The Computational Complexity of Randomness

by

Thomas Weir Watson

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Luca Trevisan, Co-Chair
Professor Umesh Vazirani, Co-Chair
Professor Satish Rao
Professor David Aldous

Spring 2013

Abstract

The Computational Complexity of Randomness

by

Thomas Weir Watson

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Luca Trevisan, Co-Chair
Professor Umesh Vazirani, Co-Chair

This dissertation explores the multifaceted interplay between *efficient computation* and *probability distributions.* We organize the aspects of this interplay according to whether the randomness occurs primarily at the level of the problem or the level of the algorithm, and orthogonally according to whether the output is random or the input is random.

Part I concerns settings where *the problem's output is random.* A sampling problem associates to each input x a probability distribution $D(x)$, and the goal is to output a sample from $D(x)$ (or at least get statistically close) when given x. Although sampling algorithms are fundamental tools in statistical physics, combinatorial optimization, and cryptography, and algorithms for a wide variety of sampling problems have been discovered, there has been comparatively little research viewing sampling through the lens of computational complexity. We contribute to the understanding of the power and limitations of efficient sampling by proving a time hierarchy theorem which shows, roughly, that "a little more time gives a lot more power to sampling algorithms."

Part II concerns settings where *the algorithm's output is random.* Even when the specification of a computational problem involves no randomness, one can still consider randomized algorithms that produce a correct output with high probability. A basic question is whether one can derandomize algorithms, i.e., reduce or eliminate their dependence on randomness without incurring much loss in efficiency. Although derandomizing arbitrary time-efficient algorithms can only be done assuming unproven complexity-theoretic conjectures, it is possible to unconditionally construct derandomization tools called pseudorandom generators for restricted classes of algorithms. We give an improved pseudorandom generator for a new class, which we call combinatorial checkerboards. The notion of pseudorandomness shows up in many contexts besides derandomization. The so-called Dense Model Theorem, which has its roots in the famous theorem of Green and Tao that the primes contain arbitrarily long arithmetic progressions, is a result about pseudorandomness that has turned out to be a useful tool in computational complexity and cryptography. At the heart of this theorem is a certain type of reduction, and in this dissertation we prove a strong lower bound on the advice complexity of such reductions, which is analogous to a list size lower bound for list-decoding of error-correcting codes.

Part III concerns settings where *the problem's input is random.* We focus on the topic of randomness extraction, which is the problem of transforming a sample from a high-entropy but non-uniform probability distribution (that represents an imperfect physical source of randomness) into a uniformly distributed sample (which would then be suitable for use by a randomized algorithm). It is natural to model the input distribution as being sampled by an efficient algorithm (since randomness is generated by efficient processes in nature), and we give a construction of an extractor for the case where the input distribution's sampler is extremely efficient in parallel time. A related problem is to estimate the min-entropy ("amount of randomness") of a given parallel-time-efficient sampler, since this dictates how many uniformly random output bits one could hope to extract from it. We characterize the complexity of this problem, showing that it is "slightly harder than NP-complete."

Part IV concerns settings where *the algorithm's input is random.* Average-case complexity is the study of the power of algorithms that are allowed to fail with small probability over a randomly chosen input. This topic is motivated by cryptography and by modeling heuristics. A fundamental open question is whether the average-case hardness of NP is implied by the worst-case hardness of NP. We exhibit a new barrier to showing this implication, by proving that a certain general technique (namely, "relativizing proofs by reduction") cannot be used. We also prove results on hardness amplification, clarifying the quantitative relationship between the running time of an algorithm and the probability of failure over a random input (both of which are desirable to minimize).

Contents

Acknowledgments

First I thank my doctoral advisor Luca Trevisan. Through many fruitful discussions, he helped me navigate the waters of research by providing just the nuggets of advice I needed to help me on my way. Luca suggested several research topics that directly or indirectly led to some of my papers, and I am very grateful for that. I hope Luca's uncanny ability to see through the facades of seemingly well-understood problems and perceive their true nature has rubbed off on me.

Next I thank my undergraduate advisor Dieter van Melkebeek, who had immeasurable influence on my understanding of theoretical computer science. Dieter's ultra-methodical teaching style, impeccable taste in research, stimulating homework problems, and exceptional clarity in writing have been a tremendous inspiration. He has also been a constant source of counsel and encouragement throughout my years in graduate school.

I thank David Aldous, Satish Rao, Luca Trevisan, and Umesh Vazirani for serving on my qual and dissertation committees, and Alistair Sinclair and Yun Song for serving on my prelim committee.

For their guidance in pedagogy, I thank the professors for whom I was a graduate student instructor: James Demmel and Alistair Sinclair. I also thank my graduate student instructor colleagues: Siu Man Chan, Lyle Mills, Dave Moore, Hoa Long Tam, Ethan Van Andel, Oriol Vinyals, and Tobias Welp. In addition, I thank my many students; I learned as much from them as I hope they did from me.

I thank my coauthors, Anindya De and Dieter van Melkebeek, for the insights about conducting and presenting research that I gained from collaborating with them.

I thank all the people who provided useful feedback on my papers or helped with my research in other ways: Stefan Arnold, Anand Bhaskar, Siu Man Chan, Siu On Chan, Sue Coppersmith, Anindya De, Andrew Drucker, Scott Diehl, Zeev Dvir, Oded Goldreich, Parikshit Gopalan, Urmila Mahadev, Dieter van Melkebeek, Anup Rao, Omer Reingold, Ronen Shaltiel, Luca Trevisan, Salil Vadhan, Gregory Valiant, Emanuele Viola, John Watrous, and Avi Wigderson, as well as many anonymous conference referees and journal reviewers. I also thank the organizers of the Dagstuhl workshop on Computational Complexity of Discrete Problems for inviting me to attend the workshop and present my research.

I thank the National Science Foundation for financially supporting much of my research through the Graduate Research Fellowship Program. I also thank the many great restaurants of Berkeley for fueling much of my research.

I had numerous professors at Madison and at Berkeley who blew me away with their fascinating lectures. Their instruction and personalities have influenced my perspective on scholarship and life in general in many ways that I cannot even articulate.

For presiding over the theory group and making it so conducive to learning and collaboration, I thank all of Berkeley's theory professors: Richard Karp, Elchanan Mossel, Christos Papadimitriou, Prasad Raghavendra, Satish Rao, Jonathan Shewchuk, Alistair Sinclair, Yun Song, Luca Trevisan, and Umesh Vazirani.

I give a special thanks to my excellent officemates Anand Bhaskar, Siu Man Chan, and Siu On Chan for never failing to entertain me.

For creating an environment of general camaraderie and intellectual excellence, I thank all the theory graduate students I knew at Berkeley: Jake Abernethy, Nima Anari, Anand Bhaskar, Antonio Blanca, Ma'ayan Bresler, Brielin Brown, Jonah Brown-Cohen, Andrew Chan, Siu Man Chan, Siu On Chan, Paul Christiano, James Cook, Anindya De, Kevin Dick, Milos Drezgic, Omid Etesami, Raf Frongillo, Luqman Hodgkinson, Sakshi Jain, Bonnie Kirkpatrick, Alexandra Kolla, Henry Lin, Urmila Mahadev, Lorenzo Orecchia, George Pierrakos, Anupam Prakash, Daniel Preda, Christos-Alexandros Psomas, Aviad Rubinstein, Grant Schoenebeck, Tselil Schramm, Jarett Schwartz, Sara Sheehan, Jonah Sherman, Seung Woo Shin, Meromit Singer, Yaron Singer, Andrew Smith, Karolina Soltys, Piyush Srivastava, Isabelle Stanton, Alexandre Stauffer, Ning Tan, Madhur Tulsiani, Greg Valiant, Thomas Vidick, Di Wang, Guoming Wang, Benjamin Weitz, and Chris Wilkens. I also thank all the theory graduate students I knew when I was an undergraduate at Madison: Matt Anderson, Baris Aydinlioglu, Siddharth Barman, Bess Berg, Scott Diehl, Matt Elder, Jeff Kinne, Mike Kowalczyk, Chi Man Liu, David Malec, William Umboh, and Dalibor Zeleny. I also thank the many people who helped make my trips to various conferences fun and memorable, including Matt Anderson, Holger Dell, Andrew Drucker, Jeff Kinne, Carla Rafols Salvador, and Steve Uurtamo.

Finally, I thank my family for their unwavering support, encouragement, and love.

To my parents

Chapter 1

Introduction

Since this dissertation is about "the computational complexity of randomness," we begin with a discussion of the terms *computational complexity* and *randomness*.

1.1 Computational Complexity

Computational complexity theory is often defined as the study of the inherent power and limitations of efficient computation. Let us explain what this theory is about using the six basic questions: *who, what, where, when, why, how?*

Who? *Who* should care about computational complexity? Everyone! Computation is ubiquitous in the 21st century, and almost every branch of computer science has deep, fundamental questions that are addressed by computational complexity. For example, computation in the presence of adversarial behavior, in decentralized settings, and by intelligent agents are areas where basic questions about feasibility fall under the scope of computational complexity. If computer science were an apple tree, then computational complexity would be the roots — although you do not pick apples from the roots, they are there, feeding the tree.

What? Using a machine to perform a computation involves feeding an input to the machine, which later produces an output. *What* is the computation supposed to accomplish? What is the goal? This is specified by a computational *problem*, which in the most general terms is a relationship between inputs and outputs, a specification of what properties the output should satisfy for each possible input. Computational problems can take on a variety of forms, among the most common being function evaluation problems (where the desired output is unique for each input), search problems (where there are potentially many acceptable outputs), and optimization problems (where an acceptable output optimizing some objective function is sought).

How? *How* does a machine solve a problem? This is the notion of an *algorithm*. Computational complexity is robust with respect to the details of how "algorithm" is defined. The model of computation does not matter — any reasonable programming language can be used to define what an algorithm is. For solving a problem, it is not good enough just to know that an algorithm is correct. We also want the algorithm to be efficient, meaning that it uses only small amounts of computational resources such as running time and memory space. The consumption of such resources is viewed as a measure of the complexity of an algorithm. We are interested in the asymptotic behavior of these complexity measures; that is, we examine the rate of growth of resource consumption in terms of the “size” of the input. For many natural problems, the naïve approaches to solving them are based on exhaustive search and lead to very inefficient algorithms. For some of these problems, clever algorithms have been devised that drastically reduce the resource consumption by exploiting special structure inherent in the problem. Other problems have resisted the best efforts of algorithm designers, which suggests these problems may be inherently intractable.

When? Given two algorithms for a problem, it is natural to consider the one that finishes earlier to be preferable. *When* does the computation finish? As mentioned above, *time* is one of the most important computational resources. This corresponds to the number of elementary steps an algorithm takes before it halts.

Where? Given two algorithms for a problem, it is natural to consider the one that uses less memory to be preferable. *Where* does the computation store its intermediate work? As mentioned above, *space* is one of the most important computational resources. If an algorithm is able to “recycle” the memory it uses throughout the computation by overwriting old “scratch work” with new scratch work, then it will consume less space.

Why? The central question in computational complexity is: *Why* is computation the way it is? Why are efficient algorithms capable of solving certain problems but not others? The goal is not just to obtain the facts about what algorithms can and cannot do, but to obtain insight into the nature of computation itself. Rather than focusing on individual computational problems that people want to solve, the field of computational complexity tends to focus on problems of a very general nature, and on entire classes of problems defined in terms of how efficiently they can be solved.

1.2 Randomness

The meaning of the term “randomness” is a matter of philosophical debate, but it is generally understood to involve uncertainty or unpredictability. For our purposes, randomness is described using the mathematical theory of discrete probability. A random experiment is represented by a probability distribution, which assigns a numerical probability to each possible outcome. This models situations such as a flip of a coin, a roll of a die, and a shuffle of a deck of cards. Distributions need not be uniform: For example, if a deck of cards starts in a particular order, then after a single imperfect shuffle, some orders will be more likely than others, since there will still be significant correlations among the locations of different cards.

Randomness is an indispensable tool in many areas of computer science. Motivated by such applications, we consider four basic ways randomness crops up, organized in a 2×2 matrix as follows: Randomness can be *inherent in the problem*, with either the desired output being a probability distribution or the given input being a probability distribution, or *useful for the algorithm*, as either a resource used to generate the output or a model for how the input is generated. We use this matrix as the structure of this dissertation. In this section we elaborate on the meaning and context of these four categories. The boundaries between these categories are not sharp; there are many overlaps and connections. In intricate computational settings, different components may naturally fall into different categories. Nevertheless, each of the specific topics we study in this dissertation has an "affinity" for a particular category.

	Problem	Algorithm
Output is random	Part I	Part II
Input is random	Part III	Part IV

The problem's output is random. In addition to function evaluation problems, search problems, and optimization problems (mentioned in Section 1.1), another important type of computational problem is *sampling problems*, for which randomness is inherent in the output of the problem. A sampling problem associates to each input x a probability distribution $D(x)$, and the goal is to output a sample from $D(x)$ (or at least get "statistically close") when given x. For example, x might encode a graph, and $D(x)$ might be uniform over all the matchings in the graph.

Sampling algorithms are fundamental tools in statistical physics, combinatorial optimization, and cryptography, and algorithms for a wide variety of sampling problems have been discovered. Comparatively little is known about the inherent limitations of sampling algorithms. We would like to learn why efficient algorithms can solve some sampling problems but not others, by viewing sampling through the lens of computational complexity.

The algorithm's output is random. Even when the specification of a computational problem involves no randomness, we can still consider algorithms whose output is random. In contrast to the previous category, where algorithms were necessarily randomized since the output had to be random to be correct, we now consider problems for which there can conceivably exist deterministic algorithms. Traditionally, randomized algorithms have been grouped into two types: ones that are always correct and probably efficient (also known as Las Vegas algorithms), and ones that are always efficient and probably correct (also known as Monte Carlo algorithms). The latter type is at least as powerful as the former type, since if an algorithm is probably efficient then it can be modified to halt and output "don't know" when its resource consumption is about to become too great (so the modified algorithm would be always efficient and probably correct).

Algorithms that produce a random output can be viewed as deterministic algorithms that take two inputs: the main input, and an auxiliary input representing the randomness used by the algorithm. The auxiliary input is generally a sequence of independent unbiased coin flips, in other words, a uniformly random bit string. In this way, randomness can be viewed as a computational resource. A shorter auxiliary input is preferable to a longer one, since obtaining a large amount of uniform randomness may be impractical. This suggests there may be tradeoffs between randomness and other computational resources. For many problems, randomized algorithms are more time- or space-efficient than known deterministic algorithms. In some cases they are *much* more efficient, but is this inherent, or do there exist deterministic algorithms of comparable efficiency to the randomized ones? A natural objective is to derandomize algorithms, i.e., reduce or eliminate their dependence on randomness without incurring much loss in time- or space-efficiency. Famous derandomization successes include time-efficient primality testing [4] and space-efficient undirected graph connectivity testing [218]. Computational complexity is especially concerned with derandomizing entire *classes* of algorithms. Can deterministic algorithms always perform almost as well as randomized ones?

The problem's input is random. For some computational problems, the input is a probability distribution. An algorithm may be given a description of the distribution, or it may only have access to one or more samples from the distribution. In either case, the distribution may or may not be assumed to have some sort of succinct description.

One topic in this category is randomness extraction, which addresses the following scenario. Suppose there is a randomized algorithm that assumes access to many uniformly random bits, but the available physical source of randomness is imperfect and does not produce a uniform sample, although it does contain a sufficient "amount" of randomness. Extraction is the problem of taking a sample from the source's distribution and transforming it into an output that is uniformly distributed (and can thus be fed into a randomized algorithm). An algorithm that solves this problem is called an extractor, and it is generally assumed that the extractor does not know what distribution it is trying to extract from (it only receives a sample from the distribution). Without any restrictions on the source (except that it contains a sufficient amount of randomness), it turns out that no such deterministic extractor can exist. However, under various assumptions about the source, extraction becomes possible. Designing extractors reveals why efficient computation is robust with respect to imperfections in sources of randomness.

The algorithm's input is random. From one perspective, the problem's input *is* the algorithm's input, so there is no point in making a distinction between the two. But when we talk about the input being random, we adopt a convention that distinguishes the two categories. In the previous category, the input was a probability distribution, and an algorithm was required to succeed on each possible input. In the present category, the input can be anything (a graph, say), and there is a distribution over inputs (or more precisely, an ensemble of distributions, one for each "input size"). Average-case complexity is the study of the power and limitations of algorithms that are allowed to fail with some probability over such a randomly chosen input. In the standard setting of average-case complexity, an algorithm may output anything when it "fails" (and need not give any indication that its output is wrong). In the alternative *errorless* setting, an algorithm is only allowed to output the correct answer or "don't know" and must never output an incorrect answer.

Average-case complexity has two main motivations. One is modeling heuristics — it is desirable to understand why some problems have algorithms that work well in practice even though no algorithm is known to be both correct and efficient on all inputs. The other motivation is cryptography, which rests on the assumption that certain problems are average-case hard (meaning no efficient algorithm can succeed with high probability over a random input). In cryptographic settings, allowing an adversary to succeed most of the time, or even a nonnegligible fraction of the time, would be completely unacceptable.

1.3 Our Contributions

In this section we summarize the results proven in this dissertation. The subsequent chapters (2 through 8) can each be read independently of the others. We strive to keep the presentation fairly self-contained throughout this dissertation, but at times we assume some familiarity with the basics of computational complexity, for which we refer the reader to the textbooks [17, 98].

1.3.1 Part I

Time seems to be a very valuable computational resource, but is that an illusion? How can we formalize what it means for time to be valuable? This question is fundamental to our understanding of computation. The answer offered by computational complexity comes in the form of *time hierarchy* theorems, which are results showing that "more time gives more power to algorithms." In other words, a time hierarchy shows that there exist problems that can be solved by algorithms running in a given time bound but not by algorithms running in a somewhat smaller time bound. In what is widely regarded as the paper that founded the field of computational complexity, Hartmanis and Stearns [128] proved a time hierarchy theorem for deterministic algorithms solving problems with yes/no answers (which are called decision problems or languages). Subsequent work has shown time hierarchies for other models of computation solving decision problems, though for randomized algorithms this remains a major open question.

In Chapter 2 (which is based on [272]) we revisit this topic by proving a time hierarchy theorem for sampling algorithms. This illustrates a hierarchy phenomenon for randomized algorithms in a context alternative to decision problems. We prove what we call a robust time hierarchy, showing that "a little more time gives *a lot* more power to sampling algorithms." In other words, there are distributions that can be sampled by algorithms running in a given time bound, but which algorithms running in a somewhat smaller time bound cannot even come close to sampling, where closeness is in statistical distance. For probability distributions over constant-size domains, we prove the best possible result in terms of the statistical distance. Formally, we prove that for every constant $k \geq 2$, every polynomial time bound t, and every polynomially small ϵ, there exists a family of distributions on k

elements that can be sampled exactly in polynomial time but cannot be sampled within statistical distance $1 - 1/k - \epsilon$ in time t. Statistical distance can never exceed 1, and over a domain of size k the uniform distribution is within statistical distance $1 - 1/k$ of every distribution simultaneously, so the above theorem would be false if we eliminated the ϵ term. As a corollary, we obtain the following general time hierarchy for sampling distributions on arbitrary-size domains such as $\{0,1\}^n$: For every polynomial time bound t and every constant $\epsilon > 0$, there exists a family of distributions that can be sampled exactly in polynomial time but cannot be sampled within statistical distance $1 - \epsilon$ in time t.

The proof of our time hierarchy involves reducing the situation to a communication problem over a certain type of noisy channel. To solve the latter problem we use a new type of list-decodable error-correcting code. Such codes can be constructed using certain known traditional list-decodable codes, but we give a new construction that is elementary, self-contained, and tailored to this setting.

Aside from the purely algorithmic motivation for studying sampling problems, another motivation comes from cryptography. A fundamental component in cryptographic settings is the ability to sample distributions of a certain form, namely *solved instances* (i.e., input-output pairs) of hard problems. Note that if a function is easy to evaluate, then input-output pairs can be sampled by just picking a uniformly random input and then computing the corresponding output. Hence, finding explicit functions for which input-output pairs are hard to sample is tougher than finding explicit functions that are hard to evaluate. We prove a lower bound on the complexity of sampling input-output pairs of a certain explicit function. This result appears in Part III (Chapter 5) since it is closely connected to our result on randomness extraction.

1.3.2 Part II

The most common and successful method for derandomizing classes of algorithms is to construct a pseudorandom generator, which is an efficient algorithm that "fools" every algorithm in the class by producing a "pseudorandom" output that can be used in place of the truly random bits expected by the algorithm. A pseudorandom generator uses a short truly random seed, and the main goal in constructing pseudorandom generators is to minimize the seed length. Full derandomization can be achieved by iterating over all seeds, so to minimize the time overhead the seed should be short.

Although derandomizing arbitrary time-efficient algorithms can only be done assuming certain major unproven conjectures, a successful research goal has been to unconditionally construct pseudorandom generators for restricted classes of algorithms. The seminal work of Impagliazzo, Nisan, and Wigderson [197, 142] gives pseudorandom generators for space-efficient algorithms, though the seed length is suboptimal. Many of the other classes that have been studied can be viewed as special cases of space-efficient algorithms, and the goal for these classes has been to beat the seed length known for the general case. We contribute to this project in Chapter 3 (which is based on [270, 267]) by constructing an improved pseudorandom generator for a new class we dub *combinatorial checkerboards*.

One natural way to define a class is to consider computations that combine the results of many smaller independent computations in a simple manner. The combining function can be thought of as a building block for forming more complex computations. We define combinatorial checkerboards to be the class of such computations where the combiner is the parity function. (Combinatorial rectangles, which have been previously studied, are the case where the combiner is the AND function.) Formally, a combinatorial checkerboard is a function $f : \{1, \ldots, m\}^d \to \{1, -1\}$ of the form $f(u_1, \ldots, u_d) = \prod_{i=1}^{d} f_i(u_i)$ for some functions $f_i : \{1, \ldots, m\} \to \{1, -1\}$ (multiplication of ± 1 values corresponds to parity, i.e., XOR). The sense in which such functions can be viewed as space-efficient is that, between evaluating successive f_i's, only one bit of information needs to be kept track of, namely the parity of the f_i's evaluated so far. If a combinatorial checkerboard f came from a randomized algorithm solving some problem, then the algorithm's auxiliary random input would correspond to f's input (a sequence of d rolls of an m-sided die), while the true input to the algorithm would be implicitly hard-wired into f itself.

We construct a pseudorandom generator that fools combinatorial checkerboards within ϵ using a seed of length $O\big(\log m + \log d \cdot \log\log d + \log^{3/2} \frac{1}{\epsilon}\big)$. This significantly improves on the seed length of $O\big(\log m + \log^2 d + \log d \cdot \log \frac{1}{\epsilon}\big)$ achieved by [142] (except when $\frac{1}{\epsilon} \geq d^{\omega(\log d)}$), and it comes close to having optimal dependence on all the parameters, which would be $O\big(\log m + \log d + \log \frac{1}{\epsilon}\big)$.

Pseudorandomness is a special case of the more general notion of computational indistinguishability. Two distributions are considered computationally indistinguishable from each other if no efficient algorithm (under some notion of efficiency) can distinguish them in the sense that the probability the algorithm accepts (outputs 1) is significantly different when its input is a sample from one distribution versus the other distribution. A distribution is considered pseudorandom if it is computationally indistinguishable from the uniform distribution. Hence, the goal of a pseudorandom generator is to produce a pseudorandom output distribution when given a uniformly random seed.

The notions of pseudorandomness and computational indistinguishability show up in many contexts besides derandomization. The so-called Dense Model Theorem [219], which has its roots in the famous theorem of Green and Tao [115] that the primes contain arbitrarily long arithmetic progressions, is a result about these notions that has turned out to be a useful tool in computational complexity and cryptography. We forgo details about the statement of the Dense Model Theorem (including the meaning of "dense") in this introductory chapter, but it can be phrased, roughly, as "if a distribution is dense in some pseudorandom distribution then it is computationally indistinguishable from some dense distribution." At the heart of the Dense Model Theorem is a certain type of reduction: an algorithm that refutes the former pseudorandomness when given black-box access to hypothetical algorithms that refute the latter computational indistinguishability. In the theorem statement there is a "gap" between the efficiency of the distinguishers considered on the former side and those on the latter side, and this gap is filled in by the efficiency of the reduction itself. This motivates the study of the complexity of such reductions; a lower bound on the complexity implies that a gap is inherent.

Here we depart from the usual time and space measures of complexity, and we consider measures that are more combinatorial in nature. The *query* complexity (number of black-box "calls" the reduction makes to the hypothetical algorithms) was studied by Zhang [278]. In Chapter 4 (which is based on [266]) we prove a strong (though not tight) lower bound on the *advice* complexity for the Dense Model Theorem. The notion of advice shows up in many areas of computational complexity, and it always refers to a trusted piece of information that is provided to an algorithm. In our setting, a reduction receives indirect, incomplete information about the original distribution via the hypothetical algorithms, and the reduction needs to do a kind of "decoding" to learn about the distribution. The advice complexity is a measure of the "amount of ambiguity" that needs to be overcome in the decoding. In addition to clarifying issues about the parameters of the Dense Model Theorem, our lower bound proof contributes techniques that may be of independent interest, such as a lemma about majority-of-majorities circuits.

1.3.3 Part III

Recall that for randomness extraction to be possible, we need to assume that the source distribution comes from some restricted class of distributions. A very natural type of class to consider is efficiently samplable distributions. In nature, randomness is generated by efficient processes, so it is reasonable to model a physical source as an efficient sampling algorithm. (However, this type of sampling algorithm does not fall under the scope of Part I because it is not "trying to solve" any problem.) The situation here is similar to the situation with pseudorandom generators: Although extractors for sources with arbitrary time-efficient samplers can only be constructed assuming certain major unproven conjectures, it is reasonable to hope for unconditional constructions of extractors for samplers satisfying more stringent efficiency constraints. This hope was fulfilled for space-efficient samplers in [153].

In Chapter 5 (which is based on joint work with Anindya De [70, 69]) we unconditionally construct extractors for distributions with *parallel-time-efficient* samplers. We consider the most extreme form of parallel-time-efficiency, where each output bit of the sampler only depends on a limited number d of the random input bits. When $d \leq o(\log n)$ and the distribution is on n bits and has $k \geq n^{2/3+\gamma}$ bits of min-entropy (which is the standard measure of the "amount" of randomness in this context) for any constant $\gamma > 0$, our extractor extracts $\Omega(k^2/nd)$ bits that are $2^{-n^{\Omega(1)}}$-close to uniform. Why are such parallel-time-efficient algorithms amenable to having their randomness extracted? The key insight involves showing a connection to so-called low-weight affine sources, for which it is already known (unconditionally) how to extract randomness [209].

The number of almost-uniformly random bits produced by an extractor depends on the source distribution's min-entropy, for which an a priori lower bound is assumed to be known. A natural question is whether, given the description of a sampler, we can efficiently estimate the min-entropy of its distribution so that we can plug this parameter into the extractor and thus extract as much of the randomness as possible. In Chapter 6 (which is based on [269]) we characterize the complexity of this min-entropy estimation problem by showing that an appropriate formulation is "SBP-complete" (and thus unlikely to be solvable in polynomial time). Our result holds even when the sampler is restricted to be parallel-time-efficient (as in our extractor result).

Note that in both our randomness extraction problem and our min-entropy estimation problem, the problem's input is a distribution with a certain type of sampler. However, in the former case an algorithm is only given a sample as input, whereas in the latter case an algorithm is given the description of the sampler as input (from which the algorithm could generate its own samples if it wanted).

1.3.4 Part IV

The complexity class NP consists of all problems with yes/no answers for which a polynomial-time algorithm can be "convinced" that any given yes-input is indeed a yes-input, provided the algorithm is given an appropriate "proof" (also called a witness or certificate). This class captures the complexity of search problems for which acceptable outputs can be verified in polynomial time.

What makes NP so significant is that a huge number of practically important problems are among the "hardest" problems in NP, the so-called NP-complete problems. These problems are all equivalent to each other, in the sense that either they can all be solved in polynomial time or none of them can. It is widely believed that the latter is the case; this is expressed as the famous P $\neq$ NP conjecture. But this conjecture merely asserts that no polynomial-time algorithm can succeed in the worst case (meaning on *all* inputs). Can NP-complete problems be efficiently solved in the average case (meaning with high probability over a random input)? Secure cryptography requires that the answer is no. A central open question in average-case complexity is whether the average-case hardness of NP is implied by the worst-case hardness of NP. An affirmative answer would strengthen our confidence in secure cryptography.

The proposition that "if NP is worst-case hard then NP is average-case hard" is certainly believed to be true, since both sides are believed to be true. A proof of this implication has turned out to be quite elusive. Why? One research goal has been to exhibit barriers to proving the implication, by ruling out certain general proof techniques, thereby illuminating the core difficulties of the issue and explaining why we need separate theories for worst-case and average-case complexity. In Chapter 7 (which is based on [271, 264]) we contribute to this project by ruling out *relativizing proofs by reduction.* To describe what this means, we first discuss reductions and then discuss relativization.

The notion of a reduction is central in computational complexity. A reduction is, generally speaking, an efficient algorithm that solves a problem A given the ability to make calls to a hypothetical algorithm (known as an oracle) that solves another problem B. The existence of a reduction from A to B shows that if B is "easy" (has an efficient algorithm) then A is also easy, since B's algorithm could be plugged into the reduction to obtain an algorithm for A. Conversely, a reduction shows that if A is hard then B is hard. A *worst-case to average-case reduction for* NP is a polynomial-time algorithm that solves an NP-complete problem (and hence, indirectly, all NP-complete problems) in the worst case given a hypothetical algorithm that solves some problem in NP in the average case. The existence of such a reduction would prove the elusive implication. The nonexistence of such a reduction would mean that efficient algorithms are not capable of bridging the gap between worst-case and average-case complexity.

The notion of relativization is syntactically similar to the notion of a reduction, but it has a different purpose altogether. A relativizing proof, roughly speaking, is one that manipulates algorithms without looking at or "reverse-engineering" their code. Formally, relativizing to a problem C means giving all algorithms the ability to make queries to an oracle for C (i.e., to obtain answers to inputs for C in unit time each). If a proof of a proposition about computation does not exploit any specific properties of algorithms' code, then the proof will carry through when all algorithms are "relative to" any oracle C. A proof with the latter property is said to relativize. If we exhibit a C relative to which a proposition is false, then that does not necessarily mean the proposition is false in the unrelativized world, but it implies that the proposition cannot be proven by relativizing techniques. Hence the exhibition of the oracle constitutes a barrier to proving the proposition. Note that C need not be a "natural" problem, since the point is not to show that if C is easy then another problem is easy. The main technique for constructing oracles is called diagonalization (we also use diagonalization to prove our time hierarchy theorem in Part I), which produces oracles that are contrived but nevertheless yield a barrier. Almost all known proofs of theorems about complexity classes relativize, with a few notable exceptions such as IP = PSPACE [232, 233]. Breaking a relativization barrier requires novel insights about the inner workings of efficient algorithms.

We show that relativizing proofs by reduction cannot be used to base the average-case hardness of NP on the worst-case hardness of NP. That is, we exhibit an oracle relative to which there is no worst-case to average-case reduction for NP. We also handle classes that are somewhat larger than NP, as well as worst-case to *errorless*-average-case reductions.

The goal of transforming worst-case hard problems into average-case hard problems can be seen as an extreme form of what is known as hardness amplification. A less extreme form (which also has applications in cryptography and derandomization) aims to transform mild average-case hardness into stronger average-case hardness. A basic objective is to understand the quantitative relationship between an algorithm's running time and its probability of failure over a random input (both of which are desirable to minimize). In Chapter 8 (which is based on [268]) we prove a hardness amplification result with optimal tradeoff between the two for the *errorless* setting. (An optimal tradeoff in the standard, non-errorless setting was already known.) We now give a more precise description of our result.

A circuit is a structure of logic gates representing the computation of an algorithm on inputs of a fixed bit length (and circuit size roughly corresponds to running time). The goal of errorless hardness amplification is to show that if a boolean function f has no size s errorless circuit that outputs "don't know" on at most a δ fraction of inputs, then some boolean function f' related to f has no size s' errorless circuit that outputs "don't know" on at most a $1 - \epsilon$ fraction of inputs. Thus the hardness is "amplified" from δ to $1 - \epsilon$. Unfortunately, this amplification comes at the cost of a loss in circuit size. This is because such results are proven by reductions which show that any size s' errorless circuit for f' that outputs "don't know" on at most a $1-\epsilon$ fraction of inputs could be used to construct a size

s errorless circuit for f that outputs “don't know” on at most a δ fraction of inputs. If the reduction makes q queries to the hypothesized errorless circuit for f', then plugging in a size s' circuit yields a circuit of size $\geq qs'$, and thus we must have $s' \leq s/q$. Hence it is desirable to keep the query complexity q to a minimum.

The first results on errorless hardness amplification were obtained by Bogdanov and Safra [44]. They achieved query complexity $O\big((\frac{1}{\delta}\log\frac{1}{\epsilon})^2 \cdot \frac{1}{\epsilon}\log\frac{1}{\delta}\big)$ when f' is the XOR of several independent copies of f. We improve the query complexity (and hence the loss in circuit size) to $O\big(\frac{1}{\epsilon}\log\frac{1}{\delta}\big)$, which is optimal up to constant factors for nonadaptive errorless hardness amplification reductions. We also give an improved result for errorless hardness amplification of problems in NP, and we prove a lower bound on the advice complexity (as in Chapter 4) of errorless hardness amplification reductions.

Part I

The Problem's Output is Random

Chapter 2

Time Hierarchies for Sampling Distributions

2.1 Introduction

The most commonly studied computational problems in theoretical computer science are *search problems*, where there is a relation specifying which outputs are acceptable, and the goal is to find any acceptable output. Another important type of computational problem is *sampling problems*, where the goal is for the output to be distributed according to (or at least statistically close to) a specified probability distribution.

Sampling problems have received much attention in the algorithms community. For example, there has been substantial work on algorithms for sampling graph colorings [90], independent sets [183, 256], matchings [147, 148], lattice tilings [180, 274], knapsack solutions [192], linear extensions of partial orders [54, 274], factored numbers [24, 152], DNF solutions [156], eulerian tours [57], stable marriages [37], words from context-free languages [111], chemical isomers [96], points on algebraic varieties [58], contingency tables [155, 64, and references within], and spanning trees [206, 273, 157, and references within]. In the complexity community, historically most research has focused on search problems (and the special case of decision problems). However, there has been a surge of interest in complexity-theoretic results that accord sampling problems a status as first-class computational problems [101, 262, 1, 173, 70, 261, 32]. Many of those works focus on proving lower bounds for explicit sampling problems on restricted models of computation.

In the context of sampling problems, we revisit the genesis of complexity theory. In their seminal paper, Hartmanis and Stearns [128] proved a *time hierarchy theorem* for decision problems, showing that there are decision problems that are solvable by deterministic algorithms running in time t but not by deterministic algorithms running in time a little less than t. This is often considered the first result in complexity theory. We study the corresponding question for sampling problems. First, observe that there is a trivial time hierarchy for exact sampling: In time t, an algorithm can produce a particular output with probability $1/2^t$, which clearly cannot be done in time less than t. The interesting question is whether a *robust* time hierarchy can be proved, showing that there exists a distribution family that can be sampled in time t, but that algorithms running in time a little less than t cannot even come close to sampling (in statistical distance). We succeed in proving such a hierarchy theorem, showing that algorithms running in a sufficiently smaller amount of time cannot sample the distribution within a statistical distance that is any constant less than 1. This is a corollary to our main theorem, which is a quantitatively tight result for distributions on constant-size domains, showing that algorithms running in a sufficiently smaller amount of time cannot sample the distribution much better than the trivial statistical distance achieved by the uniform distribution. Our results can be summarized as "a little more time gives a lot more power to sampling algorithms."

There are several proofs of time hierarchy theorems for nondeterministic algorithms and other models of computation [63, 225, 277, 88], but these proofs do not directly carry over to our setting. On the surface, our setting may seem more closely related to the long-standing open problem of proving a hierarchy for polynomial-time randomized algorithms solving decision problems. The chief difficulty in the latter setting is that an algorithm must satisfy the "promise" of having bounded error on every input, and it is not known how to guarantee this while diagonalizing against a randomized algorithm that may not have bounded error. There is a beautiful line of research that circumvents this obstacle by working in slightly-nonuniform or average-case settings [25, 87, 104, 89, 204, 116, 254, 205, 159, 146]. Our setting is intrinsically different because there is no "promise" that could be violated: Whatever algorithm we consider, it is guaranteed to sample some family of distributions. We have fundamentally different issues to address.

2.1.1 Results

We start with our definitions. We let $[k] = \{1, \ldots, k\}$ and let $\mathbb{N}$ denote the set of positive integers. For distributions D, D' on $[k]$, the statistical distance is defined as $\|D - D'\| = \max_{S \subseteq [k]} \big|\Pr_D(S) - \Pr_{D'}(S)\big|$. Our results hold for any reasonable uniform model of computation; for concreteness we may assume the model is Turing machines with access to unbiased independent coin flips.

For a function $k : \mathbb{N} \to \mathbb{N}$, we define a *$k$-family* to be a sequence $D = (D_1, D_2, D_3, \ldots)$ where D_n is a distribution on $[k(n)]$. For a function $\delta : \mathbb{N} \to [0, 1]$, we say a randomized algorithm A *δ-samples* a k-family D if when given n as input, A outputs an element $A(n) \in [k(n)]$ such that the output distribution satisfies $\|A(n) - D_n\| \leq \delta(n)$. For a function $t : \mathbb{N} \to \mathbb{N}$, we say that A *runs in time t* if for all $n \in \mathbb{N}$, A always halts in at most $t(n)$ steps when given n as input.[1] We define

$$\textsc{SampTime}_{k,\delta}(t)$$

to be the class of k-families δ-sampled by algorithms running in time t.[2]

[1]We measure the running time here as a function of the value of the input, not the bit length of the input. Alternatively, we could view the input as the string 1^n and measure the running time as a function of the bit length.

[2]If we write something such as $\textsc{SampTime}_{O(\log n), 1/2 - 1/\operatorname{poly}(n)}(\operatorname{poly}(n))$, we formally mean the union of

Theorem 2.1. *For every constant $k \geq 2$ and every constant $c \geq 1$,*

$$\text{SampTime}_{k,0}(\text{poly}(n)) \not\subseteq \text{SampTime}_{k,1-1/k-\epsilon}(t)$$

where $\epsilon(n) = 1/n^c$ and $t(n) = n^c$.

Corollary 2.2. *For every function $k(n) \geq \omega(1)$ and every constant $c \geq 1$,*

$$\text{SampTime}_{k,0}(\text{poly}(n)) \not\subseteq \text{SampTime}_{k,1-\epsilon}(t)$$

where $\epsilon = 1/c$ and $t(n) = n^c$.

Proof of Corollary 2.2. We explain how Theorem 2.1 implies Corollary 2.2 by the contrapositive. Supposing $k(n)$ and c are a counterexample to Corollary 2.2, we claim that k' and c' are a counterexample to Theorem 2.1 where $k' = c' = c+1$. A family $D \in \text{SampTime}_{k',0}(\text{poly}(n))$ can be viewed as being in $\text{SampTime}_{k,0}(\text{poly}(n))$ (since $[k'] \subseteq [k(n)]$ for all but finitely many n) and thus also in $\text{SampTime}_{k,1-1/c}(n^c)$. Now to get a $\text{SampTime}_{k',1-1/k'-1/n^{c'}}(n^{c'})$ algorithm for D, we just run the $\text{SampTime}_{k,1-1/c}(n^c)$ algorithm except that if it outputs a value $> k'$ then we output k' instead. This modification does not cause the statistical distance to go up. Thus the new algorithm runs in time $n^c + O(1)$, and for all but finitely many n it samples a distribution within statistical distance $1 - 1/c \leq 1 - 1/k' - 1/n^{c'}$ from D_n. $\square$

2.1.2 Discussion

Our definition of k-families is "unary", since there is one distribution for each n. We could alternatively define a k-family to be a function mapping bit strings of length n to distributions on $[k(n)]$ (for all n). This would more realistically model algorithmic sampling problems, but our hierarchy results are stronger with the unary definition (since a "non-unary" hierarchy follows by just ignoring all but one input of each length). Also, in average-case complexity (see Part IV), unary sampling arises naturally: The random input to an algorithm is often modeled as coming from an efficiently samplable distribution on $\{0,1\}^n$ (or $[2^n]$, in our notation) for all n. This can be viewed as a secondary motivation for our results.

For Corollary 2.2 it may seem like it would be cleaner to omit the domain size k from the complexity classes and just say, for example, that the domain is always $\{0,1\}^*$. However, this would make the corollary true for trivial reasons: A poly(n)-time samplable distribution could be supported on bit strings of length $> t(n)$, whereas a t-time samplable distribution must be supported on bit strings of length $\leq t(n)$. Corollary 2.2 is only meaningful when the domain size is at most 2^t.

Note that the $1 - 1/k - \epsilon$ statistical distance bound in Theorem 2.1 is tight since the theorem becomes false if $\epsilon = 0$. This is because the uniform distribution (which is samplable in constant time) is within statistical distance $1 - 1/k$ from every distribution on $[k]$. We mention that our proof of Theorem 2.1 generalizes straightforwardly to show that for every

$\text{SampTime}_{k,\delta}(t)$ over all functions of the form $k = O(\log n)$, $\delta = 1/2 - 1/\text{poly}(n)$, and $t = \text{poly}(n)$.

constant $k \geq 2$ and all sufficiently constructible monotone functions t and ϵ such that $2^{t(n)} \leq t(2^{\mathrm{poly}(n)})$, we have $\textsc{SampTime}_{k,0}\big(\mathrm{poly}\big(t(\mathrm{poly}(n))/\epsilon(\mathrm{poly}(n))\big)\big) \not\subseteq \textsc{SampTime}_{k,1-1/k-\epsilon}(t)$. Finally, we mention that our proofs of Theorem 2.1 and Corollary 2.2 relativize, and that they carry through without change for quantum algorithms instead of classical randomized algorithms.

We give the intuition for Theorem 2.1 in Section 2.2. We give the formal proof of Theorem 2.1 in Section 2.3. One key ingredient in the proof is a certain type of code, which we construct in Section 2.4.

2.2 Intuition for Theorem 2.1

Why standard techniques do not work. The original deterministic time hierarchy of [128] is proved by diagonalization: A separate input length n_i is reserved for each algorithm A_i running in the smaller time bound, and an algorithm running in the larger time bound is designed which, when given an input of length n_i, simulates A_i and outputs the opposite answer. In our setting, Brouwer's fixed point theorem gives a barrier to using this "direct complementation" strategy: Suppose we design an algorithm running in the larger time bound which takes n_i and simulates $A_i(n_i)$ any number of times (drawing samples from the distribution $A_i(n_i)$ as a black box) and then performs some computation and produces an output. This algorithm would implement a continuous function from distributions on $[k]$ to distributions on $[k]$,[3] where the input to the function represents the distribution $A_i(n_i)$. This function would have a fixed point, so there would be some distribution, which $A_i(n_i)$ might sample, that would cause the diagonalizing algorithm to produce exactly the same distribution. The trivial time hierarchy for exact sampling mentioned in the introduction gets around this by exploiting the fact that $A_i(n_i)$ cannot be an arbitrary distribution; it must be "discretized". However, the latter observation cannot be used to get a robust time hierarchy with a nonnegligible statistical distance gap. Another potential way to bypass the fixed point barrier would be to argue that A_i cannot sample anything close to a fixed point, but it is not clear how to make this approach work.

Since a straightforward direct complementation does not work, we take as a starting point the *delayed diagonalization* technique introduced by Žák [277]. This technique can be used to prove a time hierarchy for solving decision problems with almost any uniform model of computation that is syntactic (meaning there is no promise to be satisfied). The idea is to space out the n_i's so that n_{i+1} is exponentially larger than n_i, and use all the input lengths from n_i to $n_{i+1}-1$ to diagonalize against the i^{th} algorithm A_i. On inputs of length $n \in \{n_i, \ldots, n_{i+1}-2\}$ the diagonalizing algorithm copies the behavior of A_i on inputs of length $n+1$, and on inputs of length $n = n_{i+1}-1$ the diagonalizing algorithm "does the opposite" of A_i on inputs of length n_i (by brute force). Thus A_i cannot agree with the diagonalizing algorithm for all $n \in \{n_i, \ldots, n_{i+1}-1\}$ or we would obtain a contradiction.

The delayed diagonalization technique leads to a straightforward proof of the $k = 2$ case of Theorem 2.1, as follows. Let us use $D = (D_1, D_2, \ldots)$ to denote the k-family 0-sampled by

[3]In general we would talk about $[k(n_i)]$, but recall that k is a constant in Theorem 2.1.

A_i (the algorithm we are diagonalizing against) and $D^* = (D_1^*, D_2^*, \ldots)$ to denote the k-family 0-sampled by our diagonalizing algorithm. We can let $D_{n_{i+1}-1}^*$ be concentrated entirely on the least likely outcome of D_{n_i}, say $M \in [2]$.[4] (This is where delayed diagonalization has an advantage over direct diagonalization: On input $n_{i+1} - 1$ the diagonalizing algorithm has enough time to determine M with certainty by brute force, which breaks the "continuity barrier" that applies to methods that merely sample D_{n_i}.) Now for $n = (n_{i+1} - 2), \ldots, n_i$, by induction we may assume that $\Pr_{D_{n+1}^*}(M) \geq 1 - \epsilon(n+1)/2$ and thus $\Pr_{D_{n+1}}(M) \geq 1/2 + \epsilon(n+1)/2$ (assuming $\|D_{n+1} - D_{n+1}^*\| \leq 1/2 - \epsilon(n+1)$).[5] By sampling from D_{n+1} many times and taking the majority outcome, we can ensure that $\Pr_{D_n^*}(M) \geq 1 - \epsilon(n)/2$. In the end we have $\Pr_{D_{n_i}^*}(M) \geq 1 - \epsilon(n_i)/2$ while $\Pr_{D_{n_i}}(M) \leq 1/2$, which gives a contradiction if $\|D_{n_i} - D_{n_i}^*\| \leq 1/2 - \epsilon(n_i)$.

This simple argument breaks down when $k \geq 3$. Suppose we let $D_{n_{i+1}-1}^*$ be concentrated on $M \in [k]$, the least likely outcome of D_{n_i}. If $D_{n_{i+1}-1}$ is uniform on $\{1, \ldots, k-1\}$ then this would be consistent with any $M \in \{1, \ldots, k-1\}$, since $D_{n_{i+1}-1}$ would simultaneously have statistical distance $1 - 1/(k-1) \ll 1 - 1/k$ from the distributions concentrated on such M's. Note that it is impossible to have statistical distance $< 1 - 1/k$ from the distributions concentrated on *all* possible M's, so $D_{n_{i+1}-1}$ would be forced to reveal *some* information about the correct M, namely it must rule out at least one value.

Tree diagonalization via list-decoding. Here is the first idea we use to fix the above problem. Instead of using a single input n_i to "close the cycle" and obtain a contradiction, suppose we reserve m inputs $n_i, n_i + 1, \ldots, n_i + m - 1$ and let M_α be the least likely outcome of $D_{n_i+\alpha}$ for $\alpha \in \{0, 1, \ldots, m-1\}$. Suppose that on these inputs, our diagonalizing algorithm could somehow obtain (with high probability) a list of m candidates for the sequence $M_0, M_1, \ldots, M_{m-1}$, where at least one candidate is correct. Then we could have $D_{n_i+\alpha}^*$ put most of its probability mass on the α^{th} value from the α^{th} candidate sequence. If the α^{th} candidate sequence is the correct one, then we get $\Pr_{D_{n_i+\alpha}^*}(M_\alpha) \geq 1 - \epsilon(n_i + \alpha)/2$ while $\Pr_{D_{n_i+\alpha}}(M_\alpha) \leq 1/k$, which gives a contradiction if $\left\|D_{n_i+\alpha} - D_{n_i+\alpha}^*\right\| \leq 1 - 1/k - \epsilon(n_i + \alpha)$.

How do we get a small list of candidates? For some input n_i^* exponentially larger than n_i, suppose we encode the message $M_0, M_1, \ldots, M_{m-1}$ in some way as $\gamma \in [k]^\ell$ and use a block of ℓ inputs $n_i^*, n_i^* + 1, \ldots, n_i^* + \ell - 1$ to "declare" the codeword γ, by having $D_{n_i^*+j-1}^*$ be concentrated entirely on γ_j for $j \in [\ell]$.[6] Then we are faced with the following communication problem over a noisy channel: For some smaller inputs $n < n_i^*$, we would like to recover the original message so we can "retransmit" it to even smaller inputs (until it eventually reaches the inputs $n_i, n_i + 1, \ldots, n_i + m - 1$). Our only way to get information about the message is by sampling from the distributions $D_{n_i^*+j-1}$ (for $j \in [\ell]$), which only weakly reflect the transmitted codeword (under the assumption that A_i $(1 - 1/k - \epsilon)$-samples D^*). Thus

[4]M might seem like unusual notation here, but it is convenient in the formal proof, and it stands for "message".

[5]Note that $\epsilon(n+1)$ is not multiplication; it is the evaluation of the function ϵ on $n+1$.

[6]For notational reasons, it turns out to be more convenient for us to use 0-based indexing for the sequence of M_α's and 1-based indexing for the coordinates of the codeword γ.

the algorithm A_i being diagonalized against serves as a noisy channel for transmitting the message from larger inputs to smaller inputs.

As noted above, it is information-theoretically impossible to uniquely recover the original message when $k \geq 3$, but provided we use a suitable encoding we may be able to recover a small list of candidates. Then for each candidate in the list we could use a disjoint block of ℓ inputs to retransmit the encoding of that candidate message. More precisely, suppose there exists a small set S of messages containing the correct one, such that by sampling from $D_{n_i^*+j-1}$ (for $j \in [\ell]$) we can discover S with high probability. Then for each message in S we could have a block of ℓ inputs (that are polynomially smaller than n_i^*) "declare" the codeword corresponding to that message. Then on even smaller inputs, the diagonalizing algorithm could sample from D on the inputs in a particular block to recover a small list of candidates for the message encoded by that block. This leads to a tree structure, illustrated in Figure 2.1.[7] Each node in the tree attempts to transmit a codeword to its children, after attempting to receive a codeword from its parent by simulating A_i to get samples from D, and running some sort of list-decoder. Each node can see "which child it is" and interpret this as advice specifying which message on the list it is responsible for encoding and transmitting (the h^{th} child is responsible for the lexicographically h^{th} smallest message). The inputs $n_i, n_i + 1, \ldots, n_i + m - 1$ are the leaves of the tree. The final overall list corresponds to these leaves; input $n_i + \alpha$ would get the α^{th} message of the overall list. So α specifies a path down the tree, and there must be some path along which the original message is faithfully transmitted. Provided the tree has height logarithmic in n_i and the list at each node has constant size, the overall list would have size polynomial in n_i, and for every input the diagonalizing algorithm would only need to get polynomially many samples from D on polynomially larger inputs, and would thus run in polynomial time.

Dealing with random lists and random received words. There are complications with implementing the above idea. It is too much to hope that when a codeword is transmitted over the channel, we can recover a unique set S of candidate messages with high probability. To cut to the chase, what we will be able to guarantee is that there exists a fixed set S of $k - 1$ messages (where S depends on the distributions of D on the block we are trying to receive from) such that we can get a random set of messages T which, with high probability, contains the correct message and is contained in S. We have no further control over the distribution of T. When $k = 3$ this is not a problem: Suppose we use the advice to specify whether the correct message is lexicographically first or last in S. The child corresponding to the correct advice will get $T = S$ with some probability, and with the remaining probability T will contain only the correct message, and in either case the child knows what the correct message is. The child corresponding to the wrong advice may output garbage, but it does not matter.

The above argument does not generalize to $k \geq 4$. For example, when $k = 4$ and the correct message is the middle message in S, if we get $|T| = 2$ then we do not know whether the correct message is the first or second message in T. We now describe the key idea to

[7]The paper [254] uses a similar tree of input lengths but for a different reason.

solve this problem. For each message in S, consider the probability it is in T. By the pigeonhole principle, using a constant amount of advice we can identify a significant "gap" in these probabilities, so that every message in S has probability either above the gap or below the gap. By taking a certain number of samples of T and intersecting these sets, the probabilities go down exponentially in the number of samples, so the probabilities below the gap become vanishingly small while the probabilities above the gap remain very close to 1. Then by a union bound over the messages in S, we find that with high probability the intersection of our sampled sets T equals T^*, the set of messages in S with probabilities above the gap (which includes the correct message). As described above, since we get the unique set T^* with high probability, we can retransmit the correct codeword provided we know which message of T^* is the correct one. The branching factor of the tree becomes k^2 because the advice needs to specify which of k possible "gaps" to use (and thus how many samples of T to take) as well as the lexicographic index of the correct message within T^*.[8]

We now explain the decoding process in more detail. Suppose we are trying to receive the codeword $\gamma \in [k]^\ell$ transmitted by some block of inputs $n, \ldots, n+\ell-1$. Then for $j \in [\ell]$, $\Pr_{D^*_{n+j-1}}(\gamma_j)$ is close to 1 (assuming we are on the "good" path down the tree) and thus $\Pr_{D_{n+j-1}}(\gamma_j)$ is somewhat larger than $1/k$ (since we are assuming for contradiction that A_i $(1-1/k-\epsilon)$-samples D^*). There is some other value $\kappa_j \in [k]$ such that $\Pr_{D_{n+j-1}}(\kappa_j) < 1/k$. Hence if we repeatedly sample from D_{n+j-1} and let $\rho_j \subseteq [k]$ be the set of values that occur with frequency at least slightly greater than $1/k$ in the empirical distribution, then with high probability we get $\gamma_j \in \rho_j \subseteq [k] \backslash \kappa_j$. In general we will not get a unique ρ_j with high probability, since under D_{n+j-1} some symbols might occur with probability very close to the threshold used in defining ρ_j. We view $\rho = \rho_1 \cdots \rho_\ell$ as the received word. There is no bound on the number of "errors" here, but each error is more informative than an erasure ($\rho_j = [k]$ would correspond to an erasure). We need a construction of a list-decodable error-correcting code for this non-traditional setting (which is related to the notion of "list-recovery" from the list-decoding literature). Our list-decoder is deterministic, but since ρ is random, the list of messages T is also random. With high probability, $T \subseteq S$ where S is the list of messages for the received word $[k]\backslash\kappa_1 \ \cdots \ [k]\backslash\kappa_\ell$.

Constructing the code. By a fairly simple reduction to the traditional setting of list-decoding, one can use certain known constructions (such as [119]) to handle our non-traditional setting. We provide a direct, self-contained construction which is tailored to this setting and is much simpler than the known traditional constructions.

We now discuss our code construction. The codeword for a message is defined by interpreting the message as a bit string[9] and evaluating all possible surjections $f : \{0,1\}^{k-1} \to [k]$ on all possible sets of $k-1$ coordinates of the bit string. It can be shown that this code is list-decodable in principle (with list size $k-1$) by using the following lemma: For every set of k distinct bit strings of the same length, there exist $k-1$ coordinates on which they remain distinct. Our polynomial-time list-decoder uses this lemma in an iterative way, building and

[8]We actually only need a branching factor of $(k-1)^2$, but for simplicity we round it up to k^2 in the proof.

[9]In the formal proof we actually use m to denote the bit length of the message, rather than the length of the sequence over $[k]$.

pruning a set of candidate strings of increasing lengths until it has arrived at the correct set of messages.

2.3 Proof of Theorem 2.1

As sketched in Section 2.2, the $k = 2$ case of Theorem 2.1 is a simple application of delayed diagonalization and estimation by repeated sampling. Henceforth we assume $k \geq 3$. We start by describing a few ingredients we use in the proof.

We need a construction of a code for the following model of error-correction. Codewords are length-ℓ strings over the alphabet $[k]$, and each coordinate of a codeword can be corrupted to a subset of $[k]$ containing the correct symbol. More formally, we say a codeword $\gamma \in [k]^\ell$ is *consistent* with a received word[10] $\rho \in (\mathcal{P}([k]))^\ell$ if $\gamma_j \in \rho_j$ for all $j \in [\ell]$. A traditional erasure corresponds to the case $\rho_j = [k]$, but in our model of error-correction that is forbidden: ρ_j must be a strict subset of $[k]$, so each coordinate of the received word is more informative than an erasure. The tradeoff is that, unlike in traditional error-correction settings, we do not assume any upper bound on the number of "errors". We give an elementary construction of a list-decodable code for this setting.

Theorem 2.3. *For every constant $k \geq 3$ there exists a polynomial-time encodable code $C : \{0,1\}^m \to [k]^\ell$ where $\ell = \Theta_k(m^{k-1})$ such that the following holds. For every received word $\rho \in (\mathcal{P}([k]))^\ell$ with $\rho_j \neq [k]$ for all $j \in [\ell]$, there are at most $k-1$ messages $\mu \in \{0,1\}^m$ whose codeword $C(\mu)$ is consistent with ρ; moreover, the list of all such μ can be found in polynomial time given ρ.*

As mentioned in Section 2.2, an alternative version of Theorem 2.3 (that is adequate for our purpose) can be derived from sophisticated off-the-shelf components (such as [119]). In Section 2.4 we give a thorough discussion of the above model of error-correction, describe the alternative construction, and give our self-contained proof of Theorem 2.3.

Now let $A_1, A_2, A_3, \ldots$ be an enumeration of all randomized algorithms that run in time t and always output an element of $[k]$. We use a procedure Estimate(A_i, n, ζ, η) which returns a vector $(\pi_1, \pi_2, \ldots, \pi_k) \in [0,1]^k$ such that

(i) with probability at least $1 - \eta$, $\big|\pi_\kappa - \Pr(A_i(n) = \kappa)\big| \leq \zeta$ for all $\kappa \in [k]$, and

(ii) with probability 1, $\pi_1 + \pi_2 + \cdots + \pi_k = 1$.

In other words, it returns a distribution that probably approximates the distribution of $A_i(n)$. If $\zeta, \eta > 0$ then by a standard Chernoff bound, Estimate(A_i, n, ζ, η) can be implemented in time $O\big(t(n) \cdot \frac{1}{\zeta^2} \log \frac{1}{\eta}\big)$ by simulating $A_i(n)$ $O\big(\frac{1}{\zeta^2} \log \frac{1}{\eta}\big)$ times and taking the empirical distribution.[11] Also, Estimate$(A_i, n, 0, 0)$ can be implemented in time $O(t(n) \cdot 2^{t(n)})$.

[10]Recall that $\mathcal{P}([k])$ denotes the power set of $[k]$.

[11]We are ignoring the logarithmic factor time overhead usually associated with simulating an algorithm using a universal algorithm.

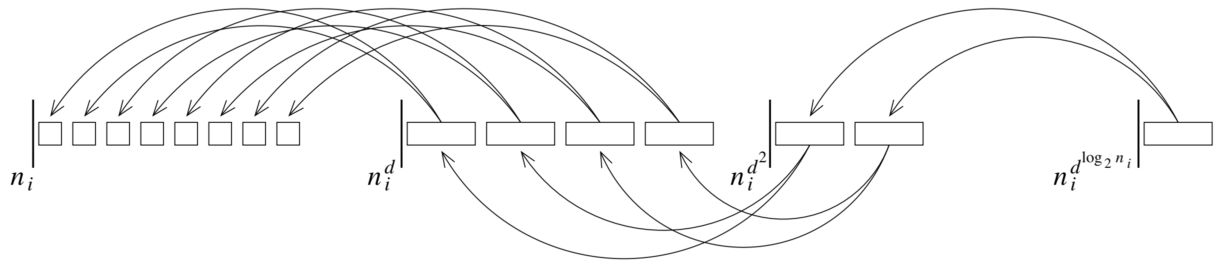

Figure 2.1: Tree of input blocks

Algorithm 1 0-samples a k-family $D^* = (D_1^*, D_2^*, \ldots)$, and we argue below that it runs in time poly(n). Thus $D^* \in \textsc{SampTime}_{k,0}(\text{poly}(n))$. We now need to prove that $D^* \notin \textsc{SampTime}_{k,1-1/k-\epsilon}(t)$. Suppose for contradiction there exists an i such that A_i $(1-1/k-\epsilon)$-samples D^*. Let $D = (D_1, D_2, \ldots)$ be the k-family that is 0-sampled by A_i. We have $\|D_n - D_n^*\| \leq 1 - 1/k - \epsilon(n)$ for all n.

The parameters used in Algorithm 1 are defined in Figure 2.2. We use the inputs from n_i through $n_{i+1} - 1$ to diagonalize against A_i. The parameters create a tree structure out of the inputs, illustrated in Figure 2.1. The tree is a full tree with branching factor k^2 and depth $\log_2 n_i$, with the leaves at level $b = 0$ and the root at level $b = \log_2 n_i$. Thus the number of leaves is $(k^2)^{\log_2 n_i}$. Each node of the tree has a contiguous block of inputs associated to it. Each leaf's block only consists of a single input, but each internal node's block has ℓ_i inputs, which represent the coordinates of codewords under the code C_i. Level b of the tree starts at input $n_{i,b} = n_i^{d^b}$. There are $(k^2)^{(\log_2 n_i)-b}$ nodes across level b, indexed by $\alpha \in \{0, 1, \ldots, (k^2)^{(\log_2 n_i)-b} - 1\}$, and their blocks of inputs $N_{i,b,\alpha}$ are consecutive from left to right across the level. Writing α in base k^2 allows us to interpret α as specifying a path down the tree from the root to the current node. The input n_1 is an unspecified constant power of 2, which just needs to be large enough so the blocks $N_{i,b,\alpha}$ are all disjoint and $\log_2 n_1 > 1$. There exists such an n_1 since $d \geq 3k \log_2 k$. Hence line 1 of Algorithm 1 will find unique values i, b, α (if they exist).

The reason we use message length $m_i = \lceil \log_2 k^{(k^2)^{\log_2 n_i}} \rceil$ is because our messages represent sequences of length $(k^2)^{\log_2 n_i}$ over the alphabet $[k]$ (one symbol for each leaf of the tree). We assume there is a canonical way of interconverting between sequences of length $(k^2)^{\log_2 n_i}$ over $[k]$ and messages in $\{0,1\}^{m_i}$. It is most convenient for us to use 0-based indexing for the sequences $M_0, M_1, \ldots, M_{(k^2)^{\log_2 n_i}-1}$ and 1-based indexing for the messages $\mu = \mu_1 \cdots \mu_{m_i}$, codewords $C(\mu) = C(\mu)_1 \cdots C(\mu)_{\ell_i}$, and received words $\rho = \rho_1 \cdots \rho_{\ell_i}$.

In general, each block of inputs $N_{i,b,\alpha}$ attempts to "receive" an encoded message via a noisy channel from its parent block and "send" the re-encoded message to its children blocks. Lines 3–24 are the *receiving phase*, and lines 25–32 are the *sending phase*. The receiving is different at the root ($b = \log_2 n_i$) because the algorithm generates the message directly without receiving it over a noisy channel. The sending is different at the leaves ($b = 0$) because instead of sending, the algorithm uses the message to attempt to deliver the coup de grâce and ensure that A_i fails to $(1 - 1/k - \epsilon)$-sample D^* if it has not already failed somewhere along the chain of "transmissions". The following claim is the heart of the

Define $d = \lceil \max(2^c, 3k \log_2 k) \rceil$

For $i \in \mathbb{N}$ define:

$$\begin{aligned}
n_i &= \begin{cases} \text{a sufficiently large constant power of 2} & \text{if } i = 1 \\ n_{i-1}^{d^{(\log_2 n_{i-1})+1}} & \text{if } i > 1 \end{cases} \\
m_i &= \left\lceil \log_2 k^{(k^2)^{\log_2 n_i}} \right\rceil \\
\ell_i &= \Theta(m_i^{k-1}), \text{ the codeword length from Theorem 2.3 for message length } m_i \\
C_i &= \{0,1\}^{m_i} \to [k]^{\ell_i}, \text{ the code from Theorem 2.3} \\
\mathrm{Dec}_i &= \text{the list-decoder from Theorem 2.3}
\end{aligned}$$

For $i \in \mathbb{N}$, $b \in \{0, 1, \ldots, \log_2 n_i\}$, $\alpha \in \{0, 1, \ldots, (k^2)^{(\log_2 n_i)-b} - 1\}$, $j \in [\ell_i]$ define:

$$\begin{aligned}
n_{i,b} &= n_i^{d^b} \\
n_{i,b,\alpha} &= \begin{cases} n_{i,b} + \alpha & \text{if } b = 0 \\ n_{i,b} + \alpha \ell_i & \text{if } b > 0 \end{cases} \\
n_{i,b,\alpha,j} &= \begin{cases} \text{undefined} & \text{if } b = 0 \\ n_{i,b,\alpha} + j - 1 & \text{if } b > 0 \end{cases} \\
N_{i,b,\alpha} &= \begin{cases} \{n_{i,b,\alpha}\} & \text{if } b = 0 \\ \{n_{i,b,\alpha,1}, \ldots, n_{i,b,\alpha,\ell_i}\} & \text{if } b > 0 \end{cases}
\end{aligned}$$

Figure 2.2: Notation for Algorithm 1

analysis. It shows that there exists a path down the tree along which the original message μ^* (generated by the root) is faithfully transmitted.

Claim 2.4. *For every $b \in \{0, 1, \ldots, \log_2 n_i\}$ there exists an $\alpha \in \{0, 1, \ldots, (k^2)^{(\log_2 n_i)-b} - 1\}$ such that for every $n \in N_{i,b,\alpha}$, with probability $\geq 1 - \epsilon(n)/2$ Algorithm 1 reaches the sending phase (lines 25–32) and the μ computed in the receiving phase (lines 3–24) equals μ^* (the message generated by the root of the tree on line 8).*

Claim 2.5. *Algorithm 1 runs in time* poly(n).

We now show how to finish the proof of Theorem 2.1 given these claims. By Claim 2.5, D^* is indeed in $\textsc{SampTime}_{k,0}(\text{poly}(n))$. Consider the good α from Claim 2.4 for $b = 0$. On input $n = n_{i,0,\alpha}$, with probability $\geq 1 - \epsilon(n)/2$ Algorithm 1 reaches the sending phase and the μ computed in the receiving phase equals μ^*. Thus the sequence $M_0, M_1, \ldots, M_{(k^2)^{\log_2 n_i} - 1}$ found on line 26 is the same as the sequence generated by the root of the tree on lines 4–7. Hence $M_\alpha = \arg\min_{\kappa \in [k]} \big(\Pr_{D_n}(\kappa)\big)$ and in particular $\Pr_{D_n}(M_\alpha) \leq 1/k$. Since $\Pr_{D_n^*}(M_\alpha) \geq 1 - \epsilon(n)/2$, this contradicts the fact that $\|D_n - D_n^*\| \leq 1 - 1/k - \epsilon(n)$ (which follows from our contradiction assumption). This finishes the proof of Theorem 2.1. All that remains is to prove Claim 2.4 and Claim 2.5.

Proof of Claim 2.4. By induction on $b = \log_2 n_i, \ldots, 0$. The base case $b = \log_2 n_i$ is trivial by the definition of μ^* (with $\alpha = 0$ and with probability 1, in fact). Now assume $b < \log_2 n_i$

Algorithm 1: Diagonalizing algorithm for Theorem 2.1

```
Input: $n \in \mathbb{N}$
Output: an element of $[k]$

find $i, b, \alpha$ such that $n \in N_{i,b,\alpha}$
if such values do not exist then halt and output an arbitrary element of $[k]$

if $b = \log_2 n_i$ then
    foreach $\alpha' \in \{0, 1, \ldots, (k^2)^{\log_2 n_i} - 1\}$ do
        let $(\pi_1^{\alpha'}, \ldots, \pi_k^{\alpha'}) = \text{Estimate}(A_i, n_{i,0,\alpha'}, 0, 0)$
        let $M_{\alpha'} = \arg\min_{\kappa \in [k]}(\pi_\kappa^{\alpha'})$ (breaking ties arbitrarily)
    end
    convert the sequence $M_0, M_1, \ldots, M_{(k^2)^{\log_2 n_i}-1}$ to a bit string $\mu \in \{0,1\}^{m_i}$
else
    write $\alpha$ in base $k^2$: $\alpha = \sum_{\tau=0}^{(\log_2 n_i)-b-1} \alpha_\tau (k^2)^\tau$ where $\alpha_\tau \in \{0, 1, \ldots, k^2 - 1\}$
    write $\alpha_0$ in base $k$: $\alpha_0 = (q-1)k + (h-1)$ where $q, h \in [k]$
    let $\alpha' = \sum_{\tau=0}^{(\log_2 n_i)-b-2} \alpha_{\tau+1} (k^2)^\tau$
    let $Q = \left(1/\epsilon(n_{i,b+1})\right)^{4q+2}$
    foreach $r \in [Q]$ do
        foreach $j' \in [\ell_i]$ do
            let $n' = n_{i,b+1,\alpha',j'}$
            let $(\pi_1^{j'}, \ldots, \pi_k^{j'}) = \text{Estimate}(A_i, n', \epsilon(n')/4, \eta)$ where $\eta = \epsilon(n_{i,b+1})/4\ell_i Q$
            let $\rho_{j'} = \{\kappa \in [k] \; : \; \pi_\kappa^{j'} \geq 1/k + \epsilon(n')/4\}$
        end
        let $T_r = \text{Dec}_i(\rho) \subseteq \{0,1\}^{m_i}$ where $\rho = \rho_1 \cdots \rho_{\ell_i} \in (\mathcal{P}([k]))^{\ell_i}$
    end
    if $|T_1 \cap \cdots \cap T_Q| < h$ then halt and output an arbitrary element of $[k]$
    let $\mu$ be the lexicographically $h^{\text{th}}$ smallest element of $T_1 \cap \cdots \cap T_Q$
end

if $b = 0$ then
    convert $\mu$ to a sequence $M_0, M_1, \ldots, M_{(k^2)^{\log_2 n_i}-1}$ over $[k]$
    halt and output $M_\alpha$
else
    compute $C_i(\mu)$
    find $j$ such that $n = n_{i,b,\alpha,j}$
    halt and output $C_i(\mu)_j$
end
```

and the claim holds for $b+1$. Let $\alpha' \in \{0, 1, \ldots, (k^2)^{(\log_2 n_i)-b-1} - 1\}$ be the good α from the induction hypothesis. For each $n' \in N_{i,b+1,\alpha'}$, say $n' = n_{i,b+1,\alpha',j'}$, the induction hypothesis says that on input n', with probability $\geq 1 - \epsilon(n')/2$ Algorithm 1 reaches the sending phase and the μ computed in the receiving phase equals μ^*. Since $b+1 > 0$, by lines 28–32 this implies that $D^*_{n'}$ puts $\geq 1 - \epsilon(n')/2$ probability mass on $C_i(\mu^*)_{j'}$. Since $\|D_{n'} - D^*_{n'}\| \leq 1 - 1/k - \epsilon(n')$, we find that $D_{n'}$ puts $\geq 1/k + \epsilon(n')/2$ probability mass on $C_i(\mu^*)_{j'}$.

We show that there exist $q, h \in [k]$ such that $\alpha = (k^2)\alpha' + \alpha_0$ satisfies the desired properties, where $\alpha_0 = (q-1)k + (h-1)$. For any such α, suppose $n \in N_{i,b,\alpha}$ and consider Algorithm 1 on input n. Note that α' computed on line 12 is indeed the α' from the induction hypothesis, and the block $N_{i,b+1,\alpha'}$ is the parent of the block $N_{i,b,\alpha}$ in the tree (see Figure 2.1).

Now consider lines 15–19. For any $j' \in [\ell_i]$, let us denote $n' = n_{i,b+1,\alpha',j'}$, and let us define $\kappa_{j'}$ to be the least likely outcome of $D_{n'}$ (breaking ties arbitrarily). Then $D_{n'}$ puts $\geq 1/k + \epsilon(n')/2$ probability mass on $C_i(\mu^*)_{j'}$ and $< 1/k$ probability mass on $\kappa_{j'}$. Hence with probability $\geq 1 - \eta$ over the estimation on line 17,

$$\pi^{j'}_{C_i(\mu^*)_{j'}} \;\geq\; \big(1/k + \epsilon(n')/2\big) - \epsilon(n')/4 \;=\; 1/k + \epsilon(n')/4$$

and $\pi^{j'}_{\kappa_{j'}} < 1/k + \epsilon(n')/4$ and thus

$$C_i(\mu^*)_{j'} \;\in\; \rho_{j'} \;\subseteq\; [k]\backslash\kappa_{j'}. \tag{2.1}$$

Note that with probability 1 we have $\rho_{j'} \neq [k]$ for all j' and thus ρ is a valid received word. For any $r \in [Q]$, let E_r be the event (depending on the randomness of lines 15–19) that Equation (2.1) holds for all j' (in the r^{th} iteration of the loop on line 14). We have

$$\Pr(E_r) \;\geq\; (1-\eta)^{\ell_i} \;\geq\; 1 - \eta\ell_i. \tag{2.2}$$

Now define

$$S \;=\; \mathrm{Dec}_i\big([k]\backslash\kappa_1 \;\cdots\; [k]\backslash\kappa_{\ell_i}\big) \;\subseteq\; \{0,1\}^{m_i}$$

and note that $|S| \leq k-1$. Conditioned on E_r, we have $\mu^* \in T_r \subseteq S$ (since $C_i(\mu^*)$ is consistent with ρ, and all codewords consistent with ρ are also consistent with $[k]\backslash\kappa_1 \;\cdots\; [k]\backslash\kappa_{\ell_i}$). Note that for different r's, T_r conditioned on E_r are independent and identically distributed. For each $\sigma \in S$ let us define p_σ to be the probability that $\sigma \in T_r$ conditioned on E_r. Note that $p_{\mu^*} = 1$ and since $|S| \leq k-1$, by the pigeonhole principle there exists a $q \in [k]$ such that for every $\sigma \in S$, either $p_\sigma \geq \exp\big(-(\epsilon^*)^{4q+4}\big)$ or $p_\sigma < \exp\big(-(\epsilon^*)^{4q}\big)$ where $\epsilon^* = \epsilon(n_{i,b+1})$. We fix this value of q and the corresponding value $Q = (1/\epsilon^*)^{4q+2}$. For each $\sigma \in S$, we have

$$\Pr\big(\sigma \in T_1 \cap \cdots \cap T_Q \;\big|\; E_1 \cap \cdots \cap E_Q\big) \;=\; (p_\sigma)^Q$$

and we have either

$$(p_\sigma)^Q \;\geq\; \exp\big(-(\epsilon^*)^2\big) \;\geq\; 1 - \epsilon(n)/4k$$

or

$$(p_\sigma)^Q \;<\; \exp\big(-(1/\epsilon^*)^2\big) \;\leq\; \epsilon(n)/4k$$

regardless of which $n \in N_{i,b,\alpha}$ we are considering.[12] Defining

$$T^* \; = \; \big\{\sigma \in S \; : \; (p_\sigma)^Q > 1/2\big\},$$

by a union bound over $\sigma \in S$ we find that

$$\Pr\big((T_1 \cap \cdots \cap T_Q) = T^* \;\big|\; E_1 \cap \cdots \cap E_Q\big) \;\geq\; 1 - \epsilon(n)/4. \tag{2.3}$$

Since $(p_{\mu^*})^Q = 1 > 1/2$, we have $\mu^* \in T^*$. Now we fix $h \in [k]$ to be such that μ^* is the lexicographically h^{th} smallest element of T^*. Then when $(T_1 \cap \cdots \cap T_Q) = T^*$, we have $\big|T_1 \cap \cdots \cap T_Q\big| \geq h$ and so Algorithm 1 reaches the sending phase, and the μ computed in the receiving phase equals μ^*, as desired. Thus for every $n \in N_{i,b,\alpha}$ where $\alpha = (k^2)\alpha' + (q-1)k + (h-1)$ we have

$$\begin{aligned}
& \Pr\big(\text{Algorithm 1 reaches the sending phase with } \mu = \mu^*\big) \\
\geq\; & \Pr\big((T_1 \cap \cdots \cap T_Q) = T^*\big) \\
\geq\; & \Pr\big(E_1 \cap \cdots \cap E_Q\big) \cdot \Pr\big((T_1 \cap \cdots \cap T_Q) = T^* \;\big|\; E_1 \cap \cdots \cap E_Q\big) \\
\geq\; & \big(1 - \eta \ell_i Q\big) \cdot \big(1 - \epsilon(n)/4\big) \\
\geq\; & 1 - \epsilon(n)/2
\end{aligned}$$

where the fourth line follows by Inequality (2.2) and Inequality (2.3), and the fifth line follows by $\eta \ell_i Q = \epsilon(n_{i,b+1})/4 \leq \epsilon(n)/4$ (where η is as on line 17 of Algorithm 1). This finishes the proof of Claim 2.4. □

Proof of Claim 2.5. Line 1 can be done in $\text{poly}(n)$ time by direct computation. If $b = \log_2 n_i$ then

$$n \;\geq\; n_i^{d^{\log_2 n_i}} \;=\; 2^{n_i^{\log_2 d} \log_2 n_i} \;\geq\; 2^{n_i^c \log_2 n_i}$$

since $d \geq 2^c$, and so the number of iterations on line 4 is $\text{polylog}(n)$ and the computation on line 5 takes time $O(t(n_i) \cdot 2^{t(n_i)}) = O(n_i^c \cdot 2^{n_i^c}) \leq \text{poly}(n)$. Suppose $b < \log_2 n_i$. Lines 10–13 are simple calculations, and we have $Q \leq \big(1/\epsilon(n^d)\big)^{4k+2} \leq \text{poly}(n)$ since $n_{i,b+1} = n_{i,b}^d \leq n^d$. We also have $m_i, \ell_i \leq \text{poly}(n_i) \leq \text{poly}(n)$ and so the loops on lines 14 and 15 have $\text{poly}(n)$ iterations. For lines 16 and 17, we have $n' \leq n_{i,b}^{d^2} \leq n^{d^2}$ and $\epsilon(n')/4 \geq 1/\text{poly}(n)$ and $\eta \geq 1/\text{poly}(n) \geq 1/\exp(\text{poly}(n))$ so the Estimate procedure takes time $\text{poly}(n)$. The list-decoding on line 20 takes time $\text{poly}(m_i) \leq \text{poly}(n)$. The sending phase (lines 25–32) trivially takes time $\text{poly}(n)$ since C_i is polynomial-time encodable. Overall, the running time is $\text{poly}(n)$. □

2.4 List-Decoding from Ubiquitous Informative Errors

In Section 2.4.1 we discuss the model of error-correction used in the proof of Theorem 2.1. Then in Section 2.4.2 we give our self-contained proof of Theorem 2.3.

[12]We can assume without loss of generality that c is large enough in terms of k for these inequalities to hold (recall that $\epsilon(n) = 1/n^c$).

2.4.1 Discussion of the Model of Error-Correction

Recall that in our model of error-correction, the received word is $\rho \in (\mathcal{P}([k]))^\ell$ where each $\rho_j \neq [k]$, and the goal is to find the list of all messages whose codeword $\gamma \in [k]^\ell$ is consistent with ρ in the sense that $\gamma_j \in \rho_j$ for all $j \in [\ell]$.

We first remark that in this setting, it can be assumed without loss of generality that $|\rho_j| = k-1$ for all $j \in [\ell]$ (since we can always enlarge each coordinate of the received word to a superset of size $k-1$, then find all the relevant messages, and then output only those messages whose codeword is consistent with the original received word). However, the way we have described the code is more convenient for our application.

Our setting is related to the notion of "list-recoverable" codes which has been studied in the list-decoding literature. In list-recovery, each coordinate of the received word is a set of symbols, but there are several differences from our setting. We allow each coordinate of the received word to be as large as possible without becoming an erasure, whereas in list-recovery each coordinate is usually restricted to be a fairly small set. Also, sometimes in list-recovery a small fraction of coordinates of the received word are allowed to violate the size restriction and become erasures. Also, in list-recovery the correct codeword is sometimes only guaranteed to agree with *many* coordinates of the received word, whereas we assume it agrees with *all* coordinates.

A simple application of the probabilistic method shows that if we drop the requirement that the encoding and list-decoding can be done in polynomial time, then there exist codes for our model with list size $k-1$ (where k is the alphabet size) and codeword length $\ell = \Theta(m)$ (where m is the message length and the hidden constant depends on k). In other words, there exist codes with $\ell = \Theta(m)$ such that for every set of k codewords, there exists a coordinate on which each element of $[k]$ appears exactly once among the k codewords. We are not aware of explicit constructions of such codes with $\ell = \Theta(m)$, but the polynomial length in Theorem 2.3 is good enough for our purpose.

For our application in Theorem 2.1, we do not need the list size to be $k-1$, as long as it is a constant depending on k. Such codes for our setting follow from certain known constructions of traditional list-decodable codes. A code is said to be (β, L)-list-decodable if for every received word in $[k]^\ell$, there are at most L codewords at relative Hamming distance $\leq \beta$, and the list of all such codewords can be found in polynomial time. Every $(1-1/(k-1), L)$-list-decodable code is also list-decodable under our model with list size $(k-1)L$: Given a received word $\rho \in (\mathcal{P}([k]))^\ell$ where each $|\rho_j| = k-1$, we can form new "received words" $\rho^{(1)}, \ldots, \rho^{(k-1)}$ by letting $\rho^{(g)} \in [k]^\ell$ consist of the g^{th} smallest symbol in each coordinate of ρ. Since a codeword consistent with ρ must have relative Hamming distance $\leq 1-1/(k-1)$ from some $\rho^{(g)}$, running the traditional list-decoder on each $\rho^{(g)}$ will reveal all the codewords consistent with ρ.

For a traditional list-decodable code construction to be used for our application via the above connection, there are several properties it should satisfy: (i) It should work for constant-size alphabets (some constructions only work for large alphabets). (ii) It should work for *every* constant-size alphabet (some constructions require the alphabet to be a finite field). (iii) The list size should be a constant depending on the alphabet size (some constructions have list size polynomial in the message length). Property (iii) is crucial, but in some cases violations of (i) and (ii) may be fixable by concatenation with a brute-force code.

The construction of Guruswami and Indyk [119], which uses expanders and spectral techniques, satisfies all these properties and is (β, L)-list-decodable with $L = O\big(1/(1 - 1/k - \beta)^3\big)$ assuming $\beta < 1 - 1/k$. Taking $\beta = 1 - 1/(k-1)$, the list size is $O(k^6)$, which becomes $O(k^7)$ after applying the reduction from our setting. The list-decoder is randomized, but that is not a problem for our application in the proof of Theorem 2.1. Thus the result of [119] yields an alternative version of Theorem 2.3 that is sufficient for our application. This alternative construction has the following advantages: The codeword length is $\Theta(m)$, and the encoding and list-decoding can be done in linear time. But it has the following disadvantages: The list size is $O(k^7)$ rather than the optimal $k-1$, the list-decoder is randomized, and the proof is much more complicated than our proof of Theorem 2.3. Although it is convenient to use this off-the-shelf machinery, our code construction demonstrates that such machinery is overkill and that elementary techniques suffice.

We now mention an interesting contrast between our setting and the traditional error-correction setting. In the traditional setting, many code constructions are linear (assuming the alphabet is a finite field). In our model of error-correction, linear codes *cannot* achieve the optimal list size of $k-1$ (where k is the alphabet size). Here is a counterexample. Recall that the property for achieving optimal list size is that for every set of k codewords, there exists a coordinate on which all k symbols appear among those codewords. Suppose the alphabet is $GF(5)$, and let x_1, x_2, x_3, x_4 be any linearly independent message vectors, and let $x_5 = 3 \times x_1 + x_2 + x_3 + x_4$. Then for any given coordinate of the codewords, if $y_1, ..., y_5 \in GF(5)$ are the symbols of the codewords in that coordinate, then they must satisfy $y_5 = 3 \times y_1 + y_2 + y_3 + y_4$ if the code is linear. It can be verified by brute force that this particular relation over $GF(5)$ forces two of the y_i's to be equal.

2.4.2 Proof of Theorem 2.3

Before giving our construction of a code C satisfying the properties in Theorem 2.3, we give a key lemma.

2.4.2.1 A Combinatorial Lemma

For a set S and number a, we let $\binom{S}{a}$ denote the set of all subsets of S of size a. For a string $\sigma \in \{0,1\}^b$ and $i \in [b]$ and $I \subseteq [b]$, we let σ_i denote the i^{th} bit of σ, and we let σ_I denote the length-$|I|$ string consisting of the bits of σ indexed by I.

Lemma 2.6. *For all $1 \le a \le b$ and every set of distinct strings $\sigma^1, \dots, \sigma^a \in \{0,1\}^b$, there exists an $I \in \binom{[b]}{a-1}$ such that $\sigma^1_I, \dots, \sigma^a_I \in \{0,1\}^{a-1}$ are distinct.*

Proof. By induction on a, with $a = 1$ and $a = 2$ being trivial. Suppose $a \ge 3$. By the induction hypothesis there exists an $I' \in \binom{[b]}{a-2}$ such that $\sigma^1_{I'}, \dots, \sigma^{a-1}_{I'}$ are distinct. If $\sigma^a_{I'}$ is different from each of $\sigma^1_{I'}, \dots, \sigma^{a-1}_{I'}$ then we can take an arbitrary $I \supseteq I'$ of size $a-1$.

Otherwise, $\sigma^a_{I'} = \sigma^h_{I'}$ for exactly one $h \in [a-1]$. Since $\sigma^a \neq \sigma^h$, there exists an $i \in [b]\backslash I'$ such that $\sigma^a_i \neq \sigma^h_i$, and we can take $I = I' \cup \{i\}$. $\square$

It is not difficult to see that the $a-1$ bound in Lemma 2.6 is tight (there do not always exist $a-2$ coordinates on which a distinct bit strings remain distinct). We remark in passing that Lemma 2.6 can be viewed in terms of a certain "dual" of VC-dimension: While the VC-dimension of a set of bit strings is the size of a *largest* set of coordinates on which every pattern appears *at least* once, we are interested in the size of a *smallest* set of coordinates on which every pattern appears *at most* once.

2.4.2.2 Code Construction

We now give our construction of the code C for an arbitrary constant $k \geq 3$ and message length $m \geq k$. By convention we use the notation $\mu \in \{0,1\}^m$ for messages, $\gamma \in [k]^\ell$ for codewords, and $\rho \in (\mathcal{P}([k]))^\ell$ for received words.

We define Surj_k to be the set of all surjections $f : \{0,1\}^{k-1} \to [k]$. The coordinates of a codeword are indexed by $\binom{[m]}{k-1} \times \mathrm{Surj}_k$, in other words by pairs I, f where I is a subset of $[m]$ of size $k-1$ and $f : \{0,1\}^{k-1} \to [k]$ is a surjection. We let $\ell = \left|\binom{[m]}{k-1} \times \mathrm{Surj}_k\right| = \Theta(m^{k-1})$, and we define the code $C : \{0,1\}^m \to [k]^\ell$ by

$$C(\mu) \;=\; \big(f(\mu_I)\big)_{I \in \binom{[m]}{k-1},\ f \in \mathrm{Surj}_k}.$$

In other words, the I, f coordinate of the codeword is the evaluation of f on the bits of the message indexed by I. Encoding can clearly be done in polynomial time.

It just remains to exhibit a polynomial-time list-decoder for C. Let us fix an arbitrary received word $\rho \in (\mathcal{P}([k]))^\ell$ with $\rho_{I,f} \neq [k]$ for all I, f. We need to show that there are at most $k-1$ messages whose codewords are consistent with ρ, and that moreover, these messages can be found in polynomial time given ρ.

For each $I \in \binom{[m]}{k-1}$ we define $\mathrm{List}(\rho, I)$ to be the set of all $\sigma \in \{0,1\}^{k-1}$ such that $f(\sigma) \in \rho_{I,f}$ for all $f \in \mathrm{Surj}_k$. Note that the set $\mathrm{List}(\rho, I)$ can be found efficiently given ρ and I by trying all possibilities.

Observation 2.7. *If $\mu \in \{0,1\}^m$ is such that $C(\mu)$ is consistent with ρ, then for all $I \in \binom{[m]}{k-1}$, $\mu_I \in \mathrm{List}(\rho, I)$.*

Lemma 2.8. *For all $I \in \binom{[m]}{k-1}$, $|\mathrm{List}(\rho, I)| \leq k-1$.*

Proof. Consider any set of k distinct strings $\sigma^1, \ldots, \sigma^k \in \{0,1\}^{k-1}$. There exists an $f \in \mathrm{Surj}_k$ such that $\{f(\sigma^1), \ldots, f(\sigma^k)\} = [k]$.[13] Therefore since $\rho_{I,f} \neq [k]$ there exists an $h \in [k]$ such that $f(\sigma^h) \notin \rho_{I,f}$, which implies that $\sigma^h \notin \mathrm{List}(\rho, I)$. $\square$

[13]Because of this, we do not actually need to use all possible surjections in the definition of the code C. We can instead use any collection of functions with the property that for every set of k distinct strings in $\{0,1\}^{k-1}$, there exists a function in the collection that assigns each of the k strings a different value.

Algorithm 2: List-decoder for Theorem 2.3

```
Input: $\rho \in (\mathcal{P}([k]))^\ell$ with $\rho_{I,f} \neq [k]$ for all $I, f$
Output: set of all $\mu \in \{0,1\}^m$ such that $C(\mu)$ is consistent with $\rho$
1 let $S_{k-1} = \mathrm{List}(\rho, [k-1])$
2 foreach $n = k, \ldots, m$ do
3     suppose $S_{n-1} = \left\{\sigma^1, \ldots, \sigma^{|S_{n-1}|}\right\} \subseteq \{0,1\}^{n-1}$
4     find an $I \in \binom{[n-1]}{k-2}$ such that $\sigma^1_I, \ldots, \sigma^{|S_{n-1}|}_I$ are distinct
5     let $S_n = \left\{ s \in \{0,1\}^n \ : \ s_{[n-1]} \in S_{n-1} \text{ and } s_{I \cup \{n\}} \in \mathrm{List}(\rho, I \cup \{n\}) \right\}$
6 end
7 output the set of all $\mu \in S_m$ such that $C(\mu)$ is consistent with $\rho$
```

Now to see that C is list-decodable in principle, suppose for contradiction that there are k distinct messages $\mu^1, \ldots, \mu^k$ whose codewords are all consistent with ρ. Applying Lemma 2.6 with $a = k$ and $b = m$, there exists an $I \in \binom{[m]}{k-1}$ such that $\mu^1_I, \ldots, \mu^k_I$ are distinct. But for all $h \in [k]$, we have $\mu^h_I \in \mathrm{List}(\rho, I)$ by Observation 2.7. Thus $\mathrm{List}(\rho, I) \geq k$, which contradicts Lemma 2.8. Hence for our arbitrary received word ρ, there are at most $k-1$ messages whose codewords are consistent with ρ. Algorithm 2 finds this list of messages in polynomial time given ρ. The correctness of the algorithm follows immediately from the following claim and line 7 of the algorithm.

Claim 2.9. *For all $n = (k-1), \ldots, m$, the following three properties hold: $S_n \subseteq \{0,1\}^n$, $|S_n| \leq k-1$, and for every $\mu \in \{0,1\}^m$ such that $C(\mu)$ is consistent with ρ we have $\mu_{[n]} \in S_n$.*

Proof. By induction on n. The base case $n = k-1$ is immediate from Lemma 2.8 and Observation 2.7, so assume $n \geq k$ and the claim holds for $n-1$. By the induction hypothesis, $|S_{n-1}| \leq k-1$ and so line 4 of the algorithm will succeed by Lemma 2.6 (with $a = |S_{n-1}|$ and $b = n-1$).

We now verify the three properties of S_n. The property $S_n \subseteq \{0,1\}^n$ is immediate. To see that $|S_n| \leq k-1$, suppose for contradiction that there are k distinct strings $s^1, \ldots, s^k \in S_n$. Then since $|\mathrm{List}(\rho, I \cup \{n\})| \leq k-1$ (by Lemma 2.8) and $s^h_{I\cup\{n\}} \in \mathrm{List}(\rho, I \cup \{n\})$ for all $h \in [k]$, there must exist $h_1 \neq h_2$ such that $s^{h_1}_{I\cup\{n\}} = s^{h_2}_{I\cup\{n\}}$. Since $s^{h_1}_{[n-1]}, s^{h_2}_{[n-1]} \in S_{n-1}$ and $s^{h_1}_I = s^{h_2}_I$, we must have $s^{h_1}_{[n-1]} = s^{h_2}_{[n-1]} = \sigma^h$ for some h. But now $s^{h_1}_{[n-1]} = s^{h_2}_{[n-1]}$ and $s^{h_1}_n = s^{h_2}_n$, which contradicts our assumption that s^{h_1} and s^{h_2} are distinct. Thus we have verified that $|S_n| \leq k-1$. To verify the third property, consider an arbitrary $\mu \in \{0,1\}^m$ such that $C(\mu)$ is consistent with ρ. By the induction hypothesis, $\mu_{[n-1]} \in S_{n-1}$, and by Observation 2.7, $\mu_{I\cup\{n\}} \in \mathrm{List}(\rho, I \cup \{n\})$. By line 5 of the algorithm, this means that $\mu_{[n]} \in S_n$. □

We now discuss the running time of the algorithm. Line 4 can be implemented in polynomial time since an efficient algorithm for finding I can be gleaned from the proof of

Lemma 2.6 (or less elegantly, since k is a constant, we can just try all possible subsets of size $k-2$). Line 5 can be implemented efficiently by looking at each string in S_{n-1} and considering extending it with each possible symbol in $[k]$ and checking whether the $I \cup \{n\}$ coordinates form a string in $\text{List}(\rho, I \cup \{n\})$. Line 7 runs in polynomial time since C is efficiently encodable and consistency is easy to check.

Since our list-decoding algorithm is correct and runs in polynomial time, this completes the proof of Theorem 2.3.

Part II

The Algorithm's Output is Random

Chapter 3

Pseudorandom Generators for Combinatorial Checkerboards

3.1 Introduction

A central question in the theory of computation is whether randomized algorithms are more powerful than deterministic algorithms. Some computational problems, such as testing whether a succinctly described polynomial is the zero polynomial, have efficient randomized algorithms but are not known to have efficient deterministic algorithms. On the other hand, a line of research in complexity theory [201, 22, 144, 238, 143, 229, 248] has shown that under widely believed conjectures (namely the existence of nonuniformly hard functions in certain uniform complexity classes), every polynomial-time randomized algorithm solving a decision problem can be *derandomized* to yield a polynomial-time deterministic algorithm solving the same decision problem. These proofs proceed by using the hypothesized hard function to construct an efficient *pseudorandom generator*, which is an algorithm that stretches a short truly random string (the *seed*) to a long "pseudorandom" string that is indistinguishable from a long truly random string by any efficient algorithm. Provided the seed is short enough, one can then cycle over all the seeds in polynomial time, running the randomized algorithm using the output of the pseudorandom generator for the randomness, to get a polynomial-time deterministic algorithm for the same decision problem.

Unfortunately, there are no known results that shed light on how to unconditionally construct pseudorandom generators that fool arbitrary polynomial-time randomized algorithms. Furthermore, there is formal evidence suggesting that unconditionally derandomizing arbitrary polynomial-time algorithms is far beyond the reach of current techniques, even if we do not insist on using a pseudorandom generator [139, 138, 3, 100].

In light of these barriers, a natural goal is to unconditionally construct pseudorandom generators with good seed lengths for restricted classes of functions. One such class of functions is those computed by *small-width read-once branching programs*, which model randomized space-bounded computations. The theory of pseudorandomness for space-bounded computations has a long and rich history [4, 23, 198, 197, 199, 200, 202, 142, 223, 16, 14, 212, 56, 218, 222, 86, 220, 187, 172, 40, 52, 53, 235, 163, 109, 66], including very general results as well as improved results for special cases. One such special case is *linear functions over* $\mathbb{Z}_2$ [195, 8, 193]. Pseudorandom generators for this class of functions are called small-bias generators. It is known how to construct small-bias generators whose seed lengths are optimal up to constant factors [195, 8]. Another special case that has been considered is *combinatorial rectangles* [83, 169, 15, 174].

We consider the problem of constructing an explicit pseudorandom generator for a new class of functions, which we dub *combinatorial checkerboards.* These functions can be viewed as

- a *special case* of small-width read-once branching programs,
- a *generalization* of linear functions over $\mathbb{Z}_2$, and
- a *variant* of combinatorial rectangles.

Other classes of functions for which constructions of good pseudorandom generators are known include juntas [195, 8], constant-depth circuits [5, 196, 182, 181, 160, 243, 259, 31, 216, 51, 67, 171], low-degree polynomials [182, 259, 39, 47, 170, 260], polynomial threshold functions [72, 188, 110, 127, 73], and read-once formulas [42].

3.1.1 Combinatorial Checkerboards

We give four equivalent ways of defining combinatorial checkerboards, which are parameterized by two positive integers m and d. Recall that $[m]$ denotes the set of integers $\{1, 2, \ldots, m\}$. For us, it is not important that the elements are integers; we only use $[m]$ as an arbitrary set of size m.

(1) A combinatorial checkerboard can be defined as a subset of $[m]^d$ of the following form. There are sets $S_1, \ldots, S_d \subseteq [m]$ such that a point $(u_1, \ldots, u_d) \in [m]^d$ is in the checkerboard if and only if the number of coordinates i such that $u_i \in S_i$ is odd. (See Figure 3.1, and note that unlike the example in the figure, the set S_i need not be a contiguous interval.) In contrast, a combinatorial rectangle can be defined similarly but where a point is in the rectangle if and only if $u_i \in S_i$ holds for all coordinates i.

(2) A combinatorial checkerboard can be defined as a function from $[m]^d$ to $\{0, 1\}$ computed by a width-2 length-d degree-m layered branching program of the following form. At layer $i \in \{1, \ldots, d\}$, the branching program reads the i^{th} symbol of the input and transitions to layer $i + 1$, and the set of symbols that cause it to cross from top to bottom is the same as the set of symbols that cause it to cross from bottom to top (call this set S_i). The start state is the bottom node in layer 1, and the accept state is the top node in layer $d + 1$. (See Figure 3.2.) In contrast, a combinatorial rectangle can be defined similarly but where the start state and the accept state are both on top, and at layer $i \in \{1, \ldots, d\}$, the bottom node transitions to the bottom node in layer $i + 1$ no matter what the i^{th} symbol is (while the behavior at the top node is arbitrary).

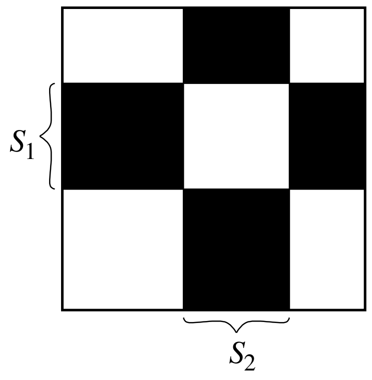

Figure 3.1

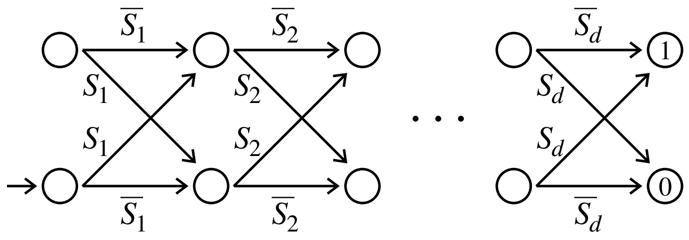

Figure 3.2

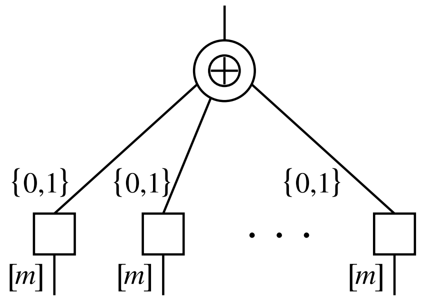

Figure 3.3

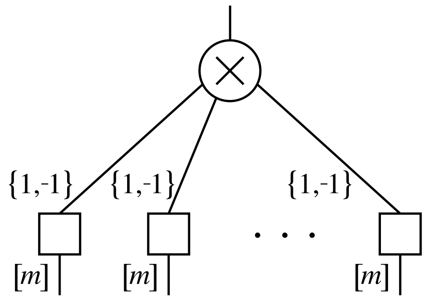

Figure 3.4

(3) A combinatorial checkerboard can be defined as a function from $[m]^d$ to $\{0,1\}$ computed by a circuit of the following form. There are d input wires, each carrying a symbol in $[m]$. Each input wire feeds into a "gate" that computes an arbitrary function from $[m]$ to $\{0,1\}$, and the resulting d bits are fed into an XOR gate. (See Figure 3.3.) In contrast, a combinatorial rectangle can be defined similarly but where the XOR gate is replaced with an AND gate.

(4) A combinatorial checkerboard can be defined as a function from $[m]^d$ to $\{1,-1\}$ computed by a circuit of the following form. There are d input wires, each carrying a symbol in $[m]$. Each input wire feeds into a "gate" that computes an arbitrary function from $[m]$ to $\{1,-1\}$, and the resulting d numbers are fed into a multiplication gate. (See Figure 3.4.) In contrast, a combinatorial rectangle can be defined similarly but where $\{1,-1\}$ is replaced with $\{0,1\}$.

For the rest of this chapter, we adopt the fourth view.

Definition 3.1 (Combinatorial Checkerboards). *We say $f : [m]^d \to \{1,-1\}$ is an (m,d)-*checkerboard *if it is of the form $f(u_1,\ldots,u_d) = \prod_{i\in[d]} f_i(u_i)$ for some functions $f_i : [m] \to \{1,-1\}$. We denote this as $f = \bigotimes_{i\in[d]} f_i$.*

Definition 3.2 (Pseudorandom Generators). *Let $\mathcal{C}$ be a class of functions from some finite universe U to $\{1, -1\}$. We say $G : \{0,1\}^s \to U$ is an ϵ-*pseudorandom generator *for $\mathcal{C}$ if for all $f \in \mathcal{C}$, $\left|\mathrm{E}_{r\in\{0,1\}^s}\left[f(G(r))\right] - \mathrm{E}_{u\in U}[f(u)]\right| \leq \epsilon$ where r and u are both chosen uniformly at random. We say s is the* seed length *of G.*

3.1.2 Our Result

The rest of this chapter is devoted to proving the following theorem.

Theorem 3.3. *There exists an explicit ϵ-pseudorandom generator for the class of (m, d)-checkerboards with seed length $O\big(\log m + \log d \cdot \log\log d + \log^{3/2} \frac{1}{\epsilon}\big)$.*

Informally, when we say *explicit* we mean that an efficient algorithm with the desired behavior is exhibited. (We do not attempt to exactly quantify the time or space efficiency parameters throughout this chapter.) In the case of Theorem 3.3, the precise meaning is that there exists a uniform deterministic algorithm $\mathcal{A}$ that takes as input the parameters m, d, ϵ and a string in $\{0,1\}^s$ (where s is the seed length), outputs an element of $[m]^d$, runs in time $\mathrm{poly}\big(\log m + d + \log \frac{1}{\epsilon}\big)$, and is such that for all m, d, ϵ the function $\mathcal{A}(m, d, \epsilon, \cdot)$ is an ϵ-pseudorandom generator for the class of (m, d)-checkerboards. A simple probabilistic argument shows that $O\big(\log m + \log d + \log \frac{1}{\epsilon}\big)$ seed length can be achieved if we allow *nonexplicit* pseudorandom generators.

Impagliazzo, Nisan, and Wigderson [142] proved a result for small-width read-once branching programs which in particular gives an explicit ϵ-pseudorandom generator for (m, d)-checkerboards with seed length $O\big(\log m + \log^2 d + \log d \cdot \log \frac{1}{\epsilon}\big)$. Our seed length is better except when $\frac{1}{\epsilon} \geq d^{\omega(\log d)}$. If m is a power of 2, then an (m, d)-checkerboard can be viewed as a polynomial over $\mathbb{Z}_2$ of degree at most $\log_2 m$ with $d \cdot \log_2 m$ variables (since each f_i can be viewed as an arbitrary function from $\mathbb{Z}_2^{\log_2 m}$ to $\mathbb{Z}_2$). Viola [260] constructed an ϵ-pseudorandom generator for n-variable, degree-k polynomials over $\mathbb{Z}_2$ with seed length $O\big(k \cdot \log n + 2^k \cdot k \cdot \log \frac{1}{\epsilon}\big)$, which yields an ϵ-pseudorandom generator for (m, d)-checkerboards with seed length $O\big(\log m \cdot \log d + m \cdot \log m \cdot \log \frac{1}{\epsilon}\big)$, assuming m is a power of 2.[1] The latter seed length is optimal when m is constant but has very poor dependence on m. If $m = 2$, then the degree of the polynomial becomes 1 (that is, a $(2, d)$-checkerboard is equivalent to a d-variable affine function over $\mathbb{Z}_2$) and the result of [260] degenerates to known constructions of small-bias generators, which have seed length $O\big(\log d + \log \frac{1}{\epsilon}\big)$.

In concurrent and independent work, Gopalan et al. [109] constructed pseudorandom generators for what they call *combinatorial shapes*, which are more general than combinatorial checkerboards. Their result immediately implies a version of Theorem 3.3 with seed length $O\big(\log m + \log d + \log^2 \frac{1}{\epsilon}\big)$, which is incomparable to our seed length. One of the components of our proof (Lemma 3.14 in Section 3.3.1 below) contributes $O\big(\log m + \log d \cdot \log\log d\big)$ to our seed length, and replacing this particular component with the result of [109] reduces the

[1] In the proof of Theorem 3.3, we show that we can assume without loss of generality that m is a power of 2. However, this is not without loss of generality when we apply the result of [260], because the reduction to the power-of-2 case blows up m to at least $4md/\epsilon$.

contribution to $O\big(\log m + \log d\big)$ (however, the construction of [109] is much more complicated than our construction for this component). In turn, this implies that Theorem 3.3 actually holds with seed length $O\big(\log m + \log d + \log^{3/2} \frac{1}{\epsilon}\big)$.

For comparison, we mention what is known for combinatorial rectangles. The two best generators (which have incomparable seed lengths) are due to Impagliazzo, Nisan, and Wigderson [142], who achieved seed length $O\big(\log m + \log^2 d + \log d \cdot \log \frac{1}{\epsilon}\big)$, and Lu [174], who achieved seed length $O\big(\log m + \log d + \log^{3/2} \frac{1}{\epsilon}\big)$. The latter result is better than the former except when $\frac{1}{\epsilon} \geq d^{\omega(\log d)}$.

3.1.3 Overview of the Proof

We partition the set of (m,d)-checkerboards into a "high-weight case" and a "low-weight case". We construct a generator that fools high-weight checkerboards and a different generator that fools low-weight checkerboards, and we combine the two generators to get a single generator that fools all checkerboards. (This technique has been used before, for example in [193, 172].) We now give our definition of the weight of a checkerboard.

Definition 3.4 (Bias and Unbias). *The* bias *of $f : U \to \{1,-1\}$ is $\beta(f) = \big|\operatorname{E}_{u\in U}[f(u)]\big|$ where u is chosen uniformly at random, and the* unbias *is $\alpha(f) = 1 - \beta(f)$.*

Definition 3.5 (Weight). *The* weight *of an (m,d)-checkerboard $f = \bigotimes_{i\in[d]} f_i$ is defined to be $\sum_{i\in[d]} \alpha(f_i)$.*

Observation 3.6. *If $f_1,\ldots,f_d,f'_1,\ldots,f'_d : [m] \to \{1,-1\}$ are such that $\bigotimes_{i\in[d]} f_i = \bigotimes_{i\in[d]} f'_i$ then for each $i \in [d]$ we have $f_i = \pm f'_i$ and thus $\alpha(f_i) = \alpha(f'_i)$. In particular, the weight of an (m,d)-checkerboard is independent of the representation as a tensor product.*

The real difficulty in proving Theorem 3.3 stems from the fact that the biases $\beta(f_i)$ are arbitrary numbers in $[0,1]$. If we knew that each bias $\beta(f_i)$ were either 0 or 1, then the techniques of [172, 163] would translate straightforwardly to our setting: The techniques of [172] would immediately yield a pseudorandom generator with seed length $O\big(\log m \cdot \log \frac{1}{\epsilon} + \log d + \log \frac{1}{\epsilon} \cdot \log\log \frac{1}{\epsilon}\big)$, and the techniques of [163] would immediately yield a pseudorandom generator with seed length $O\big(\log m + \log d \cdot \log \frac{1}{\epsilon}\big)$.

For the known results on combinatorial rectangles [83, 169, 15, 174], there is an analogous (but different) notion of "bias", and these results give techniques for handling arbitrary biases in $[0,1]$. We adapt these techniques to fool low-weight checkerboards. However, in the case of combinatorial rectangles, there basically is no "high-weight case" — to fool high-weight rectangles it suffices to fool low-weight rectangles. In our setting we are not so fortunate, and we must do something genuinely different to fool high-weight checkerboards. To accomplish the latter, we build on the techniques of Lovett, Reingold, Trevisan, and Vadhan [172].

The threshold we use to distinguish "high-weight" from "low-weight" is $\Theta\big(\log \frac{1}{\epsilon}\big)$.

Lemma 3.7 (High-Weight Case). *There exists a universal constant C such that the following holds. There exists an explicit ϵ-pseudorandom generator for the class of (m,d)-checkerboards of weight at least $C \cdot \log_2 \frac{1}{\epsilon}$ with seed length $O\big(\log m + \log d \cdot \log\log d + \log \frac{1}{\epsilon} \cdot \log\log \frac{1}{\epsilon}\big)$, provided m and d are powers of 2.*

Lemma 3.8 (Low-Weight Case). *There exists an explicit ϵ-pseudorandom generator for the class of (m,d)-checkerboards of weight less than $C \cdot \log_2 \frac{1}{\epsilon}$ with seed length $O\big(\log m + \log d + \log^{3/2} \frac{1}{\epsilon}\big)$, provided m and d are powers of 2, where C is the constant from Lemma 3.7.*

We derive Theorem 3.3 from Lemma 3.7 and Lemma 3.8 in Section 3.2. This just amounts to showing that (i) we can assume without loss of generality that m and d are powers of 2, and (ii) the two generators can be combined to fool all checkerboards. Both are simple and standard; we include the arguments for completeness.

We prove Lemma 3.7 in Section 3.3. Here is the outline of the proof. Suppose we can construct a generator with seed length $O\big(\log m + \log d \cdot \log\log d\big)$ that fools, within a constant, checkerboards of at least constant weight. Then we can use the following technique of [172] to get the final generator. First use a hash function to randomly partition the coordinates into a small number of buckets such that most buckets have at least constant weight. Then apply the hypothesized generator to each bucket, but instead of using independent seeds for the different instantiations of the hypothesized generator, sample the seeds from an appropriate pseudorandom distribution. This technique of [172] only contributes an additive $O\big(\log d + \log \frac{1}{\epsilon} \cdot \log\log \frac{1}{\epsilon}\big)$ to the seed length. Thus we just need to be able to fool, within a constant, checkerboards of at least constant weight. The heart of our proof of Lemma 3.7 is a new analysis of the generator of Impagliazzo, Nisan, and Wigderson [142] showing that for this special case, it suffices to use expander graphs of degree polylog d. In [172], the analysis of the corresponding part of the argument is considerably simpler because the authors exploit the fact that in their setting, the bias of each coordinate is either 0 or 1.

We prove Lemma 3.8 in Section 3.4. We take as a starting point the techniques of [15, 174]. Numerous small modifications to these techniques are needed. One bigger modification is the following. Lu's proof [174] critically makes use of the Bonferroni inequalities, which state that the probability of a union of events is alternately upper and lower bounded by the successive truncations of the inclusion-exclusion formula. In our proof we use an alternative analogous principle which is a bit tougher to prove than the Bonferroni inequalities, but which follows from elementary combinatorial techniques, and which may be folklore.

3.1.4 Preliminaries

Before diving into the proofs, we mention some conventions that we use for convenience throughout the proofs. We identify $\{0,1\}^s$ with $[2^s]$, and we always use the latter notation. Thus for example, a pseudorandom generator with seed length s is a function with domain $[2^s]$. We may also identify $[2^s]$ with $[2^{s_1}] \times [2^{s_2}]$ if $s = s_1 + s_2$. We also freely flatten trees of Cartesian products of sets; for example, we identify $\big((U_1 \times U_2) \times (U_3 \times U_4)\big)$ with $U_1 \times U_2 \times U_3 \times U_4$.

3.2 Deriving Theorem 3.3 from Lemma 3.7 and Lemma 3.8

In this section we prove Theorem 3.3.

Definition 3.9. *We say $\pi : [m] \times [m] \to [m]$ is a* quasigroup operation *if for every $v \in [m]$, the mappings $u \mapsto \pi(u, v)$ and $u \mapsto \pi(v, u)$ are both permutations.*

Definition 3.10. *We say a class $\mathcal{C}$ of (m, d)-checkerboards is* closed under permutations *if the following holds. For all functions $f_1, \ldots, f_d : [m] \to \{1, -1\}$ and all permutations $p_1, \ldots, p_d : [m] \to [m]$, if $\bigotimes_{i\in[d]} f_i \in \mathcal{C}$ then $\bigotimes_{i\in[d]}(f_i \circ p_i) \in \mathcal{C}$.*

Definition 3.11. *Given $\pi : [m] \times [m] \to [m]$ and $G_1 : [2^{s_1}] \to [m]^d$ and $G_2 : [2^{s_2}] \to [m]^d$, we define $\big(G_1 +_\pi G_2\big) : [2^{s_1}] \times [2^{s_2}] \to [m]^d$ by*

$$\big(G_1 +_\pi G_2\big)(r_1, r_2)_i \;=\; \pi\big(G_1(r_1)_i, G_2(r_2)_i\big)$$

for $i \in [d]$.

Proposition 3.12. *Suppose $\pi : [m] \times [m] \to [m]$ is a quasigroup operation, and suppose $\mathcal{C}_1$ and $\mathcal{C}_2$ are two classes of (m, d)-checkerboards both closed under permutations. If $G_1 : [2^{s_1}] \to [m]^d$ is an ϵ-pseudorandom generator for $\mathcal{C}_1$ and $G_2 : [2^{s_2}] \to [m]^d$ is an ϵ-pseudorandom generator for $\mathcal{C}_2$, then $G = G_1 +_\pi G_2$ is an ϵ-pseudorandom generator for $\mathcal{C}_1 \cup \mathcal{C}_2$.*

Proof. Consider an arbitrary $f \in \mathcal{C}_1 \cup \mathcal{C}_2$. Assume $f \in \mathcal{C}_1$; the other case is symmetric. To show that

$$\Big|\mathrm{E}_{r_1\in[2^{s_1}],r_2\in[2^{s_2}]}\big[(f \circ G)(r_1, r_2)\big] - \mathrm{E}_{u\in[m]^d}[f(u)]\Big| \;\le\; \epsilon$$

it suffices to show that for each $r_2 \in [2^{s_2}]$,

$$\Big|\mathrm{E}_{r_1\in[2^{s_1}]}\big[(f \circ G)(r_1, r_2)\big] - \mathrm{E}_{u\in[m]^d}[f(u)]\Big| \;\le\; \epsilon. \tag{3.1}$$

Fix an arbitrary $r_2 \in [2^{s_2}]$ and define $(v_1, \ldots, v_d) = G_2(r_2)$. Define an (m, d)-checkerboard $f' = \bigotimes_{i\in[d]} f'_i$ where $f'_i(u) = f_i\big(\pi(u, v_i)\big)$. Observe that $(f \circ G)(r_1, r_2) = (f' \circ G_1)(r_1)$ holds for each $r_1 \in [2^{s_1}]$, and thus

$$\mathrm{E}_{r_1\in[2^{s_1}]}\big[(f \circ G)(r_1, r_2)\big] \;=\; \mathrm{E}_{r_1\in[2^{s_1}]}\big[(f' \circ G_1)(r_1)\big] \tag{3.2}$$

(this holds even if π is not a quasigroup operation). Observe that $\mathrm{E}_{u\in[m]}[f_i(u)] = \mathrm{E}_{u\in[m]}[f'_i(u)]$ holds for each $i \in [d]$ since π is a quasigroup operation, and thus

$$\mathrm{E}_{u\in[m]^d}[f(u)] \;=\; \mathrm{E}_{u\in[m]^d}[f'(u)]. \tag{3.3}$$

Since $\mathcal{C}_1$ is closed under permutations and $f \in \mathcal{C}_1$, we have $f' \in \mathcal{C}_1$. Since G_1 is an ϵ-pseudorandom generator for $\mathcal{C}_1$, we have

$$\Big|\mathrm{E}_{r_1\in[2^{s_1}]}\big[(f' \circ G_1)(r_1)\big] - \mathrm{E}_{u\in[m]^d}[f'(u)]\Big| \;\le\; \epsilon. \tag{3.4}$$

Now Inequality (3.1) follows from Equality (3.2), Equality (3.3), and Inequality (3.4). $\square$

Proposition 3.13. *Suppose that for some W there exists an explicit ϵ-pseudorandom generator G_1 for the class $\mathcal{C}_1$ of (m,d)-checkerboards of weight at least W with seed length s_1, and there exists an explicit ϵ-pseudorandom generator G_2 for the class $\mathcal{C}_2$ of (m,d)-checkerboards of weight less than W with seed length s_2. Then there exists an explicit ϵ-pseudorandom generator for the class of all (m,d)-checkerboards with seed length s_1+s_2.*

Proof. Let $\pi : [m] \times [m] \to [m]$ be any explicit quasigroup operation. For example, we can identify $[m]$ with $\{0,1,\ldots,m-1\}$ and let π be addition modulo m. Observe that both $\mathcal{C}_1$ and $\mathcal{C}_2$ are closed under permutations. Then Proposition 3.12 guarantees that $G_1 +_\pi G_2$ is an explicit ϵ-pseudorandom generator for $\mathcal{C}_1 \cup \mathcal{C}_2$, which is the class of all (m,d)-checkerboards. Furthermore, $G_1 +_\pi G_2$ has seed length s_1+s_2. $\square$

Proposition 3.13 is also used in the proof of Lemma 3.8. We are now ready to prove Theorem 3.3.

Proof of Theorem 3.3. Given Lemma 3.7, Lemma 3.8, and Proposition 3.13, the only thing remaining is to handle when m or d is not a power of 2. Let m' be the smallest power of 2 that is at least $4md/\epsilon$, let d' be the smallest power of 2 that is at least d, and let $\epsilon' = \epsilon/2$. For the parameters m', d', ϵ', combining Lemma 3.7 with Lemma 3.8 using Proposition 3.13 (with $W = C \cdot \log_2 \frac{1}{\epsilon'}$ where C is the constant from Lemma 3.7) we find that there exists an explicit ϵ'-pseudorandom generator G' for the class of (m',d')-checkerboards with seed length $s = O\big(\log m' + \log d' \cdot \log\log d' + \log^{3/2} \frac{1}{\epsilon'}\big) = O\big(\log m + \log d \cdot \log\log d + \log^{3/2} \frac{1}{\epsilon}\big)$.

Now let $h : [m'] \to [m]$ be any explicit function such that every element of $[m]$ has at least $\big\lfloor \frac{m'}{m} \big\rfloor$ preimages and at most $\big\lceil \frac{m'}{m} \big\rceil$ preimages. Define $H : [m']^{d'} \to [m]^d$ by $H(u_1,\ldots,u_{d'}) = \big(h(u_1),\ldots,h(u_d)\big)$. Then we claim that the function $G = H \circ G'$, which also has seed length s, is an ϵ-pseudorandom generator for the class of (m,d)-checkerboards. Consider an arbitrary (m,d)-checkerboard $f = \bigotimes_{i\in[d]} f_i$, and define $f' = f \circ H$. Notice that $f' = \bigotimes_{i \in [d']} f'_i$ where

$$f'_i \;=\; \begin{cases} f_i \circ h & \text{if } i \in [d] \\ 1 & \text{otherwise} \end{cases}$$

where 1 denotes the constant 1 function on $[m']$. Since f' is an (m',d')-checkerboard, we have

$$\Big| \mathrm{E}_{r\in[2^s]} \big[(f' \circ G')(r)\big] - \mathrm{E}_{u \in [m']^{d'}}[f'(u)] \Big| \;\le\; \epsilon/2. \tag{3.5}$$

Since $f \circ G = f' \circ G'$, we have

$$\mathrm{E}_{r\in[2^s]} \big[(f \circ G)(r)\big] \;=\; \mathrm{E}_{r\in[2^s]} \big[(f' \circ G')(r)\big]. \tag{3.6}$$

A simple calculation shows that for each $i \in [d]$ we have

$$\Big| \mathrm{E}_{u\in[m']} \big[(f_i \circ h)(u)\big] - \mathrm{E}_{u\in[m]}[f_i(u)] \Big| \;\le\; 2m/m' \;\le\; \epsilon/2d.$$

Thus we have

$$\big| \mathrm{E}_{u\in[m']^{d'}}[f'(u)] - \mathrm{E}_{u\in[m]^d}[f(u)] \big| \;=\; \Big| \textstyle\prod_{i\in[d]} \mathrm{E}_{u\in[m']} \big[(f_i \circ h)(u)\big] - \prod_{i\in[d]} \mathrm{E}_{u\in[m]}[f_i(u)] \Big|$$

$$\begin{aligned} &\le \sum_{i\in[d]} \Big| \mathrm{E}_{u\in[m']} \big[(f_i \circ h)(u)\big] - \mathrm{E}_{u\in[m]}[f_i(u)] \Big| \\ &\le \epsilon/2 \end{aligned}$$

where the second line follows by the simple fact that for all $x_1, \ldots, x_d, y_1, \ldots, y_d \in [-1, 1]$ we have $\big|\prod_{i\in[d]} x_i - \prod_{i\in[d]} y_i\big| \le \sum_{i\in[d]} |x_i - y_i|$. Combining this with Inequality (3.5) and Equality (3.6) yields

$$\Big| \mathrm{E}_{r\in[2^s]} \big[(f \circ G)(r)\big] - \mathrm{E}_{u\in[m]^d}[f(u)] \Big| \le \epsilon. \qquad \square$$

3.3 The High-Weight Case

This section is devoted to the proof of Lemma 3.7. The main component in the proof of Lemma 3.7 is Lemma 3.14 below, and the main component in the proof of Lemma 3.14 is Lemma 3.24 below. We prove these three lemmas in Section 3.3.1, Section 3.3.2, and Section 3.3.3 respectively.

3.3.1 Proof of Lemma 3.7

We first discuss notation. We use G to denote the generator we construct to witness Lemma 3.7. The parameters m, d, ϵ are fixed, with m and d powers of 2. We can also assume without loss of generality that $\log_2 \frac{1}{\epsilon}$ is a power of 2, since otherwise we could decrease ϵ to make this so, while only affecting the seed length by a constant factor. For the rest of this section, we define $b = 16 \cdot \log_2 \frac{1}{\epsilon}$, which represents the number of "buckets" of a certain hash function. We use $i \in [d]$ to index coordinates of the original checkerboard and $j \in [b]$ to index buckets. The construction has three steps, and we use s_1, s_2, s_3 to denote the contributions of the three steps to the final seed length s.

Lemma 3.14 (Step 1). *There exists an explicit function $G_1 : [2^{s_1}] \to [m]^d$ with $s_1 = O\big(\log m + \log d \cdot \log\log d\big)$ such that if f is an (m, d)-checkerboard of weight at least 1, then $\beta(f \circ G_1) \le 3/4$.*

Lemma 3.15 (Step 2). *There exists an explicit function $G_2 : [2^{s_1}] \times [2^{s_2}] \to [2^{s_1}]^b$ with $s_2 = O\big(\log \frac{1}{\epsilon}\big)$ such that the following holds. Suppose $g = \bigotimes_{j\in[b]} g_j$ is a $(2^{s_1}, b)$-checkerboard such that*

$$\Pr_{j\in[b]} \big[\beta(g_j) > 3/4\big] \le 1/16$$

where j is chosen uniformly at random. Then $\beta(g \circ G_2) \le \epsilon/4$.

Lemma 3.16 (Step 3). *There exists a universal constant $C \ge 1$ and an explicit function $G_3 : [2^{s_3}] \times [d] \to [b]$ with $s_3 = O\big(\log d + \log \frac{1}{\epsilon} \cdot \log\log \frac{1}{\epsilon}\big)$ such that the following holds. Suppose $\alpha_1, \ldots, \alpha_d \in [0, 1]$ are such that $\sum_{i\in[d]} \alpha_i \ge C \cdot \log_2 \frac{1}{\epsilon}$. Then*

$$\Pr_{r_3\in[2^{s_3}]} \Big[\Pr_{j\in[b]} \Big[\sum_{i\,:\,G_3(r_3,i)=j} \alpha_i < 1 \Big] > 1/16 \Big] \le \epsilon/4$$

where r_3 and j are chosen uniformly at random.

We prove Lemma 3.14 in Section 3.3.2. The proof involves a new analysis of the generator of Impagliazzo, Nisan, and Wigderson [142] for the setting of combinatorial checkerboards. The heart of the analysis, which we call the Tree Labeling Lemma, is proven in Section 3.3.3.

Lovett et al. [172] implicitly proved Lemma 3.15, although they did not phrase it in terms of combinatorial checkerboards. Their proof (which we do not reproduce here) uses an instantiation of the generator of Impagliazzo, Nisan, and Wigderson [142].

In Lemma 3.16, G_3 is viewed as a family of hash functions parameterized by the first argument. Lovett et al. [172] proved Lemma 3.16 assuming each number α_i is 0 or 1, but their proof goes through for arbitrary $\alpha_i \in [0,1]$. Their proof (which we do not reproduce here) makes use of a concentration result for sums of k-wise independent random variables, due to Bellare and Rompel [33].

We now show how Lemma 3.7 follows from Lemma 3.14, Lemma 3.15, and Lemma 3.16.

Proof of Lemma 3.7. We construct a generator $G : [2^s] \to [m]^d$ with $s = s_1 + s_2 + s_3 = O\big(\log m + \log d \cdot \log\log d + \log \frac{1}{\epsilon} \cdot \log\log \frac{1}{\epsilon}\big)$ that witnesses Lemma 3.7. Identifying $[2^s]$ with $[2^{s_1}] \times [2^{s_2}] \times [2^{s_3}]$, we let

$$G(r_1, r_2, r_3)_i \;=\; G_1\Big(G_2(r_1, r_2)_{G_3(r_3,i)}\Big)_i$$

for $i \in [d]$. That is, the generator first runs $G_2(r_1, r_2)$ to obtain b seeds for G_1 and then it runs G_1 on each of these seeds. From the execution of G_1 on the j^{th} seed, the generator obtains d values in $[m]$ but it only keeps those corresponding to indices in the j^{th} bucket of the hash function $G_3(r_3, \cdot)$.

We claim that G witnesses Lemma 3.7. Consider an arbitrary (m,d)-checkerboard $f = \bigotimes_{i\in[d]} f_i$ of weight at least $C \cdot \log_2 \frac{1}{\epsilon}$ where C is the constant from Lemma 3.16. Note that

$$\beta(f) \;=\; \textstyle\prod_{i\in[d]} \beta(f_i) \;\le\; e^{-\sum_{i\in[d]} \alpha(f_i)} \;\le\; \epsilon/2.$$

We just need to argue that $\beta(f \circ G) \le \epsilon/2$, because then it follows that

$$\Big| \mathrm{E}_{r\in[2^s]}\big[(f\circ G)(r)\big] - \mathrm{E}_{u\in[m]^d}[f(u)] \Big| \;\le\; \beta(f\circ G) + \beta(f) \;\le\; \epsilon/2 + \epsilon/2 \;=\; \epsilon.$$

Define

$$\mathrm{Bad} \;=\; \Big\{ r_3 \in [2^{s_3}] \;:\; \mathrm{Pr}_{j\in[b]}\Big[\textstyle\sum_{i\,:\,G_3(r_3,i)=j} \alpha(f_i) < 1\Big] > 1/16 \Big\}$$

and let Good $= [2^{s_3}]\backslash$Bad. Applying Lemma 3.16 with $\alpha_i = \alpha(f_i)$ for each $i \in [d]$, we find that $\mathrm{Pr}_{r_3\in[2^{s_3}]}\big[r_3 \in \mathrm{Bad}\big] \le \epsilon/4$. We claim that for each $r_3 \in$ Good,

$$\Big| \mathrm{E}_{r_1\in[2^{s_1}],r_2\in[2^{s_2}]}\big[(f\circ G)(r_1,r_2,r_3)\big] \Big| \;\le\; \epsilon/4. \tag{3.7}$$

This will finish the argument since then

$$\beta(f\circ G) \;\le\; \Big| \mathrm{E}_{r_1\in[2^{s_1}],r_2\in[2^{s_2}],r_3\in[2^{s_3}]}\big[(f\circ G)(r_1,r_2,r_3) \;\big|\; r_3 \in \mathrm{Good}\big] \Big| + \mathrm{Pr}_{r_3\in[2^{s_3}]}\big[r_3 \in \mathrm{Bad}\big]$$

$$\begin{aligned}&\leq\ \epsilon/4+\epsilon/4\\&=\ \epsilon/2.\end{aligned}$$

To prove the claim, fix an arbitrary $r_3 \in \text{Good}$. For each $j \in [b]$ define an (m,d)-checkerboard $f^{(j)} = \bigotimes_{i\in[d]} f_i^{(j)}$ by

$$f_i^{(j)} \ = \ \begin{cases} f_i & \text{if } G_3(r_3,i) = j \\ 1 & \text{otherwise} \end{cases}$$

where 1 denotes the constant 1 function on $[m]$. Define a $(2^{s_1},b)$-checkerboard $g = \bigotimes_{j\in[b]} g_j$ by $g_j = f^{(j)} \circ G_1$. Note that for each $r_1 \in [2^{s_1}], r_2 \in [2^{s_2}]$, we have

$$\begin{aligned}(g\circ G_2)(r_1,r_2) \ &= \ \textstyle\prod_{j\in[b]} g_j\big(G_2(r_1,r_2)_j\big)\\ &= \ \textstyle\prod_{j\in[b]}\prod_{i\in[d]} f_i^{(j)}\Big(G_1\big(G_2(r_1,r_2)_j\big)_i\Big)\\ &= \ \textstyle\prod_{j\in[b]}\prod_{i\,:\,G_3(r_3,i)=j} f_i\Big(G_1\big(G_2(r_1,r_2)_j\big)_i\Big)\\ &= \ \textstyle\prod_{i\in[d]} f_i\bigg(G_1\Big(G_2(r_1,r_2)_{G_3(r_3,i)}\Big)_i\bigg)\\ &= \ (f\circ G)(r_1,r_2,r_3)\end{aligned}$$

by commutativity of multiplication. It follows that

$$\mathrm{E}_{r_1\in[2^{s_1}],r_2\in[2^{s_2}]}\big[(f\circ G)(r_1,r_2,r_3)\big] \ = \ \mathrm{E}_{r_1\in[2^{s_1}],r_2\in[2^{s_2}]}\big[(g\circ G_2)(r_1,r_2)\big]$$

and hence to prove Inequality (3.7) it suffices to show that $\beta(g\circ G_2) \leq \epsilon/4$. If we have $\sum_{i\,:\,G_3(r_3,i)=j} \alpha(f_i) \geq 1$ then the weight of $f^{(j)}$ is at least 1 and thus by Lemma 3.14 we have $\beta(g_j) \leq 3/4$. Since $r_3 \in \text{Good}$, we have $\Pr_{j\in[b]}\big[\beta(g_j) > 3/4\big] \leq 1/16$. Thus Lemma 3.15 implies that $\beta(g\circ G_2) \leq \epsilon/4$, as desired. □

3.3.2 Proof of Lemma 3.14

In this section we prove Lemma 3.14. The proof uses explicit constructions of expander graphs. One can view a $(2^n,2^k,\lambda)$-expander as a symmetric $2^n \times 2^n$ matrix M of nonnegative integers such that each row and each column sums to 2^k, and such that every eigenvalue of $M/2^k$, except the first, is at most λ in absolute value. An equivalent way of viewing an expander is as a regular symmetric directed multigraph on 2^n vertices with degree 2^k whose adjacency matrix is M. A third view, which we use, is a function $E : [2^{n+k}] \to [2^n] \times [2^n]$ that maps the edges (of which there are 2^{n+k}, and which are identified with the elements of $[2^{n+k}]$ in an arbitrary way) to their (head, tail) pairs.

Definition 3.17. *A* $(2^n,2^k,\lambda)$-expander *is a function* $E : [2^{n+k}] \to [2^n] \times [2^n]$ *such that the* $2^n \times 2^n$ *matrix* M *defined by* $M_{\nu_1,\nu_2} = \big|E^{-1}(\nu_1,\nu_2)\big|$ *satisfies the following:* M *is symmetric, each row and each column sums to* 2^k*, and every eigenvalue of* $M/2^k$*, except the first, is at most* λ *in absolute value.*

Many explicit constructions of good expanders are known [185, 92, 150, 9, 7, 179, 186, 191, 221]. The Gabber-Galil construction [92] in particular yields the following.

Lemma 3.18. *For every $\lambda > 0$ there exists an integer $k = O\big(\log \frac{1}{\lambda}\big)$ such that for all integers $n \geq 1$ there exists an explicit $(2^n, 2^k, \lambda)$-expander.*

We use the classic Expander Mixing Lemma, which is generally attributed to [6].

Lemma 3.19 (Expander Mixing Lemma). *For every $(2^n, 2^k, \lambda)$-expander E, every $S \subseteq [2^n]$, and every $T \subseteq [2^n]$, we have*

$$\begin{aligned} &\Big| \Pr_{\mu \in [2^{n+k}]} \big[E(\mu) \in S \times T\big] - \Pr_{\nu \in [2^n]}[\nu \in S] \cdot \Pr_{\nu \in [2^n]}[\nu \in T] \Big| \\ \leq\ & \lambda \sqrt{\Pr_{\nu \in [2^n]}[\nu \in S] \cdot \Pr_{\nu \in [2^n]}[\nu \in T]} \end{aligned}$$

where μ and ν are both chosen uniformly at random.

Definition 3.20 (Cartesian Product with Respect to E). *Given a $(2^n, 2^k, \lambda)$-expander E and two functions $h_1, h_2 : [2^n] \to U$ for some finite U, we define $h_1 \times_E h_2 : [2^{n+k}] \to U \times U$ by*

$$\big(h_1 \times_E h_2\big)(\mu) \ = \ \Big(h_1\big(E(\mu)_1\big), h_2\big(E(\mu)_2\big)\Big).$$

In other words, $h_1 \times_E h_2 = \big(h_1 \times h_2\big) \circ E$.

Definition 3.21 (Tensor Product with Respect to E). *Given a $(2^n, 2^k, \lambda)$-expander E and two functions $h_1, h_2 : [2^n] \to \{1, -1\}$, we define $h_1 \otimes_E h_2 : [2^{n+k}] \to \{1, -1\}$ by*

$$\big(h_1 \otimes_E h_2\big)(\mu) \ = \ h_1\big(E(\mu)_1\big) \cdot h_2\big(E(\mu)_2\big).$$

In other words, $h_1 \otimes_E h_2 = \big(h_1 \otimes h_2\big) \circ E$.

Observation 3.22. *For all $(2^n, 2^k, \lambda)$-expanders E and all functions $g_1, g_2 : [2^n] \to U$ and $h_1, h_2 : U \to \{1, -1\}$ for some finite U, we have*

$$\big(h_1 \otimes h_2\big) \circ \big(g_1 \times_E g_2\big) \ = \ \big(h_1 \circ g_1\big) \otimes_E \big(h_2 \circ g_2\big).$$

Proposition 3.23. *For all $(2^n, 2^k, \lambda)$-expanders E and all functions $h_1, h_2 : [2^n] \to \{1, -1\}$, we have*

$$\beta\big(h_1 \otimes_E h_2\big) \ \leq \ \beta(h_1)\beta(h_2) + \lambda \cdot \big(\alpha(h_1) + \alpha(h_2)\big).$$

Proof. We may assume without loss of generality that $\mathrm{E}_{\nu \in [2^n]}[h_1(\nu)] \geq 0$ and $\mathrm{E}_{\nu \in [2^n]}[h_2(\nu)] \geq 0$ because replacing h_1 by $-h_1$ and/or h_2 by $-h_2$ changes none of the quantities in the inequality we wish to prove. Now we have

$$\begin{aligned} &\beta\big(h_1 \otimes_E h_2\big) - \beta(h_1)\beta(h_2) \\ \leq\ & \Big| \mathrm{E}_{\mu \in [2^{n+k}]} \big[\big(h_1 \otimes_E h_2\big)(\mu)\big] - \mathrm{E}_{\nu \in [2^n]}[h_1(\nu)] \cdot \mathrm{E}_{\nu \in [2^n]}[h_2(\nu)] \Big| \end{aligned}$$

$$
\begin{aligned}
&= \Big|\mathrm{E}_{\mu\in[2^{n+k}]}\big[\big(h_1\otimes_E h_2\big)(\mu)\big] - \mathrm{E}_{\nu_1\in[2^n],\nu_2\in[2^n]}\big[\big(h_1\otimes h_2\big)(\nu_1,\nu_2)\big]\Big| \\
&= 2\cdot\Big|\Pr_{\mu\in[2^{n+k}]}\big[\big(h_1\otimes_E h_2\big)(\mu)=-1\big] - \Pr_{\nu_1\in[2^n],\nu_2\in[2^n]}\big[\big(h_1\otimes h_2\big)(\nu_1,\nu_2)=-1\big]\Big| \\
&\le 2\cdot\Big|\Pr_{\mu\in[2^{n+k}]}\big[\big(h_1\times_E h_2\big)(\mu)=(-1,1)\big] - \Pr_{\nu_1\in[2^n],\nu_2\in[2^n]}\big[\big(h_1\times h_2\big)(\nu_1,\nu_2)=(-1,1)\big]\Big| + \\
&\quad\ 2\cdot\Big|\Pr_{\mu\in[2^{n+k}]}\big[\big(h_1\times_E h_2\big)(\mu)=(1,-1)\big] - \Pr_{\nu_1\in[2^n],\nu_2\in[2^n]}\big[\big(h_1\times h_2\big)(\nu_1,\nu_2)=(1,-1)\big]\Big|
\end{aligned}
$$

by simple manipulations. Since every row of the matrix corresponding to E has the same sum, we find that

$$
\Pr_{\mu\in[2^{n+k}]}\big[\big(h_1\times_E h_2\big)(\mu)_1=-1\big] \;=\; \Pr_{\nu_1\in[2^n]}\big[h_1(\nu_1)=-1\big]
$$

and thus

$$
\begin{aligned}
&\Big|\Pr_{\mu\in[2^{n+k}]}\big[\big(h_1\times_E h_2\big)(\mu)=(-1,1)\big] - \Pr_{\nu_1\in[2^n],\nu_2\in[2^n]}\big[\big(h_1\times h_2\big)(\nu_1,\nu_2)=(-1,1)\big]\Big| \\
&= \Big|\Pr_{\mu\in[2^{n+k}]}\big[\big(h_1\times_E h_2\big)(\mu)=(-1,-1)\big] - \Pr_{\nu_1\in[2^n],\nu_2\in[2^n]}\big[\big(h_1\times h_2\big)(\nu_1,\nu_2)=(-1,-1)\big]\Big| \\
&\le \lambda\sqrt{\Pr_{\nu\in[2^n]}\big[h_1(\nu)=-1\big]\cdot\Pr_{\nu\in[2^n]}\big[h_2(\nu)=-1\big]} \\
&= \tfrac{1}{2}\lambda\sqrt{\alpha(h_1)\alpha(h_2)}
\end{aligned}
$$

by applying the Expander Mixing Lemma with $S=h_1^{-1}(-1)$ and $T=h_2^{-1}(-1)$ and using the fact that $\alpha(h_1)=2\cdot\Pr_{\nu\in[2^n]}\big[h_1(\nu)=-1\big]$ and $\alpha(h_2)=2\cdot\Pr_{\nu\in[2^n]}\big[h_2(\nu)=-1\big]$. A symmetric argument gives

$$
\begin{aligned}
&\Big|\Pr_{\mu\in[2^{n+k}]}\big[\big(h_1\times_E h_2\big)(\mu)=(1,-1)\big] - \Pr_{\nu_1\in[2^n],\nu_2\in[2^n]}\big[\big(h_1\times h_2\big)(\nu_1,\nu_2)=(1,-1)\big]\Big| \\
&\le \tfrac{1}{2}\lambda\sqrt{\alpha(h_1)\alpha(h_2)}.
\end{aligned}
$$

Putting the pieces together, we have

$$
\beta\big(h_1\otimes_E h_2\big)-\beta(h_1)\beta(h_2) \;\le\; 2\lambda\sqrt{\alpha(h_1)\alpha(h_2)} \;\le\; \lambda\cdot\big(\alpha(h_1)+\alpha(h_2)\big)
$$

by the arithmetic mean – geometric mean inequality. □

In the proof of Proposition 3.23 we showed a stronger bound than the one given in the statement, and we weakened it via the arithmetic mean – geometric mean inequality. We did this because our arguments exploit the additive structure of the weaker bound. A result similar to Proposition 3.23 was proven in [172], though the proof in that paper is direct (not going through the Expander Mixing Lemma) and achieves a slightly different bound.

We are now ready to prove Lemma 3.14.

Proof of Lemma 3.14. The generator G_1 we construct has the same form as the generator of Impagliazzo, Nisan, and Wigderson [142] but with a different setting of parameters.

For $\ell = 0, \ldots, \log_2 d$ we define a function $G_1^{(\ell)} : \left[2^{\log_2 m + k\ell}\right] \to [m]^{2^\ell}$ as follows, where $k = O(\log\log d)$ is the value corresponding to $\lambda = \frac{1}{8\log_2 d}$ according to Lemma 3.18. We let $G_1^{(0)}$ be the identity function, and for $\ell > 0$ we let $G_1^{(\ell)} = G_1^{(\ell-1)} \times_{E_\ell} G_1^{(\ell-1)}$ where E_ℓ is an explicit $\left(2^{\log_2 m + k(\ell-1)}, 2^k, \lambda\right)$-expander. Then we take $G_1 = G_1^{(\log_2 d)}$. Note that the seed length of G_1 is $s_1 = \log_2 m + k \cdot \log_2 d = O\left(\log m + \log d \cdot \log\log d\right)$ as required.

We claim that G_1 witnesses Lemma 3.14. Let $f = \bigotimes_{i\in[d]} f_i$ be an arbitrary (m,d)-checkerboard of weight at least 1. Consider a full binary tree with exactly d leaves which correspond to the coordinates $i = 1, \ldots, d$ from left to right. Let ρ denote the root. We say the leaves are at level 0, their parents are at level 1, and so on, with ρ at level $\log_2 d$. For each node v at level ℓ we define a function $f^{(v)} : [m]^{2^\ell} \to \{1, -1\}$ as follows. If v is a leaf, say the i^{th} one, then we let $f^{(v)} = f_i$. If v is an internal node with children v_1 and v_2 then we let $f^{(v)} = f^{(v_1)} \otimes f^{(v_2)}$. In other words, $f^{(v)} = \bigotimes_{i\in[2^\ell]} f_i^{(v)}$ where $f_i^{(v)} = f_{i_v+i-1}$ where i_v is the index of the leftmost leaf in v's subtree. Observe that $f = f^{(\rho)}$. For each node v at level ℓ we define $F^{(v)} = f^{(v)} \circ G_1^{(\ell)}$.

Thus our goal is to show that $\beta\left(F^{(\rho)}\right) \le 3/4$. For each node v at level $\ell \ge 1$ with children v_1 and v_2, applying Observation 3.22 with $h_1 = f^{(v_1)}$ and $h_2 = f^{(v_2)}$ and $g_1 = g_2 = G_1^{(\ell-1)}$ we find that $F^{(v)} = F^{(v_1)} \otimes_{E_\ell} F^{(v_2)}$. Now we have the following two things.

(i) For each internal node v with children v_1 and v_2, applying Proposition 3.23 with $h_1 = F^{(v_1)}$ and $h_2 = F^{(v_2)}$ we find that
$$\beta\left(F^{(v)}\right) \;\le\; \beta\left(F^{(v_1)}\right)\beta\left(F^{(v_2)}\right) + \lambda \cdot \Big(\alpha\left(F^{(v_1)}\right) + \alpha\left(F^{(v_2)}\right)\Big).$$

(ii) For each leaf v, say the i^{th} one, we have $f^{(v)} \circ G_1^{(0)} = f_i$, and hence
$$\textstyle\sum_{\text{leaves } v} \alpha\left(F^{(v)}\right) \;=\; \sum_{i\in[d]} \alpha(f_i) \;\ge\; 1.$$

Using the notation $\beta_v = \beta\left(F^{(v)}\right)$ and $\alpha_v = \alpha\left(F^{(v)}\right)$ for each node v, Lemma 3.14 now follows immediately from the following lemma, which is proved in Section 3.3.3. $\square$

Lemma 3.24 (Tree Labeling Lemma). *Suppose a full binary tree with d leaves has each node v labeled with numbers $\alpha_v, \beta_v \in [0,1]$ with $\alpha_v + \beta_v = 1$, such that*

(i) for each internal node v with children v_1 and v_2 we have $\beta_v \le \beta_{v_1}\beta_{v_2} + \lambda \cdot \left(\alpha_{v_1} + \alpha_{v_2}\right)$ where $\lambda = \frac{1}{8\log_2 d}$, and

(ii) $\sum_{\text{leaves } v} \alpha_v \ge 1$.

Then the root node ρ satisfies $\beta_\rho \le 3/4$.

3.3.3 Proof of the Tree Labeling Lemma

We now prove Lemma 3.24. We give the intuition in Section 3.3.3.1 and then the formal argument in Section 3.3.3.2.

3.3.3.1 Intuition

Very roughly, the intuition is as follows. For each node v, β_v represents an approximation to the product of β_w over all the leaves w in v's subtree. Thus for the root ρ, β_ρ represents an approximation to $\prod_{\text{leaves } v} \beta_v \le e^{-\sum_{\text{leaves } v} \alpha_v} \le 1/e$. However, each internal node v introduces an "error" of $\lambda \cdot \left(\alpha_{v_1} + \alpha_{v_2}\right)$ in addition to the errors already accumulated at the children v_1 and v_2. If these errors are small on average throughout the tree, then β_ρ will be small. On the other hand, if the errors are large on average, then this means the labels α_v are large on average and hence the labels β_v are small on average, so we expect β_ρ to be small in this case as well. However, this is just intuition for why the lemma is true, and the formal argument does not follow the dichotomy suggested by this intuition.

In the formal argument we attempt to reduce to a "worst-case scenario", by which we mean that all the inequalities in both (i) and (ii) in the statement of Lemma 3.24 hold as equalities. Provided the tree obeys a certain "monotonicity" property, we can decrease the α labels and increase the β labels to reach such a worst-case scenario. For a worst-case scenario, we can argue that the "errors" (as in the previous paragraph) must be small on average, and thus the new value of β_ρ must be small. Since we only increased all the β labels, the original value of β_ρ must also be small.

It turns out that the aforementioned monotonicity property is obeyed provided the β labels of all nodes are not too small. What if β_v is too small for some node v? Then we would like to conclude that β_ρ is small. Unfortunately, in general it might be the case that $\beta_\rho > \beta_v$, for example if v is a child of ρ and the other child of ρ has a β label very close to 1.[2] However, a calculation shows that β_ρ cannot be too much larger than β_v, so we are still safe.

3.3.3.2 Formal Argument

First, suppose there exists a node v with $\beta_v \le 1/2$. Then we can prove $\beta_\rho \le 3/4$ as follows. Let $v_0, v_1, v_2, \ldots, v_\ell$ denote the path from v to ρ, with $v = v_0$ and $\rho = v_\ell$. Then for each $i \in \{1, \ldots, \ell\}$, we have $\beta_{v_i} \le \beta_{v_{i-1}} + 2\lambda$ by condition (i) in the statement of Lemma 3.24. Since $\ell \le \log_2 d$, we conclude that

$$\beta_\rho \;\le\; \beta_v + 2\lambda \cdot \log_2 d \;=\; \beta_v + 1/4 \;\le\; 3/4.$$

Thus we are done, assuming there exists a node v with $\beta_v \le 1/2$. To prove the latter, suppose for contradiction that $\beta_v > 1/2$ holds for all v. We show that this implies $\beta_\rho \le 1/2$, which is a contradiction.

Let us assign new labels $\alpha'_v, \beta'_v \in [0,1]$ with $\alpha'_v + \beta'_v = 1$ to each node v as follows. For the leaf nodes, let α'_v equal α_v except arbitrarily decrease some of them so as to achieve $\sum_{\text{leaves } v} \alpha'_v = 1$. Then working our way up the tree, for each internal node v with children v_1 and v_2 let $\beta'_v = \beta'_{v_1}\beta'_{v_2} + \lambda \cdot \left(\alpha'_{v_1} + \alpha'_{v_2}\right)$. For each internal node v with children v_1 and v_2, define the error label $\delta'_v = \lambda \cdot \left(\alpha'_{v_1} + \alpha'_{v_2}\right)$, and for a leaf v define $\delta'_v = 0$. We say the leaves

[2]This issue would arise even if we tried to take advantage of the stronger version of Proposition 3.23 that results by not applying the arithmetic mean – geometric mean inequality.

are at level 0, their parents are at level 1, and so on, with ρ at level $\log_2 d$. We have the following three claims.

Claim 3.25. *For each node v, we have $\beta'_v \leq \prod_{\textit{leaves } w \textit{ in } v\textit{'s subtree}} \beta'_w + \sum_{\textit{nodes } w \textit{ in } v\textit{'s subtree}} \delta'_w$.*

Claim 3.26. *For each node v at level 2 or higher with children v_1 and v_2, we have $\delta'_v \leq \delta'_{v_1} + \delta'_{v_2}$.*

Claim 3.27. *For each node v, we have $\beta'_v \geq \beta_v$.*

We now stitch the three claims together to get the desired contradiction. By Claim 3.25 we have

$$\beta'_\rho \;\leq\; \prod_{\text{leaves } v} \beta'_v + \sum_{\text{nodes } v} \delta'_v \;\leq\; e^{-\sum_{\text{leaves } v} \alpha'_v} + \sum_{\text{nodes } v} \delta'_v \;=\; 1/e + \sum_{\text{nodes } v} \delta'_v.$$

Claim 3.26 implies that for each $\ell \in \{2, \ldots, \log_2 d\}$ we have

$$\sum_{\text{nodes } v \text{ at level } \ell} \delta'_v \;\leq\; \sum_{\text{nodes } v \text{ at level } \ell-1} \delta'_v.$$

Since $\sum_{\text{leaves } v} \delta'_v = 0$ and

$$\sum_{\text{nodes } v \text{ at level } 1} \delta'_v \;=\; \lambda \cdot \sum_{\text{leaves } v} \alpha'_v \;=\; \lambda$$

we find that $\sum_{\text{nodes } v} \delta'_v \leq \lambda \cdot \log_2 d = 1/8$. Using Claim 3.27 we conclude that

$$\beta_\rho \;\leq\; \beta'_\rho \;\leq\; 1/e + 1/8 \;\leq\; 1/2$$

which is the desired contradiction.

We now prove the three claims. Claim 3.27 is the only part where we need the assumption that $\beta_v > 1/2$ holds for all nodes v.

Proof of Claim 3.25. We prove this by structural induction on the tree. If v is a leaf then this holds trivially with equality. Suppose v is an internal node with children v_1 and v_2 and the claim holds for v_1 and v_2. Then we have

$$\begin{aligned}
\beta'_v &= \beta'_{v_1}\beta'_{v_2} + \delta'_v \\
&\leq \beta'_{v_1} \prod_{\text{leaves } w \text{ in } v_2\text{'s subtree}} \beta'_w + \delta'_v + \sum_{\text{nodes } w \text{ in } v_2\text{'s subtree}} \delta'_w \\
&\leq \prod_{\text{leaves } w \text{ in } v_1\text{'s subtree}} \beta'_w \cdot \prod_{\text{leaves } w \text{ in } v_2\text{'s subtree}} \beta'_w + \\
&\quad\; \delta'_v + \sum_{\text{nodes } w \text{ in } v_1\text{'s subtree}} \delta'_w + \sum_{\text{nodes } w \text{ in } v_2\text{'s subtree}} \delta'_w \\
&= \prod_{\text{leaves } w \text{ in } v\text{'s subtree}} \beta'_w + \sum_{\text{nodes } w \text{ in } v\text{'s subtree}} \delta'_w
\end{aligned}$$

where the first inequality follows by the induction hypothesis for v_2 and by $\beta'_{v_1} \leq 1$, and the second inequality follows by the induction hypothesis for v_1 and by $\prod_{\text{leaves } w \text{ in } v_2\text{'s subtree}} \beta'_w \leq 1$. □

Proof of Claim 3.26. Consider a node v at level 2 or higher with children v_1 and v_2. Let $v_{1,1}$ and $v_{1,2}$ be v_1's children, and let $v_{2,1}$ and $v_{2,2}$ be v_2's children. Note that

$$\beta'_{v_1} \;\ge\; \beta'_{v_{1,1}}\beta'_{v_{1,2}} \;\ge\; 1-\alpha'_{v_{1,1}}-\alpha'_{v_{1,2}}.$$

Thus $\alpha'_{v_1} \le \alpha'_{v_{1,1}} + \alpha'_{v_{1,2}}$ and similarly $\alpha'_{v_2} \le \alpha'_{v_{2,1}} + \alpha'_{v_{2,2}}$. It follows that

$$\begin{aligned}\delta'_v &= \lambda\cdot\big(\alpha'_{v_1}+\alpha'_{v_2}\big)\\ &\le \lambda\cdot\big(\alpha'_{v_{1,1}}+\alpha'_{v_{1,2}}+\alpha'_{v_{2,1}}+\alpha'_{v_{2,2}}\big)\\ &= \delta'_{v_1}+\delta'_{v_2}.\end{aligned}$$

□

Proof of Claim 3.27. We prove this by structural induction on the tree. For a leaf v, $\beta'_v \ge \beta_v$ holds by definition. For an internal node v with children v_1 and v_2, assume that $\beta'_{v_1} \ge \beta_{v_1}$ and $\beta'_{v_2} \ge \beta_{v_2}$. Then

$$\begin{aligned}\beta'_v-\beta_v &\ge \Big(\beta'_{v_1}\beta'_{v_2}+\lambda\cdot\big(\alpha'_{v_1}+\alpha'_{v_2}\big)\Big)-\Big(\beta_{v_1}\beta_{v_2}+\lambda\cdot\big(\alpha_{v_1}+\alpha_{v_2}\big)\Big)\\ &= \big(\beta'_{v_1}-\beta_{v_1}\big)\big(\beta_{v_2}-\lambda\big)+\big(\beta'_{v_2}-\beta_{v_2}\big)\big(\beta'_{v_1}-\lambda\big)\\ &\ge 0\end{aligned}$$

since $\beta_{v_2} > 1/2 \ge \lambda$ and $\beta'_{v_1} \ge \beta_{v_1} > 1/2 \ge \lambda$. □

This finishes the proof of Lemma 3.24.

3.4 The Low-Weight Case

This section is devoted to the proof of Lemma 3.8. The main component in the proof of Lemma 3.8 is Lemma 3.34 below, and one of the key tools in the proof of Lemma 3.34 is Lemma 3.36 below. We prove these three lemmas in Section 3.4.1, Section 3.4.2, and Section 3.4.3 respectively.

3.4.1 Proof of Lemma 3.8

We first discuss notation. The parameters m, d, ϵ are fixed, with m and d powers of 2. Let C be the constant from Lemma 3.7. Given a function $h : U_1 \times U_2 \to U_3$, we use the notation $h(u_1, \cdot)$ to represent the function from U_2 to U_3 that maps u_2 to $h(u_1, u_2)$. The construction has five steps, and we use s_1, s_2, s_3, s_4, s_5 to denote the contributions of the five steps to the final seed length s.

Lemma 3.28 (Step 1). *There exists an explicit $\epsilon/4$-pseudorandom generator $G_1 : [2^{s_1}] \times [m_1]^{d_1} \to [m]^d$ for the class of (m, d)-checkerboards of weight less than $C \cdot \log_2 \frac{1}{\epsilon}$ with $s_1 = O\big(\log d + \log \frac{1}{\epsilon}\big)$ and $m_1 = (m+d)^{O(1)}$ and $d_1 = \big(\frac{1}{\epsilon}\big)^{O(1)}$ such that for all (m, d)-checkerboards f and all $r_1 \in [2^{s_1}]$, $f \circ G_1(r_1, \cdot)$ is an (m_1, d_1)-checkerboard. Moreover, m_1 and d_1 are powers of 2.*

Lemma 3.29 (Step 2). *There exists an explicit $\epsilon/4$-pseudorandom generator $G_2 : [2^{s_2}] \times [m_2]^{d_2} \to [m_1]^{d_1}$ for the class of (m_1, d_1)-checkerboards with $s_2 = O\bigl(\log m + \log d + \log \frac{1}{\epsilon}\bigr)$ and $m_2 = \bigl(\frac{1}{\epsilon}\bigr)^{O(1)}$ and $d_2 = d_1$ such that for all (m_1, d_1)-checkerboards f and all $r_2 \in [2^{s_2}]$, $f \circ G_2(r_2, \cdot)$ is an (m_2, d_2)-checkerboard. Moreover, m_2 is a power of 2.*

Lemma 3.30 (Step 3). *There exists an explicit $\epsilon/4$-pseudorandom generator $G_3 : [2^{s_3}] \times [m_3]^{d_3} \to [m_2]^{d_2}$ for the class of (m_2, d_2)-checkerboards of weight less than $C \cdot \log_2 \frac{1}{\epsilon/2}$ with $s_3 = O\bigl(\log^{3/2} \frac{1}{\epsilon}\bigr)$ and $m_3 = 2^{O(\log^{3/2} \frac{1}{\epsilon})}$ and $d_3 = 2^{O(\log^{1/2} \frac{1}{\epsilon})}$ such that for all (m_2, d_2)-checkerboards f and all $r_3 \in [2^{s_3}]$, $f \circ G_3(r_3, \cdot)$ is an (m_3, d_3)-checkerboard. Moreover, m_3 and d_3 are powers of 2.*

Lemma 3.31 (Step 4). *There exists an explicit $\epsilon/4$-pseudorandom generator $G_4 : [2^{s_4}] \to [m_3]^{d_3}$ for the class of (m_3, d_3)-checkerboards with $s_4 = O\bigl(\log^{3/2} \frac{1}{\epsilon}\bigr)$.*

Lemma 3.32 (Step 5). *There exists an explicit $\epsilon/2$-pseudorandom generator $G_5 : [2^{s_5}] \to [m_2]^{d_2}$ for the class of (m_2, d_2)-checkerboards of weight at least $C \cdot \log_2 \frac{1}{\epsilon/2}$ with $s_5 = O\bigl(\log \frac{1}{\epsilon} \cdot \log \log \frac{1}{\epsilon}\bigr)$.*

The parameters in the first four steps are essentially the same as those used by Lu [174]. At a very high level, the motivation for these steps is as follows. Applying the generator of Impagliazzo, Nisan, and Wigderson [142] directly would give a seed length with poor dependence on the dimension d, so the plan is to first reduce the dimension and then (Step 4) apply the generator of [142]. Step 3 reduces the dimension quite nicely, balancing a tradeoff between how much the dimension is reduced and the cost in seed length to accomplish this dimension reduction. However, achieving the strong parameters of Step 3 requires that the parameter m has been reduced to $\bigl(\frac{1}{\epsilon}\bigr)^{O(1)}$. Step 2 accomplishes this, but it requires the dimension to have already been reduced to $\bigl(\frac{1}{\epsilon}\bigr)^{O(1)}$. Fortunately, the latter can be accomplished (Step 1) without any further requirements. We refer to Lu's paper [174] for more intuition about the parameters. Step 5 is needed because the dimension reduction steps only work for low-weight checkerboards, but Step 1 and Step 2 do not always preserve the low-weight property.

Lemma 3.28 and Lemma 3.30 are special cases of a more general result (Lemma 3.34 below) which is stated and proven in Section 3.4.2. The proof is an adaptation of an argument due to Lu [174] (which itself generalizes an argument due to Armoni et al. [15]).

Lemma 3.29 follows from a result of Nisan and Zuckerman [202], which uses extractors. Lu [174] used a similar lemma for combinatorial rectangles, which he obtained by plugging in an extractor due to Goldreich and Wigderson [105] and giving a somewhat simplified version of Nisan and Zuckerman's argument for his setting. Lemma 3.29 can be obtained using the same extractor with essentially the same parameters as in [174]. Although Lu's simplified argument does not directly work for combinatorial checkerboards, Nisan and Zuckerman's original argument still applies, yielding Lemma 3.29. We do not reproduce the proof of Lemma 3.29 here.

Lemma 3.31 is an instantiation of the generator of Impagliazzo, Nisan, and Wigderson [142], and Lemma 3.32 is just an instantiation of Lemma 3.7. We now prove a simple proposition showing how the above pseudorandom generators can be composed with each other (a similar proposition was used in [15, 174] though with different terminology).

Proposition 3.33. *Suppose $G'' : [2^{s''}] \times [m'']^{d''} \to [m']^{d'}$ is an ϵ'-pseudorandom generator for some class $\mathcal{C}'$ of (m', d')-checkerboards such that for all (m', d')-checkerboards f and all $r'' \in [2^{s''}]$, $f \circ G''(r'', \cdot)$ is an (m'', d'')-checkerboard. Further suppose $G''' : [2^{s'''}] \to [m'']^{d''}$ is an ϵ''-pseudorandom generator for the class of all (m'', d'')-checkerboards. Then the function $G' : [2^{s''}] \times [2^{s'''}] \to [m']^{d'}$ defined by $G'(r'', r''') = G''\big(r'', G'''(r''')\big)$ is an $(\epsilon' + \epsilon'')$-pseudorandom generator for $\mathcal{C}'$.*

Proof. Consider any $f \in \mathcal{C}'$. By the pseudorandom property of G'' we get

$$\Big| \mathrm{E}_{r'' \in [2^{s''}], u \in [m'']^{d''}} \big[(f \circ G'')(r'', u) \big] - \mathrm{E}_{u \in [m']^{d'}} [f(u)] \Big| \;\leq\; \epsilon'. \tag{3.8}$$

For each $r'' \in [2^{s''}]$, since $f \circ G''(r'', \cdot)$ is an (m'', d'')-checkerboard, by the pseudorandom property of G''' we get

$$\Big| \mathrm{E}_{r''' \in [2^{s'''}]} \Big[\Big(\big(f \circ G''(r'', \cdot) \big) \circ G''' \Big)(r''') \Big] - \mathrm{E}_{u \in [m'']^{d''}} \Big[\big(f \circ G''(r'', \cdot) \big)(u) \Big] \Big| \;\leq\; \epsilon''.$$

Noticing that

$$\Big(\big(f \circ G''(r'', \cdot) \big) \circ G''' \Big)(r''') \;=\; (f \circ G')(r'', r''')$$

and

$$\big(f \circ G''(r'', \cdot) \big)(u) \;=\; (f \circ G'')(r'', u)$$

we find that

$$\begin{aligned} & \Big| \mathrm{E}_{r'' \in [2^{s''}], r''' \in [2^{s'''}]} \big[(f \circ G')(r'', r''') \big] - \mathrm{E}_{r'' \in [2^{s''}], u \in [m'']^{d''}} \big[(f \circ G'')(r'', u) \big] \Big| \\ \leq\; & \mathrm{E}_{r'' \in [2^{s''}]} \Big[\Big| \mathrm{E}_{r''' \in [2^{s'''}]} \big[(f \circ G')(r'', r''') \big] - \mathrm{E}_{u \in [m'']^{d''}} \big[(f \circ G'')(r'', u) \big] \Big| \Big] \\ \leq\; & \epsilon''. \end{aligned}$$

Combining this with Inequality (3.8) yields

$$\Big| \mathrm{E}_{r'' \in [2^{s''}], r''' \in [2^{s'''}]} \big[(f \circ G')(r'', r''') \big] - \mathrm{E}_{u \in [m']^{d'}} [f(u)] \Big| \;\leq\; \epsilon' + \epsilon''. \qquad \square$$

Lemma 3.8 now follows straightforwardly.

Proof of Lemma 3.8. Combining G_3 with G_4 using Proposition 3.33 yields an explicit $\epsilon/2$-pseudorandom generator for the class of (m_2, d_2)-checkerboards of weight less than $C \cdot \log_2 \frac{1}{\epsilon/2}$ with seed length $s_3 + s_4$. Combining this generator with G_5 using Proposition 3.13 we obtain an explicit $\epsilon/2$-pseudorandom generator for the class of all (m_2, d_2)-checkerboards with seed length $s_3 + s_4 + s_5$. Combining this generator with G_2 using Proposition 3.33 yields an explicit $3\epsilon/4$-pseudorandom generator for the class of (m_1, d_1)-checkerboards with seed length $s_2+s_3+s_4+s_5$. Finally, combining this generator with G_1 using Proposition 3.33 we obtain an explicit ϵ-pseudorandom generator for the class of (m, d)-checkerboards of weight less than $C \cdot \log_2 \frac{1}{\epsilon}$ with seed length $s_1 + s_2 + s_3 + s_4 + s_5 = O\big(\log m + \log d + \log^{3/2} \frac{1}{\epsilon}\big)$. □

3.4.2 Dimension Reduction

In this section, m, d, ϵ are free parameters (not necessarily the same as the fixed values that were assumed throughout Section 3.4.1). Again, C is the constant from Lemma 3.7.

Lemma 3.34 (Dimension Reduction). *Let m and d be powers of 2, and let $2 \leq k \leq d$ be an integer parameter. There exists an explicit ϵ-pseudorandom generator $G : [2^s] \times [m']^{d'} \to [m]^d$ for the class of (m, d)-checkerboards of weight less than $C \cdot \log_2 \frac{1}{\epsilon}$ with $s = k \cdot \max(\log_2 d, \log_2 d')$ and $m' = \max(d, m)^k$ and $d' = 2^{O(\frac{1}{k} \cdot \log \frac{1}{\epsilon})}$ such that for all (m, d)-checkerboards f and all $r \in [2^s]$, $f \circ G(r, \cdot)$ is an (m', d')-checkerboard. Moreover, m' and d' are powers of 2.*

To obtain Lemma 3.28, just use $k = 2$ and plug in $\epsilon/4$ for ϵ. To obtain Lemma 3.30, just use $k = \Theta\big(\log^{1/2} \frac{1}{\epsilon}\big)$ and plug in m_2 for m, d_2 for d, and $\epsilon/4$ for ϵ. In both instantiations, the generator given by Lemma 3.34 actually fools a slightly larger class of checkerboards than necessary.

In the proof of Lemma 3.34 we employ the standard k-wise independent generator. It turns out that using *almost* k-wise independence does not improve the final seed length in Lemma 3.8, for the same reason it does not help in [174]. We refer to the latter paper for a discussion of this issue.

Lemma 3.35. *Let n_1, n_2, k be positive integers. There exists an explicit function $h : [2^s] \times [2^{n_1}] \to [2^{n_2}]$ with $s = k \cdot \max(n_1, n_2)$ such that for every $S \subseteq [2^{n_1}]$ with $|S| \leq k$, the random variables $h(r, \nu_1)$ for $\nu_1 \in S$ are fully independent and uniformly distributed, where $r \in [2^s]$ is chosen uniformly at random.*

See [250] for a folklore proof of Lemma 3.35, which is based on using the seed to specify a polynomial of degree $< k$ over the field with $2^{\max(n_1,n_2)}$ elements. We also need the following tool.

Lemma 3.36 (Parity Version of Bonferroni Inequalities). *Let $E_1, \ldots, E_\ell$ be events in any finite probability space. Let p be the probability that an odd number of E_i's hold. For $k = 1, \ldots, \ell$ let*

$$t_k \;=\; \textstyle\sum_{\kappa=1}^{k} (-2)^{\kappa-1} \sum_{S \subseteq [\ell] \,:\, |S|=\kappa} \Pr\big[\bigcap_{i \in S} E_i\big].$$

Then (i) $p \leq t_k$ if k is odd, (ii) $p \geq t_k$ if k is even, and (iii) $p = t_\ell$.

We prove Lemma 3.36 in Section 3.4.3 below. We are now ready to prove Lemma 3.34.

Proof of Lemma 3.34. Let $d' = 2^{\lceil \frac{2C+1}{k-1} \log_2 \frac{1}{\epsilon} \rceil}$. By convention, we use the notation $i \in [d]$, $j \in [d']$, $u \in [m]$, $w \in [m']$, and $r \in [2^s]$. Let $h_1 : [2^s] \times [d] \to [d']$ and $h_2 : [m'] \times [d] \to [m]$ be k-wise independent generators given by Lemma 3.35. For $r \in [2^s]$ and $j \in [d']$ we define $I_{r,j} = \big\{ i \in [d] \ : \ h_1(r,i) = j \big\}$.

For $i \in [d]$ we define

$$G(r, w_1, \ldots, w_{d'})_i \ = \ h_2\big(w_{h_1(r,i)}, i\big).$$

That is, we use h_1 to partition the d coordinates into d' buckets, and for each bucket we use an independent seed for h_2 to generate the symbols for the coordinates in that bucket. We claim that G witnesses Lemma 3.34. For an arbitrary (m,d)-checkerboard $f = \bigotimes_{i\in[d]} f_i$ and arbitrary $r \in [2^s]$, define the (m',d')-checkerboard $f' = \bigotimes_{j\in[d']} f'_j$ where

$$f'_j(w) \ = \ \textstyle\prod_{i\in I_{r,j}} f_i\big(h_2(w,i)\big).$$

Observe that $f'(w_1,\ldots,w_{d'}) = \prod_{i\in[d]} f_i\big(G(r,w_1,\ldots,w_{d'})_i\big)$. Thus $f' = f \circ G(r,\cdot)$ and hence $f \circ G(r,\cdot)$ is an (m',d')-checkerboard. It remains to prove that G is an ϵ-pseudorandom generator for the class of (m,d)-checkerboards of weight less than $C \cdot \log_2 \frac{1}{\epsilon}$. Fix an arbitrary (m,d)-checkerboard $f = \bigotimes_{i\in[d]} f_i$ of weight less than $C \cdot \log_2 \frac{1}{\epsilon}$.

Claim 3.37. *For every $r \in [2^s]$ and $j \in [d']$ we have*

$$\left| \mathrm{E}_{w\in[m']} \left[\textstyle\prod_{i\in I_{r,j}} f_i\big(h_2(w,i)\big) \right] - \mathrm{E}_{(u_1,\ldots,u_d)\in[m]^d} \left[\prod_{i\in I_{r,j}} f_i(u_i) \right] \right| \ \leq \ \textstyle\sum_{S\subseteq I_{r,j} \ : \ |S|=k} \prod_{i\in S} \alpha(f_i).$$

Proof of Claim 3.37. Fix arbitrary $r \in [2^s]$ and $j \in [d']$. Let $\mu_i = -\operatorname{sgn}\big(\mathrm{E}_{u\in[m]}[f_i(u)]\big)$ for $i \in [d]$ (and if $\mathrm{E}_{u\in[m]}[f_i(u)] = 0$ then let $\mu_i = \pm 1$ arbitrarily). Note that

$$\mathrm{Pr}_{u\in[m]} \big[f_i(u) = \mu_i \big] \ = \ \alpha(f_i)/2.$$

Define $b_{r,j} = (-1)^{|I_{r,j}|+1} \prod_{i\in I_{r,j}} \mu_i$. For every $(u_1,\ldots,u_d) \in [m]^d$, we have $\prod_{i\in I_{r,j}} f_i(u_i) = b_{r,j}$ if and only if the number of $i \in I_{r,j}$ such that $f_i(u_i) = \mu_i$ is odd. Relative to our fixed r and j, for any distribution $\mathcal{D}$ on $[m]^d$ and any integer $k' \geq 1$ we define

$$t^{(\mathcal{D})}_{k'} \ = \ \textstyle\sum_{\kappa=1}^{k'} (-2)^{\kappa-1} \sum_{S\subseteq I_{r,j} \ : \ |S|=\kappa} \mathrm{Pr}_{(u_1,\ldots,u_d)\sim\mathcal{D}} \big[\forall i \in S \ : \ f_i(u_i) = \mu_i \big].$$

Applying Lemma 3.36 identifying $I_{r,j}$ with $[\ell]$ and letting $E_{i'}$ be the event that $f_i(u_i) = \mu_i$ where i is the i'^{th} element of $I_{r,j}$, we find that $\mathrm{Pr}_{(u_1,\ldots,u_d)\sim\mathcal{D}} \big[\prod_{i\in I_{r,j}} f_i(u_i) = b_{r,j} \big]$ lies between $t^{(\mathcal{D})}_{k-1}$ and $t^{(\mathcal{D})}_{k}$ inclusive. (This follows from part (i) and part (ii) when $k \leq |I_{r,j}|$ and from part (iii) when $k > |I_{r,j}|$.) Now let $\mathcal{U}$ denote the uniform distribution over $[m]^d$, and let $\mathcal{D}$ be the distribution $\big(h_2(w,1),\ldots,h_2(w,d)\big)$ where $w \in [m']$ is chosen uniformly at random. By the k-wise independence of h_2, we have $t^{(\mathcal{D})}_{k} = t^{(\mathcal{U})}_{k}$ and $t^{(\mathcal{D})}_{k-1} = t^{(\mathcal{U})}_{k-1}$. Now since $\mathrm{Pr}_{w\in[m']} \big[\prod_{i\in I_{r,j}} f_i\big(h_2(w,i)\big) = b_{r,j} \big]$ and $\mathrm{Pr}_{(u_1,\ldots,u_d)\in[m]^d} \big[\prod_{i\in I_{r,j}} f_i(u_i) = b_{r,j} \big]$ both lie between $t^{(\mathcal{U})}_{k-1}$ and $t^{(\mathcal{U})}_{k}$ inclusive, we have

$$\left| \mathrm{Pr}_{w\in[m']} \left[\textstyle\prod_{i\in I_{r,j}} f_i\big(h_2(w,i)\big) = b_{r,j} \right] - \mathrm{Pr}_{(u_1,\ldots,u_d)\in[m]^d} \left[\prod_{i\in I_{r,j}} f_i(u_i) = b_{r,j} \right] \right|$$

$$\begin{aligned}
&\le \ \left|t_k^{(\mathcal{U})} - t_{k-1}^{(\mathcal{U})}\right| \\
&= \ 2^{k-1} \textstyle\sum_{S \subseteq I_{r,j} \ : \ |S|=k} \Pr_{(u_1,\ldots,u_d)\in[m]^d}\left[\forall i \in S \ : \ f_i(u_i) = \mu_i\right] \\
&= \ 2^{k-1} \textstyle\sum_{S \subseteq I_{r,j} \ : \ |S|=k} \prod_{i\in S} \alpha(f_i)/2 \\
&= \ \tfrac{1}{2} \textstyle\sum_{S \subseteq I_{r,j} \ : \ |S|=k} \prod_{i\in S} \alpha(f_i).
\end{aligned}$$

It follows that

$$\begin{aligned}
&\left| \mathrm{E}_{w\in[m']}\left[\textstyle\prod_{i\in I_{r,j}} f_i\big(h_2(w,i)\big)\right] - \mathrm{E}_{(u_1,\ldots,u_d)\in[m]^d}\left[\textstyle\prod_{i\in I_{r,j}} f_i(u_i)\right]\right| \\
&= \ 2\cdot\left| \Pr_{w\in[m']}\left[\textstyle\prod_{i\in I_{r,j}} f_i\big(h_2(w,i)\big) = b_{r,j}\right] - \Pr_{(u_1,\ldots,u_d)\in[m]^d}\left[\textstyle\prod_{i\in I_{r,j}} f_i(u_i) = b_{r,j}\right]\right| \\
&\le \ \textstyle\sum_{S \subseteq I_{r,j} \ : \ |S|=k} \prod_{i\in S} \alpha(f_i).
\end{aligned}$$

This finishes the proof of the claim. □

We now continue with the proof of Lemma 3.34. We have

$$\begin{aligned}
&\left| \mathrm{E}_{r\in[2^s],(w_1,\ldots,w_{d'})\in[m']^{d'}}\left[(f\circ G)(r,w_1,\ldots,w_{d'})\right] - \mathrm{E}_{(u_1,\ldots,u_d)\in[m]^d}\left[f(u_1,\ldots,u_d)\right]\right| \\
\le \ & \mathrm{E}_{r\in[2^s]}\left[\left| \mathrm{E}_{(w_1,\ldots,w_{d'})\in[m']^{d'}}\left[(f\circ G)(r,w_1,\ldots,w_{d'})\right] - \mathrm{E}_{(u_1,\ldots,u_d)\in[m]^d}\left[f(u_1,\ldots,u_d)\right]\right|\right] \\
= \ & \mathrm{E}_{r\in[2^s]}\left[\left| \textstyle\prod_{j\in[d']} \mathrm{E}_{w\in[m']}\left[\prod_{i\in I_{r,j}} f_i\big(h_2(w,i)\big)\right] - \prod_{j\in[d']} \mathrm{E}_{(u_1,\ldots,u_d)\in[m]^d}\left[\prod_{i\in I_{r,j}} f_i(u_i)\right]\right|\right] \\
\le \ & \mathrm{E}_{r\in[2^s]}\left[\textstyle\sum_{j\in[d']}\left| \mathrm{E}_{w\in[m']}\left[\prod_{i\in I_{r,j}} f_i\big(h_2(w,i)\big)\right] - \mathrm{E}_{(u_1,\ldots,u_d)\in[m]^d}\left[\prod_{i\in I_{r,j}} f_i(u_i)\right]\right|\right] \\
\le \ & \mathrm{E}_{r\in[2^s]}\left[\textstyle\sum_{j\in[d']}\sum_{S\subseteq I_{r,j} \ : \ |S|=k} \prod_{i\in S}\alpha(f_i)\right] \\
= \ & \textstyle\sum_{j\in[d']}\sum_{S\subseteq[d] \ : \ |S|=k} \Pr_{r\in[2^s]}[S\subseteq I_{r,j}]\cdot\prod_{i\in S}\alpha(f_i) \\
= \ & \textstyle\sum_{j\in[d']}\sum_{S\subseteq[d] \ : \ |S|=k} \frac{1}{(d')^k}\cdot\prod_{i\in S}\alpha(f_i) \\
= \ & \textstyle\frac{1}{(d')^{k-1}}\sum_{S\subseteq[d] \ : \ |S|=k} \prod_{i\in S}\alpha(f_i) \\
\le \ & \textstyle\frac{1}{(d')^{k-1}}\cdot\frac{1}{k!}\sum_{(i_1,\ldots,i_k)\in[d]^k}\prod_{\kappa\in[k]}\alpha(f_{i_\kappa}) \\
= \ & \textstyle\frac{1}{(d')^{k-1}}\cdot\frac{1}{k!}\Big(\sum_{i\in[d]}\alpha(f_i)\Big)^k \\
< \ & \textstyle\frac{1}{(d')^{k-1}}\cdot\frac{1}{k!}\big(C\cdot\log_2\frac{1}{\epsilon}\big)^k
\end{aligned}$$

where the fourth line follows by the simple fact that for all $x_1,\ldots,x_{d'},y_1,\ldots,y_{d'}\in[-1,1]$ we have $\left|\prod_{j\in[d']}x_j - \prod_{j\in[d']}y_j\right| \le \sum_{j\in[d']}|x_j-y_j|$, the fifth line follows by Claim 3.37, the seventh line follows by the k-wise independence of h_1, and the other lines follow by simple manipulations. All that remains is to show that $\frac{1}{(d')^{k-1}}\cdot\frac{1}{k!}\big(C\cdot\log_2\frac{1}{\epsilon}\big)^k \le \epsilon$. We have

$(d')^{k-1} \geq \left(\frac{1}{\epsilon}\right)^{2C+1}$. Using the standard bound $k! \geq \left(\frac{k}{e}\right)^k$ we have

$$\tfrac{1}{k!}\big(C \cdot \log_2 \tfrac{1}{\epsilon}\big)^k \;\leq\; \Big(\tfrac{e \cdot C \cdot \log_2 \frac{1}{\epsilon}}{k}\Big)^k \;\leq\; e^{C \cdot \log_2 \frac{1}{\epsilon}} \;\leq\; \big(\tfrac{1}{\epsilon}\big)^{2C}$$

where the middle inequality follows by a little calculus (and holds no matter what k is). We conclude that $\frac{1}{(d')^{k-1}} \cdot \frac{1}{k!}\big(C \cdot \log_2 \frac{1}{\epsilon}\big)^k \leq \big(\frac{1}{\epsilon}\big)^{2C} / \big(\frac{1}{\epsilon}\big)^{2C+1} = \epsilon$. This finishes the proof of Lemma 3.34. $\square$

3.4.3 Proof of Lemma 3.36

Let Ω denote the sample space. Let $P : \Omega \to \{0,1\}$ be the indicator for the event that an odd number of E_i's hold, and for $S \subseteq [\ell]$ let $\chi_S : \Omega \to \{0,1\}$ be the indicator for the event $\bigcap_{i \in S} E_i$. For $k = 1, \ldots, \ell$ let $T_k : \Omega \to \mathbb{Z}$ be defined by

$$T_k(x) \;=\; \textstyle\sum_{\kappa=1}^{k} (-2)^{\kappa-1} \sum_{S \subseteq [\ell] \,:\, |S|=\kappa} \chi_S(x).$$

We have $p = \mathrm{E}[P]$ and $t_k = \mathrm{E}[T_k]$. To prove the lemma, it suffices to show that for all $x \in \Omega$, (i) $P(x) \leq T_k(x)$ if k is odd, (ii) $P(x) \geq T_k(x)$ if k is even, and (iii) $P(x) = T_\ell(x)$.

Fix an arbitrary $x \in \Omega$ and let $\ell_x = \big|\{i \,:\, x \in E_i\}\big|$. Note that

$$T_k(x) \;=\; \textstyle\sum_{\kappa=1}^{\min(k,\ell_x)} (-2)^{\kappa-1} \binom{\ell_x}{\kappa} \;=\; \frac{1}{2} - \frac{1}{2} T'_k(x)$$

where

$$T'_k(x) \;=\; \textstyle\sum_{\kappa=0}^{\min(k,\ell_x)} (-2)^{\kappa} \binom{\ell_x}{\kappa}.$$

Now if $k \geq \ell_x$ then by the binomial theorem we have

$$T'_k(x) \;=\; \textstyle\sum_{\kappa=0}^{\ell_x} 1^{\ell_x - \kappa} (-2)^{\kappa} \binom{\ell_x}{\kappa} \;=\; (-1)^{\ell_x}$$

and thus $P(x) = T_k(x)$. This establishes (iii) since $\ell \geq \ell_x$, and it establishes (i) and (ii) assuming $k \geq \ell_x$. Now assume $k \leq \ell_x$. We claim that $T'_k(x) \leq -1$ if k is odd, and $T'_k(x) \geq 1$ if k is even. Assuming the claim, (i) follows by $P(x) \leq 1$, and (ii) follows by $P(x) \geq 0$. We already established the claim for $k = \ell_x$, and the claim trivially holds for $k = 0$. For $\kappa = 0, \ldots, \ell_x$ define $\tau_\kappa(x) = 2^\kappa \binom{\ell_x}{\kappa}$ so that $T'_k(x) = \sum_{\kappa=0}^{k} (-1)^\kappa \tau_\kappa(x)$. Note that the sequence $\tau_0(x), \tau_1(x), \ldots, \tau_{\ell_x}(x)$ is unimodal, since for $\kappa \geq 1$ we have

$$\tau_\kappa(x)/\tau_{\kappa-1}(x) \;=\; 2(\ell_x - \kappa + 1)/\kappa$$

which is at least 1 when $\kappa \leq \frac{2}{3}(\ell_x + 1)$ and at most 1 when $\kappa \geq \frac{2}{3}(\ell_x + 1)$. We now show by induction on $k = 0, 1, \ldots, \big\lfloor \frac{2}{3}(\ell_x + 1) \big\rfloor$ that the claim holds for these values of k (and a symmetric argument shows by induction on $k = \ell_x, \ell_x - 1, \ldots, \big\lceil \frac{2}{3}(\ell_x + 1) \big\rceil$ that the claim holds for these values of k). For the base cases, we know the claim holds for $k = 0$, and for $k = 1$ we have $T'_k(x) = 1 - 2\ell_x \leq -1$ since $\ell_x \geq k = 1$. Now assuming the claim holds for $k-2$, we prove it for k. Assume k is even (a symmetric argument handles the case k is odd). We have $T'_{k-2}(x) \geq 1$ and $T'_k(x) = T'_{k-2}(x) - \tau_{k-1}(x) + \tau_k(x) \geq 1$ since $\tau_k(x) \geq \tau_{k-1}(x)$. This finishes the proof of Lemma 3.36.

Chapter 4

Advice Lower Bounds for the Dense Model Theorem

4.1 Introduction

The question of whether the prime numbers contain arbitrarily long arithmetic progressions was a long-standing and famous open problem until Green and Tao [115] answered the question in the affirmative in a breakthrough paper in 2004. A key ingredient in their proof is a certain transference principle which, very roughly, states the following. Let U denote the set of positive integers. Then for every $D \subseteq U$, if there exists an $R \subseteq U$ such that D is dense in R and R is "indistinguishable" from U, then there exists an $M \subseteq U$ such that M is dense in U and D is "indistinguishable" from M. Tao and Ziegler [239] proved a much more general version of the transference principle, which has come to be known as the Dense Model Theorem (since M is a dense "model" for D).

Reingold, Trevisan, Tulsiani, and Vadhan [219] demonstrated the relevance of the Dense Model Theorem to computer science, and they gave a new proof which is much simpler and achieves better parameters than the proof of Green, Tao, and Ziegler. Gowers [112] independently came up with a similar proof. In addition to the original application of showing that the primes contain arbitrarily long arithmetic progressions, the Dense Model Theorem has found applications in differential privacy [190], pseudoentropy and leakage-resilient cryptography [30, 219, 81], and graph decompositions [219], as well as further applications in additive combinatorics [113, 114]. Subsequent variants of the Dense Model Theorem have found applications in cryptography [95] and pseudorandomness [245].

To formally state the Dense Model Theorem, we first need some definitions. We identify $\{0,1\}^{2^n}$ with the set of functions from $\{0,1\}^n$ to $\{0,1\}$. We use $\mathcal{D}_n$ to denote the set of all distributions on $\{0,1\}^n$. The domain $\{0,1\}^n$ could be replaced by any finite set of size 2^n; we use the domain $\{0,1\}^n$ for concreteness.

Definition 4.1. *We say $D_1 \in \mathcal{D}_n$ is δ-*dense *in $D_2 \in \mathcal{D}_n$ if for all $x \in \{0,1\}^n$, $\Pr_{D_1}[x] \leq \frac{1}{\delta}\Pr_{D_2}[x]$.*

Definition 4.2. *We say $f \in \{0,1\}^{2^n}$ ϵ-*distinguishes *$D_1, D_2 \in \mathcal{D}_n$ if $\left|\mathrm{E}_{D_1}[f] - \mathrm{E}_{D_2}[f]\right| > \epsilon$.*

$$\begin{tikzcd}
D \arrow[r, "\delta\text{-dense}"] \arrow[d, leftrightarrow, "(\epsilon, F)\text{-indistinguishable}"'] & R \arrow[d, leftrightarrow, "(\epsilon', F')\text{-indistinguishable}"] \\
M \arrow[r, "\delta\text{-dense}"'] & U
\end{tikzcd}$$

Figure 4.1: Relations among distributions in the Dense Model Theorem

Definition 4.3. *For $F \subseteq \{0,1\}^{2^n}$, we say $D_1, D_2 \in \mathcal{D}_n$ are* (ϵ, F)-indistinguishable *if there is no $f \in F$ that ϵ-distinguishes D_1 and D_2.*

The following is quantitatively the best known version of the theorem, due to Zhang [278] (building on [219, 26]).

Theorem 4.4 (Dense Model Theorem). *For every $F \subseteq \{0,1\}^{2^n}$ and every $D \in \mathcal{D}_n$, if there exists an $R \in \mathcal{D}_n$ such that D is δ-dense in R and (R, U) are (ϵ', F')-indistinguishable where $U \in \mathcal{D}_n$ is the uniform distribution, then there exists an $M \in \mathcal{D}_n$ such that M is δ-dense in U and (D, M) are (ϵ, F)-indistinguishable, where $\epsilon' \geq \Omega(\epsilon\delta)$ and F' consists of all linear threshold functions with ± 1 coefficients applied to $O\big((1/\epsilon^2)\log(1/\delta)\big)$ functions from F.*

The relations among the four distributions in Theorem 4.4 are illustrated in Figure 4.1. We remark that the theorem also holds when we allow $[0,1]$-valued functions f rather than just $\{0,1\}$-valued functions f. The proof of [219] gives the same result but where $O\big((1/\epsilon^2)\log(1/\epsilon\delta)\big)$ functions from F are combined to get a function from F'. The original proof of [239] achieves an F' which is qualitatively simpler, namely all products of $\text{poly}(1/\epsilon, 1/\delta)$ functions from F, but it only achieves $\epsilon' \geq \exp(-\text{poly}(1/\epsilon, 1/\delta))$.[1] We note that the dependence $\epsilon' \geq \Omega(\epsilon\delta)$ is tight in two senses.

- The Dense Model Theorem is actually false when $\epsilon' > \epsilon\delta$, even if $F' = \{0,1\}^{2^n}$. See [278] for the simple argument.
- The following converse to the Dense Model Theorem holds: If there exists an $M \in \mathcal{D}_n$ such that M is δ-dense in U and (D, M) are (ϵ, F)-indistinguishable, then there exists an $R \in \mathcal{D}_n$ such that D is δ-dense in R and (R, U) are (ϵ', F')-indistinguishable, where $\epsilon' = \epsilon\delta$ and $F' = F$. To see this, note that $U = \delta M + (1-\delta)\widehat{M}$ for some $\widehat{M} \in \mathcal{D}_n$, so we can let $R = \delta D + (1-\delta)\widehat{M}$; then D is δ-dense in R, and for every $f \in \{0,1\}^{2^n}$ we have $\mathrm{E}_R[f] - \mathrm{E}_U[f] = \delta\big(\mathrm{E}_D[f] - \mathrm{E}_M[f]\big)$ and thus if $\big|\mathrm{E}_R[f] - \mathrm{E}_U[f]\big| > \epsilon'$ then $\big|\mathrm{E}_D[f] - \mathrm{E}_M[f]\big| > \epsilon$.

[1]Another proof that also achieves this is given in [219].

The Dense Model Theorem has an undesirable feature: The class F' is more complex than the class F. Thus, if we wish to conclude that D and M are indistinguishable for a class F, we need to assume that R and U are indistinguishable for a more complex class F'. The less complex F' is, the stronger the theorem is. The reason for this loss in complexity is because the theorem is proved using a *black-box reduction*. In other words, the contrapositive is proved: We assume that for every M δ-dense in U there exists a function from F that ϵ-distinguishes D and M, and we show that some of these functions can be plugged into the reduction to get a function that ϵ'-distinguishes R and U. Thus the resulting function is necessarily more complex than the functions that get plugged into the reduction. There are three notions of complexity that are interesting to address in this context.

1. *Computational complexity.* If F consists of functions computed by small constant-depth circuits (AC^0), then can we let F' consist of functions computed by (slightly larger) constant-depth circuits? This is not known to be true when $\epsilon' \geq \Omega(\epsilon\delta)$, because the reductions of [219, 278] involve a linear threshold function, which cannot be computed by small constant-depth circuits. Is it necessary that the reduction computes a linear threshold function? The original result of [239] shows that this is *not* necessary if ϵ' is inverse exponentially small.

2. *Query complexity.* If F consists of functions computed by circuits of size s, then F' will need to consist of functions computed by circuits of a larger size s' — but how much larger? If the reduction makes q queries to functions from F, then plugging in size-s circuits for these functions yields a circuit of size $\geq q \cdot s$, and thus we must have $s' \geq q \cdot s$. Hence it is desirable to minimize q. Can we do better than $q \leq O\big((1/\epsilon^2)\log(1/\delta)\big)$ as in Theorem 4.4?

3. *Advice complexity.* Suppose F consists of functions computed by uniform algorithms running in time t (that is, a single algorithm computes a sequence of functions, one for each $n = 1, 2, 3, \ldots,$). Then can we let F' consist of functions computed by uniform algorithms running in some (slightly larger) time t'? (Here, the distributions D, M, R, U would need to be sequences of distributions, and a distinguisher would only be required to succeed for infinitely many n.) The proofs of [219, 278] do not yield this, because the reductions need a nonuniform advice string to provide some extra information about the nth distribution D.[2] How many bits of advice are needed?

When we call item 1 *computational complexity*, we are referring to resources that are directly associated with the circuit computing the reduction. All three notions of complexity, however, fall under the scope of "computational complexity" in general (the topic of this dissertation).

Before proceeding we draw attention to the fact that, as we just alluded to, the advice strings used by the reductions of [219, 278] depend on D *but do not depend on* R. Hence something a little stronger than Theorem 4.4 actually holds: Although the statement of

[2]The paper [219] also proves a version of the Dense Model Theorem that satisfies a certain technical relaxation of being "uniform", where ϵ figures into the quantification over all algorithms and all n.

Theorem 4.4 says we need to assume that for some R in which D is δ-dense, there is no function in F' that ϵ'-distinguishes R and U, we actually only need to assume that there is no function in F' that simultaneously ϵ'-distinguishes U from every R in which D is δ-dense (the quantifiers are swapped). We are interested in proving lower bounds on the complexity of this type of black-box reduction for the Dense Model Theorem, where the advice does not depend on R.

The *query complexity* was considered by Zhang [278], who showed that for a certain type of nonadaptive black-box reduction, $\Omega\big((1/\epsilon^2)\log(1/\delta)\big)$ queries are necessary when $\epsilon' \geq \Omega(\epsilon\delta)$ and $\epsilon \geq \delta^{O(1)}$, matching the upper bound of $O\big((1/\epsilon^2)\log(1/\delta)\big)$ for this case. In this chapter we consider the *advice complexity*. We show that for arbitrary black-box reductions, $\Omega\big(\sqrt{(1/\epsilon)\log(1/\delta)}\cdot\log|F|\big)$ advice bits are necessary when $\epsilon' \geq \Omega(\epsilon\delta)$ and $\epsilon \geq \delta^{O(1)}$, which comes close to matching the upper bound of $O\big((1/\epsilon^2)\log(1/\delta)\cdot\log|F|\big)$ for this case. Our result also holds for much more general settings of the parameters (with some degradation in the lower bound). Proving lower bounds on the *computational complexity* remains open.

Let us formally state what we mean by a black-box reduction. Recall the standard notation $[k] = \{1,\ldots,k\}$.

Definition 4.5. *An* $(n,\epsilon,\delta,\epsilon',k,\alpha)$-reduction *(for the Dense Model Theorem) is a function*

$$\mathrm{Dec} : \big(\{0,1\}^{2^n}\big)^k \times \{0,1\}^\alpha \to \{0,1\}^{2^n}$$

such that for all $f_1,\ldots,f_k \in \{0,1\}^{2^n}$ and all $D \in \mathcal{D}_n$, if for every $M \in \mathcal{D}_n$ that is δ-dense in the uniform distribution $U \in \mathcal{D}_n$ there exists an $i \in [k]$ such that f_i ϵ-distinguishes D and M, then there exists an advice string $a \in \{0,1\}^\alpha$ such that for every $R \in \mathcal{D}_n$ in which D is δ-dense, $\mathrm{Dec}(f_1,\ldots,f_k,a)$ *ϵ'-distinguishes R and U.*

The proofs of [219, 278] work by exhibiting such reductions. The functions $\{f_1,\ldots,f_k\}$ correspond to the class F (which, if we were considering uniform algorithms, would be the restrictions of all the algorithms in the class to a particular input length n). We now state our theorem.

Theorem 4.6. *If there exists an $(n,\epsilon,\delta,\epsilon',k,\alpha)$-reduction for the Dense Model Theorem, and if $w > 1$ is an integer such that $2^{w+2}\cdot\delta^{w/160} \leq \epsilon'$, then*

$$\alpha \;\geq\; \Big\lfloor \tfrac{1}{160w}\sqrt{(1/\epsilon)\log_2(1/\delta)}\Big\rfloor \cdot \log_2 k - \log_2 w - 1$$

provided $2^n \geq \frac{w\log_2 k}{\epsilon\delta^2(\epsilon')^2}$, $\epsilon \leq 1/64\log_2(1/\delta)$, and $k \geq 1/16\epsilon^4$.

For the case where $\epsilon' \geq \Omega(\epsilon\delta)$ and $\epsilon \geq \delta^{O(1)}$ (which is reasonable), the condition $2^{w+2}\cdot\delta^{w/160} \leq \epsilon'$ is met provided w is a sufficiently large constant and δ is less than a sufficiently small constant,[3] and thus we get a lower bound $\alpha \geq \Omega\big(\sqrt{(1/\epsilon)\log(1/\delta)}\cdot\log k\big)$. Note that the three conditions at the end of the statement of Theorem 4.6 are very generous.[4]

[3]The statement of Theorem 4.6 requires $\delta < 2^{-160}$. This constant can be drastically improved; we chose 2^{-160} since it is convenient for the proof.

[4]The bound $2^n \geq \frac{w\log_2 k}{\epsilon\delta^2(\epsilon')^2}$ can be relaxed somewhat. We chose this bound since it is reasonable and is convenient to state.

We point out that the black-box reductions we consider can be viewed as a kind of analogue of list-decoders for error-correcting codes. In the setting of list-decoding, the decoder is given a received word and is required to output a list containing all the codewords that are within a certain Hamming distance of the received word. In the setting of "dense model decoding", the distribution D is analogous to the correct codeword, and the decoder is only provided with some "corrupted" information about D, namely the functions $f_1, \ldots, f_k$. Note that these functions contain some information about D, since D can be distinguished from any M δ-dense in U using one of the f_i's. The decoder must output a list (corresponding to all possible values of the advice string $a \in \{0,1\}^\alpha$), but the goal is less ambitious than finding D itself; the list just needs to contain a function that distinguishes U from every R in which D is δ-dense. Thus advice lower bounds for the Dense Model Theorem are a kind of analogue of list size lower bounds for list-decoding. See Section 4.1.1 for previous work on list size lower bounds (as well as other previous work that is relevant to the topic of this chapter). For the case of approximate local list-decoding (also known as hardness amplification), getting the number of advice bits from $\mathrm{poly}(1/\epsilon)$ down to the lower bound of $\Omega(\log(1/\epsilon))$ proved to be quite a challenge [136, 137]. In contrast, we show that in the setting of the Dense Model Theorem, the known advice lengths are already in the right regime, namely $\mathrm{poly}(1/\epsilon)$.

The rest of this chapter is devoted to proving Theorem 4.6. In Section 4.2 we give some intuition for the proof, and then in Section 4.3 we give the formal proof. We now give a quick preview of some of the ingredients that go into the proof. We use the probabilistic method to find a class of functions $f_1, \ldots, f_k$ for which many advice strings are needed to "cover" all the distributions D that do not have dense models. The key technical ingredients in the analysis include (1) a combinatorial argument identifying when several distributions D cannot share the same advice string, and (2) an analysis of a majority of majorities applied to overlapping sets of p-biased bits, where the sets form an almost-disjoint family (see Figure 4.2). The latter analysis makes use of extremely tight lower bounds on the tail probabilities of the binomial distribution, which we also prove.

4.1.1 Related Work

Lu, Tsai, and Wu [178] proved lower bounds on the computational complexity, query complexity, and advice complexity of black-box reductions for the Hardcore Lemma (which was introduced by Impagliazzo [133]). Our proof bears some similarity to their advice lower bound proof, but it diverges significantly. Zhang [278] proved tight lower bounds on the query complexity of nonadaptive black-box reductions for the Dense Model Theorem, again with a proof somewhat reminiscent of the corresponding argument in [178].

There has been extensive work on lower bounds for black-box hardness amplification and list-decoding of error-correcting codes. Regarding *advice complexity*, Guruswami and Vadhan [122] and Blinovsky [38] proved a tight $\Omega(1/\epsilon^2)$ list size lower bound for decoding arbitrary binary error-correcting codes up to radius $1/2 - \epsilon$. Lu, Tsai, and Wu [177] proved similar advice lower bounds for black-box hardness amplification. List size lower bounds for decoding from erasures can be found in [117, 268] (see Chapter 8). Regarding *query com-*

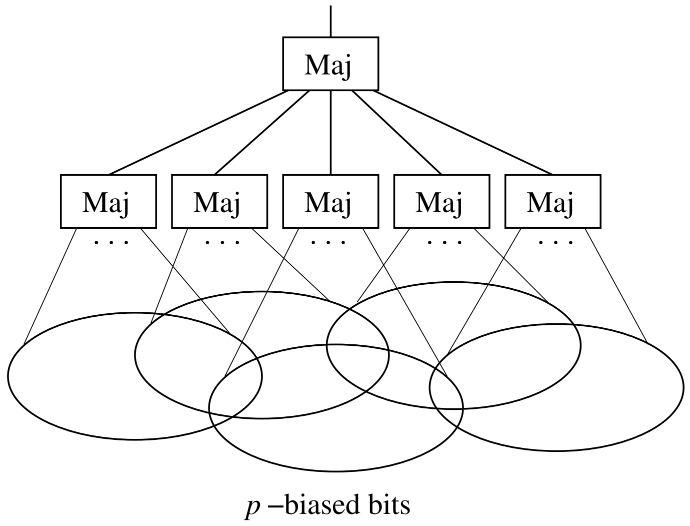

Figure 4.2: The majority of majorities

plexity, Shaltiel and Viola [231] proved lower bounds for nonadaptive black-box hardness amplification, matching the upper bounds known for the XOR Lemma [166, 133, 102, 161]. Artemenko and Shaltiel [19] proved lower bounds for adaptive black-box hardness amplification, which are not tight in general but are tight in some settings. Regarding *computational complexity*, Shaltiel and Viola [231] showed that decoders for black-box hardness amplification must implicitly compute majority on $\Theta(1/\epsilon)$ bits and hence such decoders cannot be implemented with small constant-depth circuits when ϵ is small. Gutfreund and Rothblum [123] showed that adaptive local list-decoders for binary codes must implicitly compute majority on $\Theta(1/\epsilon)$ bits to handle radius $1/2 - \epsilon$, under a restriction on the list size.

4.2 Intuition

According to Definition 4.5, for Dec to succeed as a reduction, it must be the case that for all $f_1, \ldots, f_k \in \{0,1\}^{2^n}$ and all $D \in \mathcal{D}_n$, if D has no "dense model" then there is some advice string a such that $\text{Dec}(f_1, \ldots, f_k, a)$ "covers" D in a certain sense. To show that Dec needs many advice strings in order to succeed, we find functions $f_1, \ldots, f_k \in \{0,1\}^{2^n}$ and a large family of distributions in $\mathcal{D}_n$ such that

(i) each distribution in the family has no dense model (with respect to $f_1, \ldots, f_k$), and

(ii) each function $f \in \{0,1\}^{2^n}$ covers few of the distributions in the family.

So (i) implies that each distribution in the family needs to get covered, while (ii) implies that for each advice string a, $\text{Dec}(f_1, \ldots, f_k, a)$ does not cover very many of them. Since the family is large, many advice strings are needed.

First we describe a technique for achieving (i), then we describe a technique for achieving (ii), and then we show how to consolidate the techniques to achieve both properties simultaneously. When we say D has no "dense model" we mean that for every $M \in \mathcal{D}_n$ that is δ-dense in U there exists an $i \in [k]$ such that f_i ϵ-distinguishes D and M. When we say a function "covers" D we mean that it ϵ'-distinguishes R and U for every $R \in \mathcal{D}_n$ in which D is δ-dense. The only distributions D we need to consider are uniform distributions over subsets of $\{0,1\}^n$.

Given $f_1, \ldots, f_k \in \{0,1\}^{2^n}$, what is an example of a distribution with no dense model? Suppose we pick any $I \subseteq [k]$ of size $1/4\epsilon$ and we let X_I be the set of all $x \in \{0,1\}^n$ such that $f_i(x) = 1$ for the majority of $i \in I$. Suppose we take D_I to be the uniform distribution over X_I. Then we have $\Pr_{x \sim D_I,\ i \sim I}[f_i(x) = 1] \geq 1/2 + 2\epsilon$ where $i \sim I$ means picking $i \in I$ uniformly at random. If X_I is roughly a $\delta/2$ fraction of $\{0,1\}^n$, then every distribution M that is δ-dense in U has at least half its mass outside of X_I, on strings x where $\Pr_{i \sim I}[f_i(x) = 1] \leq 1/2 - 2\epsilon$. It is possible to show that $\Pr_{x \sim M,\ i \sim I}[f_i(x) = 1] < \Pr_{x \sim D_I,\ i \sim I}[f_i(x) = 1] - \epsilon$ and thus there exists an $i \in I$ (depending on M) such that f_i ϵ-distinguishes D_I and M. So if $|X_I| \approx (\delta/2)2^n$ then D_I has no dense model. This is the technique we use for finding distributions without dense models.

Now, what is an example of a pair of distributions such that no function can cover both simultaneously? If we can show that every pair of distributions in the family is like this, then we will have achieved (ii). Because of an issue described below, we actually need to consider small collections of distributions rather than just pairs, but for now we consider pairs. Suppose D is uniform over some $X \subseteq \{0,1\}^n$ of size roughly $(\delta/2)2^n$, and similarly D' is uniform over some $X' \subseteq \{0,1\}^n$ of size roughly $(\delta/2)2^n$. If $X \cap X' = \emptyset$, then it can be shown that no function covers both D and D'.[5] Furthermore, if $|X \cap X'|$ is at most roughly $\epsilon' 2^n$ then this property still holds.

To consolidate the two techniques, we would like to find a large family of sets $I \subseteq [k]$ each of size $1/4\epsilon$, such that

(A) $|X_I| \approx (\delta/2)2^n$ for each I in the family, and

(B) the pairwise intersections of the X_I's (for I in the family) all have size at most roughly $\epsilon' 2^n$.

This would imply that the corresponding distributions D_I (for I in the family) have no dense models, and no function would cover more than one of them, so (i) and (ii) would be achieved.

We choose the functions $f_1, \ldots, f_k \in \{0,1\}^{2^n}$ randomly in some way, and we argue that for an appropriate family of sets I, properties (A) and (B) both hold with high probability. Property (A) suggests that we should choose p so that the probability a majority of $1/4\epsilon$ independent coins each with expectation p come up 1 is exactly $\delta/2$. Then we can set $f_i(x) = 1$ with probability p independently for each $i \in [k]$ and each $x \in \{0,1\}^n$, so for each I of size $1/4\epsilon$, $\Pr[x \in X_I] = \delta/2$. Then by concentration, $|X_I| \approx (\delta/2)2^n$ with high probability over $f_1, \ldots, f_k$.

[5] Actually, there is an issue having to do with the absolute value signs in the definition of distinguishing; this is dealt with in the formal proof.

If we choose $f_1, \ldots, f_k$ randomly in this way, how big will $|X_I \cap X_{I'}|$ be, for I and I' in the family? By concentration, we would have that with high probability over $f_1, \ldots, f_k$, $|X_I \cap X_{I'}|$ is roughly 2^n times $\Pr[x \in X_I \cap X_{I'}]$ (which is the same for all $x \in \{0,1\}^n$), so we would like the latter probability to be $\leq \epsilon'$. So what is the probability that the conjunction of two majorities of p-biased bits is 1? The best case is if $I \cap I' = \emptyset$, in which case the probability is exactly $(\delta/2)^2$. There are two problems with this.

(1) We cannot get a very large family of sets I if we require them to be pairwise disjoint.

(2) This requires $\epsilon' \geq (\delta/2)^2$. In a typical setting where $\epsilon' \geq \Omega(\epsilon\delta)$, this would require $\epsilon > \delta$, which is an odd and somewhat severe restriction.

To solve problem (1), we use the natural idea to allow the sets I to be pairwise almost-disjoint, rather than disjoint (which allows us to get a much larger family). So if $|I \cap I'|$ is at most some value b, how small does b have to be to ensure that the probability both majorities are 1 is not much more than $(\delta/2)^2$? We analyze this using the following trick: If both majorities are 1, then the fraction of coins that are 1 among $I \cup I'$ is at least q, where $q = 1/2 - 2\epsilon b = \frac{1/4\epsilon - b}{1/2\epsilon} \leq \frac{|I|/2 + |I'|/2 - b}{|I \cup I'|}$. Using an extremely tight characterization of the tail probabilities of the binomial distribution (which we prove using known techniques but which we could not find in the literature), we can show that $p \approx 1/2 - \sqrt{\epsilon \log(1/\delta)}$ and the probability of getting $\geq q$ fraction of 1's among the $|I \cup I'|$ coins is not much more than $(\delta/2)^2$ provided q is at least a constant factor closer to 1/2 than p is, say $q \approx 1/2 - \sqrt{\epsilon \log(1/\delta)}/4$. Thus it suffices to have $b \approx \sqrt{\epsilon \log(1/\delta)}/8\epsilon \geq \Omega\big(\sqrt{(1/\epsilon)\log(1/\delta)}\big)$. Since the family of sets I needs to be in the universe $[k]$, there exists such a family of roughly k^b many sets with pairwise intersections bounded in size by b. Since each function can cover D_I for only one I in the family, roughly k^b advice strings are needed, which gives an advice lower bound of roughly $\log(k^b) \geq \Omega\big(\sqrt{(1/\epsilon)\log(1/\delta)} \cdot \log k\big)$.

Problem (2) is solved in the formal proof by considering small collections of sets from the family, rather than pairs. The parameter w in Theorem 4.6 is used to determine how big these collections should be. Then instead of requiring that the conjunction of two majorities accepts with small probability, we need that the majority of several majorities accepts with small probability, which explains where Figure 4.2 comes from.

4.3 Formal Proof

In Section 4.3.1, Section 4.3.2, and Section 4.3.3 we give preliminary lemmas, definitions, and notation. Then in Section 4.3.4 we give the proof of Theorem 4.6.

4.3.1 Binomial Distribution Tail

We let $\mathrm{Tail}(m, p, q)$ denote the probability that when m independent coins are flipped each with probability p of heads, at least a q fraction of the coins are heads (in other words, the probability the (m, p) binomial distribution is at least qm). For our proof of Theorem 4.6

we need extremely tight upper and lower bounds on the value of $\mathrm{Tail}(m,p,q)$. Such bounds can be given in terms of the fundamental quantity

$$\mathrm{RE}(q\|p) \;=\; q\log_2(\tfrac{q}{p}) + (1-q)\log_2(\tfrac{1-q}{1-p})$$

which is known by a variety of names such as relative entropy, information divergence, and Kullback-Leibler distance.[6]

We need the following fact, which can be seen using derivatives.

Fact 4.7. *For all* $1/4 \le p \le q \le 3/4$*, we have* $2(q-p)^2 \le \mathrm{RE}(q\|p) \le 4(q-p)^2$.

We also need the following standard and well-known form of the Chernoff-Hoeffding bound.

Lemma 4.8. *For all* $m \ge 1$ *and all* $0 \le p \le q \le 1$*, we have* $\mathrm{Tail}(m,p,q) \le 2^{-\mathrm{RE}(q\|p)m}$.

Lemma 4.8 is very tight, as shown by the following lemma, which we prove for completeness.

Lemma 4.9. *For all* $m \ge 1$ *and all* $1/4 \le p \le q \le 1$*, we have* $\mathrm{Tail}(m,p,q) \ge \frac{1}{48\sqrt{m}} \cdot 2^{-\mathrm{RE}(q\|p)m}$.

Proof. First, assume that qm is an integer. Then lower bounding $\mathrm{Tail}(m,p,q)$ by the first term of the sum, we have

$$\begin{aligned}
2^{\mathrm{RE}(q\|p)m}\cdot \mathrm{Tail}(m,p,q) &\ge 2^{\mathrm{RE}(q\|p)m}\cdot \tbinom{m}{qm} p^{qm}(1-p)^{(1-q)m} \\
&= \tfrac{q^{qm}(1-q)^{(1-q)m}}{p^{qm}(1-p)^{(1-q)m}} \cdot \tbinom{m}{qm} p^{qm}(1-p)^{(1-q)m} \\
&= q^{qm}(1-q)^{(1-q)m}\cdot \tbinom{m}{qm} \\
&\ge q^{qm}(1-q)^{(1-q)m}\cdot \tfrac{1}{3\sqrt{qm}} \cdot \tfrac{1}{q^{qm}(1-q)^{(1-q)m}} \\
&\ge \tfrac{1}{3\sqrt{m}}
\end{aligned}$$

where the fourth line follows by Stirling approximations. Now suppose qm is not an integer, and let $q' = \lceil qm \rceil / m$. Then we have $\mathrm{Tail}(m,p,q) = \mathrm{Tail}(m,p,q') \ge \frac{1}{3\sqrt{m}} \cdot 2^{-\mathrm{RE}(q'\|p)m}$. We claim that $\mathrm{RE}(q'\|p) - \mathrm{RE}(q\|p) \le 4/m$, from which it follows that $2^{-\mathrm{RE}(q'\|p)m}/2^{-\mathrm{RE}(q\|p)m} \ge 1/16$ and thus $\mathrm{Tail}(m,p,q) \ge \frac{1}{48\sqrt{m}} \cdot 2^{-\mathrm{RE}(q\|p)m}$. We now argue the claim. Since $q' \le q + 1/m$, we have $q'\log_2(\frac{q'}{p}) - q\log_2(\frac{q}{p}) \le (1/m)\log_2(\frac{1}{p}) + q\log_2(\frac{q+1/m}{q})$. We have $(1/m)\log_2(\frac{1}{p}) \le 2/m$ since $p \ge 1/4$, and we have $q\log_2(\frac{q+1/m}{q}) \le q\cdot 2/qm = 2/m$. Thus $q'\log_2(\frac{q'}{p}) - q\log_2(\frac{q}{p}) \le 4/m$. Since $q' \ge q$, we have $(1-q')\log_2(\frac{1-q'}{1-p}) - (1-q)\log_2(\frac{1-q}{1-p}) \le 0$. Summing gives the claim. ☐

[6] RE is more often notated D or D_{KL}, but we use RE to avoid confusion with our distributions D.

Although Lemma 4.9 is very simple and general, for our purpose we can only use it for a limited range of parameters, namely when $\epsilon \gg \delta$. This is because $\mathrm{RE}(q\|p)$ could be so close to 0 that $\frac{1}{48\sqrt{m}}$ completely swamps $2^{-\mathrm{RE}(q\|p)m}$, in which case Lemma 4.9 is not very tight. To handle the full range of ϵ and δ, we use the following stronger lower bound for the case $q = 1/2$.

Lemma 4.10. *For all $m \geq 9$ and all $1/4 \leq p < 1/2$, we have*

$$\mathrm{Tail}(m, p, 1/2) \;\geq\; \min\left(\tfrac{1}{256},\ \tfrac{1}{128\sqrt{m}(1/2-p)}\right) \cdot 2^{-\mathrm{RE}(1/2\|p)m}.$$

Proof. Let $q = \lceil m/2 \rceil / m$. Let $h = \lfloor \sqrt{m}/3 \rfloor$, and note that $1 \leq h \leq (1-q)m$. We have

$$\begin{aligned}
2^{\mathrm{RE}(q\|p)m} \cdot \mathrm{Tail}(m,p,q) &= \tfrac{q^{qm}(1-q)^{(1-q)m}}{p^{qm}(1-p)^{(1-q)m}} \cdot \textstyle\sum_{i=0}^{(1-q)m} \binom{m}{qm+i} p^{qm+i}(1-p)^{(1-q)m-i} \\
&= q^{qm}(1-q)^{(1-q)m} \cdot \textstyle\sum_{i=0}^{(1-q)m} \binom{m}{qm+i} \left(\tfrac{p}{1-p}\right)^i \\
&= q^{qm}(1-q)^{(1-q)m} \cdot \textstyle\sum_{i=0}^{(1-q)m} \binom{m}{qm} \left(\tfrac{p}{1-p}\right)^i \prod_{j=1}^{i} \left(1 + \tfrac{m-2qm-i}{qm+j}\right) \\
&\geq \tfrac{1}{3\sqrt{qm}} \cdot \textstyle\sum_{i=0}^{(1-q)m} \left(\tfrac{p}{1-p}\right)^i \prod_{j=1}^{i} \left(1 + \tfrac{m-2qm-i}{qm+j}\right) \\
&\geq \tfrac{1}{3\sqrt{qm}} \cdot \textstyle\sum_{i=0}^{(1-q)m} \left(\tfrac{p}{1-p}\right)^i \left(1 - \tfrac{2(i+1)}{m}\right)^i \\
&\geq \tfrac{1}{3\sqrt{qm}} \cdot \left(1 - \tfrac{2(h+1)}{m}\right)^h \cdot \textstyle\sum_{i=0}^{h} \left(\tfrac{p}{1-p}\right)^i \\
&\geq \tfrac{1}{3\sqrt{qm}} \cdot \left(1 - \tfrac{2h(h+1)}{m}\right) \cdot \textstyle\sum_{i=0}^{h} \left(\tfrac{p}{1-p}\right)^i \\
&\geq \tfrac{1}{6\sqrt{qm}} \cdot \textstyle\sum_{i=0}^{h} \left(\tfrac{p}{1-p}\right)^i \\
&= \tfrac{1}{6\sqrt{qm}} \cdot \tfrac{1-\left(\frac{p}{1-p}\right)^{h+1}}{1-\frac{p}{1-p}} \\
&\geq \tfrac{1}{6\sqrt{qm}} \cdot \tfrac{1-e^{-(1-\frac{p}{1-p})h}}{1-\frac{p}{1-p}}
\end{aligned}$$

where the fourth line follows by $\binom{m}{qm} \geq \frac{1}{3\sqrt{qm}} \cdot \frac{1}{q^{qm}(1-q)^{(1-q)m}}$ which holds by Stirling approximations, the fifth line follows by $1/2 \leq q \leq 1/2 + 1/2m$, and the eighth line follows by the definition of h and $m \geq 9$. If $(1 - \frac{p}{1-p})h < 1$ then the expression is at least

$$\frac{1}{6\sqrt{qm}} \cdot \frac{1-\left(1-\frac{1}{2}(1-\frac{p}{1-p})h\right)}{1-\frac{p}{1-p}} \;=\; \frac{h}{12\sqrt{qm}} \;\geq\; \frac{1}{64}.$$

If $(1 - \frac{p}{1-p})h \geq 1$ then the expression is at least

$$\frac{1}{6\sqrt{qm}} \cdot \frac{1-1/e}{1-\frac{p}{1-p}} \;\geq\; \frac{1}{10\sqrt{qm}} \cdot \frac{1}{1-\frac{p}{1-p}} \;\geq\; \frac{1}{10\sqrt{qm}} \cdot \frac{1}{4\cdot(1/2-p)} \;\geq\; \frac{1}{32\sqrt{m}(1/2-p)}$$

where the last inequality uses $m \geq 9$. Thus we have shown that in either case,

$$\mathrm{Tail}(m,p,1/2) \;=\; \mathrm{Tail}(m,p,q) \;\geq\; \min\left(\tfrac{1}{64},\ \tfrac{1}{32\sqrt{m}(1/2-p)}\right) \cdot 2^{-\mathrm{RE}(q\|p)m}.$$

To finish the proof, we just need to show that $2^{-\mathrm{RE}(q\|p)m}/2^{-\mathrm{RE}(1/2\|p)m} \geq 1/4$. For this, it suffices to show that $\mathrm{RE}(q\|p) - \mathrm{RE}(1/2\|p) \leq 2/m$. Since $q \leq 1/2 + 1/2m$, we have $q\log_2(\frac{q}{p}) - (1/2)\log_2(\frac{1/2}{p}) \leq (1/2m)\log_2(\frac{1}{p}) + (1/2)\log_2(\frac{1/2+1/2m}{1/2})$. We have $(1/2m)\log_2(\frac{1}{p}) \leq 1/m$ since $p \geq 1/4$, and we have $(1/2)\log_2(\frac{1/2+1/2m}{1/2}) \leq (1/2)\cdot 2/m = 1/m$. Thus $q\log_2(\frac{q}{p}) - (1/2)\log_2(\frac{1/2}{p}) \leq 2/m$. Since $q \geq 1/2$, we have $(1-q)\log_2(\frac{1-q}{1-p}) - (1-1/2)\log_2(\frac{1-1/2}{1-p}) \leq 0$. Summing yields $\mathrm{RE}(q\|p) - \mathrm{RE}(1/2\|p) \leq 2/m$. □

4.3.2 Combinatorial Designs

For our proof of Theorem 4.6 we need the existence of large families of almost-disjoint subsets of a finite set. Such combinatorial designs have numerous applications in theoretical computer science, one of the more famous being in the pseudorandom generator construction of Nisan and Wigderson [201].

Definition 4.11. *An* (ℓ, k, s, b)-design *is a family of sets* $I_1, \ldots, I_\ell \subseteq [k]$ *all of size* s *such that* $|I_j \cap I_{j'}| \leq b$ *for every* $j \neq j'$.

Lemma 4.12. *For every* k, s, b *there exists an* (ℓ, k, s, b)*-design with* $\ell \geq k^{b/8}$*, provided* $k \geq 16s^4$.

There is nothing very novel about this lemma, and this precise version follows from a result in [82], but we provide a simple, self-contained proof here. The proof uses the probabilistic method with a simple concentration bound for the hypergeometric distribution.

Proof. If $b = 0$ then the lemma holds trivially, so assume $b \geq 1$. Let $\ell = \lceil k^{b/8} \rceil$. We pick sets $I_1, \ldots, I_\ell$ independently and uniformly at random from all subsets of $[k]$ of size s, and we argue that with positive probability $I_1, \ldots, I_\ell$ forms an (ℓ, k, s, b)-design. We claim that for every $j, j' \in [\ell]$ with $j \neq j'$, $\Pr\big[|I_j \cap I_{j'}| > b\big] \leq 2k^{-b/2}$. From this it follows by a union bound that

$$\Pr\big[I_1, \ldots, I_\ell \text{ does not form an } (\ell, k, s, b)\text{-design}\big] \;\leq\; \tbinom{\ell}{2} \cdot 2k^{-b/2} \;<\; \ell^2 \cdot k^{-b/2} \;\leq\; 1$$

where the final inequality $\lceil k^{b/8} \rceil \leq k^{b/4}$ follows by $k \geq 16$ and $b \geq 1$. To prove the claim, consider any $j \neq j'$ and fix any particular choice of I_j. Now consider picking points $i_1, \ldots, i_s \in [k]$ independently and uniformly at random (with replacement). Since the expected number of points that land in I_j is s^2/k, a standard relative-error form of the Chernoff bound tells us that

$$\Pr_{i_1,\ldots,i_s}\big[|\{h \,:\, i_h \in I_j\}| > b\big] \;\leq\; \left(\tfrac{es^2}{bk}\right)^b \;\leq\; k^{-b/2}$$

using $es^2 \leq k^{1/2}$ and $b \geq 1$ (where e is the base of the natural logarithm). The probability that $i_1, \ldots, i_s$ are all distinct is at least $1 - \binom{s}{2}/k \geq 1/2$ by a union bound and using $k \geq s^2$. Conditioning on the event that $i_1, \ldots, i_s$ are all distinct is equivalent to sampling $I_{j'} = \{i_1, \ldots, i_s\}$ of size s uniformly at random, and probabilities increase by at most a factor of 2 conditioned on this event. Thus $\Pr\big[|I_j \cap I_{j'}| > b\big] \leq 2k^{-b/2}$, as claimed. □

4.3.3 Notational Preliminaries

The parameters $n, \epsilon, \delta, \epsilon', k$, and w are fixed as in the statement of Theorem 4.6, and we always use D, M, R, U (possibly subscripted) to denote distributions in $\mathcal{D}_n$, in their respective roles as in Definition 4.5.

We let Maj denote the majority function on bit strings, and for even length strings we break ties by returning 1. We let And denote the and function on bit strings. We let Maj^t denote the function that takes t bit strings and returns their majorities as a length-t bit string. We use $\circ$ for function composition.

We also adhere to the following notational conventions. We use x for elements of $\{0,1\}^n$ and X for subsets of $\{0,1\}^n$. We use f for elements of $\{0,1\}^{2^n}$ (identified with functions from $\{0,1\}^n$ to $\{0,1\}$) and F for subsets of $\{0,1\}^{2^n}$. We use $[k]$ to index functions f, and we use i for elements of $[k]$ and I for subsets of $[k]$. We use $[\ell]$ to index subsets I (as in Definition 4.11), and we use j for elements of $[\ell]$ and J for subsets of $[\ell]$. Furthermore, we generally use s for the size of I, and t for the size of J.

The following notation is with respect to fixed $f_1, \ldots, f_k \in \{0,1\}^{2^n}$. Given $I \subseteq [k]$ we define

- f_I is the function that takes $x \in \{0,1\}^n$ and returns the length-$|I|$ bit string $(f_i(x))_{i \in I}$;
- X_I is the set of $x \in \{0,1\}^n$ on which $\text{Maj} \circ f_I$ returns 1;
- D_I is the uniform distribution over X_I (and if $X_I = \emptyset$ then D_I is undefined).

The following notation is with respect to fixed $f_1, \ldots, f_k \in \{0,1\}^{2^n}$ and fixed $I_1, \ldots, I_\ell \subseteq [k]$. Given $J \subseteq [\ell]$ we define

- f_{I_J} is the function that takes $x \in \{0,1\}^n$ and returns the $|J|$-tuple $(f_{I_j}(x))_{j \in J}$;
- X_{I_J} is the set of $x \in \{0,1\}^n$ on which $\text{Maj} \circ \text{Maj}^{|J|} \circ f_{I_J}$ returns 1.

We use $\sim$ to denote sampling from a distribution (for example $x \sim D$), and we use the convention that sampling from a set (for example $i \sim I$) means sampling from the uniform distribution over that set.

4.3.4 Proof of Theorem 4.6

Consider an arbitrary function

$$\text{Dec} : \left(\{0,1\}^{2^n}\right)^k \times \{0,1\}^\alpha \to \{0,1\}^{2^n}.$$

Supposing that $\alpha < \left\lfloor \frac{1}{160w} \sqrt{(1/\epsilon)\log_2(1/\delta)} \right\rfloor \cdot \log_2 k - \log_2 w - 1$, we show that Dec is not an $(n, \epsilon, \delta, \epsilon', k, \alpha)$-reduction. We first introduce some terminology to make things concise. Given $f_1, \ldots, f_k \in \{0,1\}^{2^n}$, a *dense model* for $D \in \mathcal{D}_n$ is an $M \in \mathcal{D}_n$ that is δ-dense in the uniform distribution $U \in \mathcal{D}_n$ and is such that for all $i \in [k]$, f_i does not ϵ-distinguish D and

M. We say a function $f \in \{0,1\}^{2^n}$ *covers* $D \in \mathcal{D}_n$ if for every $R \in \mathcal{D}_n$ in which D is δ-dense, f ϵ'-distinguishes R and U.

Thus to show that Dec is not an $(n, \epsilon, \delta, \epsilon', k, \alpha)$-reduction, we need to find $f_1, \ldots, f_k \in \{0,1\}^{2^n}$ such that some D has no dense model but is not covered by $\mathrm{Dec}(f_1, \ldots, f_k, a)$ for any advice string $a \in \{0,1\}^\alpha$.

4.3.4.1 Distributions Without Dense Models

The following claim is our tool for finding distributions that have no dense models.

Claim 4.13. *For every $f_1, \ldots, f_k \in \{0,1\}^{2^n}$ and every $I \subseteq [k]$ of size $0 < s \leq 1/4\epsilon$ (for some s), if $0 < |X_I| \leq (2\delta/3)2^n$ then D_I has no dense model.*

Proof. We only consider the case when s is odd (essentially the same argument works when s is even). Suppose we pick $i \in I$ uniformly at random. Then for each $x \in X_I$ we have $\Pr_{i\sim I}[f_i(x) = 1] \geq 1/2 + 1/2s \geq 1/2 + 2\epsilon$, and for each $x \in \{0,1\}^n \backslash X_I$ we have $\Pr_{i\sim I}[f_i(x) = 1] \leq 1/2 - 1/2s \leq 1/2 - 2\epsilon$. Thus we have $\Pr_{x\sim D_I,\ i\sim I}[f_i(x) = 1] \geq 1/2 + 2\epsilon$. Now consider an arbitrary M that is δ-dense in U. We have

$$\begin{aligned}
& \Pr_{x\sim M,\ i\sim I}[f_i(x) = 1] \\
\leq\ & \Pr_{x\sim M}[x \notin X_I] \cdot (1/2 - 2\epsilon) + \textstyle\sum_{x^*\in X_I} \Pr_{x\sim M}[x = x^*] \cdot \Pr_{i\sim I}[f_i(x^*) = 1] \\
\leq\ & \big(1 - |X_I|/\delta 2^n\big) \cdot (1/2 - 2\epsilon) + \textstyle\sum_{x^*\in X_I} (1/\delta 2^n) \cdot \Pr_{i\sim I}[f_i(x^*) = 1] \\
=\ & \big(1 - |X_I|/\delta 2^n\big) \cdot (1/2 - 2\epsilon) + (|X_I|/\delta 2^n) \cdot \Pr_{x\sim D_I,\ i\sim I}[f_i(x) = 1] \\
\leq\ & (1/3) \cdot (1/2 - 2\epsilon) + (2/3) \cdot \Pr_{x\sim D_I,\ i\sim I}[f_i(x) = 1] \\
=\ & \Pr_{x\sim D_I,\ i\sim I}[f_i(x) = 1] - (1/3) \cdot \big(\Pr_{x\sim D_I,\ i\sim I}[f_i(x) = 1] - (1/2 - 2\epsilon)\big) \\
\leq\ & \Pr_{x\sim D_I,\ i\sim I}[f_i(x) = 1] - (1/3) \cdot 4\epsilon.
\end{aligned}$$

Here the third line follows because $\Pr_{i\sim I}[f_i(x^*) = 1] > 1/2 - 2\epsilon$ holds for all $x^* \in X_I$ and thus the whole expression only gets larger by shifting probability mass from $\{0,1\}^n \backslash X_I$ to X_I. Similarly, the fifth line follows because the fourth line is a convex combination of $1/2 - 2\epsilon$ and $\Pr_{x\sim D_I,\ i\sim I}[f_i(x) = 1]$, so the whole expression gets larger by shifting weight to the larger of the two.

Since $\Pr_{x\sim D_I,\ i\sim I}[f_i(x) = 1] - \Pr_{x\sim M,\ i\sim I}[f_i(x) = 1] > \epsilon$, there must exist an $i \in I$ such that $\mathrm{E}_{D_I}[f_i] - \mathrm{E}_M[f_i] > \epsilon$ and thus f_i ϵ-distinguishes D_I and M. Hence M is not a dense model for D_I. This finishes the proof of Claim 4.13. ☐

4.3.4.2 Distributions That Cannot Be Covered

We say a function $f \in \{0,1\}^{2^n}$ *positively covers* $D \in \mathcal{D}_n$ if for every $R \in \mathcal{D}_n$ in which D is δ-dense, $\mathrm{E}_R[f] - \mathrm{E}_U[f] > \epsilon'$ (note the absence of absolute value signs). Observe that if $f \in \{0,1\}^{2^n}$ covers D then either f or its complement positively covers D. This is because if there existed $R_1, R_2 \in \mathcal{D}_n$ in which D is δ-dense and such that $\mathrm{E}_{R_1}[f] < \mathrm{E}_U[f] < \mathrm{E}_{R_2}[f]$, then some convex combination R_3 of R_1 and R_2 would have $\mathrm{E}_{R_3}[f] = \mathrm{E}_U[f]$. However, D

would be δ-dense in R_3 since the set of R in which D is δ-dense is convex, so f would not cover D.

Claim 4.14. *For every $f_1, \ldots, f_k \in \{0,1\}^{2^n}$, every $I_1, \ldots, I_\ell \subseteq [k]$ (for some ℓ), and every $J \subseteq [\ell]$ of size $t > 1$ (for some t), if $|X_{I_J}| \leq (\epsilon'/2)2^n$ and $|X_{I_j}| \geq (\delta/2 - \epsilon'/4)2^n$ for all $j \in J$ then there is no function that simultaneously positively covers D_{I_j} for all $j \in J$.*

Proof. Assume that $|X_{I_J}| \leq (\epsilon'/2)2^n$ and $|X_{I_j}| \geq (\delta/2 - \epsilon'/4)2^n$ for all $j \in J$. Consider an arbitrary $f \in \{0,1\}^{2^n}$ and let X be the set of $x \in \{0,1\}^n$ such that $f(x) = 1$. For $\tau \in \{0, 1, \ldots, t\}$ let $X^{(\tau)}$ be the set of $x \in \{0,1\}^n$ such that there are exactly τ values of $j \in J$ for which $x \in X_{I_j}$ (in other words, $(\mathrm{Maj}^t \circ f_{I_J})(x)$ has Hamming weight τ). Note that $X_{I_J} = \bigcup_{\tau=t'}^{t} X^{(\tau)}$ where $t' = \lceil t/2 \rceil$. Let $\pi = \min_{j \in J} \left[\mathrm{E}_{D_{I_j}}[f] \right]$. Then for every $j \in J$ we have $|X \cap X_{I_j}| \geq \pi \cdot |X_{I_j}| \geq \pi \cdot (\delta/2 - \epsilon'/4)2^n$. We have

$$\begin{aligned}
(t/2) \cdot \big(|X| + |X_{I_J}|\big) &\geq (t/2) \cdot |X \cap \overline{X_{I_J}}| + t \cdot |X \cap X_{I_J}| \\
&\geq \textstyle\sum_{\tau=0}^{t} \tau \cdot |X \cap X^{(\tau)}| \\
&= \textstyle\sum_{j \in J} |X \cap X_{I_j}| \\
&\geq t \cdot \pi \cdot (\delta/2 - \epsilon'/4)2^n
\end{aligned}$$

which implies that

$$|X| \;\geq\; \pi \cdot (\delta - \epsilon'/2)2^n - |X_{I_J}| \;\geq\; \pi\delta 2^n - \epsilon' 2^n \;=\; (\pi - \epsilon'/\delta) \cdot \delta 2^n$$

since $\pi \leq 1$ and $|X_{I_J}| \leq (\epsilon'/2)2^n$. We might have $\pi - \epsilon'/\delta < 0$, but this is not problematic. Let M be a distribution δ-dense in U that maximizes $\mathrm{E}_M[f]$, and observe that

$$\mathrm{E}_M[f] \;=\; \min\big(|X|/\delta 2^n, 1\big) \;\geq\; \pi - \epsilon'/\delta.$$

We have $U = \delta M + (1-\delta)\widehat{M}$ for some $\widehat{M} \in \mathcal{D}_n$. Let $j \in J$ be such that $\mathrm{E}_{D_{I_j}}[f] = \pi$, and define the distribution $R = \delta D_{I_j} + (1-\delta)\widehat{M}$ so that D_{I_j} is δ-dense in R. Then we have

$$\mathrm{E}_R[f] \;=\; \delta\pi + (1-\delta)\,\mathrm{E}_{\widehat{M}}[f]$$

and

$$\mathrm{E}_U[f] \;=\; \delta\,\mathrm{E}_M[f] + (1-\delta)\,\mathrm{E}_{\widehat{M}}[f] \;\geq\; \delta\pi - \epsilon' + (1-\delta)\,\mathrm{E}_{\widehat{M}}[f] \;=\; \mathrm{E}_R[f] - \epsilon'$$

so f does not positively cover D_{I_j}. This finishes the proof of Claim 4.14. □

4.3.4.3 Setting the Parameters

Define $s = \lfloor 1/4\epsilon \rfloor$ and $t = w$ and $b = \left\lfloor \frac{1}{20t}\sqrt{(1/\epsilon)\log_2(1/\delta)} \right\rfloor$. By Lemma 4.12 there exists an (ℓ, k, s, b)-design $I_1, \ldots, I_\ell$ with $\ell = \lceil k^{b/8} \rceil$ (note that we do have $k \geq 16s^4$). Define p to be such that $\mathrm{Tail}(s, p, 1/2) = \delta/2$.

Claim 4.15. $\frac{1}{2}\sqrt{\epsilon\log_2(1/\delta)} \;\leq\; 1/2-p \;\leq\; 2\sqrt{\epsilon\log_2(1/\delta)} \;\leq\; 1/4.$

Proof. The bound $2\sqrt{\epsilon\log_2(1/\delta)} \leq 1/4$ holds by our assumption $\epsilon \leq 1/64\log_2(1/\delta)$. To prove the upper bound on $1/2-p$, define $p' = 1/2 - 2\sqrt{\epsilon\log_2(1/\delta)} \geq 1/4$. Then we have

$$\mathrm{Tail}\big(s,p',1/2\big) \;\leq\; 2^{-\mathrm{RE}(1/2\|p')s} \;\leq\; 2^{-2(1/2-p')^2 s} \;=\; \delta^{8\epsilon s} \;\leq\; \delta^{8/5} \;<\; \delta/2$$

by Lemma 4.8 and Fact 4.7, and where the penultimate inequality uses $\epsilon \leq 1/20$. Thus $p \geq p'$. To prove the lower bound on $1/2-p$, assume it does not hold. Then we would have the contradiction

$$\begin{aligned}
\delta/2 \;&\geq\; \min\big(\tfrac{1}{256},\ \tfrac{1}{128\sqrt{s}(1/2-p)}\big)\cdot 2^{-\mathrm{RE}(1/2\|p)s}\\
&\geq\; \min\Big(\tfrac{1}{256},\ \tfrac{1}{32\sqrt{\log_2(1/\delta)}}\Big)\cdot 2^{-\mathrm{RE}(1/2\|p)s}\\
&\geq\; \delta^{1/2}\cdot 2^{-\mathrm{RE}(1/2\|p)s}\\
&\geq\; \delta^{1/2}\cdot 2^{-4(1/2-p)^2 s}\\
&\geq\; \delta^{1/2}\cdot 2^{-(1/2-p)^2/\epsilon}\\
&\geq\; \delta^{1/2}\cdot \delta^{1/4}
\end{aligned}$$

where the first line follows by Lemma 4.10 (note that we do have $s \geq 9$), the third line follows by[7] $\delta \leq 2^{-16}$, and the fourth line follows by Fact 4.7. This finishes the proof of Claim 4.15.□

4.3.4.4 The Majority of Majorities

We choose $f_1,\ldots,f_k$ randomly by setting $f_i(x)=1$ with probability p independently for each $i\in[k]$ and each $x\in\{0,1\}^n$.

Claim 4.16. *For every $J\subseteq[\ell]$ of size t and every $x\in\{0,1\}^n$, we have $\Pr_{f_1,\ldots,f_k}[x\in X_{I_J}] \leq \epsilon'/4$.*

Proof. Define $t' = \lceil t/2\rceil$. Note that if $(\mathrm{Maj}\circ\mathrm{Maj}^t\circ f_{I_J})(x)=1$ then there exists a subset $J'\subseteq J$ of size t' such that $(\mathrm{And}\circ\mathrm{Maj}^{t'}\circ f_{I_{J'}})(x)=1$. Thus we have

$$\begin{aligned}
&\Pr_{f_1,\ldots,f_k}\big[(\mathrm{Maj}\circ\mathrm{Maj}^t\circ f_{I_J})(x)=1\big]\\
\leq\;& 2^t\cdot\max_{J'\subseteq J\,:\,|J'|=t'}\Pr_{f_1,\ldots,f_k}\big[(\mathrm{And}\circ\mathrm{Maj}^{t'}\circ f_{I_{J'}})(x)=1\big].
\end{aligned}$$

Consider an arbitrary $J'\subseteq J$ of size t'. Define $m=\big|\bigcup_{j\in J'} I_j\big|$ and notice that since $I_1,\ldots,I_\ell$ is an (ℓ,k,s,b)-design, by inclusion-exclusion we have

$$t's-\tbinom{t'}{2}b \;\leq\; m \;\leq\; t's. \tag{4.1}$$

[7]The existence of w in the statement of Theorem 4.6 actually implies $\delta \leq 2^{-160}$.

Define $s' = \lceil s/2 \rceil$ and $q = 1/2 - t'b/2s$. If $(\text{And} \circ \text{Maj}^{t'} \circ f_{I_{J'}})(x) = 1$ then for each $j \in J'$ we have $\sum_{i \in I_j} f_i(x) \geq s'$ and so by inclusion-exclusion we have

$$\textstyle\sum_{i \in \bigcup_{j \in J'} I_j} f_i(x) \;\geq\; \Big(\sum_{j \in J'} \sum_{i \in I_j} f_i(x)\Big) - \binom{t'}{2} b \;\geq\; t's' - \binom{t'}{2} b \;\geq\; qt's \;\geq\; qm.$$

It follows that

$$\begin{aligned}
\Pr_{f_1,\dots,f_k}\left[(\text{And} \circ \text{Maj}^{t'} \circ f_{I_{J'}})(x) = 1\right] &\leq \Pr_{f_1,\dots,f_k}\left[\textstyle\sum_{i \in \bigcup_{j \in J'} I_j} f_i(x) \geq qm\right] \\
&= \text{Tail}(m, p, q) \\
&\leq 2^{-\operatorname{RE}(q\|p)m} \\
&= \left(2^{-\operatorname{RE}(1/2\|p)s}\right)^{(m/s)\cdot(\operatorname{RE}(q\|p)/\operatorname{RE}(1/2\|p))} \\
&\leq \left(\delta^{1/10}\right)^{(m/s)\cdot(\operatorname{RE}(q\|p)/\operatorname{RE}(1/2\|p))}
\end{aligned}$$

where the third line follows by Lemma 4.8 and the fifth line follows by nonnegativity of RE and

$$2^{-\operatorname{RE}(1/2\|p)s} \;\leq\; 2^{-2(1/2-p)^2 s} \;\leq\; \delta^{\epsilon s/2} \;\leq\; \delta^{1/10}$$

which holds by Fact 4.7, Claim 4.15, and $\epsilon \leq 1/20$. We have

$$m/s \;\geq\; t' - (t')^2 b/2s \;\geq\; t'/2 \;\geq\; t/4 \tag{4.2}$$

by (4.1) and $b \leq s/t'$ (which can be shown using the final inequality in Claim 4.15). We also have $t'b/2s \leq \frac{1}{8}\sqrt{\epsilon \log_2(1/\delta)}$ and thus $q - p \geq \frac{3}{4}(1/2 - p)$ by Claim 4.15. Hence by Fact 4.7 we have

$$\operatorname{RE}(q\|p)/\operatorname{RE}(1/2\|p) \;\geq\; \tfrac{(q-p)^2}{2(1/2-p)^2} \;\geq\; \tfrac{(\frac{3}{4}(1/2-p))^2}{2(1/2-p)^2} \;\geq\; 1/4. \tag{4.3}$$

Using (4.2) and (4.3) we get

$$\Pr_{f_1,\dots,f_k}\left[(\text{And} \circ \text{Maj}^{t'} \circ f_{I_{J'}})(x) = 1\right] \;\leq\; \left(\delta^{1/10}\right)^{(t/4)\cdot(1/4)} \;=\; \delta^{t/160}.$$

We conclude that

$$\Pr_{f_1,\dots,f_k}[x \in X_{I_J}] \;\leq\; 2^t \cdot \delta^{t/160} \;\leq\; \epsilon'/4.$$

This finishes the proof of Claim 4.16. $\square$

4.3.4.5 Putting It All Together

For every $j \in [\ell]$ and every $x \in \{0,1\}^n$, we have $\Pr_{f_1,\dots,f_k}[x \in X_{I_j}] = \text{Tail}(s, p, 1/2) = \delta/2$. Standard relative-error forms of the Chernoff bound give

$$\begin{aligned}
\Pr_{f_1,\dots,f_k}\left[|X_{I_j}| < (\delta/2 - \epsilon'/4)2^n\right] &\leq e^{-2^n(\epsilon')^2/16\delta} \\
\Pr_{f_1,\dots,f_k}\left[|X_{I_j}| > (2\delta/3)2^n\right] &\leq e^{-2^n\delta/54} \\
\Pr_{f_1,\dots,f_k}\left[|X_{I_J}| > (\epsilon'/2)2^n\right] &\leq e^{-2^n\epsilon'/12}
\end{aligned}$$

where the latter holds for each $J \subseteq [\ell]$ of size t, using Claim 4.16. Thus by a union bound we have

$$\begin{aligned}
&\Pr_{f_1,\ldots,f_k}\left[\begin{matrix}(\delta/2 - \epsilon'/4)2^n \le |X_{I_j}| \le (2\delta/3)2^n \text{ for all } j \in [\ell] \text{ and} \\ |X_{I_J}| \le (\epsilon'/2)2^n \text{ for all } J \subseteq [\ell] \text{ of size } t\end{matrix}\right] \\
\ge\; & 1 - \ell \cdot e^{-2^n(\epsilon')^2/16\delta} - \ell \cdot e^{-2^n\delta/54} - \binom{\ell}{t} \cdot e^{-2^n\epsilon'/12} \\
>\; & 0
\end{aligned}$$

since $2^n \ge \frac{t \log_2 k}{\epsilon\delta^2(\epsilon')^2}$. Fix a choice of $f_1, \ldots, f_k$ such that the above event occurs.

For every $J^* \subseteq [\ell]$ of size $2t-1$, there is no $a \in \{0,1\}^\alpha$ such that $\mathrm{Dec}(f_1,\ldots,f_k,a)$ simultaneously covers D_{I_j} for all $j \in J^*$, because otherwise for some $J \subseteq J^*$ of size t, either $\mathrm{Dec}(f_1,\ldots,f_k,a)$ or its complement would simultaneously positively cover D_{I_j} for all $j \in J$, which would contradict Claim 4.14.

So for each $a \in \{0,1\}^\alpha$, the number of $j \in [\ell]$ such that D_{I_j} is covered by $\mathrm{Dec}(f_1,\ldots,f_k,a)$ is at most $2t-2$. This implies that the number of $j \in [\ell]$ for which there exists an $a \in \{0,1\}^\alpha$ such that $\mathrm{Dec}(f_1,\ldots,f_k,a)$ covers D_{I_j} is at most $2^\alpha \cdot (2t-2) < k^{b/8} \le \ell$ since $\alpha \le (b/8)\log_2 k - \log_2 t - 1$. Thus there exists a $j \in [\ell]$ such that D_{I_j} is not covered by $\mathrm{Dec}(f_1,\ldots,f_k,a)$ for any $a \in \{0,1\}^\alpha$. By Claim 4.13, D_{I_j} has no dense model, so Dec is not an $(n,\epsilon,\delta,\epsilon',k,\alpha)$-reduction. This finishes the proof of Theorem 4.6.

Part III

The Problem's Input is Random

Chapter 5

Extractors and Lower Bounds for Locally Samplable Sources

5.1 Introduction

Randomness extraction is the following general problem. Given a sample from an imperfect physical source of randomness, which is modeled as a probability distribution on bit strings of length n, we wish to apply an efficient deterministic algorithm to the sample to produce an output which is almost uniformly distributed (and thus is suitable for use by a randomized algorithm). Of course, to extract randomness from a source, the source needs to "contain" a certain amount of randomness in the first place. It is well established that the most suitable measure of the amount of randomness in a source is its *min-entropy* (a distribution is said to have at least k bits of min-entropy if each outcome occurs with probability at most 2^{-k}). However, even if the source is known to have at least $n-1$ bits of min-entropy, no algorithm can extract even a single bit that is guaranteed to be close to uniformly distributed (see, for example, [250, 228] for proofs of this folklore observation). To deal with this problem, researchers have constructed *seeded extractors* (introduced by [202]), which have access to a short uniformly random seed that is statistically independent of the source and which acts as a catalyst for the extraction process (see [226, 250, 228] for introductions).

However, there is a sense in which seeded extractors are overkill: They are guaranteed to work for completely arbitrary sources that have high enough min-entropy. It is reasonable to assume the physical source of randomness has some limited structure, in which case deterministic (that is, seedless) extraction may become viable. There are several classes of sources for which researchers have constructed good deterministic extractors. One such class is independent sources, where the n bits are partitioned into blocks which are assumed to be statistically independent of each other [60, 74, 27, 49, 28, 211, 227, 208, 29, 207, 210, 215, 240, 167]. Other such classes include so-called bit-fixing sources [59, 154, 94, 209], affine sources [93, 50, 209, 71, 276, 168], polynomial sources [77, 34], and algebraic varieties [76].

Trevisan and Vadhan [246] considered deterministic extractors for the class of sources that are samplable by efficient algorithms given uniform random bits. One may initially be concerned that extracting randomness from such sources is somehow circular or vacuous: We are assuming uniform random bits are used to sample the source, and our goal then is to "undo" the sampling and get uniform random bits back. The point is that this class of sources is just a model for physical sources. This is motivated by the following postulate about the universe: A physical source of randomness is generated by an efficient process in nature, so it is reasonable to model the source as being sampled by an efficient algorithm.

Trevisan and Vadhan constructed extractors for the class of sources samplable by general *time-bounded* algorithms, but their constructions are conditional on (somewhat nonstandard) complexity-theoretic conjectures. It is common in other areas of research, such as proving lower bounds and constructing pseudorandom generators, that proving unconditional limits on the power of time-bounded algorithms is beyond the reach of current techniques. Thus researchers consider more restricted types of algorithms, such as small-space algorithms and bounded-depth circuits, which are combinatorially simple enough for us to prove unconditional results. Hence it is natural to try to construct unconditional deterministic extractors for sources samplable by such restricted algorithms. Kamp et al. [153] succeeded in doing so for small-space samplers with streaming/one-way access to the random input bits.

However, at the time the paper this chapter is based on was written, it was an open problem to construct an unconditional deterministic extractor for sources samplable by polynomial-size constant-depth circuits with unbounded fan-in gates. A basic obstacle is that this requires that input-output pairs of the extractor cannot be sampled by such circuits, and it was not even known how to construct an explicit function with the latter property. For example, although the parity function is known not to have subexponential-size constant-depth circuits [275, 129], input-output pairs can be sampled very efficiently: Just take uniformly random bits $x_1, \ldots, x_n$ and output $x_1, x_1 \oplus x_2, x_2 \oplus x_3, \ldots, x_{n-1} \oplus x_n, x_n$. In independent and concurrent work, Viola [261] has constructed unconditional deterministic extractors for sources samplable by polynomial-size constant-depth circuits with unbounded fan-in gates, which in particular yields an explicit function whose input-output pairs cannot be sampled by such circuits (see Section 5.1.3).

Our goal in this chapter is to expand the frontier of unconditional deterministic randomness extraction for sources with low-complexity samplers. We succeed in constructing extractors for sources samplable by small-depth circuits with *bounded* fan-in gates (which corresponds to the class NC^0 when the depth is constant). This is equivalent to requiring that each output bit of the sampler only depends on a small number of input bits. We call such sources *locally samplable.* Even constructing extractors for sources where each output bit depends on at most one input bit is nontrivial, as such sources are a natural generalization of bit-fixing sources.

As pointed out above, a necessary condition for a function to be an extractor for sources sampled by a class of algorithms is that input-output pairs of the function cannot be sampled by such algorithms. Finding explicit functions with the latter property is tougher than finding explicit functions that are hard to compute, because if a function is easy to compute, then input-output pairs can be obtained by just sampling a random input and then computing the corresponding output. Viola [262] initiated the study of finding explicit boolean functions whose input-output pairs are hard to sample for low-complexity samplers (specifically, local samplers). Another contribution of this chapter is an application of our extractor result to obtain an improvement of Viola's result.

5.1.1 Results

We first give the formal definitions of extractors and locally samplable sources.

A distribution on a finite set S is said to have *min-entropy* at least k if each element of S occurs with probability at most 2^{-k}. The *statistical distance* between two distributions D_1 and D_2 on a finite set S is defined to be $\|D_1 - D_2\| = \max_{T \subseteq S} \big|\Pr_{D_1}[T] - \Pr_{D_2}[T]\big|$. If $\|D_1 - D_2\| \leq \epsilon$ then we also say D_1 and D_2 are ϵ-close. If $f : S \to S'$ and D is a distribution on S, then we let $f(D)$ denote the distribution on S' obtained by drawing a sample from D and applying f to it. When we mention a distribution multiple times in an expression, all instantiations refer to a single sample from the distribution; for example, $\big(D, f(D)\big)$ denotes the distribution obtained by sampling $w \sim D$ and outputting the pair $\big(w, f(w)\big)$. We use U_n to denote the uniform distribution on $\{0,1\}^n$. If $\mathcal{C}$ is a class of distributions on $\{0,1\}^n$, then a function $\text{Ext} : \{0,1\}^n \to \{0,1\}^m$ is called a *(k,ϵ)-extractor for $\mathcal{C}$* if for every distribution $D \in \mathcal{C}$ with min-entropy at least k, $\big\|\text{Ext}(D) - U_m\big\| \leq \epsilon$. Informally, when we say an extractor is *explicit* we mean that a uniform polynomial-time deterministic algorithm with the desired behavior is exhibited.

We define a *d-local sampler* to be a function $f : \{0,1\}^r \to \{0,1\}^n$ such that each output bit depends on at most d input bits. In other words, for every $j \in \{1, \ldots, n\}$ there exists a subset $I_j \subseteq \{1, \ldots, r\}$ with $|I_j| \leq d$ and a function $f_j : \{0,1\}^{|I_j|} \to \{0,1\}$ such that the j^{th} output bit of f is obtained by evaluating f_j on the input bits indexed by I_j. The output distribution of the sampler is $f(U_r)$. We say a distribution D on $\{0,1\}^n$ is a *d-local source* if there exists a d-local sampler (with any input length r) whose output distribution is D.

We have three main theorems. Our first main theorem gives an extractor for locally samplable sources.

Theorem 5.1. *For every constant $\gamma > 0$ there exists a constant $\beta > 0$ such that there exists an explicit (k,ϵ)-extractor for the class of d-local sources with output length $m = k^2/8nd$ and error $\epsilon = 2^{-n^\beta}$, provided $k \geq n^{2/3+\gamma}$ and $d \leq \beta \log n$.*

Our second main theorem gives an extractor for 1-local sources (which generalize bit-fixing sources), achieving better min-entropy requirement and better output length than Theorem 5.1.

Theorem 5.2. *For every constant $\gamma > 0$ there exists a constant $\beta > 0$ such that there exists an explicit (k,ϵ)-extractor for the class of 1-local sources with output length $m = k - o(k)$ and error $\epsilon = 2^{-n^\beta}$, provided $k \geq n^{1/2+\gamma}$.*

Our third main theorem concerns the problem of finding explicit functions whose input-output pairs are hard to sample, as discussed in the paragraph right before Section 5.1.1.

Theorem 5.3. *There exists a universal constant $\beta > 0$ and an explicit function $F : \{0,1\}^n \to \{0,1\}$ such that for every d-local source D on $\{0,1\}^{n+1}$ with $d \leq \beta \log n$, $\big\|D - \big(U_n, F(U_n)\big)\big\| \geq 1/2 - 2^{-n^{\beta}}$.*

5.1.2 Techniques

We now discuss the techniques we use to prove these three theorems. The proof of Theorem 5.1 has three steps.

The first step is to construct a certain extractor for 1-local sources (which in particular yields Theorem 5.2). To do this, we observe that extractors for so-called low-weight affine sources also work for 1-local sources. Rao [209] constructed an extractor for low-weight affine sources. Using Rao's extractor off-the-shelf would lead to a weaker version of Theorem 5.1 with min-entropy requirement $k \geq n^{1-\gamma}$ for *some* constant $\gamma > 0$. To improve the min-entropy requirement, we construct an improved extractor for low-weight affine sources by building on [209]. While Rao's extractor handles affine sources of min-entropy at least k and weight at most k^{γ} for some constant $\gamma > 0$, our improvement handles sources with weight at most $k^{1-\gamma}$ for *any* constant $\gamma > 0$. The key ingredient in our improvement is the strong condenser of Guruswami, Umans, and Vadhan [121]. We present this step in Section 5.3 and Section 5.7.

The second step is to show that extractors for 1-local sources also work for $o(\log n)$-local sources. To do this, we relate the problem to a concept we call *superindependent matchings* in bipartite graphs, and we prove a combinatorial lemma about the existence of such matchings. We present this step in Section 5.4.

The third step is to increase the output length of the extractor using the technique of "obtaining an independent seed" introduced by Gabizon et al. [94] (see also [227]). Combining step 1 and step 2 yields an extractor with output length $\Omega(k^2/nd^3 2^d)$. To increase the output length to $\Omega(k^2/nd)$, we adapt the technique from [94]. A key ingredient in our argument is a lemma due to Vadhan [249], which is a strengthened version of a classic lemma due to Nisan and Zuckerman [202]. While the result of [94] achieves output length $k - o(k)$ for bit-fixing sources, we lose a factor of $\Omega(k/n)$ in the output length due to the way we use Vadhan's lemma, and we lose another factor of $\Omega(1/d)$ since conditioning on p bits of the output of a d-local sampler could cause a loss of pd bits of min-entropy. We present this step in Section 5.5.

Viola [262] proved a version of Theorem 5.3 where the statistical distance lower bound is only $1/2 - O(1/\log n)$, and the d-local sampler is restricted to use at most $n + n^{1-\delta}$ random bits for any constant $\delta > 0$. His function F is what he calls "majority mod p". Using a different function F (namely, any bit of the extractor underlying Theorem 5.1), we simultaneously improve the lower bound to $1/2 - 2^{-n^{\Omega(1)}}$ and eliminate the restriction on the number of random bits. Our proof of Theorem 5.3 uses ideas similar to Viola's, but is actually somewhat simpler given the extraction property of F. In [262], Viola also showed that for symmetric functions F, one cannot hope to get such a strong lower bound for samplers that are polynomial-size constant-depth circuits. Our extractor function F is not symmetric. We present the proof of Theorem 5.3 in Section 5.6.[1]

5.1.3 Concurrent Work

In independent and concurrent work, Viola [261] obtained extractors for d-local sources with $d \leq n^{o(1)}$ and for sources sampled by polynomial-size constant-depth circuits. The high level idea behind the extractor is the same as in our work: Show that the given source is close to a convex combination of 1-local sources, and use the extractor in [209]. However, the proofs in [261] are much more involved than in this chapter. For d-local sources with $d \leq n^{o(1)}$, Viola requires min-entropy $k \geq n^{3/4+\gamma}$ (for any constant $\gamma > 0$) and achieves output length $m = \widetilde{\Omega}(k^3/n^2d^3)$ and error $\epsilon = 2^{-n^{\Omega(1)}}$ (though the output length can be improved to $\Omega(k^2/nd)$ using the technique we present in Section 5.5 based on [94]). When $d \leq o(\log n)$ he obtains a result similar to our Theorem 5.1 but with worse output length: He requires min-entropy $k \geq n^{2/3+\gamma}$ and achieves output length $m = \Omega(k^2/nd^2 2^d)$ and error $\epsilon = 2^{-n^{\Omega(1)}}$. For sources sampled by polynomial-size constant-depth circuits, he requires min-entropy $k \geq n^{2/3+\gamma}$ and achieves output length $m = \Omega(k^2/n^{1+\Omega(1)})$ and error $\epsilon = n^{-\omega(1)}$.

5.1.4 Previous Work on the Power of Locally Computable Functions

There has been a substantial amount of work on whether various complexity-theoretic and cryptographic objects can be computed locally. Several works [80, 175, 249, 13, 279, 68] have studied the problem of constructing locally computable seeded extractors (that is, the extractor itself is locally computable, as opposed to our setting where the sampler for the source is locally computable). A variety of works [65, 193, 12, 13, 145, 10, 11] have given positive and negative results on the existence of locally computable pseudorandom generators. Several works [130, 99, 12, 62, 43] have studied the possibility of locally computable one-way functions. Goldwasser et al. [106] gave positive and negative results on interactive proof systems with locally computable verifiers. Arora et al. [18] show that the adjacency list of certain logarithmic-degree expander graphs can be computed with constant locality, and they ask whether the same holds for constant-degree expander graphs.

5.2 Preliminaries

In this chapter we work with bipartite graphs $G = (L, R, E)$, where L, R are disjoint finite sets (the left and right nodes) and E is a set of unordered pairs where one element comes from L and the other from R. The distance between two nodes is the number of edges on a shortest path between them.

[1]We also mention in passing that Lovett and Viola [173] exhibited an explicit distribution on $\{0,1\}^n$ that cannot be sampled within statistical distance $1 - 1/n^{\Omega(1)}$ by polynomial-size constant-depth circuits, namely the uniform distribution over the codewords of any asymptotically good error-correcting code. However, this distribution is not of the same form as sampling input-output pairs.

To every function $f : \{0,1\}^r \to \{0,1\}^n$ we associate a bipartite graph $G = (L, R, E)$ where $L = \{1, \ldots, r\} \times \{\text{in}\}$, $R = \{1, \ldots, n\} \times \{\text{out}\}$, and $\big\{(i, \text{in}), (j, \text{out})\big\} \in E$ if and only if the j^{th} output bit of f depends on the i^{th} input bit of f (that is, for some setting of all input bits except the i^{th}, the j^{th} output bit equals the i^{th} input bit or its complement). Note that we include no unnecessary edges, and the graph is unique. We use $I_j \times \{\text{in}\}$ to denote the set of neighbors of (j, out) and $J_i \times \{\text{out}\}$ to denote the set of neighbors of (i, in). Observe that if $f(U_r)$ has min-entropy at least k, then there are at least k non-isolated nodes in L, and in particular $r \geq k$.

We say f is a *d-local sampler* if each node in R has degree at most d, and we say a distribution on $\{0,1\}^n$ is a *d-local source* if it equals $f(U_r)$ for some d-local sampler f (with any input length r). We say f is a *(d, c)-local sampler* if each node in R has degree at most d and each node in L has degree at most c, and we say a distribution on $\{0,1\}^n$ is a *(d, c)-local source* if it equals $f(U_r)$ for some (d, c)-local sampler f (with any input length r).

Suppose Y is a finite set of indices, $(p_y)_{y \in Y}$ is a distribution on Y, and for each $y \in Y$, D_y is a distribution on a finite set S. Then the *convex combination* $\sum_{y \in Y} p_y D_y$ is defined to be the distribution on S obtained by sampling y according to $(p_y)_{y \in Y}$, then outputting a sample from D_y.

Lemma 5.4. *Suppose* $\text{Ext} : \{0,1\}^n \to \{0,1\}^m$ *is any function and* $D = \sum_{y \in Y} p_y D_y$ *is a distribution on* $\{0,1\}^n$. *Then for every* $\epsilon \geq 0$,

$$\big\|\text{Ext}(D) - U_m\big\| \;\leq\; \epsilon + \Pr_{y \sim (p_y)_{y \in Y}}\Big[\big\|\text{Ext}(D_y) - U_m\big\| > \epsilon\Big].$$

Proof. First, observe that $\text{Ext}(D) = \sum_{y \in Y} p_y \,\text{Ext}(D_y)$. Now for every $T \subseteq \{0,1\}^n$ we have

$$\begin{aligned}
& \big|\Pr_{\text{Ext}(D)}[T] - \Pr_{U_m}[T]\big| \\
= \;& \Big|\textstyle\sum_{y \in Y} p_y \big(\Pr_{\text{Ext}(D_y)}[T] - \Pr_{U_m}[T]\big)\Big| \\
\leq \;& \textstyle\sum_{y \in Y} p_y \big|\Pr_{\text{Ext}(D_y)}[T] - \Pr_{U_m}[T]\big| \\
\leq \;& \epsilon \cdot \Pr_{y \sim (p_y)_{y \in Y}}\Big[\big|\Pr_{\text{Ext}(D_y)}[T] - \Pr_{U_m}[T]\big| \leq \epsilon\Big] + 1 \cdot \Pr_{y \sim (p_y)_{y \in Y}}\Big[\big\|\text{Ext}(D_y) - U_m\big\| > \epsilon\Big] \\
\leq \;& \epsilon + \Pr_{y \sim (p_y)_{y \in Y}}\Big[\big\|\text{Ext}(D_y) - U_m\big\| > \epsilon\Big]
\end{aligned}$$

which gives the desired bound on $\big\|\text{Ext}(D) - U_m\big\|$. $\square$

Corollary 5.5. *Suppose every distribution in* $\mathcal{C}$ *with min-entropy at least* k *can be written as a convex combination* $\sum_{y \in Y} p_y D_y$ *where*

$$\Pr_{y \sim (p_y)_{y \in Y}}\big[D_y \textit{ is in } \mathcal{C}' \textit{ and has min-entropy at least } k'\big] \;\geq\; 1 - \delta.$$

Then every (k', ϵ')*-extractor for* $\mathcal{C}'$ *is also a* (k, ϵ)*-extractor for* $\mathcal{C}$ *where* $\epsilon = \epsilon' + \delta$.

Corollary 5.6. *Suppose every distribution in* $\mathcal{C}$ *with min-entropy at least* k *is a convex combination of distributions in* $\mathcal{C}'$ *with min-entropy at least* k'. *Then every* (k', ϵ)*-extractor for* $\mathcal{C}'$ *is also a* (k, ϵ)*-extractor for* $\mathcal{C}$.

Lemma 5.7. *Every d-local source with min-entropy at least k is a convex combination of (d,c)-local sources with min-entropy at least $k - nd/c$.*

Proof. Consider an arbitrary d-local sampler $f : \{0,1\}^r \to \{0,1\}^n$ whose output distribution has min-entropy at least k, and let $G = (L, R, E)$ be the associated bipartite graph. Since $|E| \le nd$, there are at most nd/c nodes in L with degree greater than c; without loss of generality these nodes are $\{r - \ell + 1, \ldots, r\} \times \{\text{in}\}$ for some $\ell \le nd/c$. For each string $y \in \{0,1\}^\ell$, define $f_y : \{0,1\}^{r-\ell} \to \{0,1\}^n$ as $f_y(x) = f(x, y)$ (hard-wiring the last ℓ bits to y). Then $f(U_r) = \sum_{y \in \{0,1\}^\ell} \frac{1}{2^\ell} f_y(U_{r-\ell})$. Moreover, each $f_y(U_{r-\ell})$ is a (d,c)-local source with min-entropy at least $k - nd/c$, since if some $z \in \{0,1\}^n$ and $y^* \in \{0,1\}^\ell$ satisfied $\Pr_{x \sim U_{r-\ell}}\big[f_{y^*}(x) = z\big] > 1/2^{k-nd/c}$ then we would have

$$\Pr_{x \sim U_{r-\ell}, y \sim U_\ell}\big[f(x,y) = z\big] \;\ge\; \Pr_{y \sim U_\ell}[y = y^*] \cdot \Pr_{x \sim U_{r-\ell}}\big[f(x, y^*) = z\big] \;>\; \tfrac{1}{2^\ell} \cdot \tfrac{1}{2^{k-nd/c}} \;\ge\; 1/2^k$$

contradicting that $f(U_r)$ has min-entropy at least k. □

In this chapter we also make use of seeded extractors. A function $\text{Ext} : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ is called a *seeded* (k,ϵ)*-extractor* if for every distribution D on $\{0,1\}^n$ with min-entropy at least k, $\big\|\text{Ext}(D, U_t) - U_m\big\| \le \epsilon$ where U_t is independent of D. We say Ext is a *strong seeded* (k,ϵ)*-extractor* if for every distribution D on $\{0,1\}^n$ with min-entropy at least k,[2]

$$\Pr_{y \sim U_t}\Big[\big\|\text{Ext}(D, y) - U_m\big\| \le \epsilon\Big] \;\ge\; 1 - \epsilon.$$

We say Ext is *linear* if for every seed $y \in \{0,1\}^t$, the function $\text{Ext}(\cdot, y) : \{0,1\}^n \to \{0,1\}^m$ is linear over $\mathbb{F}_2$, where $\mathbb{F}_q$ denotes the finite field of size q.

If $z \in \{0,1\}^n$ and $J \subseteq \{1, \ldots, n\}$, then we let $z|_J \in \{0,1\}^{|J|}$ denote the substring of z indexed by the coordinates in J. If D is a distribution on $\{0,1\}^n$ and $J \subseteq \{1, \ldots, n\}$, then we let $D|_J$ denote the marginal distribution on the coordinates in J.

Finally, all logarithms in this chapter are base 2.

5.3 1-Local Sources

An *affine source* is a distribution on $\{0,1\}^n$ which is uniform over an affine subspace (where $\{0,1\}^n$ is viewed as a vector space over $\mathbb{F}_2$). If the subspace has dimension k then it has size 2^k and hence the source has min-entropy k. The distribution can be sampled by picking $x_1, \ldots, x_k \in \{0,1\}$ uniformly at random and outputting $z_0 + x_1 z_1 + \cdots + x_k z_k$ where $z_0 \in \{0,1\}^n$ is a shift vector and $z_1, \ldots, z_k \in \{0,1\}^n$ are a basis of the associated linear subspace. The source is said to be a *weight-c affine source* if there exist basis vectors $z_1, \ldots, z_k$ each of which has Hamming weight at most c.

Observation 5.8. *Every $(1,c)$-local source is also a weight-c affine source.*

[2]According to this definition, every strong seeded (k,ϵ)-extractor is also a seeded $(k, 2\epsilon)$-extractor.

Proof. Consider an arbitrary $(1,c)$-local sampler $f : \{0,1\}^k \to \{0,1\}^n$ and assume without loss of generality that there are no isolated nodes on the left side of the associated bipartite graph. For each $i \in \{1,\ldots,k\}$, let $J_i \times \{\text{out}\}$ be the set of neighbors of (i,in), and let $1_{J_i} \in \{0,1\}^n$ be the characteristic vector of this set. For each $i \in \{1,\ldots,k\}$ we have $|J_i| \le c$ and hence 1_{J_i} has Hamming weight at most c (since f is a $(1,c)$-local sampler). It is straightforward to verify that the output distribution of f is sampled by picking $x_1,\ldots,x_k \in \{0,1\}$ uniformly at random and outputting $f(0^k)+x_1 1_{J_1}+\cdots+x_k 1_{J_k}$. Moreover, the vectors 1_{J_i} are linearly independent. $\square$

Rao [209] (building on [208]) constructed extractors for low-weight affine sources.

Theorem 5.9 ([209]). *There exist universal constants $C,\gamma > 0$ such that for all $k \ge \log^C n$ there exists an explicit $(k, 2^{-k^{\Omega(1)}})$-extractor with output length $m = k - o(k)$ for the class of weight-k^γ affine (and in particular, $(1,k^\gamma)$-local) sources.*

We improve Rao's result to obtain the following theorem, which we prove in Section 5.7.

Theorem 5.10. *There exists a universal constant $C > 0$ such that for every constant $\gamma > 0$ and all $k \ge \log^{C/\gamma} n$ there exists an explicit $(k, 2^{-k^{\Omega(1)}})$-extractor with output length $m = k - o(k)$ for the class of weight-$k^{1-\gamma}$ affine (and in particular, $(1,k^{1-\gamma})$-local) sources.*

We now explain how Theorem 5.2 follows from Theorem 5.10, Lemma 5.7, and Corollary 5.6. We first note the following immediate corollary of Theorem 5.10.

Corollary 5.11. *For every constant $\gamma > 0$ there exists a constant $\beta > 0$ such that for all $k \ge n^{1/2+\gamma}$ there exists an explicit $(k, 2^{-n^\beta})$-extractor with output length $m = k - o(k)$ for the class of weight-$n^{1/2}$ affine (and in particular, $(1,n^{1/2})$-local) sources.*

Lemma 5.7 implies that every 1-local source with min-entropy at least $k \ge n^{1/2+\gamma}$ is a convex combination of $(1,n^{1/2})$-local sources with min-entropy at least $k - n^{1/2} \ge k - o(k)$. Theorem 5.2 then follows from Corollary 5.6 and Corollary 5.11.

Bourgain [50], Yehudayoff [276], and Li [168] constructed extractors for linear min-entropy affine sources (of arbitrary weight), achieving better error but worse output length than Theorem 5.10.

Theorem 5.12 ([50]). *For every constant $\delta > 0$ there exists an explicit $(\delta n, 2^{-\Omega(n)})$-extractor with output length $m = \Omega(n)$ for the class of affine (and in particular, 1-local) sources.*

Theorem 5.12 can be used to improve the error in Theorem 5.1 and Theorem 5.2 when $k \ge \Omega(n)$ and $d \le O(1)$. We omit the details, so as to avoid having a laundry list of results.

5.4 d-Local Sources

The following theorem shows that to get extractors for d-local sources, it suffices to construct extractors for 1-local sources.

Theorem 5.13. *Every (k', ϵ')-extractor for $(1, 2nd/k)$-local sources is also a (k, ϵ)-extractor for d-local sources, where $k' = k^2/4nd^3 2^d$ and $\epsilon = \epsilon' + e^{-k'/4}$.*

Assuming $k \geq n^{2/3+\gamma}$ (for constant $\gamma > 0$) and $d \leq \beta \log n$ (for small enough constant $\beta > 0$) in Theorem 5.13, we find that it suffices to have a (k', ϵ')-extractor for $(1, c)$-local sources where $k' \geq n^{1/3+\gamma}$ and $c = 2nd/k \leq n^{1/3} \leq (k')^{1-\gamma}$. Such an extractor is given by Theorem 5.10, with error $\epsilon' = 2^{-n^{\Omega(1)}}$ (and thus $\epsilon = \epsilon' + e^{-k'/4} \leq 2^{-n^{\Omega(1)}}$). This already yields a version of Theorem 5.1 with output length $k' - o(k') = \Omega(k^2/nd^3 2^d)$.

As a corollary to Theorem 5.13, we also find that if we could construct an explicit extractor for 1-local sources with min-entropy at least n^γ for arbitrarily small constants $\gamma > 0$ (with output length $m \geq 1$ and error $\epsilon \leq 1/2$, say) then we would get explicit extractors for $o(\log n)$-local sources with min-entropy at least $n^{1/2+\gamma}$ for arbitrarily small constants $\gamma > 0$. This $n^{1/2}$ min-entropy barrier is common in extractor constructions.

5.4.1 Superindependent Matchings

We first prove a combinatorial lemma that is needed for the proof of Theorem 5.13.

Definition 5.14. *Given a bipartite graph $G = (L, R, E)$, we say a set of edges $M \subseteq E$ is a* superindependent matching *if there is no path of length at most two in G from an endpoint of an edge in M to an endpoint of a different edge in M.*

Lemma 5.15. *Suppose $G = (L, R, E)$ is a bipartite graph with no isolated nodes and such that each node in L has degree at most c and each node in R has degree at most d. Then G has a superindependent matching of size at least $|L|/d^2c$.*

Proof. Let M be a largest superindependent matching in G, and suppose for contradiction that $|M| < |L|/d^2c$. Note that for each node in R, the number of nodes in L within distance three in G is at most $d\big(1 + (c-1)(d-1)\big) \leq d^2c$. Thus the number of nodes in L within distance three of the right endpoints of edges in M is at most $|M| \cdot d^2c < |L|$. Hence there exists a node $u \in L$ at distance greater than three from the right endpoint of every edge in M. Since G has no isolated nodes, there exists a node $v \in R$ such that $\{u, v\} \in E$. Note that there is no path of length at most two from either u or v to an endpoint of an edge in M, since otherwise a simple case analysis would show that u is within distance three of the right endpoint of an edge in M. Thus $M \cup \{\{u, v\}\}$ is a superindependent matching, contradicting the maximality of M. □

5.4.2 Proof of Theorem 5.13

Suppose $\mathrm{Ext} : \{0,1\}^n \to \{0,1\}^m$ is a (k', ϵ')-extractor for $(1, 2nd/k)$-local sources. By Corollary 5.6 and Lemma 5.7 it suffices to show that Ext is a $(k/2, \epsilon)$-extractor for (d, c)-local sources where $c = 2nd/k$. The plan is to show that every (d, c)-local source with min-entropy at least $k/2$ is a convex combination of $(1, c)$-local sources most of which have min-entropy at least k', and then apply Corollary 5.5.

So consider an arbitrary (d, c)-local sampler $f : \{0,1\}^r \to \{0,1\}^n$ whose output distribution has min-entropy at least $k/2$, and let $G = (L, R, E)$ be the associated bipartite graph. If we obtain $\widetilde{G}$ from G by removing any isolated nodes, then $\widetilde{G}$ still has at least $k/2$ nodes on its left side. Applying Lemma 5.15 to $\widetilde{G}$ tells us that G has a superindependent matching M of size at least $k/(2d^2c)$. Let $\ell = |M|$, and without loss of generality assume that the left endpoints of M are $L' = \{1, \ldots, \ell\} \times \{\mathrm{in}\}$. We write inputs to f as (x, y) where $x \in \{0,1\}^\ell$ and $y \in \{0,1\}^{r-\ell}$. Since M is superindependent, each node in R is adjacent to at most one node in L'. Thus if we define $f_y : \{0,1\}^\ell \to \{0,1\}^n$ as $f_y(x) = f(x, y)$ (hard-wiring the last $r - \ell$ input bits to y) then for each y, f_y is a $(1, c)$-local sampler. Observe that $f(U_r) = \sum_{y \in \{0,1\}^{r-\ell}} \frac{1}{2^{r-\ell}} f_y(U_\ell)$.

Let $G_y = (L', R, E_y)$ denote the bipartite graph associated with f_y. As implied by the proof of Observation 5.8, the min-entropy of $f_y(U_\ell)$ is the number of nodes in L' that are non-isolated in G_y. Although each node in L' is non-isolated in G (since $M \subseteq E$), edges incident to L' may disappear when we hard-wire y. We claim that with high probability over y, plenty of nodes in L' are still non-isolated in G_y and hence $f_y(U_\ell)$ has high min-entropy. For $i \in \{1, \ldots, \ell\}$ let $(j_i, \mathrm{out}) \in R$ be the neighbor of (i, in) in M, and let $I_{j_i} \times \{\mathrm{in}\}$ be the set of neighbors of (j_i, out) in G. Since the j_i^{th} output bit of f depends on the i^{th} input bit, there exists a string $w_i \in \{0,1\}^{|I_{j_i}|-1}$ such that hard-wiring the input bits corresponding to $I_{j_i} \backslash \{i\}$ to w_i leaves the edge $\big\{(i, \mathrm{in}), (j_i, \mathrm{out})\big\}$ in place, and in particular ensures that (i, in) is non-isolated. Since M is superindependent, the sets I_{j_i} for $i \in \{1, \ldots, \ell\}$ are pairwise disjoint and in particular, each $I_{j_i} \backslash \{i\} \subseteq \{\ell + 1, \ldots, r\}$. We assume the bits of y are indexed starting at $\ell + 1$, so for example $y|_{\{\ell+1\}}$ is the first bit of y. By the disjointness, we find that the events $y|_{I_{j_i} \backslash \{i\}} = w_i$ (for $i \in \{1, \ldots, \ell\}$) are fully independent over $y \sim U_{r-\ell}$. Moreover, each of these events occurs with probability at least $1/2^{d-1}$ since $|w_i| \leq d - 1$. Thus we have

$$\begin{aligned}
& \Pr_{y \sim U_{r-\ell}}\big[f_y(U_\ell) \text{ does not have min-entropy at least } k'\big] \\
= \;& \Pr_{y \sim U_{r-\ell}}\Big[\big|\big\{i \in \{1, \ldots, \ell\} \;:\; (i, \mathrm{in}) \text{ is non-isolated in } G_y\big\}\big| < k'\Big] \\
\leq \;& \Pr_{y \sim U_{r-\ell}}\Big[\big|\big\{i \in \{1, \ldots, \ell\} \;:\; y|_{I_{j_i} \backslash \{i\}} = w_i\big\}\big| < k'\Big] \\
\leq \;& e^{-k/8d^2c2^d}
\end{aligned}$$

by a standard Chernoff bound.

To summarize, we have shown that every (d, c)-local source with min-entropy at least $k/2$ is a uniform convex combination of $(1, c)$-local sources, at most $e^{-k/8d^2c2^d}$ fraction of which do not have min-entropy at least k'. It now follows from Corollary 5.5 that Ext is a $(k/2, \epsilon)$-extractor for (d, c)-local sources. This finishes the proof of Theorem 5.13.

5.5 Increasing the Output Length

Combining the results from Section 5.3 and Section 5.4 yields an extractor for d-local sources with output length $\Omega(k^2/nd^3 2^d)$, provided $d \le o(\log n)$ and the min-entropy k is at least $n^{2/3+\gamma}$. In this section we show how to improve the output length to $\Omega(k^2/nd)$, which is a significant improvement when $k \ge \Omega(n)$ and d is large. We present the general method in Section 5.5.1, and then we apply the general method to obtain Theorem 5.1 in Section 5.5.2.

5.5.1 The General Method

We now present our general theorem on increasing the output length of extractors for d-local sources (Theorem 5.19 below), which uses the technique of "obtaining an independent seed". As in [94], the strategy is to take the output of a deterministic extractor and use part of it to sample a set of coordinates of the source, which are then plugged into a seeded extractor, using the other part of the deterministic extractor's output as the seed. The key property of d-local sources that enables us to adapt the technique from [94] is that conditioning on any p bits of the source gives a convex combination of d-local sources that lose at most pd in the min-entropy.

A key ingredient (which was not used in [94]) is a fundamental lemma of Nisan and Zuckerman [202], which roughly says that if we sample the coordinates appropriately, then the min-entropy rate of the marginal distribution on those coordinates is almost as high as the min-entropy rate of the whole source.[3] However, the original Nisan-Zuckerman lemma loses a logarithmic factor in the min-entropy rate. We use a strengthened version of the lemma, due to Vadhan [249], which only loses a constant factor.

We use $\binom{\{1,\ldots,n\}}{p}$ to denote the set of subsets of $\{1,\ldots,n\}$ of size p.

Definition 5.16. *We say* Samp : $\{0,1\}^s \to \binom{\{1,\ldots,n\}}{p}$ *is a* (μ,η)-sampler *if for every* $g : \{1,\ldots,n\} \to [0,1]$ *with* $\frac{1}{n}\sum_{j=1}^n g(j) \ge \mu$ *it holds that* $\Pr_{\sigma\sim U_s}\left[\frac{1}{p}\sum_{j\in\mathrm{Samp}(\sigma)} g(j) < \mu/2\right] \le \eta$.

Lemma 5.17 ([249]). *There exists a universal constant $\alpha > 0$ such that the following holds. Suppose* Samp : $\{0,1\}^s \to \binom{\{1,\ldots,n\}}{p}$ *is a $\big(k/2n\log(4n/k),\eta\big)$-sampler and D is a distribution on $\{0,1\}^n$ with min-entropy at least k. Then with probability at least $1-\sqrt{\eta+2^{-\alpha k}}$ over $\sigma \sim U_s$ it holds that $D|_{\mathrm{Samp}(\sigma)}$ is $\sqrt{\eta+2^{-\alpha k}}$-close to a distribution with min-entropy at least $pk/4n$.*

We also need the following lemma from [94], which we state in a slightly nonstandard way for convenience when we apply the lemma.

Lemma 5.18 ([94]). *Consider any distribution on $\{0,1\}^{s_1}\times\{0,1\}^{s_2}\times\{0,1\}^{s_3}$ which is ϵ'-close to uniform, and suppose σ is in the support of the marginal distribution on the second coordinate. Then the marginal distribution on the first and third coordinates, conditioned on the second coordinate being σ, is $(\epsilon' 2^{s_2+1})$-close to uniform.*

Ingredients:

$\mathrm{Ext}' : \{0,1\}^n \to \{0,1\}^{m'}$

$\mathrm{Ext}'_1 : \{0,1\}^n \to \{0,1\}^s$ is the first s bits of Ext'

$\mathrm{Ext}'_2 : \{0,1\}^n \to \{0,1\}^{m'-s}$ is the last $m'-s$ bits of Ext'

$\mathrm{Samp} : \{0,1\}^s \to \binom{\{1,\ldots,n\}}{p}$

$\mathrm{SExt} : \{0,1\}^p \times \{0,1\}^{m'-s} \to \{0,1\}^m$

Result:

$\mathrm{Ext} : \{0,1\}^n \to \{0,1\}^m$ defined as $\mathrm{Ext}(z) = \mathrm{SExt}\big(z|_{\mathrm{Samp}(\mathrm{Ext}'_1(z))}, \mathrm{Ext}'_2(z)\big)$

Figure 5.1: Increasing the output length of an extractor for d-local sources

We now present the general theorem on increasing the output length.

Theorem 5.19. *Consider the construction in Figure 5.1, and let α be as in Lemma 5.17. Suppose* Ext' *is a (k', ϵ')-extractor for d-local sources,* Samp *is a $\big(k/2n\log(4n/k), \eta\big)$-sampler, and* SExt *is a seeded $(pk/4n, \epsilon'')$-extractor. Then* Ext *is a (k, ϵ)-extractor for d-local sources, where $k = k' + pd$ and $\epsilon = \epsilon'(2^{s+1}+1) + 2\sqrt{\eta + 2^{-\alpha k}} + \epsilon''$.*

Proof. Consider an arbitrary d-local sampler $f : \{0,1\}^r \to \{0,1\}^n$ whose output distribution has min-entropy at least k, and let $G = (L, R, E)$ be the associated bipartite graph. Our goal is to show that $\big\|\mathrm{Ext}\big(f(U_r)\big) - U_m\big\| \le \epsilon$.

Let us call $\sigma \in \{0,1\}^s$ *good* if $f(U_r)|_{\mathrm{Samp}(\sigma)}$ is $\sqrt{\eta + 2^{-\alpha k}}$-close to a distribution with min-entropy at least $pk/4n$, and *bad* otherwise. For each σ we let $U_r^{(\sigma)}$ be the uniform distribution over $w \in \{0,1\}^r$ such that $\mathrm{Ext}'_1\big(f(w)\big) = \sigma$.[4]

Claim 5.20. *For each good σ, $\big\|\mathrm{Ext}\big(f(U_r^{(\sigma)})\big) - U_m\big\| \le \epsilon' 2^{s+1} + \sqrt{\eta + 2^{-\alpha k}} + \epsilon''$.*

Assuming Claim 5.20, we can prove the theorem as follows. Observe that

$$f(U_r) \;=\; \textstyle\sum_{\sigma \in \{0,1\}^s} \Pr_{w \sim U_r}\big[\mathrm{Ext}'_1\big(f(w)\big) = \sigma\big] f(U_r^{(\sigma)}).$$

Then using the shorthand $\epsilon''' = \epsilon' 2^{s+1} + \sqrt{\eta + 2^{-\alpha k}} + \epsilon''$ we have

$$\begin{aligned}
\big\|\mathrm{Ext}\big(f(U_r)\big) - U_m\big\| \;&\le\; \epsilon''' + \Pr_{w \sim U_r}\Big[\big\|\mathrm{Ext}\big(f(U_r^{(\sigma)})\big) - U_m\big\| > \epsilon''' \text{ where } \sigma = \mathrm{Ext}'_1\big(f(w)\big)\Big] \\
&\le\; \epsilon''' + \Pr_{w \sim U_r}\big[\mathrm{Ext}'_1\big(f(w)\big) \text{ is bad}\big] \\
&\le\; \epsilon''' + \epsilon' + \Pr_{\sigma \sim U_s}[\sigma \text{ is bad}] \\
&\le\; \epsilon''' + \epsilon' + \sqrt{\eta + 2^{-\alpha k}} \\
&=\; \epsilon
\end{aligned}$$

[3]Min-entropy rate just means the min-entropy divided by the length of the source.

[4]Formally, we only consider σ's in the support of $\mathrm{Ext}'_1\big(f(U_r)\big)$.

where the first line follows by Lemma 5.4, the second line follows by Claim 5.20, the third line follows by $\big\|\mathrm{Ext}'_1\big(f(U_r)\big) - U_s\big\| \le \epsilon'$ (since $f(U_r)$ is a d-local source with min-entropy at least $k \ge k'$), and the fourth line follows by Lemma 5.17.

It remains to prove Claim 5.20. Consider an arbitrary fixed good $\sigma \in \{0,1\}^s$, and without loss of generality assume the nodes in L adjacent to $\mathrm{Samp}(\sigma) \times \{\mathrm{out}\}$ are $\{r-\ell+1,\ldots,r\}\times\{\mathrm{in}\}$ for some $\ell \le pd$. For each string $y \in \{0,1\}^\ell$, define $f_y : \{0,1\}^{r-\ell} \to \{0,1\}^n$ as $f_y(x) = f(x,y)$ (hard-wiring the last ℓ bits to y). Then each $f_y(U_{r-\ell})$ is a d-local source with min-entropy at least $k - \ell \ge k'$ (see the proof of Lemma 5.7), and this is the key point that enables us to use the technique of [94]. Thus, $\big\|\mathrm{Ext}'\big(f_y(U_{r-\ell})\big) - U_{m'}\big\| \le \epsilon'$. Now consider the joint distribution

$$\Big(U_r|_{\{r-\ell+1,\ldots,r\}}, \mathrm{Ext}'_1\big(f(U_r)\big), \mathrm{Ext}'_2\big(f(U_r)\big)\Big).$$

That is, sample $(x,y) \sim U_r$ and output y along with both parts of $\mathrm{Ext}'\big(f(x,y)\big)$. We have just argued that conditioned on the first coordinate of this distribution being any particular $y \in \{0,1\}^\ell$, the marginal distribution of the other two coordinates is ϵ'-close to uniform. Thus the entire distribution is ϵ'-close to uniform. By Lemma 5.18 (with $s_1 = \ell$, $s_2 = s$, and $s_3 = m' - s$), the joint distribution

$$\Big(U_r^{(\sigma)}|_{\{r-\ell+1,\ldots,r\}}, \mathrm{Ext}'_2\big(f(U_r^{(\sigma)})\big)\Big)$$

is $(\epsilon' 2^{s+1})$-close to the uniform distribution $\big(U_\ell, U_{m'-s}\big)$ where U_ℓ and $U_{m'-s}$ are independent. Let us define $f^{(\sigma)} : \{0,1\}^\ell \to \{0,1\}^p$ by $f^{(\sigma)}(y) = f(x,y)|_{\mathrm{Samp}(\sigma)}$ for any $x \in \{0,1\}^{r-\ell}$ (this value does not depend on x since nodes in $\mathrm{Samp}(\sigma) \times \{\mathrm{out}\}$ are only adjacent to nodes in $\{r-\ell+1,\ldots,r\} \times \{\mathrm{in}\}$). Then we have

$$\mathrm{Ext}\big(f(U_r^{(\sigma)})\big) \;=\; \mathrm{SExt}\Big(f^{(\sigma)}\big(U_r^{(\sigma)}|_{\{r-\ell+1,\ldots,r\}}\big), \mathrm{Ext}'_2\big(f(U_r^{(\sigma)})\big)\Big)$$

and thus

$$\big\|\mathrm{Ext}\big(f(U_r^{(\sigma)})\big) - \mathrm{SExt}\big(f^{(\sigma)}(U_\ell), U_{m'-s}\big)\big\| \;\le\; \epsilon' 2^{s+1}. \tag{5.1}$$

Letting D denote a distribution on $\{0,1\}^p$ with min-entropy at least $pk/4n$ that $f^{(\sigma)}(U_\ell) = f(U_r)|_{\mathrm{Samp}(\sigma)}$ is $\sqrt{\eta + 2^{-\alpha k}}$-close to (such a D exists since σ is good), we have

$$\big\|\mathrm{SExt}\big(f^{(\sigma)}(U_\ell), U_{m'-s}\big) - \mathrm{SExt}\big(D, U_{m'-s}\big)\big\| \;\le\; \sqrt{\eta + 2^{-\alpha k}}. \tag{5.2}$$

Since SExt is a seeded $(pk/4n, \epsilon'')$-extractor, we have

$$\big\|\mathrm{SExt}\big(D, U_{m'-s}\big) - U_m\big\| \;\le\; \epsilon''. \tag{5.3}$$

Combining Inequality (5.1), Inequality (5.2), and Inequality (5.3) yields Claim 5.20. This finishes the proof of Theorem 5.19. □

5.5.2 Applying Theorem 5.19

In order to apply Theorem 5.19, we need explicit constructions of Ext$'$, Samp, and SExt. An appropriate construction of Samp is given by the following lemma.

Lemma 5.21 ([202]). *There exists an explicit* (μ, η)*-sampler* Samp : $\{0,1\}^s \to \binom{\{1,\ldots,n\}}{p}$ *with* $s = 4 \log n \cdot \log \frac{1}{\eta}$*, provided* $\mu p \geq 64 \log \frac{1}{\eta}$ *and* $\eta < 1/16$*.*

The interesting thing about samplers as defined in Definition 5.16 is that they produce a set of fixed size. (Typically, samplers produce either a multiset of fixed size or a set of random size, and the latter is sufficient for the argument in [94].) Nisan and Zuckerman [202] proved Lemma 5.21 by partitioning the n coordinates into p blocks, picking one coordinate from each block in an $O(\log \frac{1}{\eta})$-wise independent way, and using the concentration bounds of [224, 33].[5] Vadhan [249] also constructed a sampler that produces a set of fixed size, and with better seed length for a certain range of parameters. However, his seed length is actually not good enough for our range of parameters.

As for the seeded extractor SExt, plenty of known constructions are good enough for our purpose. For example, we can use the following construction, due to Raz, Reingold, and Vadhan.

Theorem 5.22 ([213]). *There exists an explicit seeded* (k, ϵ)*-extractor* SExt : $\{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ *with* $t = O\big((\log^2 n + \log \frac{1}{\epsilon}) \cdot \log k\big)$ *and* $m = k$*.*

At last, we can prove Theorem 5.1.

Proof of Theorem 5.1. Assume $k \geq n^{2/3+\gamma}$ and $d \leq \beta \log n$ for small enough constant $\beta > 0$. Then for some constant $\beta' > 0$ to be specified shortly, define the following parameters.

- $\epsilon' = 2^{-n^{\beta'}}$
- $k' = k/2$
- $m' = (k')^2/8nd^3 2^d$
- $p = k/2d$
- $\mu = k/2n \log(4n/k)$
- $\eta = 2^{-n^{\beta'/2}}$
- $s = 4 \log n \cdot \log \frac{1}{\eta}$
- $\epsilon'' = 2^{-n^{1/4}}$

[5] Actually, Nisan and Zuckerman proved a version with slightly different constants and where the sampler only needs to work for boolean functions g, but the proof goes through to yield Lemma 5.21.

- $m = pk/4n$
- $t = m' - s$

As shown in the discussion after the statement of Theorem 5.13, combining Theorem 5.13 with Theorem 5.10 yields an explicit (k', ϵ')-extractor $\text{Ext}' : \{0,1\}^n \to \{0,1\}^{m'}$ for d-local sources, provided β' is small enough. By Lemma 5.21 there exists an explicit (μ, η)-sampler $\text{Samp} : \{0,1\}^s \to \binom{\{1,\ldots,n\}}{p}$. Since $t \geq \omega\big((\log^2 p + \log \frac{1}{\epsilon''}) \cdot \log m\big)$, by Theorem 5.22 there exists an explicit seeded (m, ϵ'')-extractor $\text{SExt} : \{0,1\}^p \times \{0,1\}^t \to \{0,1\}^m$. Thus by Theorem 5.19, Ext is a (k, ϵ)-extractor for d-local sources, where $\epsilon = \epsilon'(2^{s+1} + 1) + 2\sqrt{\eta + 2^{-\alpha k}} + \epsilon'' \leq 2^{-n^{\beta}}$ provided β is small enough. □

5.6 Improved Lower Bounds for Sampling Input-Output Pairs

For this section, we define a (d, c, k)*-local sampler* to be a (d, c)-local sampler with at least k non-isolated nodes on the left side of its associated bipartite graph (that is, it makes nontrivial use of at least k random bits). We say a distribution on $\{0,1\}^n$ is a (d, c, k)*-local source* if it equals $f(U_r)$ for some (d, c, k)-local sampler f (with any input length r). Note that a (d, c, k)-local source might not have min-entropy at least k.

Theorem 5.23. *Suppose* $\text{Ext} : \{0,1\}^n \to \{0,1\}$ *is a* $(0, \epsilon)$*-extractor for* $(d, 8d, n/4)$*-local sources, where* $d < n/8$*. Then for every* d*-local source* D *on* $\{0,1\}^{n+1}$ *we have* $\big\|D - \big(U_n, \text{Ext}(U_n)\big)\big\| \geq 1/2 - \epsilon - 2^{-n/2}$.

It might seem suspicious that we are assuming Ext is a $(0, \epsilon)$-extractor. We are not, in fact, extracting from sources with 0 min-entropy — it is possible to derive a lower bound on the min-entropy of any $(d, 8d, n/4)$-local source.[6] The point is that for Theorem 5.23, we do not care about the min-entropy, only the number of non-isolated input nodes. Before proving Theorem 5.23, we show how it implies Theorem 5.3.

Proof of Theorem 5.3. In the proof of Theorem 5.13, we implicitly showed that for all n, k, d, c, ϵ', every (k', ϵ')-extractor for $(1, c)$-local sources is also a (k, ϵ)-extractor for (d, c)-local sources where $k' = k/d^2c2^d$ and $\epsilon = \epsilon' + e^{-k'/4}$ (by replacing $k/2$ with k in the proof). The only property of having min-entropy at least k we used in that proof was that the sampler must make nontrivial use of at least k random bits; thus we can conclude that the extractor is a $(0, \epsilon)$-extractor for (d, c, k)-local sources.

Assume $d \leq \beta \log n$ for some small enough constant $\beta > 0$. Set $c = 8d$ and $k = n/4$ and $k' = k/d^2c2^d = n/32d^3 2^d \geq n^{1/2}$. Using $\gamma = 1/2$ in Theorem 5.10, there exists an explicit (k', ϵ')-extractor for $(1, c)$-local sources with output length 1, where $\epsilon' = 2^{-n^{\Omega(1)}}$

[6]Specifically, a combinatorial argument shows that the source must have many bits that are fully independent of each other and that each have probability $\geq 1/2^d$ for both outcomes 0 and 1. A lower bound on the min-entropy can be derived from this fact.

(since $k' \geq \log^{\omega(1)} n$ and $(k')^{1/2} \geq c$ and $k' - o(k') \geq 1$). By the observation in the previous paragraph, this function is a $(0, \epsilon)$-extractor for $(d, 8d, n/4)$-local sources with error $\epsilon = 2^{-n^{\Omega(1)}}$. Theorem 5.3 follows immediately from this and Theorem 5.23. $\square$

Proof of Theorem 5.23. Consider an arbitrary d-local sampler $f : \{0,1\}^r \to \{0,1\}^{n+1}$, and let $G = (L, R, E)$ be the associated bipartite graph. Since $|E| \leq (n+1)d$, there are at most $(n+1)/8$ nodes in L with degree greater than $8d$. Also, at most $d \leq (n-1)/8$ nodes in L are adjacent to $(n+1, \text{out})$. Without loss of generality, the nodes in L that either have degree greater than $8d$ or are adjacent to $(n+1, \text{out})$ are $\{r - \ell + 1, \ldots, r\} \times \{\text{in}\}$ for some $\ell \leq (n+1)/8 + (n-1)/8 = n/4$. For each string $y \in \{0,1\}^\ell$, define $f_y : \{0,1\}^{r-\ell} \to \{0,1\}^{n+1}$ as $f_y(x) = f(x, y)$ (hard-wiring the last ℓ bits to y) and let $G_y = (L', R, E_y)$ be the associated bipartite graph, where $L' = \{1, \ldots, r - \ell\} \times \{\text{in}\}$. Observe that $f(U_r) = \sum_{y \in \{0,1\}^\ell} \frac{1}{2^\ell} f_y(U_{r-\ell})$. We define the tests

$$\begin{aligned} T_1 \;=\; \big\{z \in \{0,1\}^{n+1} \;:\; & \exists x \in \{0,1\}^{r-\ell}, y \in \{0,1\}^\ell \text{ such that } f(x,y) = z \text{ and} \\ & \big|\{i \in \{1, \ldots, r-\ell\} \;:\; (i, \text{in}) \text{ is non-isolated in } G_y\}\big| < n/4\big\} \end{aligned}$$

and

$$T_2 \;=\; \big\{z \in \{0,1\}^{n+1} \;:\; \text{Ext}(z|_{\{1,\ldots,n\}}) \neq z|_{\{n+1\}}\big\}$$

(in other words, the support of $\big(U_n, \text{Ext}(U_n)\big)$ is the complement of T_2). Finally, we define the test $T = T_1 \cup T_2$.

Claim 5.24. $\Pr_{f(U_r)}[T] \geq 1/2 - \epsilon$.

Claim 5.25. $\Pr_{(U_n, \text{Ext}(U_n))}[T] \leq 2^{-n/2}$.

Combining the two claims, we have $\big|\Pr_{f(U_r)}[T] - \Pr_{(U_n,\text{Ext}(U_n))}[T]\big| \geq 1/2 - \epsilon - 2^{-n/2}$, thus witnessing that $\big\|f(U_r) - \big(U_n, \text{Ext}(U_n)\big)\big\| \geq 1/2 - \epsilon - 2^{-n/2}$.

Proof of Claim 5.24. It suffices to show that for each $y \in \{0,1\}^\ell$, $\Pr_{f_y(U_{r-\ell})}[T] \geq 1/2 - \epsilon$. If y is such that $\big|\{i \in \{1, \ldots, r-\ell\} \;:\; (i, \text{in}) \text{ is non-isolated in } G_y\}\big| < n/4$ then of course $\Pr_{f_y(U_{r-\ell})}[T_1] = 1$. Otherwise, $f_y(U_{r-\ell})$ is a $(d, 8d, n/4)$-source on $\{0,1\}^{n+1}$. Note that $(n+1, \text{out})$ is isolated in G_y; we define $b_y \in \{0,1\}$ to be the fixed value of the $(n+1)$st output bit of f_y, and we define $f'_y : \{0,1\}^{r-\ell} \to \{0,1\}^n$ to be the first n output bits of f_y. Since $f'_y(U_{r-\ell})$ is a $(d, 8d, n/4)$-source on $\{0,1\}^n$, we have $\big\|\text{Ext}\big(f'_y(U_{r-\ell})\big) - U_1\big\| \leq \epsilon$ and thus $\Pr_{b \sim \text{Ext}(f'_y(U_{r-\ell}))}[b \neq b_y] \geq 1/2 - \epsilon$. In other words, $\Pr_{f_y(U_{r-\ell})}[T_2] \geq 1/2 - \epsilon$. This finishes the proof of Claim 5.24. $\square$

Proof of Claim 5.25. By definition, $\Pr_{(U_n,\text{Ext}(U_n))}[T_2] = 0$. Note that $|T_1| \leq 2^{n/2}$ since each string in T_1 can be described by a string of length at most $\ell + n/4 \leq n/2$, namely an appropriate value of y along with the bits of x such that the corresponding nodes in L' are non-isolated in G_y. Since $\big(U_n, \text{Ext}(U_n)\big)$ is uniform over a set of size 2^n, we get $\Pr_{(U_n,\text{Ext}(U_n))}[T_1] \leq 2^{n/2}/2^n = 2^{-n/2}$. This finishes the proof of Claim 5.25. $\square$

This finishes the proof of Theorem 5.23. $\square$

5.7 Improved Extractors for Low-Weight Affine Sources

We now describe the proof of Theorem 5.10. To do this, we need a construction of linear strong seeded extractors with good seed length, which we present in Section 5.7.1. Then in Section 5.7.2 we derive Theorem 5.10.

5.7.1 A Linear Strong Seeded Extractor with Seed Length $\log n + O(\log k)$

Our goal in this section is to prove the following theorem.[7]

Theorem 5.26. *There exists a constant c such that for all $k \geq c\log^2 n$ there exists an explicit linear strong seeded $(k, 1/4)$-extractor* $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ *with $t = \log n + c\log k$ and $m = k^{1/4}$.*

It is very important to us that the seed length here has $\log n$ and not $O(\log n)$. If instead we use an extractor with $c\log n$ in the seed length, then in Theorem 5.10 we would only be able to get an extractor for the class of weight-$k^{(1/c)-\gamma}$ affine sources as opposed to weight-$k^{1-\gamma}$ affine sources.

We also note that without the linearity property, such an extractor is explicitly constructed and stated in [121, Theorem 5.12]. We construct such an extractor with the linearity property by using a construction from [121] and then bootstrapping it with another known construction. To do this, we first define and construct objects called linear strong condensers.

Definition 5.27. *We say $C : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ is a* strong $k \to_\epsilon k'$ condenser *if for every distribution D on $\{0,1\}^n$ with min-entropy at least k,*

$$\Pr_{y \sim U_t}\left[C(D, y) \text{ is } \epsilon\text{-close to a distribution with min-entropy at least } k'\right] \;\geq\; 1 - \epsilon.$$

Recall that we say C is *linear* if for every $y \in \{0,1\}^t$, the function $C(\cdot, y) : \{0,1\}^n \to \{0,1\}^m$ is linear over $\mathbb{F}_2$.

Our construction of a linear strong condenser is the same as one of the constructions in [121], which in turn is based on an idea from [120]. However, we need to argue about its linearity as well as the parameters, so we state the construction and result of [121] here. Consider a finite field $\mathbb{F}_q$ for some $q = 2^t$. Let ζ be a generator of the multiplicative group $\mathbb{F}_q^*$. Then the function $C : \mathbb{F}_q^{n'} \times \mathbb{F}_q \to \mathbb{F}_q^{m'}$ is as follows.

> Given $f = (f_0, \ldots, f_{n'-1}) \in \mathbb{F}_q^{n'}$, we interpret it as a polynomial $f : \mathbb{F}_q \to \mathbb{F}_q$ such that $f : y \mapsto \sum_{0 \leq i < n'} f_i y^i$. We now describe C as $C : (f, y) \mapsto \big(f(y), f(\zeta y), \ldots, f(\zeta^{m'-1} y)\big)$.

[7]We note that Theorem 5.26 can be generalized by setting the parameters appropriately in our argument. For general error ϵ, we can get seed length $\log n + O(\log(k/\epsilon))$, and the output length can be improved to $k^{1-\alpha}$ for any constant $\alpha > 0$ at the expense of increasing the lower bound on k. However, we only prove the version we need for Theorem 5.10.

Observation 5.28. *For all $y \in \mathbb{F}_q$, the function $C(\cdot, y) : f \mapsto C(f, y)$ is $\mathbb{F}_q$-linear.*

Observation 5.29. *For $q = 2^t$, there is an isomorphism between $(\mathbb{F}_q, +)$ and $(\mathbb{F}_2^t, \oplus)$. Further, this isomorphism is computable in time polynomial in t.*

Thus we can interpret C as a function $C : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ where $n = n' \cdot t$ and $m = m' \cdot t$. Further, C is $\mathbb{F}_2$-linear, and it is polynomial time computable since a generator of $\mathbb{F}_q^*$ can be computed in time polynomial in t [234]. The fact that C is a strong condenser follows from [121, Theorem 7.2].

Theorem 5.30 ([121]). *For every $\ell \leq n$ such that 2^ℓ is an integer, and for every $\alpha, \epsilon > 0$, the function $C : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ as defined above is a*

$$\text{strong } (1 + 1/\alpha)\ell d + \log(1/\epsilon) \to_{\sqrt{3\epsilon}} \ell d - 2 \text{ condenser}$$

with $t \leq (1 + 1/\alpha)d$ and $m \leq (1 + 1/\alpha)\ell d$ where $d = \lceil \alpha \log(4n\ell/\epsilon) \rceil$, provided $\ell d \geq \log(1/\epsilon)$.

The following important corollary follows by setting parameters correctly in the result. Assume $k \geq c \log^2 n$ for some large constant $c > 0$, and set the parameters as follows.

- $\epsilon = 1/2^8$
- $\alpha = (\log k)/(2^8 \cdot \log n)$
- $\ell = k/(2^8 \cdot \log n)$

This implies the following.

- $d = \left\lceil \frac{(\log k)(\log n + \log k - \log\log n + O(1))}{2^8 \cdot \log n} \right\rceil = \frac{(\log k)(\log n + \log k - \log\log n)(1 + o(1))}{2^8 \cdot \log n}$
- $t \leq \big(\log n + \log k - \log\log n\big)\big(1 + (2\log k/\log n)\big) \leq \log n + 5\log k$
- $m \leq \frac{k}{2^8 \cdot \log n} \cdot (\log n + 5\log k) \leq k$
- $\ell d \geq (k \log k)/(2^{16} \cdot \log n) \geq k^{1/2} + 2$
- $k \geq (1 + 1/\alpha)\ell d + \log(1/\epsilon)$

Hence, we now get the following corollary.

Corollary 5.31. *There exists a constant c such that for all $k \geq c \log^2 n$ there exists an explicit linear strong $k \to_{1/8} k^{1/2}$ condenser $C : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ with $t = \log n + 5\log k$ and $m = k$.*

We now recall that the strong seeded extractors in [241, 214] are also linear.

Theorem 5.32 ([241]). *There exists an explicit linear strong seeded $(n^{1/2}, 1/8)$-extractor* $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ *with $t = O(\log n)$ and $m = n^{1/4}$.*

Theorem 5.26 follows from Corollary 5.31 and Theorem 5.32.

5.7.2 Proof of Theorem 5.10

In this section we prove Theorem 5.10. As we have said before, Rao [209] proves the same kind of theorem except it is weaker in the upper bound on the weight allowed for the affine sources. Our extractor construction uses the same steps as [209], except the components used in our construction are tailor-made for our purposes thus helping us achieve better parameters. Throughout this section, all references to particular theorems in [209] actually refer to the ECCC version of the paper (technical report TR08-015). Also, throughout this section we let c be the constant from Theorem 5.26.

In order to describe the better extractors, we first recall the following linear error-correcting code construction (the BCH code).

Theorem 5.33. *For every $d < n$ there exists an explicit parity check function $P : \mathbb{F}_2^n \to \mathbb{F}_2^m$ for a linear code with distance greater than d, such that $m = O(d \log n)$.*

We now recall the following claim from [209, Lemma 6.1].

Claim 5.34. *Let $P : \mathbb{F}_2^n \to \mathbb{F}_2^m$ be a parity check function for a linear code with distance greater than d. Let D be any weight-w affine source with min-entropy at least d/w. Then $P(D)$ is an affine source with min-entropy at least d/w.*

Combining Theorem 5.33 and Claim 5.34 (using $d = k^{1-\gamma/2}$), we get the following.

Lemma 5.35. *For every constant $\gamma > 0$ and all k there exists an explicit linear function $P : \mathbb{F}_2^n \to \mathbb{F}_2^m$ with $m = O(k^{1-\gamma/2} \cdot \log n)$ such that if D is a weight-$k^{1-\gamma}$ affine source with min-entropy at least k, then $P(D)$ is an affine source with min-entropy at least $k^{\gamma/2}$.*

Now let γ and k be as in Theorem 5.10. Let $m_0 = O(k^{1-\gamma/2} \cdot \log n)$ be the output length from Lemma 5.35. Let $\mathrm{Ext}_1 : \{0,1\}^{m_0} \times \{0,1\}^{t_1} \to \{0,1\}^{m_1}$ be the linear strong seeded extractor from Theorem 5.26 set up to work for min-entropy $k^{\gamma/4c}$ (which is less than $k^{\gamma/2}$ and is at least $c \log^2 m_0$ assuming $k \geq \log^{10c/\gamma} n$). Thus we have $t_1 = \log m_0 + (\gamma/4) \log k$ and $m_1 = k^{\gamma/16c}$. In Figure 5.2 we present the routine LOW-CONVERT from [209]. The following lemma was proven in [209, Lemma 6.3]. We note that for this, the error of Ext_1 only needs to be $< 1/2$.

Definition 5.36. *A distribution D on $\{0,1\}^{\ell \times \ell'}$ is said to be an* affine somewhere random source *if D is an affine source and for some $1 \leq i \leq \ell$, the i^{th} row of D is uniformly random.*

Lemma 5.37. *For every constant $\gamma > 0$, if D is a weight-$k^{1-\gamma}$ affine source with min-entropy at least $k \geq \log^{10c/\gamma} n$, then $LC(D)$ is an affine somewhere random source of size $2^{t_1} \times m_1$.*

Note that the number of rows in the output of LC is $2^{t_1} = m_0 \cdot k^{\gamma/4} = O(k^{1-\gamma/4} \cdot \log n) \leq k^{1-\gamma/8}$. At this stage, we also point out how Theorem 5.26's optimized dependence of the seed length on the length of the source is crucial for the construction. For the rest of the

LOW-CONVERT(D)

Input: $x \in \{0,1\}^n$
Output: $z \in \{0,1\}^{2^{t_1} \times m_1}$

Subroutines used: $P : \{0,1\}^n \to \{0,1\}^{m_0}$ from Lemma 5.35, and $\text{Ext}_1 : \{0,1\}^{m_0} \times \{0,1\}^{t_1} \to \{0,1\}^{m_1}$ from Theorem 5.26. Here $m_0 = O(k^{1-\gamma/2} \cdot \log n)$, $t_1 = \log m_0 + (\gamma/4) \log k$, and $m_1 = k^{\gamma/16c}$.

For $1 \le i \le 2^{t_1}$, the i^{th} row of the output is defined by $LC(x)_i = \text{Ext}_1\big(P(x), i\big)$.

Figure 5.2: LOW-CONVERT

AFFINE-CONVERT(D)

Input: $x \in \{0,1\}^n$
Output: $z \in \{0,1\}^{2^{t_1} \times m_2}$

Subroutines used: $LC : \{0,1\}^n \to \{0,1\}^{2^{t_1} \times m_1}$ from Lemma 5.37, and $\text{Ext}_2 : \{0,1\}^n \times \{0,1\}^{t_2} \to \{0,1\}^{m_2}$ from Theorem 5.38. Here $t_2 = m_1$ and $m_2 = k - o(k)$.

For $1 \le i \le 2^{t_1}$, the i^{th} row of the output is defined by $AC(x)_i = \text{Ext}_2\big(x, LC(x)_i\big)$.

Figure 5.3: AFFINE-CONVERT

argument to go through, we require the number of rows in $LC(x)$ (namely 2^{t_1}) to be smaller than k. If we used a linear strong seeded extractor for which $t_1 \ge c' \log m_0$ then this would force $m_0 < k^{1/c'}$. However, our use of Theorem 5.33 and Claim 5.34 requires $m_0 > w$, which would imply that we need $w < k^{1/c'}$. Instead, using our optimized extractor from Theorem 5.26, we are able to handle any weight $w \le k^{1-\gamma}$.

In order to define the next routine, we recall an extractor construction from [214] for a particular setting of parameters.

Theorem 5.38 ([214]). *There exists an explicit linear strong seeded* (k, ϵ)*-extractor* $\text{Ext}_2 : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ *with* $t = O(\log^3(n/\epsilon))$ *and* $m = k - O(\log^3(n/\epsilon))$.

We set up Ext_2 to work for min-entropy $k - 2^{t_1} \cdot m_1 \ge k - k^{1-\gamma/8+\gamma/16c} = k - o(k)$ and seed length $t_2 = m_1$ and thus we get output length $m_2 = k - o(k)$ with error $2^{-k^{\Omega(1)}}$. In Figure 5.3 we present the routine AFFINE-CONVERT from [209]. The following lemma was proven in [209, Theorem 6.5].

Lemma 5.39. *For every constant $\gamma > 0$, if D is a weight-$k^{1-\gamma}$ affine source with min-entropy at least $k \geq \log^{10c/\gamma} n$, then $AC(D)$ is $2^{-k^{\Omega(1)}}$-close to a convex combination of affine somewhere random sources of size $2^{t_1} \times m_2$.*

Lemma 5.39 says that the output of $AC(D)$ is close to a convex combination of affine somewhere random sources. Since the length of each row is m_2 and the number of rows is $2^{t_1} \leq k^{1-\gamma/8} \leq m_2^{1-\gamma/9} \ll m_2$, we can apply the routine AFFINE-SREXT from [209]. The following lemma was proven in [209, Theorem 5.1].[8]

Lemma 5.40. *For every constant $\alpha > 0$ there exists an explicit function $A : \{0,1\}^{k^{1-\alpha} \times k} \to \{0,1\}^m$ such that if D is an affine somewhere random source of size $k^{1-\alpha} \times k$, then $\|A(D) - U_m\| \leq 2^{-k^{\Omega(1)}}$ where $m = k - o(k)$.*

Theorem 5.10 follows from Lemma 5.39 and Lemma 5.40.

[8]We note that [209, Theorem 5.1] discusses affine somewhere random sources of size $k^{0.7} \times k$. However, it is straightforward to see that the result just requires the number of rows in the affine somewhere random source to be polynomially smaller than the length of each row.

Chapter 6

The Complexity of Estimating Min-Entropy

6.1 Introduction

Deterministic randomness extraction is the problem of taking a sample from an imperfect physical source of randomness (modeled as a probability distribution on bit strings) and applying an efficient deterministic algorithm to transform it into a uniformly random string, which can be used by a randomized algorithm (see [228, 250] for surveys of this topic). For such extraction to be possible, the source of randomness must satisfy two properties: (i) it must contain a sufficient "amount of randomness", and (ii) it must be "structured", meaning that it has a simple description.

Regarding property (i), the most useful measure of the "amount of randomness" is the *min-entropy*, which is the logarithm of the reciprocal of the probability of the most likely outcome. In other words, if a distribution has high min-entropy then every outcome has small probability. The number of uniformly random bits produced by the extractor cannot exceed the min-entropy of the source, and one of the goals in designing extractors is to get as close to the min-entropy as possible. Regarding property (ii), if the distribution is generated by an efficient process in the physical world, then it can be modeled as being sampled by an efficient algorithm given uniform random bits. This sampling algorithm is a simple description of the distribution. Trevisan and Vadhan [246] initiated the study of extracting from efficiently samplable distributions. Assuming certain complexity-theoretic conjectures, they constructed extractors for time-efficient samplers.[1] Kamp et al. [153] gave an *unconditional* construction of extractors for space-efficient samplers with streaming (one-way) access to their random input bits. De and Watson [70] (see Chapter 5) and Viola [261] gave constructions of extractors for *local* samplers (where each output bit of the sampler only depends on a small number of the random input bits), and Viola [261] generalized this to get extractors for samplers that are constant-depth circuits (of the AC^0 type). Viola [263] has constructed extractors for sequential-access one-tape Turing machine samplers.

[1]We mention that a somewhat related but incomparable problem was studied in [75].

All of these extractor constructions need to be given a lower bound on the min-entropy of the distribution. The output length of the extractor depends on this lower bound. Thus, if we had a sampling algorithm (assumed to model the physical source), it would be nice to know the min-entropy so we could plug this parameter into the extractor, and thus extract as much of the randomness as possible.[2] This motivates the following computational problem: Given an efficient algorithm that outputs a sample from a probability distribution, estimate the min-entropy of the distribution. The upshot of our results is that this problem is intractable even for the extremely simple samplers studied in [153, 70, 261], and we pinpoint the precise complexity of the problem.

Goldreich, Sahai, and Vadhan [103] considered the problem of estimating the *Shannon entropy* of a distribution sampled by a given circuit. They showed that an appropriate formulation of the problem is complete for the complexity class NISZK (non-interactive statistical zero-knowledge) and is thus believed to be intractable. For the *min-entropy* version, we show that the problem is interreducible with the "approximate lower bound" problem that was famously studied by Goldwasser and Sipser [107]. The latter formulation of multiplicative approximate counting of NP witnesses deserves its own complexity class. Indeed, the class has already been named SBP by [48], and it is perhaps the only natural example of a class sandwiched between MA and AM. We prove that the min-entropy estimation promise problem is SBP-complete even when restricted to 3-local samplers (as studied in [70, 261]).

For logarithmic-space samplers that have one-way access to their randomness (as studied in [153]), it turns out that our min-entropy estimation promise problem has already been studied (though in a very different context and with different terminology) by Lyngsø and Pedersen [184], who proved that an equivalent problem is NP-complete. We discuss the relationship between their problem and our problem.

6.1.1 Definitions

The *min-entropy* of a distribution D over a finite set S is $\mathrm{H}_\infty(D) = \min_{s \in S} \log_2\big(1/\Pr_D[s]\big)$. Let U_r denote the uniform distribution over $\{0,1\}^r$. If $A : \{0,1\}^r \to \{0,1\}^m$ is an algorithm that takes r uniformly random bits and outputs m bits, then $A(U_r)$ denotes the output distribution of A. We write $A(U)$ with the convention that $U = U_r$ for the appropriate value of r. We consider three classes of sampling algorithms.

- *Circuits* are the usual boolean circuits.
- *d-Local* samplers are functions where each of the m output bits depends on at most d of the r input bits (where d is a constant).
- *Logarithmic-space* samplers can be defined in several equivalent ways; the following is the most convenient for us. The sampler is a layered directed graph where each edge

[2] We remark that the aforementioned extractor constructions *do not* assume knowledge of the sampling algorithm itself, only knowledge of the class of algorithms the sampler comes from. It is not known how to exploit knowledge of the description of the distribution for extraction purposes, except in trivial cases such as when the distribution is uniform over an affine subspace of $GF(2)^n$.

goes from one layer to the immediate next layer. There is a unique start vertex in layer 0. For each vertex except the ones in the last layer, there is at least one outgoing edge, and the outgoing edges are labeled with a probability distribution. Each vertex except the start vertex is labeled with a bit. A sample is obtained by taking a random walk (starting at the start vertex) and outputting the bit labels of the visited vertices.[3]

Such d-local samplers and logarithmic-space samplers have been studied in other contexts besides randomness extraction. For example, there are many positive and negative results on whether d-local samplers can implement pseudorandom generators and one-way functions (see Section 5.1.4 for extensive pointers to the literature). Trevisan et al. [247] showed how to efficiently perform near-optimal prefix-free compression of distributions with logarithmic-space samplers.

The min-entropy estimation problem that we study is formulated in terms of promise problems (see [97] for a survey on promise problems).

Definition 6.1. *For any class $\mathcal{A}$ of algorithms, $\mathcal{A}$-*Min-Ent-Gap *is the following promise problem.*

$$\begin{aligned} \mathcal{A}\text{-}\textsc{Min-Ent-Gap}_{\text{YES}} &= \big\{(A,h) \ : \ A \in \mathcal{A} \textit{ and } \mathrm{H}_\infty(A(U)) \le h\big\} \\ \mathcal{A}\text{-}\textsc{Min-Ent-Gap}_{\text{NO}} &= \big\{(A,h) \ : \ A \in \mathcal{A} \textit{ and } \mathrm{H}_\infty(A(U)) > h+1\big\} \end{aligned}$$

Taking $\mathcal{A}$ to be circuits, d-local samplers, or logarithmic-space samplers, we get the problems Circuit-Min-Ent-Gap, d-Local-Min-Ent-Gap, and Logspace-Min-Ent-Gap. The size of the input (A,h) is the bit length of the description of the algorithm A, plus the bit length of the integer h. Note that if one of these problems has a polynomial-time algorithm, then the min-entropy can be estimated within an additive 1 in polynomial time by trying all possible values of $h \in \{0,1,\ldots,m\}$ (or using binary search).

Throughout this chapter, when we talk about reductions and completeness, we are always referring to deterministic polynomial-time mapping reductions.

Definition 6.2. prSBP *is the class of promise problems $\Pi = (\Pi_{\text{YES}}, \Pi_{\text{NO}})$ for which there exist polynomial-time algorithms M, K (where M outputs a bit and K outputs a nonnegative integer) and a polynomial p such that the following hold for all $x \in \{0,1\}^*$.*

$$\begin{aligned} x \in \Pi_{\text{YES}} &\implies \Big|\big\{y \in \{0,1\}^{p(|x|)} \ : \ M(x,y)=1\big\}\Big| \ge K(x) \\ x \in \Pi_{\text{NO}} &\implies \Big|\big\{y \in \{0,1\}^{p(|x|)} \ : \ M(x,y)=1\big\}\Big| < K(x)/2 \end{aligned}$$

Equivalently, prSBP *is the class of all promise problems reducible to the following promise problem* Circuit-Count-Gap.

$$\begin{aligned} \textsc{Circuit-Count-Gap}_{\text{YES}} &= \big\{(C,k) \ : \ C \textit{ is a circuit that accepts} \ge k \textit{ inputs}\big\} \\ \textsc{Circuit-Count-Gap}_{\text{NO}} &= \big\{(C,k) \ : \ C \textit{ is a circuit that accepts} < k/2 \textit{ inputs}\big\} \end{aligned}$$

SBP *is defined as the class of languages in* prSBP.

[3]The model in [153] is the same except the output bits are on the edges rather than on the vertices (Mealy style rather than Moore style). The two models are equivalent up to a small difference in the size of the graph.

Böhler, Glaßer, and Meister [48] introduced the class SBP (which stands for "small bounded-error probability") and provided a fairly comprehensive study of it from a structural complexity perspective, analyzing its relationship to other classes (inclusions and relativized separations), its closure properties, and the possibility of it having complete languages. We have MA $\subseteq$ SBP $\subseteq$ AM, where MA $\subseteq$ SBP follows by observing that the standard proof of MA $\subseteq$ PP [255] automatically yields a multiplicative gap, and SBP $\subseteq$ AM follows immediately from the Goldwasser-Sipser lower bound protocol [107]. Both inclusions relativize. There is an oracle relative to which SBP $\not\subseteq \Sigma_2$P [48] and thus MA $\neq$ SBP (since MA $\subseteq \Sigma_2$P relativizes), and there is an oracle relative to which AM $\not\subseteq$ PP [255] and thus SBP $\neq$ AM (since SBP $\subseteq$ PP relativizes). Since AM can be derandomized to NP under complexity assumptions [162, 20, 189, 230], it is believed that SBP = NP. The factor of 1/2 in the gap in Definition 6.2 is arbitrary and can be replaced by $1 - 1/q(|x|)$ for any polynomial q, by a standard trick.[4]

Although very few papers explicitly mention the class SBP, the Goldwasser-Sipser protocol for Circuit-Count-Gap has countless applications in complexity and cryptography, and thus SBP has been implicitly studied many times. For example, it is shown in [2, 21] that $\mathrm{E}^{\mathrm{prSBP}}$ contains languages of circuit complexity $\Omega(2^n/n)$.

6.1.2 Results

Theorem 6.3. Circuit-Min-Ent-Gap *is* prSBP-*complete.*

Theorem 6.4. 3-Local-Min-Ent-Gap *is* prSBP-*complete.*

Theorem 6.5. Logspace-Min-Ent-Gap *is* prNP-*complete.*

We prove Theorem 6.3 and Theorem 6.4 in Section 6.2. In our proof of Theorem 6.3, we implicitly use a "closure under nondeterminism" property of SBP, which was not shown in [48]. This is analogous to how AM (and trivially, MA) is "closed under nondeterminism". In Section 6.4 we explicitly state and prove a more general form of this property of SBP. The general form is not needed for our theorems about min-entropy, but it demonstrates the robustness of the class SBP and may be useful for future results about SBP.

Regarding Theorem 6.4, our proof shows that the completeness holds even when each output bit of the sampler is the disjunction of exactly three unnegated input bits. Note that the min-entropy of a 1-local sampler's distribution is trivial to compute exactly since the distribution is affine. The complexity of estimating min-entropy for 2-local samplers remains open.

During our proof of Theorem 6.4, we also show that Monotone-2-Sat-Count-Gap (which is defined in the natural way) is prSBP-complete. It was previously known (presumably folklore) that this problem is in prBPP iff NP = RP (roughly speaking, it is "NP-complete modulo randomness"). This is implied by our result, which more precisely

[4]Modify K so its output is raised to the power q, and modify M so its number of accepted strings y is also raised to the power q (by taking $y_1, \ldots, y_{q(|x|)} \in \{0,1\}^{p(|x|)}$ and accepting iff $M(x, y_i) = 1$ for all i).

quantifies the complexity in terms of deterministic reductions. Also, it follows from our argument that the exact threshold analogue of Monotone-2-Sat-Count-Gap (distinguishing $\geq k$ satisfying assignments from $< k$ satisfying assignments) is PP-complete (under mapping reductions), and that #Monotone-2-Sat (where the goal is to output the number of satisfying assignments) is #P-complete (under one-query reductions). The former did not seem to be known. The latter is well-known, but the only proof we could find mentioned in the literature is due to Valiant [252, 253] and is based on the #P-completeness of computing the permanent. Our argument is much more elementary and demonstrates that this heavy machinery is not needed for the #P-completeness of #Monotone-2-Sat.

We discuss Theorem 6.5 in Section 6.3. The prNP-hardness follows without difficulty from a result of Lyngsø and Pedersen on hidden Markov models [184]. To cut to the chase, the only issue is that their result allows the sampler to output strings of different lengths, while our definition requires it to output fixed-length strings. This issue is straightforward to resolve.

6.1.3 Related Work

Goldreich et al. [103] showed that the variant of Circuit-Min-Ent-Gap where Shannon entropy replaces min-entropy and the roles of YES and NO instances are interchanged is prNISZK-complete. Dvir et al. [78] gave some upper and lower bounds on the complexity of estimating various types of entropy for distributions where a sample is obtained by plugging a uniform input into a sequence of low-degree multivariate polynomials over a finite field. Dvir et al. [78] also studied the complexity of estimating Shannon entropy for d-local samplers and for logarithmic-space samplers that have *two*-way access to their randomness.

Other papers that are in a somewhat similar spirit as ours include [194, 85, 41, 36]. See Section 5.1.4 for an overview of past work on locally computable functions.

The study of multiplicative approximate counting of NP witnesses was initiated in [237]. Derandomization of approximate counting was studied in [230]. See [79] for a more algorithmic perspective on the complexity of approximate counting. Kuperberg [164] showed that two variants of SBP are actually equal: SBQP (the quantum variant) and $\mathrm{A_0PP}$ (the variant where we consider a difference of two #P functions rather than a single #P function). Kabanets et al. [151] defined and studied a complexity class that captures *additive* approximate counting in a more direct way than BPP does.

6.2 Proof of Theorem 6.3 and Theorem 6.4

Theorem 6.3 and Theorem 6.4 follow from the following four lemmas.

Lemma 6.6. Circuit-Min-Ent-Gap $\in$ prSBP.

Lemma 6.7. Circuit-Min-Ent-Gap *is* prSBP-*hard.*

Lemma 6.8. d-Sat-Count-Gap *reduces to* $(d+1)$-Local-Min-Ent-Gap.

Lemma 6.9. Monotone-2-Sat-Count-Gap *is* prSBP*-hard.*

Lemma 6.7 is implied by Lemma 6.8 and Lemma 6.9. However, the proof of Lemma 6.7 serves as a warmup for the proof of Lemma 6.8, and only the former is needed for Theorem 6.3.

By Lemma 6.9, Monotone-2-Sat-Count-Gap is prSBP-complete. This problem was previously known to be prNP-hard, by combining any reduction to Vertex-Cover with a "blow-up" trick that is usually attributed to [149, 236]. To get the prSBP-completeness, we need to use a particular reduction to Vertex-Cover, satisfying certain properties.

We now prove the above four lemmas.

Proof of Lemma 6.6. We reduce Circuit-Min-Ent-Gap to Circuit-Count-Gap. Given an instance (A, h) of Circuit-Min-Ent-Gap where $A : \{0,1\}^r \to \{0,1\}^m$ is a circuit and without loss of generality $h \le \min(r, m)$, we construct a circuit $C : \{0,1\}^m \times (\{0,1\}^r)^{m+1} \to \{0,1\}$ by

$$C(x, y_1, \ldots, y_{m+1}) \;=\; \begin{cases} 1 & \text{if } A(y_i) = x \text{ for all } i \\ 0 & \text{otherwise} \end{cases}$$

and let $k = 2^{(r-h)(m+1)}$. We show the following two things.

$$\begin{aligned} (A, h) \in \textsc{Circuit-Min-Ent-Gap}_{\text{YES}} &\implies (C, k) \in \textsc{Circuit-Count-Gap}_{\text{YES}} \\ (A, h) \in \textsc{Circuit-Min-Ent-Gap}_{\text{NO}} &\implies (C, k) \in \textsc{Circuit-Count-Gap}_{\text{NO}} \end{aligned}$$

For the YES case, the assumption $\mathrm{H}_\infty(A(U_r)) \le h$ means there exists an $x \in \{0,1\}^m$ such that $\Pr_{A(U_r)}[x] \ge 1/2^h$ and thus there are $\ge 2^{r-h}$ strings y for which $A(y) = x$. Thus there are $\ge 2^{(r-h)(m+1)}$ choices of $y_1, \ldots, y_{m+1}$ for which $A(y_i) = x$ for all i, which implies that C accepts $\ge k$ inputs. For the NO case, the assumption $\mathrm{H}_\infty(A(U_r)) > h+1$ means that for all $x \in \{0,1\}^m$, $\Pr_{A(U_r)}[x] < 1/2^{h+1}$ and thus there are $< 2^{r-h-1}$ strings y for which $A(y) = x$. Thus there are $< 2^{(r-h-1)(m+1)}$ choices of $y_1, \ldots, y_{m+1}$ for which $A(y_i) = x$ for all i. By summing over x, this implies that C accepts $< 2^m \cdot 2^{(r-h-1)(m+1)} = k/2$ inputs. □

Proof of Lemma 6.7. We reduce Circuit-Count-Gap to Circuit-Min-Ent-Gap. Given an instance (C, k) of Circuit-Count-Gap where $C : \{0,1\}^n \to \{0,1\}$ is a circuit and without loss of generality $1 \le k \le 2^n$, by the standard amplification trick we may assume that C accepts $\ge k$ inputs in the YES case and $< k/8$ inputs in the NO case. We construct a circuit $A : \{0,1\}^n \times \{0,1\}^{2n} \to \{0,1\}^{2n}$ by

$$A(y, z) \;=\; \begin{cases} 1^{2n} & \text{if } C(y) = 1 \\ z & \text{otherwise} \end{cases}$$

and let h be the smallest integer such that $1/2^h \le k/2^n$. We show the following two things.

$$\begin{aligned} (C, k) \in \textsc{Circuit-Count-Gap}_{\text{YES}} &\implies (A, h) \in \textsc{Circuit-Min-Ent-Gap}_{\text{YES}} \\ (C, k) \in \textsc{Circuit-Count-Gap}_{\text{NO}} &\implies (A, h) \in \textsc{Circuit-Min-Ent-Gap}_{\text{NO}} \end{aligned}$$

For the YES case, the assumption that C accepts $\geq k$ inputs implies that $\Pr_{A(U_{3n})}[1^{2n}] \geq k/2^n \geq 1/2^h$ and thus $\mathrm{H}_\infty(A(U_{3n})) \leq h$. For the NO case, the assumption that C accepts $< k/8$ inputs implies that

$$\begin{aligned}\Pr_{A(U_{3n})}[1^{2n}] &\leq \Pr_{y\sim U_n}[C(y)=1] + \Pr_{z\sim U_{2n}}[z=1^{2n}] \\ &< (k/8)/2^n + 1/2^{2n} \\ &< (k/4)/2^n \\ &< 1/2^{h+1}\end{aligned}$$

by the minimality of h. Since 1^{2n} is the most probable string under $A(U_{3n})$, this implies that $\mathrm{H}_\infty(A(U_{3n})) > h+1$. ☐

Proof of Lemma 6.8. Given an instance (φ, k) of d-Sat-Count-Gap,[5] where φ is a d-Sat formula having n variables and m clauses and without loss of generality $1 \leq k \leq 2^n$, by the standard amplification trick we may assume that φ has $\geq k$ satisfying assignments in the YES case and $< k/8$ satisfying assignments in the NO case. (This does not affect the constant d, since the amplified formula is just a conjunction of several copies of the original d-Sat formula on disjoint variables.) Suppose the i^{th} clause of φ consists of the literals $\ell_{i,1} \vee \cdots \vee \ell_{i,d}$. We construct a $(d+1)$-local function $A : \{0,1\}^n \times \{0,1\}^{2n} \to \{0,1\}^{m\cdot 2n}$ where the first input is the variables of φ (denoted y) and the bits of the second input are labeled as z_j. We let the i,j bit of the output (for $i \in \{1,\ldots,m\}$, $j \in \{1,\ldots,2n\}$) be

$$A(y,z)_{i,j} = \ell_{i,1} \vee \cdots \vee \ell_{i,d} \vee z_j$$

and let h be the smallest integer such that $1/2^h \leq k/2^n$. We show the following two things.

$$\begin{aligned}(\varphi,k) \in d\text{-}\textsc{Sat-Count-Gap}_{\text{YES}} &\implies (A,h) \in (d+1)\text{-}\textsc{Local-Min-Ent-Gap}_{\text{YES}} \\ (\varphi,k) \in d\text{-}\textsc{Sat-Count-Gap}_{\text{NO}} &\implies (A,h) \in (d+1)\text{-}\textsc{Local-Min-Ent-Gap}_{\text{NO}}\end{aligned}$$

Note that for any fixed assignment y, if the i^{th} clause is satisfied then the i,j output bits are all 1 (with probability 1 over random z), and if the i^{th} clause is not satisfied then the i,j output bits are uniformly distributed (equal to z). It follows that if y satisfies φ then $\Pr_{A(y,U_{2n})}[1^{m\cdot 2n}] = 1$, and if y does not satisfy φ then $\Pr_{A(y,U_{2n})}[1^{m\cdot 2n}] = 1/2^{2n}$. In either case, $1^{m\cdot 2n}$ is a most probable string under $A(y,U_{2n})$, and thus $1^{m\cdot 2n}$ is a most probable string under $A(U_{3n})$.

For the YES case, the assumption that φ has $\geq k$ satisfying assignments implies that

$$\Pr_{A(U_{3n})}[1^{m\cdot 2n}] \geq \Pr_{y\sim U_n}[y \text{ satisfies } \varphi] \geq k/2^n \geq 1/2^h$$

and thus $\mathrm{H}_\infty(A(U_{3n})) \leq h$. For the NO case, the assumption that φ has $< k/8$ satisfying assignments implies that

$$\Pr_{A(U_{3n})}[1^{m\cdot 2n}] = 1 \cdot \Pr_{y\sim U_n}[y \text{ satisfies } \varphi] + (1/2^{2n}) \cdot \Pr_{y\sim U_n}[y \text{ does not satisfy } \varphi]$$

[5]In fact, the proof works for arbitrary d-CSPs over the binary alphabet.

$$
\begin{aligned}
&< (k/8)/2^n + 1/2^{2n} \\
&< (k/4)/2^n \\
&< 1/2^{h+1}
\end{aligned}
$$

by the minimality of h. Since $1^{m\cdot 2n}$ is a most probable string, this implies that $\mathrm{H}_\infty(A(U_{3n})) > h+1$. □

Proof of Lemma 6.9. We reduce Circuit-Count-Gap to Monotone-2-Sat-Count-Gap. Given an instance (C, k) of Circuit-Count-Gap where without loss of generality $k \geq 1$, by the standard amplification trick we may assume that C accepts $\geq k$ inputs in the YES case and $< k/4$ inputs in the NO case. We first apply the standard parsimonious reduction from Circuit-Sat to 3-Sat to obtain a formula φ with the same number of satisfying assignments as C. Next we apply a careful reduction from 3-Sat to Vertex-Cover: For each clause of φ, create seven vertices representing the satisfying assignments for the three variables in the clause, and put an edge between two vertices if they conflict (i.e., they assign some variable opposite truth values). Let G denote this graph, and let $\ell = 6m$ where m is the number of clauses in φ. This particular reduction has the following two properties: (i) it is parsimonious (i.e., the number of vertex covers of G of size at most ℓ equals the number of satisfying assignments of φ), and (ii) every vertex cover of G has size at least ℓ.

Next we apply the "blow-up" trick. Create a new graph G' by transforming each vertex of G into a cloud of $10m$ vertices and transforming each edge of G into a complete bipartite graph between its two clouds. We view G' as a Monotone-2-Sat formula φ' where vertices become variables and edges become clauses, and we let $k' = k \cdot (2^{10m} - 1)^m$. We show the following two things.

$$
\begin{aligned}
(C, k) \in \textsc{Circuit-Count-Gap}_{\text{YES}} &\implies (\varphi', k') \in \textsc{Monotone-2-Sat-Count-Gap}_{\text{YES}} \\
(C, k) \in \textsc{Circuit-Count-Gap}_{\text{NO}} &\implies (\varphi', k') \in \textsc{Monotone-2-Sat-Count-Gap}_{\text{NO}}
\end{aligned}
$$

Each vertex cover of G, say S of size s, gives rise to $(2^{10m} - 1)^{7m-s}$ vertex covers of G' as follows: For each cloud representing a vertex in S, include all vertices of the cloud, and for each cloud representing a vertex not in S, include any subset of the cloud except the entire cloud. These are indeed vertex covers of G', and every vertex cover of G' can be obtained in this way. Hence the vertex covers of G' are partitioned according to the vertex cover of G they correspond to. The total number of vertex covers of G', and hence the total number of satisfying assignments of φ', is thus

$$
\sum_s (2^{10m} - 1)^{7m-s} \cdot (\text{number of vertex covers of } G \text{ of size } s).
$$

For the YES case, the assumption that C accepts $\geq k$ inputs implies that G has $\geq k$ vertex covers of size at most ℓ (by property (i)), which implies that the number of satisfying assignments of φ' is $\geq k \cdot (2^{10m} - 1)^{7m-\ell} = k'$. For the NO case, the assumption that C accepts $< k/4$ inputs implies that G has: 0 vertex covers of size $< \ell$ (by property (ii)), $< k/4$

vertex covers of size $= \ell$ (by property (i)), and trivially $\leq 2^{7m}$ vertex covers of size $> \ell$. Thus the number of satisfying assignments of φ' is

$$< \ (k/4) \cdot (2^{10m} - 1)^{7m-\ell} + 2^{7m} \cdot (2^{10m} - 1)^{7m-\ell-1} \ = \ k'/4 + k' \cdot 2^{7m}/k(2^{10m} - 1) \ < \ k'/2.$$

□

6.3 Proof of Theorem 6.5

First note that Logspace-Min-Ent-Gap $\in$ prNP since the most probable output string can be nondeterministically guessed, and then the probability of that string can be computed exactly by simple dynamic programming. The prNP-hardness follows without difficulty from a result of Lyngsø and Pedersen on hidden Markov models [184]; we now elaborate.

A hidden Markov model consists of a (time-invariant) Markov chain with a designated start state, where each state is either "silent" or has a distribution over symbols from some alphabet. Running the Markov chain for a certain number of steps yields a random output string whose length is the number of non-silent states visited. The result of [184] shows, by a clever reduction from Max-Clique, that it is NP-hard to estimate the probability of the most likely output string, even in the special case where the Markov chain is a DAG with a unique source and sink (which are the only silent states) and where each non-silent state deterministically outputs a bit. In fact, the result shows that the gap version of the estimation problem is prNP-hard with a multiplicative gap of $n^{\Omega(1)}$ (where n is the size of the Markov chain), by using the tight lower bounds on the approximability of Max-Clique (see [131] and the references within).

The hidden Markov model produced by the reduction in [184] may output bit strings of different lengths on different runs. We fix this and make the model conform to our definition of logarithmic-space samplers as follows. Letting m denote the length of a longest path in the DAG, take m copies of the DAG (except the source) and put each copy in a separate layer, which represents a time step. Retain all the original transitions but make them go between adjacent layers, and make the sink always transition to itself in the next layer. Now each non-source vertex outputs two bits: The copies of the sink output 00, and the copies of non-sinks output 1 followed by their original bit. We can then make each vertex output a single bit at the cost of doubling the number of layers. Each output string (of length $\leq m$) of the original model corresponds injectively to an output string of length $2m$ of this new sampler (which inserts 1's in every other position and then pads with 00's). The output distribution is the same, so in particular the highest probability of an output is the same. Thus it is prNP-hard to estimate the min-entropy with an additive gap of $\Omega(\log n)$ (which is stronger than the gap of 1 stated in Theorem 6.5).

Interestingly, it is also shown in [184] that for logarithmic-space samplers, estimating the statistical distance between two distributions is closely related to estimating the *min-entropy* of a distribution. In contrast, for polynomial-size circuits it is known that estimating the statistical distance is closely related to estimating the *Shannon entropy* (see [251]).

6.4 SBP is Closed Under Nondeterminism

Consider the following nondeterministic generalization of Circuit-Count-Gap: Given (C, k) where C is a circuit that takes two inputs y and z, does there exist a y for which C accepts $\geq k$ strings z, or is it the case that for all y, C accepts $< k/2$ strings z? In showing that Circuit-Min-Ent-Gap $\in$ prSBP, we implicitly showed that the above promise problem is in prSBP, by the standard trick of amplifying the gap. We now observe that it remains in prSBP *even if we allow the location of the gap to depend on the nondeterministic guess.*

Definition 6.10. prNSBP *is the class of promise problems* $\Pi = (\Pi_{\text{YES}}, \Pi_{\text{NO}})$ *for which there exist polynomial-time algorithms* M, K *(where* M *outputs a bit and* K *outputs a nonnegative integer) and a polynomial* p *such that the following hold for all* $x \in \{0,1\}^*$.

$$\begin{aligned} x \in \Pi_{\text{YES}} &\implies \exists y \in \{0,1\}^{p(|x|)} \ : \ \big|\{z \in \{0,1\}^{p(|x|)} \ : \ M(x,y,z) = 1\}\big| \geq K(x,y) \\ x \in \Pi_{\text{NO}} &\implies \forall y \in \{0,1\}^{p(|x|)} \ : \ \big|\{z \in \{0,1\}^{p(|x|)} \ : \ M(x,y,z) = 1\}\big| < K(x,y)/2 \end{aligned}$$

NSBP *is defined as the class of languages in* prNSBP*.*

Analogously to Circuit-Count-Gap, it is possible to define a promise problem such that prNSBP is the class of all promise problems reducible to that problem.

Theorem 6.11. prNSBP = prSBP *and thus* NSBP = SBP*.*

The basic idea is to modify the computation so the number of accepted z's (for a given y) is multiplied by an efficiently computable factor, so as to shift the threshold to be close to some value that does not depend on y (indeed, does not even depend on x). Then as before we can use the amplification trick so that the gap swamps out the effect of the nondeterministic y. However, there is a slight wrinkle to iron out: If $K(x,y) = 0$ then we cannot shift the threshold by multiplying it by something. But in this case x is automatically a YES instance, so if we happen to observe $K(x,y) = 0$ then we can just accept while ignoring z.

Proof of Theorem 6.11. We reduce an arbitrary $\Pi \in$ prNSBP (with associated M, K, p) to Circuit-Count-Gap. Given $x \in \{0,1\}^n$, we let p denote $p(n)$ and we assume without loss of generality that $0 \leq K(x,y) \leq 2^p + 1$ for all $y \in \{0,1\}^p$. We construct a circuit $C : \{0,1\}^p \times \big(\{0,1\}^p \times \{0,1\}^{p+3}\big)^{2p} \to \{0,1\}$ by

$$C\big(y, (z_i, w_i)_{i=1}^{2p}\big) \ = \ \begin{cases} 1 & \text{if } \ K(x,y) = 0 \ \vee \ \forall i \ \Big[M(x,y,z_i) = 1 \ \wedge \ w_i < \lceil 2^{p+3}/K(x,y) \rceil\Big] \\ 0 & \text{otherwise} \end{cases}$$

where the w_i's are viewed as binary integers, and we let $k = (2^{p+3})^{2p}$. We show the following two things.

$$\begin{aligned} x \in \Pi_{\text{YES}} &\implies (C,k) \in \text{Circuit-Count-Gap}_{\text{YES}} \\ x \in \Pi_{\text{NO}} &\implies (C,k) \in \text{Circuit-Count-Gap}_{\text{NO}} \end{aligned}$$

For the YES case, consider the good y from Definition 6.10. If $K(x,y) = 0$ then C accepts $y, (z_i, w_i)_{i=1}^{2p}$ for every choice of $(z_i, w_i)_{i=1}^{2p}$, and thus the total number of accepted inputs is $\geq (2^{2p+3})^{2p} \geq k$. On the other hand, assume $K(x,y) > 0$. Since $M(x,y,z_i) = 1$ holds for $\geq K(x,y)$ choices of z_i, and $w_i < \lceil 2^{p+3}/K(x,y) \rceil$ holds for $\lceil 2^{p+3}/K(x,y) \rceil$ choices of w_i, the conjunction of these holds for $\geq K(x,y) \cdot \lceil 2^{p+3}/K(x,y) \rceil \geq 2^{p+3}$ choices of z_i, w_i. Hence C accepts $y, (z_i, w_i)_{i=1}^{2p}$ for $\geq (2^{p+3})^{2p} = k$ choices of $(z_i, w_i)_{i=1}^{2p}$, and thus the total number of accepted inputs is also $\geq k$. For the NO case, consider an arbitrary y. We must have $K(x,y) > 0$. Since $M(x,y,z_i) = 1$ holds for $< K(x,y)/2$ choices of z_i, and $w_i < \lceil 2^{p+3}/K(x,y) \rceil$ holds for $\lceil 2^{p+3}/K(x,y) \rceil$ choices of w_i, the conjunction of these holds for $< (K(x,y)/2) \cdot \lceil 2^{p+3}/K(x,y) \rceil \leq \big(2^{p+3} + K(x,y)\big)/2 \leq 2^{p+3} \cdot (5/8)$ choices of z_i, w_i (using $K(x,y) \leq 2^p + 1$). Hence C accepts $y, (z_i, w_i)_{i=1}^{2p}$ for $< \big(2^{p+3} \cdot (5/8)\big)^{2p}$ choices of $(z_i, w_i)_{i=1}^{2p}$. Summing over y, the total number of accepted inputs is $< 2^p \cdot \big(2^{p+3} \cdot (5/8)\big)^{2p} = k \cdot (25/32)^p < k/2$. $\square$

Part IV

The Algorithm's Input is Random

Chapter 7

Relativized Worlds Without Worst-Case to Average-Case Reductions for NP

7.1 Introduction

The study of average-case complexity concerns the power of algorithms that are allowed to make mistakes on a small fraction of inputs. Of particular importance is the relationship between worst-case complexity and average-case complexity. For example, cryptographic applications require average-case hard problems, and it would be desirable to base the existence of such problems on minimal, worst-case complexity assumptions.

For the class PSPACE, it is known that worst-case hardness and average-case hardness are equivalent [22]. That is, if PSPACE is worst-case hard then it is also average-case hard. For the class NP, the situation is not well-understood. A central open problem in average-case complexity is to prove that if NP is worst-case hard then it is also average-case hard. Considering the lack of progress toward proving this proposition, a natural goal is to exhibit barriers to proving it, by ruling out certain general proof techniques. Bogdanov and Trevisan [46] considered the possibility of a *proof by reduction*. Building on [84], they showed that the proposition cannot be proven by a nonadaptive randomized reduction unless the polynomial-time hierarchy collapses; it remains open to provide evidence against the existence of adaptive reductions.[1] Another possibility that has been considered is a *relativizing proof*. In 1995, Impagliazzo and Rudich claimed [134] that they had constructed a relativized heuristica, which is a world in which NP is worst-case hard but average-case easy, thus ruling out this possibility. However, they have since retracted their claim. We make progress toward obtaining relativized heuristica, by ruling out the possibility of a *relativizing proof by reduction*. Our barrier holds even for adaptive reductions. More formally, we prove that

[1]It can be shown without difficulty that there is no *deterministic* adaptive worst-case to average-case reduction for NP unless P = NP.

there exists an oracle relative to which there is no reduction of type

$$\big(\mathrm{NP}, \mathrm{PS{\small AMP}}\big) \subseteq \mathrm{HeurBPP} \;\Rightarrow\; \mathrm{NP} \subseteq \mathrm{BPP}$$

where $\big(\mathrm{NP}, \mathrm{PS{\small AMP}}\big)$ is the class of distributional NP problems under polynomial-time samplable distributions, and HeurBPP is the class of distributional problems with polynomial-time average-case randomized algorithms.

We also generalize this result in various ways. The proposition that if NP is worst-case hard then it is also average-case hard concerns average-case algorithms that may output the wrong answer on a small fraction of inputs. In light of the aforementioned barriers, it is natural to consider the following proposition, which is potentially easier to prove: If NP is worst-case hard then it is also hard for *errorless* average-case algorithms, which may output "don't know" on a small fraction of inputs but must never output the wrong answer.[2] Our result generalizes to rule out relativizing proofs by reduction of this proposition. Further, we show how to rule out relativizing proofs by reduction that if NP is worst-case hard then certain classes larger than NP are errorless-average-case hard.

Independently of our work, Impagliazzo [135] has succeeded in constructing a relativized heuristica; we discuss his result in Section 7.1.3 below.

7.1.1 Notions of Reductions and Relationship to Previous Work

Various models of worst-case to average-case reductions for NP have been considered in the literature, and they can be informally classified as follows.

For the moment let us gloss over the issue of which distribution on inputs an average-case algorithm is judged with respect to. A worst-case to average-case reduction for NP must show that for every $L_1 \in \mathrm{NP}$ there exists an $L_2 \in \mathrm{NP}$ such that if L_2 has a polynomial-time average-case algorithm then L_1 has a polynomial-time worst-case algorithm. The worst-case algorithm for L_1 depends on the hypothesized average-case algorithm for L_2 in some way, which we call the *decoding*. There are the following four natural types of dependence, in decreasing order of strength.

(1) Black-box dependence means that the worst-case algorithm for L_1 has oracle access to the average-case algorithm for L_2, and it must solve L_1 on all inputs for *every* oracle that solves L_2 on most inputs, regardless of whether the oracle represents an efficient algorithm.

(2) The worst-case algorithm for L_1 might have oracle access to the average-case algorithm for L_2 but only be guaranteed to solve L_1 when the oracle is, in fact, an efficient average-case algorithm for L_2.

(3) The worst-case algorithm for L_1 might require the code of an efficient average-case algorithm for L_2.

[2] An equivalent notion of an errorless average-case algorithm is one that always outputs the correct answer but whose running time is only "polynomial-on-average" [165].

(4) The dependence can be arbitrary, meaning that if L_2 has an efficient average-case algorithm then L_1 has an efficient worst-case algorithm. This type of dependence allows for arbitrary proofs that if NP is worst-case hard then it is also average-case hard.

For the first three types, the algorithm that solves L_1 with the aid of a hypothesized average-case algorithm for L_2 is called the reduction itself. In this chapter we consider type (1) decoding. Note that since our results are about relativization, the reductions we consider have access to two oracles: the reduction oracle (representing the hypothesized average-case algorithm) and the relativization oracle.

Bogdanov and Trevisan [46] also considered type (1) decoding. They showed that such a reduction cannot exist unless the polynomial-time hierarchy collapses, provided the reduction is nonadaptive in its oracle access to the hypothesized average-case algorithm. Compared to the Bogdanov-Trevisan barrier, our barrier has the advantages that it is unconditional and it applies to adaptive reductions, but has the disadvantage that it only applies to reductions that relativize.

Gutfreund et al. [124] showed a *positive* result, namely that there is a worst-case to average-case reduction for NP with type (2) decoding, under a distribution on inputs that is samplable in slightly-superpolynomial time. Building on this result, Gutfreund and Ta-Shma [125] showed that under a certain weak derandomization hypothesis, there is a worst-case to average-case reduction from NP to nondeterministic slightly-superpolynomial time with type (2) decoding, under the uniform distribution on inputs. Moreover, the results of [124, 125] relativize.

A natural goal is to extend our results to handle type (2) decoding. However, this turns out to be as hard as extending our results to handle type (4) decoding (which was independently accomplished by Impagliazzo [135], at least for NP). For example, we claim that relative to every oracle, the following are equivalent.

(A) There is no reduction of type

$$\big(\mathrm{NP}, \mathrm{PS{\scriptstyle AMP}}\big) \subseteq \mathrm{HeurBPP} \;\Rightarrow\; \mathrm{NP} \subseteq \mathrm{BPP}$$

with type (2) decoding.

(B) $\big(\mathrm{NP}, \mathrm{PS{\scriptstyle AMP}}\big) \subseteq \mathrm{HeurBPP}$ and $\mathrm{NP} \not\subseteq \mathrm{BPP}$.

Clearly (B) implies (A). To see that (A) implies (B), consider two cases. If $\mathrm{NP} \subseteq \mathrm{BPP}$, then there is a trivial reduction that ignores the hypothesized HeurBPP algorithm for $\big(\mathrm{NP}, \mathrm{PS{\scriptstyle AMP}}\big)$. If $\big(\mathrm{NP}, \mathrm{PS{\scriptstyle AMP}}\big) \not\subseteq \mathrm{HeurBPP}$, then there is some problem in $\big(\mathrm{NP}, \mathrm{PS{\scriptstyle AMP}}\big)$ for which every algorithm is vacuously an appropriate type (2) decoder, because the universal quantification over HeurBPP algorithms for that problem is over an empty set.

The classes P^{PP} and PSPACE both have an $O(n)$-query worst-case to average-case reduction under the uniform distribution, with type (1) decoding, using multilinear extensions [22]. Moreover, these results relativize.

For certain promise problems regarding lattices, e.g., certain versions of the shortest vector problem, worst-case to average-case reductions to problems in $\big(\mathrm{NP}, \mathrm{PS{\small AMP}}\big)$ are known, with type (1) decoding. However, these lattice problems are not known or believed to be NP-hard. We refer to [45, 217] for surveys of these results.

Another aspect of worst-case to average-case reductions is the *encoding*, which refers to the way in which L_2 depends on L_1. Black-box encoding means that the algorithm that defines L_2 has oracle access to L_1, and for *every* language L_1 (not just those in NP), if the corresponding L_2 has an efficient average-case algorithm then L_1 has an efficient worst-case algorithm (via one of the above four types of decoding).

Viola [257, 258] proved two results about worst-case to average-case reductions with black-box encoders implementable in the polynomial-time hierarchy. In [257] he proved unconditionally that such a reduction with type (1) decoding does not exist. In [258] he proved that if such a reduction with type (4) decoding exists then PH is average-case hard, and thus basing the average-case hardness of PH on the worst-case hardness of PH in this way is no easier than unconditionally proving the average-case hardness of PH.

7.1.2 Results

Our first result concerns the class $\mathrm{BPP}_{\parallel}^{\mathrm{NP}}$. Recall that AvgZPP denotes the class of distributional problems with polynomial-time errorless average-case randomized algorithms.

Theorem 7.1. *There exists an oracle relative to which there is no reduction of type*

$$\big(\mathrm{BPP}_{\parallel}^{\mathrm{NP}}, \mathrm{PS{\small AMP}}\big) \subseteq \mathrm{AvgZPP} \;\Rightarrow\; \mathrm{UP} \subseteq \mathrm{BPP}.$$

Note that the type of reduction considered in Theorem 7.1 is weaker than a worst-case to average-case reduction for NP, because $\mathrm{BPP}_{\parallel}^{\mathrm{NP}}$ is larger than NP, AvgZPP is smaller than HeurBPP, and UP is smaller than NP. Ruling out weaker reductions yields a stronger result.

If we restrict our attention to reductions that use a limited number of queries, then we can handle classes even larger than $\mathrm{BPP}_{\parallel}^{\mathrm{NP}}$.

Theorem 7.2. *For every polynomial q there exists an oracle relative to which there is no q-query reduction of type*

$$\big(\mathrm{PH}, \mathrm{PS{\small AMP}}\big) \subseteq \mathrm{AvgZPP} \;\Rightarrow\; \mathrm{UP} \subseteq \mathrm{BPP}.$$

Since $\mathrm{BPP}_{\parallel}^{\mathrm{NP}} \subseteq \mathrm{PH}$ holds relative to every oracle, it may appear at first glance that Theorem 7.2 subsumes Theorem 7.1. The reason it does not is because of the order of the quantifiers. In Theorem 7.2, the reduction may not make as many queries as it likes; it may only make a fixed polynomial q number of queries even though its running time may be an arbitrarily high degree polynomial.

If we are willing to sacrifice all but two queries, then we can go quite a bit further than PH.

Theorem 7.3. *For every uniform complexity class of languages $\mathcal{C}$ there exists an oracle relative to which there is no 2-query reduction of type*

$$\big(\mathcal{C}, \text{PS}\textsc{amp}\big) \subseteq \text{AvgZPP} \;\Rightarrow\; \text{UP} \subseteq \text{BPP}.$$

The term "uniform complexity class of languages" has a somewhat technical meaning, which is explained in Section 7.2, but it encompasses all "ordinary" complexity classes such as PSPACE and EXP^{EXP}.

Our theorems can be generalized in various ways. For example, Theorem 7.1 and Theorem 7.2 both hold with AvgZPP replaced by the deterministic version AvgP, by essentially the same proofs.[3] We have chosen to state the results using AvgZPP because we feel it is more natural to allow randomized algorithms in average-case complexity. As another example, Theorem 7.1 holds with BPP replaced by BQP, by inserting a quantum query lower bound for the OR function [35] at the appropriate point in the argument, instead of a randomized lower bound. We have chosen the particular statements of our three theorems so as to highlight the interesting aspects and make the relationships among them clear.

In the original ECCC version of the paper this chapter is based on [265], we also proved two results similar to Theorem 7.1. One is a generalization of Theorem 7.1 where $\text{BPP}_{\|}^{\text{NP}}$ is generalized to allow multiple rounds of adaptivity in the NP oracle access (up to $o(n/\log n)$ rounds). In the other result, $\text{BPP}_{\|}^{\text{NP}}$ is replaced with the class BPP_{path}, which was introduced in [126] and which captures the power of polynomial-time randomized computations conditioned on efficiently testable events. Relative to every oracle, $\text{P}_{\|}^{\text{NP}} \subseteq \text{BPP}_{\text{path}} \subseteq \text{BPP}_{\|}^{\text{NP}}$, and thus this result is subsumed by Theorem 7.1. However, our proof of the BPP_{path} result is still interesting because the heart of the proof is genuinely different from the heart of our proof of Theorem 7.1, and it exploits the definition of BPP_{path} in a particularly intuitive way, without going through approximate counting. We refer the reader to [265] for details about these results.

7.1.3 Independent Work

Independently of our work, Impagliazzo [135] has succeeded in constructing a relativized heuristica, even for errorless average-case algorithms. In fact, he constructs an oracle relative to which $\big(\text{NP}, \text{PS}\textsc{amp}\big) \subseteq \text{AvgP}$ but $\text{UP} \not\subseteq \text{P/poly}$. Thus relative to his oracle, there is no worst-case to average-case reduction for NP with any of the four types of decoding discussed in Section 7.1.1. This subsumes our result for NP (which only applies to black-box decoding).

The results of [135] do not subsume our results for classes higher than NP, although Impagliazzo conjectures that this may be possible using his techniques. Furthermore, our techniques can be adapted without difficulty (though we do not argue this here) to show that there exists an oracle relative to which there is no reduction of type

$$\big(\text{NP}, \text{PS}\textsc{amp}\big) \subseteq \text{HeurBPP} \;\Rightarrow\; \big(\text{NP}, \text{PS}\textsc{amp}\big) \subseteq \text{AvgZPP}$$

[3]For Theorem 7.1 exactly the same proof works; for Theorem 7.2 a minor tweak is needed.

while it remains open to prove that there exists an oracle relative to which $\big(\text{NP}, \text{PS}\textsc{amp}\big) \subseteq$ HeurBPP but $\big(\text{NP}, \text{PS}\textsc{amp}\big) \not\subseteq$ AvgZPP.

Another benefit of this chapter is that our techniques are genuinely different (and more elementary) than Impagliazzo's. The outline of his argument is that he puts a random permutation into the oracle so that inverting the permutation is a worst-case hard NP problem, and to make NP average-case easy he puts the answers to all NP problems into the oracle but "censors" a certain small fraction of the answers in a way that preserves the worst-case hardness of inverting the permutation. Ensuring consistency between the conflicting requirements is a delicate business, and involves results on random restrictions of so-called matching DNFs. In contrast, our proof of our result for NP does not use such machinery. While much of the work in Impagliazzo's proof is in the analysis rather than the construction of the oracle, our argument is more directly adversarial. We use a potential function technique to guide the construction to converge to an oracle with the desired properties. We employ elementary counting arguments to achieve this.

Also, our proofs of Theorem 7.2 and Theorem 7.3 illustrate a new connection between lower bounds for error-correcting codes and relativized lower bounds.

7.1.4 Organization

In Section 7.2 we provide preliminaries, which clarify the precise meanings of our theorems. In Section 7.3 we give the intuition for our proofs. In Section 7.4 we describe the basic setup that is common to the formal proofs of all three theorems. Section 7.5 contains the formal proof of Theorem 7.1. Section 7.6 contains the formal proof of Theorem 7.2. Section 7.7 contains the formal proof of Theorem 7.3.

7.2 Preliminaries

We refer the reader to the textbooks [17, 98] for background on complexity theory and definitions of standard complexity classes. We refer the reader to the survey paper [45] for background on average-case complexity. In this section we provide preliminaries that are not completely standard.

7.2.1 Complexity Classes

For any randomized algorithm M, we let M_r denote M using internal randomness r.

We now define the average-case complexity classes we need. In average-case complexity, we study distributional problems (L, D) where L is a language and $D = (D_1, D_2, \ldots)$ is an ensemble of probability distributions, where D_n is distributed over $\{0,1\}^n$. Recall that PSAMP denotes the class of polynomial-time samplable ensembles, and $\mathcal{U}$ denotes the class consisting of only the uniform ensemble U. If $\mathcal{C}$ is a class of languages and $\mathcal{D}$ is a class of ensembles then $(\mathcal{C}, \mathcal{D}) = \big\{(L, D)\ :\ L \in \mathcal{C} \text{ and } D \in \mathcal{D}\big\}$.

Definition 7.4. HeurBPP *denotes the class of distributional problems (L, D) that have a polynomial-time heuristic scheme, that is, a randomized algorithm M that takes as input x and $\delta > 0$, runs in time polynomial in $|x|$ and $1/\delta$, and for all n and all $\delta > 0$ satisfies*

$$\Pr_{x \sim D_n, r} \left[M_r(x, \delta) \neq L(x) \right] \leq \delta.$$

Definition 7.5. AvgZPP *denotes the class of distributional problems (L, D) that have a polynomial-time errorless heuristic scheme, that is, a randomized algorithm M that takes as input x and $\delta > 0$, runs in time polynomial in $|x|$ and $1/\delta$, always outputs $L(x)$ or $\perp$, and for all n and all $\delta > 0$ satisfies*

$$\Pr_{x \sim D_n, r} \left[M_r(x, \delta) = \perp \right] \leq \delta.$$

7.2.2 Reductions

In this section we informally explain what we mean when we say there exists a reduction of type

$$\mathcal{C}_2' \subseteq \mathcal{C}_2 \;\Rightarrow\; \mathcal{C}_1' \subseteq \mathcal{C}_1$$

where $\mathcal{C}_2', \mathcal{C}_2, \mathcal{C}_1', \mathcal{C}_1$ are four complexity classes. In Section 7.2.3 below we give formal definitions for the specific classes to which our theorems apply.

A complexity class is a set of computational problems, such as languages or distributional problems. We assume for concreteness that each of $\mathcal{C}_1$ and $\mathcal{C}_2$ is defined in the following way. By an *input-output relationship* we mean a randomized function. There is a set of algorithms, each of which induces an input-output relationship. That is, each algorithm takes an input and produces an output sampled from some distribution depending on the input. There is a predicate that indicates for each input-output relationship and each computational problem whether the input-output relationship *solves* the problem. There is a notion of computational resources used by the algorithms, and an algorithm is said to be *efficient* if it satisfies certain resource constraints. The class is defined as the set of problems solved by efficient algorithms. This type of definition encompasses classes defined in terms of (uniform or nonuniform) deterministic, randomized, or quantum algorithms, but it could be generalized to handle other models as well.

We also assume that for $\mathcal{C}_1$ there is an analogous set of algorithms that can make queries to a *reduction oracle*, which represents an input-output relationship.[4] We assume that plugging any algorithm from $\mathcal{C}_2$'s set into the reduction oracle yields an algorithm from $\mathcal{C}_1$'s set.

Now suppose P_1 is a computational problem of the appropriate kind for $\mathcal{C}_1$ and P_2 is a computational problem of the appropriate kind for $\mathcal{C}_2$.

Definition 7.6. *A reduction of type*

$$P_2 \in \mathcal{C}_2 \;\Rightarrow\; P_1 \in \mathcal{C}_1$$

[4]In particular, the reduction oracle is not like a relativization oracle, which just answers queries to a language.

is an algorithm from $\mathcal{C}_1$'s set of reduction oracle algorithms, such that (i) for every reduction oracle that solves P_2 according to $\mathcal{C}_2$, the reduction solves P_1 according to $\mathcal{C}_1$, and (ii) for every reduction oracle, the reduction satisfies $\mathcal{C}_1$'s resource constraints if we pretend each query to the reduction oracle uses any amount of resources allowed by $\mathcal{C}_2$'s resource constraints.

In other words, if the reduction oracle is correct then the reduction is correct, and if we pretend the reduction oracle is efficient then the reduction is efficient.

Note that if we plug an actual, efficient algorithm for P_2 (according to $\mathcal{C}_2$) into the reduction oracle of such a reduction, then the reduction becomes an efficient algorithm for P_1 (according to $\mathcal{C}_1$). Thus if there exists a reduction satisfying Definition 7.6 then $P_2 \in \mathcal{C}_2$ implies $P_1 \in \mathcal{C}_1$. But the reduction must work even when the reduction oracle is an input-output relationship that is not efficiently implementable.

As an example, suppose $\mathcal{C}_2 = \text{BPTIME}(2^{n^\epsilon})$. Then the reduction must solve P_1 according to $\mathcal{C}_1$ when the reduction oracle is any randomized function from $\{0,1\}^*$ to $\{0,1\}$ that, on input w, returns $P_2(w)$ with probability $\geq 2/3$.[5] Further, the reduction must satisfy the resource constraints of $\mathcal{C}_1$ when we pretend each query of length n to the reduction oracle takes time $O(2^{n^\epsilon})$.

Definition 7.7. *We say there exists a reduction of type*

$$\mathcal{C}_2' \subseteq \mathcal{C}_2 \;\Rightarrow\; \mathcal{C}_1' \subseteq \mathcal{C}_1$$

if for every $P_1 \in \mathcal{C}_1'$ there exists a $P_2 \in \mathcal{C}_2'$ and a reduction of type

$$P_2 \in \mathcal{C}_2 \;\Rightarrow\; P_1 \in \mathcal{C}_1.$$

We make a few remarks about Definition 7.7.

- When $\mathcal{C}_1'$ has an appropriately complete problem P_1, this is equivalent to saying there exists a $P_2 \in \mathcal{C}_2'$ and a reduction of the above type, for the fixed problem P_1.
- Note that we do not require that the reduction is uniform in the sense of there being a fixed algorithm R that computes the reduction for every $P_1 \in \mathcal{C}_1'$ given the code for a $\mathcal{C}_1'$-type algorithm for P_1.
- Note that when we say there is a reduction of the above type, this assertion gets weaker as $\mathcal{C}_2'$ and $\mathcal{C}_1$ get larger and $\mathcal{C}_2$ and $\mathcal{C}_1'$ get smaller.

[5]One might wonder about reductions that can also choose the randomness used by the reduction oracle. While this would be more general in one sense, it would be more restrictive in the sense that it would limit the randomness complexity of the reduction oracle. In this chapter, queries are always just inputs to an input-output relationship as defined above.

7.2.3 Relativization

When we relativize to an oracle language A, every computation gets unrestricted oracle access to A. This includes samplers and reductions. Thus reductions have access to two oracles: the reduction oracle and the relativization oracle. When we write $R^{B,A}$ we mean B is the reduction oracle and A is the relativization oracle for reduction R.

To illustrate the formal framework set up so far, we give the precise statement of Theorem 7.1. There exists a language A and a language $L_1 \in \mathrm{UP}^A$ such that for all languages $L_2 \in \left(\mathrm{BPP}_{\parallel}^{\mathrm{NP}}\right)^A$, all ensembles $D \in \mathrm{PS}\textsc{amp}^A$, and all polynomial-time randomized reductions $R^{\circ,\circ}$, $R^{\circ,A}$ is not of type

$$(L_2, D) \in \mathrm{AvgZPP}^A \;\Rightarrow\; L_1 \in \mathrm{BPP}^A.$$

The latter means that there exists an $x \in \{0,1\}^*$ and a randomized function $B : \{0,1\}^* \times \mathbb{R}_{>0} \to \{0,1,\perp\}$ which is a valid AvgZPP oracle for (L_2, D), such that

$$\Pr_{r,B}\left[R_r^{B,A}(x) = L_1(x)\right] \;<\; 2/3$$

where the probability is over both the internal randomness of R and the randomness of B (each query is answered with fresh independent randomness). When we say B is a valid AvgZPP oracle for (L_2, D) we mean that $B(w,\delta)$ always returns $L_2(w)$ or $\perp$, and for all n and all $\delta > 0$,

$$\Pr_{w\sim D_n,B}\left[B(w,\delta) = \perp\right] \;\le\; \delta.$$

When we say $R^{\circ,\circ}$ runs in polynomial time, this includes the fact that each query $B(w,\delta)$ to the reduction oracle is charged time polynomial in $|w|$ and $1/\delta$. In other words, δ must always be at least inverse polynomial. Throughout this chapter we tacitly assume that "polynomial-time reductions" have this restriction, since $\mathcal{C}_2$ is always AvgZPP. We clarify that $D \in \mathrm{PS}\textsc{amp}^A$ means that for some randomized algorithm S°, $S^A(n)$ runs in time polynomial in n and outputs a sample distributed according to D_n. Finally, we clarify that $\left(\mathrm{BPP}_{\parallel}^{\mathrm{NP}}\right)^A$ is the class of languages L_2 for which there exists a language $L_3 \in \mathrm{NP}^A$ and a polynomial-time randomized algorithm $M^{\circ,\circ}$ that accesses its first oracle nonadaptively, such that for all $x \in \{0,1\}^*$,

$$\Pr_r\left[M_r^{L_3,A}(x) = L_2(x)\right] \;\ge\; 2/3.$$

Regarding Theorem 7.2 and Theorem 7.3, there is one further issue to consider. For reductions that are allowed an unlimited number of queries (like in Theorem 7.1), the error probability of 1/3 in the definition of BPP is unimportant since it can be amplified from $1/2 - 1/\operatorname{poly}(n)$ to $1/2^{\operatorname{poly}(n)}$. However, amplification increases the number of queries, so the error probability is not arbitrary for Theorem 7.2 and Theorem 7.3. For example, the existence of a q-query $\big(1/2 - 1/\operatorname{poly}(n)\big)$-error reduction of type

$$\big(\mathrm{PH}, \mathrm{PS}\textsc{amp}\big) \subseteq \mathrm{AvgZPP} \;\Rightarrow\; \mathrm{UP} \subseteq \mathrm{BPP}$$

does not seem to imply the existence of a q-query 1/3-error reduction of the same type, but it still does imply that if $\big(\mathrm{PH}, \mathrm{PS{\scriptsize AMP}}\big) \subseteq \mathrm{AvgZPP}$ then $\mathrm{UP} \subseteq \mathrm{BPP}$. For this reason, we allow an error probability of $1/2 - 1/\operatorname{poly}(n)$ (for arbitrarily high degree polynomials) in Theorem 7.2 and in Theorem 7.3.

7.2.4 Uniform Complexity Classes

We now precisely define the restriction on $\mathcal{C}$ in Theorem 7.3.

Definition 7.8. *We say that $\mathcal{C}$ is a* uniform complexity class of languages *if there is a countable collection of functions $\{M_1, M_2, \ldots\}$ mapping oracle languages A to languages M_i^A, such that the following three conditions all hold.*

- *For every i and every x, $M_i^A(x)$ only depends on a finite number of bits of A.*
- *For every i and every x there exists a property $P_{i,x}(A)$ that only depends on the bits of A that $M_i^A(x)$ depends on, such that $\mathcal{C}^A = \big\{M_i^A \ : \ \forall x \ P_{i,x}(A)\big\}$.*
- *For every i and every linear-time computable function $f : \{0,1\}^* \to \{0,1\}^*$ there exists a j such that for all A the following two conditions hold: $M_j^A = M_i^A \circ f$, and if $M_i^A \in \mathcal{C}^A$ then $M_j^A \in \mathcal{C}^A$.*

The second condition says the class is defined by a property of the computation (for example, bounded error) holding for all inputs. The third condition says the class is closed under linear-time deterministic mapping reductions. Observe that $\mathrm{BPP}_{\|}^{\mathrm{NP}}$, PH, PSPACE, and $\mathrm{EXP}^{\mathrm{EXP}}$ are all examples of uniform complexity classes under this definition.

7.3 Intuition

In Section 7.3.1 we describe the intuition behind the proof of Theorem 7.1. Then in Section 7.3.2 we describe the intuition behind the proofs of Theorem 7.2 and Theorem 7.3.

7.3.1 Intuition for Theorem 7.1

We start by informally describing how to construct an oracle relative to which there is no reduction of type

$$\big(\mathrm{NP}, \mathcal{U}\big) \subseteq \mathrm{HeurBPP} \ \Rightarrow \ \mathrm{UP} \subseteq \mathrm{BPP}.$$

To obtain Theorem 7.1 we must strengthen HeurBPP to AvgZPP,[6] strengthen $\mathcal{U}$ to PSAMP, and strengthen NP to $\mathrm{BPP}_{\|}^{\mathrm{NP}}$. We describe how to do these things below.

Fix an arbitrary NP-type algorithm M and an arbitrary polynomial-time randomized reduction R, and fix a sufficiently large n. For simplicity we assume that on inputs of length

[6]Usually AvgZPP is thought of as being a weaker class than HeurBPP (since AvgZPP $\subseteq$ HeurBPP), but it is stronger in our situation.

n, R only queries the reduction oracle on inputs of length n^d (for some positive integer d) and only with some fixed polynomially small δ; thus we can omit the δ from the queries. We consider relativization oracles of the form $A : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, which we think of as $2^n \times 2^n$ tables. Let $L_1^A : \{0,1\}^n \to \{0,1\}$ be defined by $L_1^A(x) = \bigvee_y A(xy)$. That is, L_1^A is the language of strings x such that there exists a 1 in the x^{th} row of A. Let $L_2^A : \{0,1\}^{n^d} \to \{0,1\}$ denote the language computed by M^A. We only consider A, L_1^A, L_2^A at these input lengths since all other input lengths are irrelevant.

We explain how to diagonalize against the pair M, R. We wish to construct an A such that for some $x \in \{0,1\}^n$ and some deterministic[7] reduction oracle $B : \{0,1\}^{n^d} \to \{0,1\}$, B agrees with L_2^A on at least a $1-\delta$ fraction of inputs and $R^{B,A}(x)$ outputs $L_1^A(x)$ with probability $< 2/3$. This will show that R fails to be a reduction of type

$$\left(L_2^A, U\right) \in \text{HeurBPP}^A \;\Rightarrow\; L_1^A \in \text{BPP}^A.$$

We also need to ensure that there is at most one 1 in each row of A so that $L_1^A \in \text{UP}^A$, but this will fall right out of the construction. We construct A through an iterative process, and we use a potential function argument to show that this process makes steady progress toward our goal. The process iteratively modifies the relativization oracle, and we use A to denote the relativization oracle throughout the whole process.[8] Thus the table denoted by A changes many times throughout our argument, and the languages L_1^A and L_2^A change accordingly. Initially A is all 0's.

Let us consider the computation of R on some input x. It is trying to figure out whether there is a 1 in the x^{th} row of A, in other words, compute $L_1^A(x)$. It has two sources of information about $L_1^A(x)$: the relativization oracle A itself, and the reduction oracle B. If R did not have access to B, then we could diagonalize in a standard way: Observe how R behaves given that the x^{th} row of A is all 0's. If R outputs 1 with high probability, then we are done. If R outputs 1 with low probability, then we find a bit in the x^{th} row that R queries with only tiny probability and flip that bit (such a bit must exist because R does not have enough time to keep an eye on the entire row); then R still outputs 1 with low probability, but now $x \in L_1^A$. Thus R must rely on the reduction oracle B for help.

Our construction has two stages. The goal of stage 1 is to gain the upper hand by rendering B useless to R. Then in stage 2 we deliver the coup de grâce with the standard diagonalization argument. We cannot guarantee that B is useless for every x, but we only need it to be useless for some x. Specifically, suppose we could set up A in such a way that there exists an x such that

(1) the x^{th} row of A is all 0's, and

(2) for all y, flipping $A(xy)$ would cause $L_2^A(w)$ to change for at most a δ fraction of w's.

[7] B will be deterministic here even though randomness is allowed; this makes the result stronger.

[8] More formally, we could say we define a sequence of relativization oracles $A_0, A_1, A_2, \ldots$ that leads to some final version $A_k = A$. We omit the subscripts throughout the argument and simply refer to A with the understanding that this means the "current" version.

Then declaring B to be L_2^A for the particular A we have set up, we know that we can leave A alone or we can flip any bit in the x^{th} row, and for all these possibilities B is a valid HeurBPP oracle for the new L_2^A. Then we can observe the behavior of R on input x, using this fixed B for the reduction oracle, and diagonalize against R in the standard way with the assurance that whatever happens to A during this second stage, B will remain valid.

How do we set up A so that such an x exists? We do this iteratively. In each iteration, we find a certain x whose row is currently all 0's, which is our "best guess" for the good x. If condition (2) is satisfied for this x, then we are done. Otherwise, there is some column y that violates condition (2). Then we flip the bit $A(xy)$ to 1 and continue with the next iteration. We just need to show that there are $< 2^n$ iterations before we succeed. For this, we define a potential function Φ^A that assigns an energy value to A. The key is to show that if y violates condition (2) for our best guess x, then flipping $A(xy)$ must cause a significant decrease in potential. Since Φ^A must remain bounded, there cannot be too many iterations before M is beaten into submission and our best guess x works.

Let us hold off on the definition of Φ^A and focus on finding a best guess x. Our ultimate goal is to ensure that if we flip any bit in the x^{th} row, most of the inputs to L_2^A "don't notice". There is an asymmetry between inputs that are accepted by M^A and those that are rejected. If $w \in \{0,1\}^{n^d}$ is such that $M^A(w)$ rejects, then if *any* of the exponentially many computation paths "notices" a change in A, the whole computation could become accepting. However, if $M^A(w)$ accepts, then we can pick an arbitrary accepting computation path of $M^A(w)$ to be the "designated" one. Only polynomially many bits of A are queried by M on this path, and as long as none of these bits is flipped, w "won't notice" any change to A because $M^A(w)$ will still accept. In particular, there are only polynomially many x's such that $M^A(w)$ queries some bit in the x^{th} row on the designated path. Thus for every w with $L_2^A(w) = 1$, the vast majority of x have the property that flipping any bit in the x^{th} row does not cause $L_2^A(w)$ to change to 0. By an averaging argument, most x have the property that for most $w \in \{0,1\}^{n^d}$, flipping any bit in the x^{th} row does not cause $L_2^A(w)$ to change from 1 to 0. For the current A, there must exist an x with the latter property and such that the x^{th} row is all 0's, since (by induction) we know there are not very many x's with a 1 in their row currently. This is our best guess x.

We know that flipping any bit in the x^{th} row causes only a small fraction of all $w \in \{0,1\}^{n^d}$ to change from 1 to 0 under L_2^A. This is good, but it is only half the story. We would also like that flipping any bit in the x^{th} row causes only a small fraction of w's to change from 0 to 1. Suppose we budget a $\delta/2$ fraction of w's to change from 1 to 0, and a $\delta/2$ fraction to change from 0 to 1. Now if some y violates condition (2), then it must be the case that flipping $A(xy)$ causes at least a $\delta/2$ fraction of w's to change from 0 to 1. We want to define the potential function so that having w's change from 0 to 1 under L_2^A causes a decrease in potential. A natural choice is

$$\Phi^A = \Pr_{w \sim U_{n^d}} \Big[L_2^A(w) = 0 \Big].$$

Flipping $A(xy)$ causes at least a $\delta/2$ probability mass to leave the event $L_2^A(w) = 0$. However, as much as a $\delta/2$ probability mass could enter the event due to w's that change from 1 to

0, which could essentially cancel out the drop in potential from the w's that changed from 0 to 1! The solution is to change our budgeting. If we budget a $\delta/3$ fraction of w's to change from 1 to 0 and a $2\delta/3$ fraction to change from 0 to 1, then flipping $A(xy)$, where y violates condition (2), causes at least a $2\delta/3$ probability mass to leave the event, while at most a $\delta/3$ probability mass enters the event. Thus Φ^A goes down by at least $\delta/3$, and there are at most $3/\delta < 2^n$ iterations before our best guess x works. This concludes the argument.

Very roughly, the big picture is as follows. For an input that is accepted by M^A, it is easy to ensure that the answer under L_2^A does not change when we make modifications to A. For an input that is rejected by M^A, we cannot ensure that the answer does not change, but the point is that if it does change, then *we can ensure that it does not change again*, since the input is now accepted.

7.3.1.1 Intuition for Strengthening HeurBPP to AvgZPP

Let x denote our best guess at the end of stage 1. Suppose we knew that there exists a set $W \subseteq \{0,1\}^{n^d}$ of density at most δ such that for all $w \notin W$ and all y, flipping $A(xy)$ does not change $L_2^A(w)$. Then setting

$$B(w) \;=\; \begin{cases} L_2^A(w) & \text{if } w \notin W \\ \perp & \text{if } w \in W \end{cases} \tag{7.1}$$

where A is the relativization oracle at the end of stage 1, we would have that B is a valid AvgZPP oracle for L_2^A no matter whether we leave A alone or flip any bit in the x^{th} row. Then we could diagonalize in the standard way, by observing how R behaves on input x using this fixed B and the current A, and either leaving A alone or flipping some bit in the x^{th} row to make R output the wrong answer with high probability.

The existence of such a W is too much to ask for. However, this is only because we were trying to find a B that would remain a valid AvgZPP oracle for *all* of the $2^n + 1$ diagonalization options. We do not really need all these options. Let Y be an arbitrary fixed set of columns of size $|Y| = 4t$, where t is the running time of R on inputs of length n. Then running R on input x with any fixed B and the current A, there must be a $y \in Y$ such that $A(xy)$ gets queried with probability $\leq 1/4$. If R outputs 1 with probability $\leq 1/3$ then after flipping this $A(xy)$, R outputs 1 with probability $< 2/3$ and hence errs. Thus it suffices to have $4t + 1$ diagonalization options, namely leaving A alone or flipping some $A(xy)$ with $y \in Y$. Suppose we knew that there exists a set $W \subseteq \{0,1\}^{n^d}$ of density at most δ such that for all $w \notin W$ and all $y \in Y$, flipping $A(xy)$ does not change $L_2^A(w)$. Then defining B as in Equation (7.1), we could diagonalize by either leaving A alone or flipping $A(xy)$ for some $y \in Y$ with the assurance that whatever happens, B will remain valid.

Now the existence of such a W is *not* too much to ask for. Using the argument for the HeurBPP case with a small adjustment of parameters, we can ensure that flipping any bit in the x^{th} row causes $L_2^A(w)$ to change for at most a $\delta/4t$ fraction of w's. Then we can take W to be the set of all w such that there exists a $y \in Y$ such that flipping $A(xy)$ changes $L_2^A(w)$.

7.3.1.2 Intuition for Strengthening $\mathcal{U}$ to PSamp

There are two approaches: one that is direct, and one that uses a result of Impagliazzo and Levin [140]. Neither is difficult. We first describe the direct approach.

First, observe that if U_{n^d} were replaced by some other distribution on $\{0,1\}^{n^d}$ that is independent of A, then the whole argument above would carry through, just by replacing "fraction of w's" with "probability mass of w's" under this distribution. Now in addition to M and R, we need to worry about an arbitrary polynomial-time sampler S, and we need to ensure that B is a valid AvgZPP oracle for $\left(L_2^A, D^A\right)$, where D^A denotes the distribution sampled by $S^A(n^d)$. If S did not query A at all, then D^A would be independent of A and thus we could use the same argument, by the above observation. Two issues arise because S is allowed to query A. First, when we flip a bit during stage 1, this affects

$$\Phi^A = \Pr_{w \sim D^A}\left[L_2^A(w) = 0\right]$$

in terms of not only the event but also the distribution. Second, when we flip a bit during stage 2, this affects the distributional problem $\left(L_2^A, D^A\right)$ for which B needs to be a valid AvgZPP oracle, in terms of not only the language but also the distribution.

Handling these issues is just a matter of tweaking the argument to ensure that our modifications to A cause only small statistical deviations in D^A. Specifically, consider the beginning of an iteration of stage 1, and let D denote D^A for the current A (thus D is fixed and will not react to changes in A). Now suppose we choose our best guess x as before, but based on this distribution D. Then by the above argument we know that for every y, flipping $A(xy)$ would either cause

$$\Pr_{w \sim D}\left[L_2^A(w) = 0\right]$$

to go down by a significant amount, or cause $L_2^A(w)$ to change with only small probability over $w \sim D$. It can be shown that this is good enough for our purpose provided that for all y, flipping $A(xy)$ results in a D^A that is statistically very close to D. To ensure the latter, we choose our best guess x not only so that the x^{th} row is all 0's and flipping any bit in the x^{th} row only causes a small probability mass of $w \sim D$ to change from 1 to 0 under L_2^A, but also so that the probability $S^A(n^d)$ queries any bit in the x^{th} row is small. This is possible because the vast majority of x's satisfy the latter condition since S runs in polynomial time.

An alternative approach to handling PSamp uses a result due to Impagliazzo and Levin [140]. They proved that if $\mathcal{C}$ is a class of languages containing NP and satisfying certain simple closure properties, then relative to every oracle, there exists a reduction of type

$$\left(\mathcal{C}, \mathcal{U}\right) \subseteq \text{AvgZPP} \;\Rightarrow\; \left(\mathcal{C}, \text{PSamp}\right) \subseteq \text{AvgZPP}.$$

The proof of this result appears in [45, Section 5.2] and is based on a result of Impagliazzo and Luby on distributionally inverting one-way functions [141]. By composing this reduction with the hypothesized reduction, we can assume without loss of generality that the distributional problem we are reducing to uses the uniform ensemble. In the formal proof of Theorem 7.1, rather than use the Impagliazzo-Levin result we opt to directly handle the samplable ensembles because doing so makes the argument self-contained at only a slight cost in complicatedness.

7.3.1.3 Intuition for Strengthening NP to $\mathrm{BPP}_{\parallel}^{\mathrm{NP}}$

There exists a simple reduction of type

$$\big(\mathrm{NP}, \mathrm{PS{\small AMP}}\big) \subseteq \mathrm{HeurBPP} \;\Rightarrow\; \big(\mathrm{BPP}_{\parallel}^{\mathrm{NP}}, \mathrm{PS{\small AMP}}\big) \subseteq \mathrm{HeurBPP}$$

(and this fact relativizes). Thus if we consider HeurBPP rather than AvgZPP, then the result for $\mathrm{BPP}_{\parallel}^{\mathrm{NP}}$ follows from the result for NP by composing reductions. To handle AvgZPP, we directly adapt the NP argument to work for $\mathrm{BPP}_{\parallel}^{\mathrm{NP}}$. Let us revert from PSAMP to $\mathcal{U}$.

Instead of a single algorithm M we have a pair M, N where N is an NP-type algorithm and M is a polynomial-time randomized algorithm that accesses its first oracle nonadaptively. We let L_3^A denote the language computed by N^A, and we let L_2^A denote the language computed by $M^{L_3^A, A}$ (assuming bounded error is satisfied for every input).[9] Let us make the simplifying assumption that M has oracle access only to L_3^A and not to A. (Extending the argument to the general case is not difficult; it just involves taking an extra precaution when picking our best guess x to ensure that hardly any w's "notice" changes to A via the second oracle.)

The differences from the above proof are in the definition of the potential function Φ^A, the choice of our best guess x, and the argument that if some y violates condition (2) for our best guess x, then flipping $A(xy)$ causes a significant decrease in potential. Let $M_r^{L_3^A}(w)_j \in \{0,1\}^*$ denote the jth query to L_3^A made by the computation $M_r^{L_3^A}(w)$ (in our simplified setting, this query does not depend on A), and consider the bits

$$L_3^A\Big(M_r^{L_3^A}(w)_j\Big)$$

over the choice of w, r, j. We define Φ^A to be the fraction of these bits that are 0.

Consider an arbitrary iteration of stage 1, and let A denote the current relativization oracle. By choosing our best guess x appropriately, we can ensure that the xth row of A is all 0's, and no matter what y is, only a tiny fraction of the w, r, j bits go from 1 to 0 when we flip $A(xy)$ (and thus bits going from 1 to 0 can only contribute a tiny increase in potential). Suppose there is a y such that flipping $A(xy)$ causes $L_2^A(w)$ to change for a significant fraction of w's. We want it to be the case that flipping $A(xy)$ also causes a significant decrease in potential, and for this it suffices to show that a significant fraction of w, r, j bits go from 0 to 1. Let A' denote A with $A(xy)$ flipped to 1. For each w such that $L_2^{A'}(w) \neq L_2^A(w)$, it must be the case that

$$M_r^{L_3^{A'}}(w) \;\neq\; M_r^{L_3^A}(w) \tag{7.2}$$

[9]We again only deal with L_2^A on inputs of length n^d, but we consider L_3^A on all input lengths. We could assume all queries M makes to its first oracle have the same length, but it turns out this would not make the proof any simpler.

for at least 1/3 of the r's. Thus we know that Inequality (7.2) holds for a significant fraction of pairs w, r. If Inequality (7.2) holds for w, r then there must exist a j such that the w, r, j bit changes when we go from A to A'. But the fraction of pairs w, r such that the w, r, j bit goes from 1 to 0 for some j is tiny (at most polynomially larger than the fraction of triples w, r, j that go from 1 to 0). Thus a significant fraction of pairs w, r are such that the w, r, j bit goes from 0 to 1 for some j, and hence a significant fraction (possibly a polynomially smaller fraction) of triples w, r, j go from 0 to 1. Thus we have a significant decrease in potential when we flip $A(xy)$.

7.3.2 Intuition for Theorem 7.2 and Theorem 7.3

It is well-known that error-correcting codes can be used to construct worst-case to average-case reductions, at least for large complexity classes such as PSPACE [22, 238]. To be applicable, the codes must have very efficient encoders (since this dictates the complexity of the language being reduced to) and very efficient decoders (since this dictates the complexity of the reduction itself). Our strategy for proving Theorem 7.2 and Theorem 7.3 is to set up the relativization oracle in such a way that error-correcting codes are in some sense the *only* way to construct worst-case to average-case reductions of the appropriate types, and then argue that the efficiency of the resulting encoders and decoders is too good to be true. That is, we would like to be able to extract a good error-correcting code from any purported reduction and then apply known lower bounds on the efficiency of encoders and decoders for such codes. For Theorem 7.2, we use a result due to Viola [257] which states that good error-correcting codes[10] cannot be encoded by small constant-depth circuits. For Theorem 7.3, we use a lower bound due to Kerenidis and de Wolf [158] on the length of 2-query locally decodable codes.

Our approach for Theorem 7.2 and Theorem 7.3 is in some sense a *dual* approach to the one we used for Theorem 7.1. As before, we have a reduction R that is trying to solve a problem with the aid of a relativization oracle A and a reduction oracle B. Before, our goal was to *render B useless to R* so we could focus on how R interacted with A. Now, our goal is to *render A useless to R* so we can focus on how R interacts with B. Before, we found a good *row* of A and filled in that row adversarially. Now, we find a good *column* of A and fill in that column adversarially.

Unlike in the proof of Theorem 7.1, we cannot use the Impagliazzo-Levin result to reduce PSamp to $\mathcal{U}$ since it uses too many queries. But again, directly handling the samplable ensembles presents no major difficulties. Thus, for the rest of this section we assume PSamp is replaced by $\mathcal{U}$.

The basic setup is the same as before. We have an algorithm M (PH-type for Theorem 7.2 or arbitrary complexity for Theorem 7.3). We have a polynomial-time randomized reduction R that uses a limited number of queries to the reduction oracle. For simplicity we assume that on inputs of length n, R only queries the reduction oracle on inputs of length n^d (for some positive integer d) and only with some fixed polynomially small δ. We construct a sequence

[10]His result even applies to list-decodable codes, but we do not need this stronger result.

of relativization oracles $A : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, and we define $L_1^A : \{0,1\}^n \to \{0,1\}$ by $L_1^A(x) = \bigvee_y A(xy)$, and we let $L_2^A : \{0,1\}^{n^d} \to \{0,1\}$ denote the language computed by M^A. For the final version of A, we want $R^{B,A}(x)$ to output $L_1^A(x)$ with probability $< 1/2 + 1/n^{\log n}$ for some $x \in \{0,1\}^n$ and some $B : \{0,1\}^{n^d} \to \{0,1,\perp\}$ that agrees with L_2^A on at least a $1-\delta$ fraction of inputs and returns $\perp$ on the rest. We have $1/2 + 1/n^{\log n}$ instead of $2/3$ for the reason discussed at the end of Section 7.2.3.

Let us start by pretending that R never queries A. Then it is completely straightforward to extract a good binary error-correcting code from M, R: Pick an arbitrary column y and define

$$C : \{0,1\}^{2^n} \to \{0,1\}^{2^{n^d}}$$

by viewing the input as a function $Z : \{0,1\}^n \to \{0,1\}$ and the output as a function $C(Z) : \{0,1\}^{n^d} \to \{0,1\}$ given by $C(Z) = L_2^{A_Z}$ where A_Z denotes the relativization oracle with Z as the y^{th} column and 0's everywhere else. If R really is of the hypothesized type no matter which Z we use, then it immediately follows that R is a decoder that recovers any bit $Z(x) = L_1^{A_Z}(x)$ of the information word from any corrupted code word B that has at most a δ fraction of erasures (and no flipped bits).

For Theorem 7.2, note that C has relative minimum distance $> \delta$ and each bit of C is encodable by a small constant-depth circuit since M is a PH-type algorithm with oracle access to Z [91]. This contradicts a result of Viola [257] which says that such a code cannot exist. Thus there must be some Z for which R is not of the hypothesized type.

For Theorem 7.3, note that C is a 2-query locally decodable code in the sense that each bit of the information word can be recovered with probability at least $1/2 + 1/n^{\log n}$ assuming there are at most a δ fraction of erasures.[11] Since the code word length is only *quasipolynomial* in the information word length, this contradicts a result of Kerenidis and de Wolf [158] which says that the length of such a code must be *nearly exponential*.[12] Thus there must be some Z for which R is not of the hypothesized type. Since the lower bound holds regardless of the complexity of encoding, we can handle *any* uniform complexity class of languages.

Now we return to the "real world" where R may query A. Then the above argument, with an arbitrary fixed y, does not work because R might know y in which case R can easily go look up the answers to L_1^A in the y^{th} column. We must choose y so as to "hide" the answers from R. Restricting the number of queries R can make to B is essential for this: If R can make n queries then M can easily let R know what y is by explicitly writing y over and over again in the truth table L_2^A, and R would have no trouble retrieving this information from any B that has sufficient agreement with L_2^A. (Of course in Theorem 7.2, R *can* use n, or any fixed polynomial, number of queries. But this is easily remedied by just adding $2^{\text{poly}(n)}$ columns to the table A, with a high enough degree polynomial, so that we *can* hide the answers from R. Henceforth we assume R only uses $n^{o(1)}$ queries, so that we can stick

[11]Usually, locally decodable codes are defined in terms of flipped bits rather than erasures, but they are equivalent up to small differences in parameters.

[12]The lower bound is only *nearly* exponential since the relative minimum distance and the advantage over 1/2 in correct decoding probability are subconstant in our case.

with 2^n columns.)

Suppose we could choose y so that for every x and every $B : \{0,1\}^{n^d} \to \{0,1,\perp\}$, the probability that $R^{B,0}(x)$ (where 0 denotes the all 0's relativization oracle) queries a bit in the y^{th} column is at most $1/2n^{\log n}$. Then we would know that for every Z, every x, and every B that is valid for L_2^{Az}, the probability $R^{B,0}(x)$ outputs $L_1^{Az}(x)$ is within $1/2n^{\log n}$ of the probability $R^{B,Az}(x)$ outputs $L_1^{Az}(x)$ and is hence at least $1/2 + 1/2n^{\log n}$. This would suffice for a contradiction, because we could use $R^{B,0}$ for the decoder. Actually this property of y is more than we really need. If we replace "every x" with "most x" then we could just remove the bad x's from consideration, at a small loss in the information word length, and we would still get a contradiction. Now to find such a y, we use the fact that quantifying over all B is the same as quantifying over all paths of adaptivity in R's access to B, and there are a limited number of such paths. Specifically, for every x and every r there are only a small number of columns of the relativization oracle that get queried by $R_r^{\circ,0}(x)$ over all possible reduction oracles (namely, at most the running time of R times 3 to the number of reduction oracle queries). By an averaging argument, there is some y such that for most x's, all but a $1/2n^{\log n}$ fraction of r's are such that $R_r^{B,0}(x)$ does not query any bit in the y^{th} column, for any B. This is good enough for our purpose.

The bottom line is that there are basically only two ways M could help R solve L_1^A: by telling R the answers, or by telling R where to find the answers in A. The former is impossible because then we would have an error-correcting code that is too good to be true, and the latter is impossible because R cannot make enough queries to B to retrieve the identity of y.

7.4 Generic Setup for the Formal Proofs

We first need the following complicated-looking lemma, which just says that in all three of our theorems, we can assume without loss of generality that on inputs of length n, any candidate reduction only queries the reduction oracle on inputs of length n^d and only with $\delta = 1/n^d$ for some positive integer d.

Lemma 7.9. *For every polynomial-time randomized reduction $R^{\circ,\circ}$ (where the reduction oracle is of the form $\{0,1\}^* \times \mathbb{R}_{>0} \to \{0,1,\perp\}$) there exists a polynomial-time randomized reduction $R_{clean}^{\circ,\circ}$ and a positive integer d such that the following holds. For every polynomial-time sampler S° there exists a polynomial-time sampler S_{clean}°, and for every uniform complexity class of languages $\mathcal{C}$ and every i there exists an i_{clean}, such that for every relativization oracle A, the following properties all hold.*

- *If $R^{\circ,A}$ is of type*
$$\big(M_i^A, D^A\big) \in \mathrm{AvgZPP}^A \;\Rightarrow\; L \in \mathrm{BPP}^A$$
for some language L, where D^A is the ensemble sampled by S^A, then $R_{clean}^{\circ,A}$ is of type
$$\big(M_{i_{clean}}^A, D_{clean}^A\big) \in \mathrm{AvgZPP}^A \;\Rightarrow\; L \in \mathrm{BPP}^A$$
where D_{clean}^A is the ensemble sampled by S_{clean}^A.

- *On inputs of length n, R_{clean} only queries the reduction oracle on inputs of length n^d and only with $\delta = 1/n^d$.*

- *R_{clean} always makes the same number of queries to the reduction oracle as R does.*

- *If $M_i^A \in \mathcal{C}^A$ then $M_{i_{clean}}^A \in \mathcal{C}^A$.*

Proof sketch. The basic idea is to take the answers to all the inputs to M_i^A up to the longest length R on inputs of length n could possibly query the reduction oracle, and put them in some larger input length n^d. Here d needs to be large enough that $1/n^d$ times the longest length R could query is less than the smallest value of δ that R could possibly query (which is at least inverse polynomial). The reason for multiplying by the longest length is that an error of $1/n^d$ in the AvgZPP oracle could get amplified by this amount when restricted to any particular input length that is stored "within" n^d. The index i_{clean} is just the j guaranteed by Definition 7.8 for index i and the mapping reduction we just informally described. □

We now describe the basic setup that is common to the proofs of all three theorems. However, this setup will need to be customized a bit for each of the three proofs.

We have a uniform complexity class of languages $\mathcal{C}$ with enumeration $\{M_1, M_2, \ldots\}$. Consider an arbitrary triple i, S, R where $i \in \mathbb{N}$, S is a polynomial-time sampler, and R is a polynomial-time randomized reduction. Using Lemma 7.9 we can assume without loss of generality that on inputs of length n, R only queries the reduction oracle on inputs of length n^d and only with $\delta = 1/n^d$ for some positive integer d. For an arbitrary relativization oracle $A \subseteq \{0,1\}^*$ we make the following definitions. Let L_1^A denote the NP^A language defined by

$$L_1^A \;=\; \big\{x \;:\; \exists y \text{ such that } |y| = |x| \text{ and } xy \in A\big\}.$$

If M_i^A defines a language in $\mathcal{C}^A$ then let L_2^A denote this language.[13] Let D^A denote the $\mathrm{PS{\scriptstyle AMP}}^A$ ensemble defined by S^A.

We wish to construct a relativization oracle A^* so that $L_1^{A^*} \in \mathrm{UP}^{A^*}$ (by ensuring that in the definition of $L_1^{A^*}$, y is always unique if it exists) and so that for all i, S, R, either $M_i^{A^*}$ fails to define a language in $\mathcal{C}^{A^*}$, or otherwise

$$\Pr_{r_R, B}\Big[R_{r_R}^{B,A^*}(x) = L_1^{A^*}(x)\Big] \;<\; 2/3$$

for some $x \in \{0,1\}^*$ and some randomized function $B : \{0,1\}^* \times \mathbb{R}_{>0} \to \{0,1,\bot\}$ which is a valid AvgZPP oracle for $\big(L_2^{A^*}, D^{A^*}\big)$, thereby ensuring that the reduction R°,A^*} fails to be of type

$$\big(L_2^{A^*}, D^{A^*}\big) \subseteq \mathrm{AvgZPP}^{A^*} \;\Rightarrow\; L_1^{A^*} \subseteq \mathrm{BPP}^{A^*}.$$

We construct a sequence of relativization oracles by starting with $\emptyset$ and adding strings and never taking them back out. We take A^* to be the limit of this sequence. Throughout

[13]Technically M_i^A *equals* the language L_2^A according to Definition 7.8, but the notation L_2^A is more convenient for the proofs.

the proofs, we simply refer to the "current" A with the understanding that this is the set of strings that have been included so far. We diagonalize against each triple i, S, R in sequence. After each round of diagonalization, we have the requirement that A^* matches the current A up through a certain input length, and we know that the current A contains no strings longer than that length. Now consider an arbitrary round, and suppose i, S, R is the triple to diagonalize against.

If there exists an A' consistent with the requirements of previous rounds and such that $M_i^{A'}$ fails to define a language in $\mathcal{C}^{A'}$, say with x as the violating input, then we update A to match A' up through the largest input length $M_i^{A'}(x)$ can query, and we require that A^* matches the new A up through this input length. This ensures that $M_i^{A^*}$ fails to define a language in $\mathcal{C}^{A^*}$, and we can move on to the next round.

Otherwise, we know that whatever we do to A, L_2^A will always be defined. Choose n large enough so that the following three things hold.

- The relativization oracle is fresh for all input lengths $\geq n$.
- The asymptotic constraints throughout the arguments are satisfied.
- The "relevant computations" all run in time $n^{\log n}$ without a big O.

The "relevant computations" include S on input n^d, R on inputs of length n, and (depending on the theorem) possibly the underlying computations of M_i on inputs of length n^d. We construct A at input length $2n$ to ensure that at the end of this round,

$$\Pr_{r_R, B}\left[R_{r_R}^{B,A}(x) = L_1^A(x)\right] \ < \ 2/3$$

for some $x \in \{0,1\}^n$ and some randomized function $B : \{0,1\}^{n^d} \to \{0,1,\bot\}$ which is a valid AvgZPP oracle for $\left(L_2^A, D^A\right)$ at input length n^d with respect to $\delta = 1/n^d$. Note that it makes sense to run $R^{B,A}(x)$ since this computation only queries B on inputs of length n^d and only with $\delta = 1/n^d$ (so we are justified in omitting the δ). This suffices to diagonalize against i, S, R because we can require that A^* matches the new A up through input length $n^{\log n}$ and up through the longest input length M_i can query on inputs of length n^d, thus ensuring the following three things.

- $L_1^{A^*}(x) = L_1^A(x)$.
- $R^{B,A^*}(x)$ behaves the same as $R^{B,A}(x)$.
- $L_2^{A^*}|_{n^d} = L_2^A|_{n^d}$ and $D_{n^d}^{A^*} = D_{n^d}^A$, which implies that B is a valid AvgZPP oracle for $\left(L_2^{A^*}, D^{A^*}\right)$ at input length n^d with respect to $\delta = 1/n^d$ and can thus be extended to a full valid AvgZPP oracle for $\left(L_2^{A^*}, D^{A^*}\right)$ without changing the behavior of $R^{B,A^*}(x)$.

7.5 Proof of Theorem 7.1

We use the setup from Section 7.4, customized as follows. We have $\mathcal{C} = \mathrm{BPP}_{\parallel}^{\mathrm{NP}}$, and M_i corresponds to a pair M, N where M is a $\mathrm{BPP}_{\parallel}^{\circ}$-type algorithm and N is an NP-type algorithm. Thus L_2^A is the $\left(\mathrm{BPP}_{\parallel}^{\mathrm{NP}}\right)^A$ language computed by $M^{L_3^A, A}$ where L_3^A denotes the NP^A language computed by N^A. Also, M on inputs of length n^d, as well as N on all inputs that could be queried by M on inputs of length n^d, count as "relevant computations" and thus all run in time $n^{\log n}$ without a big O.

Assume without loss of generality that for some positive integer e, M on inputs of length n^d always makes exactly n^e queries to its first oracle, and let $M_{r_M}^{L_3^A, A}(w)_j \in \{0,1\}^*$ denote the j^{th} of these queries when the input is w and the randomness is r_M.

7.5.1 Main Construction

Recall that M, N, S, R, n are fixed. For all relativization oracles A (not just the one we have constructed so far) we define the potential

$$\Phi^A \;=\; \mathop{\mathrm{E}}_{r_S, r_M, j}\left[1 - L_3^A\Big(M_{r_M}^{L_3^A, A}\big(S_{r_S}^A(n^d)\big)_j\Big)\right]$$

where $j \in \{1, \ldots, n^e\}$ is chosen uniformly at random. The construction has two stages.

Stage 1. This stage proceeds in iterations. For a given iteration, let A denote the current relativization oracle after the previous iteration. If there exist $x \in \{0,1\}^n$ and $y \in \{0,1\}^n$ such that $x \notin L_1^A$ and $\Phi^{A \cup \{xy\}} \le \Phi^A - 1/n^{3\log n}$ then update $A := A \cup \{xy\}$ and continue with the next iteration. Otherwise, halt stage 1 and proceed to stage 2.

The following lemma is the technical heart of the proof of Theorem 7.1. We first finish the proof of Theorem 7.1 assuming the lemma, and then we prove the lemma in Section 7.5.2.

Lemma 7.10. *At the end of stage 1, there exists an $x \in \{0,1\}^n$ such that $x \notin L_1^A$ and for all $y \in \{0,1\}^n$,*

$$\Pr_{r_S}\left[L_2^{A \cup \{xy\}}\big(S_{r_S}^A(n^d)\big) \neq L_2^A\big(S_{r_S}^A(n^d)\big)\right] \;\le\; 1/8n^{d+\log n} \tag{7.3}$$

and

$$\Pr_{r_S}\left[S_{r_S}^{A \cup \{xy\}}(n^d) \neq S_{r_S}^A(n^d)\right] \;\le\; 1/2n^d. \tag{7.4}$$

Stage 2. Let A denote the current relativization oracle at the end of stage 1, and let x be as guaranteed by Lemma 7.10. Let $Y \subseteq \{0,1\}^n$ be an arbitrary set of size $4n^{\log n}$. Define a deterministic reduction oracle $B : \{0,1\}^{n^d} \to \{0,1,\bot\}$ by

$$B(w) \;=\; \begin{cases} L_2^A(w) & \text{if } L_2^{A \cup \{xy\}}(w) = L_2^A(w) \text{ for all } y \in Y \\ \bot & \text{otherwise} \end{cases}.$$

There are two cases.

Case 1. If

$$\Pr_{r_R}\left[R^{B,A}_{r_R}(x) = 1\right] \;>\; 1/3$$

then we will use A for the relativization oracle at the beginning of the next round of diagonalization, without changing it. Since $x \notin L^A_1$, we have

$$\Pr_{r_R}\left[R^{B,A}_{r_R}(x) = L^A_1(x)\right] \;<\; 2/3.$$

We just need to verify that B is a valid AvgZPP oracle for $\left(L^A_2, D^A\right)$ at input length n^d with respect to $\delta = 1/n^d$. Obviously, $B(w)$ always returns $L^A_2(w)$ or $\bot$, by our definition of B. We have

$$\begin{aligned}
\Pr_{w \sim D^A_{n^d}}\left[B(w) = \bot\right] &= \Pr_{r_S}\left[B\big(S^A_{r_S}(n^d)\big) = \bot\right] \\
&= \Pr_{r_S}\left[\exists y \in Y \text{ such that } L^{A\cup\{xy\}}_2\big(S^A_{r_S}(n^d)\big) \neq L^A_2\big(S^A_{r_S}(n^d)\big)\right] \\
&\le \sum_{y\in Y} \Pr_{r_S}\left[L^{A\cup\{xy\}}_2\big(S^A_{r_S}(n^d)\big) \neq L^A_2\big(S^A_{r_S}(n^d)\big)\right] \\
&\le \sum_{y\in Y} 1/8n^{d+\log n} \\
&= |Y| \cdot 1/8n^{d+\log n} \\
&= 1/2n^d \\
&\le 1/n^d = \delta
\end{aligned}$$

where the fourth line follows by Lemma 7.10. Thus we have succeeded in diagonalizing against M, N, S, R as described at the end of Section 7.4.

Case 2. If

$$\Pr_{r_R}\left[R^{B,A}_{r_R}(x) = 1\right] \;\le\; 1/3$$

then for each $y \in Y$ we define

$$\pi_y \;=\; \Pr_{r_R}\left[R^{B,A}_{r_R}(x) \text{ queries } A(xy)\right].$$

Since $R^{B,A}(x)$ runs in time $n^{\log n}$, we have $\sum_{y\in Y} \pi_y \le n^{\log n}$. Thus there exists a $y \in Y$ such that $\pi_y \le n^{\log n}/|Y| = 1/4$. Fix this y. We will update the relativization oracle to be $A \cup \{xy\}$ for the end of this round of diagonalization. Since $x \in L^{A\cup\{xy\}}_1$, we have

$$\begin{aligned}
\Pr_{r_R}\left[R^{B,A\cup\{xy\}}_{r_R}(x) = L^{A\cup\{xy\}}_1(x)\right] &\le \Pr_{r_R}\left[R^{B,A}_{r_R}(x) = 1 \text{ or } R^{B,A\cup\{xy\}}_{r_R}(x) \neq R^{B,A}_{r_R}(x)\right] \\
&\le \Pr_{r_R}\left[R^{B,A}_{r_R}(x) = 1 \text{ or } R^{B,A}_{r_R}(x) \text{ queries } A(xy)\right]
\end{aligned}$$

$$\begin{aligned}
&\le \Pr_{r_R}\left[R^{B,A}_{r_R}(x)=1\right]+\pi_y\\
&\le 1/3+1/4\\
&< 2/3.
\end{aligned}$$

We just need to verify that B is a valid AvgZPP oracle for $\left(L_2^{A\cup\{xy\}}, D^{A\cup\{xy\}}\right)$ at input length n^d with respect to $\delta = 1/n^d$. Since $y \in Y$, we have that for all w, if $B(w) \neq \perp$ then $B(w) = L_2^A(w) = L_2^{A\cup\{xy\}}(w)$, by our definition of B. We also have

$$\begin{aligned}
\Pr_{w\sim D^{A\cup\{xy\}}_{n^d}}\left[B(w)=\perp\right] &= \Pr_{r_S}\left[B\big(S^{A\cup\{xy\}}_{r_S}(n^d)\big)=\perp\right]\\
&\le \Pr_{r_S}\left[B\big(S^{A}_{r_S}(n^d)\big)=\perp \text{ or } S^{A\cup\{xy\}}_{r_S}(n^d)\neq S^{A}_{r_S}(n^d)\right]\\
&\le \Pr_{r_S}\left[B\big(S^{A}_{r_S}(n^d)\big)=\perp\right]+\Pr_{r_S}\left[S^{A\cup\{xy\}}_{r_S}(n^d)\neq S^{A}_{r_S}(n^d)\right]\\
&\le 1/2n^d+1/2n^d\\
&= 1/n^d=\delta
\end{aligned}$$

where the fourth line follows by the calculation from case 1 and by Lemma 7.10. Thus we have succeeded in diagonalizing against M, N, S, R as described at the end of Section 7.4.

7.5.2 Proof of Lemma 7.10

For all A (not just the one we have constructed so far) and all r_S, r_M, j, let us define

$$\Phi^A_{r_S,r_M,j} \;=\; 1-L_3^A\Big(M^{L_3^A,A}_{r_M}\big(S^A_{r_S}(n^d)\big)_j\Big)$$

and

$$\Phi^A_{r_S,r_M} \;=\; \operatorname*{E}_j\left[\Phi^A_{r_S,r_M,j}\right]$$

so that $\Phi^A = \mathrm{E}_{r_S,r_M}\left[\Phi^A_{r_S,r_M}\right]$. We always have $0 \le \Phi^A \le 1$.

From here on out, A denotes the current relativization oracle at the end of stage 1. Since there are at most $n^{3\log n}$ iterations before stage 1 terminates, we have

$$\Pr_{x\in\{0,1\}^n}\left[x\in L_1^A\right] \;\le\; n^{3\log n}/2^n$$

where x is chosen uniformly at random. For $x \in \{0,1\}^n$ define

$$p_x \;=\; \Pr_{r_S,r_M}\Big[\exists y\in\{0,1\}^n \text{ such that } M^{L_3^A,A}_{r_M}\big(S^A_{r_S}(n^d)\big) \text{ queries } A(xy)\Big].$$

Since $M^{L_3^A,A}(w)$ runs in time $n^{\log n}$ for all $w \in \{0,1\}^{n^d}$, we have $\sum_x p_x \le n^{\log n}$ and thus

$$\Pr_{x\in\{0,1\}^n}\left[p_x > 1/n^{3\log n}\right] \;<\; n^{4\log n}/2^n.$$

For every $v \in L_3^A$ pick an arbitrary accepting computation path of $N^A(v)$ to be the "designated" path. For $x \in \{0,1\}^n$ define

$$q_x \;=\; \Pr_{r_S, r_M, j}\Big[M_{r_M}^{L_3^A, A}\big(S_{r_S}^A(n^d)\big)_j \in L_3^A \text{ and } \exists y \in \{0,1\}^n \text{ such that } N^A\Big(M_{r_M}^{L_3^A, A}\big(S_{r_S}^A(n^d)\big)_j\Big) \text{ queries } A(xy) \text{ on the designated path}\Big]$$

where j is chosen uniformly at random. Since $N^A(v)$ runs in time $n^{\log n}$ for every v of interest, we have $\sum_x q_x \leq n^{\log n}$ and thus

$$\Pr_{x \in \{0,1\}^n}\big[q_x > 1/n^{4\log n}\big] \;<\; n^{5\log n}/2^n.$$

For $x \in \{0,1\}^n$ define

$$s_x \;=\; \Pr_{r_S}\Big[\exists y \in \{0,1\}^n \text{ such that } S_{r_S}^A(n^d) \text{ queries } A(xy)\Big].$$

Since $S^A(n^d)$ runs in time $n^{\log n}$, we have $\sum_x s_x \leq n^{\log n}$ and thus

$$\Pr_{x \in \{0,1\}^n}\big[s_x > 1/n^{3\log n}\big] \;<\; n^{4\log n}/2^n.$$

By a union bound we find that

$$\begin{aligned}
&\Pr_{x \in \{0,1\}^n}\Big[x \notin L_1^A \text{ and } p_x \leq 1/n^{3\log n} \text{ and } q_x \leq 1/n^{4\log n} \text{ and } s_x \leq 1/n^{3\log n}\Big] \\
&> \; 1 - \big(n^{3\log n}/2^n\big) - \big(n^{4\log n}/2^n\big) - \big(n^{5\log n}/2^n\big) - \big(n^{4\log n}/2^n\big) \\
&> \; 0.
\end{aligned}$$

Thus there exists an $x \in \{0,1\}^n$ such that $x \notin L_1^A$ and $p_x \leq 1/n^{3\log n}$ and $q_x \leq 1/n^{4\log n}$ and $s_x \leq 1/n^{3\log n}$. Fix this x. We claim that this x satisfies the conditions of Lemma 7.10. Suppose for contradiction that there exists a $y \in \{0,1\}^n$ such that either Inequality (7.3) does not hold or Inequality (7.4) does not hold. Fix this y. We claim that $\Phi^{A\cup\{xy\}} \leq \Phi^A - 1/n^{3\log n}$, thus contradicting the fact that stage 1 halted. Henceforth we let A' denote $A \cup \{xy\}$. We partition the joint sample space of S's internal randomness and M's internal randomness into five events.

$$\begin{aligned}
E_1 &= \Big\{(r_S, r_M) \;:\; S_{r_S}^{A'}(n^d) \neq S_{r_S}^A(n^d)\Big\} \\
E_2 &= \Big\{(r_S, r_M) \;:\; (r_S, r_M) \notin E_1 \text{ and } M_{r_M}^{L_3^A, A}\big(S_{r_S}^A(n^d)\big) \text{ queries } A(xy)\Big\} \\
E_3 &= \Big\{(r_S, r_M) \;:\; (r_S, r_M) \notin E_1 \cup E_2 \text{ and } \exists j \text{ such that } M_{r_M}^{L_3^A, A}\big(S_{r_S}^A(n^d)\big)_j \in L_3^A \backslash L_3^{A'}\Big\} \\
E_4 &= \Big\{(r_S, r_M) \;:\; (r_S, r_M) \notin E_1 \cup E_2 \cup E_3 \text{ and } \exists j \text{ such that } M_{r_M}^{L_3^A, A}\big(S_{r_S}^A(n^d)\big)_j \in L_3^{A'} \backslash L_3^A\Big\} \\
E_5 &= \Big\{(r_S, r_M) \;:\; (r_S, r_M) \notin E_1 \cup E_2 \cup E_3 \cup E_4\Big\}
\end{aligned}$$

Claim 7.11. $\Pr_{r_S,r_M}\big[(r_S,r_M)\in E_1\big]\le 1/n^{3\log n}$ *and for all* $(r_S,r_M)\in E_1$, $\Phi^{A'}_{r_S,r_M}-\Phi^{A}_{r_S,r_M}\le 1$.

Claim 7.12. $\Pr_{r_S,r_M}\big[(r_S,r_M)\in E_2\big]\le 1/n^{3\log n}$ *and for all* $(r_S,r_M)\in E_2$, $\Phi^{A'}_{r_S,r_M}-\Phi^{A}_{r_S,r_M}\le 1$.

Claim 7.13. $\Pr_{r_S,r_M}\big[(r_S,r_M)\in E_3\big]\le 1/n^{3\log n}$ *and for all* $(r_S,r_M)\in E_3$, $\Phi^{A'}_{r_S,r_M}-\Phi^{A}_{r_S,r_M}\le 1$.

Claim 7.14. $\Pr_{r_S,r_M}\big[(r_S,r_M)\in E_4\big]\ge 1/n^{2\log n}$ *and for all* $(r_S,r_M)\in E_4$, $\Phi^{A'}_{r_S,r_M}-\Phi^{A}_{r_S,r_M}\le -1/n^e$.

Claim 7.15. $\Pr_{r_S,r_M}\big[(r_S,r_M)\in E_5\big]\le 1$ *and for all* $(r_S,r_M)\in E_5$, $\Phi^{A'}_{r_S,r_M}-\Phi^{A}_{r_S,r_M}\le 0$.

From these five claims it follows that

$$\begin{aligned}
\Phi^{A'}-\Phi^{A} &= \mathop{\mathrm{E}}_{r_S,r_M}\Big[\Phi^{A'}_{r_S,r_M}-\Phi^{A}_{r_S,r_M}\Big]\\
&= \sum_{k=1}^{5}\mathop{\mathrm{E}}_{r_S,r_M}\Big[\Phi^{A'}_{r_S,r_M}-\Phi^{A}_{r_S,r_M}\;\Big|\;(r_S,r_M)\in E_k\Big]\cdot\Pr_{r_S,r_M}\big[(r_S,r_M)\in E_k\big]\\
&\le 1/n^{3\log n}+1/n^{3\log n}+1/n^{3\log n}-1/n^{e+2\log n}\\
&\le -1/n^{3\log n}
\end{aligned}$$

which is what we wanted to show.

Proof of Claim 7.11. The first assertion follows because

$$\Pr_{r_S,r_M}\big[(r_S,r_M)\in E_1\big]\;\le\;\Pr_{r_S}\Big[S^A_{r_S}(n^d)\text{ queries }A(xy)\Big]\;\le\;s_x\;\le\;1/n^{3\log n}.$$

The second assertion follows trivially from the fact that $\Phi^{A'}_{r_S,r_M}\le 1$ and $\Phi^{A}_{r_S,r_M}\ge 0$. □

Proof of Claim 7.12. The first assertion follows because

$$\Pr_{r_S,r_M}\big[(r_S,r_M)\in E_2\big]\;\le\;\Pr_{r_S,r_M}\Big[M^{L_3^A,A}_{r_M}\big(S^A_{r_S}(n^d)\big)\text{ queries }A(xy)\Big]\;\le\;p_x\;\le\;1/n^{3\log n}.$$

The second assertion follows trivially from the fact that $\Phi^{A'}_{r_S,r_M}\le 1$ and $\Phi^{A}_{r_S,r_M}\ge 0$. □

Proof of Claim 7.13. The first assertion follows because

$$\begin{aligned}
&\Pr_{r_S,r_M}\big[(r_S,r_M)\in E_3\big]\\
\le\;&\Pr_{r_S,r_M}\Big[\exists j\text{ such that }M^{L_3^A,A}_{r_M}\big(S^A_{r_S}(n^d)\big)_j\in L_3^A\backslash L_3^{A'}\Big]
\end{aligned}$$

$$
\begin{aligned}
&\le \Pr_{r_S,r_M}\Big[\exists j \text{ such that } M_{r_M}^{L_3^A,A}\big(S_{r_S}^A(n^d)\big)_j \in L_3^A \text{ and} \\
&\qquad N^A\Big(M_{r_M}^{L_3^A,A}\big(S_{r_S}^A(n^d)\big)_j\Big) \text{ queries } A(xy) \text{ on the designated path}\Big] \\
&\le n^e \cdot q_x \\
&\le 1/n^{3\log n}
\end{aligned}
$$

where the second-to-last line follows using a union bound and the last line follows because $q_x \le 1/n^{4\log n}$. The second assertion follows trivially from the fact that $\Phi_{r_S,r_M}^{A'} \le 1$ and $\Phi_{r_S,r_M}^{A} \ge 0$. □

Proof of Claim 7.14. This is, in some sense, the crux of the whole proof. Since $1/n^{3\log n} \le 1/2n^d$, Claim 7.11 implies that Inequality (7.4) holds and therefore Inequality (7.3) does not hold. We claim that if $(r_S, r_M) \notin E_1 \cup E_2 \cup E_3 \cup E_4$ then

$$M_{r_M}^{L_3^{A'},A'}\big(S_{r_S}^A(n^d)\big) \;=\; M_{r_M}^{L_3^A,A}\big(S_{r_S}^A(n^d)\big).$$

This is because every query $M_{r_M}^{L_3^A,A}\big(S_{r_S}^A(n^d)\big)$ makes to its second oracle has the same answer under A' and A, and every query it makes to its first oracle has the same answer under $L_3^{A'}$ and L_3^A. Thus the computations $M_{r_M}^{L_3^{A'},A'}\big(S_{r_S}^A(n^d)\big)$ and $M_{r_M}^{L_3^A,A}\big(S_{r_S}^A(n^d)\big)$ proceed identically, making the same queries and receiving the same answers, and hence they produce the same output. The first assertion now follows because

$$
\begin{aligned}
&\Pr_{r_S,r_M}\big[(r_S,r_M)\in E_4\big] \\
\ge\;& \Pr_{r_S,r_M}\Big[(r_S,r_M)\notin E_1\cup E_2\cup E_3 \text{ and } M_{r_M}^{L_3^{A'},A'}\big(S_{r_S}^A(n^d)\big) \ne M_{r_M}^{L_3^A,A}\big(S_{r_S}^A(n^d)\big)\Big] \\
\ge\;& \Pr_{r_S,r_M}\Big[M_{r_M}^{L_3^{A'},A'}\big(S_{r_S}^A(n^d)\big) \ne M_{r_M}^{L_3^A,A}\big(S_{r_S}^A(n^d)\big)\Big] - \sum_{k=1}^{3}\Pr_{r_S,r_M}\big[(r_S,r_M)\in E_k\big] \\
\ge\;& \Pr_{r_S}\Big[L_2^{A'}\big(S_{r_S}^A(n^d)\big) \ne L_2^A\big(S_{r_S}^A(n^d)\big)\Big]/3 - \sum_{k=1}^{3}\Pr_{r_S,r_M}\big[(r_S,r_M)\in E_k\big] \\
>\;& 1/24n^{d+\log n} - 1/n^{3\log n} - 1/n^{3\log n} - 1/n^{3\log n} \\
\ge\;& 1/n^{2\log n}
\end{aligned}
$$

where the third line follows by a union bound and the second-to-last line follows by the negation of Inequality (7.3) and by Claim 7.11, Claim 7.12, and Claim 7.13.

We now argue the second assertion. Since $(r_S, r_M) \notin E_1$, we have $S_{r_S}^{A'}(n^d) = S_{r_S}^A(n^d)$. Let w denote this string. Since $(r_S, r_M) \notin E_1 \cup E_2$, we have

$$M_{r_M}^{L_3^{A'},A'}(w)_j \;=\; M_{r_M}^{L_3^A,A}(w)_j$$

for all j. Let v_j denote these strings, and note that $\Phi^{A'}_{r_S,r_M,j} = 1 - L_3^{A'}(v_j)$ and $\Phi^{A}_{r_S,r_M,j} = 1 - L_3^{A}(v_j)$. Since $(r_S, r_M) \notin E_3$, we have $\Phi^{A'}_{r_S,r_M,j} \le \Phi^{A}_{r_S,r_M,j}$ for all j. By the definition of E_4, we have $\Phi^{A'}_{r_S,r_M,j} = 0$ and $\Phi^{A}_{r_S,r_M,j} = 1$ for some j. Therefore,

$$\Phi^{A'}_{r_S,r_M} - \Phi^{A}_{r_S,r_M} \;=\; \frac{1}{n^e}\sum_j \left(\Phi^{A'}_{r_S,r_M,j} - \Phi^{A}_{r_S,r_M,j}\right) \;\le\; -1/n^e.$$

□

Proof of Claim 7.15. The first assertion is trivial. We now argue the second assertion. In the proof of Claim 7.14 we argued that if $(r_S, r_M) \notin E_1 \cup E_2 \cup E_3 \cup E_4$ then the computations $M_{r_M}^{L_3^{A'},A'}\big(S_{r_S}^{A'}(n^d)\big)$ and $M_{r_M}^{L_3^{A},A}\big(S_{r_S}^{A}(n^d)\big)$ proceed identically, making the same queries and receiving the same answers. In particular, $\Phi^{A'}_{r_S,r_M,j} = \Phi^{A}_{r_S,r_M,j}$ for all j, which implies that $\Phi^{A'}_{r_S,r_M} = \Phi^{A}_{r_S,r_M}$. □

7.6 Proof of Theorem 7.2

Fix a polynomial q. We use the setup from Section 7.4, customized as follows. We have $\mathcal{C} = \mathrm{PH}$, and M_i corresponds to a PH-type algorithm M. We redefine

$$L_1^A \;=\; \big\{x \;:\; \exists y \text{ such that } |y| = |x| + 2q(|x|) \text{ and } xy \in A\big\}$$

using $|y| = |x| + 2q(|x|)$ instead of $|y| = |x|$, and thus we need to construct A at input length $2n + 2q(n)$ rather than $2n$. We only diagonalize against reductions R that use at most q queries to the reduction oracle. Also, M on inputs of length n^d counts as "relevant computations" and thus runs in time $n^{\log n}$ without a big O. For the reason discussed at the end of Section 7.2.3, we have the stronger requirement that at the end of this round,

$$\Pr_{r_R,B}\Big[R_{r_R}^{B,A}(x) = L_1^A(x)\Big] \;<\; 1/2 + 1/n^{\log n}$$

with $1/2 + 1/n^{\log n}$ instead of 2/3. Finally, note that it can never be the case that M^A fails to define a language in PH^A, since PH is a syntactically defined class.

We generalize the notion of a reduction oracle: If $B : \{0,1\}^{n^d} \to \{0,1,\bot\}^{\mathbb{N}}$ is a deterministic function then running $R_{r_R}^{B,A}(x)$ means that for each w, the i^{th} time the computation queries $B(w)$ it gets $B(w)(i)$ as a response. Thus a randomized function $B : \{0,1\}^{n^d} \to \{0,1,\bot\}$ is a distribution over such deterministic functions, where each $B(w)(i)$ is independent and the distribution of $B(w)(i)$ depends only on w and not on i.

7.6.1 Main Construction

Recall that M, S, R, n are fixed. Let A denote the current relativization oracle at the beginning of this round. For $x \in \{0,1\}^n$ and $y \in \{0,1\}^{n+2q(n)}$ define

$$p_{x,y} \;=\; \Pr_{r_R}\Big[\exists B : \{0,1\}^{n^d} \to \{0,1,\bot\}^{\mathbb{N}} \text{ and } \exists x' \in \{0,1\}^n \text{ such that } R_{r_R}^{B,A}(x) \text{ queries } A(x'y)\Big]$$

and

$$p_y \;=\; \mathop{\mathrm{E}}_{x\in\{0,1\}^n}\big[p_{x,y}\big]$$

where x is chosen uniformly at random. For each $x \in \{0,1\}^n$ and r_R, the computation $R^{B,A}_{r_R}(x)$ has at most $3^{q(n)}$ computation paths over the possible responses it could get from B (recall that A is fixed). On each of these computation paths, $R^{B,A}_{r_R}(x)$ can query at most $n^{\log n}$ bits of A since it runs in time $n^{\log n}$. Thus there are at most $n^{\log n}3^{q(n)}$ pairs $(x', y) \in \{0,1\}^n \times \{0,1\}^{n+2q(n)}$ for which there exists a $B : \{0,1\}^{n^d} \to \{0,1,\bot\}^{\mathbb{N}}$ such that $R^{B,A}_{r_R}(x)$ queries $A(x'y)$. It follows that $\sum_y p_y \le n^{\log n}3^{q(n)}$ and thus

$$\Pr_{y\in\{0,1\}^{n+2q(n)}}\big[p_y > 1/2n^{\log n}\big] \;<\; 2n^{2\log n}3^{q(n)}/2^{n+2q(n)}$$

where y is chosen uniformly at random. For $y \in \{0,1\}^{n+2q(n)}$ define

$$s_y \;=\; \Pr_{r_S}\Big[\exists x' \in \{0,1\}^n \text{ such that } S^A_{r_S}(n^d) \text{ queries } A(x'y)\Big].$$

Since $S^A(n^d)$ runs in time $n^{\log n}$, we have $\sum_y s_y \le n^{\log n}$ and thus

$$\Pr_{y\in\{0,1\}^{n+2q(n)}}\big[s_y > 1/2n^d\big] \;<\; 2n^{d+\log n}/2^{n+2q(n)}.$$

By a union bound we find that

$$\begin{aligned}&\Pr_{y\in\{0,1\}^{n+2q(n)}}\Big[p_y \le 1/2n^{\log n} \text{ and } s_y \le 1/2n^d\Big]\\ &>\; 1 - \big(2n^{2\log n}3^{q(n)}/2^{n+2q(n)}\big) - \big(2n^{d+\log n}/2^{n+2q(n)}\big)\\ &>\; 0.\end{aligned}$$

Thus there exists a $y \in \{0,1\}^{n+2q(n)}$ such that $p_y \le 1/2n^{\log n}$ and $s_y \le 1/2n^d$. Fix this y. Now

$$\Pr_{x\in\{0,1\}^n}\big[p_{x,y} \ge 1/n^{\log n}\big] \;\le\; 1/2$$

and thus there exists a set $X \subseteq \{0,1\}^n$ of size $|X| = 2^{n-1}$ such that for all $x \in X$, $p_{x,y} < 1/n^{\log n}$. To prove the theorem, it suffices to show that there exists a $Z \subseteq \big\{xy \;:\; x \in X\big\}$, an $x \in X$, and a randomized function $B : \{0,1\}^{n^d} \to \{0,1,\bot\}$ which is a valid AvgZPP oracle for $\big(L_2^{A\cup Z}, D^{A\cup Z}\big)$ at input length n^d with respect to $\delta = 1/n^d$, such that

$$\Pr_{r_R,B}\Big[R^{B,A\cup Z}_{r_R}(x) = L_1^{A\cup Z}(x)\Big] \;<\; 1/2 + 1/n^{\log n}$$

because we can then update the relativization oracle to be $A \cup Z$ for the end of this round.

Suppose for contradiction that this does not hold. We can assume that r_S is sampled uniformly at random from $\{0,1\}^{n^{\log n}}$ when S is run on input n^d. Define an error-correcting code

$$C : \{0,1\}^{2^{n-1}} \to \{0,1\}^{2^{n^{\log n}}}$$

as follows, where the information word is viewed as a subset $Z \subseteq \{xy : x \in X\}$ and the code word is viewed as a function $C(Z) : \{0,1\}^{n^{\log n}} \to \{0,1\}$.

$$C(Z)(r_S) \;=\; L_2^{A\cup Z}\big(S^A_{r_S}(n^d)\big)$$

Claim 7.16. *The relative minimum distance of C is $> 1/2n^d$.*

We prove Claim 7.16 shortly. Let k denote the number of quantifiers M uses, and recall that M runs in time $n^{\log n}$ on inputs of length n^d. Since each bit of $C(Z)$ corresponds to running $M^{A\cup Z}$ on a fixed input of length n^d, each bit of $C(Z)$ is computable by a circuit of depth k and size $2^{n^{\log n}}$ where each input to the circuit is the output of a deterministic computation running in time $n^{\log n}$ with oracle access to $A \cup Z$. Since A is fixed, each of the inputs to this circuit is computable by a DNF with top fan-in $2^{n^{\log n}}$ and bottom fan-in $n^{\log n}$ whose inputs correspond to strings in $\{xy : x \in X\}$, that is, coordinates of the information word.

The bottom line is that there exists a binary error-correcting code with information word length 2^{n-1} and relative minimum distance $> 1/2n^d$ such that each bit of the code word is computable by a circuit of depth $k+2$ and size $2^{2n^{\log n}} n^{\log n}$. This contradicts the following result of Viola.

Theorem 7.17 ([257]). *If there exists a binary error-correcting code with information word length ν and relative minimum distance γ such that each bit of the code word is computable by a circuit of depth κ and size σ, then $\nu\gamma \leq O(\log^{\kappa-1}\sigma)$.*

Theorem 7.17 holds regardless of the rate of the code.

Proof of Claim 7.16. In Figure 7.1 we exhibit a decoder that can handle up to a $1/2n^d$ fraction of erasures. For an arbitrary $Z \subseteq \{xy : x \in X\}$, assume that C' agrees with $C(Z)$ on at least a $1 - 1/2n^d$ fraction of r_S's and outputs $\perp$ on the rest. Then we just need to show that $Z' = Z$. We do this by showing that for an arbitrary $x \in X$,

$$\Pr_{r_R,B}\Big[R^{B,A}_{r_R}(x) = L_1^{A\cup Z}(x)\Big] \;>\; 1/2$$

which implies that $xy \in Z'$ if and only if $x \in L_1^{A\cup Z}$ if and only if $xy \in Z$.

We start by showing that B is a valid AvgZPP oracle for $\big(L_2^{A\cup Z}, D^{A\cup Z}\big)$ at input length n^d with respect to $\delta = 1/n^d$. We have that $B(w)$ always equals $L_2^{A\cup Z}(w)$ or $\perp$, since if r_S is such that $S^A_{r_S}(n^d) = w$ and $C'(r_S) \neq \perp$ then

$$C'(r_S) \;=\; C(Z)(r_S) \;=\; L_2^{A\cup Z}\big(S^A_{r_S}(n^d)\big) \;=\; L_2^{A\cup Z}(w).$$

We have

$$\Pr_{r_S,B}\Big[B\big(S^A_{r_S}(n^d)\big) = \perp\Big]$$

- **Input:** $C' : \{0,1\}^{n^{\log n}} \to \{0,1,\bot\}$
- **Output:** $Z' \subseteq \big\{xy \;:\; x \in X\big\}$ given by

$$Z' \;=\; \Big\{xy \;:\; \Pr_{r_R,B}\Big[R^{B,A}_{r_R}(x) = 1\Big] > 1/2\Big\}$$

where the randomized function $B : \{0,1\}^{n^d} \to \{0,1,\bot\}$ is defined by

$$\Pr_B\big[B(w) = b\big] \;=\; \Pr_{r_S}\Big[C'(r_S) = b \;\Big|\; S^A_{r_S}(n^d) = w\Big]$$

if

$$\Pr_{r_S}\Big[S^A_{r_S}(n^d) = w\Big] \;>\; 0$$

and otherwise

$$\Pr_B\big[B(w) = \bot\big] \;=\; 1$$

Figure 7.1: Decoder for Claim 7.16

$$\begin{aligned}
&= \sum_{w \in \{0,1\}^{n^d}} \Pr_{r_S,B}\Big[B\big(S^A_{r_S}(n^d)\big) = \bot \;\Big|\; S^A_{r_S}(n^d) = w\Big] \cdot \Pr_{r_S,B}\Big[S^A_{r_S}(n^d) = w\Big] \\
&= \sum_{w \in \{0,1\}^{n^d}} \Pr_B\big[B(w) = \bot\big] \cdot \Pr_{r_S}\Big[S^A_{r_S}(n^d) = w\Big] \\
&= \sum_{w \in \{0,1\}^{n^d}} \Pr_{r_S}\Big[C'(r_S) = \bot \;\Big|\; S^A_{r_S}(n^d) = w\Big] \cdot \Pr_{r_S}\Big[S^A_{r_S}(n^d) = w\Big] \\
&= \Pr_{r_S}\big[C'(r_S) = \bot\big] \\
&\le 1/2n^d
\end{aligned}$$

and

$$\Pr_{r_S}\Big[S^{A\cup Z}_{r_S}(n^d) \ne S^A_{r_S}(n^d)\Big] \;\le\; \Pr_{r_S}\Big[\exists z \in Z \text{ such that } S^A_{r_S}(n^d) \text{ queries } A(z)\Big] \;\le\; s_y \;\le\; 1/2n^d$$

and thus

$$\begin{aligned}
\Pr_{w\sim D^{A\cup Z},B}\big[B(w) = \bot\big] \;&=\; \Pr_{r_S,B}\Big[B\big(S^{A\cup Z}_{r_S}(n^d)\big) = \bot\Big] \\
&\le\; \Pr_{r_S,B}\Big[B\big(S^A_{r_S}(n^d)\big) = \bot \text{ or } S^{A\cup Z}_{r_S}(n^d) \ne S^A_{r_S}(n^d)\Big] \\
&\le\; \Pr_{r_S,B}\Big[B\big(S^A_{r_S}(n^d)\big) = \bot\Big] + \Pr_{r_S}\Big[S^{A\cup Z}_{r_S}(n^d) \ne S^A_{r_S}(n^d)\Big]
\end{aligned}$$

$$\begin{aligned} &\le\ 1/2n^d + 1/2n^d \\ &=\ 1/n^d = \delta. \end{aligned}$$

Now we have

$$\begin{aligned} \Pr_{r_R,B}\left[R^{B,A\cup Z}_{r_R}(x) \neq R^{B,A}_{r_R}(x)\right] &\le\ \mathop{\mathrm{E}}_{B}\left[\Pr_{r_R}\left[\exists z \in Z \text{ such that } R^{B,A}_{r_R}(x) \text{ queries } A(z)\right]\right] \\ &\le\ \mathop{\mathrm{E}}_{B}\left[p_{x,y}\right] \\ &=\ p_{x,y} \\ &<\ 1/n^{\log n} \end{aligned}$$

and thus

$$\begin{aligned} \Pr_{r_R,B}\left[R^{B,A}_{r_R}(x) = L^{A\cup Z}_1(x)\right] &\ge\ \Pr_{r_R,B}\left[R^{B,A\cup Z}_{r_R}(x) = L^{A\cup Z}_1(x) \text{ and } R^{B,A\cup Z}_{r_R}(x) = R^{B,A}_{r_R}(x)\right] \\ &\ge\ \Pr_{r_R,B}\left[R^{B,A\cup Z}_{r_R}(x) = L^{A\cup Z}_1(x)\right] - \Pr_{r_R,B}\left[R^{B,A\cup Z}_{r_R}(x) \neq R^{B,A}_{r_R}(x)\right] \\ &>\ \left(1/2 + 1/n^{\log n}\right) - 1/n^{\log n} \\ &=\ 1/2 \end{aligned}$$

where the third line follows by our contradiction assumption. □

7.7 Proof of Theorem 7.3

We use the setup from Section 7.4, customized as follows. We only diagonalize against reductions R that use at most 2 queries to the reduction oracle. For the reason discussed at the end of Section 7.2.3, we have the stronger requirement that at the end of this round,

$$\Pr_{r_R,B}\left[R^{B,A}_{r_R}(x) = L^{A}_1(x)\right]\ <\ 1/2 + 1/n^{\log n}$$

with $1/2 + 1/n^{\log n}$ instead of $2/3$. The proof is so similar to the proof of Theorem 7.2 that we just sketch how it plays out. We can work with $|y| = n$ (rather than $|y| = n + 2q(n)$ as in the proof of Theorem 7.2).

7.7.1 Main Construction

Recall that M_i, S, R, n are fixed. Let A denote the current relativization oracle at the beginning of this round. There exists a $y \in \{0,1\}^n$ such that $p_y \le 1/4n^{\log n}$ and $s_y \le 1/2n^d$, and there exists a set $X \subseteq \{0,1\}^n$ of size $|X| = 2^{n-1}$ such that for all $x \in X$, $p_{x,y} \le 1/2n^{\log n}$. Then there exists a $Z \subseteq \left\{xy\ :\ x \in X\right\}$, an $x \in X$, and a randomized function $B : \{0,1\}^{n^d} \to \{0,1,\bot\}$ which is a valid AvgZPP oracle for $\left(L^{A\cup Z}_2, D^{A\cup Z}\right)$ at input length n^d with respect to $\delta = 1/n^d$, such that

$$\Pr_{r_R,B}\left[R^{B,A\cup Z}_{r_R}(x) = L^{A\cup Z}_1(x)\right]\ <\ 1/2 + 1/n^{\log n}$$

since otherwise we can extract an error-correcting code

$$C : \{0,1\}^{2^{n-1}} \rightarrow \{0,1\}^{2^{n^{\log n}}}$$

with the following properties. There is a randomized decoder that can handle up to a $1/2n^d$ fraction of erasures, and it recovers any bit of the information word with probability at least

$$\left(1/2 + 1/n^{\log n}\right) - 1/2n^{\log n} \;=\; 1/2 + 1/2n^{\log n}.$$

To recover any bit, the decoder runs $R^{B,A}(x)$ for some $x \in \{0,1\}^n$ and some randomized function B. Since R makes at most 2 queries to B, and since each query to B can be answered with at most 1 query to the corrupted code word C', the decoder makes at most 2 queries to C'.

The bottom line is that there exists a binary error-correcting code with information word length 2^{n-1} and code word length $2^{n^{\log n}}$ and a decoder that uses 2 queries to recover any bit of the information word with probability at least $1/2 + 1/2n^{\log n}$ when at most a $1/2n^d$ fraction of the code word bits are erased. This contradicts the following result of Kerenidis and de Wolf.

Theorem 7.18 ([158]). *If there exists a binary error-correcting code with information word length ν and code word length μ and a decoder that uses 2 queries to recover any bit of the information word with probability at least $1/2 + \epsilon$ when at most a γ fraction of the code word bits are erased, then $\mu \geq 2^{\Omega(\gamma\epsilon^3\nu)}$.*

Remarkably, the proof of Theorem 7.18 is based on quantum information theory. Kerenidis and de Wolf proved the stronger bound $\mu \geq 2^{\Omega(\gamma\epsilon^2\nu)}$ assuming that the decoder is guaranteed to work even if a γ fraction of the code word bits are flipped rather than just erased. The extra ϵ in the exponent in Theorem 7.18 grossly accounts for the generalization from flips to erasures. It may be possible to prove the stronger bound for erasure decoders, but Theorem 7.18 as stated is already good enough for our purpose.

The complexity of M_i is immaterial because Theorem 7.18 holds without any constraints on the efficiency of the encoder.

Chapter 8

Query Complexity in Errorless Hardness Amplification

8.1 Introduction

Traditionally, an algorithm for solving a computational problem is required to be correct on all inputs and is judged in terms of its efficiency (the amount of computational resources it uses). One criticism of this model is that it is too strict: In practice, an algorithm only needs to be correct on "real-world" inputs and not on contrived worst-case inputs. To address this issue within the framework of complexity theory, researchers developed the theory of average-case complexity (starting with the work of Levin [165]). In this theory, an algorithm is judged in terms of both its efficiency and the fraction of inputs on which it fails to solve the problem correctly. The topic of this chapter is the relationship between these two measures of the quality of an algorithm.

There are two standard settings for average-case complexity. In the original setting proposed by Levin [165], one only considers *errorless algorithms*, which are required to output the correct answer or "don't know" on each input.[1] An errorless algorithm is judged in terms of both its efficiency and the fraction of inputs on which it outputs "don't know". We refer to this setting as *errorless average-case complexity.* In the other setting, one considers arbitrary algorithms which may output the wrong answer rather than just "don't know" on an input. We refer to this setting as *non-errorless average-case complexity.* Errorless average-case complexity is an intermediate setting between worst-case complexity and non-errorless average-case complexity.

We first discuss non-errorless average-case complexity. A boolean function is said to be *mildly average-case hard* if no efficient algorithm can compute it on almost all inputs. Applications such as derandomization and cryptography require functions that are *strongly average-case hard*, meaning that no efficient algorithm can compute the function on notice-

[1]Actually, Levin proposed considering algorithms that are correct on all inputs but which are efficient "on average" with respect to a random input. Under a suitable formalization, such algorithms are equivalent to errorless algorithms that may fail on a small fraction of inputs but are efficient on all inputs.

ably more than half the inputs. This motivates hardness amplification, which is the problem of transforming a mildly average-case hard function into a strongly average-case hard function. A classic result in this area is the XOR Lemma [166, 133, 102, 144], which states that the XOR of sufficiently many independent copies of a mildly average-case hard function is strongly average-case hard, provided the model of efficient algorithms is small circuits.

However, the XOR Lemma (as well as the numerous subsequent results on hardness amplification) incurs an unfortunate loss in circuit size. Suppose the original function f is mildly average-case hard in the sense that no size s circuit succeeds on at least a $1-\delta$ fraction of inputs, and we wish for the new function f' to be strongly average-case hard in the sense that no size s' circuit succeeds on at least a $1/2+\epsilon$ fraction of inputs. Then we would like s' to be as large as possible, but the XOR Lemma requires that s' is actually smaller than s. This is because such results are proven by reductions which show that if f' is not strongly average-case hard, then a circuit witnessing this could be used to construct a circuit witnessing that f is not mildly average-case hard. If the reduction makes q queries to the hypothesized circuit, then plugging in a size s' circuit yields a circuit of size $\geq qs'$, and thus we must have $s' \leq s/q$. Hence the query complexity q governs the loss in circuit size. For the XOR Lemma, the query complexity is well-understood. The proof due to Impagliazzo [133] and Klivans and Servedio [161] shows that $q = O\big(\frac{1}{\epsilon^2}\log\frac{1}{\delta}\big)$ queries are sufficient, and Shaltiel and Viola [231] showed that in a certain sense, $q = \Omega\big(\frac{1}{\epsilon^2}\log\frac{1}{\delta}\big)$ queries are necessary.

Bogdanov and Safra [44] initiated the study of hardness amplification in Levin's original setting of errorless average-case complexity. A boolean function is said to be *mildly errorless average-case hard* if no efficient errorless algorithm (say, size s circuit) can compute it on almost all inputs (say, a $1-\delta$ fraction). A function is said to be *strongly errorless average-case hard* if no efficient errorless algorithm (say, size s' circuit) can compute it on a noticeable fraction of inputs (say, an ϵ fraction). Note that in the non-errorless setting, computing a boolean function on half the inputs is trivial (using constant 0 or constant 1), but in the errorless setting, computing a boolean function on even a small fraction of inputs is nontrivial. The goal of errorless hardness amplification is to transform a mildly errorless average-case hard function f into a strongly errorless average-case hard function f'. Such results suffer from a loss in circuit size for the same reason as in the non-errorless setting. Bogdanov and Safra [44] showed that $q = O\big((\frac{1}{\delta}\log\frac{1}{\epsilon})^2 \cdot \frac{1}{\epsilon}\log\frac{1}{\delta}\big)$ queries are sufficient when f' is the XOR of several independent copies of f. The result of Shaltiel and Viola [231] can be modified without difficulty to show that in a certain sense, $q = \Omega\big(\frac{1}{\epsilon}\log\frac{1}{\delta}\big)$ queries are necessary. We close the gap by showing that $q = O\big(\frac{1}{\epsilon}\log\frac{1}{\delta}\big)$ queries are sufficient.

Another natural goal for hardness amplification is to guarantee that if f represents an NP language at some input length, then f' also represents an NP language at some input length. In the non-errorless setting this goal has been studied in numerous works [203, 242, 132, 244, 55, 176, 108], and in the errorless setting this goal has been studied by Bogdanov and Safra [44]. We significantly improve the query complexity of the Bogdanov-Safra result.

8.1.1 The Errorless XOR Lemma

Given $f : \{0,1\}^n \to \{0,1\}$ we define $f^{\oplus k} : \{0,1\}^{n\times k} \to \{0,1\}$ as follows: $f^{\oplus k}(x_1, \ldots, x_k) = f(x_1) \oplus \cdots \oplus f(x_k)$.[2]

Definition 8.1 (Errorless Average-Case Hardness). *We say a circuit $A : \{0,1\}^n \to \{0,1,\perp\}$ is a δ-*errorless *circuit for $f : \{0,1\}^n \to \{0,1\}$ if*

(i) $A(x) \in \{f(x), \perp\}$ for all $x \in \{0,1\}^n$, and

(ii) $\Pr_x[A(x) = \perp] \le \delta$ where $x \in \{0,1\}^n$ is chosen uniformly at random.

*We say f is (s,δ)-*hard *if it has no δ-errorless circuit of size $\le s$.*

Theorem 8.2 (Query-Optimal Errorless XOR Lemma). *If f is (s,δ)-hard then $f' = f^{\oplus k}$ is $(s', 1-\epsilon)$-hard where $s' = s/\big(\frac{4}{\epsilon}\ln\frac{2}{\delta}\big)$, provided $k \ge \frac{16}{\delta}\ln\frac{2}{\epsilon}$.*

We prove Theorem 8.2 in Section 8.2.[3] Bogdanov and Safra [44] proved a version of Theorem 8.2 where $s' = s/\big(k^2 \cdot \frac{2}{\epsilon}\ln\frac{2}{\delta}\big)$, provided $k \ge \frac{2}{\delta}\ln\frac{2}{\epsilon}$.[4] Even for the best value of k, they only achieve $O\big((\frac{1}{\delta}\log\frac{1}{\epsilon})^2 \cdot \frac{1}{\epsilon}\log\frac{1}{\delta}\big)$ query complexity. Also, our bound on the query complexity does not depend on k.

We prove Theorem 8.2 by a reduction similar to the one used in [44]. Our contribution is a new, tight analysis of the reduction. The crux of the reduction is a randomized procedure that solves f errorlessly (meaning that for each input x it $\underline{\text{may}}$ output $f(x)$ with some probability and $\perp$ with some probability, but it never outputs $\overline{f(x)}$) while making one query to a hypothesized $(1-\epsilon)$-errorless circuit A' for f'. Suppose for some $\beta > 0$ we knew that $\le \delta/2$ fraction of inputs x are bad in the sense that the probability the procedure outputs $f(x)$ is $< \beta$. Then by amplifying the success probability on the good inputs and hard-wiring the randomness appropriately, we obtain a δ-errorless circuit A for f, via a reduction with query complexity $O\big(\frac{1}{\beta}\log\frac{1}{\delta}\big)$. The heart of our improvement over the Bogdanov-Safra proof is in arguing that we can take $\beta = \epsilon/4$. To prove this, we suppose the fraction of bad inputs is $> \delta/2$ and prove that then A' must compute f' on $< \epsilon$ fraction of inputs. The procedure outputs $f(x)$ if and only if the query is an input on which A' computes f'; furthermore the distribution of this query $(x_1, \ldots, x_k)$ is obtained by setting $x_i = x$ for a uniformly random i and picking $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_k$ uniformly at random. Consider the following two distributions on queries to A': the uniform distribution, and the distribution obtained by picking a random bad x and running the procedure on input x. We know A'

[2]In the terminology of Chapter 3, $f^{\oplus k}$ is a combinatorial checkerboard where all the component functions are f.

[3]In the statement of Theorem 8.2, it would be more accurate to say $s' = s/\big(\frac{4}{\epsilon}\ln\frac{2}{\delta}\big) - O(1)$ to account for the trivial circuitry needed to combine the results of the $\frac{4}{\epsilon}\ln\frac{2}{\delta}$ queries the reduction makes. Throughout this chapter, we ignore such details. We also ignore details arising from the fact that numbers such as $\frac{4}{\epsilon}\ln\frac{2}{\delta}$ might not be integers.

[4]Actually, their proof gives $s' = s/\big(k^2 \cdot \frac{2}{\epsilon}n\big)$, provided $k \ge \frac{1}{\delta}\ln\frac{2}{\epsilon}$, but a minor tweak to their proof yields the stated result.

computes f' with probability $< \beta = \epsilon/4$ under the latter distribution, and we wish to show that A' computes f' with probability $< \epsilon$ under the former. For this, we show the two distributions are "close" in the sense that the probability of any event under the former is less than twice the probability under the latter plus $\epsilon/2$. The argument involves a dichotomy: Since we assume a large fraction of x's are bad, a uniform query is unlikely to have few bad coordinates. Assuming there are many bad coordinates, we can essentially pretend there is one bad coordinate and then argue that we have overcounted the probability by a lot. This is the intuition for the ideas behind the proof of Theorem 8.2.

It can be shown that $\Omega\big(\frac{1}{\epsilon}\log\frac{1}{\delta}\big)$ queries are needed by any nonadaptive black-box reduction achieving errorless hardness amplification, regardless of how f' is constructed from f (see Section 8.1.3 for the precise statement). Since our proof of Theorem 8.2 (and the Bogdanov-Safra proof) is by a nonadaptive black-box reduction, this shows that Theorem 8.2 is optimal in a sense. Shaltiel and Viola [231] gave a general technique for lower bounding the query complexity of nonadaptive black-box reductions, and they noted that their technique applies to non-errorless hardness amplification (including the XOR Lemma and the Direct Product Lemma) and to constructions of pseudorandom generators from average-case hard functions. Similarly, we observe that their technique applies to errorless hardness amplification. Artemenko and Shaltiel [19] have proven a $\Omega\big(\frac{1}{\epsilon}\big)$ query lower bound even for *adaptive* black-box reductions. The optimal $\Omega\big(\frac{1}{\epsilon}\log\frac{1}{\delta}\big)$ lower bound for adaptive reductions remains open.

8.1.2 Monotone Errorless Amplification

Consider the problem of errorless hardness amplification within NP. That is, if f is computable in nondeterministic polynomial time, then we want f' to also be computable in nondeterministic polynomial time. Taking $f' = f^{\oplus k}$ does not guarantee this. We instead consider more general constructions of the form $f' = C \circ f^k$ where $C : \{0,1\}^k \to \{0,1\}$, and $f^k : \{0,1\}^{n\times k} \to \{0,1\}^k$ is defined as $f^k(x_1,\ldots,x_k) = \big(f(x_1),\ldots,f(x_k)\big)$. In the setting of the XOR Lemma, the combiner function C is the k-bit parity function. If C is monotone (that is, $C(y_1,\ldots,y_k) \le C(z_1,\ldots,z_k)$ whenever $y_i \le z_i$ for all $i \in [k]$) and f and C are both computable in nondeterministic polynomial time, then f' is guaranteed to be computable in nondeterministic polynomial time. This approach dates back to [203, 132].

Bogdanov and Safra [44] showed that this construction yields errorless hardness amplification provided the monotone combiner function C satisfies a certain combinatorial property. To describe this property, we need some definitions from [44] (though we use somewhat different notation). Fix $b \in \{0,1\}$. Given a monotone function $C : \{0,1\}^k \to \{0,1\}$ and a string $y \in \{0,1\}^k$, we say that coordinate $i \in [k]$ is *b-sensitive* if flipping the i^{th} bit of y causes the value of C to flip from b to $\overline{b}$, and we let $\sigma(C,y,b)$ denote the set of b-sensitive coordinates. That is,

$$\sigma(C,y,b) \;=\; \big\{i \in [k] \;:\; C(y) = b \text{ and } C(y \oplus e_i) = \overline{b}\big\}.$$

Note that if $C(y) = \overline{b}$ then $\sigma(C,y,b) = \emptyset$ and if $C(y) = b$ then by the monotonicity of C, $\sigma(C,y,b)$ only contains coordinates i such that $y_i = b$. For $p \in [0,1]$, we use $y \sim_p \{0,1\}^k$ to denote that y is sampled from the p-biased distribution, that is, each bit is independently set to 1 with probability p.

Definition 8.3. *For $b \in \{0,1\}$, a function $C : \{0,1\}^k \to \{0,1\}$ is a (t, ρ, p, b)-*amplifier *if C is monotone and*

$$\Pr_{y \sim_p \{0,1\}^k} \Big[\big| \sigma(C, y, b) \big| \geq t \Big] \;\geq\; 1 - \rho.$$

Note that a monotone function $C : \{0,1\}^k \to \{0,1\}$ is a (t, ρ, p, b)-amplifier if and only if its monotone complement $C^\dagger : \{0,1\}^k \to \{0,1\}$ is a $(t, \rho, 1-p, \overline{b})$-amplifier, where $C^\dagger$ is defined as

$$C^\dagger(y_1, \ldots, y_k) \;=\; \overline{C(\overline{y_1}, \ldots, \overline{y_k})}.$$

For reasons discussed in [44], it is necessary to consider the following one-sided version of Definition 8.1.

Definition 8.4 (One-Sided Errorless Average-Case Hardness). *For $b \in \{0,1\}$, we say a circuit $A : \{0,1\}^n \to \{0,1,\perp\}$ is a (δ, b)-*errorless *circuit for $f : \{0,1\}^n \to \{0,1\}$ if*

(i) $A(x) \in \{f(x), \perp\}$ for all $x \in \{0,1\}^n$, and

(ii) $\Pr_x[A(x) = \perp] \leq \delta$ where $x \in f^{-1}(b)$ is chosen uniformly at random.

*We say f is (s, δ, b)-*hard *if it has no (δ, b)-errorless circuit of size $\leq s$.*

Note that if f is (s, δ)-hard then f is either $(s/2, \delta, 0)$-hard or $(s/2, \delta, 1)$-hard.

Theorem 8.5 (Monotone Errorless Amplification Lemma). *For $b \in \{0,1\}$, if f is (s, δ, b)-hard and $C : \{0,1\}^k \to \{0,1\}$ is a (t, ρ, p, b)-amplifier then $f' = C \circ f^k$ is $(s', 1-\epsilon)$-hard where $s' = s / \big(\frac{k}{t} \cdot \frac{4}{\epsilon} \ln \frac{2}{\delta} \big)$, provided $t \geq \frac{16}{\delta} \ln \frac{4}{\epsilon}$, $\rho \leq \epsilon/4$, and $p = \Pr_x \big[f(x) = 1 \big]$.*

We prove Theorem 8.5 in Section 8.3. Bogdanov and Safra [44] proved a version of Theorem 8.5 where $s' = s / \big(k^3 \cdot \frac{64}{\epsilon^2} \ln \frac{2}{\delta} \big)$, provided $t \geq \frac{4}{\delta} \ln \frac{8}{\epsilon}$ and $\rho \leq \epsilon/2$. Their argument involves considering the subcubes of $\{0,1\}^{n \times k}$ given by $f^k(x_1, \ldots, x_k) = y$ for each y individually and then combining the results for the different subcubes using a nontrivial probabilistic argument. We show how to give a direct argument that handles all the subcubes simultaneously. This idea alone actually simplifies the proof and reduces the query complexity to $O\big(k^2 \cdot \frac{1}{\epsilon} \log \frac{1}{\delta}\big)$. Combining this idea with the ideas from our analysis in the proof of Theorem 8.2 allows us to further reduce the query complexity to $O\big(\frac{k}{t} \cdot \frac{1}{\epsilon} \log \frac{1}{\delta}\big)$. We believe this bound on the query complexity cannot be improved without exploiting some non-obvious structural property of (t, ρ, p, b)-amplifiers; however, we could not come up with a compelling way to formalize this.

Bogdanov and Safra [44] showed how to construct good amplifiers (with large t and small ρ) and how to use Theorem 8.5 with these amplifiers to do uniform and nonuniform errorless hardness amplification within NP.

8.1.3 Black-Box Lower Bounds

We give lower bounds on the query complexity and advice complexity of black-box errorless hardness amplification proofs. We allow ourselves to identify strings with functions; for example, we identify $\{0,1\}^{2^n}$ with the set of all functions from $\{0,1\}^n$ to $\{0,1\}$.

Definition 8.6. *An* $(n, n', \delta, \epsilon, \alpha)$-black-box errorless hardness amplification *is defined to be a pair* (Enc, Dec) *with* Enc : $\{0,1\}^{2^n} \to \{0,1\}^{2^{n'}}$ *and* Dec : $\{0,1,\perp\}^{2^{n'}} \times \{0,1\}^\alpha \to \{0,1,\perp\}^{2^n}$*, such that for all* $f \in \{0,1\}^{2^n}$ *and* $A' \in \{0,1,\perp\}^{2^{n'}}$ *there exists an* $a \in \{0,1\}^\alpha$ *such that the following holds, where* $f' = \mathrm{Enc}(f)$ *and* $A = \mathrm{Dec}(A', a)$*. If*

(i) $A'(x') \in \{f'(x'), \perp\}$ *for all* $x' \in \{0,1\}^{n'}$*, and*

(ii) $\Pr_{x'}[A'(x') = \perp] \leq 1 - \epsilon$,

then

(i) $A(x) \in \{f(x), \perp\}$ *for all* $x \in \{0,1\}^n$*, and*

(ii) $\Pr_x[A(x) = \perp] \leq \delta$.

We say it is q-query *if there is an algorithm that takes* (x, a) *as input, makes* q *(possibly adaptive) queries to* A'*, and outputs* $A(x)$*. We further say it is* nonadaptive *if the algorithm makes its queries to* A' *nonadaptively.*

In fact, our lower bounds hold even for the following weaker type of reduction.

Definition 8.7. *An* $(n, n', \delta, \epsilon, \alpha)$-black-box non-errorless to errorless hardness amplification *is defined to be a pair* (Enc, Dec) *with* Enc : $\{0,1\}^{2^n} \to \{0,1\}^{2^{n'}}$ *and* Dec : $\{0,1,\perp\}^{2^{n'}} \times \{0,1\}^\alpha \to \{0,1\}^{2^n}$*, such that for all* $f \in \{0,1\}^{2^n}$ *and* $A' \in \{0,1,\perp\}^{2^{n'}}$ *there exists an* $a \in \{0,1\}^\alpha$ *such that the following holds, where* $f' = \mathrm{Enc}(f)$ *and* $A = \mathrm{Dec}(A', a)$*. If*

(i) $A'(x') \in \{f'(x'), \perp\}$ *for all* $x' \in \{0,1\}^{n'}$*, and*

(ii) $\Pr_{x'}[A'(x') = \perp] \leq 1 - \epsilon$,

then $\Pr_x[A(x) \neq f(x)] \leq \delta$*. We say it is* q-query *if there is an algorithm that takes* (x, a) *as input, makes* q *(possibly adaptive) queries to* A'*, and outputs* $A(x)$*. We further say it is* nonadaptive *if the algorithm makes its queries to* A' *nonadaptively.*

In the proof of Theorem 8.2 we show that there exists a q-query nonadaptive $(n, n', \delta, \epsilon, \alpha)$-black-box errorless hardness amplification where $q = \frac{4}{\epsilon} \ln \frac{2}{\delta}$, $n' = kn$, and $\alpha = \big(\log_2 k + (k-1)n + 1\big) \cdot q$, provided $k \geq \frac{16}{\delta} \ln \frac{2}{\epsilon}$.

Theorem 8.8. *There exists a universal constant* $c > 1$ *such that the following holds. If there exists a* q*-query nonadaptive* $(n, n', \delta, \epsilon, \alpha)$*-black-box non-errorless to errorless hardness amplification then* $q \geq \frac{1}{c} \cdot \frac{1}{\epsilon} \ln \frac{1}{\delta}$*, provided* $n \geq c$*,* $n' \geq c$*,* $2^{-n/c} \leq \delta \leq 1/3$*,* $2^{-n/c} \leq \epsilon \leq 1/3$*, and* $\alpha \leq 2^{n/c}$*.*

Shaltiel and Viola [231] proved a similar result for the fully non-errorless setting (with the conclusion $q \geq \frac{1}{c} \cdot \frac{1}{\epsilon^2} \ln \frac{1}{\delta}$). Their proof can be adapted to our setting as follows. Where they use noise that flips a bit with probability $1/2 - \epsilon$, instead use noise that masks the bit with $\perp$ with probability $1 - \epsilon$ and reveals the correct bit with probability ϵ. Where they use noise that flips a bit with probability $1/2$, instead use "noise" that masks the bit with $\perp$ with probability 1. The rest of their proof goes through with some minor changes but without major changes.

Building on the techniques of [231], Artemenko and Shaltiel [19] proved the following lower bound, which applies to *adaptive* reductions but which falls short of the tight $\Omega\big(\frac{1}{\epsilon} \log \frac{1}{\delta}\big)$ lower bound by a factor of $\log \frac{1}{\delta}$.

Theorem 8.9. *There exists a universal constant $c > 1$ such that the following holds. If there exists a q-query $(n, n', \delta, \epsilon, \alpha)$-black-box non-errorless to errorless hardness amplification then $q \geq \frac{1}{c} \cdot \frac{1}{\epsilon}$, provided $n \geq c$, $n' \geq c$, $2^{-n/c} \leq \delta \leq 1/3$, $2^{-n/c} \leq \epsilon \leq 1/3$, and $\alpha \leq 2^{n/c}$.*

We now turn our attention to lower bounds on advice complexity.

Theorem 8.10. *If there exists an $(n, n', \delta, \epsilon, \alpha)$-black-box non-errorless to errorless hardness amplification then $2^\alpha \geq \log_3 \frac{1}{\epsilon}$, provided $\delta < 1/8$ and $\epsilon \geq 1/1.01^{1.01^{2^n}}$.*

We prove Theorem 8.10 in Section 8.4. In the fully non-errorless setting, lower bounds on advice complexity correspond to lower bounds on list size for approximately list-decoding error-correcting codes from flipped bits. Such a lower bound was given in [177] (see also [122]). In the non-errorless to errorless setting, lower bounds on advice complexity correspond to lower bounds on list size for approximately list-decoding error-correcting codes from erasures. For unique decoding, such lower bounds were given in [118, 61]. Our proof of Theorem 8.10 is simpler and cleaner than the proofs of the latter results (at the cost of achieving worse constants), and it handles approximate decoding. Also, the presentation in [118], which is geared toward coding theorists, views the rate, list size, and fraction of erasures as constants for an infinite family of codes. Our presentation is geared toward complexity theorists, who are interested more generally in the asymptotic relationships among all the parameters.

It is an open problem to prove some sort of uniform version of the Errorless XOR Lemma. Impagliazzo et al. [137] proved a sort of uniform version of the (non-errorless) XOR Lemma, but their techniques do not seem to apply to the errorless setting.

8.1.4 Preliminaries

We use the following standard Chernoff bound several times.

Theorem 8.11. *If $X_1, \ldots, X_\tau$ are fully independent indicator random variables each with expectation π, then $\Pr\big[\sum_{j=1}^{\tau} X_j < \pi\tau/2\big] < e^{-\pi\tau/8}$.*

8.2 Proof of Theorem 8.2

We prove the contrapositive. Suppose f' is not $(s', 1-\epsilon)$-hard and thus there is a circuit A' of size $\leq s'$ such that

(i) $A'(x_1, \ldots, x_k) \in \{f'(x_1, \ldots, x_k), \perp\}$ for all $x_1, \ldots, x_k$, and

(ii) $\Pr_{x_1,\ldots,x_k}\left[A'(x_1, \ldots, x_k) = \perp\right] \leq 1 - \epsilon$.

We give a nonuniform reduction that makes $\frac{4}{\epsilon} \ln \frac{2}{\delta}$ nonadaptive queries to A' and combines the results with some trivial computation, yielding a circuit A that witnesses that f is not (s, δ)-hard. To start out, we give a randomized algorithm (Algorithm 3) that solves f errorlessly using oracle access to A' and oracle access to f. The oracle queries to f only depend on the randomness (and not on the input), and later we will hard-wire a particular choice of randomness to get a circuit without oracle access to f.

Algorithm 3: Reduction for Theorem 8.2

```
  Input: $x \in \{0,1\}^n$
  Output: $f(x)$ or $\perp$
1 repeat $\frac{4}{\epsilon} \ln \frac{2}{\delta}$ times
2     pick $i \in [k]$ and $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_k \in \{0,1\}^n$ uniformly at random
3     if $A'(x_1, \ldots, x_{i-1}, x, x_{i+1}, \ldots, x_k) \neq \perp$ then halt and output
          $f(x_1) \oplus \cdots \oplus f(x_{i-1}) \oplus A'(x_1, \ldots, x_{i-1}, x, x_{i+1}, \ldots, x_k) \oplus f(x_{i+1}) \oplus \cdots \oplus f(x_k)$
4 end
5 halt and output $\perp$
```

Define the good set

$$G = \left\{x \in \{0,1\}^n : \Pr_{i, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_k}\left[A'(x_1, \ldots, x_{i-1}, x, x_{i+1}, \ldots, x_k) \neq \perp\right] \geq \epsilon/4\right\},$$

and define the bad set $B = \{0,1\}^n \backslash G$. That is, G is the set of inputs for which each iteration of the loop has at least an $\epsilon/4$ probability of producing output.

Claim 8.12. $|B| \leq (\delta/2) \cdot 2^n$.

Proof. Suppose for contradiction that $|B| > (\delta/2) \cdot 2^n$. Let $\gamma = |B|/2^{n+1}$. We define the event

$$W = \left\{(x_1, \ldots, x_k) \in \{0,1\}^{n \times k} : \left|\{i : x_i \in B\}\right| \geq \gamma k\right\}.$$

That is, W is the event that at least a γ fraction of coordinates are bad. We have

$$\Pr_{x_1,\ldots,x_k}\left[A'(x_1, \ldots, x_k) \neq \perp\right] \leq \Pr_{x_1,\ldots,x_k}\left[(x_1, \ldots, x_k) \notin W\right] + \Pr_{x_1,\ldots,x_k}\left[A'(x_1, \ldots, x_k) \neq \perp \text{ and } (x_1, \ldots, x_k) \in W\right].$$

We show that both terms on the right side are $< \epsilon/2$, thus contradicting property (ii) of A'.

Bounding the first term. Applying Theorem 8.11 with X_i as the indicator variable for $x_i \in B$, and with $\tau = k$ and $\pi = |B|/2^n$, we have

$$\Pr_{x_1,\ldots,x_k}\big[(x_1,\ldots,x_k) \notin W\big] \;<\; e^{-k\cdot|B|/2^{n+3}} \;<\; e^{-k\delta/16} \;\le\; \epsilon/2$$

where the middle inequality follows by our assumption on $|B|$ and the last inequality follows by $k \ge \frac{16}{\delta}\ln\frac{2}{\epsilon}$.

Bounding the second term. For each $S \subseteq [k]$ we define the event

$$W_S \;=\; \big\{(x_1,\ldots,x_k) \in \{0,1\}^{n\times k} \;:\; \forall i \;\; x_i \in B \Leftrightarrow i \in S\big\}.$$

Note that the W_S's are disjoint and

$$W \;=\; \bigcup_{S\,:\,|S|\ge\gamma k} W_S.$$

We have

$$\begin{aligned}
&\Pr_{x_1,\ldots,x_k}\big[A'(x_1,\ldots,x_k) \ne \perp \text{ and } (x_1,\ldots,x_k) \in W\big] \\
=\;& \sum_{S\,:\,|S|\ge\gamma k} \Pr_{x_1,\ldots,x_k}\big[A'(x_1,\ldots,x_k) \ne \perp \text{ and } (x_1,\ldots,x_k) \in W_S\big] \\
\le\;& \frac{1}{\gamma k}\sum_{S\subseteq[k]} |S| \cdot \Pr_{x_1,\ldots,x_k}\big[A'(x_1,\ldots,x_k) \ne \perp \text{ and } (x_1,\ldots,x_k) \in W_S\big] \\
=\;& \frac{1}{\gamma k}\sum_{i\in[k]}\sum_{S\ni i} \Pr_{x_1,\ldots,x_k}\big[A'(x_1,\ldots,x_k) \ne \perp \text{ and } (x_1,\ldots,x_k) \in W_S\big] \\
=\;& \frac{1}{\gamma k}\sum_{i\in[k]} \Pr_{x_1,\ldots,x_k}\big[A'(x_1,\ldots,x_k) \ne \perp \text{ and } x_i \in B\big] \\
=\;& \frac{1}{\gamma k}\sum_{i\in[k]}\sum_{x\in B} \Pr_{x_1,\ldots,x_k}\big[A'(x_1,\ldots,x_k) \ne \perp \;\big|\; x_i = x\big] \cdot \Pr_{x_1,\ldots,x_k}[x_i = x] \\
=\;& \frac{1}{\gamma k 2^n}\sum_{x\in B}\sum_{i\in[k]} \Pr_{x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_k}\big[A'(x_1,\ldots,x_{i-1},x,x_{i+1},\ldots,x_k) \ne \perp\big] \\
=\;& \frac{1}{\gamma k 2^n}\sum_{x\in B} k \cdot \Pr_{i,x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_k}\big[A'(x_1,\ldots,x_{i-1},x,x_{i+1},\ldots,x_k) \ne \perp\big] \\
<\;& \frac{1}{\gamma k 2^n}\sum_{x\in B} k \cdot \epsilon/4 \\
=\;& \frac{\epsilon/4}{\gamma 2^n} \cdot |B|
\end{aligned}$$

$$= \epsilon/2$$

where the second and fifth lines follow by the disjointness of the W_S's, and the remaining lines follow by simple rearrangements. □

The rest of the proof of Theorem 8.2 is similar to the argument from [44]. First we note that for all $x \in \{0,1\}^n$ and all choices of randomness, Algorithm 3 does indeed output either $f(x)$ or $\perp$. This follows trivially from the fact that if $A'(x_1, \dots, x_{i-1}, x, x_{i+1}, \dots, x_k) \neq \perp$ then

$$A'(x_1, \dots, x_{i-1}, x, x_{i+1}, \dots, x_k) \;=\; f(x_1) \oplus \cdots \oplus f(x_{i-1}) \oplus f(x) \oplus f(x_{i+1}) \oplus \cdots \oplus f(x_k)$$

by property (i) of A'. Next we observe that for each $x \in G$, we have

$$\Pr_{\text{randomness}} \big[\text{Algorithm 3 outputs } \perp\big] \;\leq\; \big(1 - \epsilon/4\big)^{\frac{4}{\epsilon}\ln\frac{2}{\delta}} \;\leq\; \delta/2.$$

Therefore

$$\begin{aligned} \Pr_{x,\text{ randomness}} \big[\text{Algorithm 3 outputs } \perp\big] &\leq \Pr_x \big[x \in B\big] + \\ &\quad \mathop{\mathrm{E}}_x \left[\Pr_{\text{randomness}} \big[\text{Algorithm 3 outputs } \perp\big] \,\middle|\, x \in G \right] \\ &\leq \delta/2 + \mathop{\mathrm{E}}_x \big[\delta/2 \bigm| x \in G\big] \\ &= \delta \end{aligned}$$

where the second inequality follows by Claim 8.12 and by the above observation. It follows that there exists a setting of the randomness such that

(i) Algorithm 3 outputs $f(x)$ or $\perp$ for all x, and

(ii) $\Pr_x \big[\text{Algorithm 3 outputs } \perp\big] \leq \delta$.

To get a circuit A that witnesses that f is not (s,δ)-hard, just hard-wire the randomness and the values of $f(x_1) \oplus \cdots \oplus f(x_{i-1}) \oplus f(x_{i+1}) \oplus \cdots \oplus f(x_k)$ needed for this choice of randomness, and plug in the hypothesized circuit A'. Since A' has size $\leq s'$ and Algorithm 3 makes $\frac{4}{\epsilon}\ln\frac{2}{\delta}$ queries to A', A has size $\leq s' \cdot \frac{4}{\epsilon}\ln\frac{2}{\delta} = s$. Note that Algorithm 3 can trivially be implemented with *nonadaptive* access to A'.

8.3 Proof of Theorem 8.5

We prove the contrapositive. Suppose f' is not $(s', 1-\epsilon)$-hard and thus there is a circuit A' of size $\leq s'$ such that

(i) $A'(x_1, \dots, x_k) \in \big\{f'(x_1, \dots, x_k), \perp\big\}$ for all $x_1, \dots, x_k$, and

(ii) $\Pr_{x_1,\dots,x_k}\left[A'(x_1,\dots,x_k)=\perp\right] \le 1-\epsilon$.

We give a nonuniform reduction that makes $\frac{k}{t}\cdot\frac{4}{\epsilon}\ln\frac{2}{\delta}$ nonadaptive queries to A' and combines the results with some trivial computation, yielding a circuit A that witnesses that f is not (s,δ,b)-hard. To start out, we give a randomized algorithm (Algorithm 4) that solves f errorlessly using oracle access to A' and oracle access to f and $\sigma(C,\cdot,b)$. The oracle queries to f and $\sigma(C,\cdot,b)$ only depend on the randomness (and not on the input), and later we will hard-wire a particular choice of randomness to get a circuit without oracle access to f or $\sigma(C,\cdot,b)$.

Algorithm 4: Reduction for Theorem 8.5

```
Input: $x \in \{0,1\}^n$
Output: $f(x)$ or $\perp$
1 repeat $\frac{k}{t}\cdot\frac{4}{\epsilon}\ln\frac{2}{\delta}$ times
2     pick $i \in [k]$ and $x_1,\dots,x_{i-1},x_{i+1},\dots,x_k \in \{0,1\}^n$ uniformly at random
3     if $A'(x_1,\dots,x_{i-1},x,x_{i+1},\dots,x_k) \neq \perp$ and $i \in \sigma(C,y,b)$ where
          $y = \big(f(x_1),\dots,f(x_{i-1}),b,f(x_{i+1}),\dots,f(x_k)\big)$
      then halt and output $A'(x_1,\dots,x_{i-1},x,x_{i+1},\dots,x_k)$
4 end
5 halt and output $\perp$
```

Define the good set

$$G = \Big\{x \in f^{-1}(b) \ : \ \Pr_{i,x_1,\dots,x_{i-1},x_{i+1},\dots,x_k}\left[A'(x_1,\dots,x_{i-1},x,x_{i+1},\dots,x_k)\neq\perp \text{ and } i\in\sigma(C,y,b)\right] \ge \epsilon t/4k\Big\}$$

where y is as in line 3 of Algorithm 4, and define the bad set $B = f^{-1}(b)\backslash G$. That is, G is the set of inputs in $f^{-1}(b)$ for which each iteration of the loop has at least an $\epsilon t/4k$ probability of producing output.

Claim 8.13. $|B| \le (\delta/2)\cdot\left|f^{-1}(b)\right|$.

Proof. Suppose for contradiction that $|B| > (\delta/2)\cdot\left|f^{-1}(b)\right|$. Let $\gamma = |B|/2\left|f^{-1}(b)\right|$. We define the event

$$W = \Big\{(x_1,\dots,x_k)\in\{0,1\}^{n\times k} \ : \ \Big|\Big\{i \ : \ x_i\in B \text{ and } i\in\sigma\big(C,f^k(x_1,\dots,x_k),b\big)\Big\}\Big| \ge \gamma t\Big\}.$$

That is, W is the event that at least a $\gamma t/k$ fraction of coordinates are both bad and b-sensitive. We have

$$\Pr_{x_1,\ldots,x_k}\big[A'(x_1,\ldots,x_k)\neq\bot\big] \;\le\; \Pr_{x_1,\ldots,x_k}\big[(x_1,\ldots,x_k)\notin W\big]+ \Pr_{x_1,\ldots,x_k}\big[A'(x_1,\ldots,x_k)\neq\bot \text{ and } (x_1,\ldots,x_k)\in W\big].$$

We show that both terms on the right side are $< \epsilon/2$, thus contradicting property (ii) of A'.

Bounding the first term. We have

$$\Pr_{x_1,\ldots,x_k}\big[(x_1,\ldots,x_k)\notin W\big] \;\le\; \Pr_{x_1,\ldots,x_k}\Big[\big|\sigma\big(C,f^k(x_1,\ldots,x_k),b\big)\big|<t\Big]+ \Pr_{x_1,\ldots,x_k}\Big[(x_1,\ldots,x_k)\notin W \;\Big|\; \big|\sigma\big(C,f^k(x_1,\ldots,x_k),b\big)\big|\ge t\Big].$$

To show that this is $< \epsilon/2$, we show that the first of the two terms on the right side is $\le \epsilon/4$ and the second is $< \epsilon/4$. Since C is a (t,ρ,p,b)-amplifier, we have

$$\Pr_{x_1,\ldots,x_k}\Big[\big|\sigma\big(C,f^k(x_1,\ldots,x_k),b\big)\big|<t\Big] \;=\; \Pr_{y\sim_p\{0,1\}^k}\Big[\big|\sigma(C,y,b)\big|<t\Big] \;\le\; \rho \;\le\; \epsilon/4.$$

We have

$$\begin{aligned}&\Pr_{x_1,\ldots,x_k}\Big[(x_1,\ldots,x_k)\notin W \;\Big|\; \big|\sigma\big(C,f^k(x_1,\ldots,x_k),b\big)\big|\ge t\Big]\\ =\;& \mathop{\mathrm{E}}_{y\sim_p\{0,1\}^k}\Big[\Pr_{x_1,\ldots,x_k}\Big[(x_1,\ldots,x_k)\notin W \;\Big|\; f^k(x_1,\ldots,x_k)=y\Big] \;\Big|\; \big|\sigma(C,y,b)\big|\ge t\Big].\end{aligned}$$

Fix any $y\in\{0,1\}^k$ such that $\big|\sigma(C,y,b)\big|\ge t$, and for now let us abbreviate $\sigma(C,y,b)$ as σ. Then we have

$$\begin{aligned}&\Pr_{x_1,\ldots,x_k}\Big[(x_1,\ldots,x_k)\notin W \;\Big|\; f^k(x_1,\ldots,x_k)=y\Big]\\ =\;& \mathop{\mathrm{E}}_{(x_i)_{i\notin\sigma}}\Big[\Pr_{(x_i)_{i\in\sigma}}\Big[(x_1,\ldots,x_k)\notin W \;\Big|\; f(x_i)=b\;\;\forall i\in\sigma\Big] \;\Big|\; f(x_i)=y_i\;\;\forall i\notin\sigma\Big]\end{aligned}$$

since $\sigma\subseteq\big\{i\;:\;y_i=b\big\}$. Now fix any $(x_i)_{i\notin\sigma}$ such that $f(x_i)=y_i$ for all $i\notin\sigma$. Then we have

$$\begin{aligned}&\Pr_{(x_i)_{i\in\sigma}}\Big[(x_1,\ldots,x_k)\notin W \;\Big|\; f(x_i)=b\;\;\forall i\in\sigma\Big]\\ =\;& \Pr_{(x_i)_{i\in\sigma}}\Big[\big|\big\{i\in\sigma\;:\;x_i\in B\big\}\big|<\gamma t \;\Big|\; f(x_i)=b\;\;\forall i\in\sigma\Big]\\ \le\;& \Pr_{(x_i)_{i\in\sigma}}\Big[\big|\big\{i\in\sigma\;:\;x_i\in B\big\}\big|<\gamma\cdot|\sigma| \;\Big|\; f(x_i)=b\;\;\forall i\in\sigma\Big]\end{aligned}$$

where the inequality follows by $t \le |\sigma|$. Applying Theorem 8.11 with X_j as the indicator variable for $x_i \in B$ where i is the j^{th} value in σ and x_i is chosen uniformly from $f^{-1}(b)$, and with $\tau = |\sigma|$ and $\pi = |B|/\big|f^{-1}(b)\big|$, we have that the latter quantity is less than

$$e^{-|\sigma|\cdot|B|/8|f^{-1}(b)|} \;<\; e^{-|\sigma|\cdot\delta/16} \;\le\; e^{-t\delta/16} \;\le\; \epsilon/4$$

where the first inequality follows by our assumption on $|B|$, the middle inequality follows by $|\sigma| \ge t$, and the last inequality follows by $t \ge \frac{16}{\delta}\ln\frac{4}{\epsilon}$. This establishes that

$$\Pr_{x_1,\ldots,x_k}\Big[(x_1,\ldots,x_k) \notin W \;\Big|\; \big|\sigma\big(C, f^k(x_1,\ldots,x_k), b\big)\big| \ge t\Big] \;<\; \epsilon/4.$$

Bounding the second term. This is similar to the corresponding part of the analysis in the proof of Theorem 8.2. For each $S \subseteq [k]$ we define the event

$$W_S \;=\; \Big\{(x_1,\ldots,x_k) \in \{0,1\}^{n\times k} \;:\; \forall i \;\;\Big(x_i \in B \text{ and } i \in \sigma\big(C, f^k(x_1,\ldots,x_k), b\big)\Big) \Leftrightarrow i \in S\Big\}.$$

Note that the W_S's are disjoint and

$$W \;=\; \bigcup_{S\,:\,|S|\ge\gamma t} W_S.$$

Using the shorthand y as in line 3 of Algorithm 4, we have

$$\begin{aligned}
&\Pr_{x_1,\ldots,x_k}\big[A'(x_1,\ldots,x_k) \ne \bot \text{ and } (x_1,\ldots,x_k) \in W\big]\\
=\;& \sum_{S\,:\,|S|\ge\gamma t}\Pr_{x_1,\ldots,x_k}\big[A'(x_1,\ldots,x_k) \ne \bot \text{ and } (x_1,\ldots,x_k) \in W_S\big]\\
\le\;& \frac{1}{\gamma t}\sum_{S\subseteq[k]}|S|\cdot\Pr_{x_1,\ldots,x_k}\big[A'(x_1,\ldots,x_k) \ne \bot \text{ and } (x_1,\ldots,x_k) \in W_S\big]\\
=\;& \frac{1}{\gamma t}\sum_{i\in[k]}\sum_{S\ni i}\Pr_{x_1,\ldots,x_k}\big[A'(x_1,\ldots,x_k) \ne \bot \text{ and } (x_1,\ldots,x_k) \in W_S\big]\\
=\;& \frac{1}{\gamma t}\sum_{i\in[k]}\Pr_{x_1,\ldots,x_k}\Big[A'(x_1,\ldots,x_k) \ne \bot \text{ and } x_i \in B \text{ and } i \in \sigma\big(C, f^k(x_1,\ldots,x_k), b\big)\Big]\\
=\;& \frac{1}{\gamma t}\sum_{i\in[k]}\sum_{x\in B}\Pr_{x_1,\ldots,x_k}\Big[A'(x_1,\ldots,x_k) \ne \bot \text{ and } i \in \sigma\big(C, f^k(x_1,\ldots,x_k), b\big) \;\Big|\; x_i = x\Big]\cdot\\
&\qquad\qquad \Pr_{x_1,\ldots,x_k}[x_i = x]\\
=\;& \frac{1}{\gamma t 2^n}\sum_{x\in B}\sum_{i\in[k]}\Pr_{x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_k}\big[A'(x_1,\ldots,x_{i-1},x,x_{i+1},\ldots,x_k) \ne \bot \text{ and } i \in \sigma(C,y,b)\big]\\
=\;& \frac{1}{\gamma t 2^n}\sum_{x\in B}k\cdot\Pr_{i,x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_k}\big[A'(x_1,\ldots,x_{i-1},x,x_{i+1},\ldots,x_k) \ne \bot \text{ and } i \in \sigma(C,y,b)\big]
\end{aligned}$$

$$
\begin{aligned}
&< \frac{1}{\gamma t 2^n} \sum_{x \in B} k \cdot \epsilon t/4k \\
&= \frac{\epsilon/4}{\gamma 2^n} \cdot |B| \\
&\le \epsilon/2
\end{aligned}
$$

where the second and fifth lines follow by the disjointness of the W_S's, the last line follows by $\left|f^{-1}(b)\right| \le 2^n$, and the remaining lines follow by simple rearrangements. For the seventh line, we used the fact that $x \in B$ implies $f(x) = b$ and thus $y = f^k(x_1, \ldots, x_{i-1}, x, x_{i+1}, \ldots, x_k)$.$\square$

We now finish the proof of Theorem 8.5. First we note that for all $x \in \{0,1\}^n$ and all choices of randomness, Algorithm 4 does indeed output either $f(x)$ or $\perp$. This follows trivially from the facts that $A'(x_1, \ldots, x_{i-1}, x, x_{i+1}, \ldots, x_k) \ne \perp$ implies

$$A'(x_1, \ldots, x_{i-1}, x, x_{i+1}, \ldots, x_k) \;=\; f'(x_1, \ldots, x_{i-1}, x, x_{i+1}, \ldots, x_k)$$

by property (i) of A', and $i \in \sigma(C, y, b)$ implies $f'(x_1, \ldots, x_{i-1}, x, x_{i+1}, \ldots, x_k) = f(x)$, where y is as in line 3 of Algorithm 4. Next we observe that for each $x \in G$, we have

$$\Pr_{\text{randomness}}\big[\text{Algorithm 4 outputs } \perp\big] \;\le\; \big(1 - \epsilon t/4k\big)^{\frac{k}{t} \cdot \frac{4}{\epsilon} \ln \frac{2}{\delta}} \;\le\; \delta/2.$$

Therefore, picking $x \in f^{-1}(b)$ uniformly at random, we have

$$
\begin{aligned}
\Pr_{x,\ \text{randomness}}\big[\text{Algorithm 4 outputs } \perp\big] &\le \Pr_x\big[x \in B\big] + \\
&\quad \mathop{\mathrm{E}}_x\left[\Pr_{\text{randomness}}\big[\text{Algorithm 4 outputs } \perp\big] \;\middle|\; x \in G\right] \\
&\le \delta/2 + \mathop{\mathrm{E}}_x\big[\delta/2 \;\big|\; x \in G\big] \\
&= \delta
\end{aligned}
$$

where the second inequality follows by Claim 8.13 and by the above observation. It follows that there exists a setting of the randomness such that

(i) Algorithm 4 outputs $f(x)$ or $\perp$ for all $x \in \{0,1\}^n$, and

(ii) $\Pr_x\big[\text{Algorithm 4 outputs } \perp\big] \le \delta$ where $x \in f^{-1}(b)$ is chosen uniformly at random.

To get a circuit A that witnesses that f is not (s, δ, b)-hard, just hard-wire the randomness and the correct responses to the $\sigma(C, \cdot, b)$ queries (which only depend on the randomness and not on x), and plug in the hypothesized circuit A'. In fact, the iterations for which $i \notin \sigma(C, y, b)$ for this particular choice of randomness can simply be eliminated. Since A' has size $\le s'$ and Algorithm 4 makes $\le \frac{k}{t} \cdot \frac{4}{\epsilon} \ln \frac{2}{\delta}$ queries to A', A has size $\le s' \cdot \frac{k}{t} \cdot \frac{4}{\epsilon} \ln \frac{2}{\delta} = s$. Note that Algorithm 4 can trivially be implemented with *nonadaptive* access to A'.

8.4 Proof of Theorem 8.10

Let $(\mathrm{Enc}, \mathrm{Dec})$ be an $(n, n', \delta, \epsilon, \alpha)$-black-box non-errorless to errorless hardness amplification with $\delta < 1/8$ and $\epsilon \geq 1/1.01^{1.01^{2^n}}$. We use the notation $\Delta(f_1, f_2)$ for the relative Hamming distance between bit strings f_1 and f_2. We begin with a completely standard claim that asserts the existence of a good error-correcting code. We include the proof for completeness.

Claim 8.14. *There exists an $F \subseteq \{0,1\}^{2^n}$ such that*

(1) $\forall f_1, f_2 \in F$: if $f_1 \neq f_2$ then $\Delta(f_1, f_2) > 2\delta$, and

(2) $|F| = 4\log_3 \frac{1}{\epsilon}$.

Proof. Pick $F \subseteq \{0,1\}^{2^n}$ randomly by choosing $f_1, \ldots, f_{4\log_3 \frac{1}{\epsilon}} \in \{0,1\}^{2^n}$ independently uniformly at random and setting $F = \bigl\{f_1, \ldots, f_{4\log_3 \frac{1}{\epsilon}}\bigr\}$. To prove (1) and (2), it suffices to show that

$$\Pr\bigl[\exists i_1, i_2 \in \{1, \ldots, 4\log_3 \tfrac{1}{\epsilon}\} : i_1 \neq i_2 \text{ and } \Delta(f_{i_1}, f_{i_2}) \leq 2\delta\bigr] \;<\; 1.$$

For each pair $i_1 \neq i_2$, since $\delta < 1/8$ we have

$$\Pr\bigl[\Delta(f_{i_1}, f_{i_2}) \leq 2\delta\bigr] \;\leq\; \Pr\bigl[\Delta(f_{i_1}, f_{i_2}) < 1/4\bigr] \;<\; e^{-2^n/16}$$

by applying Theorem 8.11 with X_j as the indicator variable for $f_{i_1}(x) \neq f_{i_2}(x)$ where x is the j^{th} string in $\{0,1\}^n$, and with $\tau = 2^n$ and $\pi = 1/2$. By a union bound, the probability in question is at most $\bigl(4\log_3 \frac{1}{\epsilon}\bigr)^2 \cdot e^{-2^n/16} < 1$ since $\epsilon \geq 1/1.01^{1.01^{2^n}}$. $\square$

For the rest of the proof of Theorem 8.10 we fix a set F as in Claim 8.14.

Claim 8.15. *There exists an $E \subseteq F$ and an $A' \in \{0, 1, \perp\}^{2^{n'}}$ such that*

(i) $\forall f \in E : A'(x') \in \{f'(x'), \perp\}$ for all $x' \in \{0,1\}^{n'}$, where $f' = \mathrm{Enc}(f)$, and

(ii) $\Pr_{x'}[A'(x') = \perp] \leq 1 - \epsilon$, and

(iii) $|E| = \log_3 \frac{1}{\epsilon}$.

Proof. For each $x' \in \{0,1\}^{n'}$ let $m_{x'} = \mathrm{majority}_{f \in F} \mathrm{Enc}(f)(x')$, breaking a tie arbitrarily, and define

$$M_{x'} \;=\; \bigl\{f \in F \;:\; \mathrm{Enc}(f)(x') = m_{x'}\bigr\}.$$

For any set $E \subseteq F$, define

$$M_E^{-1} \;=\; \bigl\{x' \in \{0,1\}^{n'} \;:\; E \subseteq M_{x'}\bigr\}.$$

We construct a sequence of sets $E_0 \subseteq E_1 \subseteq E_2 \subseteq \cdots \subseteq E_{\log_3 \frac{1}{\epsilon}} \subseteq F$ such that for all i, $|E_i| = i$ and $\left|M_{E_i}^{-1}\right| \geq 2^{n'}/3^i$. Then we can take $E = E_{\log_3 \frac{1}{\epsilon}}$ and

$$A'(x') \;=\; \begin{cases} m_{x'} & \text{if } x' \in M_E^{-1} \\ \perp & \text{otherwise} \end{cases}$$

and (i), (ii), and (iii) all follow immediately. We do the construction inductively. The base case $i = 0$ is trivial. Now assume $i \in \left\{0, 1, \ldots, \log_3 \frac{1}{\epsilon} - 1\right\}$ and we have a set $E_i \subseteq F$ with $|E_i| = i$ and $\left|M_{E_i}^{-1}\right| \geq 2^{n'}/3^i$. For each $x' \in M_{E_i}^{-1}$, since $i \leq |F|/4$ we have

$$\Pr_{f \in F \backslash E_i}[f \in M_{x'}] \;=\; \frac{|M_{x'} \backslash E_i|}{|F \backslash E_i|} \;\geq\; \frac{\frac{1}{2}|F| - i}{|F| - i} \;\geq\; 1/3$$

where f is chosen uniformly at random. Thus for some $f \in F \backslash E_i$ we have $\Pr_{x' \in M_{E_i}^{-1}}[f \in M_{x'}] \geq 1/3$ where x' is chosen uniformly at random. For this fixed f, setting $E_{i+1} = E_i \cup \{f\}$ we have $|E_{i+1}| = |E_i| + 1 = i + 1$ and

$$\left|M_{E_{i+1}}^{-1}\right| \;=\; \left|\left\{x' \in M_{E_i}^{-1} \;:\; f \in M_{x'}\right\}\right| \;=\; \left|M_{E_i}^{-1}\right| \cdot \Pr_{x' \in M_{E_i}^{-1}}[f \in M_{x'}] \;\geq\; \left|M_{E_i}^{-1}\right|/3 \;\geq\; 2^{n'}/3^{i+1}.$$

This finishes the induction step. □

Now to prove Theorem 8.10, suppose for contradiction that $2^\alpha < \log_3 \frac{1}{\epsilon}$. Then by the pigeonhole principle there must exist $f_1, f_2 \in E$ such that $f_1 \neq f_2$ and the advice string corresponding to f_1 and A' equals the advice string corresponding to f_2 and A'. Call this advice string a, and let $A = \mathrm{Dec}(A', a)$. By Definition 8.7, we must have $\Delta(A, f_1) \leq \delta$ and $\Delta(A, f_2) \leq \delta$. But this is impossible because $\Delta(f_1, f_2) > 2\delta$ by property (1) in Claim 8.14.

Bibliography

[1] Scott Aaronson. The equivalence of sampling and searching. In *Proceedings of the 6th International Computer Science Symposium in Russia*, pages 1–14, 2011.

[2] Scott Aaronson, Barış Aydınlıoğlu, Harry Buhrman, John Hitchcock, and Dieter van Melkebeek. A note on exponential circuit lower bounds from derandomizing Arthur-Merlin games. Technical Report TR10-174, Electronic Colloquium on Computational Complexity, 2010.

[3] Scott Aaronson and Dieter van Melkebeek. On circuit lower bounds from derandomization. *Theory of Computing*, 7(1):177–184, 2011.

[4] Miklós Ajtai, János Komlós, and Endre Szemerédi. Deterministic simulation in LOGSPACE. In *Proceedings of the 19th ACM Symposium on Theory of Computing*, pages 132–140, 1987.

[5] Miklós Ajtai and Avi Wigderson. Deterministic simulation of probabilistic constant-depth circuits. *Advances in Computing Research – Randomness and Computation*, 5:199–223, 1989.

[6] Noga Alon and Fan Chung. Explicit construction of linear sized tolerant networks. *Discrete Mathematics*, 72(1-3):15–19, 1988.

[7] Noga Alon, Zvi Galil, and Vitali Milman. Better expanders and superconcentrators. *Journal of Algorithms*, 8(3):337–347, 1987.

[8] Noga Alon, Oded Goldreich, Johan Håstad, and René Peralta. Simple constructions of almost k-wise independent random variables. *Random Structures and Algorithms*, 3(3):289–304, 1992.

[9] Noga Alon and Vitali Milman. λ_1, isoperimetric inequalities for graphs, and superconcentrators. *Journal of Combinatorial Theory, Series B*, 38(1):73–88, 1985.

[10] Benny Applebaum. Pseudorandom generators with long stretch and low locality from random local one-way functions. In *Proceedings of the 44th ACM Symposium on Theory of Computing*, pages 805–816, 2012.

[11] Benny Applebaum, Andrej Bogdanov, and Alon Rosen. A dichotomy for local small-bias generators. In *Proceedings of the 9th Theory of Cryptography Conference*, pages 600–617, 2012.

[12] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0. *SIAM Journal on Computing*, 36(4):845–888, 2006.

[13] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. On pseudorandom generators with linear stretch in NC^0. *Computational Complexity*, 17(1):38–69, 2008.

[14] Roy Armoni. On the derandomization of space-bounded computations. In *Proceedings of the 2nd International Workshop on Randomization and Computation*, pages 47–59, 1998.

[15] Roy Armoni, Michael Saks, Avi Wigderson, and Shiyu Zhou. Discrepancy sets and pseudorandom generators for combinatorial rectangles. In *Proceedings of the 37th IEEE Symposium on Foundations of Computer Science*, pages 412–421, 1996.

[16] Roy Armoni, Amnon Ta-Shma, Avi Wigderson, and Shiyu Zhou. An $O(\log^{4/3}(n))$ space algorithm for (s, t) connectivity in undirected graphs. *Journal of the ACM*, 47(2):294–311, 2000.

[17] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach.* Cambridge University Press, 2009.

[18] Sanjeev Arora, David Steurer, and Avi Wigderson. Towards a study of low-complexity graphs. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming*, pages 119–131, 2009.

[19] Sergei Artemenko and Ronen Shaltiel. Lower bounds on the query complexity of non-uniform and adaptive reductions showing hardness amplification. In *Proceedings of the 15th International Workshop on Randomization and Computation*, pages 377–388, 2011.

[20] Vikraman Arvind and Johannes Köbler. On pseudorandomness and resource-bounded measure. *Theoretical Computer Science*, 255(1-2):205–221, 2001.

[21] Barış Aydınlıoğlu, Dan Gutfreund, John Hitchcock, and Akinori Kawachi. Derandomizing Arthur-Merlin games and approximate counting implies exponential-size lower bounds. *Computational Complexity*, 20(2):329–366, 2011.

[22] László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3:307–318, 1993.

[23] László Babai, Noam Nisan, and Mario Szegedy. Multiparty protocols, pseudorandom generators for logspace, and time-space trade-offs. *Journal of Computer and System Sciences*, 45(2):204–232, 1992.

[24] Eric Bach. How to generate factored random numbers. *SIAM Journal on Computing*, 17(2):179–193, 1988.

[25] Boaz Barak. A probabilistic-time hierarchy theorem for slightly non-uniform algorithms. In *Proceedings of the 6th International Workshop on Randomization and Computation*, pages 194–208, 2002.

[26] Boaz Barak, Moritz Hardt, and Satyen Kale. The uniform hardcore lemma via approximate Bregman projections. In *Proceedings of the 20th ACM-SIAM Symposium on Discrete Algorithms*, pages 1193–1200, 2009.

[27] Boaz Barak, Russell Impagliazzo, and Avi Wigderson. Extracting randomness using few independent sources. *SIAM Journal on Computing*, 36(4):1095–1118, 2006.

[28] Boaz Barak, Guy Kindler, Ronen Shaltiel, Benny Sudakov, and Avi Wigderson. Simulating independence: New constructions of condensers, Ramsey graphs, dispersers, and extractors. *Journal of the ACM*, 57(4), 2010.

[29] Boaz Barak, Anup Rao, Ronen Shaltiel, and Avi Wigderson. 2-source dispersers for sub-polynomial entropy and Ramsey graphs beating the Frankl-Wilson construction. In *Proceedings of the 38th ACM Symposium on Theory of Computing*, pages 671–680, 2006.

[30] Boaz Barak, Ronen Shaltiel, and Avi Wigderson. Computational analogues of entropy. In *Proceedings of the 7th International Workshop on Randomization and Computation*, pages 200–215, 2003.

[31] Louay Bazzi. Polylogarithmic independence can fool DNF formulas. *SIAM Journal on Computing*, 38(6):2220–2272, 2009.

[32] Christopher Beck, Russell Impagliazzo, and Shachar Lovett. Large deviation bounds for decision trees and sampling lower bounds for AC^0-circuits. In *Proceedings of the 53rd IEEE Symposium on Foundations of Computer Science*, pages 101–110, 2012.

[33] Mihir Bellare and John Rompel. Randomness-efficient oblivious sampling. In *Proceedings of the 35th IEEE Symposium on Foundations of Computer Science*, pages 276–287, 1994.

[34] Eli Ben-Sasson and Ariel Gabizon. Extractors for polynomials sources over constant-size fields of small characteristic. In *Proceedings of the 16th International Workshop on Randomization and Computation*, pages 399–410, 2012.

[35] Charles Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. Strengths and weaknesses of quantum computing. *SIAM Journal on Computing*, 26(5):1510–1523, 1997.

[36] Nayantara Bhatnagar, Andrej Bogdanov, and Elchanan Mossel. The computational complexity of estimating MCMC convergence time. In *Proceedings of the 15th International Workshop on Randomization and Computation*, pages 424–435, 2011.

[37] Nayantara Bhatnagar, Sam Greenberg, and Dana Randall. Sampling stable marriages: Why spouse-swapping won't work. In *Proceedings of the 19th ACM-SIAM Symposium on Discrete Algorithms*, pages 1223–1232, 2008.

[38] Volodia Blinovsky. Bounds for codes in the case of list decoding of finite volume. *Problems of Information Transmission*, 22(1):7–19, 1986.

[39] Andrej Bogdanov. Pseudorandom generators for low degree polynomials. In *Proceedings of the 37th ACM Symposium on Theory of Computing*, pages 21–30, 2005.

[40] Andrej Bogdanov, Zeev Dvir, Elad Verbin, and Amir Yehudayoff. Pseudorandomness for width 2 branching programs. Technical Report TR09-070, Electronic Colloquium on Computational Complexity, 2009.

[41] Andrej Bogdanov, Elchanan Mossel, and Salil Vadhan. The complexity of distinguishing Markov random fields. In *Proceedings of the 12th International Workshop on Randomization and Computation*, pages 331–342, 2008.

[42] Andrej Bogdanov, Periklis Papakonstantinou, and Andrew Wan. Pseudorandomness for read-once formulas. In *Proceedings of the 52nd IEEE Symposium on Foundations of Computer Science*, pages 240–246, 2011.

[43] Andrej Bogdanov and Youming Qiao. On the security of Goldreich's one-way function. *Computational Complexity*, 21(1):83–127, 2012.

[44] Andrej Bogdanov and Muli Safra. Hardness amplification for errorless heuristics. In *Proceedings of the 48th IEEE Symposium on Foundations of Computer Science*, pages 418–426, 2007.

[45] Andrej Bogdanov and Luca Trevisan. Average-case complexity. *Foundations and Trends in Theoretical Computer Science*, 2(1), 2006.

[46] Andrej Bogdanov and Luca Trevisan. On worst-case to average-case reductions for NP problems. *SIAM Journal on Computing*, 36(4):1119–1159, 2006.

[47] Andrej Bogdanov and Emanuele Viola. Pseudorandom bits for polynomials. *SIAM Journal on Computing*, 39(6):2464–2486, 2010.

[48] Elmar Böhler, Christian Glaßer, and Daniel Meister. Error-bounded probabilistic computations between MA and AM. *Journal of Computer and System Sciences*, 72(6):1043–1076, 2006.

[49] Jean Bourgain. More on the sum-product phenomenon in prime fields and its applications. *International Journal of Number Theory*, 1:1–32, 2005.

[50] Jean Bourgain. On the construction of affine-source extractors. *Geometric and Functional Analysis*, 1:33–57, 2007.

[51] Mark Braverman. Polylogarithmic independence fools AC^0 circuits. *Journal of the ACM*, 57(5), 2010.

[52] Mark Braverman, Anup Rao, Ran Raz, and Amir Yehudayoff. Pseudorandom generators for regular branching programs. In *Proceedings of the 51st IEEE Symposium on Foundations of Computer Science*, pages 40–47, 2010.

[53] Joshua Brody and Elad Verbin. The coin problem, and pseudorandomness for branching programs. In *Proceedings of the 51st IEEE Symposium on Foundations of Computer Science*, pages 30–39, 2010.

[54] Russ Bubley and Martin Dyer. Faster random generation of linear extensions. *Discrete Mathematics*, 201(1-3):81–88, 1999.

[55] Joshua Buresh-Oppenheim, Valentine Kabanets, and Rahul Santhanam. Uniform hardness amplification in NP via monotone codes. Technical Report TR06-154, Electronic Colloquium on Computational Complexity, 2006.

[56] Jin-Yi Cai, Venkatesan Chakaravarthy, and Dieter van Melkebeek. Time-space tradeoff in derandomizing probabilistic logspace. *Theory of Computing Systems*, 39(1):189–208, 2006.

[57] Prasad Chebolu, Mary Cryan, and Russell Martin. Exact counting of Euler tours for generalized series-parallel graphs. *Journal of Discrete Algorithms*, 10:110–122, 2012.

[58] Mahdi Cheraghchi and Amin Shokrollahi. Almost-uniform sampling of points on high-dimensional algebraic varieties. In *Proceedings of the 26th International Symposium on Theoretical Aspects of Computer Science*, pages 277–288, 2009.

[59] Benny Chor, Joel Friedman, Oded Goldreich, Johan Håstad, Steven Rudich, and Roman Smolensky. The bit extraction problem or t-resilient functions. In *Proceedings of the 26th IEEE Symposium on Foundations of Computer Science*, pages 396–407, 1985.

[60] Benny Chor and Oded Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM Journal on Computing*, 17(2):230–261, 1988.

[61] Gérard Cohen, Simon Litsyn, and Gilles Zémor. Upper bounds on generalized distances. *IEEE Transactions on Information Theory*, 40(6):2090–2092, 1994.

[62] James Cook, Omid Etesami, Rachel Miller, and Luca Trevisan. Goldreich's one-way function candidate and myopic backtracking algorithms. In *Proceedings of the 6th Theory of Cryptography Conference*, pages 521–538, 2009.

[63] Stephen Cook. A hierarchy for nondeterministic time complexity. *Journal of Computer and System Sciences*, 7(4):343–353, 1973.

[64] Mary Cryan, Martin Dyer, and Dana Randall. Approximately counting integral flows and cell-bounded contingency tables. *SIAM Journal on Computing*, 39(7):2683–2703, 2010.

[65] Mary Cryan and Peter Bro Miltersen. On pseudorandom generators in NC^0. In *Proceedings of the 26th International Symposium on Mathematical Foundations of Computer Science*, pages 272–284, 2001.

[66] Anindya De. Pseudorandomness for permutation and regular branching programs. In *Proceedings of the 26th IEEE Conference on Computational Complexity*, pages 221–231, 2011.

[67] Anindya De, Omid Etesami, Luca Trevisan, and Madhur Tulsiani. Improved pseudorandom generators for depth 2 circuits. In *Proceedings of the 14th International Workshop on Randomization and Computation*, pages 504–517, 2010.

[68] Anindya De and Luca Trevisan. Extractors using hardness amplification. In *Proceedings of the 13th International Workshop on Randomization and Computation*, pages 462–475, 2009.

[69] Anindya De and Thomas Watson. Extractors and lower bounds for locally samplable sources. In *Proceedings of the 15th International Workshop on Randomization and Computation*, pages 483–494, 2011.

[70] Anindya De and Thomas Watson. Extractors and lower bounds for locally samplable sources. *ACM Transactions on Computation Theory*, 4(1), 2012.

[71] Matt DeVos and Ariel Gabizon. Simple affine extractors using dimension expansion. In *Proceedings of the 25th IEEE Conference on Computational Complexity*, pages 50–57, 2010.

[72] Ilias Diakonikolas, Parikshit Gopalan, Ragesh Jaiswal, Rocco Servedio, and Emanuele Viola. Bounded independence fools halfspaces. *SIAM Journal on Computing*, 39(8):3441–3462, 2010.

[73] Ilias Diakonikolas, Daniel Kane, and Jelani Nelson. Bounded independence fools degree-2 threshold functions. In *Proceedings of the 51st IEEE Symposium on Foundations of Computer Science*, pages 11–20, 2010.

[74] Yevgeniy Dodis, Ariel Elbaz, Roberto Oliveira, and Ran Raz. Improved randomness extraction from two independent sources. In *Proceedings of the 8th International Workshop on Randomization and Computation*, pages 334–344, 2004.

[75] Yevgeniy Dodis, Thomas Ristenpart, and Salil Vadhan. Randomness condensers for efficiently samplable, seed-dependent sources. In *Proceedings of the 9th Theory of Cryptography Conference*, pages 618–635, 2012.

[76] Zeev Dvir. Extractors for varieties. *Computational Complexity*, 21(4):515–572, 2012.

[77] Zeev Dvir, Ariel Gabizon, and Avi Wigderson. Extractors and rank extractors for polynomial sources. *Computational Complexity*, 18(1):1–58, 2009.

[78] Zeev Dvir, Dan Gutfreund, Guy Rothblum, and Salil Vadhan. On approximating the entropy of polynomial mappings. In *Proceedings of the 2nd Innovations in Computer Science Conference*, pages 460–475, 2011.

[79] Martin Dyer, Leslie Ann Goldberg, Catherine Greenhill, and Mark Jerrum. The relative complexity of approximate counting problems. *Algorithmica*, 38(3):471–500, 2003.

[80] Stefan Dziembowski and Ueli Maurer. Optimal randomizer efficiency in the bounded-storage model. *Journal of Cryptology*, 17(1):5–26, 2004.

[81] Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *Proceedings of the 49th IEEE Symposium on Foundations of Computer Science*, pages 293–302, 2008.

[82] Paul Erdős, Péter Frankl, and Zoltán Füredi. Families of finite sets in which no set is covered by the union of r others. *Israel Journal of Mathematics*, 51(1-2):79–89, 1985.

[83] Guy Even, Oded Goldreich, Michael Luby, Noam Nisan, and Boban Velickovic. Efficient approximation of product distributions. *Random Structures and Algorithms*, 13(1):1–16, 1998.

[84] Joan Feigenbaum and Lance Fortnow. Random-self-reducibility of complete sets. *SIAM Journal on Computing*, 22(5):994–1005, 1993.

[85] Lance Fortnow, Russell Impagliazzo, Valentine Kabanets, and Christopher Umans. On the complexity of succinct zero-sum games. *Computational Complexity*, 17(3):353–376, 2008.

[86] Lance Fortnow and Adam Klivans. Linear advice for randomized logarithmic space. In *Proceedings of the 23rd International Symposium on Theoretical Aspects of Computer Science*, pages 469–476, 2006.

[87] Lance Fortnow and Rahul Santhanam. Hierarchy theorems for probabilistic polynomial time. In *Proceedings of the 45th IEEE Symposium on Foundations of Computer Science*, pages 316–324, 2004.

[88] Lance Fortnow and Rahul Santhanam. Robust simulations and significant separations. In *Proceedings of the 38th International Colloquium on Automata, Languages and Programming*, pages 569–580, 2011.

[89] Lance Fortnow, Rahul Santhanam, and Luca Trevisan. Hierarchies for semantic classes. In *Proceedings of the 37th ACM Symposium on Theory of Computing*, pages 348–355, 2005.

[90] Alan Frieze and Eric Vigoda. A survey on the use of Markov chains to randomly sample colorings. In *Combinatorics, Complexity, and Chance*. Oxford University Press, 2007.

[91] Merrick Furst, James Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, 1984.

[92] Ofer Gabber and Zvi Galil. Explicit constructions of linear-sized superconcentrators. *Journal of Computer and System Sciences*, 22(3):407–420, 1981.

[93] Ariel Gabizon and Ran Raz. Deterministic extractors for affine sources over large fields. *Combinatorica*, 28(4):415–440, 2008.

[94] Ariel Gabizon, Ran Raz, and Ronen Shaltiel. Deterministic extractors for bit-fixing sources by obtaining an independent seed. *SIAM Journal on Computing*, 36(4):1072–1094, 2006.

[95] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *Proceedings of the 43rd ACM Symposium on Theory of Computing*, pages 99–108, 2011.

[96] Leslie Ann Goldberg and Mark Jerrum. Randomly sampling molecules. *SIAM Journal on Computing*, 29(3):834–853, 1999.

[97] Oded Goldreich. On promise problems: A survey. In *Essays in Memory of Shimon Even*, pages 254–290. Springer, 2006.

[98] Oded Goldreich. *Computational Complexity: A Conceptual Perspective.* Cambridge University Press, 2008.

[99] Oded Goldreich. Candidate one-way functions based on expander graphs. *Studies in Complexity and Cryptography*, pages 76–87, 2011.

[100] Oded Goldreich. In a world of P = BPP. *Studies in Complexity and Cryptography*, pages 191–232, 2011.

[101] Oded Goldreich, Shafi Goldwasser, and Asaf Nussboim. On the implementation of huge random objects. *SIAM Journal on Computing*, 39(7):2761–2822, 2010.

[102] Oded Goldreich, Noam Nisan, and Avi Wigderson. On Yao's XOR-Lemma. *Studies in Complexity and Cryptography*, pages 273–301, 2011.

[103] Oded Goldreich, Amit Sahai, and Salil Vadhan. Can statistical zero knowledge be made non-interactive? or On the relationship of SZK and NISZK. In *Proceedings of the 19th International Cryptology Conference*, pages 467–484, 1999.

[104] Oded Goldreich, Madhu Sudan, and Luca Trevisan. From logarithmic advice to single-bit advice. *Studies in Complexity and Cryptography*, pages 109–113, 2011.

[105] Oded Goldreich and Avi Wigderson. Tiny families of functions with random properties: A quality-size trade-off for hashing. *Random Structures and Algorithms*, 11(4):315–343, 1997.

[106] Shafi Goldwasser, Dan Gutfreund, Alexander Healy, Tali Kaufman, and Guy Rothblum. Verifying and decoding in constant depth. In *Proceedings of the 39th ACM Symposium on Theory of Computing*, pages 440–449, 2007.

[107] Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. In *Proceedings of the 18th ACM Symposium on Theory of Computing*, pages 59–68, 1986.

[108] Parikshit Gopalan and Venkatesan Guruswami. Hardness amplification within NP against deterministic algorithms. *Journal of Computer and System Sciences*, 77(1):107–121, 2011.

[109] Parikshit Gopalan, Raghu Meka, Omer Reingold, and David Zuckerman. Pseudorandom generators for combinatorial shapes. In *Proceedings of the 43rd ACM Symposium on Theory of Computing*, pages 253–262, 2011.

[110] Parikshit Gopalan, Ryan O'Donnell, Yi Wu, and David Zuckerman. Fooling functions of halfspaces under product distributions. In *Proceedings of the 25th IEEE Conference on Computational Complexity*, pages 223–234, 2010.

[111] Vivek Gore, Mark Jerrum, Sampath Kannan, Z. Sweedyk, and Stephen Mahaney. A quasi-polynomial-time algorithm for sampling words from a context-free language. *Information and Computation*, 134(1):59–74, 1997.

[112] Timothy Gowers. Decompositions, approximate structure, transference, and the Hahn-Banach Theorem. *Bulletin of the London Mathematical Society*, 42(4):573–606, 2010.

[113] Timothy Gowers and Julia Wolf. Linear forms and higher-degree uniformity for functions on $\mathbb{F}_p^n$. *Geometric and Functional Analysis*, 21(1):36–69, 2011.

[114] Timothy Gowers and Julia Wolf. Linear forms and quadratic uniformity for functions on $\mathbb{F}_p^n$. *Mathematika*, 57(2):215–237, 2012.

[115] Ben Green and Terence Tao. The primes contain arbitrarily long arithmetic progressions. *Annals of Mathematics*, 167(2):481–547, 2008.

[116] Dima Grigoriev, Edward Hirsch, and Konstantin Pervyshev. Time hierarchies for cryptographic function inversion with advice. Technical Report TR05-076, Electronic Colloquium on Computational Complexity, 2005.

[117] Venkatesan Guruswami. List decoding from erasures: Bounds and code constructions. *IEEE Transactions on Information Theory*, 49(11):2826–2833, 2003.

[118] Venkatesan Guruswami. Chapter 10: List decoding from erasures. In *List Decoding of Error-Correcting Codes (Lecture Notes in Computer Science)*, volume 3282. Springer-Verlag, 2005.

[119] Venkatesan Guruswami and Piotr Indyk. Linear time encodable and list decodable codes. In *Proceedings of the 35th ACM Symposium on Theory of Computing*, pages 126–135, 2003.

[120] Venkatesan Guruswami and Atri Rudra. Explicit codes achieving list decoding capacity: Error-correction with optimal redundancy. *IEEE Transactions on Information Theory*, 54(1):135–150, 2008.

[121] Venkatesan Guruswami, Christopher Umans, and Salil Vadhan. Unbalanced expanders and randomness extractors from Parvaresh-Vardy codes. *Journal of the ACM*, 56(4), 2009.

[122] Venkatesan Guruswami and Salil Vadhan. A lower bound on list size for list decoding. *IEEE Transactions on Information Theory*, 56(11):5681–5688, 2010.

[123] Dan Gutfreund and Guy Rothblum. The complexity of local list decoding. In *Proceedings of the 12th International Workshop on Randomization and Computation*, pages 455–468, 2008.

[124] Dan Gutfreund, Ronen Shaltiel, and Amnon Ta-Shma. If NP languages are hard on the worst-case, then it is easy to find their hard instances. *Computational Complexity*, 16(4):412–441, 2007.

[125] Dan Gutfreund and Amnon Ta-Shma. Worst-case to average-case reductions revisited. In *Proceedings of the 11th International Workshop on Randomization and Computation*, pages 569–583, 2007.

[126] Yenjo Han, Lane Hemaspaandra, and Thomas Thierauf. Threshold computation and cryptographic security. *SIAM Journal on Computing*, 26(1):59–78, 1997.

[127] Prahladh Harsha, Adam Klivans, and Raghu Meka. An invariance principle for polytopes. In *Proceedings of the 42nd ACM Symposium on Theory of Computing*, pages 543–552, 2010.

[128] Juris Hartmanis and Richard Stearns. On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, 117:285–306, 1965.

[129] Johan Håstad. Almost optimal lower bounds for small depth circuits. In *Proceedings of the 18th ACM Symposium on Theory of Computing*, pages 6–20, 1986.

[130] Johan Håstad. One-way permutations in NC^0. *Information Processing Letters*, 26(3):153–155, 1987.

[131] Johan Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica*, 182:105–142, 1999.

[132] Alexander Healy, Salil Vadhan, and Emanuele Viola. Using nondeterminism to amplify hardness. *SIAM Journal on Computing*, 35(4):903–931, 2006.

[133] Russell Impagliazzo. Hard-core distributions for somewhat hard problems. In *Proceedings of the 36th IEEE Symposium on Foundations of Computer Science*, pages 538–545, 1995.

[134] Russell Impagliazzo. A personal view of average-case complexity. In *Proceedings of the 10th IEEE Conference on Structure in Complexity Theory*, pages 134–147, 1995.

[135] Russell Impagliazzo. Relativized separations of worst-case and average-case complexities for NP. In *Proceedings of the 26th IEEE Conference on Computational Complexity*, pages 104–114, 2011.

[136] Russell Impagliazzo, Ragesh Jaiswal, and Valentine Kabanets. Approximate list-decoding of direct product codes and uniform hardness amplification. *SIAM Journal on Computing*, 39(2):564–605, 2009.

[137] Russell Impagliazzo, Ragesh Jaiswal, Valentine Kabanets, and Avi Wigderson. Uniform direct product theorems: Simplified, optimized, and derandomized. *SIAM Journal on Computing*, 39(4):1637–1665, 2010.

[138] Russell Impagliazzo and Valentine Kabanets. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004.

[139] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: Exponential time vs. probabilistic polynomial time. *Journal of Computer and System Sciences*, 65(4):672–694, 2002.

[140] Russell Impagliazzo and Leonid Levin. No better ways to generate hard NP instances than picking uniformly at random. In *Proceedings of the 31st IEEE Symposium on Foundations of Computer Science*, pages 812–821, 1990.

[141] Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography. In *Proceedings of the 30th IEEE Symposium on Foundations of Computer Science*, pages 230–235, 1989.

[142] Russell Impagliazzo, Noam Nisan, and Avi Wigderson. Pseudorandomness for network algorithms. In *Proceedings of the 26th ACM Symposium on Theory of Computing*, pages 356–364, 1994.

[143] Russell Impagliazzo, Ronen Shaltiel, and Avi Wigderson. Reducing the seed length in the Nisan-Wigderson generator. *Combinatorica*, 26(6):647–681, 2006.

[144] Russell Impagliazzo and Avi Wigderson. P = BPP if E requires exponential circuits: Derandomizing the XOR Lemma. In *Proceedings of the 29th ACM Symposium on Theory of Computing*, pages 220–229, 1997.

[145] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography with constant computational overhead. In *Proceedings of the 40th ACM Symposium on Theory of Computing*, pages 433–442, 2008.

[146] Dmitry Itsykson. Structural complexity of AvgBPP. *Annals of Pure and Applied Logic*, 162(3):213–223, 2010.

[147] Mark Jerrum and Alistair Sinclair. Approximating the permanent. *SIAM Journal on Computing*, 18(6):1149–1178, 1989.

[148] Mark Jerrum, Alistair Sinclair, and Eric Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. *Journal of the ACM*, 51(4):671–697, 2004.

[149] Mark Jerrum, Leslie Valiant, and Vijay Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science*, 43:169–188, 1986.

[150] Shuji Jimbo and Akira Maruoka. Expanders obtained from affine transformations. *Combinatorica*, 7(4):343–355, 1987.

[151] Valentine Kabanets, Charles Rackoff, and Stephen Cook. Efficiently approximable real-valued functions. Technical Report TR00-034, Electronic Colloquium on Computational Complexity, 2000.

[152] Adam Kalai. Generating random factored numbers, easily. *Journal of Cryptology*, 16(4):287–289, 2003.

[153] Jesse Kamp, Anup Rao, Salil Vadhan, and David Zuckerman. Deterministic extractors for small-space sources. *Journal of Computer and System Sciences*, 77(1):191–220, 2011.

[154] Jesse Kamp and David Zuckerman. Deterministic extractors for bit-fixing sources and exposure-resilient cryptography. *SIAM Journal on Computing*, 36(5):1231–1247, 2007.

[155] Ravi Kannan, Prasad Tetali, and Santosh Vempala. Simple Markov-chain algorithms for generating bipartite graphs and tournaments. *Random Structures and Algorithms*, 14(4):293–308, 1999.

[156] Richard Karp, Michael Luby, and Neal Madras. Monte-Carlo approximation algorithms for enumeration problems. *Journal of Algorithms*, 10(3):429–448, 1989.

[157] Jonathan Kelner and Aleksander Madry. Faster generation of random spanning trees. In *Proceedings of the 50th IEEE Symposium on Foundations of Computer Science*, pages 13–21, 2009.

[158] Iordanis Kerenidis and Ronald de Wolf. Exponential lower bound for 2-query locally decodable codes via a quantum argument. *Journal of Computer and System Sciences*, 69(3):395–420, 2004.

[159] Jeff Kinne and Dieter van Melkebeek. Space hierarchy results for randomized and other semantic models. *Computational Complexity*, 19(3):423–475, 2010.

[160] Adam Klivans. On the derandomization of constant depth circuits. In *Proceedings of the 5th International Workshop on Randomization and Computation*, pages 249–260, 2001.

[161] Adam Klivans and Rocco Servedio. Boosting and hard-core sets. *Machine Learning*, 53(3):217–238, 2003.

[162] Adam Klivans and Dieter van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. *SIAM Journal on Computing*, 31(5):1501–1526, 2002.

[163] Michal Koucký, Prajakta Nimbhorkar, and Pavel Pudlák. Pseudorandom generators for group products. In *Proceedings of the 43rd ACM Symposium on Theory of Computing*, pages 263–272, 2011.

[164] Greg Kuperberg. How hard is it to approximate the Jones polynomial? *CoRR*, abs/0908.0512, 2009.

[165] Leonid Levin. Average case complete problems. *SIAM Journal on Computing*, 15(1):285–286, 1986.

[166] Leonid Levin. One-way functions and pseudorandom generators. *Combinatorica*, 7(4):357–363, 1987.

[167] Xin Li. Improved constructions of three source extractors. In *Proceedings of the 26th IEEE Conference on Computational Complexity*, pages 126–136, 2011.

[168] Xin Li. A new approach to affine extractors and dispersers. In *Proceedings of the 26th IEEE Conference on Computational Complexity*, pages 137–147, 2011.

[169] Nathan Linial, Michael Luby, Michael Saks, and David Zuckerman. Efficient construction of a small hitting set for combinatorial rectangles in high dimension. *Combinatorica*, 17(2):215–234, 1997.

[170] Shachar Lovett. Unconditional pseudorandom generators for low degree polynomials. *Theory of Computing*, 5(1):69–82, 2009.

[171] Shachar Lovett, Partha Mukhopadhyay, and Amir Shpilka. Pseudorandom generators for $\mathrm{CC}_0[p]$ and the Fourier spectrum of low-degree polynomials over finite fields. In *Proceedings of the 51st IEEE Symposium on Foundations of Computer Science*, pages 695–704, 2010.

[172] Shachar Lovett, Omer Reingold, Luca Trevisan, and Salil Vadhan. Pseudorandom bit generators that fool modular sums. In *Proceedings of the 13th International Workshop on Randomization and Computation*, pages 615–630, 2009.

[173] Shachar Lovett and Emanuele Viola. Bounded-depth circuits cannot sample good codes. *Computational Complexity*, 21(2):245–266, 2012.

[174] Chi-Jen Lu. Improved pseudorandom generators for combinatorial rectangles. *Combinatorica*, 22(3):417–434, 2002.

[175] Chi-Jen Lu. Encryption against storage-bounded adversaries from on-line strong extractors. *Journal of Cryptology*, 17(1):27–42, 2004.

[176] Chi-Jen Lu, Shi-Chun Tsai, and Hsin-Lung Wu. Improved hardness amplification in NP. *Theoretical Computer Science*, 370(1-3):293–298, 2007.

[177] Chi-Jen Lu, Shi-Chun Tsai, and Hsin-Lung Wu. On the complexity of hardness amplification. *IEEE Transactions on Information Theory*, 54(10):4575–4586, 2008.

[178] Chi-Jen Lu, Shi-Chun Tsai, and Hsin-Lung Wu. Complexity of hard-core set proofs. *Computational Complexity*, 20(1):145–171, 2011.

[179] Alexander Lubotzky, Ralph Phillips, and Peter Sarnak. Ramanujan graphs. *Combinatorica*, 8(3):261–277, 1988.

[180] Michael Luby, Dana Randall, and Alistair Sinclair. Markov chain algorithms for planar lattice structures. *SIAM Journal on Computing*, 31(1):167–192, 2001.

[181] Michael Luby and Boban Velickovic. On deterministic approximation of DNF. *Algorithmica*, 16(4-5):415–433, 1996.

[182] Michael Luby, Boban Velickovic, and Avi Wigderson. Deterministic approximate counting of depth-2 circuits. In *Proceedings of the 2nd Israel Symposium on Theory of Computing Systems*, pages 18–24, 1993.

[183] Michael Luby and Eric Vigoda. Fast convergence of the Glauber dynamics for sampling independent sets. *Random Structures and Algorithms*, 15(3-4):229–241, 1999.

[184] Rune Lyngsø and Christian Pedersen. The consensus string problem and the complexity of comparing hidden Markov models. *Journal of Computer and System Sciences*, 65(3):545–569, 2002.

[185] Gregory Margulis. Explicit constructions of expanders. *Problemy Peredaci Informacii*, 9(4):71–80, 1973.

[186] Gregory Margulis. Explicit group-theoretic constructions of combinatorial schemes and their applications in the construction of expanders and concentrators. *Problems of Information Transmission*, 24(1):39–46, 1988.

[187] Raghu Meka and David Zuckerman. Small-bias spaces for group products. In *Proceedings of the 13th International Workshop on Randomization and Computation*, pages 658–672, 2009.

[188] Raghu Meka and David Zuckerman. Pseudorandom generators for polynomial threshold functions. In *Proceedings of the 42nd ACM Symposium on Theory of Computing*, pages 427–436, 2010.

[189] Peter Bro Miltersen and N. V. Vinodchandran. Derandomizing Arthur-Merlin games using hitting sets. *Computational Complexity*, 14(3):256–279, 2005.

[190] Ilya Mironov, Omkant Pandey, Omer Reingold, and Salil Vadhan. Computational differential privacy. In *Proceedings of the 29th International Cryptology Conference*, pages 126–142, 2009.

[191] Moshe Morgenstern. Existence and explicit constructions of $q + 1$ regular Ramanujan graphs for every prime power q. *Journal of Combinatorial Theory, Series B*, 62(1):44–62, 1994.

[192] Ben Morris and Alistair Sinclair. Random walks on truncated cubes and sampling 0-1 knapsack solutions. *SIAM Journal on Computing*, 34(1):195–226, 2004.

[193] Elchanan Mossel, Amir Shpilka, and Luca Trevisan. On epsilon-biased generators in NC^0. *Random Structures and Algorithms*, 29(1):56–81, 2006.

[194] Elchanan Mossel and Christopher Umans. On the complexity of approximating the VC dimension. *Journal of Computer and System Sciences*, 65(4):660–671, 2002.

[195] Joseph Naor and Moni Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM Journal on Computing*, 22(4):838–856, 1993.

[196] Noam Nisan. Pseudorandom bits for constant depth circuits. *Combinatorica*, 11(1):63–70, 1991.

[197] Noam Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992.

[198] Noam Nisan. On read-once vs. multiple access to randomness in logspace. *Theoretical Computer Science*, 107(1):135–144, 1993.

[199] Noam Nisan. RL $\subseteq$ SC. *Computational Complexity*, 4:1–11, 1994.

[200] Noam Nisan, Endre Szemerédi, and Avi Wigderson. Undirected connectivity in $O(\log^{1.5}(n))$ space. In *Proceedings of the 33rd IEEE Symposium on Foundations of Computer Science*, pages 24–29, 1992.

[201] Noam Nisan and Avi Wigderson. Hardness vs. randomness. *Journal of Computer and System Sciences*, 49(2):149–167, 1994.

[202] Noam Nisan and David Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, 52(1):43–52, 1996.

[203] Ryan O'Donnell. Hardness amplification within NP. *Journal of Computer and System Sciences*, 69(1):68–94, 2004.

[204] Konstantin Pervyshev. Time hierarchies for computations with a bit of advice. Technical Report TR05-054, Electronic Colloquium on Computational Complexity, 2005.

[205] Konstantin Pervyshev. On heuristic time hierarchies. In *Proceedings of the 22nd IEEE Conference on Computational Complexity*, pages 347–358, 2007.

[206] James Propp and David Wilson. How to get a perfectly random sample from a generic Markov chain and generate a random spanning tree of a directed graph. *Journal of Algorithms*, 27(2):170–217, 1998.

[207] Anup Rao. A 2-source almost-extractor for linear entropy. In *Proceedings of the 12th International Workshop on Randomization and Computation*, pages 549–556, 2008.

[208] Anup Rao. Extractors for a constant number of polynomially small min-entropy independent sources. *SIAM Journal on Computing*, 39(1):168–194, 2009.

[209] Anup Rao. Extractors for low-weight affine sources. In *Proceedings of the 24th IEEE Conference on Computational Complexity*, pages 95–101, 2009.

[210] Anup Rao and David Zuckerman. Extractors for three uneven-length sources. In *Proceedings of the 12th International Workshop on Randomization and Computation*, pages 557–570, 2008.

[211] Ran Raz. Extractors with weak random seeds. In *Proceedings of the 37th ACM Symposium on Theory of Computing*, pages 11–20, 2005.

[212] Ran Raz and Omer Reingold. On recycling the randomness of states in space bounded computation. In *Proceedings of the 31st ACM Symposium on Theory of Computing*, pages 159–168, 1999.

[213] Ran Raz, Omer Reingold, and Salil Vadhan. Error reduction for extractors. In *Proceedings of the 40th IEEE Symposium on Foundations of Computer Science*, pages 191–201, 1999.

[214] Ran Raz, Omer Reingold, and Salil Vadhan. Extracting all the randomness and reducing the error in Trevisan's extractors. *Journal of Computer and System Sciences*, 65(1):97–128, 2002.

[215] Ran Raz and Amir Yehudayoff. Multilinear formulas, maximal-partition discrepancy and mixed-sources extractors. *Journal of Computer and System Sciences*, 77(1):167–190, 2011.

[216] Alexander Razborov. A simple proof of Bazzi's Theorem. *ACM Transactions on Computation Theory*, 1(1), 2009.

[217] Oded Regev. Lattice-based cryptography. In *Proceedings of the 26th International Cryptology Conference*, pages 131–141, 2006.

[218] Omer Reingold. Undirected connectivity in log-space. *Journal of the ACM*, 55(4), 2008.

[219] Omer Reingold, Luca Trevisan, Madhur Tulsiani, and Salil Vadhan. Dense subsets of pseudorandom sets. In *Proceedings of the 49th IEEE Symposium on Foundations of Computer Science*, pages 76–85, 2008.

[220] Omer Reingold, Luca Trevisan, and Salil Vadhan. Pseudorandom walks on regular digraphs and the RL vs. L problem. In *Proceedings of the 38th ACM Symposium on Theory of Computing*, pages 457–466, 2006.

[221] Omer Reingold, Salil Vadhan, and Avi Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders. *Annals of Mathematics*, 155(1):157–187, 2002.

[222] Eyal Rozenman and Salil Vadhan. Derandomized squaring of graphs. In *Proceedings of the 9th International Workshop on Randomization and Computation*, pages 436–447, 2005.

[223] Michael Saks and Shiyu Zhou. $\mathrm{BP_H SPACE}(S) \subseteq \mathrm{DSPACE}(S^{3/2})$. *Journal of Computer and System Sciences*, 58(2):376–403, 1999.

[224] Jeanette Schmidt, Alan Siegel, and Aravind Srinivasan. Chernoff-Hoeffding bounds for applications with limited independence. *SIAM Journal on Discrete Mathematics*, 8(2):223–250, 1995.

[225] Joel Seiferas, Michael Fischer, and Albert Meyer. Separating nondeterministic time complexity classes. *Journal of the ACM*, 25(1):146–167, 1978.

[226] Ronen Shaltiel. Recent developments in explicit constructions of extractors. *Bulletin of the European Association for Theoretical Computer Science*, 77:67–95, 2002.

[227] Ronen Shaltiel. How to get more mileage from randomness extractors. *Random Structures and Algorithms*, 33(2):157–186, 2008.

[228] Ronen Shaltiel. An introduction to randomness extractors. In *Proceedings of the 38th International Colloquium on Automata, Languages and Programming*, pages 21–41, 2011.

[229] Ronen Shaltiel and Christopher Umans. Simple extractors for all min-entropies and a new pseudorandom generator. *Journal of the ACM*, 52(2):172–216, 2005.

[230] Ronen Shaltiel and Christopher Umans. Pseudorandomness for approximate counting and sampling. *Computational Complexity*, 15(4):298–341, 2006.

[231] Ronen Shaltiel and Emanuele Viola. Hardness amplification proofs require majority. *SIAM Journal on Computing*, 39(7):3122–3154, 2010.

[232] Adi Shamir. IP = PSPACE. *Journal of the ACM*, 39(4):869–877, 1992.

[233] Alexander Shen. IP = PSPACE: Simplified proof. *Journal of the ACM*, 39(4):878–880, 1992.

[234] Victor Shoup. New algorithms for finding irreducible polynomials over finite fields. In *Proceedings of the 29th IEEE Symposium on Foundations of Computer Science*, pages 283–290, 1988.

[235] Jiří Šíma and Stanislav Žák. A polynomial time construction of a hitting set for read-once branching programs of width 3. Technical Report TR10-088, Electronic Colloquium on Computational Complexity, 2010.

[236] Alistair Sinclair. *Algorithms for Random Generation and Counting: A Markov Chain Approach.* Progress in Theoretical Computer Science. Birkhäuser, 1993.

[237] Larry Stockmeyer. On approximation algorithms for #P. *SIAM Journal on Computing*, 14(4):849–861, 1985.

[238] Madhu Sudan, Luca Trevisan, and Salil Vadhan. Pseudorandom generators without the XOR Lemma. *Journal of Computer and System Sciences*, 62(2):236–266, 2001.

[239] Terence Tao and Tamar Ziegler. The primes contain arbitrarily long polynomial progressions. *Acta Mathematica*, 201:213–305, 2008.

[240] Yael Tauman Kalai, Xin Li, and Anup Rao. 2-source extractors under computational assumptions and cryptography with defective randomness. In *Proceedings of the 50th IEEE Symposium on Foundations of Computer Science*, pages 617–626, 2009.

[241] Luca Trevisan. Extractors and pseudorandom generators. *Journal of the ACM*, 48(4):860–879, 2001.

[242] Luca Trevisan. List decoding using the XOR Lemma. In *Proceedings of the 44th IEEE Symposium on Foundations of Computer Science*, pages 126–135, 2003.

[243] Luca Trevisan. A note on approximate counting for k-DNF. In *Proceedings of the 8th International Workshop on Randomization and Computation*, pages 417–426, 2004.

[244] Luca Trevisan. On uniform amplification of hardness in NP. In *Proceedings of the 37th ACM Symposium on Theory of Computing*, pages 31–38, 2005.

[245] Luca Trevisan, Madhur Tulsiani, and Salil Vadhan. Regularity, boosting, and efficiently simulating every high-entropy distribution. In *Proceedings of the 24th IEEE Conference on Computational Complexity*, pages 126–136, 2009.

[246] Luca Trevisan and Salil Vadhan. Extracting randomness from samplable distributions. In *Proceedings of the 41st IEEE Symposium on Foundations of Computer Science*, pages 32–42, 2000.

[247] Luca Trevisan, Salil Vadhan, and David Zuckerman. Compression of samplable sources. *Computational Complexity*, 14(3):186–227, 2005.

[248] Christopher Umans. Pseudo-random generators for all hardnesses. *Journal of Computer and System Sciences*, 67(2):419–440, 2003.

[249] Salil Vadhan. Constructing locally computable extractors and cryptosystems in the bounded-storage model. *Journal of Cryptology*, 17(1):43–77, 2004.

[250] Salil Vadhan. Pseudorandomness. *Foundations and Trends in Theoretical Computer Science*, 7(1-3), 2012.

[251] Salil Vadhan. *A Study of Statistical Zero-Knowledge Proofs*. Springer, 2013.

[252] Leslie Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8:189–201, 1979.

[253] Leslie Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.

[254] Dieter van Melkebeek and Konstantin Pervyshev. A generic time hierarchy with one bit of advice. *Computational Complexity*, 16(2):139–179, 2007.

[255] Nikolai Vereshchagin. On the power of PP. In *Proceedings of the 7th Structure in Complexity Theory Conference*, pages 138–143, 1992.

[256] Eric Vigoda. A note on the Glauber dynamics for sampling independent sets. *Electronic Journal of Combinatorics*, 8(1), 2001.

[257] Emanuele Viola. The complexity of constructing pseudorandom generators from hard functions. *Computational Complexity*, 13(3-4):147–188, 2005.

[258] Emanuele Viola. On constructing parallel pseudorandom generators from one-way functions. In *Proceedings of the 20th IEEE Conference on Computational Complexity*, pages 183–197, 2005.

[259] Emanuele Viola. Pseudorandom bits for constant-depth circuits with few arbitrary symmetric gates. *SIAM Journal on Computing*, 36(5):1387–1403, 2007.

[260] Emanuele Viola. The sum of d small-bias generators fools polynomials of degree d. *Computational Complexity*, 18(2):209–217, 2009.

[261] Emanuele Viola. Extractors for circuit sources. In *Proceedings of the 52nd IEEE Symposium on Foundations of Computer Science*, pages 220–229, 2011.

[262] Emanuele Viola. The complexity of distributions. *SIAM Journal on Computing*, 41(1):191–218, 2012.

[263] Emanuele Viola. Extractors for Turing-machine sources. In *Proceedings of the 16th International Workshop on Randomization and Computation*, pages 663–671, 2012.

[264] Thomas Watson. Relativized worlds without worst-case to average-case reductions for NP. In *Proceedings of the 14th International Workshop on Randomization and Computation*, pages 752–765, 2010.

[265] Thomas Watson. Relativized worlds without worst-case to average-case reductions for NP. Technical Report TR10-042, Electronic Colloquium on Computational Complexity, 2010.

[266] Thomas Watson. Advice lower bounds for the dense model theorem. Technical Report TR11-120, Electronic Colloquium on Computational Complexity, 2011.

[267] Thomas Watson. Pseudorandom generators for combinatorial checkerboards. In *Proceedings of the 26th IEEE Conference on Computational Complexity*, pages 232–242, 2011.

[268] Thomas Watson. Query complexity in errorless hardness amplification. In *Proceedings of the 15th International Workshop on Randomization and Computation*, pages 688–699, 2011.

[269] Thomas Watson. The complexity of estimating min-entropy. Technical Report TR12-070, Electronic Colloquium on Computational Complexity, 2012.

[270] Thomas Watson. Pseudorandom generators for combinatorial checkerboards. *Computational Complexity*, Online First, 2012.

[271] Thomas Watson. Relativized worlds without worst-case to average-case reductions for NP. *ACM Transactions on Computation Theory*, 4(3), 2012.

[272] Thomas Watson. Time hierarchies for sampling distributions. In *Proceedings of the 4th Innovations in Theoretical Computer Science Conference*, pages 429–439, 2013.

[273] David Wilson. Generating random spanning trees more quickly than the cover time. In *Proceedings of the 28th ACM Symposium on Theory of Computing*, pages 296–303, 1996.

[274] David Wilson. Mixing times of lozenge tiling and card shuffling Markov chains. *Annals of Applied Probability*, 14(1):274–325, 2004.

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

[276] Amir Yehudayoff. Affine extractors over prime fields. *Combinatorica*, 31(2):245–256, 2011.

[277] Stanislav Žák. A Turing machine time hierarchy. *Theoretical Computer Science*, 26:327–333, 1983.

[278] Jiapeng Zhang. On the query complexity for showing dense model. Technical Report TR11-038, Electronic Colloquium on Computational Complexity, 2011.

[279] Marius Zimand. Simple extractors via constructions of cryptographic pseudo-random generators. *Theoretical Computer Science*, 411(10):1236–1250, 2010.