuits for Approximate Majority, where the bottom fan-in is defined to be the fan-in of the gates adjacent to the input bits.

Theorem 5.1. *Let C be a depth-3 circuit computing approximate majority on n bits. If the bottom fan-in of C is at most $\log(n)/2$ then the size of C is at least $2^{n^{.2}}$, for big enough n.*

We point out that in '83 Ajtai [Ajt1] gave a striking probabilistic construction of non-uniform polynomial-size depth-3 circuits for approximate majority.[2] Ajtai's circuits have bottom fan-in $O(\log n)$, which is optimal up to constant factors by Theorem 5.1. Since Ajtai's construction [Ajt1] is non-uniform, it is natural to ask whether Approximate Majority has uniform polynomial-size depth-3 circuits. We remark that in [Ajt2] Ajtai gives another construction of polynomial-size circuits for Approximate Majority; these circuits are uniform, but they have large constant depth $c > 3$. In this chapter we show that approximate majority is computable by uniform polynomial-size depth-3 circuits.

Theorem 5.2. *Approximate majority is computable by* P*-uniform polynomial-size depth-3 circuits.*

5.1.2 Our results on simulating probabilistic time by alternating time

We now describe our results regarding simulating probabilistic time by deterministic alternating time. Our main results are summarized and compared to previous work in Table 5.2.

On the positive side, we show the following new quasilinear-time simulation of $\mathsf{BPTime}(\mathsf{t})$, which holds in any reasonable model of computation that can compute

[2] While Ajtai [Ajt1] does not explicitly bound the depth of his circuits, it can be verified easily that it equals 3.

Fourier transforms in time $O(n \cdot \mathrm{poly}\log n)$ (see, e.g., [CLRS]). This simulation was independently obtained by Diehl and van Melkebeek (personal communication, Oct. 2005).

Theorem 5.3. $\mathsf{BPTime}(\mathsf{t}) \subseteq \Sigma_3\mathsf{Time}(\mathsf{t} \cdot \mathrm{poly}\log \mathsf{t})$ *for every constructible function* $t = t(n)$.

Since $\mathsf{BPTime}(\mathsf{t})$ is closed under complement, we obtain the following corollary, which plays a crucial role in our lower bounds discussed in sections 5.1.3 and 5.1.4 below.

Corollary 5.4. *If* $\Sigma_3\mathsf{Time}(\mathsf{n}) \subseteq \mathsf{BPTime}(\mathsf{n} \cdot \mathrm{poly}\log(\mathsf{n}))$ *then the quasilinear-time hierarchy collapses to the third level, i.e.* $\bigcup_c \Sigma_c\mathsf{Time}(\mathsf{n} \cdot \mathrm{poly}\log(\mathsf{n})) = \Sigma_3\mathsf{Time}(\mathsf{n} \cdot \mathrm{poly}\log(\mathsf{n}))$.

On the negative side, we prove the following quadratic lower bound on the running time of Σ_2 simulations that *relativize*, i.e. hold with respect to any oracle. We note that all previous simulations [Sip, Lau, Ajt2, Can, RS], as well as ours (Theorem 5.3), relativize.

Theorem 5.5. *For every constructible function* $t = t(n)$, $\mathsf{BPTime}(\mathsf{t}) \not\subseteq \Sigma_2\mathsf{Time}(\mathsf{o}(\mathsf{t}^2))$ *with respect to some oracle.*

Theorem 5.5 shows that, for relativizing techniques, the running time of the Sipser-Gács-Lautemann [Sip, Lau] Σ_2 simulation of $\mathsf{BPTime}(\mathsf{t})$ is optimal (up to logarithmic factors), and that consequently the level of our quasilinear-time simulation in Theorem 5.3 is also optimal, in the sense that it cannot be reduced to 2. For completeness, let us point out that Stockmeyer [Sto] proves that $\mathsf{BPP} \not\subseteq \mathsf{P}^{\mathsf{NP}} \subseteq \Sigma_2^P$ with respect to some oracle. His result is incomparable to our Theorem 5.5 which addresses the running time of Σ_2 simulations.

5.1.3 Our results on time-space lower bounds

We now discuss our results on time-space lower bounds for probabilistic machines. We prove a new time-space lower bound for simulating $\Sigma_3\mathsf{Time}(\mathsf{n})$ and, in particular, for solving QSAT_3, the problem of deciding the validity of a given Boolean first-order formula with at most 2 quantifier alternations. (The results for QSAT_3 will not be stated explicitly but follow from techniques in [FLvMV].) The computational model on which we prove this negative result is that of a probabilistic Turing machine that can access the tape cells on the input and work tapes by writing a logarithmic-sized index on an associated index tape (random access), while it can only move to adjacent tape cells on the random-bit[3] tape in one time step (sequential access).

[3]It will be always clear from the context whether the word 'random' refers to the machine's ability of addressing tape cells by writing down their index, or to the machine being probabilistic.

In what follows we define our computational model, and then state our lower bound.

Definition 5.6. *We denote by* $\overleftrightarrow{\mathsf{BP}}\mathsf{TiSp}\,(t,s)$ *the set of languages accepted by probabilistic Turing machines, with two-sided error, that run simultaneously in time t and space s, with random access to input and work tapes, and two-way sequential access to the random-bit tape.*

Theorem 5.7. *For every constant* $\epsilon > 0$, $\Sigma_3\mathsf{Time}\,(\mathsf{n}) \not\subseteq \overleftrightarrow{\mathsf{BP}}\mathsf{TiSp}\left(n^{1+o(1)}, n^{1-\epsilon}\right)$.

Theorem 5.7 is the first lower bound of the form $t = n^{1+\Omega(1)}$ on a probabilistic computational model with random access to the input tape and *two-way* access to its random bits (for a function computable in, say, linear space). We now elaborate on the strength of models with two-way access to random bits, and then compare our result to the previous ones.

On one-way vs. two-way access to random bits: To appreciate the difference between one-way access and two-way access to random bits, consider log-space computation (L). If one extends L by allowing *one-way* access to random bits, then one gets a complexity class (BPL) that is contained in P. On the other hand, if one allows for *two-way* access to random bits, then one gets a richer complexity class ($\mathsf{BP}\cdot\mathsf{L}$), the power of which is essentially unknown, and conceivably contains NEXP. To further appreciate this difference, we refer to a paper by Nisan [Nis3] which shows that every probabilistic logspace algorithm with one-way access to the random bits can be simulated by a probabilistic logspace algorithm with two-way access to the random bits *with zero error* (i.e., $\mathsf{BPL} \subseteq \mathrm{ZP}\cdot\mathrm{L}$).

An example where two-way access to random bits can be proven to give more power than one-way access is the language of palindromes: it can be recognized in linear time on a sequential one-tape Turing machine with two-way access to the random-bit tape, while it requires time $\Omega(n \cdot \log n)$ if we only allow one-way access to the random-bit tape.[4]

The above discussion begs the question of how large a time bound one can prove on probabilistic computational models with two-way access to the random bits. Our Theorem 5.7 is a qualitatively new answer to this question.

Comparison with previous lower lounds: Beame et al. [BSSV] prove that there is a function in P that requires time $\Omega(n\sqrt{\log(n/s)/\log\log(n/s)})$ on non-uniform randomized branching programs using space at most s. Since branching programs are more powerful than random-access machines, their lower bound applies to our model

[4]The linear-time upper bound can be obtained as follows: On input xy, we accept iff $\langle x, u\rangle = \langle y^R, u\rangle$, where $\langle .,.\rangle$ denotes inner product, y^R the reverse of y, and u the random bits. The lower bound can be derived from the fact that palindromes requires $\Omega(\log n)$ communication for randomized private-coin protocols; see [KN], Example 3.1.

(Def. 5.6), but the time lower bound of $\Omega(n\sqrt{\log n})$ they achieve is weaker than our $n^{1+\Omega(1)}$ bound. Allender et al. [AKR+] prove an $n^{1+\Omega(1)}$ time lower bound on probabilistic random-access machines that have one-way access to random bits and use space at most $n^{1-\epsilon}$, for a function in the counting hierarchy (not believed to be in the polynomial-time hierarchy). In recent and interesting work, Diehl and van Melkebeek [DvM] prove that probabilistic random-access machines that have one-way access to random bits and use space at most n^{ϵ} require time at least $n^{c-\epsilon}$ to simulate $\Sigma_c \mathsf{Time}(\mathsf{n})$, for every constant $c \geq 2$. In [DvM] they also point out that their space bound becomes $n^{.25-\epsilon}$ on machines running in time $n^{1+\epsilon}$. This space bound (for the particular case of machines running in time $n^{1+\epsilon}$) later has been improved to $n^{.5-\epsilon}$ (for simulating $\Sigma_2 \mathsf{Time}(\mathsf{n})$) and to $n^{1-\epsilon}$ (for simulating $\Sigma_3 \mathsf{Time}(\mathsf{n})$). These improvements have been obtained by Diehl and van Melkebeek (personal communication, Oct. 2005), and independently by us.

5.1.4 Our results on time lower bounds

We now discuss our results about time lower bounds on probabilistic machines. We prove a new lower bound on a probabilistic extension of the two-tape Turing machine model (i.e. a Turing machine with a read-only input tape and one sequential-access work tape with no space restrictions). Our probabilistic extension, denoted $\mathsf{BPTime}_1(t)$, is obtained by initializing the work tape of the machine with random bits, and allowing the machine to err with small probability.

Theorem 5.8. $\Sigma_3 \mathsf{Time}(\mathsf{n}) \not\subseteq \mathsf{BPTime}_1\left(n^{1+o(1)}\right)$.

Theorem 5.8 is the first lower bound on the model $\mathsf{BPTime}_1(t)$ (for a function computable in, say, linear space). In fact, as we point out in Appendix 5.6, our lower bound applies to a single model that simultaneously extends $\mathsf{BPTime}_1\left(n^{1+o(1)}\right)$ and the previously considered $\overleftrightarrow{\mathsf{BP}}\mathsf{TiSp}\left(n^{1+o(1)}, n^{1-\epsilon}\right)$ (Def. 5.6).

5.1.5 Organization

This chapter is organized as follows. In Section 5.2 we prove our lower bound for approximate majority (Theorem 5.1), and we prove that it implies our oracle separation in Theorem 5.5. In Section 5.3 we prove that $\mathsf{BPTime}(\mathsf{t}) \subseteq \Sigma_3 \mathsf{Time}\left(\mathsf{t} \cdot \log^2 \mathsf{t}\right)$ (Theorem 5.3). In Section 5.4 we prove that Approximate Majority is computable by uniform polynomial-size circuits of depth 3 (i.e., Theorem 5.2). In Section 5.5 we prove our time-space lower bound (Theorem 5.7). Section 5.6 contains the proof of our time lower bound on Turing machines (Theorem 5.8). Section 5.7 discusses the dependence of our results on the error probability ϵ of the $\mathsf{BPTime}(\mathsf{t})$ machines, while we set $\epsilon = 1/3$ in the rest of the chapter. In Section 5.8 we mention a few open problems.

5.2 Lower bound for Approximate Majority

In this section we prove our lower bound on the bottom fan-in of (unbounded fan-in) depth-3 circuits computing approximate majority. At the end of the section we compare our techniques to previous ones, and informally discuss why our lower bound implies our oracle separation in Theorem 5.5.

Let us begin by formally defining approximate majority and restating our main result.

Definition 5.9. Approximate Majority *is the following promise problem:*

$$\begin{aligned} \text{ApprMaj}_{YES} &:= \{x : \textit{ at least } 2|x|/3 \textit{ bits of } x \textit{ are set to } 1\}, \\ \text{ApprMaj}_{NO} &:= \{x : \textit{ at most } |x|/3 \textit{ bits of } x \textit{ are set to } 1\}. \end{aligned}$$

Theorem (5.1, restated)**.** *Let C be a depth-3 circuit computing approximate majority on n bits. If the bottom fan-in of C is at most* $\log(n)/2$ *then the size of C is at least* $2^{n^{.2}}$*, for big enough n.*

We now explain the proof of Theorem 5.1. It is convenient to work with the following distribution, which generates $x \in \text{ApprMaj}_{NO}$ with sufficiently high probability (for our purposes).

Definition 5.10. *Let D^n be the distribution on $\{0,1\}^n$ that sets each bit to* 1 *independently with probability* 1/3 *(and to* 0 *with probability* 2/3*).*

The core of the proof of Theorem 5.1 is the following lemma.

Lemma 5.11. *Let φ be a* DNF *on n variables with terms of size at most k. If $\varphi(x) = 1$ for every $x \in \{0,1\}^n$ with at least $2n/3$ bits set to* 1*, then* $\Pr_{x \in D^n}[\varphi(x) = 0] \leq e^{-\frac{n}{3 \cdot k^2 \cdot 3^k}}$.

Before we explain the intuition for the proof of Lemma 5.11, let us see why it implies Theorem 5.1.

Proof of Theorem 5.1 assuming Lemma 5.11. First note that $\Pr_{x \in D^n}[x \in \text{ApprMaj}_{NO}] \geq 1/3$ by the standard Central Limit Theorem,[5] and therefore we have that $\Pr_{x \in D^n}[C(x) = 0] \geq 1/3$.

Assume that the output gate of C is an And gate (otherwise we can negate the circuit and carry through essentially the same argument as below for approximate majority with YES and NO swapped). So $C = \bigwedge_i \varphi_i$, where each φ_i is a Or-And depth-2 circuit, i.e. a DNF. Note that, by definition of And, for every $x \in \text{ApprMaj}_{YES}$ we have $\varphi_i(x) = 1$ for all i, while for random $x \in D^n$ we have that with probability at least 1/3 there is an i such that $\varphi_i(x) = 0$. By an averaging argument we can fix a DNF $\varphi = \varphi_i$ such that: (1) for every $x \in \text{ApprMaj}_{YES}$ we have $\varphi(x) = 1$; (2) $\Pr_{x \in D^n}[\varphi(x) = 0] \geq 1/(3 \cdot |C|)$, where $|C|$ is the size of the circuit C, i.e. the number

[5] Alternatively, one can change the parameter 1/3 in the distribution D^n to any other smaller constant, and prove, using Markov inequality, a bound which is enough to obtain our results modulo different constants.

of its gates. Note that if C has bottom fan-in at most k then the same holds for the subcircuit φ fixed above, i.e. φ is a DNF with term size at most k. By Lemma 5.11, we have

$$\frac{1}{3 \cdot |C|} \leq e^{-n/(3 \cdot k^2 \cdot 3^k)} = e^{-n^{\Omega(1)}},$$

and so we conclude $|C| \geq e^{n^{\Omega(1)}}$. In fact, $|C| \geq 2^{n^{.2}}$, because $3^k = n^{(\log_2 3)/2} \approx n^{.792}$. □

Overview of the proof of Lemma 5.11: The proof of Lemma 5.11 is an inductive argument inspired by a recent switching lemma by Segerlind et al. [SBI], which we discuss later in this section. A similar argument is also a component of a technically intricate lower bound on the round complexity of protocols for two-party random selection [SV2].

Let us define a *covering* of a DNF φ as a subset Γ of the variables such that each term in φ contains at least one variable, possibly negated, in Γ. For example, the smallest covering of the DNF $\varphi(x_1, x_2, x_3, x_4) := (x_1 x_2 x_3) \vee (\neg x_1) \vee (x_2 x_4)$ is $\Gamma := \{x_1, x_2\}$. To prove Lemma 5.11 we then argue as follows. Consider the smallest covering Γ of φ. There are two cases: Either $|\Gamma| > n/(3k)$ or $|\Gamma| \leq n/(3k)$. (Recall $k = (\log n)/2$ is the maximum term size of φ.)

Case $|\Gamma| > n/(3k)$: In this case φ must contain at least $|\Gamma|/k = n/\operatorname{poly}\log(n)$ *disjoint* terms, where disjoint means that no two terms share a variable. Such terms can be found greedily, using the fact that the terms are of size at most k. Now, the probability that $\varphi(x) = 0$ for random $x \in D^n$ is at most the probability that all these $n/\operatorname{poly}\log(n)$ terms evaluate to 0. Since the terms are disjoint, this can be bound by raising to the power of $n/\operatorname{poly}\log(n)$ the probability that a single term is 0. But since terms have at most $k = (\log n)/2$ variables, the probability that a term is 0 can be shown to be at most $(1 - 1/n^\epsilon)$, for a constant $\epsilon < 1$. Thus we have

$$\Pr_{x \in D^n}[\varphi(x) = 0] \leq \left(1 - \frac{1}{n^\epsilon}\right)^{n/\operatorname{poly}\log(n)} = 2^{-n^{\Omega(1)}},$$

which proves Lemma 5.11.

Case $|\Gamma| \leq n/(3k)$: In this case by an averaging argument we fix the variables in Γ and obtain a new DNF φ' such that $\Pr_{x \in D^n}[\varphi'(x) = 0] \geq \Pr_{x \in D^n}[\varphi(x) = 0]$. Then we iterate the argument on φ' (i.e. we consider the size of the smallest covering of φ', etc.).

We keep iterating the argument until we obtain a DNF φ' whose smallest covering has size at least $n/(3k)$, or a DNF φ' that computes a constant function. By construction, $\Pr_{x \in D^n}[\varphi'(x) = 0] \geq \Pr_{x \in D^n}[\varphi(x) = 0]$, so we only need to worry about the case $\varphi' \equiv 0$ (otherwise Lemma 5.11 is proven as stated above). We rule out the case $\varphi' \equiv 0$ by exhibiting $x \in \mathrm{ApprMaj}_{YES}$ such that $\varphi(x) = 0$, which contradicts the hypothesis of Lemma 5.11. To construct such an x, note that each iteration assigns values to the variables in a covering of the DNF, and so at each iteration the term size of the DNF decreases (since each term has at least one variable in the covering). Therefore we iterate the argument at most k times. Since each iteration fixes at most $n/(3k)$ variables, in the end we have fixed at most $k \cdot n/(3k) = n/3$ variables. By setting all the remaining variables to 1 we set at least $2n/3$ variables to 1 and thus we have $x \in \mathrm{ApprMaj}_{YES}$ such that $\varphi(x) = \varphi'(x) = 0$, which contradicts the hypothesis of Lemma 5.11 and concludes this proof sketch.

We now present the formal proof of Lemma 5.11.

Proof of Lemma 5.11. We define a *covering* Γ of a DNF φ to be a subset of variables such that each term in φ contains at least one variable in Γ (possibly negated). Consider the following procedure:

Procedure(φ)
If φ computes a constant function then *stop*. Remove all terms in φ that compute the constant function 0 (e.g. $x_1 \wedge \neg x_1$). Let Γ be a minimum-size covering of the terms of φ. If $\lvert\Gamma\rvert \geq n/(3 \cdot k)$ then *stop*. Partition the v variables of $\varphi(x)$ into $x = y \bigcup z$, where $y = \Gamma$ and $z \bigcap \Gamma = \emptyset$. Fix an assignment $y = a$ such that $\Pr_{z \in D^{v-\lvert\Gamma\rvert}}[\varphi(a \cdot z) = 0] \geq \Pr_{x \in D^v}[\varphi(x) = 0]$. (Such an assignment exists by an averaging argument.) Consider the new DNF $\varphi'(z) := \varphi(a \cdot z)$ obtained by hardwiring a in φ. Repeat the procedure on the DNF φ'.

Claim 5.12. *The procedure stops after at most k iterations.*

Proof. At each iteration we fix the variables y in a covering Γ of φ. Since by definition of covering each term of φ contains a variable from Γ, this decreases the maximum term size of φ. When the term size is 0 then the DNF is a constant and we stop. □

The procedure constructs a sequence of DNF's $\varphi = \varphi_1, \varphi_2, \ldots, \varphi_t$, where DNF φ_i is on n_i variables. We have $t \leq k$ by the above claim. Also, by construction we have

$$\Pr_{x \in D^n}[\varphi(x) = 0] = \Pr_{x \in D^{n_1}}[\varphi_1(x) = 0] \leq \Pr_{x \in D^{n_2}}[\varphi_2(x) = 0] \leq \ldots \leq \Pr_{x \in D^{n_t}}[\varphi_t(x) = 0]. \tag{5.1}$$

To finish the proof, we simply analyze what happens when the procedure stops. If the procedure stops at the i-th iteration because φ_i computes a constant function, we argue as follows: If $\varphi_i(x) = 1$ for every x then $\Pr_{x \in D^{n_i}}[\varphi_i(x) = 0] = 0$, and by Equation (5.1) we have $\Pr_{x \in D^n}[\varphi(x) = 0] \leq 0$, which proves the lemma.

If $\varphi_i(x) = 0$ for every x then notice that at each iteration we fix at most $n/(3k)$ variables, and therefore by Claim 5.12 φ_i equals φ with at most $k \cdot n/(3k) = n/3$ variables fixed. By setting all the remaining variables to 1, we have found an input $\hat{x}$ with at least $2n/3$ bits set to 1 such that $\varphi(\hat{x}) = 0$, which contradicts the hypothesis of the lemma.

Otherwise, the procedure stops at the i-th iteration because every covering of φ_i has size at least $n/(3k)$. Therefore there is a set S of at least $n/(3k^2)$ disjoint terms, where disjoint means that no two terms share a variable. To see why this is true, let S be a *maximal* set of disjoint terms. The union of the variables in the terms in S is a covering of φ_i of size at most $|S| \cdot k$: if it were not a covering, we could add a term to S, contradicting its maximality. Thus $|S| \geq (n/3k)/k$.

Then we can bound $\Pr_x[\varphi(x) = 0]$ as follows:

$$\Pr_{x \in D^n}[\varphi(x) = 0] \leq \Pr_{x \in D^{n_i}}[\varphi_i(x) = 0] \leq \left(1 - \frac{1}{3^k}\right)^{n/(3k^2)} \leq e^{-\frac{n}{3 \cdot k^2 \cdot 3^k}},$$

where the first inequality follows by Equation (5.1), and the second follows by considering the terms on disjoint sets of variables in φ_i. Specifically, we notice that the probability that any of them is 0 is at most $(1 - 1/3^k)$. This holds because in the procedure we always remove terms computing the constant function 0, and thus each term must be 1 under at least one assignment of its (at most k) variables. This assignment is picked under the distribution D^{n_i} with probability at least $1/3^k$. Moreover, these events are independent for the $n/(3 \cdot k^2)$ terms with disjoint variables. □

Why one cannot use previous techniques to prove our circuit lower bound: A standard approach to prove lower bounds for constant-depth circuits is to use a *switching lemma* (see, e.g., [Hås]). Håstad's switching lemma [Hås] cannot be used directly to prove lower bounds for the promise problem Approximate Majority. This is because to apply this lemma to a circuit with bottom fan-in k, we need to assign values to at least a $(1 - 1/k)$ fraction of the variables. The switching lemma would assign 0 to roughly half of this fraction of variables, and so as soon as $k \geq 3$, we would produce an input $x \notin \mathrm{ApprMaj}_{YES} \bigcup \mathrm{ApprMaj}_{NO}$.

Recently, Segerlind et al. [SBI] proved a new switching lemma that assigns values to much fewer variables than does Håstad's switching lemma. One can apply this switching lemma [SBI] to prove that small depth-3 circuits for approximate majority on n bits require bottom fan-in at least $\Omega(\sqrt{\log n})$; however, we were unable to apply their results to circuits with bigger bottom fan-in, such as $(\log n)/2$. Our proof is also conceptually simpler than a proof based on the switching lemma.

5.2.1 Oracle separation

For completeness, in this section we prove why why our circuit lower bound for approximate majority in Theorem 5.1 implies our oracle separation in Theorem 5.5. We start with some intutition and then we present a formal proof.

Intuition behind the oracle separation: Consider simulating a probabilistic machine $M \in \mathsf{BPTime}(\mathsf{t})$ by a machine in $\Sigma_2\mathsf{Time}(\mathsf{t}')$. Given an input x, by definition of $\mathsf{BPTime}(\mathsf{t})$ we are *promised* that either $\Pr_u[M(x;u) = 1] \geq 2/3$ or $\Pr_u[M(x;u) = 1] \leq 1/3$. This corresponds to an approximate majority instance Y of exponential length $|Y| = 2^{|u|} = 2^t$, where the i-th bit of Y is $M(x;i)$ (the equality $2^{|u|} = 2^t$ holds because the machine runs time t). Thus, intuitively, the task of the Σ_2 machine is to distinguish $Y \in \mathrm{ApprMaj}_{YES}$ from $Y \in \mathrm{ApprMaj}_{NO}$. By standard techniques reviewed in Appendix B, the Σ_2 computation can be seen as an unbounded fan-in circuit of depth 3: the two quantifiers give rise to the first two levels of the circuit (Or-And). After its quantifications, the Σ_2 simulation is followed by a deterministic computation running in time t'. Since computing $M(x;u)$ for fixed $(x;u)$ takes (deterministic) time t, the Σ_2 computation can depend on at most $k := t'/t$ evaluations of M. We can write this part of the Σ_2 computation as a depth-2 (And-Or) circuit with bottom fan-in k. Finally, by collapsing the top And gate of this circuit with the And arising from the second quantifier, we obtain a small circuit of depth 3 with bottom fan-in $k = t'/t$. By our lower bound (Theorem 5.1), small depth-3 circuit for Approximate Majority on $|Y|$ bits have bottom fan-in $\Omega(\log |Y|) = t$, and therefore we obtain that $t' \geq \Omega(t^2)$.

The above sketch can be shown to be exact for relativizing simulations, using the standard convention that the oracle tape is erased after each query (see, e.g., [BDG]). We stress that this convention is natural: it is intuitively capturing the fact that the $\Sigma_2\mathsf{Time}(\mathsf{t}')$ machine cannot run the code of the $\mathsf{BPTime}(\mathsf{t})$ machine more than t'/t times, since each execution of the latter runs in time t. Finally, we would like to point out that our result in Theorem 5.5 also applies to simulations of $\mathsf{BPTime}(\mathsf{t})$ by $\Sigma_2\mathsf{Time}(\mathsf{t}')$ that are *black-box* (as opposed to relativizing), because black-box simulations relativize (folklore).

We now present a formal proof.

Proof of Theorem 5.5 assuming Theorem 5.1. The overall structure of the proof follows standard arguments (see e.g., [FSS], or Theorem 14.5 in [Pap]). Fix a time bound $t = t(n)$. Consider the collection of oracles $\mathcal{A}$ where for every $\mathrm{A} \in \mathcal{A}$ and for every integer m, the fraction of strings of length m in the oracle A is at least $2/3$ or at most $1/3$. For $\mathrm{A} \in \mathcal{A}$ we consider the language

$$L^{\mathrm{A}} := \left\{ 0^n : \Pr_{x:|x|=t(n)}[x \in \mathrm{A}] \geq 2/3 \right\}.$$

For every $\mathrm{A} \in \mathcal{A}$ we have that $L^{\mathrm{A}} \in \mathsf{BPTime}(\mathsf{t})$ with oracle access to A: on input 0^n, we simply query the oracle at a random x of length $t(n)$ and return its answer.

On the other hand, we show that there is $\mathrm{A} \in \mathcal{A}$ such that $L^{\mathrm{A}} \notin \Sigma_2\mathsf{Time}(\mathsf{o}(\mathsf{t}^2))$ with oracle access to A. The construction of A proceeds by diagonalization. Consider an enumeration $M_1, M_2, \ldots$ of $\Sigma_2\mathsf{Time}(\mathsf{o}(\mathsf{t}^2))$ machines. We assume that these machines only make oracle queries of length $t(n)$, on input of length n. This assumption can be removed by standard arguments, e.g. by considering 'sufficiently sparse' input lengths. We make this assumption because it simplifies the proof while preserving all the main ideas, and also because it holds for all known simulations (in particular all those in Table 5.2).

At stage i we fix a finite initial segment of the oracle A in such a way that $L^{\mathrm{A}} \neq M_i^{\mathrm{A}}$. Stage i works as follows. Let M_i run in time $t' = o(t^2)$. By standard techniques reviewed in Appendix B the machine M_i gives rise to an unbounded fan-in circuit of depth 3 as follows. The input to the circuit is the truth table of length $T = T(n) := 2^t$ of the oracle (as well as the bit-wise complement of this truth table). The top and the middle fan-ins of the circuit are $2^{O(t'(n))}$ (where the top fan-in is defined as the fan-in of the output gate), while the bottom fan-in is $k = k(n) := t'(n)/t(n) = o(t^2/t) = o(t)$. The bottom fan-in corresponds to the maximum number of oracle queries the machine makes on any computation path. Since each query is of length t and the oracle tape is erased after each query (see, e.g., [BDG]), the machine can only ask $t'(n)/t(n)$ queries.

We are now in the apply our Theorem 5.1. For sufficiently large n, a circuit with the above parameters cannot compute approximate majority on $T = 2^t$ bits. (Note that the bottom fan-in of the circuit is $k = o(t) = o(\log T)$.) Therefore we can augment the oracle A in such a way that $L^{\mathrm{A}} \neq M_i^{\mathrm{A}}$, while still keeping $\mathrm{A} \in \mathcal{A}$. $\square$

5.3 $\mathsf{BPTime}(\mathsf{t}) \subseteq \Sigma_3\mathsf{Time}(\mathsf{t} \cdot \mathrm{poly}\log \mathsf{t})$

In this section we prove that $\mathsf{BPTime}(\mathsf{t}) \subseteq \Sigma_3\mathsf{Time}(\mathsf{t} \cdot \mathrm{poly}\log \mathsf{t})$, i.e. Theorem 5.3. The ideas in the proof of this result will also be a component of our proof that Approximate Majority has uniform polynomial-size circuits of depth-3, i.e. Theorem 5.2, though more work is needed for that (see Section 5.4). We start with some intuition and then we present a formal proof.

Intuition for $\mathsf{BPTime}(\mathsf{t}) \subseteq \Sigma_3\mathsf{Time}(\mathsf{t} \cdot \mathrm{poly}\log \mathsf{t})$: Let us focus on the case $t = n$ and review Lautemann's proof of $\mathsf{BPTime}(\mathsf{n}) \subseteq \Sigma_2\mathsf{Time}(\mathsf{n}^2 \cdot \mathrm{poly}\log \mathsf{n})$, as this is the starting point of our argument. Let $M(x; u)$ be a probabilistic machine using n random bits u. Lautemann's approach is to *guess* n 'shifts' $w_1 \in \{0,1\}^n, w_2 \in \{0,1\}^n, \ldots, w_n \in \{0,1\}^n$ and check if *for every* $u \in \{0,1\}^n$ we have

$$\Big(M(x; u \oplus w_1) = 1\Big) \bigvee \Big(M(x; u \oplus w_2) = 1\Big) \bigvee \ldots \bigvee \Big(M(x; u \oplus w_n) = 1\Big). \quad (5.2)$$

Noting that we use two quantifiers, each ranging over at most n^2 bits, and that the computation in Equation 5.2 takes time n^2, we have that this is a $\Sigma_2\mathsf{Time}(\mathsf{n}^2)$ simulation. The proof of correctness is a counting argument, which we omit.[6]

[6] Actually, this approach requires that the error probability of M is at most $1/n^2$. We can achieve this by paying an extra $O(\log n)$ factor in the running time, if M starts off with error 1/3. We ignore this issue to simplify the exposition.

There are two reasons why this simulation takes at least quadratic time. The first is that the computation in Equation (5.2) runs M for n times (over fixed random bits). Since M runs in time $O(n)$, this takes at least time n^2. The second reason is that we initially guess $n^2 = |w_1, w_2, \ldots, w_n|$ bits.

Let us focus on the first problem, that is, the fact that computing Equation (5.2) requires quadratic time. This problem *cannot* be avoided using relativizing techniques: Our negative result (Theorem 5.5) shows that every relativizing Σ_2 simulation *must* run M at least $\Omega(n)$ times, and thus must have total run time at least n^2. As a first step towards our quasilinear Σ_3 simulation, we observe that the computation in Equation (5.2) is an Or over n evaluations of M; thus, we can use another quantifier for this Or, and then run M once.

To obtain a Σ_3 simulation that runs in quasilinear time, we still have to solve the second problem, the quantification over $w_1, w_2, \ldots, w_n$. As it turns out, the only property of these w_i's that is used in Lautemann's proof is a *hitting* property, i.e. for any 'big' set $A \subseteq \{0,1\}^n$, the probability over random $w_1, w_2, \ldots, w_n$ that none of the w_i's lands in A is exponentially small in n. It is well known that there are more 'randomness-efficient' ways to generate w_i's with this property (see, e.g., [Gol2]). In particular, it is possible to generate such w_i's using a *hitting* generator with a seed of length $|\sigma| = O(n)$. We use this approach: instead of guessing n^2 bits for $w_1, w_2, \ldots, w_n$, we only guess $O(n)$ bits σ and let $w_i := G(\sigma)_i$, where $G(\sigma) = G(\sigma)_1 \cdot G(\sigma)_2 \cdots G(\sigma)_n \in (\{0,1\}^n)^n$ for an appropriate generator G.

However, for our simulation we need a generator that runs in quasilinear time in the sense that given σ and i we need to compute the i-th output $G(\sigma)_i = w_i \in \{0,1\}^n$ of the generator in quasilinear time. Well-known generators based on *random walks on expander graphs* achieve seed length $O(n)$ (see, e.g., [Gol2]), but we do not know how to compute any of them in less than quadratic time.[7]

Instead, we use a generator by Nisan [Nis2] which has slightly worse seed length $|\sigma| = O(n \cdot \log n)$, but which on the other hand can be computed in time $O(n \cdot \operatorname{poly}\log n)$: to compute one output of the generator we only have to evaluate $\log n$ *pairwise independent hash functions* $h : \{0,1\}^n \to \{0,1\}^n$. Using hash functions based on convolution or finite field arithmetic, one can compute each such hash function in time $O(n \cdot \operatorname{poly}\log n)$ using the Fast Fourier Transform.

We now present a formal proof.

Theorem (5.3, restated)**.** $\mathsf{BPTime}(\mathsf{t}) \subseteq \Sigma_3\mathsf{Time}(\mathsf{t} \cdot \operatorname{poly}\log \mathsf{t})$ *for every constructible function $t = t(n)$.*

The proof of Theorem 5.3 uses as a component a hitting generator by Nisan.

[7]Concurrently with our work, Diehl and van Melkebeek show how to compute walks on the Margulis-Gabber-Galil expander graph [Mar, GG, JM] in quasilinear time (personal communication, Oct. 2005).

Lemma 5.13 ([Nis2], Theorem 3). *For every r and $k \le 2^r$, there exists a generator*

$$N_k : \{0,1\}^l \to (\{0,1\}^r)^k\,,\ \ N_k(\sigma) = N_k(\sigma)_1 \cdot N_k(\sigma)_2 \cdots N_k(\sigma)_k,$$

such that:

1. *N_k has seed length $l = |\sigma| = O(r \log k)$,*
2. *for every set $A \subseteq \{0,1\}^r$ we have*
$$\left| \Pr_\sigma[\forall i \le k : N_k(\sigma)_i \in A] - \left(\frac{|A|}{2^r}\right)^k \right| \le 4^{-r},$$
3. *given a seed σ and $i \le k$ we can compute $N_k(\sigma)_i \in \{0,1\}^r$ in time $(r \cdot \operatorname{poly}\log r) \log k$.*

Proof. Items (1) and (2) in Lemma 5.13 are proven in Nisan's paper [Nis2]. To prove Item (3), we note that computing one r-bit output of the generator requires computing $\log k$ pairwise independent hash functions. Using hash functions based on convolution or finite field arithmetic one can compute each such hash function in time $O(r \cdot \operatorname{poly}\log r)$ using the Fast Fourier Transform (see, e.g., [Nis2] Section 2.2 for the definition of hash functions based on convolution, and Theorem 30.8 in [CLRS] for the use of the Fast Fourier Transform to compute convolution). □

Proof of Theorem 5.3. We prove $\mathsf{BPTime}(\mathsf{n}) \subseteq \Sigma_3\mathsf{Time}(\mathsf{n} \cdot \operatorname{poly}\log \mathsf{n})$, the inclusion for generic time bound $t = t(n)$ then follows by a standard padding argument (see, e.g., [Pap]). Let M be an algorithm in $\mathsf{BPTime}(\mathsf{n})$ and let us write $M(x;u)$ for algorithm M on input x and random bits u.

Claim 5.14. *Let M be an algorithm in $\mathsf{BPTime}(\mathsf{n})$. There is an algorithm M' that accepts the same language as M such that: M' has error probability at most $1/n^2$, M' uses $O(n)$ random bits, and given x, u we can compute $M'(x;u)$ in time $O(n \cdot \log n)$.*

We postpone the proof of Claim 5.14 and we proceed with the proof of the theorem. (Note that if one is willing to have M' use $O(n \log n)$ random bits, instead of $O(n)$, then one can obtain such a M' by simply taking the majority of $O(\log n)$ repetitions of $M(x)$ with independent random bits. Using this instead of Claim 5.14 and proceeding with the proof gives a simulation with worse running time, but still $n \cdot \operatorname{poly}\log n$. However, the better parameters achieved here are used later in Section 5.4.)

Let M' be the algorithm from Claim 5.14 and let $r = r(n) = O(n)$ be the number of random bits used by M' on input of length n. Let $N_k : \{0,1\}^l \to (\{0,1\}^r)^k$ be the generator from Lemma 5.13 with $k = r$. Note that the seed length of N_k is $|\sigma| = l = O(r \log k) = O(n \cdot \log n)$.

Now consider the Σ_3 machine that accepts input x if and only if

$$\exists \sigma \in \{0,1\}^l\, \forall u \in \{0,1\}^r\, \exists i \le k : M'(x; N_k(\sigma)_i \oplus u) = 1,$$

where $\oplus$ denotes bitwise xor.

Correctness: Assume $M'(x) = 1$. We must show that the Σ_3 machine accepts. Consider

$$\Pr_{\sigma}[\exists u \in \{0,1\}^r \, \forall i \leq k : M'(x; N_k(\sigma)_i \oplus u) = 0]. \tag{5.3}$$

We show that this probability is less than 1 and therefore that the machine accepts. By a union bound this probability (5.3) is at most

$$2^r \cdot \Pr_{\sigma}[\forall i \leq k : M'(x; N_k(\sigma)_i) = 0].$$

(Note we removed $\oplus u$ because this just shifts the space of random bits of the algorithm and does not change the probability.) Since $M'(x) = 1$, and M' has error at most $1/n^2$, we get by Lemma 5.13 that the probability is at most (recall $k = r$)

$$2^r \left((1/n^2)^k + 4^{-r}\right) < 1.$$

Now assume that $M'(x) = 0$. Fix any seed σ, we must show that $\exists u \in \{0,1\}^r \forall i \leq k : M'(x; N_k(\sigma)_i \oplus u) = 0$, and thus the machine rejects. Consider

$$\Pr_{u}[\exists i \leq k : M'(x; N_k(\sigma)_i \oplus u) = 1.$$

Again, we show that this probability is less than 1 and therefore that the machine accepts. By a union bound this probability is at most

$$k \cdot \Pr_{u}[M'(x; u) = 1].$$

(Note again we removed $N_k(\sigma)_i\oplus$ because this just shifts the space of random bits of the algorithm and does not change the following analysis.) Since M' has error at most $1/n^2$ this probability is at most $k/n^2 = r/n^2 = O(n \cdot \log n)/n^2 < 1$.

Complexity: The machine uses three quantifiers, each on at most $O(n \cdot \log n)$ bits, by inspection. Each computation branch only runs $M'(x; N_k(\sigma)_i \oplus u)$ once. $N_k(\sigma)_i$ can be computed in time $(r \cdot \operatorname{poly}\log r) \cdot \log k = n \cdot \operatorname{poly}\log n$ by Lemma 5.13, and for given x, u', $M'(x; u')$ can be computed in time $O(n \cdot \log n)$ by Claim 5.14. □

Proof sketch of Claim 5.14. We use the $O(n)$ random bits of M' to encode a walk $w_1, w_2, \ldots, w_l$ of length $l = O(\log n)$ on an expander graph of $2^{O(n)}$ vertices. We then define $M'(x; u) := \mathrm{Maj}_{j \leq l} M(x; w_j)$. The bound on the error probability of the algorithm follows by the Chernoff Bound for random walks on expander graphs [Gil]. Using the expander graph in [Mar] (the expansion of which is analyzed in [GG, JM]), we can compute all the w_i's in time $O(n \cdot \log n)$: computing w_i from w_{i-1} amounts to a constant number of additions, which can be done in time $O(n)$. □

5.4 Uniform depth-3 circuits for Approximate Majority

In this section we prove the following theorem regarding the complexity of computing approximate majority.

Theorem (5.2, restated)**.** *Approximate majority is computable by* P*-uniform polynomial-size depth-*3 *circuits.*

The proof of Theorem 5.2 makes use of the following hitting generator (see Section 5.3 for a discussion of hitting generators).

Lemma 5.15. *For every* n *there exists a polynomial-time computable generator*

$$Z : \{0,1\}^{O(n)} \to (\{0,1\}^n)^k\,,\;\; Z(\sigma) = Z(\sigma)_1 \cdot Z(\sigma)_2 \cdots Z(\sigma)_k,$$

with $k = O(n/\log n)$*, such that for every set* $A \subseteq \{0,1\}^n$ *of size* $|A| \le 2^n/n^2$ *we have*

$$\Pr_\sigma[\forall i \le k : Z(\sigma)_i \in A] \le 4^{-n}.$$

Proof sketch of Lemma 5.15. Use the input σ to encode a random walk (started at a random vertex) of length $c \cdot n/\log n$ on an $n^{O(1)}$-regular expander graph on 2^n vertices with second largest eigenvalue $\lambda = 1/n^2$, for a constant c to be determined later.[8] Note that one can encode such a walk using $(c \cdot n/\log n) \cdot \log n^{O(1)} = O(n)$ bits. Kahale [Kah] shows that the probability that all the t steps of the random walk fall inside a fixed set of density $\mu := 1/n^2$ can be bounded as

$$\mu \cdot (\mu + (1-\mu)\cdot\lambda)^{t-1} \le (1/n^{\Omega(1)})^{c \cdot n/\log n} \le 4^{-n},$$

where the last inequality holds by choosing a sufficiently large constant c. □

We now proceed with the proof of Theorem 5.2.

Proof of Theorem 5.2. The main ideas of the proof are the same as those of the proof of Theorem 5.3. It is convenient to think of an approximate majority instance of length N as $M(0) \cdot M(1) \cdots M(N)$, where $M(i)$ is the i-th bit of the instance.

Let $Z : \{0,1\}^{O(n)} \to (\{0,1\}^n)^k$ be the hitting generator from Lemma 5.15, where $k = O(n/\log n)$. We decide approximate majority by checking whether the following equation is true:

$$\exists \sigma \in \{0,1\}^{O(n)}\, \forall u \in \{0,1\}^n\, \exists i \le k : M'\left(Z(\sigma)_i \oplus u\right) = 1, \tag{5.4}$$

[8] Such an expander G can be obtained by taking a λ-biased set $S \subseteq \{0,1\}^n$ of size $\mathrm{poly}(n)$ [NN, AGHP] and setting $G := (\{0,1\}^n, \{(x,y) : x - y \in S\})$. Alternatively, we can take a $O(\log n)$ power of a constant-degree expander.

where $\oplus$ denotes bitwise xor, and M' denotes the machine with error $1/n^2$ from Claim 5.14, which we think of as an approximate majority instance with amplified gap (i.e., either at least a $1 - 1/n^2$ fraction of input bits is set to 1, or at most a $1/n^2$ fraction of input bits is set to 1).

Correctness: The proof of correctness is the same as that of Theorem 5.3.

Computable by uniform $\text{poly}(N)$*-size circuits of depth* 3*:* We note that the computation of $M'(G(\sigma)_i \oplus u)$ is the majority of $O(\log n)$ evaluations of M, and therefore the computation of

$$\exists i \leq k : M'(G(\sigma)_i \oplus u) = 1$$

only depends on $k \cdot O(\log n) = O(n)$ evaluations of M. We write this computation as a CNF of size $2^{O(n)} = \text{poly}(N)$. We then collapse the output And of this CNF with the second quantifier in Equation (5.4). Finally, since the first two quantifiers in Equation (5.4) range over $O(n)$ bits, they give rise to gates with fan-in $2^{O(n)} = \text{poly}(N)$, and thus the whole computation in Equation (5.4) can be written as a $\text{poly}(N)$-size circuit of depth 3.

It is easy to check that the circuit can be constructed in time $\text{poly}(N)$. (Note that the generator Z in Lemma 5.15 is computable in time $\text{poly}(n) = \text{poly} \log N$.) $\square$

On Depth-3 circuits vs. Σ_3 time: The above proof of Theorem 5.2 is similar to the proof of Theorem 5.3 (which is the result that $\mathsf{BPTime}(\mathsf{t}) \subseteq \Sigma_3\mathsf{Time}(\mathsf{t} \cdot \text{poly} \log \mathsf{t})$), and thus it is worth pointing out why the two results (Theorem 5.2 and Theorem 5.3) actually are incomparable.

The depth-3 circuit in the above proof of Theorem 5.2 is P-uniform, i.e. it can be constructed in time polynomial in its size. P-uniformity is too loose to obtain Theorem 5.3 for which one needs the circuit to satisfy the following stronger uniformity condition: given the indices to two gates in the circuit, it is possible to decide whether the two gates are connected in time quasilinear in the length of their indices. It is conceivable that the circuit does satisfy this stronger uniformity condition, but this does not seem straightforward to us.

Conversely, the circuit obtained in Theorem 5.3 (i.e. $\mathsf{BPTime}(\mathsf{t}) \subseteq \Sigma_3\mathsf{Time}(\mathsf{t} \cdot \text{poly} \log \mathsf{t})$) does satisfy the above stronger uniformity condition, but has superpolynomial size and depth 4. (The circuit has depth 4 because the 3 alternations are followed by a computation that depends on several evaluations of the $\mathsf{BPTime}(\mathsf{t})$ machine. This computation gives rise to another layer of gates in the circuit.)

5.5 Time-space lower bound

In this section we prove our time-space lower bound (Theorem 5.7). We start with an informal overview of the techniques, and then proceed with a formal proof sketch.

Overview of techniques: Our time-space lower bound for $\Sigma_3\mathsf{Time}(\mathsf{n})$ is inspired by an interesting recent paper by Diehl and van Melkebeek [DvM] which, among other results, proves that $\Sigma_3\mathsf{Time}(\mathsf{n}) \not\subseteq \overrightarrow{\mathsf{BP}}\mathsf{TiSp}(n^{1+\epsilon}, n^{\epsilon})$ for some constant ϵ, where $\overrightarrow{\mathsf{BP}}\mathsf{TiSp}(t, s)$ denotes the class of problems that can be solved simultaneously in time t and space s on a probabilistic random-access Turing machine with one-way access to its random bits. It is convenient for this exposition to think of the approach in [DvM] as giving a sublinear-time Σ_3 simulation of $\overrightarrow{\mathsf{BP}}\mathsf{TiSp}(n^{1+\epsilon}, n^{\epsilon})$, in other words, proving that $\overrightarrow{\mathsf{BP}}\mathsf{TiSp}(n^{1+\epsilon}, n^{\epsilon}) \subseteq \Sigma_3\mathsf{Time}(\mathsf{o}(\mathsf{n}))$. Their result that $\Sigma_3\mathsf{Time}(\mathsf{n}) \not\subseteq \overrightarrow{\mathsf{BP}}\mathsf{TiSp}(n^{1+\epsilon}, n^{\epsilon})$ then follows by a standard time hierarchy for alternating time.[9] To prove this simulation one argues as follows. First one *derandomizes* the $\overrightarrow{\mathsf{BP}}\mathsf{TiSp}(n^{1+\epsilon}, n^{\epsilon})$ machine using a pseudorandom generator by Nisan [Nis2] that has seed length n^{ϵ}. This gives $\overrightarrow{\mathsf{BP}}\mathsf{TiSp}(n^{1+\epsilon}, n^{\epsilon}) \subseteq \mathsf{BP}^{n^{\epsilon}}\mathsf{TiSp}(n^{1+\epsilon}, n^{\epsilon})$, where '$\mathsf{BP}^{n^{\epsilon}}$' means 'using at most n^{ϵ} random bits.' (Note we do not specify anymore if the machine has one-way or two-way access to its random bits, because now the machine simply could copy its n^{ϵ} random bits onto a random-access work tape.) Second, one replaces the n^{ϵ} random bits by two quantifiers, using Lautemann's result [Lau] discussed in Section 5.1. For this replacement, Lautemann's result needs to quantify over $(n^{\epsilon})^2$ bits, and thus one obtains $\mathsf{BP}^{n^{\epsilon}}\mathsf{TiSp}(n^{1+\epsilon}, n^{\epsilon}) \subseteq \exists^{n^{2\epsilon}}\forall^{n^{2\epsilon}}\mathsf{TiSp}\left(n^{1+O(\epsilon)}, n^{\epsilon}\right)$. By using another quantifier to speed up the running time of the simulation (cf. [vM]), one gets $\overrightarrow{\mathsf{BP}}\mathsf{TiSp}(n^{1+\epsilon}, n^{\epsilon}) \subseteq \Sigma_3\mathsf{Time}(\mathsf{o}(\mathsf{n}))$.

We now explain the difficulties we encounter in the proof of our result that $\Sigma_3\mathsf{Time}(\mathsf{n}) \not\subseteq \overleftrightarrow{\mathsf{BP}}\mathsf{TiSp}\left(n^{1+o(1)}, n^{1-\epsilon}\right)$. Following the above outline, our task is to show the following simulation: $\overleftrightarrow{\mathsf{BP}}\mathsf{TiSp}(n^{1+\epsilon}, n^{1-\epsilon}) \subseteq \Sigma_3\mathsf{Time}(\mathsf{o}(\mathsf{n}))$. The main difficulty that we face in this extension to machines with two-way access to random bits is that in this setting only generators much weaker than Nisan's [Nis2] are known. Specifically, we use a generator by Impagliazzo et al. [INW], which raises two problems. The first problem is that, regardless of the amount of space used by the machine, this generator always has seed length $|\sigma| \gg \sqrt{n}$ (as opposed to n^{ϵ}). This is problematic because, as explained above, the approach in [DvM] is to replace the random bits for the seed σ of the generator by alternations. However, Lautemann's approach would need to quantify over $|\sigma|^2 \gg n$ bits, which would prevent us from obtaining a sublinear-time simulation. We solve this problem using our quasilinear-time simulation of probabilistic time (Theorem 5.3) instead of Lautemann's simulation.

The second problem is that we do not know how to compute the generator of [INW] in less than time $|\sigma|^2 \gg n$. Again, this seems to prevent us from obtaining a sublinear-time simulation. In fact, the argument in [DvM] directly exploits the fact that Nisan's generator [Nis2] is computable quickly. We solve this problem as follows. First we notice that the generator in [INW] can be computed time-efficiently *using*

[9] This is a simplification of the techniques in [DvM]: in [DvM] they obtain the stronger result that $\Sigma_3\mathsf{Time}(\mathsf{n}) \not\subseteq \overrightarrow{\mathsf{BP}}\mathsf{TiSp}\left(n^{3-\epsilon}, n^{\epsilon}\right)$ using a more involved argument.

alternations. Then we apply our Corollary 5.4, which shows that our assumption ($\Sigma_3\mathsf{Time}(\mathsf{n}) \subseteq \overleftrightarrow{\mathsf{BP}}\mathsf{TiSp}\left(n^{1+o(1)}, n^{1-\epsilon}\right) \subseteq \mathsf{BPTime}\left(\mathsf{n}^{1+o(1)}\right)$) implies a time-efficient collapse of the alternating-time hierarchy to the third level.

We now restate our time-space lower bound and then prove it.

Definition (5.6, restated)**.** *We denote by $\overleftrightarrow{\mathsf{BP}}\mathsf{TiSp}(t, s)$ the set of languages accepted by probabilistic Turing machines, with two-sided error, that run simultaneously in time t and space s, with random access to input and work tapes, and two-way sequential access to the random-bit tape.*

Theorem (5.7, restated)**.** *For every constant $\epsilon > 0$, $\Sigma_3\mathsf{Time}(\mathsf{n}) \not\subseteq \overleftrightarrow{\mathsf{BP}}\mathsf{TiSp}\left(n^{1+o(1)}, n^{1-\epsilon}\right)$.*

We use the following generator by Impagliazzo, Nisan, and Wigderson.

Lemma 5.16 ([INW], Theorem 4)**.** *For every sufficiently small constant $\epsilon > 0$ and sufficiently large n, there exists a pseudorandom generator $G : \{0,1\}^{m^{1-\delta}} \to \{0,1\}^{m^{1+\delta}}$, with $m = m(n) := n^2$ and $\delta = \epsilon/4$, that satisfies the following:*

- Pseudorandomness: *Let M be a machine in $\overleftrightarrow{\mathsf{BP}}\mathsf{TiSp}(t, m^{1-\epsilon})$ where $t = t(n) = m^{1+o(1)}$. For every fixed $x \in \{0,1\}^n$,*

$$\left| \Pr_{u \in \{0,1\}^t}[M(x; u) = 1] - \Pr_{\sigma \in \{0,1\}^{m^{1-\delta}}}[M(x; G(\sigma)) = 1] \right| \leq 1/9.$$

- Complexity: *Given a seed $\sigma \in \{0,1\}^{m^{1-\delta}}$ and the index $i \leq m^{1+\delta}$ to an output bit, the i-th output bit of $G(\sigma)$ is computable simultaneously in time* $\mathrm{poly}(m)$ *and space $O(m^{1-\delta})$.*

Remark 5.17 (Remark on the proof of Lemma 5.16)**.** *In [INW] they sketch a proof that one can fool Turing machines running simultaneously in time t and space s using a generator G with seed length $O(\sqrt{t \cdot s} \cdot \log t)$. We now explain why this implies Lemma 5.16 above.*

First note that in our case $t = t(n) = m^{1+o(1)}$ and $s = m^{1-\epsilon}$. When $t \leq m^{1+\delta}$ then the generator in [INW] has seed length $O(\sqrt{m^{1+\delta+1-\epsilon}} \cdot \log m) = O(m^{1-3\cdot\epsilon/8} \cdot \log m) \leq m^{1-\epsilon/4} = m^{1-\delta}$ for sufficiently large m.

In [INW], they only argue that the generator fools Turing machines (as opposed to $\overleftrightarrow{\mathsf{BP}}\mathsf{TiSp}(t, s)$). However, their proof is only exploiting that the machine has sequential access to the random-bit tape, and that the state of the machine can be described by s bits, both of which hold in our model $\overleftrightarrow{\mathsf{BP}}\mathsf{TiSp}(t, s)$. In particular, their proof does not rely on Turing machines being a uniform model of computation, and thus their generator works even after we hardwire an arbitrary input x.

Finally, the authors of [INW] claim without proof (Section 4 in [INW]) that the generator can be computed "in polynomial time and polylog space in the size of the output of the generator." We do not see how to do that when the seed is polynomially related to the size of the output of the generator. However, it is not too hard to see that, given a seed, the generator is computable simultaneously in polynomial time and linear space (details omitted).

Notation for the proof of Theorem 5.7: The proof of Theorem 5.7 involves a series of inclusions between complexity classes. To reason about these classes, it is convenient to give the following definitions. We denote by $\mathsf{TiSp}\,(t,s)$ the set of (languages accepted by) Turing machines running simultaneously in time t and space s, with random access to input and work tapes (cf. [vM, FLvMV]). For a complexity class C (e.g. $\mathsf{TiSp}\,(t,s)$) and a function $f = f(n)$ we define the class $\mathsf{BP}^f\mathsf{C}$ to be the set of languages L for which there exists $M(x;u) \in \mathsf{C}$ such that $x \in L \Rightarrow \Pr_{u\in\{0,1\}^f}[M(x;u)=1] \geq 2/3$, and $x \notin L \Rightarrow \Pr_{u\in\{0,1\}^f}[M(x;u)=1] \leq 1/3$, where the complexity of M is measured in terms of $|x|$ (as opposed to $|x|+|u|$). We analogously define $\exists^f\mathsf{C}$ (namely $x \in L \Leftrightarrow \exists u \in \{0,1\}^f : M(x;u)=1$), and $\forall^f\mathsf{C}$.

Proof of Theorem 5.7. We assume that $\Sigma_3\mathsf{Time}\,(\mathsf{n}) \subseteq \overleftrightarrow{\mathsf{BP}}\mathsf{TiSp}\left(n^{1+o(1)}, n^{1-\epsilon}\right)$, and derive a contradiction to the time hierarchy for alternating time, namely $\Sigma_3\mathsf{Time}\,(\mathsf{m}) \subseteq \Sigma_3\mathsf{Time}\,(\mathsf{o(m)})$ (see, e.g., Section 3.1 in [vM]). Let $m = m(n) := n^2$ (any $m = n^{1+\Omega(1)}$ would do). We have the following contradiction:

$$\begin{aligned}
\Sigma_3\mathsf{Time}\,(\mathsf{m}) &\subseteq \overleftrightarrow{\mathsf{BP}}\mathsf{TiSp}\left(m^{1+o(1)}, m^{1-\epsilon}\right) && (5.5)\\
&\subseteq \mathsf{BP}^{m^{1-\delta}}\mathsf{TiSp}\left(\mathrm{poly}(m), m^{1-\delta}\right) && (5.6)\\
&\subseteq \exists^{m^{1-\delta/2}}\forall^{m^{1-\delta/2}}\exists^{m^{1-\delta/2}}\mathsf{TiSp}\left(\mathrm{poly}(m), m^{1-\delta/2}\right) && (5.7)\\
&\subseteq \Sigma_{O(1)} && (5.8)\\
&\subseteq \Sigma_3\mathsf{Time}\,(\mathsf{o(m)}) && (5.9)\\
&\ \text{contradiction.} && (5.10)
\end{aligned}$$

Inclusion (5.5) holds by assumption plus a padding argument (see, e.g., [Pap]).

Inclusion (5.6) follows by using the INW generator from Lemma 5.16. More specifically, let M be a machine in $\overleftrightarrow{\mathsf{BP}}\mathsf{TiSp}\left(m^{1+o(1)}, m^{1-\epsilon}\right)$ and let $G : \{0,1\}^{m^{1-\delta}} \to \{0,1\}^{m^{1+\delta}}$ be the generator from Lemma 5.16. Now consider the machine M' that uses $m^{1-\delta}$ random bits σ, and simulates the machine M, but whenever M accesses the i-th random bit, M' computes on the fly the i-th output bit of $G(\sigma)$ and uses that instead. By reusing the same space to compute $G(\sigma)$ every time M accesses a random bit, and because G is computable simultaneously in time poly(m) and space $O(m^{1-\delta})$ by Lemma 5.16, we have that $M' \in \mathsf{BP}^{m^{1-\delta}}\mathsf{TiSp}\left(\mathrm{poly}(m), m^{1-\delta}\right)$. Also, for every input x we have $M(x) = M'(x)$ by the pseudorandomness property of INW in

Lemma 5.16 (assuming without loss of generality that the error probability of M is a sufficiently small constant).

Inclusion (5.7) follows by (a variant of) Theorem 5.3. Specifically, it is easy to check that Theorem 5.3 can be extended to obtain the following inclusion, which implies Inclusion (5.7): For any $t = t(n)$ and $r = r(n) \leq t(n)$ polynomially related to n, we have

$$\mathsf{BP}^r\mathsf{TiSp}\,(t, r) \subseteq \exists^{r'}\forall^{r'}\exists^{r'}\mathsf{TiSp}\,(t', r')\,,$$

where $r' = r \cdot \operatorname{poly}\log(n)$ and $t' = t \cdot \operatorname{poly}\log(n)$.

Inclusion (5.8) follows by the fact, usually credited to [Nep], that one can trade alternations for time in sublinear-space computations. More formally, one can prove the following inclusion (see e.g. [vM], Section 3.2):

$$\mathsf{TiSp}\,(t, s) \subseteq \Sigma_{2k}\mathsf{Time}\left((\mathsf{t} \cdot \mathsf{s}^{\mathsf{k}})^{1/(\mathsf{k}+1)}\right).$$

(The number of alternations in the above inclusion can be reduced to $k + 1$ from $2k$ [FLvMV]. This gives a better constant for the $\Omega(1)$ term in our lower bound $n^{1+\Omega(1)}$.)

Inclusion (5.9) follows by a collapse similar to Corollary 5.4. Specifically, by assumption we have that $\Sigma_3\mathsf{Time}\,(\mathsf{n}) \subseteq \overleftrightarrow{\mathsf{BP}}\mathsf{TiSp}\left(n^{1+o(1)}, n^{1-\epsilon}\right) \subseteq \mathsf{BPTime}\left(\mathsf{n}^{1+o(1)}\right)$. Since probabilistic time is closed under complement, by Theorem 5.3 we get that $\mathsf{BPTime}\left(\mathsf{n}^{1+o(1)}\right) \subseteq \Pi_3\mathsf{Time}\left(\mathsf{n}^{1+o(1)}\right)$. Combining the two things we get

$$\Sigma_3\mathsf{Time}\,(\mathsf{n}) \subseteq \Pi_3\mathsf{Time}\left(\mathsf{n}^{1+o(1)}\right). \tag{5.11}$$

Intuitively, this means that whenever we have a computation with three quantifiers we can complement them only paying a subpolynomial blow up in the running time, which gives the collapse. More formally, consider a computation in $\Sigma_{O(1)}\mathsf{Time}\left(\mathsf{m}^{1-\delta/4}\right)$. We can assume that $1-\delta/4 \geq 2/3$ without loss of generality, and thus the $\Sigma_{O(1)}\mathsf{Time}\left(\mathsf{m}^{1-\delta/4}\right)$ machine runs in time $m^{1-\delta/4} \geq n^{4/3} \geq n$. Consequently, Equation (5.11) implies that

$$\Sigma_3\mathsf{Time}\left(\mathsf{m}^{1-\delta/4}\right) \subseteq \Pi_3\mathsf{Time}\left((\mathsf{m}^{1-\delta/4})^{1+o(1)}\right)$$

by padding. Applying this padded Equation (5.11) a constant number of times and collapsing adjacent quantifiers, we obtain that $\Sigma_{O(1)}\mathsf{Time}\left(\mathsf{m}^{1-\delta/4}\right)$ collapses to $\Sigma_3\mathsf{Time}\,(\mathsf{m}')$, where m' is $m^{1-\delta/4}$ raised, a constant number of times, to an exponent that is $1 + o(1)$, and so $m' = m^{(1-\delta/4)(1+o(1))} = m^{1-\Omega(1)} = o(m)$.

The contradiction (5.10) is obtained by diagonalization (see, e.g., Section 3.1 in [vM]). □

5.6 Lower bound on stronger computational models

In this section we prove our time lower bound on two-tape Turing machines (Theorem 5.8). Rather than working directly in the model $\mathsf{BPTime}_1\,(t)$ (introduced in

Section 5.1.4), which is incomparable to the previously considered space-bounded model (Def. 5.6), we proceed by extending the space-bounded model so that it includes $\mathsf{BPTime}_1(t)$ as a special case, and we prove a lower bound on this stronger model. Specifically, we extend $\overleftrightarrow{\mathsf{BP}}\mathsf{TiSp}(t, s)$ (Def. 5.6) by allowing machines to write on the random-bit tape, and denote this extension by $\overleftrightarrow{\mathsf{BP}}_w\mathsf{TiSp}(t, s)$. By ignoring the space-bounded random-access tapes, we see that $\overleftrightarrow{\mathsf{BP}}_w\mathsf{TiSp}(t, s)$ includes the model $\mathsf{BPTime}_1(t)$ as a special case. Let us formally define $\overleftrightarrow{\mathsf{BP}}_w\mathsf{TiSp}(t, s)$ and then state our lower bound.

Definition 5.18. *We denote by* $\overleftrightarrow{\mathsf{BP}}_w\mathsf{TiSp}(t, s)$ *the extension of* $\overleftrightarrow{\mathsf{BP}}\mathsf{TiSp}(t, s)$ *obtained by allowing machines to write on the random-bit tape. In other words, we denote by* $\overleftrightarrow{\mathsf{BP}}_w\mathsf{TiSp}(t, s)$ *the set of languages accepted by probabilistic Turing machines, with two-sided error, running in time* t*, which have the following tapes:*

- *a random-access input tape,*
- *several random-access work tapes for at most* s *bits, and*
- *one sequential-access two-way read-write tape that is initially filled with random bits.*

Theorem 5.19. *For every constant* $\epsilon > 0$, $\Sigma_3\mathsf{Time}(\mathsf{n}) \not\subseteq \overleftrightarrow{\mathsf{BP}}_w\mathsf{TiSp}\left(n^{1+o(1)}, n^{1-\epsilon}\right)$*, and in particular* $\mathrm{QSAT}_3 \notin \overleftrightarrow{\mathsf{BP}}_w\mathsf{TiSp}\left(n^{1+o(1)}, n^{1-\epsilon}\right)$.

The outline of the proof of Theorem 5.19 is similar to that of the proof of Theorem 5.7 (cf. Section 5.5). The two main differences are: (1) we need to observe that the INW generator in Lemma 5.16 also works if the machine is allowed to write on the random-bit tape, and (2) we need to show that it is still possible to simulate the machine using small space (and quantifiers). The observation (1) is in Remark 5.20 below, and the simulation (2) is proved in Claim 5.21 using ideas from a simulation by Maass and Schorr [MS] which was recently rediscovered by van Melkebeek and Raz [vMR].

Remark 5.20 (On the pseudorandomness property of the INW generator)**.** *We point out that the INW generator in Lemma 5.16 also fools machines that are allowed to write on the sequential two-way random bits tape, i.e.* $\overleftrightarrow{\mathsf{BP}}_w\mathsf{TiSp}(t, s)$ *machines. While this extension is not explicitly stated in [INW], it can be obtained using the techniques in [INW]. Roughly, their communication-complexity simulation in the proof of Theorem 5 in [INW] (which is only stated for machines with one tape) can be extended to machines with several space-bounded work tapes, by having Alice and Bob also communicate all the contents of the random-access work tapes (details omitted).*

Proof of Theorem 5.19. The outline of the proof follows closely that of the proof of Theorem 5.7. Let $m = m(n) := n^2$. We have the following contradiction:

$$\begin{aligned}
\Sigma_3\mathsf{Time}\,(\mathsf{m}) &\subseteq \overleftrightarrow{\mathsf{BP}}_w\mathsf{TiSp}\left(m^{1+o(1)}, m^{1-\epsilon}\right) && (5.12)\\
&\subseteq \mathsf{BP}^{m^{1-\Omega(1)}}\exists^{m^{1-\Omega(1)}}\mathsf{TiSp}\left(\mathrm{poly}(m), m^{1-\Omega(1)}\right) && (5.13)\\
&\subseteq \exists^{m^{1-\Omega(1)}}\forall^{m^{1-\Omega(1)}}\exists^{m^{1-\Omega(1)}}\mathsf{TiSp}\left(\mathrm{poly}(m), m^{1-\Omega(1)}\right) && (5.14)\\
&\subseteq \Sigma_{O(1)}\mathsf{Time}\left(\mathsf{m}^{1-\Omega(1)}\right) && (5.15)\\
&\subseteq \Sigma_3\mathsf{Time}\,(\mathsf{o(m)}) && (5.16)\\
&\ \text{contradiction.} && (5.17)
\end{aligned}$$

Inclusions (5.12), (5.14)-(5.16), and the final contradiction 5.17 can be obtained exactly as in the proof of Theorem 5.7. The next claim proves Inclusion (5.13) and thus concludes the proof of the theorem.

Claim 5.21. *Let $M(x;u)$ be a machine in $\overleftrightarrow{\mathsf{BP}}_w\mathsf{TiSp}\left(m^{1+o(1)}, m^{1-\epsilon}\right)$ where $\epsilon > 0$ is any fixed constant and $m = m(n) := n^2$. Let $G : \{0,1\}^{m^{1-\delta}} \to \{0,1\}^{m^{1+\delta}}$ be the generator from Lemma 5.16. Then the language $\{(x;\sigma) : M(x;G(\sigma)) = 1\}$ is in $\exists^{m^{1-\Omega(1)}}\mathsf{TiSp}\left(\mathrm{poly}(m), m^{1-\Omega(1)}\right)$. In particular,*

$$\overleftrightarrow{\mathsf{BP}}_w\mathsf{TiSp}\left(m^{1+o(1)}, m^{1-\Omega(1)}\right) \subseteq \mathsf{BP}^{m^{1-\Omega(1)}}\exists^{m^{1-\Omega(1)}}\mathsf{TiSp}\left(\mathrm{poly}(m), m^{1-\Omega(1)}\right).$$

Proof. Let $t = m^{1+o(1)}$ be the running time of the machine M, $s = m^{1-\epsilon}$ the random-access space used by M, and let us denote by τ the (two-way) read-write sequential tape of M, which we will initially fill with the pseudorandom bits $G(\sigma)$. We assume that τ has tape squares numbered as $0, 1, 2, \ldots$, and that M starts with the head on tape square 0 (the same proof carries through for the case where τ has tape squares numbered as $\ldots, -2, -1, 0, 1, 2, \ldots$).

We use the idea of *crossing sequences*, (see, e.g., [vMR] for background). Let us divide τ in consecutive blocks where the first block is of size d and all the others are of size $b := m^{1-\epsilon/3}$. The parameter d will be specified later. Note b and d completely specify this subdivision in blocks. For fixed b, d, let us define a *crossing* to be a pair $(a \to a', S)$ where a is an index to a block in τ, $a' = a \pm 1$ and S completely specifies the state of the machine M *except* the contents of τ. More specifically, S specifies the content of the random-access tapes, the position of the head on the input tape, and the internal state of the machine. As usual, we think of crossing $(a \to a', S)$ as meaning that M is crossing the boundary of block a towards block a' while being in state S (but let us stress again that S does not specify the content of τ).

Consider the computation $M(x;G(\sigma))$ that runs in time t. Since different values of $d \in \{0, 1, \ldots, b-1\}$ give rise to disjoint sets of blocks, it is not hard to see that there must exist $d < b$ such that the number of crossings induced by the computation $M(x;G(\sigma))$ is at most $t/b = m^{1+o(1)-1+\epsilon/3} \leq m^{\epsilon/2}$.

Simulation: We simulate the machine M as follows: First we guess a shift d and a sequence of t/b crossings. Then we check consistency of the computation. For this, let us divide the space of the simulation in two parts: a current-block part of b bits and a current-state part of s bits (jumping ahead, note $b+s = m^{1-\epsilon/3}+m^{1-\epsilon} = m^{1-\Omega(1)}$).

As a base case, we initialize the bits in current-block with $G(\sigma)_0 \cdot G(\sigma)_1 \cdots G(\sigma)_{d-1}$, and we simulate the machine, using current-state as its random-access space and current-block as the first block of τ, until it crosses the block boundary towards block 1 (which starts at tape square d). (If the machine never crosses this boundary then the simulation is easy.) When that happens, we look at the first crossing $(a' \to a'', S')$ in the guessed list and check that $a' = 0, a'' = 1$, and current-state equals S'.

Next, for every block number i, we proceed as follows. First we initialize the bits in current-block with $G(\sigma)_{d+ib} \cdot G(\sigma)_{d+ib+1} \cdots G(\sigma)_{d+ib+(b-1)}$. Then we scan, in order, the list of guessed crossings. Whenever we encounter a crossing of the form $(a \to i, S)$ we do the following. We copy S on current-state and we simulate M (using current-state as the random-access space of M) until it crosses the block boundary towards block a'. Then we look at the next crossing $(\alpha \to \alpha', S')$ in the guessed list and check that $\alpha = i$, $\alpha' = a'$ and current-state equals S'. We continue in this way until the list is over.

Finally, we check that the last crossing corresponds to the accepting state of M. (Without loss of generality we can assume that M, if it accepts, it does so by entering its accepting state while crossing a block boundary.)

Correctness: It is easy to verify that the simulation accepts if and only if M does.

Complexity: Note that each crossing can be specified by $O(m^{1-\epsilon})$ bits, and therefore we guess at most $O(m^{1-\epsilon+\epsilon/2}) = m^{1-\Omega(1)}$ bits total. The rest of the computation can be done simultaneously in time poly(m) and space $m^{1-\Omega(1)}$. To see this, note that the INW generator is computable in these resources by Lemma 5.16, while the rest of the simulation only uses space for the current-block part of b bits and the current-state part of s bits, which sum up to $b+s = m^{1-\epsilon/3}+m^{1-\epsilon} = m^{1-\Omega(1)}$. (The simulation is easily seen to run in polynomial time.)

The 'in particular' part of the claim follows by the first part of the claim plus the pseudorandomness property of the INW generator in Lemma 5.16 (cf. remark 5.20). (I.e. we use $m^{1-\Omega(1)}$ random bits for the seed σ of the INW generator G.) □

□

5.7 On the error parameter

In this section we discuss the dependence of our results on the error probability ϵ of the $\mathsf{BPTime}(\mathsf{t})$ machines.

While we proved our results only for $\mathsf{BPTime}(\mathsf{t})$ machines with error $\epsilon = 1/3$, our results immediately generalize to any constant error $0 < \epsilon < 1/2$. In fact, they also apply to more general error parameters $\epsilon = \epsilon(t)$ as we now discuss. For this discussion, let $\epsilon-$ApprMaj denote the promise problem of computing Majority on an input bit string whose fraction of 1's is promised to be either at least $1-\epsilon$ or at most ϵ. (Note that the previously considered approximate majority equals $1/3-$ApprMaj.) For the case where the error ϵ is shrinking (e.g. $\epsilon := 1/t^2$) it is possible to generalize our Theorem 5.1 to show that computing $\epsilon-$ApprMaj on n bits requires bottom fan-in at least $\Omega(\log(n)/\log(1/\epsilon))$, which implies that any relativizing Σ_2 simulation of $\mathsf{BPTime}\,(\mathsf{t})$ must have running time at least $\Omega(t^2/\log(1/\epsilon))$ (e.g. $\Omega(t^2/\log t)$ when $\epsilon = 1/t^2$). In particular, polynomially small error does not give polynomial savings in the running time, and even in this case our results show that the running time of previous simulations [Sip, Lau] is optimal up to polylogarithmic factors for relativizing techniques.

On the other hand, for large error $\epsilon = 1/2 - \alpha$ where $\alpha = o(1)$, one can use Håstad's switching lemma [Hås] (using a restriction that leaves free a $\Omega(\alpha)$ fraction of bits) to argue that small depth-3 circuits for $\epsilon-$ApprMaj need bottom fan-in at least $\Omega(1/\alpha)$ (details omitted). Again, this implies that any relativizing Σ_2 simulation of $\mathsf{BPTime}\,(\mathsf{t})$ must have running time at least $\Omega(t/\alpha)$ (which is only better than Theorem 5.5 when $\alpha = o(1/t)$). This latter bound is met, up to a polynomial factor, by first applying standard error reduction techniques to bring the error down to, say, $\epsilon = 1/3$, and then plugging in previous results [Sip, Lau].

5.8 Open problems

In this section we list a few open problems.

1. Prove an $n^{1+\Omega(1)}$ time lower bound on probabilistic Turing machines using space $O(\log n)$ with random access to both the input tape and the random-bit tape (for a function computable in linear space). Our results (Theorem 5.19) only apply to models with sequential access to the random-bit tape.

2. Prove a superlinear time lower bound on probabilistic Turing machines with random-access to the input, two-way sequential access to the work tape with no space restrictions, and one-way access to the random-bit tape (for a function computable in linear space). Our results (Theorem 5.19) only apply to the weaker model where the work tape is initialized with random bits.

Chapter 6

The Complexity of Decoding

6.1 Introduction

In this chapter we study the complexity of decoding error-correcting codes. Specifically, we are interested in the complexity of the decoding procedures that are associated with black-box hardness amplification. Before motivating this study and presenting our contributions, let us recall the notion of black-box hardness amplification.

Definition 6.1. *A* $[.25 \to 1/2-\epsilon]$-black-box hardness amplification $Amp : \{0,1\}^n \to \{0,1\}$ *for size* s *and length* k *is a map from functions* $f : \{0,1\}^k \to \{0,1\}$ *to functions* $Amp^f : \{0,1\}^n \to \{0,1\}$ *that satisfies the following: for every* $f : \{0,1\}^k \to \{0,1\}$ *and* $R : \{0,1\}^n \to \{0,1\}$ *such that*

$$\Pr_{x \in \{0,1\}^n} \left[R(x) \neq Amp^f(x)\right] < 1/2 - \epsilon$$

there is a "decoder" circuit D *of size at most* s *such that*

$$\Pr_{x \in \{0,1\}^k} \left[D^R(x) \neq f(x)\right] < .25.$$

(In the above definition we set $\delta = .75$ for simplicity. It is also useful to think of $k = n^{\Omega(1)}$.) The rationale behind the above definition is that if the function f is .25-hard for circuits of size g (e.g., $g = k^{\omega(1)}$) then Amp^f is $(1/2-\epsilon)$-hard for circuits of size g/s. To show this, we argue by contradiction as follows. Suppose that there is a circuit R of size g/s that computes Amp^f on a $(1/2+\epsilon)$-fraction of the inputs. By definition of black-box hardness amplification there is a circuit D of size s such that D^R computes f on a .75 fraction of the inputs. Since the size of D^R is at most $s \cdot g/s = g$, this contradicts our assumption that f is .25-hard for circuits of size s.

While we do not know of any explicit function (e.g. in NEXP) that is even worst-case hard for superpolynomial size circuits, there has been much exciting progress in exhibiting hard functions for *restricted* classes of circuits (we will see some examples in Section 6.1.1 below). Thus it is natural to ask whether hardness amplification can be used to construct average-case hard functions for these restricted classes of circuits. The answer to this question lies in *the complexity of the decoder circuit D in Definition 6.1.* To see why this is the case, let us imagine to start with a function f that is .25-hard for small circuits in a restricted class of circuits C (for simplicity we now ignore the size parameter). Aiming to show that Amp^f is $(1/2-\epsilon)$-hard for small circuits in C, and following the reasoning above, we suppose that there is a small circuit R in C that computes Amp^f on a $(1/2+\epsilon)$-fraction of the inputs. By definition of black-box hardness amplification, there is a small circuit D such that D^R computes f on a .75 fraction of the inputs. This contradicts our assumption that f is .25-hard for small circuits in C *as long as we can argue that D^R belongs to* C.

In this chapter we suggest that the most restricted circuit class C where the circuit D can be placed is that of constant-depth circuits with Majority gates (TC^0 circuits). In other words, we believe that to apply black-box hardness amplification we need to start from an explicit function that cannot be computed by polynomial-size TC^0 circuits. The existence of such a function is a major open problem in complexity theory, and it has been shown by Razborov and Rudich [RR] that a large class of proof techniques, which includes most of the known techniques, cannot be used to prove the existence of such a function.

In the next section we describe two well-studied circuit classes where black-box hardness amplification does not seem to be applicable. We highlight the reason for the inapplicability, and this will motivate the presentation of our results in Section 6.1.2.

6.1.1 Motivation

Constant-depth circuits with parity gates: One of the open problems that arguably best represents the current state of knowledge in computational complexity is that it is not known whether there is an explicit function $f:\{0,1\}^n \to \{0,1\}$ that is $\left(1/2 - 1/n^{\omega(1)}\right)$-hard for polynomial-size constant-depth circuits with Parity gates ($\mathsf{AC}^0[\oplus]$ circuits),[1] i.e. such that for any polynomial-size $\mathsf{AC}^0[\oplus]$ circuit C:

$$\Pr_{x\in\{0,1\}^n}[C(x) \neq f(x)] \geq 1/2 - 1/n^{\omega(1)}.$$

This problem is perhaps most puzzling because we do know of explicit functions that are $\Omega(1)$-hard for $\mathsf{AC}^0[\oplus]$ circuits of size $2^{n^{\Omega(1)}}$. An example is the Mod 3 function, i.e. counting the number of 1's mod 3 in a given n-bit input(see [Raz, Smo] and the survey by Beigel [Bei1, Corollary 22]). Moreover, throughout this thesis we discussed

[1] For context, this problem is also open if C varies over GF(2) polynomials of degree $2\log(n)$. This is a slightly different setting because these polynomials in general correspond to circuits of size $n^{\Omega(\log n)}$. On the other hand, if the degree is $\epsilon\log(n)$ then an explicit $\left(1/2 - 1/n^{\omega(1)}\right)$-hard function is known [BNS, HG].

several hardness amplification techniques that allow to construct a $\left(1/2 - 1/n^{\omega(1)}\right)$-hard function starting from a $\Omega(1)$-hard function. *The problem is that we do not know how to prove that any of these hardness amplification results holds for* $\mathsf{AC^0}[\oplus]$ *circuits.* In short, this is because the decoder circuit D in Definition 6.1 needs to compute Majority, and $\mathsf{AC^0}[\oplus]$ circuits cannot compute Majority. We now elaborate on this issue. Let us start by stating a celebrated result about the complexity of computing Majority by $\mathsf{AC^0}[\oplus]$ circuits.

Theorem 6.2 ([Raz, Smo])**.** *Let C be a $\mathsf{AC^0}[\oplus]$ circuit of depth d that computes Majority on l bits. Then the size of C is at least $s = 2^{l^{\Omega(1/d)}}$.*

We observe that with all known hardness amplifications, amplifying hardness up to $1/2 - \epsilon$ requires the decoder circuit D in Definition 6.1 to compute Majority on $1/\epsilon$ bits. Combining this observation with Theorem 6.2, we see that, to construct a $\left(1/2 - 1/n^{\omega(1)}\right)$-hard function for $\mathsf{AC^0}[\oplus]$ circuits, known hardness amplifications require to start with a function on $k \leq n$ bits that cannot be computed by $\mathsf{AC^0}[\oplus]$ circuits of size $s = 2^{n^{\omega(1)/d}} = 2^{n^{\omega(1)}}$, which of course does not exist. (Any function on n bits can be computed by a depth-2 circuit of size $\mathrm{poly}(n) \cdot 2^n$ via simple table look-up.)

Recall that in Chapter 4 (Theorem 4.3) we have constructed a function that is $\left(1/2 - 1/n^{\omega(1)}\right)$-hard for small constant-depth circuits with few arbitrary symmetric gates. Our result in particular gives a function that is $\left(1/2 - 1/n^{\omega(1)}\right)$-hard for $n^{\epsilon \cdot \log n}$-size circuits with $\epsilon \cdot \log n$ parity gates, but does not apply to circuits with an *unbounded* number of Parity gates, such as $\mathsf{AC^0}[\oplus]$. In fact, the hard function considered in Chapter 4 (Theorem 4.3) is *defined* as the function computed by a polynomial-size depth-3 $\mathsf{AC^0}[\oplus]$ circuit.

Constant-depth circuits with one Majority gate: Another puzzling case is that of constant-depth circuits with one majority gate ($\mathsf{Maj} \circ \mathsf{AC^0}$ circuits). It is known that the n-bits Parity function is $\Omega(1)$-hard for $\mathsf{Maj} \circ \mathsf{AC^0}$ circuits of size $2^{n^{\Omega(1)}}$ [ABFR] (see also [Kli, Theorem 6]). However, it is not known whether there is an explicit function $f : \{0,1\}^n \to \{0,1\}$ that is $\left(1/2 - 1/2^{n^{\Omega(1)}}\right)$-hard for $\mathsf{Maj} \circ \mathsf{AC^0}$ circuits of size $2^{n^{\Omega(1)}}$, or even $\left(1/2 - 1/n^{\omega(\log n)}\right)$-hard for $\mathsf{Maj} \circ \mathsf{AC^0}$ circuits of size $n^{\omega(\log n)}$. The strongest result in this direction is our result from Chapter 4 (Theorem 4.3) that in particular gives a function that is $\left(1/2 - 1/n^{\epsilon \cdot \log n}\right)$-hard for $\mathsf{Maj} \circ \mathsf{AC^0}$ circuits of size $n^{\epsilon \cdot \log n}$.

Even though these $\mathsf{Maj} \circ \mathsf{AC^0}$ circuits can trivially compute Majority, we still cannot use hardness amplification to construct an average-case hard functions for small $\mathsf{Maj} \circ \mathsf{AC^0}$ circuits. In short, this is because the decoder circuit in Definition 6.1 needs to make many queries to its oracle, and this increases the number of Majority gates up to an amount that we do not know how to handle (i.e., prove a lower bound for). Specifically, all known hardness amplifications, to amplify hardness up to $1/2 - \epsilon$, require the decoder circuit D in Definition 6.1 to make $1/\epsilon$ oracle queries. Thus, even when starting with a circuit R with only one Majority gate, the circuit D^R in Definition 6.1 will have at least $1/\epsilon$ Majority gates. Therefore, to construct a $\left(1/2 - 1/n^{\omega(1)}\right)$-hard function for small $\mathsf{Maj} \circ \mathsf{AC}^0$ circuits, known hardness amplifications require to start from a function on $k \leq n$ bits that cannot be computed by circuits of size $s = n^{\omega(1)}$ with s Majority gates, i.e., is not in TC^0.

6.1.2 Our results and organization

In this chapter we state two conjectures on the complexity of the decoder circuit D, and we prove some special cases of them. We now present our contributions in more detail.

Decoding requires Majority gates (Section 6.2): We conjecture that any circuit class C where the decoder circuit in Definition 6.1 can be implemented also contains a small circuit that can compute Majority on $\Omega(1/\epsilon)$ bits (as long as C is closed under AC^0 reductions).

While we are unable to prove our full-fledged conjecture, we present a proof of the conjecture in significant special cases where the decoder D achieves a certain strong setting of parameters (jumping ahead, this setting of parameters has the number of possible decoding circuits depend polynomially on $1/\epsilon$, as opposed to exponentially). This strong setting of parameters is actually achieved by some known constructions (e.g., [GL, STV]), and thus the presented result shows that the decoder circuits of these constructions cannot be implemented in circuit classes that cannot compute Majority (such as $\mathsf{AC}^0[\oplus]$).

Decoding requires many queries (Section 6.3): We conjecture that the decoder circuit in Definition 6.1 must make at least $q = \Omega(1/\epsilon)$ oracle queries.

We prove that the weaker bound $q = \Omega(\min\{1/\epsilon, k/\log k\})$ holds for decoder circuits that make *non-adaptive* oracle queries. We remark that in all known hardness amplifications the decoder circuit D does indeed make *non-adaptive* oracle queries. Thus, our result explains why known hardness amplification techniques cannot be applied to construct, say, a $(1/2 - 1/n)$-hard function for poly(n)-size constant-depth circuits with one Majority gate (see Section 6.1.1) or few arbitrary symmetric gates (see Chapter 4): to construct such a function using known black-box hardness amplification techniques we would need to start from an explicit function $f : \{0,1\}^k \to \{0,1\}$ that cannot be computed by poly(k)-size circuits with $\Omega(k/\log k)$ Majority gates. Again, no such function is known to exist.

6.2 Decoding requires Majority gates

Let us start by formally stating our conjecture that decoding requires Majority gates. We use the terminology 'class of circuits closed under AC^0 operations' to refer to any set C of oracle circuits that contains the class of oracle AC^0 circuits and is closed under the operation of replacing oracle gates with circuits from C. An example of a 'class of circuits closed under AC^0 operations' is the class of oracle $\mathsf{AC}^0[\oplus]$ circuits.

Conjecture 6.3. *Let C be a class of circuits closed under AC^0 operations (e.g., $\mathsf{C} = \mathsf{AC}^0[\oplus]$ circuits). Let Amp be a map from functions $f : \{0,1\}^k \to \{0,1\}$ to functions $Amp^f : \{0,1\}^n \to \{0,1\}$. Suppose that for every $f : \{0,1\}^k \to \{0,1\}$ and $R : \{0,1\}^n \to \{0,1\}$ such that*

$$\Pr_{x \in \{0,1\}^n}\left[R(x) \neq Amp^f(x)\right] < 1/2 - \epsilon$$

there is a circuit $D \in \mathsf{C}$ of size s and depth d such that

$$\Pr_{x \in \{0,1\}^k}\left[D^R(x) \neq f(x)\right] < 1/10.$$

Then, provided $s \leq 2^{k/2}$, there is a circuit $C \in \mathsf{C}$ of size $\mathrm{poly}(s/\epsilon)$ and depth $O(d)$ that computes Majority on $(1/\epsilon)^{\Omega(1)}$ bits.

In particular, if the circuit D in Conjecture 6.3 is an $\mathsf{AC}^0[\oplus]$ circuit of size s and depth d, then $s \geq 2^{1/\epsilon^{\Omega(1/d)}}$ by Theorem 6.2.

While we do not know how to prove Conjecture 6.3, Madhu Sudan (personal communication, 2005) pointed out a proof of the analogue of Conjecture 6.3 in the 'standard' list-decoding setting. We thank him for his permission to include a proof of this result here. The 'standard' list-decoding setting is where the length of the messages, the length of the codewords, the error parameter $1/\epsilon$, and the number of possible messages the decoder outputs are all polynomially related. This should be compared to the setting of Conjecture 6.3 where, for a given received word R, the number of possible messages is only bounded by the number of circuits D of size s, i.e. $2^{O(s \cdot \log s)}$, which is exponential in $1/\epsilon$ when $s = \mathrm{poly}(1/\epsilon)$. (See Section 3.3 for more on the connection between black-box hardness amplification and error-correcting codes.)

Theorem 6.4 (List-decoding requires Majority gates)**.** *Let $E : \{0,1\}^K \to \{0,1\}^N$ be a code. Suppose that there is a probabilistic circuit D of size s, for some $s \leq 2^K/4$, that on input a received word $r \in \{0,1\}^N$ outputs a list of (at most s) messages such that the following holds: for every message $m \in \{0,1\}^K$ whose encoding has relative hamming distance at most $1/2 - \epsilon$ from r, m is in the list $D(r)$ with high probability $(1 - o(1))$.*

Then there is an oracle AC^0-circuit C of size $\mathrm{poly}(s)$ and depth $O(1)$ such that C^D computes majority on $1/(2 \cdot \epsilon)$ bits.

In particular, if the circuit D in Theorem 6.4 is an $\mathsf{AC}^0[\oplus]$ circuit of size s and depth d, then $s \geq 2^{1/\epsilon^{\Omega(1/d)}}$ by Theorem 6.2. By contrast, if one does not restrict the decoder D to be an $\mathsf{AC}^0[\oplus]$ circuit, then there are codes that can be list-decoded by circuits of size $s = \text{poly}(N/\epsilon)$ (and thus in particular with lists of size $\text{poly}(N/\epsilon)$). For example, Guruswami and Sudan [GS] establish this for Reed-Solomon codes concatenated with Hadamard codes.

We now prove Theorem 6.4. We start with some intuition and then we present the formal proof.

Intuition for the proof of Theorem 6.4. The idea in the proof of Theorem 6.4 is the following. Assume that D is the decoder claimed in the theorem. First, it is well-known that it is impossible to decode from error rate $1/2$ (when the message length K is close to the block length N). Therefore, we can fix a particular message $\tilde{m}$ that cannot be decoded when its encoding $E(\tilde{m})$ is corrupted with error at rate $1/2$. However, by assumption D can decode from error rate $1/2-\epsilon$, and consequently can recover $\tilde{m}$ when $E(\tilde{m})$ is corrupted with error at rate $1/2-\epsilon$.

Our approach is to use D to solve the promise problem IsBal, which asks to distinguish bit-strings of length $1/\epsilon$ that are balanced (i.e. have one half of the bits set to 1) from those that have less than one half of the bits set to 1. We then show that solving IsBal is enough to compute Majority on $1/\epsilon$ bits, and this concludes the reduction.

To solve IsBal using D, we use an instance x of IsBal to generate an error vector N_x and then try to decode the corrupted string $E(\tilde{m}) \oplus N_x$ using D. We then check whether $\tilde{m}$ is in the output of the decoder, and decide accordingly. The error vector N_x is obtained by independently setting each bit of N_x equal to a random bit of x. It is evident that if x is balanced then N_x will correspond to error at rate $1/2$. By what we said above, in this case the decoder C will fail to recover $\tilde{m}$. Conversely, if x has less than $|x|/2 = \epsilon^{-1}/2$ bits set to 1, then the error vector N_x will correspond to error at rate $(\epsilon^{-1}/2-1)/\epsilon^{-1} = 1/2-\epsilon$ and, again by what we said above, the decoder D will be able to recover $\tilde{m}$.

We now present a formal proof. Let us denote by $C(x;u)$ a probabilistic circuit on input x and coin tosses u, and let us define the problem IsBal.

Definition 6.5. *The promise problem* IsBal *is defined on inputs of even length as follows:*

- $\text{IsBal}_{\text{YES}} := \{x : \text{weight}(x) = |x|/2\}$,
- $\text{IsBal}_{\text{NO}} := \{x : \text{weight}(x) < |x|/2\}$.

We say that a probabilistic circuit $C(x;u)$ solves IsBal *if for every x:*

- $x \in \text{IsBal}_{\text{YES}} \Rightarrow \Pr_u[C(x;u) = 1] \geq 2/3$,

- $x \in \text{IsBal}_{\text{NO}} \Rightarrow \Pr_u[C(x;u) = 1] \leq 1/2$.

Lemma 6.6. *Let $C(x;u)$ be a probabilistic circuit that solves* IsBal *on instances of length l, for even l. Then there is a deterministic oracle circuit A of size* $\text{poly}(l)$ *and depth $O(1)$ such that A^C computes Majority on l bits.*

Proof. First, let us construct a deterministic circuit C' to solve IsBal. On input x, we run $\text{poly}(l)$ copies of C with independent random bits. By a Chernoff bound (see Appendix A), we can fix the random bits of these $\text{poly}(l)$ copies in such a way that the following holds: if $x \in \text{IsBal}_{\text{YES}}$ then at least a .55 fraction of the copies will output 1, while if $x \in \text{IsBal}_{\text{NO}}$ then at least a .55 fraction of the copies will output 0. We are thus left with the task of distinguishing these two cases, namely distinguishing strings of length $\text{poly}(l)$ whose fraction of 1's is at least .55 from those whose fraction of 1's is at most .45. This can be done by a $\text{poly}(l)$-size circuit of depth 3 by a result by Ajtai [Ajt1], because the probabilities .55 and .45 are bounded away from 1/2 by a constant. (The proof of a stronger result is given in Theorem 5.2, Chapter 5.) Thus we have deterministic circuit $C' = A^C$ that decides IsBal, where A has size $\text{poly}(l)$ and depth $O(1)$.

Then we decide Majority on l bits as follows. Given an input $y \in \{0,1\}^l$ we compute inputs $y_0, \ldots, y_l$ where y_i is obtained from y by setting the first i bits to 0 (so $y_0 = y$ and $y_l = 0^l$). We then run a copy of C' in parallel on each of the y_i's and we answer 'yes' if and only if we ever get two different values.

If y has less than $l/2$ 1's then all the circuits will output 0. Conversely, if y has at least $l/2$ 1's then $C'(y_l) = 0$ but $C'(y_i) = 1$ where y_i has exactly $l/2$ 1's.

It is easy to check that this new circuit that computes Majority is of the form A^C where A has size $\text{poly}(l)$ and depth $O(1)$. □

We now prove Theorem 6.4.

Proof of Theorem 6.4. We start with the following claim that quantifies our intuition that one cannot list-decode from error at rate 1/2. Let $N_{1/2}$ denote a random string chosen uniformly in $\{0,1\}^N$, and recall that D is a probabilistic circuit of size $s \leq 2^K/4$ that on input a received word $r \in \{0,1\}^N$ outputs a list of (at most s) messages.

Claim 6.7. *There is a message $\tilde{m} \in \{0,1\}^K$ such that*

$$\Pr_{N_{1/2}}\left[\tilde{m} \in D(E(\tilde{m}) + N_{1/2})\right] \leq 1/4.$$

Proof. We have

$$\begin{aligned}
\Pr_{M \in \{0,1\}^K, N_{1/2}}\left[\tilde{m} \in D(E(M) + N_{1/2})\right] &= \Pr_{M \in \{0,1\}^K, N_{1/2}}\left[M \in D(N_{1/2})\right] \\
&\leq s/2^K \\
&\leq 1/4,
\end{aligned}$$

where the second to last inequality is a union bound, and the last inequality is our assumption that $s \leq 2^K/4$. The claim follows by fixing $M = \tilde{m}$. □

We can assume that $\epsilon \geq 1/N$. This is because if $\epsilon < 1/N$ then the decoder D is decoding when half the bits of the encoded message are errors, which can be shown to be impossible by an argument similar to the above Claim 6.7 (details omitted).

We further assume $\epsilon \geq 1000/\sqrt{N}$. This clearly does not change the conclusion of the theorem, but lets the analysis below go through.

Now we construct a new circuit C that has $\tilde{m}$ and $E(\tilde{m})$ hardwired in it and operates as follows. Given an instance x of IsBal of length $1/(2 \cdot \epsilon)$ (which we assume to be an even integer w.l.o.g.) it constructs the error vector N_x, computes $E(\tilde{m}) \oplus N_x$, and answers 0 if and only if $\tilde{m} \in C(E(\tilde{m}) + N_x)$. As explained in the intuition paragraph, the error vector N_x is obtained by independently setting each bit of N_x equal to a random bit of x.

Analysis: If $x \in \text{IsBal}_{\text{YES}}$ then by Claim 6.7 the circuit outputs 0 with probability at least 1/2. If $x \in \text{IsBal}_{\text{NO}}$: suppose the relative Hamming weight of x is at most $(1/2 \cdot 1/(2\epsilon) - 1)/(1/2\epsilon) = 1/2 - 2\epsilon$. By a Chernoff bound (see Appendix A) the number of 1's in N_x is at most $1/2 - 2\epsilon + \epsilon = 1/2 - \epsilon$ with probability very close to 1, at least $2/3 + \Omega(1)$ (recall $\epsilon \geq 1000/\sqrt{N}$). Whenever this is the case, by assumption, $\tilde{m}$ appears in the list returned by the decoder D with high probability $1 - o(1)$.

The new oracle circuit C is of the form $C = A^D$ where A has size poly(s) and depth $O(1)$, and is solving IsBal on instances of length $1/(2 \cdot \epsilon)$. The result follows from Lemma 6.6. □

6.2.1 List-decoding the Hadamard code requires Majority gates

The Hadamard code, $\text{Had} : \{0,1\}^k \to \{0,1\}^{2^k}$, is defined as $\text{Had}(x)_i := \langle x, i \rangle$ where $\langle \cdot, \cdot \rangle$ denotes inner product modulo 2. The Hadamard code is omnipresent in the hardness amplification literature (see, e.g., [GL, GNW, STV] and the survey by Trevisan [Tre3]). In particular, Yao's Xor Lemma (see Section 2.1) can be proven by first using a Direct Product Lemma to construct a *non-boolean* function that is very hard on average and then using the Hadamard code to obtain a boolean function (see [GNW]). It can be verified that the proof of correctness of the first step, i.e. the Direct Product Lemma, carries through for constant-depth circuits (without Majority gates). However, the proof of correctness of the second step does *not* carry through for constant-depth circuits (without Majority gates). The proof of correctness of this second step relies on a list-decoding algorithm for the Hadamard code, implicit in [GL]. We obtain the following theorem that shows that the parameters achieved by known decoding procedures of the Hadamard code cannot be achieved by constant-depth decoders without Majority gates.

Theorem 6.8. *Let* Had : $\{0,1\}^k \to \{0,1\}^{2^k}$ *be the Hadamard code, and let* C *be a class of circuits closed under* AC^0 *operations (e.g.,* $\mathsf{C} = \mathsf{AC}^0[\oplus]$ *circuits). Suppose that there is an oracle probabilistic circuit* $D \in \mathsf{C}$ *of size* s *and depth* d *such that for every oracle* $R \in \{0,1\}^{2^k}$, D^R *outputs a list of size* s *that, with high probability, contains all the strings* $m \in \{0,1\}^k$ *such that the relative hamming distance between* R *and* $\mathrm{Had}(m)$ *is at most* $1/2 - \epsilon$.

Then there is a circuit $C \in \mathsf{C}$ *of size* $\mathrm{poly}(s)$ *and depth* $O(d)$ *that computes Majority on* $1/(2 \cdot \epsilon)$ *bits.*

In particular, if the circuit D in Theorem 6.8 is an $\mathsf{AC}^0[\oplus]$ circuit of size s and depth d, then $s \geq 2^{1/\epsilon^{\Omega(1/d)}}$ by Theorem 6.2. By contrast, if one does not restrict the decoder to be an $\mathsf{AC}^0[\oplus]$ circuit, Goldreich and Levin [GL] show that $s = \mathrm{poly}(k/\epsilon)$ is achievable (see also [Tre3, Theorem 12]).

Proof of Theorem 6.8. Observe that if we have a message of the form $m = z \cdot 0^{k-k'}$, where $z \in \{0,1\}^{k'}$, then the value of the i-th bit of the codeword $\mathrm{Had}(m)$ only depends on the first k' bits of i. This follows easily from the definition of the Hadamard code.

The above observation suggests the following approach. Let $k' := 2 \cdot \log s$ and consider the smaller Hadamard code $\mathrm{Had}' : \{0,1\}^{k'} \to \{0,1\}^{2^{k'}}$. From the decoder $D \in \mathsf{C}$ one can easily derive a decoder D' for Had' of size $\mathrm{poly}(s)$ and depth $O(d)$ that does not use any oracle but rather is given as input the whole received word in $\{0,1\}^{2^{k'}}$. Specifically, to decode the received word $r' \in \{0,1\}^{2^{k'}}$, the decoder D' simply simulates D answering any oracle query of D to location $i \in \{0,1\}^k$ with the bit of r' indexed by the first k' bits of i. The correctness of this decoding procedure follows from our assumption on D and the observation at the beginning of this proof.

The result now follows from Theorem 6.4. □

6.2.2 List-decoding the Reed-Muller code requires Majority gates

We now describe the Reed-Muller code. For given parameters k and ϵ, which for simplicity of exposition we assume to be such that $\epsilon \geq 2^{-k}$, let $\mathcal{F}$ be a field of size $(k/\epsilon)^c$, for a certain universal constant $c \geq 1$ the exact magnitude of which is not relevant here. Now let $\mathcal{H} \subseteq \mathcal{F}$ be a set of size $\sqrt{|\mathcal{F}|}$. We map any given function $f : \{0,1\}^k \to \{0,1\}$ into another function $f' : \{0,1\}^{O(k)} \to \{0,1\}$ as follows. Let $v := k/\log(|\mathcal{H}|)$, which again for simplicity of exposition we assume to be a positive integer, and view f as a function $f : \mathcal{H}^v \to \{0,1\} \subseteq \mathcal{F}$. We define $f' : \mathcal{F}^v \to \mathcal{F}$ to be the unique polynomial over $\mathcal{F}$ in v variables, of degree $|\mathcal{H}|$ in each variable, that agrees with f on $\mathcal{H}^v$.

To obtain a binary code, we concatenate this construction with the Hadamard code, defined in Section 6.2.1, and obtain the following:

$$\mathcal{RM}^f(x_1, \ldots, x_v, t) := \langle f'(x_1, \ldots, x_v), t \rangle,$$

where f' is the extension of f defined above, $x_i \in \mathcal{F}$, and $|t| = \log|F|$. Note that our initial assumption that $\epsilon \geq 2^{-k}$ implies that the input length of $\mathcal{RM}^f$ is $O(k)$.

We have the following theorem.

Theorem 6.9. *Let $\mathcal{RM}$ be the Reed-Muller code defined above, mapping functions $f : \{0,1\}^k \to \{0,1\}$ to functions $\mathcal{RM}^f : \{0,1\}^{O(k)} \to \{0,1\}$, and let C be a class of circuits closed under AC^0 operations (e.g., $\mathsf{C} = \mathsf{AC}^0[\oplus]$ circuits).*

Suppose that there is an oracle probabilistic circuit $D(x; u; \alpha) \in \mathsf{C}$ of size s, where $s \leq 2^{k/\epsilon}/4$, and depth d, which uses coin tosses u and advice $\alpha \in \{1, \dots, s\}$, such that the following holds: for all $f : \{0,1\}^k \to \{0,1\}$ and $R : \{0,1\}^{O(k)} \to \{0,1\}$, if

$$\Pr_{x_1,\dots,x_v,t}\left[R(x_1,\dots,x_v,t) \neq \mathcal{RM}^f(x_1,\dots,x_v,t)\right] < 1/2 - \epsilon$$

then with high probability ($1 - o(1)$) over the randomness u of D there is an advice string $\alpha \in \{1,\dots,s\}$ such that for every $x \in \{0,1\}^k$:

$$D^R(x; u; \alpha) = f(x).$$

Then there is a circuit $C \in \mathsf{C}$ of size $\mathrm{poly}(s)$ and depth $O(d)$ that computes Majority on $1/(2\cdot\epsilon)$ bits.

In particular, if the circuit D in Theorem 6.9 is an $\mathsf{AC}^0[\oplus]$ circuit of size s and depth d, then $s \geq 2^{1/\epsilon^{\Omega(1/d)}}$ (by Theorem 6.2). By contrast, Sudan, Trevisan, and Vadhan [STV] show that if we do not restrict the circuit D to be an $\mathsf{AC}^0[\oplus]$ circuit then $s = \mathrm{poly}(k/\epsilon)$ is achievable.

Proof of Theorem 6.9. Observe that if we start with a message f whose value on $x_1,\dots,x_v$ only depends on x_1, then the same is true for $\mathcal{RM}^f$, i.e. the value of $\mathcal{RM}^f$ on $x_1,\dots,x_v,t$ only depends on x_1 and t. (To see this, note that clearly there is a univariate polynomial f' in the variable x_1 of degree at most $|\mathcal{H}|$ that agrees with f on $\mathcal{H}^v$, and we know that such a polynomial is unique.)

The above observation suggests the following approach. Consider the new code E with message length $|\mathcal{H}| = \mathrm{poly}(k/\epsilon)$ and block length $|\mathcal{F}|^2 = \mathrm{poly}(k/\epsilon)$ defined as follows: given a message m, view m as the truth table of a function $f : \mathcal{H}^v \to \{0,1\}$ that only depends on the first variable $x_1 \in \mathcal{H}$. The (x,t) position of the codeword $E(m)$ (where $x \in \mathcal{F}$, $|t| = \log|\mathcal{F}|$) is defined as $\mathcal{RM}^f(x, 0, \dots, 0, t)$.

Now note that the assumption of the theorem implies that E can be list-decoded by a circuit $D' \in \mathsf{C}$ of size $\mathrm{poly}(s \cdot k/\epsilon)$ and depth $O(d)$. This can be seen as follows. Consider the probabilistic decoder D' that given a received word r (a corrupted codeword of E) of length $\mathrm{poly}(k/\epsilon)$ operates as follows. First, it tosses coins u for D. Second, it runs in parallel $D(x; u; \alpha)$ for every input $x \in \mathcal{H}$ and every possible advice $\alpha \in \{1,\dots,s\}$, answering the oracle queries of D as follows: whenever a query $(x_1, x_2, \dots, x_v, t)$ is made, the query is replied by ignoring $x_2,\dots,x_v$, and by answering with the position in the received word r indexed by (x_1, t).

The correctness of this decoder D' follows from our assumption on D and the observation at the beginning of this proof. Thus, we have a decoder $D' \in \mathsf{C}$ of size $\mathrm{poly}(s \cdot k/\epsilon)$ and depth $O(d)$ that list-decodes a code with message length and block length $\mathrm{poly}(k/\epsilon)$ from error $1/2 - \epsilon$. Recalling that $s \leq 2^{k/\epsilon}/4$ in the hypothesis of the theorem, the result follows by applying Theorem 6.4. $\square$

6.3 Decoding requires many queries

In this section we prove a lower bound on the number of oracle queries of *non-adaptive* decoding procedures. We state our result in terms of decoding procedures that take advice strings $\alpha \in \{0,1\}^s$ because this is more natural in a coding-theoretic setting. This can be immediately translated to decoding procedures implemented by a circuit of size s (as in Definition 6.1) by using the fact that a circuit of size s can be described by a string of length $O(s \cdot \log s)$.

Theorem 6.10 (Decoding requires many queries)**.** *Let Amp be a map from functions $f : \{0,1\}^k \to \{0,1\}$ to functions $Amp^f : \{0,1\}^n \to \{0,1\}$. Let $D(.,.)$ be an oracle decoding procedure satisfying the following: for every $f : \{0,1\}^k \to \{0,1\}$ and $R : \{0,1\}^n \to \{0,1\}$ such that*

$$\Pr_{x \in \{0,1\}^n} \left[R(x) \neq Amp^f(x) \right] < 1/2 - \epsilon,$$

there is an advice string $\alpha \in \{0,1\}^s$, $s \leq 2^{k/2}$, such that

$$\Pr_{x \in \{0,1\}^k} \left[D^R(\alpha, x) \neq f(x) \right] < .25.$$

If the oracle queries made by D are non-adaptive then D makes at least $q \geq \min\{1/(10 \cdot \epsilon), k/(5 \cdot \log k)\}$ (for sufficiently large k, n and $1/\epsilon$).

Intuition for the proof of Theorem 6.10. Let us now explain the main ideas behind the proof of Theorem 6.10. Let us choose $F : \{0,1\}^k \to \{0,1\}$ to be a random function, and $N : \{0,1\}^n \to \{0,1\}$ to be a random function such that $N(x)$ equals 1 with probability $1/2 - \epsilon$, independently for every x. Let us write $Amp^F \oplus N$ for the function that maps x to $Amp^F(x) \oplus N(x)$. By our assumption, the decoder D can always decode $Amp^F \oplus N$ for some value of the advice. Therefore, by a union bound, we can fix a particular advice string $\alpha \in \{0,1\}^s$ and hence a particular oracle decoder $C(\cdot) := D(\alpha, \cdot)$ such that

$$\Pr_{F,N} \left[\Pr_{x \in \{0,1\}^k} \left[C^{Amp^F \oplus N}(x) = f(x) \right] \geq .75 \right] \geq 2^{-s}. \tag{6.1}$$

We would like to show that Equation (6.1) cannot hold.

Let us fix a particular input x and a particular query $y \in \{0,1\}^n$ that $C(x)$ makes. Since $N(y) = 1$ with probability $1/2-\epsilon$, the statistical difference between the answer $Amp^F(y) \oplus N(y)$ to this query and an unbiased random bit is only ϵ. Since by assumption $C(x)$ makes only q queries, the statistical difference between all the q query answers of $C(x)$ and q unbiased and independent random bits will be at most $q \cdot \epsilon$ which by assumption is very small, say .1. This suggests to consider a new decoder C' without any oracle that simulates C and answers each oracle query C makes with an unbiased random bit. By the above argument, for every x, $C'(x) = C^{Amp^F \oplus N}(x)$ with probability at least .9. Therefore, we expect C' and $C^{Amp^F \oplus N}$ to agree on a .9 fraction of the inputs. If we could prove that C' and $C^{Amp^F \oplus N}$ agree on a .9 fraction of x's *almost always*, say with probability $1 - 2^{-s}/2$, then we could combine this fact with the above Equation (6.1) and derive a contradiction with a counting argument. Specifically, we would have that C' recovers a .9 fraction of the bits of the message with probability at least $2^{-s} - 2^{-s}/2 = 2^{-s}/2$, which contradicts a counting argument.

However, there remains the problem to show that C' and $C^{Amp^F \oplus N}$ agree on a .9 fraction of x's almost always. We observe that if the queries made by C on different inputs are *disjoint*, then the events that C' and $C^{Amp^F \oplus N}$ agree on a particular x are independent for different x's, and therefore we can apply a Chernoff bound to argue that in fact C' and $C^{Amp^F \oplus N}$ agree on a large fraction of x's almost always. Of course, the queries that C makes need not be disjoint. The idea is then to reduce to the case where the queries are indeed disjoint using an inductive argument based on *coverings* that is very similar to the one used in the proof of Theorem 5.11.

Proof of Theorem 6.10. Assume for the sake of contradiction that $q \le \min\{.1/\epsilon, k/(5 \cdot \log k)\}$. It is also convenient to assume that q is sufficiently large, which we can do without loss of generality because n, k and $1/\epsilon$ are sufficiently large.

Let us choose $F : \{0,1\}^k \to \{0,1\}$ to be a random function, and $N : \{0,1\}^n \to \{0,1\}$ to be a random function such that $N(x)$ equals 1 with probability $1/2 - \epsilon$, independently for every x. Let us write $Amp^F \oplus N$ for the function that maps x to $Amp^F(x) \oplus N(x)$.

By the standard Central Limit Theorem we have that $N(x) = 1$ for at most $1/2-\epsilon$ fraction of inputs x with probability at least, say, .3. Combining this fact with our assumption we have:

$$\Pr_{F,N}\left[\exists \alpha \in \{0,1\}^s : \Pr_{x \in \{0,1\}^k}\left[D^{Amp^F \oplus N}(\alpha, x) = f(x)\right] \ge .75\right] \ge .3.$$

By a union bound we can fix a particular advice string $\alpha \in \{0,1\}^s$ and hence a particular oracle decoder $C(\cdot) := D(\alpha, \cdot)$ such that

$$\Pr_{F,N}\left[\Pr_{x \in \{0,1\}^k}\left[C^{Amp^F \oplus N}(x) = f(x)\right] \ge .75\right] \ge .3 \cdot 2^{-s}. \tag{6.2}$$

In the rest of the proof we show that Equation (6.2) cannot hold. For $x \in \{0,1\}^k$, and a fixed decoder C (which will always be clear from the context) let us denote by $Q(x)$ the set of queries that $C(x)$ makes. Note $Q(x)$ is well-defined because we are dealing with decoders that make non-adaptive oracle queries. A *covering* of sets of queries $Q_1, Q_2, \ldots, Q_{2^k}$ is a subset $\Gamma \subseteq \{0,1\}^n$ that intersects each of the Q_i's. Note that, if we have sets of queries $Q_1, Q_2, \ldots, Q_{2^k}$ of size q each, either there exist d disjoint sets of queries among them or else there must be a covering of $Q_1, Q_2, \ldots, Q_{2^k}$ of size at most $d \cdot q$. (To see why this is the case, see the analogous reasoning in the proof of Theorem 5.11.) Let us define the sequence

$$g_i := s \cdot q^{2 \cdot i}.$$

Consider the following procedure, starting with $i := 1$ and where C_1 is the decoder circuit C in Equation (6.2).

Procedure(C_i)
If there exist g_i/q distinct $x_1, x_2, \ldots, x_{g_i/q} \in \{0,1\}^k$ such that $Q(x_1), Q(x_2), \ldots, Q(x_{g_i/q})$ are disjoint then *stop*. Otherwise, let Γ be a covering of $Q(1), Q(2), \ldots, Q(2^k)$ of size at most g_i. (Such a covering exists by the reasoning presented above.) Let C_{i+1} be the decoder that simulates C_i but answers queries in Γ with unbiased and independent random bits. Let $i := i + 1$ and iterate the procedure.

Claim 6.11. *The procedure stops.*

Proof. In the above procedure, note that decoder C_i makes at most $q - i + 1$ queries, for every i. Thus, after $q + 1$ steps we have a decoder that makes 0 queries on every input x, and therefore the query sets $Q(1), Q(2), \ldots, Q(2^k) = \emptyset$ are trivially disjoint. We only need to argue that it is possible to find sufficiently many *distinct* inputs x's $\in \{0,1\}^k$. In other words, we need that $g_{q+1} = s \cdot q^{2(q+1)} \leq 2^k$. This holds because $s \leq 2^{k/2}$ and $q \leq k/(5 \cdot \log k)$. $\square$

Note that when going from C_i to C_{i+1} the success probability multiplies by 2^{-g_i}, because C_{i+1} is answering with random bits the queries in a set of size g_i. More formally, denoting by Coins(C) the random bits used by decoder C, we have:

$$\Pr_{F,N,\mathrm{Coins}(C_{i+1})}\left[\Pr_{x\in\{0,1\}^k}\left[C_{i+1}^{Amp^F \oplus N}(x) = f(x)\right] \geq .75\right] \geq$$
$$2^{-g_i} \cdot \Pr_{F,N,\mathrm{Coins}(C_i)}\left[\Pr_{x\in\{0,1\}^k}\left[C_{i}^{Amp^F \oplus N}(x) = f(x)\right] \geq .75\right].$$

Combining this with Equation 6.2, and letting t be the number of times the procedure is iterated, we obtain (note $g_0 = s$):

$$\Pr_{F,N,\mathrm{Coins}(C_t)}\left[\Pr_{x\in\{0,1\}^k}\left[C_{t}^{Amp^F \oplus N}(x) = f(x)\right] \geq .75\right] \geq .3 \cdot 2^{-\sum_{0\leq i\leq t-1} g_i}. \tag{6.3}$$

Now let X be the set of at least g_t/q distinct $x_1, x_2, \ldots, x_{g_t/q} \in \{0,1\}^k$ such that

$$Q(x_1), Q(x_2), \ldots, Q(x_{g_t/q})$$

are disjoint, as guaranteed by the fact that the procedure stops after t iterations. Consider the decoder C' that simulates C_t but answers the queries with random bits. Note that C' does not use any oracle. Also note that for every fixed f and over the choice of N, the statistical difference (see Definition 2.5) between a query answer from $Amp^{f \oplus N}$ and a truly random bit is at most ϵ. Consequently, the statistical difference between q query answers from $Amp^{f \oplus N}$ and q truly random bits is at most $q \cdot \epsilon \leq .1$ (see, e.g., Fact 2.3 in [SV1]). Therefore, for every fixed $x \in X$, we have that

$$\Pr_{F,N,\mathrm{Coins}(C_t)}\left[C_t^{Amp^F \oplus N}(x) = C'(x)\right] \geq .9.$$

Since the query sets associated to elements of X are disjoint, we can apply a Chernoff Bound (see Appendix A) to argue that the probability (over N) that C' equals C_t for at least a .8 fraction of the elements of X is at least $1 - 2^{-\Omega(|X|)}$. Combining this fact with Equation 6.3 we obtain that

$$\Pr_{F,N,\mathrm{Coins}(C')}\left[\Pr_{x \in X}\left[C'(x) = f(x)\right] \geq .55\right] \geq .3 \cdot 2^{-\sum_{0 \leq i \leq t-1} g_i} - 2^{-\Omega(|X|)} \geq 2^{-2 \cdot |X|/q}, \quad (6.4)$$

where the last inequality is true because we are assuming (see the beginning of the proof) that q is sufficiently large and because

$$\begin{aligned}
\sum_{0 \leq i \leq t-1} g_i &= s \sum_{0 \leq i \leq t-1} q^{2i} \\
&\leq s \sum_{0 \leq i \leq 2t-2} q^i \\
&= s \cdot (q^{2t-1} - 1)/(q-1) \\
&\leq (2/q) \cdot s \cdot q^{2t}/q \\
&= (2/q) \cdot g_t/q = (2/q) \cdot |X|.
\end{aligned}$$

When q is sufficiently large, Equation 6.4 contradicts the fact that there are only $2^{H(.45)|X|} = 2^{(1-\Omega(1))|X|}$ strings of length $|X|$ that are at relative Hamming distance at most .45 from a fixed string of length $|X|$. Specifically, by an averaging argument we can fix the choice of the coin tosses of C' while having the probability bound in Equation (6.4) still hold. Now we can consider the fixed string z obtained by concatenating the output of C' on every $x \in X$, that is, $z := \bigcirc_{x \in X} C'(x)$. By Equation (6.4) (after the fixing of the coins of C'), the random string $\bigcirc_{x \in X} F(x) \in \{0,1\}^{|X|}$ (for random F) is at relative Hamming distance at most .45 from z with probability at least $2^{-2 \cdot |X|/q}$, which contradicts the fact that there are only $2^{H(.45)|X|} = 2^{(1-\Omega(1))|X|}$ strings of length $|X|$ that are at relative Hamming distance at most .45 from the fixed string z of length $|X|$. $\square$

Bibliography

[AKS] M. Agrawal, N. Kayal, and N. Saxena. PRIMES in P. *Ann. of Math.*, 160(2):781–793, 2004.

[Ajt1] M. Ajtai. Σ_1^1-formulae on finite structures. *Ann. Pure Appl. Logic*, 24(1):1–48, 1983.

[Ajt2] M. Ajtai. Approximate counting with uniform constant-depth circuits. In *Advances in computational complexity theory (New Brunswick, NJ, 1990)*, pages 1–20. Amer. Math. Soc., Providence, RI, 1993.

[ABO] M. Ajtai and M. Ben-Or. A Theorem on Probabilistic Constant Depth Computation. In ACM, editor, *Proceedings of the sixteenth annual ACM Symposium on Theory of Computing, Washington, DC, April 30–May 2, 1984*, pages 471–474, 1984. ACM order no. 508840.

[All] E. Allender. A Note on the Power of Threshold Circuits. In *30th Annual Symposium on Foundations of Computer Science*, pages 580–584, Research Triangle Park, North Carolina, 30 Oct.–1 Nov. 1989. IEEE.

[AKR$^+$] E. Allender, M. Koucký, D. Ronneburger, S. Roy, and V. Vinay. Time-Space Tradeoffs in the Counting Hierarchy. In *Proceedings of the Sixteenth Annual Conference on Computational Complexity*, pages 295–302. IEEE, June 18–21 2001.

[AGHP] N. Alon, O. Goldreich, J. Håstad, and R. Peralta. Simple constructions of almost k-wise independent random variables. *Random Structures & Algorithms*, 3(3):289–304, 1992.

[And] I. Anderson. *Combinatorics of finite sets*. Dover Publications Inc., Mineola, NY, 2002. Corrected reprint of the 1989 edition.

[ABFR] J. Aspnes, R. Beigel, M. Furst, and S. Rudich. The expressive power of voting polynomials. *Combinatorica*, 14(2):135–148, 1994.

[BFL] L. Babai, L. Fortnow, and C. Lund. Nondeterministic exponential time has two-prover interactive protocols. *Comput. Complexity*, 1(1):3–40, 1991.

[BFNW] L. Babai, L. Fortnow, N. Nisan, and A. Wigderson. BPP Has Subexponential Time Simulations Unless EXPTIME has Publishable Proofs. *Computational Complexity*, 3(4):307–318, 1993.

[BNS] L. Babai, N. Nisan, and M. Szegedy. Multiparty protocols, pseudorandom generators for logspace, and time-space trade-offs. *J. Comput. System Sci.*, 45(2):204–232, 1992. Twenty-first Symposium on the Theory of Computing (Seattle, WA, 1989).

[BGS] T. Baker, J. Gill, and R. Solovay. Relativizations of the *P=?NP* question. *SIAM J. Comput.*, 4(4):431–442, 1975.

[BDG] J. L. Balcazar, J. Diaz, and J. Gabarro. *Structural complexity Theory.* Springer-Verlag New York, Inc., New York, NY, USA, 1988.

[Bea] P. Beame. A switching lemma primer. Technical Report UW-CSE-95-07-01, Department of Computer Science and Engineering, University of Washington, November 1994. Available from `http://www.cs.washington.edu/homes/beame/`.

[BSSV] P. Beame, M. Saks, X. Sun, and E. Vee. Time-space trade-off lower bounds for randomized computation of decision problems. *J. ACM*, 50(2):154–195 (electronic), 2003.

[BF] D. Beaver and J. Feigenbaum. Hiding Instances in Multioracle Queries. In *7th Annual Symposium on Theoretical Aspects of Computer Science*, volume 415 of *Lecture Notes in Computer Science*, pages 37–48, Rouen, France, 22–24 Feb. 1990. Springer.

[Bei1] R. Beigel. The polynomial method in circuit complexity. In *Proceedings of the Eighth Annual Structure in Complexity Theory Conference (San Diego, CA, 1993)*, pages 82–95, Los Alamitos, CA, 1993. IEEE Comput. Soc. Press.

[Bei2] R. Beigel. When do extra majority gates help? 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.

[BRS] R. Beigel, N. Reingold, and D. A. Spielman. The Perceptron Strikes Back. In *Structure in Complexity Theory Conference*, pages 286–291, 1991.

[BT] R. Beigel and J. Tarui. On ACC. *Comput. Complexity*, 4(4):350–3–66, 1994. Special issue devoted to the 4th Annual McGill Workshop on Complexity Theory. Preliminary version in FOCS '91.

[BL] M. Ben-Or and N. Linial. Collective Coin-Flipping. In S. Micali, editor, *Randomness and Computation*, pages 91–115. Academic Press, New York, 1990.

[BM] M. Blum and S. Micali. How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits. *SIAM J. on Computing*, 13(4):850–864, Nov. 1984.

[BT] A. Bogdanov and L. Trevisan. On Worst-Case to Average-Case Reductions for NP Problems. In *44th Annual Symposium on Foundations of Computer Science*, pages 308–317, Cambridge, Massachusetts, 11–14 Oct. 2003. IEEE.

[Bop] R. B. Boppana. The average sensitivity of bounded-depth circuits. *Inform. Process. Lett.*, 63(5):257–261, 1997.

[CPS] J.-Y. Cai, A. Pavan, and D. Sivakumar. On the Hardness of the Permanent. In *16th International Symposium on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science, Volume 1563, pages 90–99, Trier, Germany, 1999. Springer-Verlag.

[CSS] J.-Y. Cai, D. Sivakumar, and M. Strauss. Constant Depth Circuits and the Lutz Hypothesis. In *38th Annual Symposium on Foundations of Computer Science*, pages 595–604, Miami Beach, Florida, 20–22 Oct. 1997. IEEE.

[Can] R. Canetti. More on BPP and the polynomial-time hierarchy. *Inform. Process. Lett.*, 57(5):237–241, 1996.

[CH] A. Chattopadhyay and K. A. Hansen. Lower Bounds for Circuits With Few Modular and Symmetric Gates. In L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, editors, *Proceedings of the 32th International Colloquium on Automata, Languages and Programming (ICALP)*, Lecture Notes in Computer Science, Lisboa, Portugal, 11–5 July 2005. Springer-Verlag.

[CR] S. Chaudhuri and J. Radhakrishnan. Deterministic Restrictions in Circuit Complexity. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, pages 30–36, Philadelphia, Pennsylvania, 22–24 May 1996.

[Che] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Ann. Math. Statistics*, 23:493–507, 1952.

[CLRS] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms.* MIT Press, Cambridge, MA, second edition, 2001.

[DvM] S. Diehl and D. van Melkebeek. Time-Space Lower Bounds for the Polynomial-Time Hierarchy on Randomized Machines. In L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, editors, *Proceedings of the 32th International Colloquium on Automata, Languages and Programming (ICALP)*, Lecture Notes in Computer Science, pages 982–993, Lisboa, Portugal, 11–5 July 2005. Springer-Verlag.

[ESY] S. Even, A. L. Selman, and Y. Yacobi. The complexity of promise problems with applications to public-key cryptography. *Inform. and Control*, 61(2):159–173, 1984.

[FL] U. Feige and C. Lund. On the Hardness of Computing the Permanent of Random Matrices. *Computational Complexity*, 6(2):101–132, 1996.

[FF] J. Feigenbaum and L. Fortnow. Random-Self-Reducibility of Complete Sets. *SIAM J. on Computing*, 22(5):994–1005, Oct. 1993.

[For] L. Fortnow. Balanced NP sets. *Computational Complexity Weblog*, 2003. `http://weblog.fortnow.com/archive/2003_09_07_archive.html`.

[FLvMV] L. Fortnow, R. Lipton, D. van Melkebeek, and A. Viglas. Time-Space Lower Bounds for Satisfiability. To appear in Journal of the ACM, Available from *http://www.cs.wisc.edu/∽dieter/Research/sat-ts.html*, 2005.

[FSS] M. L. Furst, J. B. Saxe, and M. Sipser. Parity, Circuits, and the Polynomial-Time Hierarchy. *Mathematical Systems Theory*, 17(1):13–27, April 1984.

[GG] O. Gabber and Z. Galil. Explicit constructions of linear size superconcentrators. *Journal of Computer and System Sciences*, 22:407–420, 1981.

[Gil] D. Gillman. A Chernoff bound for random walks on expander graphs. *SIAM J. Comput.*, 27(4):1203–1220 (electronic), 1998.

[Gol1] O. Goldreich. A Sample of Samplers - A Computational Perspective on Sampling (survey). *Electronic Colloquium on Computational Complexity (ECCC)*, 4(020), 1997.

[Gol2] O. Goldreich. *Modern cryptography, probabilistic proofs and pseudorandomness*, volume 17 of *Algorithms and Combinatorics*. Springer-Verlag, Berlin, 1999.

[Gol3] O. Goldreich. *Foundations of Cryptography: Volume 1, Basic Tools*. Cambridge University Press, Cambridge, 2001.

[GL] O. Goldreich and L. A. Levin. A Hard-Core Predicate for all One-Way Functions. In *Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing*, pages 25–32, Seattle, Washington, 15–17 May 1989.

[GNW] O. Goldreich, N. Nisan, and A. Wigderson. On Yao's XOR lemma. Technical Report TR95–050, Electronic Colloquium on Computational Complexity, March 1995. `http://www.eccc.uni-trier.de/eccc`.

[GZ] O. Goldreich and D. Zuckerman. Another proof that BPP $\subseteq$ PH (and more). *Electronic Colloquium on Computational Complexity*, Technical Report TR97-045, September 1997. `http://www.eccc.uni-trier.de/eccc`.

[GS] V. Guruswami and M. Sudan. List decoding algorithms for certain concatenated codes. In *Proceedings of the Thirty-second Annual ACM Symposium on Theory of Computing*, pages 181–190, 21–23 May 2000.

[GV] D. Gutfreund and E. Viola. Fooling Parity Tests with Parity Gates. In *Proceedings of the Eight International Workshop on Randomization and Computation (RANDOM)*, Lecture Notes in Computer Science, Volume 3122, pages 381–392. Springer-Verlag, August 22–24 2004.

[HMP$^+$] A. Hajnal, W. Maass, P. Pudlák, M. Szegedy, and G. Turán. Threshold circuits of bounded depth. *J. Comput. System Sci.*, 46(2):129–154, 1993.

[HM] K. A. Hansen and P. B. Miltersen. Some Meet-in-the-Middle Circuit Lower Bounds. In *Proceedings of the 29th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, Lecture Notes in Computer Science, Volume 3153, pages 334 – 345, August 22–27 2004.

[Hås] J. Håstad. *Computational limitations of small-depth circuits*. MIT Press, 1987.

[HG] J. Håstad and M. Goldmann. On the power of small-depth threshold circuits. *Comput. Complexity*, 1(2):113–129, 1991.

[HVV] A. Healy, S. P. Vadhan, and E. Viola. Using Nondeterminism to Amplify Hardness. *SIAM J. Comput.*, 35(4):903–931, 2006.

[HV] A. Healy and E. Viola. Constant-Depth Circuits for Arithmetic in Finite Fields of Characteristic Two. In *Proceedings of the 23rd International Symposium on Theoretical Aspects of Computer Science (STACS)*, 2006. To appear.

[Imp1] R. Impagliazzo. Hard-core distributions for somewhat hard problems. In *36th Annual Symposium on Foundations of Computer Science*, pages 538–545, Milwaukee, Wisconsin, 23–25 Oct. 1995. IEEE.

[Imp2] R. Impagliazzo. A Personal View of Average-Case Complexity. In *Proceedings of the Tenth Annual Structure in Complexity Theory Conference*, pages 134–147. IEEE, 1995.

[INW] R. Impagliazzo, N. Nisan, and A. Wigderson. Pseudorandomness for Network Algorithms. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on the Theory of Computing*, pages 356–364, Montréal, Québec, Canada, 23–25 May 1994.

[IR] R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing*, pages 44–61, Seattle, Washington, 15–17 May 1989.

[ISW] R. Impagliazzo, R. Shaltiel, and A. Wigderson. Extractors and Pseudo-Random Generators with Optimal Seed Length. In *Proceedings of the Thirty-second Annual ACM Symposium on Theory of Computing*, pages 1–10, Portland, Oregon, May 2000. See also ECCC TR00-009.

[IW1] R. Impagliazzo and A. Wigderson. $P = BPP$ if E Requires Exponential Circuits: Derandomizing the XOR Lemma. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 220–229, El Paso, Texas, 4–6 May 1997.

[IW2] R. Impagliazzo and A. Wigderson. Randomness vs time: derandomization under a uniform assumption. *J. Comput. System Sci.*, 63(4):672–688, 2001. Special issue on FOCS 98.

[JM] S. Jimbo and A. Maruoka. Expanders obtained from affine transformations. *Combinatorica*, 7(4):343–355, 1987.

[Kah] N. Kahale. Eigenvalues and expansion of regular graphs. *J. Assoc. Comput. Mach.*, 42(5):1091–1106, 1995.

[KKL] J. Kahn, G. Kalai, and N. Linial. The Influence of Variables on Boolean Functions (Extended Abstract). In *29th Annual Symposium on Foundations of Computer Science*, pages 68–80, White Plains, New York, 24–26 Oct. 1988. IEEE.

[KS] A. Klivans and R. A. Servedio. Boosting and Hard-Core Sets. *Machine Learning*, 53(3):217–238, 2003.

[Kli] A. R. Klivans. On the Derandomization of Constant Depth Circuits. In *Proceedings of the Fifth International Workshop on Randomization and Approximation Techniques in Computer Science*, August 18–20 2001.

[KN] E. Kushilevitz and N. Nisan. *Communication complexity*. Cambridge University Press, Cambridge, 1997.

[Lau] C. Lautemann. BPP and the polynomial hierarchy. *Inform. Process. Lett.*, 17(4):215–217, 1983.

[Lev] L. A. Levin. Average Case Complete Problems. *SIAM J. on Computing*, 15(1):285–286, Feb. 1986.

[LMN] N. Linial, Y. Mansour, and N. Nisan. Constant depth circuits, Fourier transform, and learnability. *J. Assoc. Comput. Mach.*, 40(3):607–620, 1993.

[Lip] R. Lipton. New Directions in Testing. In *Proceedings of DIMACS Workshop on Distributed Computing and Cryptography*, volume 2, pages 191–202. ACM/AMS, 1991.

[LTW] C.-J. Lu, S.-C. Tsai, and H.-L. Wu. On the Complexity of Hardness Amplification. In *Proceedings of the Twentieth Annual Conference on Computational Complexity*, pages 170–182. IEEE, June 12–15 2005.

[LVW] M. Luby, B. Velickovic, and A. Wigderson. Deterministic Approximate Counting of Depth-2 Circuits. In *In Proceedings of the 2nd Israeli Symposium on Theoretical Computer Science (ISTCS)*, pages 18–24, 1993.

[MS] W. Maass and A. Schorr. Speed-up of Turing machines with one work tape and a two-way input tape. *SIAM J. Comput.*, 16(1):195–202, 1987.

[Mar] G. A. Margulis. Explicit construction of concentrator. *Problems Inform. Transmission*, 9:325–332, 1973.

[MO] E. Mossel and R. O'Donnell. On the noise sensitivity of monotone functions. *Random Struct. Algorithms*, 23(3):333–350, 2003.

[NN] J. Naor and M. Naor. Small-bias probability spaces: efficient constructions and applications. In *Proceedings of the 22nd Annual ACM Symposium on the Theory of Computing*, pages 213–223, 1990.

[Nep] V. A. Nepomnjaščiĭ. Rudimentary predicates and Turing calculations. *Soviet Mathematics-Doklady*, 11(6):1462–1465, 1970.

[Nis1] N. Nisan. Pseudorandom bits for constant depth circuits. *Combinatorica*, 11(1):63–70, 1991.

[Nis2] N. Nisan. Pseudorandom Generators for Space-bounded Computation. *Combinatorica*, 12(4):449–461, 1992.

[Nis3] N. Nisan. On read-once vs. multiple access to randomness in logspace. *Theoret. Comput. Sci.*, 107(1):135–144, 1993. Structure in complexity theory (Barcelona, 1990).

[NW] N. Nisan and A. Wigderson. Hardness vs Randomness. *J. Computer & Systems Sciences*, 49(2):149–167, Oct. 1994.

[O'D] R. O'Donnell. Hardness Amplification Within NP. *J. Comput. Syst. Sci.*, 69(1):68–94, Aug. 2004.

[Pap] C. H. Papadimitriou. *Computational Complexity*. Addison–Wesley, 1994.

[RW] A. Razborov and A. Wigderson. $n^{\Omega(\log n)}$ lower bounds on the size of depth-3 threshold circuits with AND gates at the bottom. *Inform. Process. Lett.*, 45(6):303–307, 1993.

[Raz] A. A. 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.

[RR] A. A. Razborov and S. Rudich. Natural Proofs. *J. Comput. Syst. Sci.*, 55(1):24–35, Aug. 1997.

[RTV] O. Reingold, L. Trevisan, and S. Vadhan. Notions of Reducibility between Cryptographic Primitives. In *Proceedings of the 1st Theory of Cryptography Conference (Feb 19-21, 2004: Cambridge, MA, USA)*. Springer-Verlag, 2004.

[Ros] S. Ross. *Probability Models for Computer Science*. Academic Press, Berlin, 2002.

[RS] A. Russell and R. Sundaram. Symmetric alternation captures BPP. *Comput. Complexity*, 7(2):152–162, 1998.

[SV1] A. Sahai and S. Vadhan. A complete problem for statistical zero knowledge. *J. ACM*, 50(2):196–249 (electronic), 2003.

[SV2] S. Sanghvi and S. Vadhan. The round complexity of two-party random selection. In *STOC '05: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 338–347, New York, NY, USA, 2005. ACM Press.

[SBI] N. Segerlind, S. Buss, and R. Impagliazzo. A switching lemma for small restrictions and lower bounds for k-DNF resolution. *SIAM J. Comput.*, 33(5):1171–1200 (electronic), 2004.

[SU1] R. Shaltiel and C. Umans. Simple Extractors for All Min-Entropies and a New Pseudo-Random Generator. In *42nd Annual Symposium on Foundations of Computer Science*. IEEE, 14–17 Oct. 2001.

[SU2] R. Shaltiel and C. Umans. Simple extractors for all min-entropies and a new pseudorandom generator. *J. ACM*, 52(2):172–216, 2005.

[Sip] M. Sipser. A Complexity Theoretic Approach to Randomness. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, pages 330–335, Boston, Massachusetts, 25–27 Apr. 1983.

[Smo] R. Smolensky. Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, pages 77–82, New York City, 25–27 May 1987.

[Sto] L. Stockmeyer. On Approximation Algorithms for #P. *SIAM J. on Computing*, 14(4):849–861, Nov. 1985.

[STV] M. Sudan, L. Trevisan, and S. Vadhan. Pseudorandom generators without the XOR lemma. *J. Comput. System Sci.*, 62(2):236–266, 2001. Special issue on the Fourteenth Annual IEEE Conference on Computational Complexity (Atlanta, GA, 1999).

[Tre1] L. Trevisan. Extractors and pseudorandom generators. *Journal of the ACM*, 48(4):860–879, 2001.

[Tre2] L. Trevisan. List Decoding Using the XOR Lemma. In *44th Annual Symposium on Foundations of Computer Science*, pages 126–135, Cambridge, Massachusetts, 11–14 Oct. 2003. IEEE.

[Tre3] L. Trevisan. Some applications of coding theory in computational complexity. In *Complexity of computations and proofs*, volume 13 of *Quad. Mat.*, pages 347–424. Dept. Math., Seconda Univ. Napoli, Caserta, 2004.

[TV] L. Trevisan and S. Vadhan. Pseudorandomness and Average-Case Complexity via Uniform Reductions. In *Proceedings of the 17th Annual IEEE Conference on Computational Complexity*, pages 129–138, Montréal, CA, May 2002. IEEE.

[Uma1] C. Umans. Pseudo-random generators for all hardnesses. In *Proceedings of the 34th ACM Symposium on Theory of Computing*, pages 627–634. ACM Press, 2002.

[Uma2] C. Umans. Pseudo-random generators for all hardnesses. *J. Comput. System Sci.*, 67(2):419–440, 2003. Special issue on STOC2002 (Montreal, QC).

[vM] D. van Melkebeek. Time-Space Lower Bounds for NP-Complete Problems. In G. Paun, G. Rozenberg, and A. Salomaa, editors, *Current Trends in Theoretical Computer Science*, pages 265–291. World Scientific, 2004.

[vMR] D. van Melkebeek and R. Raz. A Time Lower Bound for Satisfiability. In *Proceedings of the 31st International Colloquium on Automata, Languages and Programming (ICALP)*, pages 971–982, 2004.

[Vio1] E. Viola. The Complexity of Constructing Pseudorandom Generators from Hard Functions. *Comput. Complexity*, 13(3-4):147–188, 2004.

[Vio2] E. Viola. On Constructing Parallel Pseudorandom Generators from One-Way Functions. In *Proceedings of the Twentieth Annual Conference on Computational Complexity*, pages 183–197. IEEE, June 12–15 2005.

[Vio3] E. Viola. Pseudorandom Bits for Constant-Depth Circuits with Few Arbitrary Symmetric Gates. In *Proceedings of the Twentieth Annual Conference on Computational Complexity*, pages 198–209. IEEE, June 12–15 2005.

[Yao1] A. C. Yao. Theory and Applications of Trapdoor Functions (Extended Abstract). In *23rd Annual Symposium on Foundations of Computer Science*, pages 80–91, Chicago, Illinois, 3–5 Nov. 1982. IEEE.

[Yao2] A. C. Yao. On ACC and threshold circuits. In *In Proc. 31st Ann. IEEE Symp. Found. Comput. Sci.*, pages 619–627, 1990.

Appendix A

Chernoff Bound

The following useful bound, due to Chernoff [Che], shows that if one has n independent events, each which occurs with probability p, then roughly $n \cdot p$ of the events occur with high probability.

Theorem A.1. *Suppose $X_1, X_2, \ldots, X_n$ are independent random variables such that for all i, $\Pr[X_i = 1] = p$ and $\Pr[X_i = 0] = 1 - p$. Then, for every $\delta \geq 0$,*

$$\Pr\left[\left|\frac{\sum_{i \leq n} X_i}{n} - p\right| \geq \delta\right] \leq c^{n \cdot \delta^2},$$

where $c < 1$ is a universal constant.

See Corollary 3.1.2 in [Ros] for a proof of Theorem A.1.

Appendix B

Alternating time versus constant-depth circuits

For completeness, in this section we include a proof of the following connection between alternating oracle computation and constant-depth circuits.

Theorem B.1 ([FSS])**.** *Let $M^f : \{0,1\}^{n'} \to \{0,1\}$ be an oracle machine, where $f : \{0,1\}^l \to \{0,1\}$. Suppose that M uses $d-1$ alternations and runs in time t for every $x \in \{0,1\}^{n'}$ and every oracle f.*

Then, for every $x \in \{0,1\}^{n'}$, there is a constant-depth circuit C_x of depth $d+1$ and size $2^{O(d \cdot t)}$ that given the truth-table of f computes $M^f(x)$. Moreover, the bottom fan-in of C_x is at most t/l.

Before proving the result, we would like to make a remark regarding oracle machines.

Remark B.2 (On the oracle tape)**.** *The bound on the bottom fan-in of the circuit in the last sentence of the statement of the theorem is only needed in one result in this thesis, specifically Theorem 5.5 in Chapter 5. The bound relies on the convention that the oracle tape of an oracle machine is* erased *after each query, and thus in particular that an oracle machine that runs in time t makes at most t/l oracle queries of length l. This convention arises naturally when studying the running time of oracle computations, is standard in the literature (see, e.g., [BDG]), and is discussed more in Chapter 5 where we need it.*

Proof. By a simple simulation, there is an oracle machine $\bar{M}$ that satisfies

$$M^f(x) = 1 \Leftrightarrow Q_1 y_1 Q_2 y_2 \ldots Q_d y_d \bar{M}^f(x, y_1, y_2, \ldots, y_d) = 1, \qquad \text{(B.1)}$$

where Q_i is either $\exists$ or $\forall$ according to the alternations of M, $|y_i| = t$ for every i, and $\bar{M}$ is a deterministic oracle algorithm whose number of oracle queries is bounded from above by the maximum number of oracle queries M makes over all computation paths. Since each oracle query is of length l and the oracle tape is erased after each query (see Remark B.2), the machine M can only ask t/l queries, and consequently so does $\bar{M}$. Therefore, for fixed $x, y_1, y_2, \ldots, y_d$ the output of $\bar{M}$ depends on at most t/l bits of the truth-table of f (possibly chosen adaptively). This computation can be carried out by a circuit $\bar{C}$ of depth 2 and size $2^{O(t/l)}$ with bottom fan-in t/l in a brute-force fashion. The output gate of $\bar{C}$ can be chosen to be And or Or. The result then follows by viewing the quantifiers in Equation B.1 as And and Or gates, and collapsing the gates corresponding to Q_d with the output gate of $\bar{C}$. $\square$