Applications of Unconditional Pseudorandomness in Complexity Theory

A dissertation presented

by

Alexander D. Healy

to

The School of Engineering and Applied Sciences

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

in the subject of

Computer Science

Harvard University

Cambridge, Massachusetts

May 2007

Thesis advisor
Michael O. Rabin

Author
Alexander D. Healy

Applications of Unconditional Pseudorandomness in Complexity Theory

Abstract

Pseudorandomness – that is, information that "appears random" even though it is generated using very little true randomness – is a fundamental notion in cryptography and complexity theory.

This thesis explores the applications of pseudorandomness within complexity theory, with a focus on pseudorandomness that can be constructed *unconditionally*, that is without relying on any unproven complexity assumptions. Such pseudorandomness only "fools" restricted classes of algorithms, and yet it can be applied to prove complexity results that concern very general models of computation. For instance, we show the following:

- *Randomness-Efficient Error Reduction for Parallel Algorithms:* Typically, to gain confidence in a randomized algorithm, one repeats the algorithm several times (with independent randomness) and takes the majority vote of the executions. While very effective, this is wasteful in terms of the number of random bits that are used. Randomness-efficient error reduction techniques are known for polynomial-time algorithms, but do not readily apply to *parallel* algorithms since the techniques seem inherently *sequential*. We achieve randomness-efficient error reduction for highly-parallel algorithms. Specifically, we can reduce the error of a parallel algorithm to any $\delta > 0$ while paying only $O(\log(1/\delta))$ additional random bits, thereby matching the results for polynomial-time.

- *Hardness Amplification within* **NP***:* A fundamental question in *average-case* complexity is whether $\mathbf{P} \neq \mathbf{NP}$ implies the existence of functions in **NP** that are hard on average (over randomly-chosen inputs). While the answer to this question seems far beyond the reach of current techniques, we show that powerful *hardness amplification* is indeed feasible within **NP**. In particular, we show that if **NP** has a mildly hard-on-average function f (i.e., any small circuit for computing f fails on at least a constant fraction of inputs), then **NP** has a function f' that is extremely hard on average (i.e., any small circuit for computing f' only succeeds with exponentially-small advantage over random guessing).

  Previous results only obtained functions f' that could not be computed with polynomial advantage over random guessing. Our stronger results are obtained by using *derandomization* and *nondeterminism* in constructing f'.

A common theme in our results is the computational efficiency of pseudorandom generators. Indeed, our results both rely upon, and enable us to construct pseudorandom generators that can be computed very efficiently (in terms of parallel complexity).

Contents

Citations to Previously Published Work

The majority of the research presented in this thesis has already been published in journals or conference proceedings.

Chapter 3 is based on the paper "Randomness-Efficient Sampling within $\mathbf{NC}^1$" [Hea06], which appeared in the proceedings of RANDOM 2006 and will appear in a *Computational Complexity* special issue on RANDOM 2006.

Chapter 4 is based on "Using Nondeterminism to Amplify Hardness", written jointly with Salil Vadhan and Emanuele Viola. A preliminary version of this work appeared at STOC 2004 [HVV04], and subsequently it was invited to appear in the SIAM Journal on Computing special issue on STOC 2004 [HVV06].

Chapter 5 is based on work with Emanuele Viola entitled "Constant-Depth Circuits for Arithmetic in Finite Fields of Characteristic Two" [HV06], which appeared in the proceedings of STACS 2006.

Acknowledgments

First and foremost, I should like to thank my advisor, Michael Rabin, for his encouragement, support and valuable advice no matter which direction my research took me. His inspiring presence, keen insight and disarming sense of humor make it a pleasure for all to learn from and interact with him.

I know Salil Vadhan not only as a committee member, but also as a teacher, mentor and colleague. I have learned a tremendous amount from him in each of these roles and it is a pleasure to thank him for all that he has taught me.

I am also grateful to Michael Mitzenmacher and Les Valiant for their roles as committee members, advisors, teachers and colleagues throughout my time at Harvard.

Oded Goldreich is a mentor to all students of the theory of computation, and I have been fortunate to interact with him on various occasions. I am particularly grateful for his encouragement and detailed feedback on the results of Chapter 3 as well as his enthusiasm for the results of Chapter 4.

Of course, my graduate school experience has been greatly enriched by the friends, collaborators and colleagues with whom I interact on a daily basis: Yan-Cheng Chang, Johnny Chen, Kai-Min Chung, Eleni Drinea, Vitaly Feldman, Danny Gutfreund, Shaili Jain, Adam Kirsch, Loizos Michael, Minh Nguyen, Shien Jin Ong, Alon Rosen and Emanuele Viola. In particular, working with Danny, Alon and Emanuele is a prime example of collaboration that skirts a wonderful boundary between research and friendship.

It has also been a pleasure to collaborate and interact with a variety of researchers beyond the halls of Harvard: Melissa Chase, Shafi Goldwasser, Tali Kaufman, Anna Lysyanskaya, Tal Malkin, Leonid Reyzin, Guy Rothblum, Eric Allender, Gudmund Skovbjerg Frandsen, Kristoffer Hansen, Ryan O'Donnell, Rocco Servedio, Richard Stanley, Luca Trevisan and Avi Wigderson.

Finally, I should like to thank my parents who – despite my best efforts to keep them blissfully ignorant of my research – insist on being continually supportive and unconditionally proud of whatever it is that I do.

Thank you to all.

The research presented in this thesis was supported by NSF grant CCR-0205423 and a Sandia Fellowship.

Chapter 1

Introduction

In everyday life, *randomness* is often nothing more than a source of frustration. Indeed, we could often do without the uncertainty of weather forecasts, the random fluctuation of financial markets and other unpredictable phenomena that seem only to make our daily existence more difficult.

The statistician, however, takes a different view of randomness. This is because she understands that randomness can be a powerful resource. For instance, she knows that she can learn a lot about a large population by examining only a small number of samples, *provided the samples are chosen randomly*. While counterintuitive at first, this use of randomization for sampling is well-known and widely-practiced. Moreover, the statistician loses no sleep over her decision to use randomness in this way, since she also knows that the randomization is essential: any *deterministic* sampling strategy is bound to be far less efficient (by requiring her to look at essentially every member of the population).

Over the last thirty years, computer scientists have discovered similarly powerful uses of randomness in algorithms, cryptography and computational complexity, and these discoveries have had a profound effect on the field. Indeed, randomized algorithms are now the standard for "efficient" computation and randomization is essential to many of the most fundamental notions in cryptography and computational complexity. Moreover, in many cases it is known that the use of randomness is unavoidable, just as in the sampling example described above. Nonetheless, there remain conspicuous applications of randomization for which it is not known that randomness is necessary, the most notable being efficient *randomized algorithms*.

In particular, although initial breakthroughs in randomized algorithms suggested that randomness can significantly improve the performance of algorithms, the common belief now is that this is in not the case for polynomial-time algorithms. Indeed, it is widely believed that any problem with a randomized polynomial-time algorithm also has a deterministic polynomial-time algorithm (although the running-time of the deterministic algorithm may be a much worse polynomial than that of the randomized algorithm).

1.1 Pseudorandomness

The conjecture that randomized algorithms can be *derandomized* is not simply wild speculation, but rather is based on the belief that it is possible to efficiently generate *pseudorandomness* – that is, bits that "appear" random, even though they are generated using very little true randomness. This is because, loosely speaking, one could replace the random bits that a randomized algorithm uses with pseudorandom bits (that were generated using very little randomness) to obtain a new algorithm that behaves in essentially the same way (because the pseudorandom bits "appear" random) but that uses very little randomness (or often none at all).

The hypothetical procedure that produces these pseudorandom bits is called a *pseudorandom generator* (or a *PRG* for short), and is the fundamental object of study in pseudorandomness. An informal definition is the following:

> A function $G : \{0,1\}^\ell \to \{0,1\}^n$ is an *ϵ-pseudorandom generator* if for any "efficient" decision algorithm A:
>
> $$\left|\Pr_s[A(G(s)) \text{ accepts}] - \Pr_x[A(x) \text{ accepts}]\right| \leq \epsilon,$$
>
> where $s \in \{0,1\}^\ell$ and $x \in \{0,1\}^n$ are chosen uniformly at random.

The input s of a PRG is referred to as the *seed* and the goal is typically to make ℓ – the *seed-length* – as short as possible. Other goals often include ensuring that G is efficiently computable and minimizing the error ϵ.

Of course, for the above definition to make sense, one must specify what is permissible as an "efficient" algorithm A. There is, however, no single correct notion of "efficiency" – this is often guided by the intended application. Indeed, it is by it is by considering different notions of efficiency that the field of pseudorandomness has developed a vast array of different PRGs, as we discuss below.

1.1.1 Conditional vs. Unconditional Pseudorandomness

One of the major open challenges in complexity theory is to *derandomize* randomized polynomial-time algorithms, i.e. to show that $\mathbf{P} = \mathbf{BPP}$, and a natural approach for doing so is to construct a pseudorandom generator that fools polynomial-time computations.[1] It is known, however, that such a pseudorandom generator would imply the existence of explicit functions with high circuit complexity, thereby resolving a notorious long-standing open problem in complexity theory.

Therefore, research in derandomization has focused on some more tractable tasks:

[1] More precisely, the PRG should fool *non-uniform* polynomial-time computations, or equivalently polynomial-size circuits; however, we shall not stress this point for the purposes of this informal discussion.

- **Conditional PRGs:** Constructing pseudorandom generators for polynomial-time computations *based on unproven complexity assumptions*.
- **Unconditional PRGs:** Constructing pseudorandom generators for models of computation that are weaker than (or incomparable to) polynomial-time *using no unproven complexity assumptions*.

Naturally, the goal of the first line of research is to use the weakest complexity assumptions possible and the goal of the second line of research is to build PRGs for the most powerful class of algorithms possible.

Both directions have enjoyed remarkable success. A major achievement in the first approach is the celebrated work of Imagliazzo and Wigderson [IW97], which shows how to construct PRGs for polynomial-time from explicit functions having exponential (worst-case) circuit complexity. Meanwhile, research in unconditional PRGs has resulted in PRGs for algorithms that run in small space/memory, constant-depth circuits as well as a wide variety of natural statistical tests.

The focus of this thesis is on the latter type – unconditional pseudorandom generators – and in particular on their applications to derandomization and other areas of computational complexity.

1.1.2 Applications of Unconditional PRGs

In light of the above discussion, one natural application of unconditional pseudorandom generators is in derandomizing randomized computations. While it is too much to hope that these generators will fool arbitrary polynomial-time algorithms, they can sometimes be shown to fool algorithms that have a particular structure. A good example of this is the case of *randomness-efficient error reduction*.

Algorithmic Derandomization and Error-Reduction

The typical strategy for reducing the error of a randomized algorithm is to repeat the algorithm (say k times) using independent randomness for each execution, and then to output the majority vote of the executions. Thus, if the original algorithm had error probability at most $1/3$ (or any constant less than $1/2$), the error probability of the new algorithm is, by a Chernoff bound, at most $2^{-\Omega(k)}$. However, this approach is quite wasteful in terms of random bits: it increases the number of random bits that are necessary by a factor of k.

A beautiful technique developed independently by Cohen & Wigderson [CW89] and Impagliazzo & Zuckerman [IZ89] achieves the same error-reduction by only paying $O(k)$ additional bits (as opposed to a multiplicative factor of k). The idea is simple yet very powerful: rather than choosing the randomness for each execution independently, choose them by taking a random walk on an *expander graph*. Since expander graphs are very sparse, the description of a random walk uses very few

random bits – indeed, if the graph has constant degree, then each of the k steps costs only $O(1)$ bits (to specify a random neighbor). Moreover, because expander graphs are very well-connected (despite being sparse) a random walk behaves in many ways like a walk on the complete graph (i.e., as if we used truly independent randomness for each execution). In particular, the error of the resulting algorithm can still be shown to be $2^{-\Omega(k)}$.

To implement this error-reduction efficiently, however, one needs very explicit constructions expander graphs that can be navigated in polynomial-time. While a highly non-trivial challenge of its own, a variety of such constructions are known [Mar73, GG81, LPS88, RVW02], and such constructions are essential to the results of [CW89, IZ89].

In Chapter 3, we address the problem error-reduction in complexity classes below polynomial-time, such as $\mathbf{NC}^1$ – that is, the class of functions computable by polynomial size formulae (or equivalently, computable in parallel time $O(\log n)$). Indeed, the approach of [CW89, IZ89] does not yield error-reduction for $\mathbf{NC}^1$ since there is no known family of expander graph that is constructible in $\mathbf{NC}^1$.

In spite of this, we construct a *sampler* that matches the parameters of random walks on expander graphs and is computable in $\mathbf{NC}^1$. In fact, our sampler is computable by constant-depth circuits that are even weaker than $\mathbf{NC}^1$, and thus we obtain randomness-efficient error reduction not only for $\mathbf{NC}^1$ but for a variety of classes below $\mathbf{NC}^1$ (all the way down to $\mathbf{AC}^0$, i.e. constant-depth circuits).

Samplers also have applications beyond error-reduction, and indeed we apply our sampler to obtain new derandomizations of constant-depth circuits, efficient *randomness extractors* and an optimally-efficient ϵ-biased generator, i.e. a generator that fool linear functions over $\mathbb{F}_2$.

Derandomized Hardness-Amplification

Remarkably, the applications of pseudorandom generators are not confined simply to questions of algorithmic derandomization. This is in part because randomization is not only a (seemingly) powerful algorithmic resource, but also a powerful resource in the analysis of algorithms and computation. Thus, we may sometimes use pseudorandomness in order to derandomize the *analysis* of some computational task.

The principal such application we shall consider is *hardness amplification*, i.e. the task of transforming mildly-hard computational problems into extremely hard problems. Specifically, we are interested in transforming problems that cannot be solved on more than a $1 - \delta$ fraction of inputs into problems that cannot be solved on more than $1/2 + \epsilon$ fraction of inputs for the smallest δ and ϵ possible. While this may seem counter-productive at first, such transformations are an important part of understanding the landscape of computational complexity. For example, it would be a major achievement to show that if $\mathbf{P} \neq \mathbf{NP}$ (i.e., polynomial-time algorithms cannot solve SAT on more than a $1 - 1/2^n$ fraction of n-variable Boolean formulae) then

there exists languages in **NP** that cannot be solved (by polynomial-time algorithms) on more than a 90% fraction of instances. Indeed, the security of most cryptographic protocols requires not only that $\mathbf{P} \neq \mathbf{NP}$, but moreover that there exist functions in **NP** that cannot be solved *on the average*, over randomly chosen inputs. It is therefore natural to ask whether whether the assumption that $\mathbf{P} \neq \mathbf{NP}$ implies the existence of **NP** problems that are hard on average – that is, does worst-case hardness of **NP** implies average-case hardness of NP? Along with proving that $\mathbf{P} \neq \mathbf{NP}$, establishing (or refuting) such a *worst-case/average-case equivalence for* **NP** is among the most important questions about the complexity of the class **NP**.

Although a variety of recent results suggest that proving such an equivalence is still beyond the reach of current complexity-theoretic techniques, in Chapter 4 we demonstrate that a powerful *hardness amplification* is indeed feasible within **NP**.

For instance, we show that if there is a random distribution of Boolean formulae on n variables for which no circuit can decide SAT with probability better than 99%, then there is in fact a random distribution of Boolean formulae on n variables for which no circuit can decide SAT with probability better than $1/2 + \epsilon(n)$ for a rapidly vanishing function $\epsilon(n)$. Prior to this work, O'Donnell [O'D04] had proven such a result for $\epsilon(n) \approx 1/n$. Our techniques improve this to $\epsilon(n) \approx 1/2^{\sqrt{n}}$ which is just short of the optimal value, $\epsilon(n) \approx 1/2^n$.

One of the main tools in our approach is a *pseudorandom generator* that fools a seemingly weak model of computation (i.e., small-space algorithms). Roughly speaking, we show that the outputs of this relatively weak pseudorandom generator still behave like random inputs for certain structured **NP** functions that arise naturally in O'Donnell's hardness amplification. Thus, a random (and very short) seed for the generator can encode a very large pseudorandom (and hence extremely difficult) instance of this **NP** problem. This has the effect of shrinking the input length of the function (since we need only specify the seed, and not the entire input), while preserving its hardness. Put another way, we dramatically increase the hardness (as measured as a function of the input length), as desired.

Since the pseudorandom generator we use is *unconditional*, this can all be achieved without employing any unproven complexity assumptions (beyond the initial hypothesis that NP has mildly average-case hard problems).

1.2 Structure of the Thesis

Chapter 2: Preliminaries This chapter reviews the basic definitions of ϵ-biased generators, k-wise independent generators, expander graphs and circuit classes.

Chapter 3: Randomness-Efficient Sampling within $\mathbf{NC}^1$ A fundamental pseudorandom object is the *expander graph*. Roughly speaking, expander graphs are

explicit graphs that share many characteristics with random graphs – in particular, they are very sparse while at the same time being very well-connected.

Expanders are used throughout theoretical computer science, and one powerful use of expanders is *randomness-efficient sampling*: that is, in many applications where one needs k uniform and independent samples from a universe of size N, one may actually substitute k (dependent) samples chosen according to a random walk on a constant-degree expander graph of size N.

On the surface, a random walk on an expander graph seems like an inherently *sequential* process. A natural question, therefore, is whether the wealth of expander-based techniques from the literature can be applied within *parallel* models of computation. We address this question by giving a highly-parallel implementation of a *sampler* that matches the parameters of expander walks.

We can then apply this sampler to obtain a variety of new derandomizations of highly-parallel algorithms. In addition, the analysis of our sampler involves proving new, stronger pseudorandomness properties of expander walks.

Chapter 4: Derandomized Hardness Amplification within NP In this chapter we show that if **NP** has a balanced function that is hard to compute on a non-negligible fraction of inputs, then **NP** contains a function that is hard to compute with more than exponentially small advantage over random guessing. This improves a results of O'Donnell [O'D04], which only produces functions that are hard to compute with advantage $1/\sqrt{n}$.

O'Donnell also proved that no construction of a certain general form could amplify much beyond the hardness achieved by his construction. We bypass this barrier by using both *derandomization* and *nondeterminism* in the construction of f'.

We also prove impossibility results demonstrating that both our use of nondeterminism and the hypothesis that f is balanced are necessary for "black-box" hardness amplification procedures.

Chapter 5: Constant-Depth Circuits for Finite Field Arithmetic Finite fields have a wide variety of applications in computer science, and in this chapter we study the complexity of arithmetic operations in finite fields. Specifically, we focus on finite fields of characteristic two; that is, finite fields $\mathbb{F}_{2^n}$ having 2^n elements, and the question we address is: To what extent can basic field operations (in particular, multiplication and exponentiation) in these fields be computed by constant-depth circuits? We demonstrate a variety of highly-parallel algorithms for arithmetic in these fields, many of which can also be proved to be optimal.

One of the original motivations for this work was to understand the complexity of certain pseudorandom generators, namely k-wise independent and ϵ-biased generators. Indeed, our results allow us to give constant-depth implementations of optimal constructions of k-wise independent and ϵ-biased generators, addressing questions

raised by Gutfreund and Viola [GV04].

Our results have other consequences as well: For instance, we also prove that uniform threshold circuits are equal to the class AE of functions computable by certain arithmetic expressions, thereby answering a question raised by Frandsen, Valence and Barrington [FVB94].

Chapter 2

Preliminaries

Notation: For a positive integer n, we denote the set $\{1, \ldots, n\}$ by $[n]$.

2.1 ϵ-Biased Sets and Generators

Small-biased spaces appear in several ways in this thesis. In Chapter 3, poly-size ϵ-biased sets are used to construct expander graphs on which our sampler construction is based (Lemma 3.2.1). In Chapters 3 and 5 we also address the task of building exponential-size ϵ-biased sets that are (very efficiently) *bitwise* computable (see the definition below, Corollary 3.1.7 and Theorem 5.2.13).

Definition 2.1.1. *For $a, b \in \mathbb{Z}_2^m$, let $\langle a, b \rangle_2$ denote the inner product of a and b modulo 2.*

A multi-set $S \subseteq \mathbb{Z}_2^m$ is ϵ-biased *if for all non-zero $y \in \mathbb{Z}_2^m$, $\Pr_{x \in S}[\langle x, y \rangle_2 = 1] \in [1/2 - \epsilon, 1/2 + \epsilon]$.*

An ϵ-biased generator *is a function $g : \{0,1\}^\ell \to \{0,1\}^m$ such that the multi-set $\{g(s) \mid s \in \{0,1\}^\ell\}$ is an ϵ-biased multi-set.*

A bitwise ϵ*-biased generator is a function $g : \{0,1\}^\ell \times [m] \to \{0,1\}$ such that the function $g'(s) = (g(s,1), g(s,2), \ldots, g(s,m))$ is an ϵ-biased generator.*

2.2 k-wise Independent Generators

Another important generator is the k-wise independent generator:

Definition 2.2.1. *For $z \in \{0,1\}^m$ and $I \subseteq [m]$, let $z|_I \in \{0,1\}^{|I|}$ denote the projection of z on the bits specified by I.*

A k-wise independent generator *is a function $g : \{0,1\}^\ell \to \{0,1\}^m$ such that for every $M : \{0,1\}^k \to \{0,1\}$ and $I \subseteq [m]$ such that $|I| = k$:*

$$\Pr_{y \in \{0,1\}^k}[M(y) = 1] = \Pr_{x \in \{0,1\}^s}[M(G(x)|_I) = 1].$$

2.3 Expander Graphs

Informally, expander graphs are sparse-yet-highly-connected graphs. While there are a variety of equivalent notions of graph expansion (see, e.g., [AS00, Gol99, HLW06]), it is most convenient for us to work with the following spectral definition.

Definition 2.3.1. *Let G be a regular directed graph*[1] *on N nodes with transition matrix P, and let $\mathbf{u} = (1/N, \ldots, 1/N) \in \mathbb{R}^N$ denote the uniform distribution on G. We say that G is a* λ-expander *if*

$$\max_{\substack{x \in \mathbb{R}^N \\ \langle x, \mathbf{u} \rangle = 0}} \frac{\|Px\|}{\|x\|} \leq \lambda.$$

When G is undirected, this definition is equivalent to the second-largest eigenvalue of P being at most λ in absolute value – see, e.g., [Mih89, Fil91].

We often abuse language and refer to an "λ-expander", when we really mean a "family of $\lambda(n)$-expanders of size $s(n)$" for some function $s(n)$. Also, when we simply refer to an "expander graph", without mention of λ, it is understood that we mean a family of λ-expanders for some constant $\lambda < 1$.

By a *random walk* on a d-regular graph G, we mean the following process: choose a random starting vertex $v_0 \in G$, and for $i = 1, \ldots, k$, let v_i be a uniformly random neighbor of v_{i-1} in G and output $v_1, \ldots, v_k$. Note that we are discarding the starting vertex v_0, although it is easy to see that the distribution is unchanged even if we keep v_0. We prefer this convention as it simplifies our notation and presentation. We also note that such a walk is described by a tuple $(v_0, s_1, \ldots, s_k) \in [|G|] \times [d] \times \cdots \times [d]$, and hence by a string of $\log |G| + O(k \log d)$ bits.

2.4 Circuits

In addition to standard Boolean circuits (over the basis AND, OR, NOT), this thesis considers a variety of restricted circuit classes. Below we recall the basic definitions of these classes as well as the relationships between them.

Recall that $\mathbf{NC}^1$ denotes the class of functions $f : \{0,1\}^n \to \{0,1\}$ computable by circuits of size $\text{poly}(n)$ and depth $O(\log n)$ over the basis $\{\wedge, \vee, \neg\}$ (where all the gates have fan-in 2).

We also consider three classes of unbounded fan-in *constant-depth circuits* of polynomial size:

- $\mathbf{AC}^0$: The class of circuits having AND and OR gates of unbounded fan-in, NOT gates and depth $O(1)$.

[1] A directed graph is *d-regular* if the in-degree and out-degree of every node is equal to some fixed d, and a directed graph is *regular* if it is d-regular for some d.

- $\mathbf{AC}^0[\oplus]$: The class of circuits having AND, OR and XOR gates of unbounded fan-in, NOT gates and depth $O(1)$.
- $\mathbf{TC}^0$: The class of circuits having AND, OR and MAJORITY gates of unbounded fan-in, NOT gates and depth $O(1)$.

We will routinely abuse language and refer to *functions* f that can be computed by $\mathbf{AC}^0$ (respectively $\mathbf{AC}^0[\oplus]$ and $\mathbf{TC}^0$) circuits (of a certain size s); by this we simply mean that, given x and $i \leq |f(x)|$, computing the i-th bit of $f(x)$ can be performed by $\mathbf{AC}^0$ (resp. $\mathbf{AC}^0[\oplus]$ and $\mathbf{TC}^0$) circuits (of size s).

Unless explicitly stated otherwise, all circuits are of polynomial size and *uniform*. When referring to the *uniformity* of a family of circuits, we mean the complexity of the *uniform* algorithm that "constructs" the n-th circuit, given input n. When working with constant-depth circuits, the issue of uniformity can be a delicate one. Nonetheless, there is a single notion of uniformity that is generally accepted to be the most appropriate for these classes, namely *Dlogtime*-uniformity. A detailed description of *Dlogtime*-uniformity can be found in [BIS90] (see also [Vol99]); below we give a more informal description.

A family of circuits $\{C_n\}_{n=1}^{\infty}$ of size $s(n)$ is said to be *Dlogtime*-uniform if there exists a random-access Turing machine that:

- On input n and $i \leq s$ determines in time $O(\log n + \log i)$ the type of gate i (e.g., AND, OR, NOT, XOR, MAJ) in the circuit C_n.
- On input n and $i, j \leq s$ decides in time $O(\log n + \log i)$ whether the output of gate i is joined to the input of gate j in the circuit C_n.

This restrictive notion of uniformity is more than adequate to ensure that the class of functions computed by uniform poly(n)-size $\mathbf{TC}^0$ circuits is contained in logarithmic space.

When referring to non-uniform circuits, we always indicate this explicitly using *slash* notation: for example, $\mathbf{AC}^0/O(n)$ is the class of boolean functions $f : \{0,1\}^n \to \{0,1\}$ such that there exists a *Dlogtime*-uniform $\mathbf{AC}^0$ circuit family $C_n : \{0,1\}^n \times \{0,1\}^{O(n)} \to \{0,1\}$ for which there is a single advice string a_n of length $O(n)$ such that $C_n(x, a_n) = f(x)$ for all $x \in \{0,1\}^n$.

The probabilistic classes $\mathbf{BP} \cdot \mathbf{AC}^0, \mathbf{BP} \cdot \mathbf{AC}^0[\oplus], \mathbf{BP} \cdot \mathbf{TC}^0$ and $\mathbf{BP} \cdot \mathbf{NC}^1$ are all defined in the natural way: the circuit takes two inputs, one of n bits and one of $r(n)$ random bits for some polynomially-bounded function $r(n)$, and for any fixed input $x \in \{0,1\}^n$, the circuit should correctly compute the function with probability at least $2/3$ over the $r(n)$ random bits.

Recall that $\mathbf{AC}^0 \subsetneq \mathbf{AC}^0[\oplus] \subsetneq \mathbf{TC}^0 \subseteq \mathbf{NC}^1 \subseteq \mathbf{L}$, where the last inclusion holds under logspace uniformity and the separations follow from works by Furst et al. [FSS84] and Razborov [Raz87], respectively (and hold even for non-uniform circuits). Despite these lower-bounds, $\mathbf{AC}^0$ can compute the *approximate* majority of

n bits [Ajt93] – in particular, for any constant $\epsilon > 0$, there exists a family of $\mathbf{AC}^0$ circuits that correctly computes the majority function for all inputs with at most a $n/2 - \epsilon n$ ones and for all inputs with at least $n/2 + \epsilon n$ ones.

Scaling Down $\mathbf{TC}^0$: It is well-known that uniform poly(n)-size $\mathbf{AC}^0$ circuits can compute the MAJORITY function on polylog(n) bits. In particular, this means that any problem that is solved by uniform poly(n)-size $\mathbf{TC}^0$ circuits on inputs of length n can, on inputs of length polylog(n), be solved by uniform poly(n)-size $\mathbf{AC}^0$ circuits that simulate the MAJORITY gates of the $\mathbf{TC}^0$ circuits. We will use these facts frequently and will often simply refer to "scaling down" a given family of uniform $\mathbf{TC}^0$ circuits to obtain the appropriate uniform $\mathbf{AC}^0$ circuits. For example, since both iterated integer multiplication of n n-bit numbers and division of n-bit numbers are in uniform poly(n)-size $\mathbf{TC}^0$ [HAB02], we have the following lemma about performing these operations by uniform $\mathbf{AC}^0$ circuits.

Lemma 2.4.1 ([HAB02], Theorem 5.1)**.** *For every constant $c > 1$, the following can be computed by Dlogtime-uniform $\mathbf{AC}^0$ circuits of size* poly(n)*:*

- *Given integers $a_1, a_2, \ldots, a_{\log^c n}$, each of length at most $\log^c n$ bits, compute $\prod_{i \leq \log^c n} a_i$.*
- *Given integers a, b, each of length at most $\log^c n$ bits, compute $\lfloor a/b \rfloor$.*

Additional background on constant-depth circuits can be found in, e.g., [Hås87, Vol99].

Chapter 3

Randomness-Efficient Sampling within NC^1

3.1 Introduction

Over the last three decades, *expander graphs* have found a wide variety of applications in Theoretical Computer Science. They have been used in designing novel algorithms (e.g., [AKS83], [Rei05]), in the study of circuit complexity (e.g., [Val77], [IW97]) and to derandomize probabilistic computation (e.g., [CW89], [IZ89]), just to name a few notable examples from this vast literature.

Many of these applications involve a *random walk* on an expander. That is, we choose a random starting node v in an expander graph G, take a k-step random walk and use the k nodes visited by this walk in some way – often as a substitute for k independently-chosen nodes. Despite its simplicity, this processes has some remarkable sampling properties which we discuss shortly. For the moment, we address the computational efficiency of expanders walks.

3.1.1 The Complexity of Walks on Expander Graphs

In applications, one often requires an expander graph that is exponentially large, say on 2^n nodes. In this case, a random walk on the graph is performed using a *strongly explicit* representation – that is, a representation in which each node is identified with an n-bit string and it is possible to efficiently (e.g., in time $\mathrm{poly}(n)$) find all the neighbors of a given node $v \in G$. Several beautiful constructions [Mar73, GG81, LPS88, RVW02] are known of such explicit constant-degree expander graphs of exponential size.

At first glance, the act of taking a random walk on an expander graph seems like an inherently *sequential* process – indeed, each step of the walk seems to rely on the previous step in an essential way. A natural question, therefore, is whether the wealth of expander-based techniques from the literature can be applied within highly-*parallel*

models of computation, such as log-depth circuits (i.e., $\mathbf{NC}^1$) or even constant-depth circuits.

The main technical contribution of this chapter is a *sampler* that is just as good as a random walk on an expander graphs (in a sense that is made precise in the next section), but which is computable in parallel time $O(\log n)$, i.e. computable by uniform $\mathbf{NC}^1$ circuits. In fact, our sampler is computable by uniform constant-depth circuits with parity gates (i.e. $\mathbf{AC}^0[\oplus]$), a class that is strictly weaker than $\mathbf{NC}^1$ as it cannot even compute the majority of n bits [Raz87].

3.1.2 The Properties of Walks on Expander Graphs

We now discuss the important sampling properties of random walks on expander graphs in order to better understand what properties we require of our sampler. A more formal definition of expander graphs is given in Section 2.3, but for the moment the reader may simply think of an expander graph as a constant-degree undirected graph, G, that is "highly-connected".

A fundamental sampling property of expander walks is the *hitting* property, first shown by Ajtai, Komlós and Szemerédi [AKS87]:

The Hitting Property: For any subset S of half the nodes of G, the probability that a k-step random walk never visits a node in S is at most $2^{-\Omega(k)}$.

This hitting property is quite useful (e.g., to reduce the error of **RP** algorithms), but some applications require an even stronger property, which we call the *strong hitting* property:

The Strong Hitting Property: For any sequence of subsets $S_1, \ldots, S_k$, each consisting of half the nodes of G, the probability that a k-step random walk does not pass through S_i on the i-th step for any $i \in \{1, \ldots, k\}$ is at most $2^{-\Omega(k)}$.

It turns out that this strong hitting property is what is necessary for the randomness-efficient error reduction techniques of [CW89] and [IZ89], the amplification technique of [GIL+90] and for the derandomized XOR Lemma of [IW97].

Clearly, the strong hitting property is a generalization of the (non-strong) hitting property. Another natural generalization of the hitting property is the following, first proved by Gillman [Gil94]:

The Chernoff Bound for Expander Walks: For any subset S of half the nodes of G, the fraction of time that a k-step random walk spends in S is $1/2 \pm \epsilon$ with probability $1 - 2^{-\Omega(\epsilon^2 k)}$.

This Chernoff Bound is quite powerful and has applications to constructing randomness extractors (see [Zuc97]) and to Markov-Chain Monte Carlo algorithms (see [Gil94]). Although, it is not clear that it subsumes the *strong* hitting property. The following property, however, generalizes both the strong hitting property *and* the Chernoff bound:

The Strong Chernoff Bound for Expander Walks: Fix a sequence of subsets $S_1, \ldots, S_k$, each consisting of half the nodes of G. Then for a k-step random walk on

G, the fraction of indices i such that the i-th step of the walk lands in S_i is $1/2 \pm \epsilon$ with probability $1 - 2^{-\Omega(\epsilon^2 k)}$.

Thus, the Strong Chernoff Bound for Expander Walks subsumes all the aforementioned sampling properties, and it seems to represent the essential abstract property of random walks on expanders that is necessary for most natural applications. This bound has only been proved recently – it follows from the work of Wigderson and Xiao [WX05].[1]

In Section 3.4, we give a direct and elementary proof of the Strong Chernoff Bound for Expander Walks (Theorem 3.1.1). In contrast to most of the proofs in this area, our proof uses only basic linear algebra and, in particular, does not require any perturbation theory or complex analysis in order to obtain a bound that matches the parameters of Gillman's (non-strong) Chernoff bound.[2] Since this bound is important to our analysis, we give a formal statement before describing our results in more detail. (In the following, a λ-*expander* is a regular graph whose normalized second-largest eigenvalue (in absolute value) is at most λ – see Section 2.3 for a precise definition.)

Theorem 3.1.1 (Implicit, up to constants, in [WX05])**.** *Let G be a regular λ-expander on V and fix a sequence of functions $f_i : V \to [0,1]$ each with mean $\mu_i = \mathrm{E}_v[f_i(v)]$. If we consider a random walk $v_1, \ldots, v_k$ on G, then for all $\epsilon > 0$,*

$$\Pr\left[\left|\sum_{i=1}^{k} f_i(v_i) - \sum_{i=1}^{k} \mu_i\right| \geq \epsilon k\right] \leq 2e^{-\frac{\epsilon^2(1-\lambda)k}{4}}.$$

In particular, by taking the functions f_i to be the characteristic functions of the sets S_i we obtain the Strong Chernoff Bound for Expander Walks (informally) stated above.

We also give a multiplicative form of the Chernoff bound (Corollary 3.4.5) that is sharper than Theorem 3.1.1 when the sets we are sampling is small (i.e., when the μ_i are small in the notation of Theorem 3.1.1). While Kahale [Kah97] has also improved Gillman's Chernoff bound in this setting, his techniques only address the case of sampling a single set; i.e., they give a non-strong Chernoff bound. As a corollary to the proof of Theorem 3.1.1, we obtain a strong Chernoff bound that improves upon Theorem 3.1.1 when the μ_i and λ are small (see Corollaries 3.4.5 and 3.4.6).

[1] Although a subsequent manuscript of Wigderson and Xiao [WX06] points out an error in [WX05], this only affects the case of sampling d-dimensional matrices for $d \geq 2$. Their proof remains valid for the case of sampling 1-dimensional matrices, which is all that is needed for the Strong Chernoff Bound stated here.

[2] [WX05] also gives a proof of a (strong) Chernoff bound using no perturbation theory but this bound does not match Gillman's. In particular, Theorem A.1 of [WX05] has a cubic dependence on the spectral gap $1 - \lambda$ in the exponent, as opposed to the (optimal) linear dependence; moreover, even when the spectral gap $1 - \lambda$ is constant, the dependence on ϵ is (slightly) worse than quadratic.

3.1.3 Our Sampler

Our main result is the construction of a *sampler* that is computable by $\mathbf{AC}^0[\oplus]$ circuits and possesses all the "sampling properties" of a random walk on a constant-degree expander graphs of size 2^n. To make this notion precise, we recall the following definition (essentially from [Zuc97]):

Definition 3.1.2. *A function* $\Gamma : \{0,1\}^m \to (\{0,1\}^n)^k$ *is said to be a* strong (γ, ϵ)-averaging[3] sampler *if: for any sequence of functions* $f_i : \{0,1\}^n \to [0,1]$ *each with mean* $\mu_i = \mathrm{E}_x[f_i(x)]$*,*

$$\Pr_s \left[\left| \sum_{i=1}^{k} f_i(\Gamma(s)_i) - \sum_{i=1}^{k} \mu_i \right| \leq \epsilon k \right] \geq 1 - \gamma.$$

We call m *the* seed-length *of the sampler, and we call* k *the* sample complexity *of the sampler.*

It is not hard to check that Theorem 3.1.1 implies that a random walk on a constant-degree expander (where λ is a constant less than 1) of size 2^n is a strong averaging sampler with seed-length $m = n + O(\log(1/\gamma)/\epsilon^2)$ and sample complexity $k = O(\log(1/\gamma)/\epsilon^2)$. Moreover, this sample complexity is known to be optimal up to constant factors, and when $\epsilon = \Omega(1)$ the seed-length is also optimal up to constant factors [CEG95]. Our main theorem is that uniform $\mathbf{AC}^0[\oplus]$ can compute a sampler that is just as good:

Theorem 3.1.3. *There exists a strong* (γ, ϵ)*-averaging sampler* $\Gamma : \{0,1\}^m \to (\{0,1\}^n)^k$ *with seed-length* $m = n + O(\log(1/\gamma)/\epsilon^2)$ *and sample complexity* $k = O(\log(1/\gamma)/\epsilon^2)$ *such that* Γ *is computable by* uniform $\mathbf{AC}^0[\oplus]$ *circuits of size* $\mathrm{poly}(n, 1/\epsilon, \log(1/\gamma))$*.*

On the one hand, Theorem 3.1.3 is superior to a random walk of length k on a constant-degree expander of size 2^n in the very low computational complexity of Γ; indeed, we do not know of any constant-degree expander walks computable in such low complexity. On the other hand, the sampler of Theorem 3.1.3 is potentially a weaker object than an expander walk: there may exist applications of expander walks in which one cannot simply substitute an arbitrary sampler. We note, however, that many applications of expander walks rely only on the fact that an expander walk is a good sampler; thus, when computational complexity is of the essence, we may employ our sampler in lieu of the expander walk.

As discussed in Section 3.2.1, the proof of Theorem 3.1.3 relies upon the *zig-zag graph product* of [RVW02] to build a sampler in $\mathbf{AC}^0[\oplus]$. In Section 3.2.3, we also mention an alternate construction of a sampler in $\mathbf{AC}^0[\oplus]$ that is inspired by the paradigm of sampler *composition* [BGG93, Gol97].

[3] [Zuc97] uses the term "oblivious sampler". We follow [Gol97] and use the more-accurate "averaging sampler".

Gutfreund and Viola have shown [GV04] that walks on the Margulis/Gabber-Galil expander graph [Mar73, GG81] with 2^n nodes are computable in space $O(\log n)$ (and therefore that logspace has strong samplers that match the parameters of Theorem 3.1.3). To the best of our knowledge, ours is the first work that implies the existence of such strong samplers within the class $\mathbf{NC}^1$ of log-depth circuits; in fact, our construction is in the strictly-weaker class $\mathbf{AC}^0[\oplus] \subsetneq \mathbf{TC}^0 \subseteq \mathbf{NC}^1 \subseteq \mathbf{L}$.

Since expander walks are a powerful and widely-applicable tool it is not surprising that our sampler construction should have a variety of applications. Indeed, we apply our construction to obtain the new results described in the remainder of this section.

Randomness-Efficient Error Reduction within $\mathbf{NC}^1$ One important application of random walks on expander graphs is in reducing the error of probabilistic algorithms. Such error reduction was achieved for **BPP** by Cohen and Wigderson [CW89] and Impagliazzo and Zuckerman [IZ89]. Bar-Yosef, Goldreich and Wigderson [BYGW99] show how to achieve modest-but-optimal error reduction for probabilistic logspace (i.e., the class **BPL**); this is accomplished by a careful implementation of *short* random walks the Margulis/Gabber-Galil expander that can be computed with *one-way* access to the random bits describing the walk. In contrast, Gutfreund and Viola [GV04] show how to compute *long* random walks on the Margulis/Gabber-Galil expander when given *two-way* access to the random bits describing the walk – this implies randomness-efficient error reduction for the class $\mathbf{BP} \cdot \mathbf{L}$.[4] As an application of our sampler construction, we obtain analogous error-reduction for a variety of classes below logspace (see Section 2.4 for the definitions of $\mathbf{BP} \cdot \mathbf{NC}^1$, $\mathbf{BP} \cdot \mathbf{TC}^0$ and $\mathbf{BP} \cdot \mathbf{AC}^0[\oplus]$):

Corollary 3.1.4. *Let $f : \{0,1\}^n \to \{0,1\}$ be a function computable by polynomial-size* uniform $\mathbf{BP} \cdot \mathbf{AC}^0[\oplus]$ *(respectively, $\mathbf{BP} \cdot \mathbf{TC}^0$ or $\mathbf{BP} \cdot \mathbf{NC}^1$) circuits with error at most $1/3$ using $r = r(n)$ random bits. Then for any $\delta = \delta(n) > 1/2^{O(\mathrm{poly}(n))}$, f has polynomial-size* uniform $\mathbf{BP} \cdot \mathbf{AC}^0[\oplus]$ *(respectively, $\mathbf{BP} \cdot \mathbf{TC}^0$ or $\mathbf{BP} \cdot \mathbf{NC}^1$) circuits with error at most δ using $r + O(\log(1/\delta))$ random bits.*

Combining our sampler with Nisan's unconditional pseudorandom generator for constant-depth circuits [Nis91], we obtain an even stronger result for $\mathbf{BP} \cdot \mathbf{AC}^0$ (see Section 2.4 for the definition of $\mathbf{BP} \cdot \mathbf{AC}^0$):

Corollary 3.1.5. *Let $f : \{0,1\}^n \to \{0,1\}$ be a function computable by polynomial-size* uniform $\mathbf{BP} \cdot \mathbf{AC}^0$ *circuits with error at most $1/3$ using $r = r(n)$ random bits.*

[4] $\mathbf{BP} \cdot \mathbf{L}$ refers to probabilistic logspace computation that allows for *two-way* access to the random bits, whereas the result of Bar-Yosef et al. concerns the standard model of probabilistic logspace computation (i.e. **BPL**), which only allows *one-way* access to the random bits. See the survey of Saks [Sak96] for a discussion of the subtleties surrounding different notions of probabilistic space-bounded computation.

Then for any $\delta = \delta(n) > 1/2^{O(\mathrm{poly}(n))}$, *$f$ has polynomial-size* uniform $\mathbf{BP} \cdot \mathbf{AC}^0$ *circuits with error at most* δ *using* $\min\{r, \mathrm{polylog}(n)\} + O(\log(1/\delta))$ *random bits.*

Derandomization with Linear Advice Recently, Fortnow and Klivans [FK06] have proved that $\mathbf{RL} \subseteq \mathbf{L}/O(n)$ – that is, one can derandomize probabilistic logspace computation at the cost of only a linear amount of non-uniform advice. Their approach is based on a clever combination of Nisan's pseudorandom generator for space-bounded computation [Nis92] and the logspace expander walks of Gutfreund and Viola [GV04]. Our techniques yield an analogous result for uniform probabilistic constant-depth circuits:

Corollary 3.1.6. uniform $\mathbf{BP} \cdot \mathbf{AC}^0 \subseteq$ uniform $\mathbf{AC}^0/O(n)$.

Ajtai & Ben-Or [ABO84] show that nonuniform $\mathbf{BP} \cdot \mathbf{AC}^0 =$ nonuniform $\mathbf{AC}^0$; however, even for derandomizing uniform $\mathbf{BP} \cdot \mathbf{AC}^0$ [Ajt93] their technique seems to require an arbitrary polynomial amount of non-uniform advice. Corollary 3.1.6 quantifies the amount of nonuniformity that is necessary to derandomize a probabilistic $\mathbf{AC}^0$ circuit, and therefore can be viewed as a refinement of their result. The same approach, together with a new pseudorandom generator of Viola [Vio05b], yields similar results for circuits with a bounded number of parity or majority gates – see Corollary 3.3.1 in Section 3.3.2.

An Optimal Bitwise ϵ-Biased Generator in $\mathbf{AC}^0[\oplus]$ Gutfreund and Viola [GV04] study the complexity of constructing *bitwise*[5] ϵ-biased generators (see Definition 2.1.1). They give a construction in uniform $\mathbf{AC}^0[\oplus]$ whose seed-length is optimal for $\epsilon = \Omega(1/\operatorname{poly}\log\log(m))$ (where m is the number of output bits) and sub-optimal for smaller ϵ. Healy and Viola [HV06] give an optimal construction in uniform $\mathbf{TC}^0$ and a sub-optimal construction in uniform $\mathbf{AC}^0[\oplus]$ whose parameters are incomparable to those of [GV04]. In this chapter, we resolve this question entirely – using our sampler (and [NN90, GV04]), we construct an *optimal* bitwise ϵ-biased generator in uniform $\mathbf{AC}^0[\oplus]$:

Corollary 3.1.7. *For every* $\epsilon > 0$ *and* m, *there is an* ϵ*-biased generator* $G : \{0,1\}^n \to \{0,1\}^m$ *with* $n = O(\log m + \log(1/\epsilon))$ *such that uniform* $\mathbf{AC}^0[\oplus]$ *circuits of size* $\mathrm{poly}(n, \log m) = \mathrm{poly}(n)$ *can compute* $G(s)_i$ *given* $(s, i) \in \{0,1\}^n \times [m]$.

It is known that ϵ-biased generators require seed length at least $\Omega(\log m + \log(1/\epsilon))$ [AGHP92], and it can be shown that bitwise ϵ-biased generators achieving the parameters of Corollary 3.1.7 require $\mathbf{AC}^0$ circuits of exponential size [GV04, MNT90]. Therefore, the construction of Corollary 3.1.7 is tight both in terms of seed-length and computational complexity.

[5] [GV04] calls such generators *explicitly computable*.

3.1.4 Organization

The remainder of this chapter is organized as follows. In Section 3.2 we prove Theorem 3.1.3 and also describe an alternate sampler construction. The proofs of the applications described above can be found in Section 3.3. Section 3.4 is devoted to proving Theorem 3.1.1 as well as an alternate Chernoff bound, and some open questions are discussed in Section 3.5.

3.2 The Sampler Construction

In this section, we describe our sampler construction and prove Theorem 3.1.3. Recall that our goal is to construct a sampler $\Gamma : \{0,1\}^m \to (\{0,1\}^n)^k$ that matches the parameters of random walks on expander graphs. Naturally, one way to achieve this would be to exhibit a family of constant-degree expander graphs on 2^n nodes and show that walks of length k on these expanders can be computed in $\mathbf{AC}^0[\oplus]$ of size $\mathrm{poly}(n,k)$. Unfortunately, we do not know of any such family of expanders.[6] Instead, we begin with a family of expander graphs of degree $\mathrm{poly}(n)$ where walks are computable in $\mathbf{AC}^0[\oplus]$ – note that a walk of length k on such a graph is described by a seed of length $n + O(k \cdot \log n)$ – and then we *derandomize* the walk on this graph to achieve the optimal seed length $n + O(k)$. This derandomization uses random walks on a smaller expander graph, and its analysis is based on the *zig-zag graph product* of [RVW02].

In the sequel, we focus on the case where $k = \Omega(\log n)$ since by [GV04] it is known that $\mathbf{AC}^0$ circuits can compute walks of length $\log n$ on the Margulis/Gabber-Galil graph of size 2^n.

3.2.1 The Construction

Our first graph, G, is a Cayley graph on the group $\mathbb{Z}_2^n$. Specifically, we construct a $1/n$-biased set $S \subset \mathbb{Z}_2^n$ of size $\mathrm{poly}(n)$ (see Definition 2.1.1) and let $\{v, w\}$ be an edge if and only if $v \oplus w \in S$. The following well-known fact guarantees that G has second-largest eigenvalue at most $2/n$ (e.g., see [AR94]).

Lemma 3.2.1. *A Cayley graph on $\mathbb{Z}_2^n$ with generators $S \subset \mathbb{Z}_2^n$ is a 2ϵ-expander if and only if S is ϵ-biased.*

Before continuing, let us see how walks on G can be computed in $\mathbf{AC}^0[\oplus]$. First, we note that a $1/n$-biased set S of size $\mathrm{poly}(n)$ can be constructed in $\mathbf{AC}^0$. For

[6] Indeed, Gutfreund and Viola [GV04] observe that $\mathbf{AC}^0[\oplus]$ cannot compute walks on the Margulis/Gabber-Galil expander, and the same argument can easily be extended to rule out the possibility of $\mathbf{AC}^0[\oplus]$ circuits that compute walks on a variety of other natural expander graphs. Nevertheless, it does seems plausible that $\mathbf{AC}^0[\oplus]$ circuits could compute walks on some constant-degree expander family – see the discussion in Section 3.5.

instance, we may use the "Powering Construction" of an ϵ-biased generator from [AGHP92] together with the results on field arithmetic of [HV06].[7] (Note that for a non-uniform construction, we could simply hard-wire such an ϵ-biased set into the circuit.)

Next, we observe that the neighbors of a node $v \in \{0,1\}^n$ are the nodes $\{v \oplus g(s) \mid s \in \{0,1\}^\ell\}$, where $g : \{0,1\}^\ell \to \{0,1\}^n$ is the ϵ-biased generator defining the ϵ-biased set S and $\ell = O(\log n)$. Thus, a random walk starting at v is obtained by letting $v_0 = v$ and $v_i = v_{i-1} \oplus g(s_i)$ for randomly chosen seeds $s_i \in \{0,1\}^\ell$, and in particular $v_i = v_0 + \bigoplus_{j=0}^{i} g(s_j)$.

Hence, given the description a walk $(v, s_1, \ldots, s_k) \in \{0,1\}^n \times \{0,1\}^\ell \times \cdots \times \{0,1\}^\ell$, to determine the i-th vertex visited by the walk, the circuit need only compute from each seed s_j (in parallel) the appropriate vector $g(s_j) \in S$ and then compute the sum $v + \bigoplus_{j=1}^{i} g(s_j)$. This is clearly computable by $\mathbf{AC}^0[\oplus]$ circuits of size $\mathrm{poly}(n,k)$. In fact, the parity gates only appear at the outputs and each parity has fan-in at most $k+1$.

Now we turn to the problem of producing a *pseudorandom* sequence of steps s_j, with the goal of reducing the seed length of a walk on G, while at the same time preserving the sampling properties of such walks. Our approach is motivated by the zig-zag product of Reingold, Vadhan and Wigderson [RVW02]. Roughly speaking, one may interpret their results as saying the following: to derandomize a walk on a graph G of degree d, it suffices to choose the steps in G according to a random walk on a constant-degree expander graph H of size d. (For technical reasons, their result requires the graph H to be the *square* of an expander graph, but we ignore this for the moment.) Specifically, to take a pseudorandom k-step walk in G:

1. Choose a random starting vertex $v_0 \in G$.
2. Choose a random $w_0 \in H$ and take a random walk of length k on H, visiting nodes $w_1, \ldots, w_k$.
3. View $w_1, \ldots, w_k$ as indices in $[d]$ (recalling that $|H| = d$).
4. Use $w_1, \ldots, w_k$ as the steps of a walk (starting at v_0) in G.
5. Output the nodes $v_1, \ldots, v_k \in G$ visited by the walk from Step 4.

[7] Specifically, let $m = \log n$ (assuming that $\log n$ is an integer for simplicity) and consider the finite field $\mathbb{F}_{2^{2m}}$ with 2^{2m} elements (viewed as the ring of polynomials over $\mathbb{F}_2$ modulo an irreducible polynomial of degree $2m$). The generator outputs $2^{4m} = n^4$ vectors $v_{\alpha,\beta}$ of dimension $2^m = n$, indexed by pairs of elements $\alpha, \beta \in \mathbb{F}_{2^{2m}}$, where the i-th bit of $v_{\alpha,\beta}$ is given by $\langle \alpha^i, \beta \rangle_2$. It is shown in [AGHP92] that such a generator has bias less than $2^m/2^{2m} = 1/n$, and it is shown in [HV06] that all the necessary field arithmetic can be carried out in uniform $\mathbf{AC}^0$ of size $\mathrm{poly}(n)$ for this range of parameters.

Note that the seed length of such a sampler is $|v_0|+(|w_0|+O(k)) = n+\log|H|+O(k) = n+O(k)$ (since we assume $k = \Omega(\log n)$), as desired. Moreover, one can show (using the results of [RVW02]) that the forgoing construction is a strong averaging sampler. What is not clear, however, is how to compute this sampler in $\mathbf{AC}^0[\oplus]$. The reason is that it requires a long walk on the graph H, and while H is small (only $\mathrm{poly}(n)$ nodes) compared to G (which has 2^n nodes), we do not know how to take such a long walk (on any constant-degree expander family) in $\mathbf{AC}^0[\oplus]$, or even in $\mathbf{NC}^1$ for that matter.

In order to circumvent this obstacle, we derandomize the walk on G by using many short walks on H, rather than a single long walk.

Construction 1.

1. *Choose a random starting vertex* $v_0 \in G$.
2. *Take* $k/\log n$ *independent random walks each of length* $\log n$ *in* H*, where the* i*-th walk visits* $w_1^{(i)}, \ldots, w_{\log n}^{(i)} \in H$.
3. *View* $w_1^{(1)}, \ldots, w_{\log n}^{(1)}, w_1^{(2)}, \ldots, w_{\log n}^{(2)}, \ldots, w_1^{(k/\log n)}, \ldots, w_{\log n}^{(k/\log n)}$ *as indices in* $[d]$.
4. *Use* $w_1^{(1)}, \ldots, w_{\log n}^{(1)}, \ldots, w_1^{(k/\log n)}, \ldots, w_{\log n}^{(k/\log n)}$ *as the steps of a walk (starting at* v_0*) in* G.
5. *Output the nodes* $v_1, \ldots, v_k \in G$ *visited by the walk from Step* 4.

This sampler has seed length $|v_0| + (k/\log n) \cdot (\log|H| + O(\log n)) = n + O(k)$ (again, since we assume that $k = \Omega(\log n)$). Furthermore, we show below that this construction is a strong averaging sampler, achieving the same parameters as a random walk on a constant-degree expander graph. Before proving this, however, we observe that this walk is computable in $\mathbf{AC}^0[\oplus]$. Indeed, it is known how to compute walks of length $O(\log n)$ on poly-sized explicit expanders of constant degree in $\mathbf{AC}^0$ [Ajt93, GV04],[8] and thus each of the five steps above is computable in constant depth.

3.2.2 The Analysis

We now show that Construction 1 is a strong averaging sampler. In particular, Theorem 3.1.3 is a consequence of the following lemma:

Lemma 3.2.2. *Let* $H = \tilde{H}^2$ *where* $\tilde{H}$ *is a constant-degree expander graph on* $\mathrm{poly}(n)$ *nodes. Then Construction 1 is a strong* (γ, ϵ)*-averaging sampler with seed length* $n + O(\log(1/\gamma)/\epsilon^2)$ *and sample complexity* $O(\log(1/\gamma)/\epsilon^2)$.

Proof. Our proof relies on the zig-zag product of [RVW02], so we briefly recall that construction.

[8] As with the $1/n$-biased set S above, the non-trivial issue here is the uniformity of the circuits; if we only wish to give a nonuniform construction we could simply hard-wire all the possible walks of length $\log n$ into the circuit.

Zig-Zag Product Let G be a regular graph of degree d on vertices V_G whose edges are labeled with the names $1, \ldots, d$ in such a way that no two incident edges share the same label.[9] (Note that under such a labeling, if w is the "i-th neighbor of v", then v is the "i-th neighbor of w" – the graph G, defined above, clearly has this property, as it is a Cayley graph on a group of characteristic 2.) Then if g is a regular graph on vertices V_g where $|V_g| = d$, we may form the *zig-zag product* graph $G \textcircled{z} g$ where:

- $G \textcircled{z} g$ has vertices $V_G \times V_g$
- $\{(v, w), (v', w')\}$ is an edge if there is an $x \in V_g$ such that (w, x, w') is a path in g and v' is the x-th neighbor of v in G. (Note that the labeling condition on G ensures this is symmetric.)

Thus, if we start at $(v, w) \in G \textcircled{z} g$, a step to a random neighbor (v', w') has following form:

- Choose a random neighbor x of w in g.
- Set v' to be the x-th neighbor of v in G.
- Choose a random neighbor w' of x in g.

In particular, if we only consider the V_G-coordinate of a random walk of length ℓ in $G \textcircled{z} g$ (starting at a random vertex), it has the same distribution as the following process:

- Choose a random start vertex $v_0 \in V_G$.
- Take a random walk $w_1, w_2 \ldots, w_\ell$ in g^2.
- For $i > 0$, let v_i to be the w_i-th neighbor of v_{i-1} in G.
- Output $v_1, v_2, \ldots, v_\ell$.

Thus, each of of the segments of length $k/\log n$ in our sampler construction corresponds to a random walk on $G \textcircled{z} \tilde{H}$, projected onto the V_G-coordinate. But what about the boundaries between these segments? In this case, Construction 1 says we choose a new, entirely-random node of $\tilde{H}$ and then continue the walk on G. This is equivalent to taking a step on $G \textcircled{z} K_d$, i.e., the zig-zag product of G with a complete graph (with self-loops) on d nodes. Therefore, the output of our sampler is the projection onto the V_G-coordinate of a random walk on a *time-varying* graph that is $G \textcircled{z} \tilde{H}$ most of the time, and $G \textcircled{z} K_d$ once every $\log n$ steps. We now show that this output satisfies Definition 3.1.2 for the desired parameters.

First we note for any function $f : V_G \to [0, 1]$ there is a natural *lift* of f to $\hat{f} : V_G \times V_{\tilde{H}} \to [0, 1]$, defined by $\hat{f}(v, w) = f(v)$, and it is clear that the lift $\hat{f}$ has the same average as f. Therefore, to conclude that the projection of a random walk yields

[9] The zig-zag product of [RVW02] actually applies in much greater generality; however, this simplification suffices for our application.

a strong averaging sampler, it suffices to show that a random walk on the forgoing time-varying graph is a strong averaging sampler. By Remark 3.4.4 (following the proof of Theorem 3.1.1), it does not matter that the graph is varying over time: as long as the graph is a λ-expander at every point in time, the random walk is a good sampler. Thus, we are left with the task of showing that $G ⓩ \tilde{H}$ and $G ⓩ K_d$ are expanders. For this, we apply the following consequence of the main theorem of [RVW02]:

Lemma 3.2.3 ([RVW02], Corollary to Theorem 4.3)**.** *Let G be a regular graph of degree d whose edges are labeled with $1, \ldots, d$ in such a way that no two incident edges share the same label, and let g be a regular graph on d nodes. If G is a λ_G-expander and g is a λ_g-expander, then $G ⓩ g$ is a $(\lambda_G + \lambda_g)$-expander.*

By Lemma 3.2.1, G is a $2/n$-expander, and by assumption $\tilde{H}$ is a λ-expander for some constant $\lambda < 1$. So by Lemma 3.2.3, $G ⓩ \tilde{H}$ is a $(2/n + \lambda)$-expander, i.e. a λ'-expander for some constant $\lambda' < 1$ (when $n > 2/(1 - \lambda)$).

It is not hard to see that K_d, the complete graph (with self-loops) on d nodes, is a 0-expander, and therefore by Lemma 3.2.3, $G ⓩ K_d$ is a $(2/n)$-expander, i.e. a λ''-expander for some constant $\lambda'' < 1$ (when $n > 2$).

Thus our sampler stretches a seed of length $n + O(k)$ into k samples (of n bits each) that satisfy the bound from Theorem 3.1.1 for some constant $\lambda < 1$. Specifically, the sampler approximates the mean of the f_i's with error ϵ and confidence $1 - \gamma = 1 - e^{-\Omega(\epsilon^2 k)}$; in other words, the seed length is $n + O(k) = n + O(\log(1/\gamma)/\epsilon^2)$ and the sample complexity is $k = O(\log(1/\gamma)/\epsilon^2)$. Lemma 3.2.2 follows. □

3.2.3 An Alternate Sampler Construction

In this section we describe an alternate implementation of a sampler in $\mathbf{AC}^0[\oplus]$. While this construction uses many of the same tools as Construction 1, the fundamental approach is different and is inspired by the paradigm of sampler composition [BGG93, Gol97], rather than the zig-zag graph product.[10]

Recall that Construction 1 employed short walks on a small expander, H, to select the steps to be made in the large expander G. Thus, H was used to derandomize the long walk on G. For the present construction, however, we shall instead use a long walk on a large auxiliary graph (denoted G' below) to select seeds for short walks on a large expander graph (the Margulis/Gabber-Galil expander).

Recall the ϵ-biased expander G from the proof of Theorem 3.1.3. Here we define G' in the same way, but on the vertex set $\{0,1\}^{n+3\log n}$ instead of $\{0,1\}^n$; that is, we construct a $1/n$-biased set $S \subset \{0,1\}^{n+3\log n}$ of size $\mathrm{poly}(n)$, and take G' to be the Cayley graph on $\mathbb{Z}_2^{n+3\log n}$ with generators S.

[10] We note that the general *median of averages* composition of [BGG93, Gol97] does not result in an *averaging* sampler, which is the kind of sampler we consider in this chapter. Nonetheless, the same ideas can be employed here to obtain an averaging sampler (albeit with weaker parameters).

Construction 2.

1. *Choose a random starting vertex* $v'_0 \in G'$.
2. *Take a* $k/\log n$*-step random walk* $v'_1, \ldots, v'_{k/\log n}$ *on* G'.
3. *View each* $v'_i \in \{0,1\}^{n+3\log n}$ *as a* $(\log n)$*-step walk on the Margulis/Gabber-Galil expander of size* 2^n *and degree* 8.
4. *Expand each such walk* v'_i *into the nodes* $v_1^{(i)}, \ldots, v_{\log n}^{(i)} \in \{0,1\}^n$ *that it visits.*
5. *Output the* k *samples* $v_1^{(1)}, \ldots, v_{\log n}^{(1)}, \ldots, v_1^{(k/\log n)}, \ldots, v_{\log n}^{(k/\log n)}$.

This generator is computable in uniform $\mathbf{AC}^0[\oplus]$ since each of the required ingredients is computable in uniform $\mathbf{AC}^0[\oplus]$, as discussed in Section 3.2.1. Moreover, it is a good sampler for constant ϵ:

Proposition 1. *For any constant* $\epsilon > 0$*, Construction 2 is a strong* (γ, ϵ)*-averaging sampler with seed-length* $n + O(k) = n + O(\log(1/\gamma))$ *and sample complexity* $k = O(\log(1/\gamma))$ *(where the hidden constants depend on* ϵ*).*

Proof. We begin by noting that the number of random bits used by Construction 2 is $n + O(\log n) + (k/\log n) \cdot O(\log n) = n + O(k)$ (since we assume that $k = \Omega(\log n)$) and its sample complexity is k by construction. We show below that this sampler has error at most $\gamma = 2^{-\Omega(k)}$ when $\epsilon > 0$ is a constant; in other words, Construction 2 has seed length $n + O(k) = n + O(\log(1/\gamma))$ and sample complexity $k = O(\log 1/\gamma)$, as claimed.

In the following analysis, we shall confine ourselves to the case of sampling a single function (i.e., showing that Construction 2 is a non-strong averaging sampler). The proof that it is a strong sampler is completely analogous and simply follows from the fact that all the Chernoff bounds we apply are strong Chernoff bounds.

Let $f : \{0,1\}^n \to [0,1]$ be the function that is being sampled, and let $\rho = \mathrm{E}_x[f(x)]$. We first observe that a $(\log n)$-step random walk on the Margulis/Gaber-Galil Expander of size 2^n is likely to estimate ρ to within additive error $\epsilon/2$. Indeed, if we let $x_1, \ldots, x_{\log n}$ be a $\log n$-step random walk on this expander, then by Theorem 3.1.1,

$$\Pr\left[\left|\frac{1}{\log n}\sum_{j=1}^{\log n} f(x_j) - \mu\right| \geq \epsilon/2\right] \leq e^{-\frac{\epsilon^2(1-\lambda)\log n}{4}} = 1/n^c,$$

for some positive constant $c < 1/4$ (since we assume ϵ is constant).

Construction 2 then says to choose the seeds to these $(\log n)$-step walks according to a walk of length $k/\log n$ on a $2/n$-expander G' of degree $\mathrm{poly}(n)$. We expect at most a $1/n^c$ fraction of the short walks chosen in this way to yield poor estimates of ρ (i.e. not estimate ρ within $\pm\epsilon/2$); however, it is enough to hope that at most an $\epsilon/2$ fraction of the short walks are poor estimates in order to conclude that the average

over all $\Theta(\log n) \cdot k/\log n = k$ samples will be $\rho \pm \epsilon$. Moreover, we would like this to happen with probability $1 - 2^{-\Omega(k)}$. However, to conclude this we need to apply a sharper Chernoff bound than Theorem 3.1.1. Indeed, for a walk of length $k/\log n$ Theorem 3.1.1 will never yield an failure probability smaller than $2^{-O(k/\log n)}$ and we would like to bound the failure probability by $2^{-\Omega(k)}$.

Fortunately, G' is a very good expander (in particular, a $2/n$-expander) and so we may apply Corollary 3.4.6. Indeed, we are interested in accurately sampling a set of density $1/n^c$ (the bad $\log n$-step walks), and G' has eigenvalue $2/n \leq (1/n^c)^2/3$ (for sufficiently large n), as required to apply Corollary 3.4.6. Specifically, we let X be the random variable that counts how many of the $\log n$-step walks are not $\rho \pm \epsilon/2$, so that X has expectation at most $\frac{1}{n^c} \cdot \frac{k}{\log n}$, and then by Corollary 3.4.6

$$\Pr\left[X \geq \frac{\epsilon}{2} \cdot \frac{k}{\log n}\right] \leq \left(\frac{2e}{\epsilon \cdot n^c}\right)^{\frac{1}{2}\cdot\frac{\epsilon}{2}\cdot\frac{k}{\log n}} = \left(\frac{1}{n^{\Omega(1)}}\right)^{\frac{k}{\log n}} = 2^{-\Omega(k)},$$

since we assume ϵ is a constant.

That is, with probability $1 - 2^{-\Omega(k)}$, at least a $1 - \epsilon/2$ fraction of the $(\log n)$-step walks estimate ρ to within additive error $\epsilon/2$, and hence the average over all the samples is $\rho \pm \epsilon$. In other words, the probability that Construction 2 does not estimate ρ within additive error ϵ is at most $\gamma = 2^{-\Omega(k)}$, and the result follows. □

3.2.4 Sampling vs. Hitting

Many applications of expander walks do not require the full power of the Chernoff bound. For example, the randomness-efficient error reduction of [CW89, IZ89], ϵ-biased sets of [NN90], the amplification of [GIL+90] and the derandomized XOR lemma of [IW97] only require the *hitting property* of expander walks; i.e., they require that for any set $T \subseteq V$ of size at most $|V|/2$, the probability that a k-step random walk never leaves T is at most $2^{-\Omega(k)}$.[11] The latter three applications use the hitting property in a very natural way: in each case, the construction requires a sequence of objects that are combined in some way (e.g., addition, concatenation or XOR) and the proof of correctness only requires that at least one of these objects is "good" – furthermore, it is shown that "good" objects are abundant. Thus, by choosing these objects according to an expander walk and applying the hitting property, at least one of them will be "good" with high probability. For error reduction, it is less obvious that the (strong) hitting property suffices, although it does.[12]

[11] Strictly speaking, some of these results seem to require the *strong hitting property* of expander walks discussed in the introduction.

[12] Roughly, this is proved as follows: we suppose the algorithm of interest uses r-bits of randomness and has error probability at most 1/20. The new algorithm chooses k random r-bit strings according to an expander walk and outputs the majority vote of the k executions of the algorithm using these random strings. For the analysis, we fix an input x and consider the set of random strings T_x that

In light of this, it would have been sufficient to simply show that Constructions 1 and 2 satisfy the strong *hitting* property in order to prove the results discussed in Section 3.3. Nonetheless, we choose to show that these constructions are strong samplers – our motivation for doing so is twofold. Firstly, for certain applications (especially error-reduction) the Chernoff-like behavior of the sampler makes for simpler and, we feel, more natural proofs than the approach based on a strong hitting generator. Secondly, we would like to say that our sampler can be used in place of an expander walk for "any conceivable application", and some applications of expander walks do seem to require the strong sampling property – for instance, constructing *randomness extractors*.

Loosely speaking, an *extractor* $\mathrm{Ext} : \{0,1\}^m \times \{0,1\}^d \to \{0,1\}^n$ is a function that takes an m-bit input that is somewhat random, together with a short d-bit *seed* that is truly random and outputs an n-bit string that is very close to random. (For background on extractors, see the survey of Shaltiel [Sha02].)

One useful construction of an extractor (for sources of high, constant min-entropy) is based on random walks on expanders. Specifically, if $W : \{0,1\}^{n+O(k)} \to (\{0,1\}^n)^k$ computes a k-step walk on a constant-degree expander, then the function $\mathrm{Ext} : \{0,1\}^{n+O(k)} \times [k] \to \{0,1\}^n$ defined by $\mathrm{Ext}(x,s) \stackrel{\mathrm{def}}{=} W(x)_s$ is a strong extractor for sources x of min-entropy at least $(1-\beta)m$ for some constant $\beta > 0$; this follows from Theorem 3.1.1 (see [Zuc97] and [Zuc06]). Furthermore, the analysis of this extractor only depends on the fact that an expander walk is a strong sampler; therefore we may replace W with the sampler Γ of Theorem 3.1.3 to obtain such an extractor that is computable by uniform $\mathbf{AC}^0[\oplus]$ circuits. In particular, by the same proof as Proposition 4.2 of [Zuc06], we have the following corollary.

Corollary 3.2.4. *For all $\epsilon, \alpha > 0$, there exists $\beta > 0$ such that there is a family of strong $((1-\beta)m, \epsilon)$-extractors $\mathrm{Ext} : \{0,1\}^m \times \{0,1\}^d \to \{0,1\}^n$ with $n \geq (1-\alpha)m$ and $d \leq \log(\alpha m)$ that is computable by $\mathbf{AC}^0[\oplus]$ circuits of size $\mathrm{poly}(m)$. That is, for any m-bit source X with min-entropy at least $(1-\beta)m$ and an independent uniform d-bit seed Y, the distribution $(\mathrm{Ext}(X,Y),Y)$ is ϵ-close (in total variation distance) to the uniform distribution on $n+d$ bits.*

---

cause the original algorithm to err on x, and thus we have $|T_x| \leq 2^r/20$; to bound the probability that at least $k/2$ of the sampled strings land in T_x, we consider all sequences of sets $S_1, \ldots, S_k$ where each S_i is either T_x or its complement and where at least $k/2$ of the S_i's are T_x. It is easy to see that there are $2^k/2$ such sequences, and by an appropriate version of the strong hitting property and a suitably good expander graph, one can show that the probability that a walk exactly follows such a given sequence of sets is less than $(1/4)^k$. Therefore, the probability that a random walk hits any of these $2^k/2$ sequences is less than $(2^k/2) \cdot (1/4)^k < 2^{-k}$.

3.3 Proofs of Other Results

3.3.1 Error Reduction

Corollary 3.1.4 (restated)**.** *Let* $f : \{0,1\}^n \to \{0,1\}$ *be a function computable by polynomial-size* uniform $\mathbf{BP} \cdot \mathbf{AC}^0[\oplus]$ *(respectively,* $\mathbf{BP} \cdot \mathbf{TC}^0$ *or* $\mathbf{BP} \cdot \mathbf{NC}^1$*) circuits with error at most* $1/3$ *using* $r = r(n)$ *random bits. Then for any* $\delta = \delta(n) > 1/2^{O(\mathrm{poly}(n))}$*,* f *has polynomial-size* uniform $\mathbf{BP} \cdot \mathbf{AC}^0[\oplus]$ *(respectively,* $\mathbf{BP} \cdot \mathbf{TC}^0$ *or* $\mathbf{BP} \cdot \mathbf{NC}^1$*) circuits with error at most* δ *using* $r + O(\log(1/\delta))$ *random bits.*

Proof sketch. Let C_f be a circuit computing f. Construct the circuit that, on input $x \in \{0,1\}^n$, runs $k = \Theta(\log(1/\delta))$ copies of C_f in parallel using independent random r-bit blocks of randomness, and then computes the $(5/12, 7/12)$-approximate majority of the outputs [Ajt93]. (For $\mathbf{BP} \cdot \mathbf{TC}^0$ and $\mathbf{BP} \cdot \mathbf{NC}^1$ we can just compute the majority exactly.) Now, instead of using independent random bits for each block, we apply the construction of $\Gamma : \{0,1\}^{r+O(k)} \to (\{0,1\}^r)^k$ from Theorem 3.1.3 (with $\epsilon = 1/12$ and $\gamma = \delta$) to generate the necessary random bits from a seed of length $r + O(k)$.

For any fixed input x, the probability that a randomly chosen $r+O(k)$-bit random string causes the algorithm to fail (i.e., that more than $5/12$ of the outputs of Γ fall in the set of random strings that cause C_f to fail) is at most $2^{-\Omega(k)} = 2^{-\Omega(\Theta(\log 1/\delta))}$ since Γ is an averaging sampler (and the latter set has density at most $1/3$). By choosing the constants appropriately, this is at most δ and the result follows. □

Corollary 3.1.5 (restated)**.** *Let* $f : \{0,1\}^n \to \{0,1\}$ *be a function computable by polynomial-size* uniform $\mathbf{BP} \cdot \mathbf{AC}^0$ *circuits with error at most* $1/3$ *using* $r = r(n)$ *random bits. Then for any* $\delta = \delta(n) > 1/2^{O(\mathrm{poly}(n))}$*,* f *has polynomial-size* uniform $\mathbf{BP} \cdot \mathbf{AC}^0$ *circuits with error at most* δ *using* $\min\{r, \mathrm{polylog}(n)\} + O(\log(1/\delta))$ *random bits.*

Proof sketch. Let C_f be a circuit computing f. By applying Nisan's pseudorandom generator for $\mathbf{BP} \cdot \mathbf{AC}^0$ [Nis91] (which has been shown to be computable in $\mathbf{AC}^0$ in [Vio04]), we may assume, with no loss of generality, that C_f uses only $r' = r'(n) = \min\{r(n), \log^c(n)\}$ random bits for some constant c that may depend on f.

If $\delta \geq 1/2^{r'}$, then we may apply the construction of Corollary 3.1.4 to obtain a $\mathbf{BP} \cdot \mathbf{AC}^0$ circuit that has error at most δ and uses $r' + O(\log(1/\delta))$ bits of randomness. (The circuit is in $\mathbf{BP} \cdot \mathbf{AC}^0$, and not just $\mathbf{BP} \cdot \mathbf{AC}^0[\oplus]$ because one can readily check that all the necessary parities are on at most $O(r') = O(\log^c n)$ bits, and can therefore be computed by a constant-depth circuit of size $\mathrm{poly}(n)$.)

If, on the other hand, $\delta < 1/2^{r'}$, then we apply Corollary 3.1.4 with $\delta(n) = 2^{-r'}$ to obtain an $\mathbf{AC}^0$ circuit that has error at most $2^{-r'}$ and uses $r' + O(r') \leq O(r')$ random bits. We now apply $\Theta(\log(1/\delta)/r')$ such circuits in parallel (on the same input, but independent random strings), and take the approximate majority of their $\Theta(\log(1/\delta)/r')$ outputs. Thus we have a circuit taking $O(r') \cdot \Theta(\log(1/\delta)/r') = O(\log(1/\delta)) \leq$

$r' + O(\log(1/\delta))$ random bits and having error less than $(2^{-r'})^{\Theta(\log(1/\delta)/r')}$ (by a multiplicative Chernoff bound, such as Theorem 4.1 on p. 68 of [MR95]). This is at most δ by an appropriate setting of constants, and the result follows. □

3.3.2 Derandomization with Linear Advice

Corollary 3.1.6 (restated)**.** uniform $\mathbf{BP} \cdot \mathbf{AC}^0 \subseteq$ uniform $\mathbf{AC}^0/O(n)$.

Proof. Apply Corollary 3.1.5 to obtain a $\mathbf{BP} \cdot \mathbf{AC}^0$ circuit with error less than 2^{-n} using $r = O(n)$ random bits. By a union bound, at least one r-bit string causes the circuit to correctly decide all inputs. Fix one such string as the non-uniform advice and the result follows. □

Corollary 3.3.1. *Let* $\mathbf{AC}^0[\oplus_{\log}]$ *be the class of boolean functions computable by* $\mathrm{poly}(n)$*-size* $\mathbf{AC}^0$ *circuits having* $O(\log n)$ *parity gates, and similarly let* $\mathbf{AC}^0[\mathrm{SYM}_{\log}]$ *be the class of boolean functions computable by* $\mathrm{poly}(n)$*-size* $\mathbf{AC}^0$ *circuits having* $O(\log n)$ *arbitrary symmetric gates (e.g., parity and majority gates). Then the following inclusions hold:*

1. $\mathbf{BP} \cdot \mathbf{AC}^0[\oplus_{\log}] \subseteq \mathbf{AC}^0[\oplus]/O(n)$
2. $\mathbf{BP} \cdot \mathbf{AC}^0[\mathrm{SYM}_{\log}] \subseteq \mathbf{TC}^0/O(n)$

Proof sketch. The proof is similar to the proofs of Corollaries 3.1.5 and 3.1.6, except that we use the generator of Viola [Vio04] instead of Nisan's. Specifically, the generator from [Vio04] allows us to assume, without loss of generality, that any function $f \in \mathbf{BP} \cdot \mathbf{AC}^0[\oplus_{\log}]$ (respectively, $\mathbf{BP} \cdot \mathbf{AC}^0[\mathrm{SYM}_{\log}]$) can be computed by a $\mathbf{BP} \cdot \mathbf{AC}^0[\oplus]$ (respectively, $\mathbf{BP} \cdot \mathbf{TC}^0$) circuit using only $n^{o(1)}$ random bits. By applying Corollary 3.1.4, we may reduce the error to less than 2^{-n} using only $n^{o(1)} + O(n) = O(n)$ random bits. Finally, a union bound yields a single advice string of $O(n)$ bits that works for all inputs. □

3.3.3 An Optimal bitwise ϵ-biased generator in $\mathbf{AC}^0[\oplus]$

Corollary 3.1.7. *For every* $\epsilon > 0$ *and* m*, there is an* ϵ*-biased generator* $G : \{0,1\}^n \to \{0,1\}^m$ *with* $n = O(\log m + \log(1/\epsilon))$ *such that uniform* $\mathbf{AC}^0[\oplus]$ *circuits of size* $\mathrm{poly}(n, \log m) = \mathrm{poly}(n)$ *can compute* $G(s)_i$ *given* $(s, i) \in \{0,1\}^n \times [m]$.

Proof idea. We follow the approach of [GV04] and implement the ϵ-biased generator of Naor and Naor [NN90]. This generator employs a 7-wise independent generator to construct a small set of "distinguishers" (an object that is weaker than an ϵ-biased set), and from these it is possible to obtain an ϵ-biased generator by choosing many such distinguishers (independently) and taking a random linear combination of them. However, to improve the seed-length of their generator, [NN90] choose these distinguishers according to a walk on an expander graph. Thus, to construct a bit-wise

ϵ-biased generator in $\mathbf{AC}^0[\oplus]$, [GV04] require a bit-wise 7-wise independent generator and long expander walks that are each computable by $\mathbf{AC}^0[\oplus]$ circuits. Constructions of bit-wise 7-wise independent generators in $\mathbf{AC}^0[\oplus]$ are known [GV04, HV06], and since the analysis of [NN90] only uses the fact that an expander walk is a good hitting generator, our sampler construction from Section 3.2 can be used in place of the expander walk. □

3.4 Strong Chernoff Bounds for Expander Walks

In this section we give an elementary proof of a generalization of Gillman's *Chernoff Bound for Expander Walks* [Gil94] (Theorem 3.1.1) as well as a (strong) multiplicative Chernoff bound for expander walks that is sharper than Theorem 3.1.1 when the eigenvalue λ is small (Corollaries 3.4.5 and 3.4.6).

3.4.1 The Proof of Theorem 3.1.1

This section is devoted to proving the following theorem.

Theorem 3.1.1 (restated)**.** *Let G be a regular λ-expander on V and fix a sequence of functions $f_i : V \to [0,1]$ each with mean $\mu_i = \mathrm{E}_v[f_i(v)]$. If we consider a random walk $v_1, \ldots, v_k$ on G, then for all $\epsilon > 0$,*

$$\Pr\left[\left|\sum_{i=1}^{k} f_i(v_i) - \sum_{i=1}^{k} \mu_i\right| \geq \epsilon k\right] \leq 2e^{-\frac{\epsilon^2(1-\lambda)k}{4}}.$$

Wigderson and Xiao [WX05] have recently established essentially the same bound (up to constants) using techniques from perturbation theory. Gillman's proof (which treats the case where $f_1 = \cdots = f_k$) also employs results from perturbation and complex analysis to obtain a similar bound. In contrast, the proof presented here has only very modest prerequisites, which are summarized in the following paragraph.

Background We work with a regular λ-expander G on N nodes (see Definition 2.3.1). In particular, if we denote G's transition matrix by P and if we write $\mathbf{u} = (1/N, \ldots, 1/N) \in \mathbb{R}^N$, then $P\mathbf{u} = \mathbf{u}$ and

$$\max_{\substack{x \in \mathbb{R}^N \\ \langle x, \mathbf{u}\rangle = 0}} \frac{\|Px\|}{\|x\|} \leq \lambda.$$

For any $\mathbf{z} \in \mathbb{R}^N$, we let $\mathbf{z}^{\|} = \langle \mathbf{1}, \mathbf{z}\rangle \mathbf{u}$ denote the component of $\mathbf{z}$ in the direction of $\mathbf{1} = (1, \ldots, 1) \in \mathbb{R}^N$ and we let $\mathbf{z}^{\perp} = \mathbf{z} - \mathbf{z}^{\|} = \mathbf{z} - \langle \mathbf{1}, \mathbf{z}\rangle \mathbf{u}$ denote the component of $\mathbf{z}$ orthogonal to $\mathbf{1}$. Thus, for any $\mathbf{z} \in \mathbb{R}^N$, we have that $P\mathbf{z}^{\|} = \mathbf{z}^{\|}$ and $\|P\mathbf{z}^{\perp}\| \leq \lambda \cdot \|\mathbf{z}^{\perp}\|$. Another useful fact is that for any $\mathbf{z} \in \mathbb{R}^N$, the vector $P\mathbf{z}^{\perp}$ is orthogonal to $\mathbf{1}$ simply because $\langle \mathbf{1}, P\mathbf{z}^{\perp}\rangle = \mathbf{1}^T P \mathbf{z}^{\perp}$ and $\mathbf{1}^T P = \mathbf{1}^T$ since we assume G is regular.

Proof of Theorem 3.1.1. Define the random variable $X = \sum_i f_i(v_i)$ where $v_1, \ldots, v_k$ is a random walk on G, and let $\mu = \sum_i \mu_i = E[X]$. We shall bound the quantity $\Pr[X \geq \mu + \epsilon k]$ and the same bound will follow for $\Pr[X \leq \mu - \epsilon k]$ by replacing the functions $f_i(v)$ with $1 - f_i(v)$. Let $r \leq \log(1/\lambda)/2$ be a positive parameter to be specified later.

$$\Pr[X \geq \mu + \epsilon k] = \Pr\left[e^{rX} \geq e^{r\mu + r\epsilon k}\right] \leq \frac{\mathrm{E}\left[e^{rX}\right]}{e^{r\mu + r\epsilon k}} \tag{3.1}$$

where the last step follows by applying Markov's inequality.

We now focus on bounding $\mathrm{E}\left[e^{rX}\right]$. Let P be the probability transition matrix for G, and for each function f_i let E_i be a diagonal matrix with diagonal entries $\left(e^{rf_i(v)}\right)_{v \in V}$. It is not hard to see that

$$\mathrm{E}\left[e^{rX}\right] = \mathbf{1}^T E_k P E_{k-1} P \cdots E_1 P \mathbf{u}, \tag{3.2}$$

since every non-zero cross-term in this matrix product corresponds to exactly one walk $v_1, \ldots, v_k$ on G and each such term is exactly the probability of the walk times $e^{\sum_i f_i(v_i)}$.

Thus far, the techniques are quite standard. Indeed, the typical recipe for proving a Chernoff bound begins by reducing the task to bounding the moment-generating function $E\left[e^{rX}\right]$, and many previous tail bounds for Markov chains make use of the identity (3.2) (albeit with $E_1 = \cdots = E_k$) to bound $E\left[e^{rX}\right]$ [Gil94, Din95, Kah97, Ló98, LP04, WX05].

At this point, however, the proof diverges from previous approaches in that we bound (3.2) inductively using elementary manipulations. This is in contrast to the previous works that rely heavily on the machinery of perturbation theory (with the exception of [Kah97] which uses a novel eigenvalue argument), and also allows us to treat the case of sampling distinct function $f_1, \ldots, f_k$ (which is not readily amenable to previous techniques, except in the case of [WX05]).

Specifically, to bound the quantity (3.2), we study the sequence of vectors $\mathbf{z}_0 = \mathbf{u}$, $\mathbf{z}_1 = E_1 P \mathbf{u}$, $\mathbf{z}_2 = E_2 P E_1 P \mathbf{u}, \ldots$ inductively. Indeed, we note that

$$\mathrm{E}\left[e^{rT}\right] = \mathbf{1}^T E_k P E_{k-1} P \cdots E_1 P \mathbf{u} = \langle \mathbf{1}, \mathbf{z}_k \rangle = \langle \mathbf{1}, \mathbf{z}_k^{\parallel} \rangle = \sqrt{N} \cdot \|\mathbf{z}_k^{\parallel}\|, \tag{3.3}$$

and so our goal is to bound $\|\mathbf{z}_k^{\parallel}\|$.

We bound $\|\mathbf{z}_k^{\parallel}\|$ by first showing (in Lemma 3.4.2) that each of the $\mathbf{z}_i$'s remains nearly parallel to $\mathbf{u}$ (since E_i is close to the identity matrix when r is small, and moreover P helps shrink any component of $\mathbf{z}_i$ that is not parallel to $\mathbf{u}$). Then we observe (in Lemma 3.4.3) that E_i stretches $\mathbf{u}$ (and hence the $\mathbf{z}_i$'s, since they are nearly parallel to $\mathbf{u}$) by a factor of $\mathrm{E}_v\left[e^{rf_i(v)}\right] \approx e^{r\,\mathrm{E}_v[f_i(v)]} = e^{r\mu_i}$ (again, when r is small) which in turn ensures that $\mathrm{E}\left[e^{rT}\right] \approx e^{r\mu}$; more precisely, we find that $\mathrm{E}\left[e^{rT}\right] \leq e^{r\mu + r^2 k/(1-\lambda)}$. This bounds the probability in (3.1) by $e^{(r^2/(1-\lambda) - \epsilon r)k}$, and the result follows by choosing r to minimize this probability, i.e. $r = \epsilon(1-\lambda)/2$.

In the manipulations that follow, it may be worthwhile to bear in mind that we ultimately choose r to be small and therefore $e^r - 1 \approx r$ is also small.

The following lemma bounds how long $\mathbf{z}_{i+1}^{\|}$ and $\mathbf{z}_{i+1}^{\perp}$ can be relative to $\mathbf{z}_i^{\|}$ and $\mathbf{z}_i^{\perp}$.

Lemma 3.4.1. *Let P and $0 < r \leq \log(1/\lambda)/2$ be as above, and let E be a diagonal matrix with diagonal entries $(e^{rf(v)})_{v \in V}$ for some function $f : V \to [0,1]$ with mean $\rho = E_v[f(v)]$. Then for any $\mathbf{z} \in \mathbb{R}^N$:*

1. $\|(EP\mathbf{z}^{\|})^{\|}\| \leq (1 + (e^r - 1)\rho) \cdot \|\mathbf{z}^{\|}\|$.
2. $\|(EP\mathbf{z}^{\|})^{\perp}\| \leq \frac{e^r - 1}{2} \cdot \|\mathbf{z}^{\|}\|$.
3. $\|(EP\mathbf{z}^{\perp})^{\|}\| \leq \frac{e^r - 1}{2} \cdot \lambda \cdot \|\mathbf{z}^{\perp}\|$.
4. $\|(EP\mathbf{z}^{\perp})^{\perp}\| \leq \sqrt{\lambda} \cdot \|\mathbf{z}^{\perp}\|$.

Proof. (1): $(EP\mathbf{z}^{\|})^{\|} = (E\mathbf{z}^{\|})^{\|} = \langle \mathbf{1}, E\mathbf{z}^{\|} \rangle \mathbf{u} = \langle \mathbf{1}, E\mathbf{u} \rangle \mathbf{z}^{\|} = \mathrm{E}_v \left[e^{rf(v)} \right] \cdot \mathbf{z}^{\|}$, and using the fact that $e^{rx} \leq 1 + (e^r - 1)x$ for all $0 \leq x \leq 1$, we have

$$\|(EP\mathbf{z}^{\|})^{\|}\| = \mathop{\mathrm{E}}_{v} \left[e^{rf(v)} \right] \cdot \|\mathbf{z}^{\|}\| \leq \mathop{\mathrm{E}}_{v} [1 + (e^r - 1)f(v)] \cdot \|\mathbf{z}^{\|}\| = (1 + (e^r - 1)\rho) \cdot \|\mathbf{z}^{\|}\|.$$

(2): Recalling that $(\mathbf{z}^{\|})^{\perp} = 0$ for all $\mathbf{z}$, we note that for any $\alpha \in \mathbb{R}$,

$$(EP\mathbf{z}^{\|})^{\perp} = (E\mathbf{z}^{\|})^{\perp} = ((E - \alpha \cdot I)\mathbf{z}^{\|})^{\perp} + (\alpha \cdot \mathbf{z}^{\|})^{\perp} = ((E - \alpha \cdot I)\mathbf{z}^{\|})^{\perp}.$$

Thus, we choose $\alpha = \frac{e^r + 1}{2}$ so that $E - \alpha \cdot I$ is diagonal with entries bounded by $\frac{e^r - 1}{2}$ in absolute value (since $e^r - \alpha = \frac{e^r - 1}{2}$ and $e^0 - \alpha = -\frac{e^r - 1}{2}$). Then,

$$\|(EP\mathbf{z}^{\|})^{\perp}\| = \|((E - \alpha \cdot I)\mathbf{z}^{\|})^{\perp}\| \leq \frac{e^r - 1}{2} \cdot \|\mathbf{z}^{\|}\|.$$

(3): Recalling that $(P\mathbf{z}^{\perp})^{\|} = 0$ for all $\mathbf{z}$, we note that for any $\alpha \in \mathbb{R}$,

$$(EP\mathbf{z}^{\perp})^{\|} = ((E - \alpha \cdot I)P\mathbf{z}^{\perp})^{\|} + (\alpha \cdot P\mathbf{z}^{\perp})^{\|} = ((E - \alpha \cdot I)P\mathbf{z}^{\perp})^{\|}.$$

Again, we choose $\alpha = \frac{e^r + 1}{2}$ so that $E - \alpha \cdot I$ is diagonal with entries bounded by $\frac{e^r - 1}{2}$ in absolute value, and get

$$\|(EP\mathbf{z}^{\perp})^{\|}\| = \|((E - \alpha \cdot I)P\mathbf{z}^{\perp})^{\|}\| \leq \frac{e^r - 1}{2} \cdot \|P\mathbf{z}^{\perp}\| \leq \frac{e^r - 1}{2} \cdot \lambda \cdot \|\mathbf{z}^{\perp}\|,$$

where the last inequality uses the fact that $\|P\mathbf{z}^{\perp}\| \leq \lambda \cdot \|\mathbf{z}^{\perp}\|$ for any vector $\mathbf{z} \in \mathbb{R}^N$.

(4): $\|(EP\mathbf{z}^{\perp})^{\perp}\| \leq \|EP\mathbf{z}^{\perp}\| \leq e^r \cdot \|P\mathbf{z}^{\perp}\| \leq e^r \lambda \cdot \|\mathbf{z}^{\perp}\|$, and since we assume that $r \leq \log(1/\lambda)/2$, this is at most $\sqrt{\lambda} \cdot \|\mathbf{z}^{\perp}\|$. □

Recall that $\mathbf{z}_0 = \mathbf{u}$ and $\mathbf{z}_{i+1} = E_{i+1}P\mathbf{z}_i$. We now show that $\mathbf{z}_i^{\perp}$ remains short compared to the previous $\mathbf{z}_j^{\|}$'s.

Lemma 3.4.2. $\|\mathbf{z}_i^{\perp}\| \leq \frac{e^r-1}{1-\lambda} \cdot \max_{j<i}\{\|\mathbf{z}_j^{\parallel}\|\}$ *for* $1 \leq i \leq k$.

Proof. By the triangle inequality,

$$\|\mathbf{z}_i^{\perp}\| = \|(E_i P \mathbf{z}_{i-1})^{\perp}\| = \|(E_i P \mathbf{z}_{i-1}^{\parallel})^{\perp} + (E_i P \mathbf{z}_{i-1}^{\perp})^{\perp}\| \leq \|(E_i P \mathbf{z}_{i-1}^{\parallel})^{\perp}\| + \|(E_i P \mathbf{z}_{i-1}^{\perp})^{\perp}\|.$$

Thus, by Items 2 and 4 of Lemma 3.4.1, we have $\|\mathbf{z}_i^{\perp}\| \leq \frac{e^r-1}{2} \cdot \|\mathbf{z}_{i-1}^{\parallel}\| + \sqrt{\lambda} \cdot \|\mathbf{z}_{i-1}^{\perp}\|$.

Recursively applying this bound, and noting that $\|\mathbf{z}_0^{\perp}\| = 0$, we have

$$\|\mathbf{z}_i^{\perp}\| \leq \frac{e^r-1}{2} \cdot \sum_{j=0}^{i-1} (\sqrt{\lambda})^j \|\mathbf{z}_{i-j-1}^{\parallel}\| \leq \frac{e^r-1}{2(1-\sqrt{\lambda})} \cdot \max_{j<i}\{\|\mathbf{z}_j^{\parallel}\|\}.$$

The lemma follows by noting that $1/(1-\sqrt{\lambda}) = (1+\sqrt{\lambda})/(1-\lambda) \leq 2/(1-\lambda)$ since $\lambda \in [0,1]$. □

We now use Lemma 3.4.2 to bound $\|\mathbf{z}_i^{\parallel}\|$ inductively.

Lemma 3.4.3. $\|\mathbf{z}_i^{\parallel}\| \leq \exp\left\{(e^r-1)\mu_i + \frac{\lambda \cdot (e^r-1)^2}{2(1-\lambda)}\right\} \cdot \max_{j<i}\{\|\mathbf{z}_j^{\parallel}\|\}$, *for* $1 \leq i \leq k$.

Proof. By the triangle inequality,

$$\|\mathbf{z}_i^{\parallel}\| = \|(E_i P \mathbf{z}_{i-1})^{\parallel}\| = \|(E_i P \mathbf{z}_{i-1}^{\parallel})^{\parallel} + (E_i P \mathbf{z}_{i-1}^{\perp})^{\parallel}\| \leq \|(E_i P \mathbf{z}_{i-1}^{\parallel})^{\parallel}\| + \|(E_i P \mathbf{z}_{i-1}^{\perp})^{\parallel}\|.$$

Thus, by Items 1 and 3 of Lemma 3.4.1, we have $\|\mathbf{z}_i^{\parallel}\| \leq (1+(e^r-1)\mu_i) \cdot \|\mathbf{z}_{i-1}^{\parallel}\| + \frac{e^r-1}{2} \cdot \lambda \cdot \|\mathbf{z}_{i-1}^{\perp}\|$, and so by Lemma 3.4.2,

$$\|\mathbf{z}_i^{\parallel}\| \leq (1+(e^r-1)\mu_i) \cdot \|\mathbf{z}_{i-1}^{\parallel}\| + \frac{\lambda \cdot (e^r-1)^2}{2(1-\lambda)} \cdot \max_{j<i-1}\{\|\mathbf{z}_j^{\parallel}\|\}$$

$$\leq \left(1 + (e^r-1)\mu_i + \frac{\lambda \cdot (e^r-1)^2}{2(1-\lambda)}\right) \cdot \max_{j<i}\{\|\mathbf{z}_j^{\parallel}\|\}.$$

Finally, using the fact that $1 + x \leq e^x$ for all $x \geq 0$, we conclude that this is at most

$$\exp\left\{(e^r-1)\mu_i + \frac{\lambda \cdot (e^r-1)^2}{2(1-\lambda)}\right\} \cdot \max_{j<i}\{\|\mathbf{z}_j^{\parallel}\|\}.$$

□

Recalling that $\|\mathbf{z}_0^{\parallel}\| = 1/\sqrt{N}$, Lemma 3.4.3 implies that for all $j \geq 0$:

$$\|\mathbf{z}_j^{\parallel}\| \leq \frac{1}{\sqrt{N}} \prod_{i=1}^{j} \exp\left\{(e^r-1)\mu_i + \frac{\lambda \cdot (e^r-1)^2}{2(1-\lambda)}\right\},$$

and in particular, by (3.3),

$$\begin{aligned} \mathrm{E}\left[e^{rX}\right] = \sqrt{N} \cdot \|\mathbf{z}_k^{\parallel}\| &\leq \prod_{i=1}^{k} \exp\left\{(e^r-1)\mu_i + \frac{\lambda\cdot(e^r-1)^2}{2(1-\lambda)}\right\} \\ &= \exp\left\{(e^r-1)\mu + \frac{\lambda\cdot(e^r-1)^2}{2(1-\lambda)}\cdot k\right\}. \end{aligned} \tag{3.4}$$

To simplify this expression, we shall assume that $r \leq 1/2$ (and thus that $e^r - 1 \leq r + 2r^2/3 \leq 4r/3$) and we note that $\mu \leq k$:

$$\mathrm{E}\left[e^{rX}\right] \leq \exp\left\{(r+r^2)\mu + \frac{\lambda\cdot(4r/3)^2}{2(1-\lambda)}\cdot k\right\} \leq e^{r\mu + r^2\cdot\left(1+\frac{\lambda}{1-\lambda}\right)\cdot k} = e^{r\mu+\frac{r^2 k}{1-\lambda}}.$$

Thus, by (3.1) we have

$$\Pr\left[X \geq \mu + \epsilon k\right] \leq \frac{\mathrm{E}\left[e^{rX}\right]}{e^{r\mu + r\epsilon k}} \leq e^{\left(\frac{r^2}{1-\lambda} - r\epsilon\right)k}.$$

Finally, we minimize this probability by setting $r = (1-\lambda)\epsilon/2$, noting that r is indeed at most $\min\{1/2, \log(1/\lambda)/2\}$ simply because $\epsilon \leq 1$ and $1 - \lambda \leq \log(1/\lambda)$ for all $\lambda \in [0,1]$. It follows that

$$\Pr\left[X \geq \mu + \epsilon k\right] \leq e^{-\frac{\epsilon^2(1-\lambda)k}{4}}.$$

□

Remark 3.4.4. *One can readily see that the same proof applies even if the graph is different for each of the k steps, as long as it is a λ-expander at each step. This is observation is important for the proof of correctness of our sampler (Theorem 3.1.3), as that construction concerns a walk on an expander graph that is varying from one step to the next step. This observation is not unique to our proof of the Chernoff bound, and this same property has been exploited before, most notably in the hardness amplification result of Goldreich et al. [GIL+90] (although there, they only require the hitting property of expander walks, and not the stronger sampling properties guaranteed here).*

3.4.2 A Multiplicative Strong Chernoff Bound

As exemplified in Section 3.2.3, it is sometimes useful to have tail bounds that are sharper than Theorem 3.1.1 when considering rare events and large deviations from the mean. In this section we prove bounds (Corollaries 3.4.5 and 3.4.6) that improve upon Theorem 3.1.1 in this case, provided that the eigenvalue λ is sufficiently small.

The motivation for such bounds comes from the case of independent random variables. Indeed, it is well known that the standard *additive* Chernoff bound of the

form $\Pr[X \geq E[X] + \epsilon k] \leq e^{-\Omega(\epsilon^2 k)}$ (where $X = X_1 + \cdots + X_k$ is a sum of i.i.d. Bernoulli random variables X_i) is suboptimal when the mean of the X_i's is very small and ϵ is large. For instance, if we take $E[X_i] = 1/k$ and we consider $\Pr[X \geq k/2]$, then the standard Chernoff bound (with $\epsilon = 1/2 - 1/k$) only bounds this probability by $e^{-\Omega(k)}$, when in fact it is possible to show that $\Pr[X \geq k/2] \leq e^{-\Omega(k \log k)}$ (e.g., via a *multiplicative* Chernoff bound, such as Theorem 4.1 of [MR95]).

The analogous case for expander walks is when the set we are trying to sample is very small (or more generally when, in the notation of Theorem 3.1.1, the μ_i are small). To obtain a sharper bound in this setting, however, one should have an eigenvalue λ that is quite small, and then it is possible to slightly modify the proof of Theorem 3.1.1 to obtain a bound analogous to Theorem 4.1 of [MR95]:

Corollary 3.4.5. *Fix a sequence of functions $f_i : V \to [0,1]$ each with mean $\mu_i = \mathrm{E}_v[f_i(v)]$ and let $\mu = \sum_{i=1}^{k} \mu_i$. If we consider a random walk $v_1, \ldots, v_k$ on a λ-expander G, then for all $\delta > 0$,*

$$\Pr\left[\sum_{i=1}^{k} f_i(v_i) \geq (1+\delta)\mu\right] \leq \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}}\right)^{\left(1 - \frac{\lambda}{1-\lambda} \cdot \left(\frac{k}{\mu}\right)^2\right)\mu}.$$

In particular, when $\lambda = 0$ this matches Theorem 4.1 of [MR95]. We also note that $e^{\delta}/(1+\delta)^{(1+\delta)} = e^{\delta - (1+\delta)\log(1+\delta)} \leq 1$ for all $\delta \geq 0$, simply because $\delta \leq (1+\delta)\log(1+\delta)$ (as can easily be verified by comparing the derivatives of both sides). Thus, the bound is nontrivial whenever $\frac{\lambda}{1-\lambda} < (\mu/k)^2$. So, for instance, if we are interested in sampling a set of density α, then we should use an expander with $\lambda \lesssim \alpha^2$ in order to meaningfully apply Corollary 3.4.5. In particular, if λ is sufficiently small compared to α, say $\lambda \leq \alpha^2/3$, then the bound from Corollary 3.4.5 is never more than the square-root of the bound for independent random variables (i.e., Theorem 4.1 of [MR95]) – see also the proof of Corollary 3.4.6.

Consequently, Corollary 3.4.5 is sharper than Theorem 3.1.1 in the same way that a multiplicative Chernoff bound is sharper than the standard additive Chernoff bound, provided that λ is sufficiently small. For instance, analogously to the example of independent random variables described above, if we have a set $S \subseteq V$ of nodes of density $1/k$ and take a random walk of length k, then the probability of landing in S at least $k/2$ times is at most $e^{-\Omega(k \log k)}$ (provided that $\lambda \lesssim 1/k^2$), and not just $e^{-\Omega(k)}$ (which is all that Theorem 3.1.1 gives). When bounding the probability of large derivations from the mean (as in the previous example), the bound from Corollary 3.4.6 is often easier to work with and essentially as good as Corollary 3.4.5. Indeed, this is the bound that we employ in the analysis of our alternate sampler construction from Section 3.2.3 (i.e., Proposition 1).

Proof of Corollary 3.4.5. The proof is identical to the proof Theorem 3.1.1 up to the derivation of (3.4), at which point we make a different choice of the parameter r. In particular, we first equate the notation of Corollary 3.4.5 with that of Theorem 3.1.1

by taking $\epsilon = \delta\mu/k$. Indeed, then $\Pr[X \geq \mu + \epsilon k] = \Pr[X \geq (1+\delta)\mu]$, and so (3.1) now becomes

$$\Pr[X \geq (1+\delta)\mu] \leq \frac{\mathrm{E}\left[e^{rX}\right]}{e^{r(1+\delta)\mu}}. \tag{3.5}$$

Next, in contrast to the proof of Theorem 3.1.1, we choose $r = \log(1+\delta)$ (rather than $r = (1-\lambda)\epsilon/2 = (1-\lambda)\delta\mu/2k$). Before proceeding, however, we must check that $r \leq \log(1/\lambda)/2$, as required throughout the proof of Theorem 3.1.1: we may assume, with no loss of generality, that $1+\delta \leq k/\mu$ (indeed, the result is trivial if $\delta > k/\mu - 1$), and we may assume that $\lambda \leq (\mu/k)^2$, since otherwise the bound stated in the Corollary is larger than 1. Therefore, we have $r \leq \log(k/\mu) \leq \log(1/\lambda)/2$.

Substituting $r = \log(1+\delta)$ into (3.4), we have

$$\mathrm{E}\left[e^{rX}\right] \leq e^{\mu\delta + \frac{\lambda\delta^2 k}{2(1-\lambda)}},$$

and thus, by (3.5), we have

$$\Pr[X \geq (1+\delta)\mu] \leq \frac{e^{\mu\delta + \frac{\lambda\delta^2 k}{2(1-\lambda)}}}{e^{\log(1+\delta)(1+\delta)\mu}} = e^{\left(\delta + \frac{\lambda\delta^2}{2(1-\lambda)}\cdot\frac{k}{\mu} - (1+\delta)\log(1+\delta)\right)\mu}. \tag{3.6}$$

To bound this expression, we first establish that

$$\frac{\delta^2}{2} \leq \frac{k}{\mu}\cdot\left[(1+\delta)\log(1+\delta) - \delta\right]. \tag{3.7}$$

Indeed, both sides of this expression are equal to 0 when $\delta = 0$, and we shall verify that the derivative of the left-hand side (with respect to $\delta \in [0, k/\mu - 1]$) is always bounded by the derivative of the right-hand side: the derivative of the left-hand side is δ and the derivative of the right-hand side is $\frac{k}{\mu}\cdot\log(1+\delta)$, and $\delta \leq (1+\delta)\log(1+\delta) \leq \frac{k}{\mu}\cdot\log(1+\delta)$ for $\delta \in [0, k/\mu - 1]$, so (3.7) holds.

Thus, by applying (3.7) to bound the $\delta^2/2$ term that appears in (3.6), we have

$$\Pr[X \geq (1+\delta)\mu] \leq e^{\left(\delta + \frac{\lambda}{1-\lambda}\cdot\left(\frac{k}{\mu}\right)^2\cdot[(1+\delta)\log(1+\delta)-\delta] - (1+\delta)\log(1+\delta)\right)\mu}$$

$$= e^{\left(1 - \frac{\lambda}{1-\lambda}\cdot\left(\frac{k}{\mu}\right)^2\right)\cdot[\delta - (1+\delta)\log(1+\delta)]\mu},$$

and the result follows. □

As mentioned above, a bound like Corollary 3.4.5 is better than Theorem 3.1.1 when μ is small and δ is large (and when we can afford to choose λ to be sufficiently small). With this in mind we mention a simpler, albeit less general, form of the bound:

Corollary 3.4.6. *Fix a sequence of functions $f_i : V \to [0,1]$ each with mean $\mu_i = \mathrm{E}_v[f_i(v)]$ and let $\mu = \sum_{i=1}^k \mu_i$. Furthermore, let G be a regular λ-expander on V for some $\lambda \leq (\mu/k)^2/3$. If we consider a random walk $v_1, \ldots, v_k$ on G then*

$$\Pr\left[\sum_{i=1}^k f_i(v_i) \geq t\right] \leq \left(\frac{e\mu}{t}\right)^{t/2}.$$

Proof. We let $\delta = t/\mu - 1$ so that $t = (1+\delta)\mu$. Then by Corollary 3.4.5,

$$\Pr\left[\sum_{i=1}^k f_i(v_i) \geq t\right] \leq \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^{\mu\left(1-\frac{\lambda}{1-\lambda}\cdot\left(\frac{k}{\mu}\right)^2\right)}.$$

Since we assume $\lambda \leq (\mu/k)^2/3 \leq 1/3$, we have that $\frac{\lambda}{1-\lambda} \cdot \left(\frac{k}{\mu}\right)^2 \leq 1/2$, and thus

$$\Pr\left[\sum_{i=1}^k f_i(v_i) \geq t\right] \leq \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^{\mu/2} < \left(\frac{e}{1+\delta}\right)^{(1+\delta)\mu/2} = \left(\frac{e\mu}{t}\right)^{t/2}.$$

□

3.5 Open Questions

In this chapter we construct an extremely efficient sampler that is in many respects "just as good" as random-walk sampling using a constant-degree expander graph; a natural question that is left open, however, is whether $\mathbf{AC}^0[\oplus]$ can actually compute long expander walks – that is, whether there exists a family of constant-degree expander graphs for which a family of $\mathbf{AC}^0[\oplus]$ circuits can compute walks $v_1, \ldots, v_k$ when given a starting node v_0 and steps $s_1, \ldots, s_k$.[13] Our techniques come close: indeed, the approach of Section 3.2.1 can easily be modified to obtain a circuit that computes walks on the zig-zag product $G \textcircled{z} H$ when given a circuit for computing walks on H. (Recall that G has size 2^n and degree $\mathrm{poly}(n)$, and H has size $\mathrm{poly}(n)$.) Thus, if we take H to be an exponentially-smaller copy of G (of size $\mathrm{poly}(n)$ and degree $\mathrm{polylog}(n)$) – rather than a constant-degree expander – $\mathbf{AC}^0[\oplus]$ can compute long walks on H and therefore can also compute walks on the graph $G \textcircled{z} H$ of degree $\mathrm{polylog}(n)$; in fact, by recursively applying a constant number of such zig-zag products, we obtain an expander G' of size at least 2^n and degree at most $\mathrm{poly}(\log^{(t)} n)$ for any constant t. Alternatively, by repeating this recursion $\log^* n$ times, we obtain a constant-degree expander G' and a family of circuits of depth $O(\log^* n)$ for computing walks on G'. Can this depth be reduced to $O(1)$? Even less ambitiously, is there any

[13] Or, one could even remove the restriction on the input format and just ask for a generator whose output distribution is the same (or even statistically close) to a random walk on an expander.

family of constant-degree expander graphs for which a family of (nonuniform) $\mathbf{NC}^1$ circuits can compute long walks?

There is also the question of lower-bounds. We suspect that $\mathbf{AC}^0$ cannot compute samplers that match the parameters of our $\mathbf{AC}^0[\oplus]$ construction. One approach to showing this is to use the equivalence of samplers and extractors from [Zuc97] (see also the discussion in Section 3.2.4) and show that $\mathbf{AC}^0$ cannot compute a (strong) extractor for sources of high constant min-entropy. Viola [Vio04] has shown that $\mathbf{AC}^0$ cannot compute an extractor for sources of low min-entropy; however, his techniques do not seem to apply directly in this setting.

Chapter 4

Derandomized Hardness Amplification within NP

4.1 Introduction

Average-case complexity is a fundamental topic in complexity theory, whose study has at least two distinct motivations. On one hand, it may provide more meaningful explanations than worst-case complexity about the intractability of problem instances actually encountered in practice. On the other hand, it provides us with methods to generate hard instances, allowing us to harness intractability for useful ends such as cryptography and derandomization.

One of the goals of this area is to establish connections between average-case complexity and worst-case complexity. While this has been accomplished for high complexity classes such #**P** and **EXP** (e.g. [Lip89, BF90, BFL91, FL96, CPS99, STV01, TV02, Vio04]), it remains a major open question for **NP**. In fact, there are results showing that such connections for **NP** are unlikely to be provable using the same kinds of techniques used for the high complexity classes [FF93, BT03, Vio05a, Vio04].

A more modest goal is "hardness amplification", where we seek to establish connections between "mild" average-case complexity and "strong" average-case complexity. That is, given a problem for which a nonnegligible fraction of inputs are "hard", can we obtain a problem for which almost all inputs are hard? To make this precise, let us define "hard".

Definition 4.1.1. *For $\delta \in [0, 1/2]$, a function $f : \{0,1\}^n \to \{0,1\}$ is* δ-hard for size *s if every circuit of size s fails to compute f on at least a δ fraction of inputs.*

Note that the maximum value of the hardness parameter δ is $1/2$ because f is boolean (and so can trivially be computed with error probability at most $1/2$). This notion of hardness is fairly standard (e.g. in the literature on derandomization starting from [NW94]), but we remark that it differs from Levin's notion of average-case

complexity [Lev86] in several ways. Most importantly, Levin's formulation corresponds to algorithms that always either give the correct answer or say "don't know," whereas we consider even "heuristic" algorithms that can make arbitrary errors. (See Impagliazzo's survey [Imp95b].)

The *hardness amplification problem* is to convert a function f that is δ-hard for size s into a function f' that is $(1/2-\epsilon)$-hard for size polynomially related to s. Typically, $\delta = 1/\operatorname{poly}(n)$ and the aim is to make $\epsilon = \epsilon(n)$ vanish as quickly as possible.

The standard approach to hardness amplification employs Yao's XOR Lemma [Yao82] (see [GNW95]): Given a mildly hard-on-average function $f : \{0,1\}^n \to \{0,1\}$, we define $f' : \{0,1\}^{n \cdot k} \to \{0,1\}$ by

$$f'(x_1, \ldots, x_k) \stackrel{\text{def}}{=} f(x_1) \oplus f(x_2) \oplus \cdots \oplus f(x_k).$$

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

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

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

However, if we are interested in hardness amplification within **NP** (i.e. f and f' are characteristic functions of languages in **NP**), we cannot use the XOR lemma; it does not ensure that f' is in **NP** when f is in **NP**. Hardness amplification within **NP** was first addressed in a recent paper of O'Donnell [O'D04], which is the starting point for our work.

4.1.1 O'Donnell's Hardness Amplification

To ensure that the new function f' is in **NP** when f is in **NP**, O'Donnell [O'D04] was led to study constructions of the form

$$f'(x_1, \ldots, x_k) \stackrel{\text{def}}{=} C(f(x_1), f(x_2), \ldots, f(x_k)), \tag{4.1}$$

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

Remarkably, O'Donnell was able to precisely characterize the amplification properties of Construction 4.1 in terms of a combinatorial property of the combining

function C, called its *expected bias*. (The actual definition is not needed for this discussion, but can be found in Section 4.3.) By finding a monotone combining function in which this expected bias is small, he obtained the first positive result on hardness amplification in **NP**:

O'Donnell's Theorem [O'D04]. *If* **NP** *has a balanced function that is* $1/\operatorname{poly}(n)$*-hard for polynomial-size circuits, then* **NP** *has a function that is* $(1/2-1/n^{1/2-\alpha})$*-hard for polynomial-size circuits (where* α *is an arbitrarily small positive constant).*

However, the amplification provided by O'Donnell's theorem is not as strong as what the XOR Lemma gives. It is limited to $1/2-1/\sqrt{n}$, regardless of the circuit size s for which the original function is hard, even if s is exponentially large. The XOR Lemma, on the other hand, amplifies to $1/2 - 1/s^{\Omega(1)}$. O'Donnell showed that this difference is inherent — no construction of the form (4.1) with a monotone combining function C can always amplify hardness to better than $1/2 - 1/n$.[1]

4.1.2 Our Result

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

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

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

Items 1–3 match the parameters of the Yao's XOR Lemma. However, subsequent "derandomizations" of the XOR Lemma [Imp95a, IW97] actually amplify up to $1/2 - 1/2^{\Omega(n)}$ rather than just $1/2 - 1/2^{\Omega(\sqrt{n})}$ in the case $s(n) = 2^{\Omega(n)}$. This gap is *not* inherent in our approach and, as mentioned below, would be eliminated given a corresponding improvement in one of the tools we employ.

Of course, our construction cannot be of the form in Construction (4.1). Below we describe our two main points of departure.

[1] The gap between O'Donnell's positive result of $1/2 - 1/\sqrt{n}$ and his negative result of $1/2 - 1/n$ is not significant for what follows, and in particular, it will be subsumed by our improvements.

4.1.3 Techniques

To explain how we bypass it, we first look more closely at the source of the $1/2 - 1/n$ barrier. The actual barrier is $1/2 - 1/k$, where k is the input length of the monotone combining function C. (This is based on a result of [KKL88], see [O'D04].) Since in Construction (4.1), f' has input length $n' = n \cdot k \geq k$, it follows that we cannot amplify beyond $1/2 - 1/n'$.

Derandomization. Given the above, our first idea is to break the link between the input length of f' and the input length of the combining function C. We do this by *derandomizing* O'Donnell's construction. That is, the inputs $x_1, \ldots, x_k$ are no longer taken independently (as in Construction (4.1)), but are generated pseudorandomly from a short seed of length $n' \ll k$, which becomes the actual input to f'. Our method for generating the x_i's is based on combinatorial designs (as in the Nisan–Wigderson generator [NW94]) and Nisan's pseudorandom generator for space-bounded computation [Nis92]; it reduces the input length of f' from $n \cdot k$ to $n' = O(n^2 + \log^2 k)$. We stress that this derandomization is unconditional, i.e. requires no additional complexity assumption. We also remark that it is the quadratic seed length of Nisan's generator that limits our amplification to $1/2 - 1/2^{\Omega(\sqrt{n})}$ rather than $1/2 - 1/2^{\Omega(n)}$ in Part 3 of our Main Theorem, and thus any improvement in Nisan's generator would yield a corresponding improvement in our result.

Similar derandomizations have previously been achieved for Yao's XOR Lemma by Impagliazzo [Imp95a] and Impagliazzo and Wigderson [IW97]. The analysis of such derandomizations is typically tailored to a particular proof, and indeed both [Imp95a, IW97] gave new proofs of the XOR Lemma for that purpose. In our case, we do not know how to derandomize O'Donnell's original proof, but instead manage to derandomize a different proof due to Trevisan [Tre03].

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

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

To overcome this difficulty, we observe that we need only give an efficient *nondeterministic* algorithm for f'. Each nondeterministic path may involve only polynomially many evaluations of f while the global outcome $f'(x)$ depends on exponentially many evaluations. To implement this idea, we exploit the specific structure of the combining function C. Namely, we (like O'Donnell) use the Tribes function of Ben-Or and Linial [BL90], which is a monotone DNF with clauses of size $O(\log k)$. Thus, the

nondeterministic algorithm for f' can simply guess a satisfied clause and (nondeterministically) evaluate f on the $O(\log k)$ corresponding inputs.

4.1.4 Other Results

We also present some complementary negative results:

- We show that the assumption that the original hard function is balanced is necessary, in the sense that no monotone "black-box" hardness amplification can amplify unbalanced functions of unknown bias (or even improve their bias).[2]
- We show that our use of nondeterminism is necessary, in the sense that any "black-box" hardness amplification in which each evaluation of f' is a monotone function of at most k evaluations of f can amplify hardness to at most $1/2-1/k$.

Informally, a "black-box" hardness amplification is one in which the construction of the amplified function f' from f only utilizes f as an oracle and is well-defined for any function f (regardless of whether or not it is in **NP**). Moreover, the correctness of the construction is proved by a generic reduction that converts any *oracle* A (regardless of its circuit size) that computes f' well on average (e.g., with probability $1/2+\epsilon$ over random choice of input) into one that computes f much better on average (e.g., with probability $1-o(1)$ over random input). (A formal definition is given in Section 4.7.1.) We note that most results on hardness amplification against circuits, including ours, are black-box (though there have been some recent results using non-black-box techniques in hardness amplification against *uniform* algorithms; see [IW01, TV02]).

Our framework also gives a new proof of the hardness amplification by Impagliazzo and Wigderson [IW97]. Our proof is simpler and in particular its analysis does not employ the Goldreich–Levin [GL89] step.

4.1.5 Organization

The rest of the chapter is organized as follows. In Section 4.2, we discuss some preliminaries. In Section 4.3, we review existing results on hardness amplification in **NP**. In Section 4.4, we present our main results and new techniques. In Section 4.5 we treat the details of the proof of our main theorem. In Section 4.6 we show how we could amplify to $1/2-1/2^{\Omega(n)}$ given an improvement in the pseudorandom generator we use, and we also give a new proof of the hardness amplification by Impagliazzo and Wigderson [IW97]. In Section 4.7 we discuss some limitations of monotone hardness amplification; in particular we show a sense in which the hypothesis that the starting function be balanced is necessary, and also that the use of nondeterminism is necessary.

[2] We note that there do exist balanced **NP**-complete problems, as observed by Barak [For03], but this has no direct implication for us because we are studying the average-case complexity of **NP**.

4.2 Preliminaries

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

Definition 4.2.1. *Let X and Y be two random variables taking values over the same set S. Then the* statistical difference *between X and Y, is*

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

We view probabilistic functions as functions of two inputs, e.g. $\mathrm{h}(x;r)$, the first being the input to the function and the second being the randomness. (Deterministic functions may be thought of as probabilistic functions that ignore the randomness.) For notational convenience, we will often omit the second input to a probabilistic function, e.g. writing $\mathrm{h}(x)$ instead of $\mathrm{h}(x;r)$, in which case we view $\mathrm{h}(x)$ as the random variable $\mathrm{h}(x;U_{|r|})$.

Definition 4.2.2. *The* bias *of a* 0-1 *random variable X is*

$$\mathrm{Bias}\,[X] \stackrel{\text{def}}{=} \Big| \Pr[X=0] - \Pr[X=1] \Big| = 2 \cdot \Delta(X, U_1).$$

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

$$\mathrm{Bias}\,[f] \stackrel{\text{def}}{=} \Big| \Pr[f(U_n)=0] - \Pr[f(U_n)=1] \Big|,$$

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

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

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

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

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

Lemma 4.2.3 ([Yao82])**.** *Let $\mathrm{h} : \{0,1\}^n \to \{0,1\}$ be any probabilistic function. If the distributions $U_n \cdot \mathrm{h}(U_n)$ and $U_n \cdot U_1$ are ϵ-indistinguishable for size s then* h *is $(1/2-\epsilon)$-hard for size $s - O(1)$. Conversely, if* h *is $(1/2-\epsilon)$-hard for size s then the distributions $U_n \cdot \mathrm{h}(U_n)$ and $U_n \cdot U_1$ are ϵ-indistinguishable for size $s - O(1)$.*

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

4.3 Overview of Previous Hardness Amplification in NP

In this section we review the essential components of existing results on hardness amplification in **NP**. We then discuss the limitations of these techniques. By the end of this section, we will have sketched the main result of O'Donnell [O'D04], following the approach of Trevisan [Tre03]. We outline this result in a way that will facilitate the presentation of our results in subsequent sections.

Let $f : \{0,1\}^n \to \{0,1\}$ be an average-case hard function, and let $\text{C} : \{0,1\}^k \to \{0,1\}$ be any function. In [O'D04], O'Donnell studies the hardness of functions of the form

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

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

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

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

O'Donnell characterizes the hardness of $\text{C} \circ f^{\otimes k}$ in terms of a combinatorial property of the combining function C, called its *expected bias* (which we define later).

We will now review the key steps in establishing this characterization and O'Donnell's final amplification theorem.

Step 1: Impagliazzo's hardcore sets. An important tool for establishing this connection is the hardcore set lemma of Impagliazzo [Imp95a], which allows us to pass from computational hardness to information-theoretic hardness.

Definition 4.3.1. *We say that a (probabilistic) function* $\text{g} : \{0,1\}^n \to \{0,1\}$ *is* δ-random *if* g *is balanced and there exists a subset* $H \subseteq \{0,1\}^n$ *with* $|H| = 2\delta 2^n$ *such that* $\text{g}(x) = U_1$ *(i.e. a coin flip) for* $x \in H$ *and* $\text{g}(x)$ *is deterministic for* $x \notin H$.

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

The following version of Impagliazzo hardcore set lemma says that any balanced δ-hard function $f : \{0,1\}^n \to \{0,1\}$ has a hardcore set $H \subseteq \{0,1\}^n$ of density $\approx 2\delta$ such that f is very hard-on-average on H. Thus, f looks like a δ-random function to small circuits (cf. Lemma 4.2.3). (Following subsequent works, our formulation of Impagliazzo's lemma differs from the original one in several respects.)

Lemma 4.3.2 ([Imp95a, KS03, STV01, O'D04])**.** *For any function $f : \{0,1\}^n \to \{0,1\}$ that is balanced and δ-hard for size s, there exists a δ'-random function $\mathrm{g} : \{0,1\}^n \to \{0,1\}$ such that $X \cdot f(X)$ and $X \cdot g(X)$ are ϵ-indistinguishable for size $\Omega(s\epsilon^2/\log(1/\delta))$, with $\delta/2 \leq \delta' \leq \delta$, where $X \equiv U_n$.*

In particular, by a standard hybrid argument (see, e.g., [Gol01]),

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

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

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

Definition 4.3.3. *Let $\mathrm{h} : \{0,1\}^n \to \{0,1\}$ be any probabilistic function. We define the* expected bias *of* h *by*

$$\mathrm{ExpBias}\left[\mathrm{h}\right] \stackrel{\mathrm{def}}{=} \mathop{\mathrm{E}}_{x \leftarrow U_n}\left[\,\mathrm{Bias}\left[\mathrm{h}(x)\right]\,\right],$$

where $\mathrm{Bias}\left[\mathrm{h}(x)\right]$ *is taken over the coin tosses of* h.

It turns out that for any function $\mathrm{C} : \{0,1\}^k \to \{0,1\}$ and any δ-random g, the quantity $\mathrm{ExpBias}\left[\mathrm{C} \circ \mathrm{g}^{\otimes k}\right]$ does not depend on the particular choice of the δ-random function g; indeed, it turns out to equal the quantity that O'Donnell [O'D04] calls the "expected bias of C with respect to noise 2δ" and denotes by $\mathrm{ExpBias}_{2\delta}(\mathrm{C})$ in [O'D04]. However, the more general notation we use will be useful in presenting our improvements.

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

Lemma 4.3.4. *For any probabilistic function $\mathrm{h} : \{0,1\}^n \to \{0,1\}$,*

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

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

In particular, for any circuit C (regardless of its size) we have $\left|\Pr[C(U_n \cdot \mathrm{h}(U_n)) = 1] - \Pr[C(U_n \cdot U_1) = 1]\right| \leq \mathrm{ExpBias}\left[\mathrm{h}\right]/2$, and thus by Lemma 4.2.3, h is $(1/2 - \mathrm{ExpBias}\left[\mathrm{h}\right]/2)$-hard for circuits of any size.

Now we characterize the hardness of $\mathrm{C} \circ f^{\otimes k}$ in terms of expected bias. Specifically, by taking, say, $\epsilon = 1/s^{1/3}$ in Lemma 4.3.2 and using Lemmas 4.2.3 and 4.3.4, one can prove the following (we defer the details until the proof of the more general Lemma 4.5.2).

Lemma 4.3.5 ([O'D04])**.** *Let $f : \{0,1\}^n \to \{0,1\}$ be balanced and δ-hard for size s, and let $\mathrm{C} : \{0,1\}^k \to \{0,1\}$ be any function. Then there exists a δ'-random function $\mathrm{g} : \{0,1\}^n \to \{0,1\}$, with $\delta/2 \leq \delta' \leq \delta$, such that $\mathrm{C} \circ f^{\otimes k} : (\{0,1\}^n)^k \to \{0,1\}$ has hardness*

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

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

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

Step 3: Noise Stability Unfortunately, it is often difficult to analyze the expected bias directly. However, the expected bias is closely related to the *noise stability*, a quantity that is more amenable to analysis and well-studied (see, e.g., [O'D04], [MO03]). The noise stability of a function is (up to normalization) the probability that the value of the function is the same on two correlated inputs x and $x + \eta$, where x is a random input and η a random vector of noise.

Definition 4.3.6. *The noise stability of C with respect to noise δ, denoted $\mathrm{NoiseStab}_\delta[\mathrm{C}]$, is defined by*

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

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

The following lemma from [O'D04] bounds the expected bias of $\mathrm{C} \circ g^{\otimes k}$ (and hence the hardness in Lemma 4.3.5) in terms of the noise stability of C.

Lemma 4.3.7. *Let $\mathrm{g} : \{0,1\}^n \to \{0,1\}$ be δ-random. Then*

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

Combining this with Lemma 4.3.5, we find that the hardness of $\mathrm{C} \circ f^{\otimes k}$ is at least (roughly) $1/2 - \sqrt{\mathrm{NoiseStab}_\delta[\mathrm{C}]}/2$. The next step is to exhibit a combining function C with a small noise stability (to ensure that the hardness of $\mathrm{C} \circ f^{\otimes k}$ is as close to $1/2$ as possible). The following is shown in [O'D04].

Lemma 4.3.8 ([O'D04])**.** *For all $\delta > 0$, there exists a $k = \mathrm{poly}(1/\delta)$ and a polynomial-time computable monotone function $\mathrm{C} : \{0,1\}^k \to \{0,1\}$ with $\mathrm{NoiseStab}_\delta[\mathrm{C}] \leq 1/k^{\Omega(1)}$.*

Finally, by combining Lemmas 4.3.5, 4.3.7 and 4.3.8, we obtain the following weaker version of O'Donnell's hardness amplification within **NP**. (While in the introduction we mentioned a stronger version of O'Donnell's result, that amplifies up to hardness $1/2 - 1/m^{1/2-\alpha}$ for every constant $\alpha > 0$, the following version will suffice as a starting point for our work. The loss in the amplification in this version comes from the fact that we did not specify the constants in Lemma 4.3.8.)

Theorem 4.3.9 ([O'D04])**.** *If there is a balanced function $f : \{0,1\}^n \to \{0,1\}$ in* **NP** *that is $1/\mathrm{poly}(n)$-hard for size $s(n)$, then there is a function $f' : \{0,1\}^m \to \{0,1\}$ in* **NP** *that is $(1/2 - 1/m^{\Omega(1)})$-hard for size $s(m^{\Omega(1)})^{\Omega(1)}$.*

Limitations of Direct Product Constructions.
O'Donnell also showed that Theorem 4.3.9 is essentially the best result that one can obtain using the techniques that we have described thus far. He showed that for all monotone combining functions C there is a δ-hard function f such that the hardness of $\mathrm{C} \circ f^{\otimes k}$ is not much better than $1/2 - \mathrm{NoiseStab}_\delta[\mathrm{C}]/2$ (assuming that C is easily computable). This is problematic because the noise stability of monotone functions cannot become too small. Specifically, by combining a result from [KKL88] with a Fourier characterization of noise stability, O'Donnell [O'D04] proves the following theorem.

Theorem 4.3.10 ([KKL88, O'D04])**.** *For every monotone function $\mathrm{C} : \{0,1\}^k \to \{0,1\}$ and every $\delta > 0$,*

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

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

4.4 Main Theorem and Overview

In this chapter, we obtain the following improvement upon Theorem 4.3.9.

Theorem 4.4.1 (Main Theorem)**.** *If there is a balanced function $f : \{0,1\}^n \to \{0,1\}$ in* **NP** *that is $1/\operatorname{poly}(n)$-hard for size $s(n)$, then there is a function $f' : \{0,1\}^m \to \{0,1\}$ in* **NP** *that is $(1/2 - 1/s(\sqrt{m})^{\Omega(1)})$-hard for size $s(\sqrt{m})^{\Omega(1)}$.*[3]

We also show that the assumption that we start with a *balanced* function f is essential for a large class of hardness amplifications. Specifically, we show (Section 4.7.1) that no monotone black-box hardness amplification can amplify the hardness of functions whose bias is unknown. Most hardness amplifications, including the one in this chapter, are black-box.

We now elaborate on the two main techniques that allow us to prove Theorem 4.4.1. As explained in the introduction, these two techniques are derandomization and nondeterminism.

4.4.1 Derandomization

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

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

Definition 4.4.2. *A* generator *is a function $G : \{0,1\}^l \to (\{0,1\}^n)^k$. We call l the* seed length *of G, and we often write $G(\sigma) = X_1 \cdots X_k$, with each $X_i \in \{0,1\}^n$. G is* explicitly computable *if given σ and $1 \le i \le k$, we can compute X_i in time $\operatorname{poly}(l, \log k)$, where $G(\sigma) = X_1 \cdots X_k$.*

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

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

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

1. **G is indistinguishability-preserving:** Analogously to Lemma 4.3.5, the generator G should be such that the computational hardness of $(\mathrm{C} \circ f^{\otimes k}) \circ G$ is at least the information-theoretic hardness of $(\mathrm{C} \circ \mathrm{g}^{\otimes k}) \circ G$ for some δ-random

[3] A comment is in order about the input lengths for which f' is hard. As it turns out, the hardness of f' on inputs of length m is related to the hardness of the original function f on inputs of length $\Theta(\sqrt{m})$. Thus if f is hard for all sufficiently large input lengths, then so is f'. Alternatively, if f is only hard infinitely often, then we may still conclude that f' is hard infinitely often.

function g – that is, at least $1/2 - \text{ExpBias}\left[(\mathrm{C} \circ \mathrm{g}^{\otimes k}) \circ G\right]$. We will see that this can be achieved provided that G is *indistinguishability-preserving*; that is (analogously to the last part of Lemma 4.3.2),

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

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

2. **G fools the expected bias:** G should be such that for any δ-random g, $\text{ExpBias}\left[(\mathrm{C} \circ \mathrm{g}^{\otimes k}) \circ G\right]$ is approximately $\text{ExpBias}\left[\mathrm{C} \circ \mathrm{g}^{\otimes k}\right]$, and thus we have, by Lemma 4.3.7:

$$\text{ExpBias}\left[(\mathrm{C} \circ \mathrm{g}^{\otimes k}) \circ G\right] \leq \sqrt{\text{NoiseStab}_\delta[\mathrm{C}] + \epsilon}, \tag{4.2}$$

for a suitably small ϵ. Actually, we will not show that G fools the expected bias directly but instead will work with a related quantity (the expected collision probability), which will still suffice to show Inequality (4.2).

Informally, the effect of the two above requirements on the generator G is that the hardness of $(\mathrm{C} \circ f^{\otimes k}) \circ G$ is roughly the hardness of $\mathrm{C} \circ f^{\otimes k}$, while the input length is dramatically reduced from nk to l (the seed length of G). More precisely, as illustrated in Figure 1, the first requirement allows us to relate the hardness of $(\mathrm{C} \circ f^{\otimes k}) \circ G$ to the information-theoretic hardness of $(\mathrm{C} \circ g^{\otimes k}) \circ G$ (where g is a δ-random function); the second allows us to relate this information-theoretic hardness to the noise stability of the combining function C. In particular, if we employ the combining function from Lemma 4.3.8, we obtain hardness $1/2 - 1/k^{\Omega(1)}$. Thus, by choosing $k \gg l$, we bypass the barrier discussed at the end of the previous section.

Now we briefly describe how the above requirements on G are met. The first requirement is achieved through a generator that outputs *combinatorial designs*. This construction is essentially from Nisan and Wigderson [Nis91, NW94] and has been used in many places, e.g. [IW97, STV01].

The second requirement is achieved as follows. We show that if G is pseudorandom against space-bounded algorithms and the combining function C is computable in small space (with one-way access to its input), then Inequality (4.2) holds. We then use Nisan's *unconditional* pseudorandom generator against space-bounded algorithms [Nis92], and show that combining functions with low noise stability can in fact be computed in small space.[4] Note that we only use the pseudorandomness of the generator G to relate the expected bias with respect to G to a combinatorial property

[4]The same approach also works using the unconditional pseudorandom generator against constant-depth circuits of [Nis91] and showing that the combining function is computable by a small constant-depth circuit; however, the space generator gives us slightly better parameters.

of the combining function C. In particular, it is *not* used to fool the circuits trying to compute the hard function. This is what allows us to use an unconditional generator against a relatively weak model of computation.

Our final generator, Γ, is the generator obtained by XORing a generator that is indistinguishability-preserving and a generator that fools the expected bias, yielding a generator that has both properties. The approach of XORing two generators in this way appeared in [IW97], and was subsequently used in [STV01].

4.4.2 Using Nondeterminism

The derandomization described above gives hardness amplification up to $1/2 - 1/n^c$ *for any* constant c. This already improves upon the best previous result, namely Theorem 4.3.9. However, to go beyond this new techniques are required. The problem is that if we want C to be computable in time $\mathrm{poly}(n)$, we must take $k = \mathrm{poly}(n)$ and thus we amplify to at most $1/2 - 1/k = 1/2 - 1/\mathrm{poly}(n)$.

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

Conversely, in Section 4.7.2 we show that any non-adaptive monotone black-box hardness amplification that amplifies to hardness $1/2 - 1/n^{\omega(1)}$ cannot be computed in **P**, i.e. the use of nondeterminism is essential.

We proceed by discussing the details of the derandomization (Sections 4.5.1, 4.5.2 and 4.5.3) and the use of nondeterminism (Section 4.5.4). The results obtained in these sections are summarized in Table 1. For clarity of exposition, we focus on the case where the original hard function f is balanced and is 1/3-hard. Hardness amplification from hardness $1/\mathrm{poly}(n)$ is discussed in Section 4.5.5, and hardness amplification of unbalanced functions is discussed in Section 4.7.1.

4.5 Proof of Main Theorem

In this section we prove our main theorem (i.e., Theorem 4.4.1).

4.5.1 Preserving Indistinguishability

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

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

Table 4.1: Hardness Amplification within **NP**.

Functions : $\{0,1\}^n \to \{0,1\}$		
Amplification up to	**Technique**	**Reference**
$1/2 - 1/\sqrt{n}$	Direct Product	[O'D04]
$1/2 - 1/n^c$, for *every* c	Derandomized Direct Product	Theorem 4.5.8
$1/2 - 1/2^{\Omega(\sqrt{n})}$	Derandomized Direct Product & Nondeterminism	Theorem 4.5.13

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

Definition 4.5.1. *A generator* $G : \{0,1\}^l \to (\{0,1\}^n)^k$ *is said to be* indistinguishability-preserving for size t *if for all (possibly probabilistic) functions* $f_1, \ldots, f_k, g_1, \ldots, g_k$ *the following holds:*

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

$$U_n \cdot f_i(U_n) \text{ and } U_n \cdot \mathrm{g}_i(U_n)$$

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

$$\sigma \cdot f_1(X_1) \cdots f_k(X_k) \text{ and } \sigma \cdot \mathrm{g}_1(X_1) \cdots \mathrm{g}_k(X_k)$$

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

The fact that in the above definition we consider k f_i's and k g_i's implies that an $indistinguishability - preserving$ generator stays $indistinguishability - preserving$ when XORed with any other generator (cf. the proof of Item 1 in Lemma 4.5.12). We will use this property in the proof of our main result.

Lemma 4.5.2. *Let* $f : \{0,1\}^n \to \{0,1\}$ *be* δ*-hard for size* s*, let* $G : \{0,1\}^l \to (\{0,1\}^n)^k$ *be a generator that is indistinguishability-preserving for size* t *and let* C :

$\{0,1\}^k \to \{0,1\}$ *be any function. Then there exists a* δ'*-random* g*, with* $\delta/2 \leq \delta' \leq \delta$ *such that the function* $(\mathrm{C} \circ f^{\otimes k}) \circ G : \{0,1\}^l \to \{0,1\}$ *has hardness*

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

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

Proof. By Lemma 4.3.2, there exists a δ'-random function g with $\delta/2 \leq \delta' \leq \delta$, such that $U_n \cdot f(U_n)$ and $U_n \cdot \mathrm{g}(U_n)$ are ϵ-indistinguishable for size $\Omega(s\epsilon^2/\log(1/\delta))$. Since G is an indistinguishability-preserving for size t by assumption, this implies that

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

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

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

(i.e., $\sigma \cdot (\mathrm{C} \circ f^{\otimes k}) \circ G(\sigma)$ and $\sigma \cdot (\mathrm{C} \circ \mathrm{g}^{\otimes k}) \circ G(\sigma)$) are $k\epsilon$-indistinguishable for size $\Omega(s\epsilon^2/\log(1/\delta)) - t - \mathrm{size}(\mathrm{C})$. By Lemma 4.3.4,

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

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

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

are $(\mathrm{ExpBias}\left[(\mathrm{C} \circ \mathrm{g}^{\otimes k}) \circ G\right]/2 + k\epsilon)$-indistinguishable for size $\Omega(s\epsilon^2/\log(1/\delta)) - t - \mathrm{size}(\mathrm{C})$. The result follows by setting $\epsilon = 1/s^{1/3}$ and applying Lemma 4.2.3. □

In particular, we note that the *identity generator* $G : \{0,1\}^{nk} \to (\{0,1\}^n)^k$, i.e. $G(x) = x$, is indistinguishability-preserving for size 0 (by a hybrid argument, see, e.g., [Gol01]), and thus Lemma 4.3.5 is a corollary of Lemma 4.5.2. However, the identity generator has seed-length nk and is therefore a very poor pseudorandom generator. Fortunately, there are indistinguishability-preserving pseudorandom generators with much shorter seeds which will allow us to use Lemma 4.5.2 to obtain much stronger hardness amplifications.

Lemma 4.5.3. *There is a constant* c *such that for every* $n \geq 2$ *and every* $k = k(n)$ *there is an explicitly computable generator* $NW_k : \{0,1\}^l \to (\{0,1\}^n)^k$ *with seed length* $l = c \cdot n^2$ *that is indistinguishability-preserving for size* k^2*.*

Proof. The generator is the main component of the generator by Nisan and Nisan and Wigderson [Nis91, NW94], and is based on combinatorial designs. Specifically, we let $S_1, \ldots, S_k \subseteq [l]$ be an explicit family of sets such that $|S_i| = n$ for all i, and $|S_i \cap S_j| \leq \log k$ for all $i \neq j$. Nisan [Nis91] gives an explicit construction of such sets with $l = O(n^2)$. Then the generator $NW_k : \{0,1\}^l \to (\{0,1\}^n)^k$ is defined by

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

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

The proof that this generator is indistinguishability preserving for size k^2 follows the arguments in [NW94, STV01]. For completeness, we sketch the proof here. Suppose that we have a circuit C of size $s - k^2$ distinguishing the distributions $\sigma \cdot f_1(\sigma|_{S_1}) \cdots f_k(\sigma|_{S_k})$ and $\sigma \cdot g_1(\sigma|_{S_1}) \cdots g_k(\sigma|_{S_k})$ with advantage greater than $k \cdot \epsilon$. For $i = 0, \ldots, k$, let H_i be the hybrid distribution

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

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

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

and

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

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

4.5.2 Fooling the Expected Bias

In this subsection we prove a derandomized version of Lemma 4.3.7. Informally, we show that if C is computable in a restricted model of computation and G "fools" that restricted model of computation, then for any δ-random function g:

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

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

Definition 4.5.4. *A (probabilistic, read-once, oblivious)* branching program *of* size s *with* block-size n *is a finite state machine with s states, over the alphabet $\{0,1\}^n$ (with a fixed start state, and an arbitrary number of accepting states). Each edge is labelled with a symbol in $\{0,1\}^n$. For every state a and symbol $\alpha \in \{0,1\}^n$, the edges leaving a and labelled with α are assigned a probability distribution. Then computation proceeds as follows. The input is read sequentially, one block of n bits at a time. If the machine is in state a and it reads α, then it chooses an edge leaving a and labelled with α according to its probability, and moves along it.*

Now we formally define pseudorandom generators against branching programs.

Definition 4.5.5. *A generator $G : \{0,1\}^l \to (\{0,1\}^n)^k$ is* ϵ-pseudorandom against branching programs of size s and block-size n *if for every branching program* B *of size s and block-size n:*

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

In [Nis92], Nisan builds an unconditional pseudorandom generator against branching programs. Its parameters (specialized for our purposes) are given in the following theorem.

Theorem 4.5.6 ([Nis92])**.** *For every n and $k \leq 2^n$, there exists a generator*

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

such that:

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

Note that Nisan [Nis92] does not mention *probabilistic* branching programs. However, if there is a probabilistic branching program distinguishing the output of the generator from uniform, then by a fixing of the coin tosses of the branching program there is a *determinisitic* branching program that distinguishes the output of the generator from uniform.

We now state the derandomized version of Lemma 4.3.7.

Lemma 4.5.7. *Let*

- $\mathrm{g} : \{0,1\}^n \to \{0,1\}$ *be a δ-random function,*
- $\mathrm{C} : \{0,1\}^k \to \{0,1\}$ *be computable by a branching program of size t and block-size* 1*,*

- $G : \{0,1\}^l \to (\{0,1\}^n)^k$ *be* $\epsilon/2$*-pseudorandom against branching programs of size* t^2 *and block-size* n*.*

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

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

$$\text{ExpCP}[\text{h}] \stackrel{\text{def}}{=} \mathop{\text{E}}_{x}\left[2 \cdot \Pr_{r,r'}[\text{h}(x; r) = \text{h}(x; r')] - 1\right].$$

The same reasoning that proves Lemma 4.3.7 also shows that for every probabilistic boolean function h:

$$\text{ExpBias}\,[\text{h}] \leq \sqrt{\text{ExpCP}[\text{h}]}. \tag{4.3}$$

More specifically, Inequality (4.3) holds because

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

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

$$\text{ExpCP}[\text{h}] = \text{NoiseStab}_\delta[\text{C}]. \tag{4.4}$$

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

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

To do this, we first note that, using the branching program for C, we can build a probabilistic branching program of size t with block-size n which computes $\text{C} \circ \text{g}^{\otimes k}$:

The states of the branching program are the same as those of the branching program for C, and we define the transitions as follows. Upon reading symbol $\alpha \in \{0,1\}^n$ in state s, if $g(\alpha) = 0$ (resp. $g(\alpha) = 1$), we deterministically go to the state given by the 0-transition (resp., 1-transition) of C from state s, and if $g(\alpha)$ is a coin flip, then we put equal probability on these two transitions.

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

$$\begin{aligned}
&\left|\mathrm{ExpCP}[(\mathrm{C}\circ \mathrm{g}^{\otimes k})\circ G] - \mathrm{NoiseStab}_\delta[\mathrm{C}]\right| \\
&= \left|\mathrm{ExpCP}[(\mathrm{C}\circ \mathrm{g}^{\otimes k})\circ G] - \mathrm{ExpCP}[\mathrm{C}\circ \mathrm{g}^{\otimes k}]\right| \quad \text{(by (4.4))} \\
&= 2\cdot\left|\Pr[M\circ G(U_l)=1] - \Pr[M(U_{n\cdot k})=1]\right| \\
&\leq \epsilon. \quad \text{(by pseudorandomness of } G)
\end{aligned}$$

The lemma follows combining this with Equation (4.3). □

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

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

Theorem 4.5.8. *If there is a balanced function $f : \{0,1\}^n \to \{0,1\}$ in* **NP** *that is $(1/3)$-hard for size $s(n) \geq n^{\omega(1)}$, then for every $c > 0$ there is a function $f' : \{0,1\}^m \to \{0,1\}$ in* **NP** *that is $(1/2 - 1/m^c)$-hard for size $(s(\sqrt{m}))^{\Omega(1)}$.*

To amplify we use the Tribes function of Ben-Or and Linial [BL90], a monotone read-once DNF.

Definition 4.5.9. *The* Tribes *function on k bits is:*

$$\mathrm{Tribes}_k(x_1,\ldots,x_k) \stackrel{\text{def}}{=}$$

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

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

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

Lemma 4.5.10 ([O'D04, MO03])**.** *For every constant $\delta > 0$,*

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

A key step in our result is that Tribes_k is easily computable by a branching program of size $O(k)$, and therefore we can use Lemma 4.5.7 to fool its expected bias.

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

Definition 4.5.11. *Given n and $k \leq 2^n$, define the generator $\Gamma_k : \{0,1\}^m \to (\{0,1\}^n)^k$ as follows:*

$$\Gamma_k(x,y) \stackrel{\text{def}}{=} NW_k(x) \oplus N_k(y),$$

where $\oplus$ denotes bitwise XOR.

We recall the properties of Γ we are interested in:

Lemma 4.5.12. *The following hold:*

1. *Γ_k is indistinguishability-preserving for size k^2.*
2. *Γ_k is 2^{-n}-pseudorandom against branching programs of size 2^n and block-size n.*
3. *Γ_k has seed length $m = O(n^2)$.*
4. *Γ_k is explicitly computable (see Definition 4.4.2 for the definition of explicit).*

Proof. Item (1) follows from Lemma 4.5.3 and the fact that an indistinguishability-preserving generator XORed with any fixed string is still indistinguishability-preserving . More specifically, suppose, for the sake of contradiction, that $\Gamma_k(x,y) = NW_k(x) \oplus N_k(y)$ is not indistinguishability-preserving . Then there are functions $f_1, \ldots, f_k$ and $g_1, \ldots, g_k$ such that for every i the distributions $U_n \cdot f_i(U_n)$ and $U_n \cdot g_i(U_n)$ are indistinguishable, yet the distributions

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

and

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

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

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

and

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

are distinguishable (for random x), where $f'_i(z) = f_i(z \oplus N_i(\tilde{y}))$, $g'_i(z) = g_i(z \oplus N_i(\tilde{y}))$. (Note that we hardwire the fixed part of the seed $\tilde{y}$ in the distinguisher.) Now observe that the indistinguishability of $U_n \cdot f_i(U_n)$ and $U_n \cdot g_i(U_n)$ implies the indistinguishability of $U_n \cdot f'_i(U_n)$ and $U_n \cdot g'_i(U_n)$, because the mapping $T(u \cdot v) = (u \oplus N_i(\tilde{y})) \cdot v$ transforms the latter pair of distributions to the former. (There is

no loss in the circuit size assuming that circuits have input gates for both the input variables and their negations.) But this is a contradiction because NW is indistinguishability-preserving .

Item (2) follow from Theorem 4.5.6 and the fact that XORing with any fixed string (in particular, $NW_k(x)$ for any x) preserves pseudorandomness against branching programs.

Item (3) is an immediate consequence of the seed lengths of NW_k (Lemma 4.5.3) and N_k (Theorem 4.5.6).

Item (4) follows from the fact that NW_k is explicit (Lemma 4.5.3) and N_k is explicit (Theorem 4.5.6). □

Proof of Theorem 4.5.8. Given $f : \{0,1\}^n \to \{0,1\}$ that is δ-hard for size $s(n)$ (for $\delta = 1/3$) and a constant c, let $k = n^{c'}$ for $c' = O(c)$ to be determined later. Consider the function $f' : \{0,1\}^m \to \{0,1\}$ defined by

$$f' \stackrel{\text{def}}{=} (\text{Tribes}_k \circ f^{\otimes k}) \circ \Gamma_k.$$

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

We now analyze the hardness of f'. Since Γ_k is indistinguishability-preserving for size k^2 by Lemma 4.5.12, Lemma 4.5.2 implies that there is a δ'-random function g (for $\delta/2 \le \delta' \le \delta$) such that f' has hardness

$$\frac{1}{2} - \frac{\text{ExpBias}\left[(\text{Tribes}_k \circ \mathrm{g}^{\otimes k}) \circ \Gamma_k\right]}{2} - \frac{k}{s(n)^{1/3}} \tag{4.5}$$

for circuits of size $\Omega\left(s(n)^{1/3}\right) - k^2 - \text{size}(\text{Tribes}_k)$. Next we bound the hardness. By Lemma 4.5.12, we know that Γ_k is 2^{-n}-pseudorandom against branching programs of size 2^n and block-size n. In particular, since $k = \text{poly}(n)$, Γ_k is $1/k$-pseudorandom against branching programs of size $9k$ and block-size n. Since, as we noted before, Tribes_k is easily computable by a branching program of size $O(k)$, we can apply Lemma 4.5.7 (noting that $O(k)^2 = \text{poly}(n) \ll 2^n$) in order to bound $\text{ExpBias}\left[(\text{Tribes}_k \circ \mathrm{g}^{\otimes k}) \circ \Gamma_k\right]$ by $\sqrt{\text{NoiseStab}_{\delta'}[\text{Tribes}_k] + 2/k}$. And the noise stability inside the square root is at most $1/k^{\Omega(1)}$ by Lemma 4.5.10. Since $k = \text{poly}(n)$ and $s(n) = n^{\omega(1)}$, the $k/s^{1/3}$ term in the hardness (4.5) is negligible and we obtain hardness at least $1/2 - 1/k^{\Omega(1)}$.

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

To conclude, note that f' has input length $m = n^2$ by Lemma 4.5.12. The result then follows by an appropriate choice of $c' = O(c)$. □

4.5.4 Using Nondeterminism

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

Theorem 4.5.13. *If there is a balanced function $f : \{0,1\}^n \to \{0,1\}$ in* **NP** *that is $(1/3)$-hard for size $s(n)$, then there is a function $f' : \{0,1\}^m \to \{0,1\}$ in* **NP** *that is $(1/2 - 1/s(\sqrt{m})^{\Omega(1)})$-hard for size $s(\sqrt{m})^{\Omega(1)}$.*

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

Lemma 4.5.14. *Let $f : \{0,1\}^n \to \{0,1\}$ be in* **NP**, *and let $G_k : \{0,1\}^l \to (\{0,1\}^n)^k$ be any explicitly computable generator (see Definition 4.4.2) with $l \geq n$. Then the function $f' \stackrel{\text{def}}{=} (\text{Tribes}_k \circ f^{\otimes k}) \circ G_k$ is computable in* **NP** *for every $k = k(n) \leq 2^n$.*

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

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

Now the proof of Theorem 4.5.13 proceeds along the same lines as the proof of Theorem 4.5.8, setting $k \stackrel{\text{def}}{=} s(n)^{\Omega(1)}$.

4.5.5 Amplifying from Hardness $1/\text{poly}$

Our amplification from hardness $\Omega(1)$ to $1/2 - \epsilon$ (Theorem 4.5.8) can be combined with O'Donnell's amplification from hardness $1/\text{poly}$ to hardness $\Omega(1)$ to obtain an amplification from $1/\text{poly}$ to $1/2 - \epsilon$. However, since O'Donnell's construction blows up the input length polynomially, we would only obtain $\epsilon = 1/s(n^{\Omega(1)})$ (where the hidden constant depends on the initial polynomial hardness) rather than $\epsilon = 1/s(\sqrt{n})^{\Omega(1)}$ (as in Theorem 4.5.8). Thus we show here how to amplify directly from $1/\text{poly}$ to $1/2 - \epsilon$ using our approach. For this we need a combining function C that is more involved than the Tribes function. The properties of C that are needed in the proof of Theorem 4.4.1 are captured by the following lemma.

Lemma 4.5.15. *For every $\delta(n) = 1/n^{O(1)}$, there is a sequence of functions $C_k : \{0,1\}^k \to \{0,1\}$, such that for every $k = k(n)$ with $n^{\omega(1)} \leq k \leq 2^n$, the following hold:*

1. $\text{NoiseStab}_\delta[C_k] \leq 1/k^{\Omega(1)}$.

2. *For every* $f : \{0,1\}^n \to \{0,1\}$ *in* **NP** *and every explicitly computable generator (see Definition 4.4.2)* $G_k : \{0,1\}^l \to (\{0,1\}^n)^k$ *with* $l \geq n$*, the function* $(C_k \circ f^{\otimes k}) \circ G_k$ *is in* **NP**.

3. C_k *can be computed by a branching program of size* $\mathrm{poly}(n) \cdot k$*, and also by a circuit of size* $\mathrm{poly}(n) \cdot k$.

Before proving Lemma 4.5.15, let us see how it can be used to prove our main theorem.

Theorem 4.5.16 (Thm. 4.4.1, restated)**.** *If there is a balanced function* $f : \{0,1\}^n \to \{0,1\}$ *in* **NP** *that is* $1/\mathrm{poly}(n)$*-hard for size* $s(n)$*, then there is a function* $f' : \{0,1\}^m \to \{0,1\}$ *in* **NP** *that is* $(1/2 - 1/s(\sqrt{m})^{\Omega(1)})$*-hard for size* $s(\sqrt{m})^{\Omega(1)}$.

Proof. Let $f : \{0,1\}^n \to \{0,1\}$ be a balanced function in **NP** that is $\delta = \delta(n)$-hard for size $s(n)$, where $\delta \geq 1/n^{O(1)}$. Let $k = k(n) \stackrel{\text{def}}{=} s(n)^{1/7}$ and let C_k be the function guaranteed by Lemma 4.5.15. Let Γ_k be the the generator from Definition 4.5.11. Consider the function $f' : \{0,1\}^m \to \{0,1\}$ defined by $f' \stackrel{\text{def}}{=} (C_k \circ f^{\otimes k}) \circ \Gamma_k$. Note that $f' \in$ **NP** by Item 2 in Lemma 4.5.15.

We now analyze the hardness of f'. Since Γ_k is indistinguishability-preserving for size k^2 (by Lemma 4.5.12), Lemma 4.5.2 implies that there is a δ'-random function g (for $\delta/2 \leq \delta' \leq \delta$) such that f' has hardness

$$\alpha(m) = \frac{1}{2} - \frac{\mathrm{ExpBias}\left[(C_k \circ \mathrm{g}^{\otimes k}) \circ \Gamma_k\right]}{2} - \frac{k}{s(n)^{1/3}} \tag{4.6}$$

for circuits of size

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

We first bound the hardness $\alpha(m)$. By Lemma 4.5.12, we know that Γ_k is 2^{-n}-pseudorandom against branching programs of size 2^n and block-size n. Since the branching program for computing C_k has size $\mathrm{poly}(n) \cdot k$, and $(\mathrm{poly}(n) \cdot k)^2 \ll 2^n$ (by our choice of $k(n)$), we may apply Lemma 4.5.7 to bound $\mathrm{ExpBias}\left[(C_k \circ \mathrm{g}^{\otimes k}) \circ \Gamma_k\right]$ by $\sqrt{\mathrm{NoiseStab}_{\delta'}[C_k] + 2/2^n}$. This noise stability is at most $1/k^{\Omega(1)}$ by Item 1 in Lemma 4.5.15. Using the fact that $k = s(n)^{1/7}$, we have

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

We now bound the circuit size $s'(m)$. Since C_k is computable by a circuit of size $\mathrm{poly}(n) \cdot k$ (by Item 3 in Lemma 4.5.15) and $\log(1/\delta) = O(\log n)$ and $s(n) = n^{\omega(1)}$, we have

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

To conclude, we note that f' has input length $m = O(n^2)$ by Lemma 4.5.12, so $s(n) = s(\Omega(\sqrt{m})) = s(\sqrt{m})^{\Omega(1)}$, and we indeed obtain hardness $\alpha(m) = 1/2 - 1/s(\sqrt{m})^{\Omega(1)}$ for size $s'(m) = s(\sqrt{m})^{\Omega(1)}$. □

The rest of this subsection is devoted to the proof of Lemma 4.5.15. Recall that amplification from hardness $\Omega(1)$ (Theorem 4.5.8) relies on the fact that the Tribes DNF has low noise stability with respect to noise parameter $\delta = \Omega(1)$ (i.e., Lemma 4.5.10). Similarly, to amplify from hardness $1/\operatorname{poly}(n)$ we need to employ a combining function that has low noise stability with respect to noise $1/\operatorname{poly}(n)$. To this end, following [O'D04], we employ the recursive-majorities function, RMaj_r. Let Maj denote the majority function.

Definition 4.5.17. *The* RMaj_r *function on* 3^r *bits is defined recursively by:*

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

The following lemma quantifies the noise stability of RMaj_r.

Lemma 4.5.18 ([O'D04], Proposition 11)**.** *There is a constant c such that for every $\delta > 0$ and every $r \geq c \cdot \log(1/\delta)$, we have*

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

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

However, when $r = O(\log n)$, RMaj_r does not have sufficiently low noise stability to be used on its own. For this reason, we will combine RMaj with Tribes. (The same combination of RMaj and Tribes is employed by O'Donnell [O'D04], albeit for a different setting of parameters.)

Proof of Lemma 4.5.15. Given n and $\delta = \delta(n) \geq 1/n^{O(1)}$, let $r \stackrel{\text{def}}{=} c \cdot \log(1/\delta)$ for a constant c to be chosen later. Assume, without loss of generality, that r and $k/3^r$ are integers. The function $C_k : \{0,1\}^k \to \{0,1\}$ is defined as follows

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

We now prove that C_k satisfies the required properties.

1. We will use the following result from [O'D04].

   Lemma 4.5.19 ([O'D04], Proposition 8)**.** *If h is a balanced boolean function and $\varphi : \{0,1\}^\ell \to \{0,1\}$ is any boolean function, then*

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

Letting c be a sufficiently large constant (recall that $r = c \cdot \log(1/\delta)$), by Lemma 4.5.18 we have that $\mathrm{NoiseStab}_\delta[\mathrm{RMaj}_r]/2 \geq 1/2 - 1/8 \geq 3/8$. Now note that RMaj_r is balanced because taking the bitwise complement of an input x also negates the value of $\mathrm{RMaj}_r(x)$. Hence, by Lemma 4.5.19,

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

where the last two equalities use Lemma 4.5.10 and the fact that $k = n^{\omega(1)}$ and $r = O(\log n)$.

2. The proof is similar to the proof of Lemma 4.5.14. In order to compute $(\mathrm{Tribes}_{k/3^r} \circ \mathrm{RMaj}_r^{\otimes k/3^r} \circ f^{\otimes k}) \circ G_k$, we guess a clause of the $\mathrm{Tribes}_{k/3^r}$ and verify that all the RMaj_r evaluations feeding into it are satisfied (using the **NP** algorithm for f). The only additional observation is that each of the recursive majorities depends only on $3^r = \mathrm{poly}(n)$ bits of the input, and hence can be computed in time polynomial in n.

3. As noted earlier, $\mathrm{Tribes}_{k/3^r}$ is easily computable by a branching program of size $O(k)$. RMaj_r, on the other hand, can be computed by a branching program of size $\mathrm{poly}(n)$. Indeed, $\mathrm{Maj}(x_1, x_2, x_3)$ is clearly computable by a branching program of constant size c, and thus

   $$\mathrm{RMaj}_r = \mathrm{RMaj}_{r-1}\big(\mathrm{Maj}(x_1, x_2, x_3), \ldots, \mathrm{Maj}(x_{3^r-2}, x_{3^r-1}, x_{3^r})\big)$$

   can be computed by a branching program whose size is at most c times the size of RMaj_{r-1}. By induction it follows that RMaj_r can be computed in size $c^r = \mathrm{poly}(n)$.

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

□

A natural question is whether one can amplify from hardness $1/n^{\omega(1)}$. A modification of the [Vio04] negative result about hardness amplification given in [LTW05] shows that this task cannot be achieved by any black-box hardness amplification computable in the polynomial-time hierarchy (and thus in particular cannot be achieved by any black-box hardness amplification computable in **NP**). (See Definition 4.7.2 for the definition of black-box hardness amplification.)

4.6 Extensions

4.6.1 On Amplifying Hardness up to $1/2 - 1/2^{\Omega(n)}$

Even when starting from a function that is δ-hard for size $2^{\Omega(n)}$, our results (Theorem 4.4.1) only amplify hardness up to $1/2 - 1/2^{\Omega(\sqrt{n})}$ (rather than $1/2 - 1/2^{\Omega(n)}$). In this section we discuss the possibility of amplifying hardness in **NP** up to $1/2 - 1/2^{\Omega(n)}$, when starting with a function that is δ-hard for size $2^{\Omega(n)}$. The problem is that the seed length of our generator in Lemma 4.5.12 is *quadratic* in n, rather than linear. To amplify hardness up $1/2 - 1/2^{\Omega(n)}$ we need a generator (for every $k = 2^{\Omega(n)}$) with the same properties of the one in Lemma 4.5.12, but with linear seed length.

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

Lemma 4.6.1. *For every constant γ, $0 < \gamma < 1$, there is a constant c such that for every n there is an explicitly computable generator $NW'_{2^{n/c}} : \{0,1\}^l \to (\{0,1\}^n)^{2^{n/c}}$ with seed length $l = c \cdot n$ that is indistinguishability-preserving for size $2^{\gamma \cdot n}$.*

The generator is due to Nisan and Wigderson [NW94] and Impagliazzo and Wigderson [IW97]. The approach is the same as for the generator used in Lemma 4.5.3, except we now require a design consisting of $2^{\Omega(n)}$ sets of size n in a universe of size $O(n)$, with pairwise intersections of size at most $\gamma n/2$. An explicit construction of such a design is given in [GV04].[5]

Theorem 4.6.2. *Suppose that there exists an explicit generator $N'_{2^n} : \{0,1\}^l \to (\{0,1\}^n)^{2^n}$ that is 2^{-n}-pseudorandom against branching programs of size 2^n and block-size n and that has seed length $l = O(n)$. Then the following holds: If there is a balanced function $f : \{0,1\}^n \to \{0,1\}$ in* **NP** *that is $1/\operatorname{poly}(n)$-hard for size $2^{\Omega(n)}$, then there is a function $f' : \{0,1\}^m \to \{0,1\}$ in* **NP** *that is $(1/2 - 1/2^{\Omega(n)})$-hard for size $2^{\Omega(n)}$.*

A slightly more careful analysis shows that all the branching programs considered in our constructions have *width*[6] much smaller than their size (specifically, the branching program for Tribes has constant width, and the one for the function in Lemma 4.5.15, i.e. Tribes composed with RMaj, has width $\operatorname{poly}(n)$ independent of k). Thus to prove the conclusion in the statement of Theorem 4.6.2 it would suffice

[5] Alternatively, we can use the randomized algorithm described in [IW97] that computes such sets $S_1, \ldots, S_M$ with probability exponentially close to 1 using $O(n)$ random bits. This is sufficient for constructing an indistinguishability-preserving generator.

[6] The width of a branching program is the maximum, over i, of the number of states that are reachable after reading i symbols.

to exhibit an explicit pseudorandom generator that fools such restricted branching programs.

For amplifying from constant hardness, it suffices to instead have a generator fooling *constant-depth circuits* of size 2^n with seed length $O(n)$. (The generator of Nisan [Nis91] has seed length $\text{poly}(n)$.) The reason is that our proof that PRGs versus branching programs "fool" the expected bias also works for PRGs versus constant-depth circuits, provided that the combining function is computable in constant depth. The Tribes function is depth 2 by definition (but the recursive majorities RMaj is not constant-depth, and hence this would only amplify from constant hardness).

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

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

where, for example, $C_k = \text{Tribes}_k$ (when δ is constant). As in the proof of Lemma 4.3.7, in proving such a statement, it may be convenient to work instead with the (polynomially related) expected collision probability. An important property of $C_k = \text{Tribes}_k$ we used in bounding the expected bias with respect to G is that it gives expected bias $2^{-\Omega(n)}$ if G is replaced with a truly random generator (i.e. using seed length $n \cdot k$) and δ is constant. One might try to use a different monotone combining function with this property, provided it can also be evaluated in nondeterministic time $\text{poly}(n)$.

4.6.2 Impagliazzo and Wigderson's Hardness Amplification

Our framework gives a new proof of the hardness amplification by Impagliazzo & Wigderson [IW97]. Impagliazzo & Wigderson [IW97] show that if $\mathbf{E} \stackrel{\text{def}}{=} DTIME\left(2^{O(n)}\right)$ contains a function $f : \{0,1\}^n \to \{0,1\}$ that requires (in the worst-case) circuits of size $2^{\Omega(n)}$, then $\mathbf{E}$ contains a function f' that is $\left(1/2 - 1/2^{\Omega(n)}\right)$-hard for circuits of size $2^{\Omega(n)}$. The main contribution in [IW97] is amplification from constant hardness, e.g. $1/3$, to $\left(1/2 - 1/2^{\Omega(n)}\right)$ (amplification from worst-case hardness and constant hardness was essentially already established in [BFNW93, Imp95a]). The improvement over the standard Yao XOR Lemma is that the input length of the amplified function increases only by a constant factor. In this section, we sketch a simple proof of this result using the framework developed in earlier sections. While other, more recent hardness amplifications achieving the same result for $\mathbf{E}$ are known [STV01, SU00, Uma02], the original one by Impagliazzo and Wigderson is still interesting because it can be implemented in "low" complexity classes, such as the polynomial-time hierarchy, while the others cannot (due to the fact that they actually amplify from worst-case hardness [Vio04]).

The construction of [IW97] uses an *expander-walk* generator $W_k : \{0,1\}^l \to (\{0,1\}^n)^k$, which uses its seed of length $l = n + O(k)$ to do a random walk of length

k (started at a random vertex) in a constant-degree expander graph on 2^n vertices. More background on such generators can be found in [Gol99, Sec 3.6.3]. The construction of [IW97] XORs the expander-walk generator with the (first k outputs of the) indistinguishability-preserving generator from Lemma 4.6.1:

Definition 4.6.3. *Let $k = c \cdot n$ for a constant $c > 1$. Let $NW''_k : \{0,1\}^{O(n)} \to (\{0,1\}^n)^k$ be a generator that is indistinguishability-preserving for size $2^{n/c}$ as given by Lemma 4.6.1. The generator $IW_k : \{0,1\}^l \to (\{0,1\}^n)^k$ is defined as*

$$IW_k(x,y) \stackrel{\text{def}}{=} NW''_k(x) \oplus W_k(y).$$

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

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

$$f' \stackrel{\text{def}}{=} (XOR \circ f^{\otimes k}) \circ IW_k : \{0,1\}^{O(n)} \to \{0,1\},$$

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

Theorem 4.6.4 ([IW97])**.** *If there is a function $f : \{0,1\}^n \to \{0,1\}$ in* **E** *that is 1/3-hard for size $2^{\Omega(n)}$, then there is a function $f' : \{0,1\}^m \to \{0,1\}$ in* **E** *that is $(1/2 - 2^{-\Omega(m)})$-hard for size $2^{\Omega(m)}$.*

Proof. By Theorem 4.5.2 there exists a δ'-random function $\mathrm{g} : \{0,1\}^n \to \{0,1\}$, where δ' is a constant, such that the hardness of $f' : \{0,1\}^{O(n)} \to \{0,1\}$ is $1/2 - \mathrm{ExpBias}\left[(XOR \circ \mathrm{g}^{\otimes k}) \circ IW_k\right] - 2^{-\Omega(n)}$ for circuits of size $2^{\Omega(n)}$.

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

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

where in the last inequality we use standard hitting properties of expander walks (see e.g. [Gol97] for a proof), and take c to be a sufficiently large constant. □

4.7 Limits of Monotone Hardness Amplification

4.7.1 On the hypothesis that f is balanced

The hardness amplification results in the previous sections start from balanced functions. In this section we study this hypothesis. Our main finding is that, while this hypothesis is not necessary for hardness amplification within **NP**/poly (i.e.,

non-deterministic polynomial size circuits), it is likely to be necessary for hardness amplification within **NP**.

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

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

It is easy to see that $\bar{f}$ is $1/\operatorname{poly}(n)$-hard if f is. Since a circuit can (non-uniformly) know $\Pr_x[f(x) = 1]$, the following hardness amplification within $\mathbf{NP}/\operatorname{poly}$ is a corollary to the proof of Theorem 4.4.1.

Corollary 4.7.1. *If there is a function $f : \{0,1\}^n \to \{0,1\}$ in $\mathbf{NP}/\operatorname{poly}$ that is $1/\operatorname{poly}(n)$-hard for size $s(n)$, then there is a function $f' : \{0,1\}^m \to \{0,1\}$ in $\mathbf{NP}/\operatorname{poly}$ that is $(1/2 - 1/s(\sqrt{m})^{\Omega(1)})$-hard for size $s(\sqrt{m})^{\Omega(1)}$.*

Now we return to hardness amplification within **NP**. First we note that, in our results, to amplify the hardness of $f : \{0,1\}^n \to \{0,1\}$ up to $1/2 - \epsilon$ it is only necessary that $\operatorname{Bias}[f] \leq \epsilon^c$ for some universal constant c. The argument is standard and can be found, for example, in [Tre03].

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

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

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

Definition 4.7.2. *An oracle algorithm $Amp : \{0,1\}^l \to \{0,1\}$ is a* black-box β-bias $[\delta \mapsto (1/2 - \epsilon)]$-hardness amplification for length n and size s *if for every $f : \{0,1\}^n \to$*

$\{0,1\}$ *such that* $\mathrm{Bias}\,[f] \leq \beta$ *and for every* $A : \{0,1\}^l \to \{0,1\}$ *such that*

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

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

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

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

Note that if *Amp* is as in Definition 4.7.2 and if f is δ-hard for size s' and $\mathrm{Bias}\,[f] \leq \beta$, then Amp^f is $(1/2-\epsilon)$-hard for size s'/s: if there were a circuit A of size s'/s computing Amp^f with error probability at most $1/2-\epsilon$, then C^A would be a circuit of size $s \cdot (s'/s) = s'$ computing f with error probability at most δ, contradicting the hardness of f. The term "black box" refers to the fact that the definition requires this to hold for *every* f and A, regardless of whether or not f is in **NP** and A is a small circuit.

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

Theorem 4.7.3. *For any constant* $\gamma > 0$, *if Amp is a monotone black-box* β*-bias* $[\delta \mapsto (1/2-\epsilon)]$*-hardness amplification for length* n *and size* $s \leq 2^{n/3}$ *such that* $1/2 - 4\epsilon > \delta + \gamma$, *then* $\beta \leq 8\epsilon + O(2^{-n})$.

The main ideas for proving this bound are the same as in the negative result for black-box hardness amplification in [Vio04]: First we show that the above kind of hardness amplification satisfies certain coding-like properties. Roughly, *Amp* can be seen as a kind of list-decodable code where the distance property is guaranteed only for δ-distant messages with bias at most β (cf. [Tre03]). Then we show that monotone functions fail to satisfy these properties. The limitation we prove on monotone functions relies on the following corollary to the Kruskal-Katona theorem (see [And02], Theorem 7.3.1).

Lemma 4.7.4. *Let* $\mathcal{S} = \{S_1, \ldots, S_m\}$ *be a collection of* m *subsets* $S_i \subseteq \{1, \ldots, N\}$, *where* $|S_i| = t$. *If* $m \geq \binom{N-1}{t} = \left(1 - \frac{t}{N}\right)\binom{N}{t}$ *then for every integer* $t' < t$

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

Let $\mathcal{F}_p$ be the uniform distribution on functions f whose truth-tables have relative Hamming weight exactly p, i.e. $\Pr_x[f(x) = 1] = p$. We use the above Lemma 4.7.4 to prove the following fact.

Lemma 4.7.5. *Let $A : \{0,1\}^l \to \{0,1\}$ be a an oracle function such that for every $x \in \{0,1\}^l$, $A^f(x)$ is a* monotone *function of the truth-table of $f : \{0,1\}^n \to \{0,1\}$. (For example, any monotone black-box hardness amplification Amp satisfies this condition.) Let τ be an integer multiple of $1/2^n$. Then there is $p \in \{1/2-\tau, 1/2+\tau\}$ such that:*

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

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

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

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

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

View the elements $f \in S_p$ as subsets of $\{1,\ldots,2^n\}$ of size exactly $p2^n$ in the natural way; i.e. the subset associated with f is the set of inputs x such that $f(x) = 1$. Note that if $f' \subseteq f \subseteq \{1,\ldots,2^n\}$ and $A^f(x) = 0$ then by the monotonicity of A, $A^{f'}(x) = 0$. Therefore, by Lemma 4.7.4,

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

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

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

Lemma 4.7.6. *Let $\gamma > 0$ be any fixed constant. Let Amp be a monotone black-box 8ϵ-bias $[\delta \mapsto (1/2-\epsilon)]$-hardness amplification for length n and size $s \leq 2^{n/3}$, where $1/2 - 4\epsilon > \delta + \gamma$, and let $\epsilon > 0$ be an integer multiple of $1/2^n$. Then for both p in $\{1/2-4\epsilon, 1/2+4\epsilon\}$ and every function G:*

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

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

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

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

Proof of Theorem 4.7.3. We assume that ϵ is a positive integer multiple of $1/2^n$ and show that $\beta \leq 8\epsilon$. The theorem then follows for general ϵ by rounding it up to the next integer multiple of $1/2^n$. We show that $\beta = 8\epsilon$ is impossible, and therefore that $\beta < 8\epsilon$ (because any β'-bias hardness amplification is clearly also a β-bias hardness amplification for every $\beta \leq \beta'$).

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

By Lemma 4.7.5, we may choose $p \in \{1/2 - 4\epsilon, 1/2 + 4\epsilon\}$ such that

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

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

$$\Pr_{U_l, F \leftarrow \mathcal{F}_p}[Amp^F(U_l) \neq G(U_l)]. \tag{4.7}$$

We have:

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

On the other hand, Lemma 4.7.6 implies that Quantity (4.7) is at least $1/2 - \epsilon - 2^{\Omega(2^n)}$ (note that the hypothesis of the lemma is satisfied by our assumption that $1/2 - 4\epsilon \geq \delta + \gamma$).

Combining the two bounds, we have that

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

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

4.7.2 Nondeterminism is necessary

In this subsection we show that *deterministic*, monotone, non-adaptive black-box hardness amplifications cannot amplify hardness beyond $1/2 - 1/\operatorname{poly}(n)$. Thus, the use of *nondeterminism* in our results (Section 4.5.4) seems necessary. Note that most hardness amplifications, including the one in this chapter, are black-box and non-adaptive.

O'Donnell [O'D04] proves that any monotone "direct product construction" (i.e. $f'(x_1, \ldots, x_k) = \mathrm{C}(f(x_1), \ldots, f(x_k))$, as in Equation 4.1) cannot amplify to hardness better than $1/2 - 1/n$, assuming that the amplification works for all functions f (not necessarily in **NP**). We relax the assumption that the hardness amplification is a direct product construction (allowing any monotone nonadaptive oracle algorithm $f' = Amp^f$). On the other hand, we require that the reduction proving its correctness is also black-box (as formalized in Definition 4.7.2).

We prove our bound even for hardness amplifications that amplify only balanced functions (i.e. $\beta = 0$ in Definition 4.7.2).

Theorem 4.7.7. *For every constant $\delta < 1/2$, if Amp is a black-box 0-bias $[\delta \mapsto (1/2-\epsilon)]$-hardness amplification for length n and size $s \leq 2^{n/3}$ such that for every x, $Amp^f(x)$ is a monotone function of $k \leq 2^{n/3}$ values of f, then*

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

The proof of this result follows closely the proof of the negative result on hardness amplification in [Vio04]. The main difference is here we use bounds on the noise stability of monotone functions rather than constant depth circuits.

The following lemma is similar to Lemma 4.7.6. The only difference is in considering functions F at distance η from f; this will correspond to perturbing the monotone amplification-function with noise having parameter η.

Lemma 4.7.8. *Let Amp be as in Definition 4.7.2 with $\beta = 0$ and $s \leq 2^{n/3}$. Then for any constant $\delta < 1/2$ there is a constant $\eta < 1/2$ such that, for sufficiently large n, the following holds: If $f : \{0,1\}^n \to \{0,1\}$ is any fixed balanced function and $F : \{0,1\}^n \to \{0,1\}$ is a random balanced function such that $Dist(f,F) = \eta$, then*

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

Proof. Let $N \stackrel{\text{def}}{=} 2^n$. It is easy to see that F is uniform on a set of size $\binom{N/2}{\eta N/2}^2$. The rest of the proof is like the proof of Lemma 4.7.6:

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

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

Proof of Theorem 4.7.7. Let η be the constant in Lemma 4.7.8. The idea is to consider

$$\Pr_{U_l, F, F'}[Amp^F(U_l) \neq Amp^{F'}(U_l)], \tag{4.8}$$

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

By the above lemma, the probability (4.8) is at least $1/2 - 2\epsilon$.

On the other hand, for every fixed x, $Amp^F(x)$ is a monotone function depending only on k bits of the truth-table of the function F. Since k is small compared to 2^n,

the distribution (F, F') induces on the input of $Amp^F(x)$ a distribution very close to $(U_k, U_k \oplus \mu)$, where μ is a noise vector with parameter η. Specifically, it can be verified that the statistical difference between these two distributions is at most $O(k^2/(\eta 2^n))$. Because this value is dominated by $\log^2 k/k$ when $k \leq 2^{n/3}$, and because $Amp^F(x)$ is a *monotone* function of k bits, we may apply Theorem 4.3.10 to conclude that the probability (4.8) is at most $1/2 - O(\log^2 k/k)$.

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

Chapter 5

Constant-Depth Circuits for Finite Field Arithmetic

5.1 Introduction

Finite fields have a wide variety of applications in computer science, ranging from Coding Theory to Cryptography to Complexity Theory. In this chapter we study the complexity of arithmetic operations in finite fields.

When considering the complexity of finite field arithmetic, there are two distinct problems one must consider. The first is the problem of actually *constructing* the desired finite field, $\mathbb{F}$; for example, one must find a prime p in order to realize the field $\mathbb{F}_p$ as $\mathbb{Z}/p\mathbb{Z}$. The second is the problem of performing arithmetic operations, such as addition, multiplication and exponentiation in the field $\mathbb{F}$. In this work, we focus on this second problem, and restrict our attention to fields $\mathbb{F}$ where a realization of the field can be found very easily, or where a realization of $\mathbb{F}$ is given as part of the input.

Specifically, we will focus on finite fields of characteristic two; that is, finite fields $\mathbb{F}_{2^n}$ having 2^n elements. In particular, the question we address is: *To what extent can basic field operations (e.g., multiplication, exponentiation) in $\mathbb{F}_{2^n}$ be computed by constant-depth circuits?* In our work, we consider three natural classes of unbounded fan-in constant-depth circuits: circuits over the bases $\{\wedge, \vee\}$ (i.e., $\mathbf{AC}^0$), $\{\wedge, \vee, \mathit{Parity}\}$ (i.e., $\mathbf{AC}^0[\oplus]$), and $\{\wedge, \vee, \mathit{Majority}\}$ (i.e., $\mathbf{TC}^0$). Moreover, we will focus on *uniform* constant-depth circuits, although we defer the discussion of uniformity until the paragraph "Uniformity" later in this section. Recall that, for polynomial-size circuits, $\mathbf{AC}^0 \subsetneq \mathbf{AC}^0[\oplus] \subsetneq \mathbf{TC}^0 \subseteq$ logarithmic space, where the last inclusion holds under logarithmic-space uniformity and the separations follow from works by Furst et al. [FSS84] and Razborov [Raz87], respectively (and hold even for non-uniform circuits). (Some additional background on constant-depth circuits is given in Section 2.4.)

Field Operations. Recall that the finite field $\mathbb{F}_{2^n}$ of characteristic two is generally realized as $\mathbb{F}_2[x]/(f(x))$ where $f(x) \in \mathbb{F}_2[x]$ is an irreducible polynomial of degree n. Thus, field elements are polynomials of degree at most $n-1$ over $\mathbb{F}_2$, addition of two field elements is addition in $\mathbb{F}_2[x]$ and multiplication of field elements is carried out modulo the irreducible polynomial $f(x)$. Throughout, we identify a field element $\alpha = a_{n-1}x^{n-1} + \cdots + a_1 x + a_0 \in \mathbb{F}_{2^n}$ with the n-dimensional bit-vector $(a_0, a_1, \ldots, a_{n-1}) \in \{0,1\}^n$, and we assume that all field elements that are given as inputs or returned as outputs of some computation are of this form.

In such a realization of $\mathbb{F}_{2^n}$, addition of two field elements is just componentwise XOR and therefore trivial, even for $\mathbf{AC}^0$ circuits. It is also easy to establish the complexity of Iterated Addition, i.e. given $\alpha_1, \alpha_2, \ldots, \alpha_t \in \mathbb{F}_{2^n}$, computing $\alpha_1 + \alpha_2 + \cdots + \alpha_t \in \mathbb{F}_{2^n}$. This is easily seen to be computable by $\mathbf{AC}^0[\oplus]$ circuits of size $\mathrm{poly}(n,t)$. On the other hand, since parity is a special case of Iterated Addition, the latter requires $\mathbf{AC}^0$ circuits of size $\mathrm{poly}(n, 2^{t^\epsilon})$ (see e.g. [Hås87]). Thus, we concentrate on the following *multiplicative* field operations:

- Iterated Multiplication: Given $\alpha_1, \alpha_2, \ldots, \alpha_t \in \mathbb{F}_{2^n}$, compute $\alpha_1 \cdot \alpha_2 \cdots \alpha_t \in \mathbb{F}_{2^n}$.
- Exponentiation: Given $\alpha \in \mathbb{F}_{2^n}$, and a t-bit integer k, compute $\alpha^k \in \mathbb{F}_{2^n}$.

The goal is to compute these functions as efficiently as possible for given parameters n, t. We note that these functions can be computed in time $\mathrm{poly}(n,t)$ (using the repeated squaring algorithm for exponentiation). In this work we ask what the smallest constant-depth circuits are for computing these functions. Note that computing Iterated Multiplication immediately implies being able to compute the product of two given field elements. While solving this latter problem is already non-trivial (for *Dlogtime*-, or even logspace-uniform constant-depth circuits), we will not address it separately.

Our Results. We present two different types of results. The first concerns field operations in a *specific* realization of $\mathbb{F}_{2^n}$, which we denote $\tilde{\mathbb{F}}_{2^n}$. The second type concerns field operations in an *arbitrary* realization of $\mathbb{F}_{2^n}$ as $\mathbb{F}_2[x]/(f(x))$, where we assume that the irreducible polynomial $f(x)$ is given as part of the input. We describe both of these kinds of results in more detail below. Then we discuss some applications of our results.

Results in the specific representation $\tilde{\mathbb{F}}_{2^n}$: In this setting, we assume that n is of the form $n = 2 \cdot 3^\ell$, for some non-negative integer ℓ, and we employ the explicit realization of $\mathbb{F}_{2^n}$ given by $\mathbb{F}_2[x]/(f(x))$ where $f(x)$ is the irreducible polynomial $x^{2\cdot 3^\ell} + x^{3^\ell} + 1 \in \mathbb{F}_2[x]$. Our results are summarized in Table 5.1.

We show that exponentiation can be computed by uniform $\mathbf{TC}^0$ circuits of size $\mathrm{poly}(n,t)$ (i.e. what is achievable by standard unbounded-depth circuits). To the best of our knowledge, prior to this work it was not even known how to compute

	$\mathbf{AC}^0$	$\mathbf{AC}^0[\oplus]$	$\mathbf{TC}^0$
Addition: $\alpha, \beta \in \tilde{\mathbb{F}}_{2^n} \to \alpha + \beta \in \tilde{\mathbb{F}}_{2^n}$	$\mathrm{poly}(n)$ [Folklore]	$\mathrm{poly}(n)$ [Folklore]	$\mathrm{poly}(n)$ [Folklore]
Iterated Addition: $\alpha_1, \ldots, \alpha_t \in \tilde{\mathbb{F}}_{2^n} \to \sum_{i \le t} \alpha_i \in \tilde{\mathbb{F}}_{2^n}$	$\mathrm{poly}(n, 2^{t^\epsilon})$ [Folklore]	$\mathrm{poly}(n, t)$ [Folklore]	$\mathrm{poly}(n, t)$ [Folklore]
Multiplication: $\alpha, \beta \in \tilde{\mathbb{F}}_{2^n} \to \alpha \cdot \beta \in \tilde{\mathbb{F}}_{2^n}$	$\mathrm{poly}(2^{n^\epsilon})$ [Cor. 5.2.6 (1)]	$\mathrm{poly}(n)$ [Thm. 5.2.4 (1)]	$\mathrm{poly}(n)$ [HAB02]
Iterated Multiplication: $\alpha_1, \ldots, \alpha_t \in \tilde{\mathbb{F}}_{2^n} \to \prod_{i \le t} \alpha_i \in \tilde{\mathbb{F}}_{2^n}$	$\mathrm{poly}(2^{n^\epsilon}, 2^{t^\epsilon})$ [Cor. 5.2.6 (1)]	$\mathrm{poly}(n, 2^{t^\epsilon})$ [Thm. 5.2.4 (1)]	$\mathrm{poly}(n, t)$ [HAB02]
Exponentiation: $\alpha \in \tilde{\mathbb{F}}_{2^n}$, t-bit $k \in \mathbb{Z} \to \alpha^k \in \tilde{\mathbb{F}}_{2^n}$	$\mathrm{poly}(2^{n^\epsilon}, 2^{t^\epsilon})$ [Cor. 5.2.6 (2)]	$\mathrm{poly}(n, 2^{t^\epsilon})$ [Thm. 5.2.4 (2)]	$\mathrm{poly}(n, t)$ [Thm. 5.2.3 (2)]
In the above, $\epsilon > 0$ is arbitrary, but the circuits have depth $O(1/\epsilon)$.			

Table 5.1: Complexity of Operations in $\tilde{\mathbb{F}}_{2^n} \equiv \mathbb{F}_2[x]/(x^{2 \cdot 3^l} + x^{3^l} + 1)$.

exponentiation in logarithmic space, i.e. space $O(\log(n + t))$, over any finite field of size $2^{\Omega(n)}$. As a corollary, we improve upon a theorem of Agrawal et al. [AAI+01] concerning exponentiation in uniform $\mathbf{AC}^0$. In the case of iterated multiplication of t field elements, results of Hesse et al. [HAB02] imply that this problem can be solved by uniform $\mathbf{TC}^0$ circuits of size $\mathrm{poly}(n, t)$.

We also show that, for every $\epsilon > 0$, iterated multiplication and exponentiation can be computed by uniform $\mathbf{AC}^0[\oplus]$ circuits of size $\mathrm{poly}(n, 2^{t^\epsilon})$. Moreover, we show that this is tight: neither iterated multiplication nor exponentiation can be computed by (nonuniform) $\mathbf{AC}^0[\oplus]$ circuits of size $\mathrm{poly}(n, 2^{t^{o(1)}})$.

Results in arbitrary representation $\mathbb{F}_2[x]/(f(x))$: In this setting we assume that the irreducible polynomial $f(x)$ is arbitrary, but is given to the circuit as part of the input. Our results are summarized in Table 5.2.

We show (with a more complicated proof than in the specific representation case) that iterated multiplication can be computed by uniform $\mathbf{AC}^0[\oplus]$ circuits of size $\mathrm{poly}(n, 2^{t^\epsilon})$, and this is again tight. We show that exponentiation can be computed by uniform $\mathbf{AC}^0[\oplus]$ circuits of size $\mathrm{poly}(n, 2^t)$, but we do not know how to match the size $\mathrm{poly}(n, 2^{t^\epsilon})$ achieved in the specific representation case. More dramatically, we do not know if there exist $\mathrm{poly}(n, 2^{o(t)})$-size $\mathbf{TC}^0$ circuits for exponentiation. While we cannot establish a lower bound for exponentiation, we observe that testing whether a given $\mathbb{F}_2[x]$ polynomial of degree n is irreducible can be $\mathbf{AC}^0[\oplus]$ reduced to computing exponentiation in a given representation of $\mathbb{F}_{2^n}$, for exponents with $t = n$ bits. Specifically, a modification of Rabin's irreducibility test [Rab80, MS83] gives a

	$\mathbf{AC}^0$	$\mathbf{AC}^0[\oplus]$	$\mathbf{TC}^0$
Addition: $\alpha, \beta \in \mathbb{F}_{2^n} \to \alpha + \beta \in \mathbb{F}_{2^n}$	$\mathrm{poly}(n)$ [Folklore]	$\mathrm{poly}(n)$ [Folklore]	$\mathrm{poly}(n)$ [Folklore]
Iterated Addition: $\alpha_1, \ldots, \alpha_t \in \mathbb{F}_{2^n} \to \sum_{i \le t} \alpha_i \in \mathbb{F}_{2^n}$	$\mathrm{poly}(n, 2^{t^\epsilon})$ [Folklore]	$\mathrm{poly}(n, t)$ [Folklore]	$\mathrm{poly}(n, t)$ [Folklore]
Multiplication: $\alpha, \beta \in \mathbb{F}_{2^n} \to \alpha \cdot \beta \in \mathbb{F}_{2^n}$	$\mathrm{poly}(2^{n^\epsilon})$ Cor. to [HAB02]	$\mathrm{poly}(n)$ [Thm. 5.2.7 (1)]	$\mathrm{poly}(n)$ [HAB02]
Iterated Multiplication: $\alpha_1, \ldots, \alpha_t \in \mathbb{F}_{2^n} \to \prod_{i \le t} \alpha_i \in \mathbb{F}_{2^n}$	$\mathrm{poly}(2^{n^\epsilon}, 2^{t^\epsilon})$ Cor. to [HAB02]	$\mathrm{poly}(n, 2^{t^\epsilon})$ [Thm. 5.2.7 (1)]	$\mathrm{poly}(n, t)$ [HAB02]
Exponentiation: $\alpha \in \mathbb{F}_{2^n}$, t-bit $k \in \mathbb{Z} \to \alpha^k \in \mathbb{F}_{2^n}$	$\mathrm{poly}(2^{n^\epsilon}, 2^{2^{\epsilon t}})$ Cor. to [HAB02]	$\mathrm{poly}(n, 2^t)$ [Thm. 5.2.7 (2)]	$\mathrm{poly}(n, 2^t)$ [HAB02]
In the above, $\epsilon > 0$ is arbitrary, but the circuits have depth $O(1/\epsilon)$.			

Table 5.2: Complexity of Operations in $\mathbb{F}_{2^n} \equiv \mathbb{F}_2[x]/(f(x))$ for given $f(x)$ of degree n.

$\mathbf{TC}^0$ reduction; we show a finer analysis that gives a $\mathbf{AC}^0[\oplus]$ reduction. Thus, any improvement on our results for exponentiation modulo a given (irreducible) polynomial of degree at most n would yield an upper bound on the complexity of testing irreducibility of a given $\mathbb{F}_2[x]$ polynomial. Some lower bounds for the latter problem are given in the recent work of Allender et. al. [ABD+03]. However, it is still open whether irreducibility of a given degree-n polynomial in $\mathbb{F}_2[x]$ can be decided by $\mathbf{AC}^0[\oplus]$ circuits of size $\mathrm{poly}(n)$.

We now discuss several applications of our results in the specific representation $\tilde{\mathbb{F}}_{2^n}$.

AE = Dlogtime-uniform $\mathbf{TC}^0$: Frandsen, Valence and Barrington [FVB94] study the relationship between uniform $\mathbf{TC}^0$ and the class AE of functions computable by certain arithmetic expressions (defined in Section 5.2.3). Remarkably, they show that *Dlogtime*-uniform $\mathbf{TC}^0$ is contained in AE. Conversely, they show that AE is contained in P-uniform $\mathbf{TC}^0$, but they leave open whether the inclusion holds under *Dlogtime* uniformity. We show that AE is in fact contained in *Dlogtime*-uniform $\mathbf{TC}^0$, thus proving that AE = *Dlogtime*-uniform $\mathbf{TC}^0$. (See paragraph "Uniformity" for a discussion of *Dlogtime*-uniformity.)

"Pseudorandom" Generators: We implement certain "pseudorandom" generators in *Dlogtime*-uniform constant-depth circuits. Specifically, we show how a construction of k-wise independent generators from [CG89, ABI86] can be implemented in uniform $\mathbf{AC}^0[\oplus]$, and how a construction of ϵ-biased generators from [AGHP92] can be implemented in uniform $\mathbf{TC}^0$. These constructions address a problem posed by

Gutfreund and Viola [GV04].

Overview of Techniques. Our results for the specific field representation $\tilde{\mathbb{F}}_{2^n} = \mathbb{F}_2[x]/(x^{2\cdot 3^\ell} + x^{3^\ell} + 1) \in \mathbb{F}_2[x]$. The crucial observation (Fact 5.3.1) is that the order of x modulo $x^{2\cdot 3^\ell} + x^{3^\ell} + 1$ is small and is easily computed, namely it is 3^{l+1}. Thus, we are able to compute large powers of the element $x \in \tilde{\mathbb{F}}_{2^n}$ by considering the exponent k modulo the order of x. To better illustrate this idea we now sketch a proof of the fact that exponentiation over $\tilde{\mathbb{F}}_{2^n}$ can be computed by uniform $\mathbf{TC}^0$ circuits of size $\mathrm{poly}(n,t)$. Let $\alpha \in \tilde{\mathbb{F}}_{2^n}$ and an exponent $0 \le k < 2^t$ be given. We think of α as a polynomial $\alpha(x) \in \mathbb{F}_2[x]$. Writing k in binary as $k = k_{t-1}k_{t-2}\cdots k_0 = \sum_{i<t} k_i 2^i$ where $k_i \in \{0,1\}$, we have:

$$\alpha(x)^k = \alpha(x)^{\sum_{i<t} k_i 2^i} = \prod_{i<t} \alpha(x)^{k_i 2^i} = \prod_{i<t} \alpha\left(x^{2^i}\right)^{k_i}$$

where the last equality follows from the fact that we are working in characteristic 2. Using the fact that the iterated product of t field elements is computable by uniform $\mathbf{TC}^0$ circuits of size $\mathrm{poly}(n,t)$ (which follows from results in [HAB02]), all that is left to do is to show how to compute $\alpha(x^{2^i})^{k_i}$. Since $k_i \in \{0,1\}$, the only hard step of this is computing x^{2^i} which can be done using the fact, discussed above, that the order of x is 3^{l+1}. Specifically, first we reduce 2^i mod 3^{l+1} using results about the complexity of integer arithmetic by Hesse et. al. [HAB02]. After the exponent is reduced, we show that computing the corresponding power of x is easy.

To prove that AE = *Dlogtime*-uniform $\mathbf{TC}^0$ we also show that $\tilde{\mathbb{F}}_{2^n}$ has an easily computable *dual basis* (as a vector-space over $\mathbb{F}_2$).

The other techniques we use are based on existing algorithms in the literature, e.g. [Kun74, Sie72, Rei86, Ebe89]. Our main contribution here is noticing that for some settings of parameters they can be implemented in $\mathbf{AC}^0[\oplus]$ and moreover that they give tight results for $\mathbf{AC}^0[\oplus]$. We now describe these techniques in more detail.

In the case of arbitrary realizations of $\mathbb{F}_{2^n}$ as $\mathbb{F}_2[x]/(f(x))$, the main technical challenge is reducing polynomials modulo $f(x)$. Previous work has addressed this problem and shown how (over arbitrary fields) this can be solved by uniform log-depth circuits (of fan-in 2) [Rei86, Ebe89], and even by uniform $\mathbf{TC}^0$ circuits [HAB02]. The approach that is usually taken is to give a parallel implementation of the Kung-Sievking algorithm [Kun74, Sie72] to reduce polynomial division to the problem of computing small powers of polynomials. However, this reduction requires summations of $\mathrm{poly}(n)$ polynomials, which is why previous results only give implementations in log-depth or by $\mathbf{TC}^0$ circuits. We take the same approach in our Lemma 5.4.1; however, we observe that in our setting we may compute these large summations using parity gates. This allows us to implement polynomial division over $\mathbb{F}_2[x]$ in $\mathbf{AC}^0[\oplus]$.

Both in our results for $\tilde{\mathbb{F}}_{2^n}$ and for arbitrary realizations of $\mathbb{F}_{2^n}$, we make use

of the Discrete Fourier Transform (DFT). This allows us to reduce the problem of multiplication or exponentiation of polynomials to the problem of multiplying or exponentiating field elements in fields of size poly(n) (and these problems are feasible for $\mathbf{AC}^0$ circuits). Eberly [Ebe89] and Reif [Rei86] have also employed the DFT in their works on performing polynomial arithmetic in log-depth circuits. However, as with polynomial division in $\mathbb{F}_2[x]$, the fact that we are working with polynomials over $\mathbb{F}_2$ allows us to compute the DFT and inverse DFT in uniform $\mathbf{AC}^0[\oplus]$ (and not just in log-depth or $\mathbf{TC}^0$).

Other Related Work: Works by Reif [Rei86] and Eberly [Ebe89] show how basic field arithmetic can be computed by log-depth circuits, and the results of Hesse, Allender and Barrington [HAB02] imply that some field arithmetic can be accomplished by uniform $\mathbf{TC}^0$. Indeed, the main result of [HAB02] states that integer division can be computed by (uniform) $\mathbf{TC}^0$ circuits, and hence addition and multiplication in the field $\mathbb{F}_p \simeq \mathbb{Z}/p\mathbb{Z}$ can be accomplished (in $\mathbf{TC}^0$) by adding or multiplying elements as integers and then reducing the result modulo p using the division result. Other results from [HAB02] imply that uniform $\mathbf{TC}^0$ circuits can compute iterated multiplication in (arbitrary realizations of) $\mathbb{F}_{2^n}$. Some results on the complexity of arithmetic in finite fields of *unbounded* characteristic are given in [SF93].

Uniformity. In the previous discussion we refer to uniform circuits for the various problems we consider. When working with restricted circuit classes, such as $\mathbf{AC}^0$, $\mathbf{AC}^0[\oplus]$ and $\mathbf{TC}^0$, one must be careful not to allow the machine constructing the circuits to be more powerful than the circuits themselves. Indeed, one of the significant technical contributions of [HAB02] is showing that integer division is in uniform $\mathbf{TC}^0$ under a very strong notion of uniformity, namely *Dlogtime*-uniformity [BIS90]. *Dlogtime*-uniformity, which is described briefly in Section 2.4, has become the generally-accepted convention for uniformity in constant-depth circuits [BIS90, FVB94, HAB02]. One reason for this is that *Dlogtime*-uniform constant-depth circuits have several elegant characterizations (see, e.g., [BIS90]); in fact, our results will prove yet another such characterization, namely *Dlogtime*-uniform $\mathbf{TC}^0 = AE$. *Unless otherwise specified, in this work "uniform" always means "Dlogtime-uniform".*

If one is willing to relax the uniformity condition to polynomial-time-uniformity, then some of our results on arithmetic in $\tilde{\mathbb{F}}_{2^n}$ can be proved more easily. For instance, the exponentiation result requires computing $x^{2^i} \in \tilde{\mathbb{F}}_{2^n}$ for a given i. Instead of actually computing x^{2^i} in the circuit, these values could be computed in polynomial time and then hardwired into the circuit. In contrast, in the case of our results in arbitrary realizations of $\mathbb{F}_{2^n}$, we do not know how to improve any of our results, even if we allow non-uniform circuits. If on the other hand, one allows non-uniformity that *depends on the irreducible polynomial* $f(x)$, then one can simplify some the proofs, and can actually improve the exponentiation result to match the parameters that we

achieve in $\tilde{\mathbb{F}}_{2^n}$ (by hardwiring the values x^{2^i} into the circuit, as above).

5.2 Our Results

In this section we formally state our results. In Section 5.2.1 we discuss our results in the specific case where n is of the form $n = 2 \cdot 3^l$, and $\mathbb{F}_{2^n}$ is realized as $\mathbb{F}_2[x]/(x^{2 \cdot 3^\ell} + x^{3^\ell} + 1)$, i.e. using the explicit irreducible polynomial $x^{2 \cdot 3^\ell} + x^{3^\ell} + 1 \in \mathbb{F}_2[x]$. In Section 5.2.2 we discuss our results in realizations of $\mathbb{F}_{2^n}$ as $\mathbb{F}_2[x]/(f(x))$ for an *arbitrary* irreducible polynomial $f(x) \in \mathbb{F}_2[x]$ that is given as part of the input. Then we discuss applications of our results. In Section 5.2.3 we prove that uniform $\mathbf{TC}^0 = AE$. In Section 5.2.4 we exhibit k-wise independent and ϵ-biased generators that are *bitwise* computable by uniform $\mathbf{AC}^0[\oplus]$ and $\mathbf{TC}^0$ circuits.

5.2.1 Field Arithmetic in $\tilde{\mathbb{F}}_{2^n}$

Below we summarize our main results concerning arithmetic in the field $\tilde{\mathbb{F}}_{2^n}$, defined below.

Fact 5.2.1 ([van99], Theorem 1.1.28)**.** *For all integers $l \geq 0$, the polynomial $x^{2 \cdot 3^\ell} + x^{3^\ell} + 1 \in \mathbb{F}_2[x]$ is irreducible.*

Definition 5.2.2. *For n of the form $n = 2 \cdot 3^\ell$, we define $\tilde{\mathbb{F}}_{2^n}$ to be the specific realization of $\mathbb{F}_{2^n}$ given by*

$$\tilde{\mathbb{F}}_{2^n} \stackrel{\text{def}}{=} \mathbb{F}_2[x]/(x^{2 \cdot 3^\ell} + x^{3^\ell} + 1).$$

The next theorem states our results about field arithmetic over $\tilde{\mathbb{F}}_{2^n}$ in uniform $\mathbf{TC}^0$. The first item follows from results of Hesse, Allender and Barrington [HAB02]; nonetheless, we state it for the sake of comparison with our other results.

Theorem 5.2.3. *Let $n = 2 \cdot 3^l$. There exist uniform $\mathbf{TC}^0$ circuits of size $\mathrm{poly}(n, t)$ that perform the following:*

1. *[HAB02] Given $\alpha_1, \alpha_2, \ldots, \alpha_t \in \tilde{\mathbb{F}}_{2^n}$, compute $\alpha_1 \cdot \alpha_2 \cdots \alpha_t \in \tilde{\mathbb{F}}_{2^n}$.*
2. *Given $\alpha \in \tilde{\mathbb{F}}_{2^n}$ and a t-bit integer k, compute $\alpha^k \in \tilde{\mathbb{F}}_{2^n}$.*

In particular, uniform $\mathbf{TC}^0$ circuits of polynomial size are capable of performing iterated multiplication and exponentiation in $\tilde{\mathbb{F}}_{2^n}$ that match the parameters that can be achieved by standard unbounded-depth circuits.

The next theorem states our results about field arithmetic over $\tilde{\mathbb{F}}_{2^n}$ in uniform $\mathbf{AC}^0[\oplus]$.

Theorem 5.2.4. *Let $n = 2 \cdot 3^l$. Then, for every constant $\epsilon > 0$, there exist uniform $\mathbf{AC}^0[\oplus]$ circuits of size $\mathrm{poly}(n, 2^{t^\epsilon})$ that perform the following:*

1. *Given* $\alpha_1, \alpha_2, \ldots, \alpha_t \in \tilde{\mathbb{F}}_{2^n}$, *compute* $\alpha_1 \cdot \alpha_2 \cdots \alpha_t \in \tilde{\mathbb{F}}_{2^n}$.

2. *Given* $\alpha \in \tilde{\mathbb{F}}_{2^n}$ *and a* t*-bit integer* k, *compute* $\alpha^k \in \tilde{\mathbb{F}}_{2^n}$.

While these parameters are considerably worse than for $\mathbf{TC}^0$ circuits, they are tight:

Theorem 5.2.5. *For every constant* d *there is* $\epsilon > 0$ *such that, for sufficiently large* t *and* $n = 2 \cdot 3^l$, *the following cannot be computed by (nonuniform)* $\mathbf{AC}^0[\oplus]$ *circuits of depth* d *and size* $2^{2^{\epsilon n}} \cdot 2^{t^\epsilon}$:

1. *Given* $\alpha_1, \alpha_2, \ldots, \alpha_t \in \tilde{\mathbb{F}}_{2^n}$, *compute* $\alpha_1 \cdot \alpha_2 \cdots \alpha_t \in \tilde{\mathbb{F}}_{2^n}$.

2. *Given* $\alpha \in \tilde{\mathbb{F}}_{2^n}$ *and a* t*-bit integer* k, *compute* $\alpha^k \in \tilde{\mathbb{F}}_{2^n}$.

In fact, Item (1) in the above negative result holds for any sufficiently large field (i.e. not only $\tilde{\mathbb{F}}_{2^n}$); and Item (2) holds for fields of a variety of different sizes. Both of these generalizations will be apparent from the proof.

By "scaling down" Theorem 5.2.3 (as described in Section 2.4) we obtain the following:

Corollary 5.2.6. *Let* $n = 2 \cdot 3^l$. *Then, for every constant* $\epsilon > 0$, *there exist uniform* $\mathbf{AC}^0$ *circuits of size* $\mathrm{poly}(2^{n^\epsilon}, 2^{t^\epsilon})$ *that perform the following:*

1. *Given* $\alpha_1, \alpha_2, \ldots, \alpha_t \in \tilde{\mathbb{F}}_{2^n}$, *compute* $\alpha_1 \cdot \alpha_2 \cdots \alpha_t \in \tilde{\mathbb{F}}_{2^n}$.

2. *Given* $\alpha \in \tilde{\mathbb{F}}_{2^n}$ *and a* t*-bit integer* k, *compute* $\alpha^k \in \tilde{\mathbb{F}}_{2^n}$.

This improves upon a theorem of Agrawal et al. [AAI+01] showing that field exponentiation is computable by uniform $\mathbf{AC}^0$ circuits of size $\mathrm{poly}(2^n, 2^t)$ (as opposed to $\mathrm{poly}(2^{n^\epsilon}, 2^{t^\epsilon})$ in our result). Corollary 5.2.6 is also tight for many settings of parameters (see Theorem 5.2.5).

5.2.2 Field Arithmetic in Arbitrary Realizations of $\mathbb{F}_{2^n}$

As noted above, one of the advantages of working with the field $\tilde{\mathbb{F}}_{2^n}$ is that we achieve tight results for $\mathbf{TC}^0$, $\mathbf{AC}^0[\oplus]$ and $\mathbf{AC}^0$. However, the use of $\tilde{\mathbb{F}}_{2^n}$ requires that $n = 2 \cdot 3^\ell$, and thus does not allow for the construction of $\mathbb{F}_{2^n}$ for all n; moreover some applications may require field computations in a specific field $\mathbb{F}_2[x]/(f(x))$ for some given irreducible polynomial $f(x)$ other than $x^{2 \cdot 3^\ell} + x^{3^\ell} + 1$. Thus we are led to study the complexity of arithmetic in the ring $\mathbb{F}_2[x]/(f(x))$ where the polynomial $f(x) \in \mathbb{F}_2[x]$ is *given as part of the input*. If, in addition, we have the promise that $f(x)$ is irreducible, then this corresponds to arithmetic in the field $\mathbb{F}_{2^n} \simeq \mathbb{F}_2[x]/(f(x))$.

Theorem 5.2.7.

1. *For every constant $\epsilon > 0$, there exist uniform* $\mathbf{AC}^0[\oplus]$ *circuits of size* $\mathrm{poly}(n, 2^{t^\epsilon})$ *that perform the following: Given $f(x) \in \mathbb{F}_2[x]$ of degree n and $\alpha_1, \alpha_2, \ldots, \alpha_t \in \mathbb{F}_2[x]/(f(x))$, compute $\alpha_1 \cdot \alpha_2 \cdots \alpha_t \in \mathbb{F}_2[x]/(f(x))$.*

2. *There exist uniform* $\mathbf{AC}^0[\oplus]$ *circuits of size* $\mathrm{poly}(n, 2^t)$ *that perform the following: Given $f(x) \in \mathbb{F}_2[x]$ of degree n, $\alpha \in \mathbb{F}_2[x]/(f(x))$ and a t-bit integer k, compute $\alpha^k \in \mathbb{F}_2[x]/(f(x))$.*

Since Item 1 of Theorem 5.2.5 actually holds for any realization of $\mathbb{F}_{2^n}$, and not just for $\tilde{\mathbb{F}}_{2^n}$ (as noted in the proof), Item 1 of Theorem 5.2.7 is tight.

Unlike Item 2 in Theorem 5.2.4, Exponentiation now requires size $\mathrm{poly}(n, 2^t)$, instead of $\mathrm{poly}(n, 2^{t^\epsilon})$. We do not know how to improve this to size $\mathrm{poly}(n, 2^{o(t)})$, even for $\mathbf{TC}^0$ circuits. On the other hand, we show that testing irreducibility of a given $\mathbb{F}_2[x]$ polynomial is $\mathbf{AC}^0[\oplus]$ reducible to computing exponentiation modulo a given irreducible polynomial.

Theorem 5.2.8. *The problem of determining whether a given polynomial $f(x) \in \mathbb{F}_2[x]$ of degree n is irreducible, is* $\mathrm{poly}(n)$*-size* $\mathbf{AC}^0[\oplus]$*-reducible to the following problem: Given an irreducible polynomial $f(x) \in \mathbb{F}_2[x]$ of degree n, compute the conjugates $x, x^2, x^{2^2}, \ldots, x^{2^{n-1}} \pmod{f(x)}$.*

5.2.3 *AE* = *Dlogtime* uniform $\mathbf{TC}^0$

Frandsen, Valence and Barrington [FVB94] study the relationship between uniform $\mathbf{TC}^0$ and the class *AE* of functions computable by certain arithmetic expressions (defined below). Remarkably, they show that *Dlogtime*-uniform $\mathbf{TC}^0$ is contained in *AE*. Conversely, they show that *AE* is contained in *P*-uniform $\mathbf{TC}^0$, but they leave open whether the inclusion holds for *Dlogtime* uniformity. We show that *AE* is in fact contained in *Dlogtime*-uniform $\mathbf{TC}^0$, thus proving that *AE* = *Dlogtime*-uniform $\mathbf{TC}^0$. (All these inclusions between classes hold in a certain technical sense that is made clear below.)

We now briefly review the definition of *AE* and then state our results.

Definition 5.2.9 ([FVB94])**.** *Let I be an infinite set of formal indices. The set of formal* arithmetic expressions *is defined as follows. The basic expressions are x (we think of this as the field element x), and* Input *(we think of this as the input field element). If e, e' are expressions (possibly containing the unbound index $i \in I$), then we may form new composite expressions $\sum_{i=1}^{u} e, \prod_{i=1}^{u} e, e + e', e \cdot e', e^{2^i}$, where $i \in I$ and u is either an index, i.e. $u \in I$, or is any polynomial in n (we think of n as the input length).*

An arithmetic expression is well-formed *if all indices are bound and they are bound in a semantically sound way (we omit details).*

We associate to every well-formed arithmetic expression e a family of functions $f_n^e : \tilde{\mathbb{F}}_{2^n} \to \tilde{\mathbb{F}}_{2^n}$*, for every n of the form* $n = 2 \cdot 3^\ell$ *(note that all computations are performed over the field* $\tilde{\mathbb{F}}_{2^n}$*).*

The complexity class AE consists of those families of functions $f_n : \tilde{\mathbb{F}}_{2^n} \to \tilde{\mathbb{F}}_{2^n}$ *that are described by arithmetic expressions (for every n of the form* $n = 2 \cdot 3^\ell$*).*

For example, the *trace* function, $\mathrm{tr} : \tilde{\mathbb{F}}_{2^n} \to \mathbb{F}_2 \subseteq \tilde{\mathbb{F}}_{2^n}$, defined by $\mathrm{tr}(\mathrm{Input}) \stackrel{\mathrm{def}}{=} \sum_{i=0}^{n-1} \mathrm{Input}^{2^i}$, is in *AE*.

Theorem 5.2.10. *AE = Dlogtime−uniform* $\mathbf{TC}^0$ *in the following sense:*

Let $f : \{0,1\}^n \to \{0,1\}^n$ *be in Dlogtime-uniform* $\mathbf{TC}^0$*. Then there is* $f' : \tilde{\mathbb{F}}_{2^n} \to \tilde{\mathbb{F}}_{2^n}$ *in AE such that for every n of the form* $2 \cdot 3^l$*, and for every x of length n,* $f(x) = f'(x)$*.*

Conversely, let $f : \tilde{\mathbb{F}}_{2^n} \to \tilde{\mathbb{F}}_{2^n}$ *be in AE. Then there is* $f' : \{0,1\}^n \to \{0,1\}^n$ *in Dlogtime-uniform* $\mathbf{TC}^0$ *such that for every n of the form* $2 \cdot 3^l$*, and for every x of length n,* $f(x) = f'(x)$*.*

Remark 5.2.11. *Our definition of arithmetic expressions is slightly different from the definition in [FVB94], which we denote [FVB94]-arithmetic expressions. We now argue that our definition only makes our results hold in a sense that is stronger than that in [FVB94]. From a syntactical point of view, [FVB94]-arithmetic expressions may use a special element g, which intuitively corresponds to our x. Then, from a semantical point of view, to define the class of functions [FVB94]-AE computed by [FVB94]-arithmetic expressions, they fix a certain* representation *of finite fields, which in particular fixes the element g. Their inclusion "uniform* $\mathbf{TC}^0$ *is contained in [FVB94]-AE" only holds* after a particular representation has been fixed. *Roughly, the representation fixes g such that* $g, g^2, g^4, \ldots, g^{2^{n-1}}$ *is a self-dual normal basis for the field. We note that computing such a g requires a lot of machinery, and it is not known (to the best of our knowledge) how to do it in, say, Dlogtime-uniform* $\mathbf{TC}^0$*. On the other hand, in our results we work over the standard representation of finite fields modulo the irreducible polynomial* $x^{2 \cdot 3^\ell} + x^{3^\ell} + 1$*, and we set* $g = x$*; it is easy to see that our representation is easily computable.*

Remark 5.2.12. *It is perhaps unsatisfactory that Theorem 5.2.10 only holds for certain input lengths. However, even the results in [FVB94] only hold for certain input lengths, though they are more "dense" than ours.*

5.2.4 k-wise and ϵ-biased generators

We use our results on computing field operations to give constant-depth implementations of certain pseudorandom generators, namely k-wise independent and ϵ-biased generators (see Definitions 2.2.1 and 2.1.1. The complexity of these generators is also studied by Gutfreund and Viola [GV04]. Our results will complement some of the results in [GV04] (see the remark at the end of this section).

Using our results on field operations we obtain the following results. Note both constructions are optimal up to constant factors (cf. [CGH$^+$85, AGHP92]).

Theorem 5.2.13.

1. *For every k and m there is a k-wise independent generator $G : \{0,1\}^s \to \{0,1\}^m$, with $s = O(k \log m)$ that is bitwise computable by uniform $\mathbf{AC}^0[\oplus]$ circuits of size $\mathrm{poly}(s, \log m) = \mathrm{poly}(s)$.*

2. *For every ϵ and m, there is an ϵ-biased generator $G : \{0,1\}^s \to \{0,1\}^m$ with $s = O(\log m + \log(1/\epsilon))$ that is bitwise computable by uniform $\mathbf{TC}^0$ circuits of size $\mathrm{poly}(s, \log m) = \mathrm{poly}(s)$.*

Remark 5.2.14. *A previous and different construction of k-wise independent generators in [GV04] matches (up to constant factors) Item 1 in Theorem 5.2.13 for the special case $k = O(1)$. The construction in Item 1 in Theorem 5.2.13 improves on the construction in [GV04] for $k = \omega(1)$. Also, in [GV04] they exhibit a construction of ϵ-biased generators computable by uniform $\mathbf{AC}^0[\oplus]$ circuits (while the construction in Item 2 in Theorem 5.2.13 uses $\mathbf{TC}^0$ circuits). However, the construction in [GV04] has worse dependence on ϵ.*

5.3 Arithmetic in $\tilde{\mathbb{F}}_{2^n}$

In this section we prove our results about field arithmetic in the field $\tilde{\mathbb{F}}_{2^n} = \mathbb{F}_2[x]/(x^{2 \cdot 3^\ell} + x^{3^\ell} + 1)$.

One useful property of $\tilde{\mathbb{F}}_{2^n}$ is that the order of $x \in \tilde{\mathbb{F}}_{2^n}$ is small, specifically it is $3^{\ell+1} = O(n)$. (A priori, it could have been as large as $2^n - 1$.)

Fact 5.3.1. *The order of $x \in \tilde{\mathbb{F}}_{2^n}$ is $3^{\ell+1}$.*

Proof. First we show that $x^{3^{l+1}} \equiv 1 \pmod{x^{2 \cdot 3^l} + x^{3^l} + 1}$. This holds because

$$x^{3^{l+1}} = x^{2 \cdot 3^l} \cdot x^{3^l} \equiv \left(x^{3^l} + 1\right) \cdot x^{3^l} = \left(x^{2 \cdot 3^l} + x^{3^l}\right) \equiv 1 \pmod{x^{2 \cdot 3^l} + x^{3^l} + 1}.$$

Thus the order of x has to divide 3^{l+1}. Noting that $x^{3^l} \not\equiv 1 \pmod{x^{2 \cdot 3^l} + x^{3^l} + 1}$, the result follows. □

One way in which Fact 5.3.1 is useful is that it allows us to easily reduce a given polynomial modulo $x^{2 \cdot 3^\ell} + x^{3^\ell} + 1$.

Lemma 5.3.2. *Let $n = 2 \cdot 3^\ell$. Then there exist uniform $\mathbf{AC}^0[\oplus]$ circuits of size $\mathrm{poly}(n, d)$ that, on input $g(x) \in \mathbb{F}_2[x]$ of degree at most d, compute $g(x) \pmod{x^{2 \cdot 3^\ell} + x^{3^\ell} + 1}$.*

We will ultimately prove a much more general statement (Lemma 5.4.1), namely that one can reduce $g(x) \in \mathbb{F}_2[x]$ modulo any given polynomial (and not only $x^{2\cdot 3^\ell} + x^{3^\ell} + 1$). However, the proof of this more general result is also much more complicated, and so we now give an easier proof for the special case of reducing modulo $x^{2\cdot 3^\ell} + x^{3^\ell} + 1$.

Proof of Lemma 5.3.2. First we show that, given $k \leq d$, we can compute $x^d \in \tilde{\mathbb{F}}_{2^n}$ by uniform $\mathbf{AC}^0[\oplus]$ circuits of size $\mathrm{poly}(n, d)$. The circuit will first use the fact that division of integers of $O(\log n + \log d)$ bits is computable by uniform $\mathbf{AC}^0$ circuits of size $\mathrm{poly}(n, d)$ (see Lemma 2.4.1) to reduce k modulo $3^{\ell+1}$ and obtain $0 \leq k' < 3^{\ell+1}$ such that $k' \equiv k \pmod{3^{\ell+1}}$. By Fact 5.3.1, $x^{k'} \equiv x^k \pmod{x^{2\cdot 3^\ell} + x^{3^\ell} + 1}$. Clearly, if $k' < 2 \cdot 3^\ell$, then the result is simply $x^{k'}$. On the other hand, if $2 \cdot 3^\ell \leq k' < 3 \cdot 3^\ell$, then $x^{k'} \equiv x^{k'-3^\ell} + x^{k'-2\cdot 3^\ell}$.

It follows that any given polynomial $g(x) \in \mathbb{F}_2[x]$ of degree d can be reduced modulo $x^{2\cdot 3^\ell} + x^{3^\ell} + 1$ by uniform $\mathbf{AC}^0[\oplus]$ circuits of size $\mathrm{poly}(n, d)$; indeed, the circuit needs only reduce each term x^i of $g(x)$ modulo $x^{2\cdot 3^\ell} + x^{3^\ell} + 1$, and then compute the sum of all the terms (using parities of d bits). □

A crucial way in which Fact 5.3.1 is useful is that it allows us to compute high powers, α^k, of field elements $\alpha \in \tilde{\mathbb{F}}_{2^n}$, in the special case when k is a power of 2.

Lemma 5.3.3. *Let n be of the form $n = 2 \cdot 3^\ell$. Then there exist uniform $\mathbf{AC}^0[\oplus]$ circuits of size $\mathrm{poly}(n, i)$ that, on input $\alpha \in \tilde{\mathbb{F}}_{2^n}$, computes $\alpha^{2^i} \in \tilde{\mathbb{F}}_{2^n}$.*

Proof. Since $\alpha^{2^n} = \alpha$ for all $\alpha \in \tilde{\mathbb{F}}_{2^n}$, we first reduce i modulo n. This can be accomplished by uniform $\mathbf{AC}^0$ circuits of size $\mathrm{poly}(n, i)$ by Lemma 2.4.1. From this point on we assume that $i \leq n$.

Let $\alpha(x) \in \mathbb{F}_2[x]$ be the polynomial representing α. Thus, it suffices to compute $\alpha(x)^{2^i} \equiv \alpha(x^{2^i})$ modulo $x^{2\cdot 3^\ell} + x^{3^\ell} + 1$. In particular, it suffices to compute $x^{h\cdot 2^i}$ in $\tilde{\mathbb{F}}_{2^n}$ for every $h, i \leq n$, since then we can then compute $\alpha(x^{2^i})$ by simply summing the appropriate terms.

We show that each $x^{h\cdot 2^i} \in \tilde{\mathbb{F}}_{2^n}$ can actually be computed in uniform $\mathbf{AC}^0$: Recall that the order of x modulo $f(x) = x^{2\cdot 3^\ell} + x^{3^\ell} + 1$ is $3^{\ell+1}$ by Fact 5.3.1. Therefore it suffices to be able to reduce $h \cdot 2^i$ modulo $3^{\ell+1}$, and then we can apply Lemma 5.3.2. The only hard part of this is reducing 2^i modulo $3^{\ell+1}$, since we can then multiply the result by h and divide by $3^{\ell+1}$ using Lemma 2.4.1.

We now show how to reduce 2^i modulo $3^{\ell+1}$. By the binomial theorem,

$$2^i \equiv (3-1)^i \equiv \sum_{j=0}^{i} \binom{i}{j} 3^j (-1)^{i-j} \bmod 3^{\ell+1}.$$

Noting that all the terms of this sum vanish for $j \geq \ell+1$ (thanks to the 3^j factor), we have

$$\sum_{j=0}^{\ell} \binom{i}{j} 3^j (-1)^{i-j} \bmod 3^{\ell+1} \equiv \sum_{j=0}^{\ell} \frac{i(i-1)\cdots(i-j+1)}{j(j-1)\cdots 1} \cdot 3^j (-1)^{i-j} \bmod 3^{\ell+1}.$$

Since $\ell = O(\log n)$ and $|i| = O(\log n)$, we can compute, for every j, $\frac{i(i-1)\cdots(i-j+1)}{j(j-1)\cdots 1}$ by using an iterated product (of $O(\log n)$ integers of $O(\log n)$ bits) for the numerator and denominator, and then performing a division of the results (i.e., of integers having polylog(n) bits); both of these can be done by uniform $\mathbf{AC}^0$ circuits of size poly(n) by Lemma 2.4.1. Additionally, the 3^j term can be computed (using iterated multiplication, say), and the $(-1)^{(i-j)}$ is easy to compute.

Finally, since $\ell = O(\log n)$, the sum can be computed by uniform $\mathbf{AC}^0$ circuits of size poly(n) using an iterated sum of integers having polylog(n) bits. Clearly, the result only has polylog(n) bits, and so we may reduce modulo $3^{\ell+1}$ one last time to find $2^i \pmod{3^{\ell+1}}$. □

We note that the above lemma is somewhat easier to prove if one is willing to settle for either $\mathbf{TC}^0$ circuits (as opposed to $\mathbf{AC}^0[\oplus]$) or size poly$(n, 2^{i^\epsilon})$ (as opposed to poly(n, i)), which is all that is needed below. Nonetheless, we prefer to state and prove this single more general result.

We now prove our main theorems about field operations in $\tilde{\mathbb{F}}_{2^n}$.

Theorem 5.2.3 (restated)**.** *Let $n = 2 \cdot 3^l$. There exist uniform $\mathbf{TC}^0$ circuits of size* poly(n, t) *that perform the following:*

1. *[HAB02] Given $\alpha_1, \alpha_2, \ldots, \alpha_t \in \tilde{\mathbb{F}}_{2^n}$, compute $\alpha_1 \cdot \alpha_2 \cdots \alpha_t \in \tilde{\mathbb{F}}_{2^n}$.*
2. *Given $\alpha \in \tilde{\mathbb{F}}_{2^n}$ and a t-bit integer k, compute $\alpha^k \in \tilde{\mathbb{F}}_{2^n}$.*

Proof of Theorem 5.2.3. (1) Each field element α_i is represented by a polynomial $\alpha_i(x) \in \mathbb{F}_2[x]$. For the moment, we will actually consider the polynomials $\alpha_i(x)$ as polynomials $\alpha_i'(x)$ over the integers, i.e. as polynomials with coefficients in $\{0, 1\} \subset \mathbb{Z}$. It is proved in [HAB02] that the product of t polynomials of degree n over $\mathbb{Z}$ can be computed by uniform $\mathbf{TC}^0$ circuits of size poly(n, t). Thus, the product $A'(x) \stackrel{\text{def}}{=} \alpha_1'(x) \cdots \alpha_t'(x)$ can be computed by uniform $\mathbf{TC}^0$ circuits of size poly(n, t). Clearly, $A(x) \stackrel{\text{def}}{=} \alpha_1(x) \cdots \alpha_t(x) \equiv A'(x) \pmod 2$, and so it remains to reduce $A(x)$ modulo $x^{2 \cdot 3^\ell} + x^{3^\ell} + 1$; however, this follows from Lemma 5.3.2 (or by results in [HAB02]).

(2) We reduce the computation of α^k to the computation of a product $\alpha_1 \cdot \alpha_2 \cdots \alpha_t$ and apply Part (1). The integer $k = \sum_{i=0}^{t-1} k_i 2^i$ is given in binary, as $k_{t-1} \cdots k_1 k_0$, $k_i \in \{0, 1\}$, and thus

$$\alpha^k = \alpha^{\sum_i k_i 2^i} = \left(\alpha^{2^0}\right)^{k_0} \cdot \left(\alpha^{2^1}\right)^{k_1} \cdots \left(\alpha^{2^{t-1}}\right)^{k_{t-1}}.$$

Hence, to apply part(1), it suffices to show that each term $\left(\alpha^{2^i}\right)^{k_i}$ can be computed by $\mathbf{TC}^0$ circuits of size poly(n, t). Computing α^{2^i} follows from Lemma 5.3.3 and, since $k_i \in \{0, 1\}$, the exponentiation by k_i is easy. □

Theorem 5.2.4 (restated)**.** *Let $n = 2 \cdot 3^l$. Then, for every constant $\epsilon > 0$, there exist uniform $\mathbf{AC}^0[\oplus]$ circuits of size $\mathrm{poly}(n, 2^{t^\epsilon})$ that perform the following:*

1. *Given $\alpha_1, \alpha_2, \ldots, \alpha_t \in \tilde{\mathbb{F}}_{2^n}$, compute $\alpha_1 \cdot \alpha_2 \cdots \alpha_t \in \tilde{\mathbb{F}}_{2^n}$.*
2. *Given $\alpha \in \tilde{\mathbb{F}}_{2^n}$ and a t-bit integer k, compute $\alpha^k \in \tilde{\mathbb{F}}_{2^n}$.*

Proof of Theorem 5.2.4. (1) The idea is to reduce the problem to computing iterated multiplication over an exponentially smaller field $\mathbb{F}'$ via the Discrete Fourier Transform. We can compute iterated multiplication over $\mathbb{F}'$ in uniform $\mathbf{AC}^0$ by scaling down the $\mathbf{TC}^0$ result (Theorem 5.2.3, Item 1). Details follow.

Consider an iterated multiplication instance $(\alpha_1, \ldots, \alpha_t)$. Recalling that $\tilde{\mathbb{F}}_{2^n} = \mathbb{F}_2[x]/(f(x))$ (where $f(x) = x^{2 \cdot 3^l} + x^{3^l} + 1$ is the irreducible polynomial) we may view each α_i as a polynomial $\alpha_i(x)$ of degree at most $n-1$ in $\mathbb{F}_2[x]$. To compute $\alpha_1 \cdot \alpha_2 \cdots \alpha_t \in \tilde{\mathbb{F}}_{2^n}$ it will then suffice to compute the *polynomial* product $A(x) \stackrel{\text{def}}{=} \alpha_1(x)\alpha_2(x)\cdots\alpha_t(x) \in \mathbb{F}_2[x]$, and then apply Lemma 5.3.2 to reduce this polynomial modulo $f(x)$.

Let $m \in \{\log n + \log t, \ldots, 3(\log n + \log t)\}$ be of the form $m = 2 \cdot 3^{l'}$ for some l' (such an m can be found by uniform $\mathbf{AC}^0$ circuits of size $\mathrm{poly}(2^m) = \mathrm{poly}(n, t)$), and consider the field $\tilde{\mathbb{F}}_{2^m}$. To compute the polynomial product $A(x)$ we will first evaluate each polynomial $\alpha_i(x)$ at every element $\gamma_i, 1 \le i < 2^m$, of $\tilde{\mathbb{F}}^\times_{2^m}$. Next, we will compute $A(\gamma_1), \ldots, A(\gamma_{2^m-1})$ by using iterated product over the field $\tilde{\mathbb{F}}_{2^m}$ to compute $A(\gamma_i) = \alpha_1(\gamma_i) \cdots \alpha_t(\gamma_i)$. Then, since $A(x)$ has degree at most $(n-1) \cdot t < 2^{\log n + \log t} - 1$, the values $A(\gamma_1), \ldots, A(\gamma_{2^m-1})$ uniquely determine $A(x)$, and we will show how to interpolate in uniform $\mathbf{AC}^0[\oplus]$ to recover $A(x)$.

To accomplish these steps we will use the Discrete Fourier Transform matrix. That is, let $g \in \tilde{\mathbb{F}}_{2^m}$ be a generator of $\tilde{\mathbb{F}}_{2^m}$ and note that such a generator can be found in uniform $\mathbf{AC}^0$ by brute force (by computing exponentiation over $\tilde{\mathbb{F}}_{2^m}$, which can be done by scaling down the $\mathbf{TC}^0$ result, i.e. Theorem 5.2.3, Item 2. Alternatively, one can use Theorem 3.2 in [AAI+01]). Now define the matrix $D = (d_{i,j})_{0 \le i,j \le 2^m-2}$, where $d_{i,j} \stackrel{\text{def}}{=} g^{i \cdot j}$ and note that $D^{-1} = (d^{-1}_{i,j})_{0 \le i,j \le 2^m-2}$. If we view α_i as a (2^m-1)-dimensional vector $\alpha_{\mathbf{i}} = (\alpha_i^{(0)}, \ldots, \alpha_i^{(2^m-2)}) \in \mathbb{F}_2^{2^m-1}$, where $\alpha_i^{(j)}$ is the coefficient of x^j in $\alpha_i(x)$ (either 0 or 1), then $D\alpha_{\mathbf{i}} = (\alpha_i(g^0), \alpha_i(g^1), \ldots, \alpha_i(g^{2^m-2}))$. The matrix-vector product $D\alpha_{\mathbf{i}}$ can be computed by uniform $\mathbf{AC}^0[\oplus]$ circuits of size $\mathrm{poly}(2^m)$ because it only involves computing parities (of fan-in $2^m - 1$) and multiplications in the field $\tilde{\mathbb{F}}_{2^m}$, which we can do by uniform $\mathbf{AC}^0$ circuits of size $\mathrm{poly}(2^m)$ by scaling down the $\mathbf{TC}^0$ result, i.e. Theorem 5.2.3, Item (1).

Once the matrix-vector products $D\alpha_{\mathbf{1}}, \ldots, D\alpha_{\mathbf{t}}$ have been computed, the resulting vectors can be multiplied component-wise (using the scaled-down version of Theorem 5.2.3, item 1) to obtain the vector $\hat{A} = (A(g^0), \ldots, A(g^{2^m-2}))$. Next, note that $\mathbf{A} = D^{-1}\hat{A}$ can also be computed in $\mathbf{AC}^0[\oplus]$, just as $D\alpha_i$ was computed above, allowing us to recover the product polynomial $A(x)$.

Finally, by Lemma 5.3.2, $A(x)$ can be reduced modulo the irreducible polynomial $f(x)$ to obtain the field element $A = \alpha_1 \cdots \alpha_t$.

(2) As in the uniform $\mathbf{TC}^0$ case we can reduce this problem to the product of t field elements. Specifically, the reduction in Item 3 in Theorem 5.2.3 needs to compute α^{2^i} for $i \leq t$. These can be computed in uniform $\mathbf{AC}^0[\oplus]$ by Lemma 5.3.3. For the iterated product we use the previous item. □

Theorem 5.2.5 (restated)**.** *For every constant d there is $\epsilon > 0$ such that, for sufficiently large t and $n = 2 \cdot 3^l$, the following cannot be computed by (nonuniform) $\mathbf{AC}^0[\oplus]$ circuits of depth d and size $2^{2^{\epsilon n}} \cdot 2^{t^\epsilon}$:*

1. *Given $\alpha_1, \alpha_2, \ldots, \alpha_t \in \tilde{\mathbb{F}}_{2^n}$, compute $\alpha_1 \cdot \alpha_2 \cdots \alpha_t \in \tilde{\mathbb{F}}_{2^n}$.*
2. *Given $\alpha \in \tilde{\mathbb{F}}_{2^n}$ and a t-bit integer k, compute $\alpha^k \in \tilde{\mathbb{F}}_{2^n}$.*

Proof of Theorem 5.2.5. (1) We reduce MAJORITY on t bits to computing $\alpha_1 \cdot \alpha_2 \cdots \alpha_t$ for given $\alpha_1, \alpha_2, \ldots, \alpha_t \in \tilde{\mathbb{F}}_{2^n}$, where $n \in \{\log(t+1), \ldots, 3\log(t+1)\}$ is of the form $n = 2 \cdot 3^l$ for some l. Since by a result of Razborov [Raz87] and Smolensky [Smo87] we know that for every constant d there is a constant $\epsilon > 0$ such that MAJORITY on t bits cannot be computed by $\mathbf{AC}^0[\oplus]$ circuits of depth d and size 2^{t^ϵ}, the result follows. We now describe the reduction. Let $g \in \mathbb{F}^\times$ be a generator. Given a MAJORITY instance $x = w_1, w_2, \ldots, w_t$, consider the following instance of iterated multiplication: $\alpha_1, \alpha_2, \ldots, \alpha_t$, where $\alpha_i \stackrel{\text{def}}{=} g \in \mathbb{F}$ if $w_i = 1$, and $\alpha_i \stackrel{\text{def}}{=} 1 \in \mathbb{F}$ if $w_i = 0$. It is easy to see that $\alpha_1 \cdot \alpha_2 \cdots \alpha_t = g^j \in \mathbb{F}$ where $j = \sum_i w_i$. We can decide majority simply by checking whether $j \geq t/2$; this last step can be accomplished by a simple look-up in a (nonuniform) table of size $\text{poly}(n, t)$.

(2) We reduce MAJORITY on t bits to computing $\alpha^k \in \tilde{\mathbb{F}}_{2^n}$ for $|k| = O(t \log t)$ and $n = O(\log t)$. Since by a result of Razborov [Raz87] and Smolensky [Smo87] we know that for every constant d there is a constant $\epsilon > 0$ such that MAJORITY on t bits cannot be computed by $\mathbf{AC}^0[\oplus]$ circuits of depth d and size 2^{t^ϵ}, the result follows. Let $l \stackrel{\text{def}}{=} \lceil \log_3 \log_2(t+1) \rceil$ and $m \stackrel{\text{def}}{=} 3^l$. Set $n \stackrel{\text{def}}{=} 2 \cdot m$ and consider the field $\tilde{\mathbb{F}}_{2^n}$. Note that since $|\tilde{\mathbb{F}}_{2^n}^\times| = 2^n - 1 = (2^m - 1)(2^m + 1)$, there is an element $\alpha \in \tilde{\mathbb{F}}_{2^n}$ of order $(2^m - 1)$.

The reduction works as follows. From the MAJORITY instance $z = z_0 z_1 \ldots z_{t-1}$ construct an integer k with binary representation

$$k = z_{t-1} \underbrace{00\cdots0}_{m-1 \text{ zeros}} z_{t-2} \underbrace{00\cdots0}_{m-1 \text{ zeros}} z_{t-3} \ldots z_1 \underbrace{00\cdots0}_{m-1 \text{ zeros}} z_0 = \sum_{i=0}^{t-1} z_i (2^m)^i.$$

Now observe that $k = \sum_i z_i (2^m)^i \equiv \sum_i z_i \pmod{2^m - 1}$. Therefore $\alpha^k = \alpha^{\sum_i z_i}$; since $t < 2^m - 1$, this uniquely determines $\sum_i z_i$, and so MAJORITY can now be decided via look-up in a (nonuniform) table of size $\text{poly}(n, t)$. □

5.4 Arithmetic in Other Realizations of $\mathbb{F}_{2^n}$

In this section we prove Theorem 5.2.7. An important difference between this setting and the case of field operations in $\tilde{\mathbb{F}}_{2^n}$ is that we must now be able to reduce a polynomial $g(x) \in \mathbb{F}_2[x]$ modulo an arbitrary given polynomial $f(x) \in \mathbb{F}_2[x]$, and not only modulo $x^{2 \cdot 3^\ell} + x^{3^\ell} + 1$. The next lemma states that polynomial division in $\mathbb{F}_2[x]$ can be computed in uniform $\mathbf{AC}^0[\oplus]$.

Lemma 5.4.1. *There exist uniform* $\mathbf{AC}^0[\oplus]$ *circuits of size* $\mathrm{poly}(n, d)$ *that, on input polynomials* $f(x), g(x) \in \mathbb{F}_2[x]$ *where* $\deg(f) = n$ *and* $\deg(g) \leq d$*, computes the unique polynomials* $q(x), r(x) \in \mathbb{F}_2[x]$*, such that* $g(x) = q(x)f(x) + r(x)$*, where* $\deg(q) = \deg(g) - n$ *and* $\deg(r) < n$*.*

The approach for proving Lemma 5.4.1 is to implement, in constant-depth, the Kung-Sieveking [Kun74, Sie72] algorithm, which reduces the problem of polynomial division to the problem of computing small powers of polynomials. A similar approach has been employed by Reif [Rei86] and Eberly [Ebe89] in constructing log-depth circuits for polynomial division. The essential difference here is the observation that log-depth is only required to compute sums of $\mathrm{poly}(n)$ polynomials and, in our setting, we may instead use parity gates to accomplish such large summations in constant depth.

Before proving Lemma 5.4.1, we show how to compute small powers of polynomials in $\mathbf{AC}^0[\oplus]$, as this is an essential component of the proof.

Lemma 5.4.2. *There exist uniform* $\mathbf{AC}^0[\oplus]$ *circuits of size* $\mathrm{poly}(n, 2^t)$ *that, on input* $s(x) \in \mathbb{F}_2[x]$ *of degree* n *and a* t*-bit integer* k*, compute the polynomial* $s(x)^k$*.*

Proof. As in the proof of Theorem 5.2.4, part 1, we also use the Discrete Fourier Transform to compute $s(x)^k$.

In particular, let $m \in \{\log n + t, \ldots, 3(\log n + t)\}$ be of the form $m = 2 \cdot 3^{l'}$ for some l' (such an m can be found by uniform $\mathbf{AC}^0$ circuits of size $\mathrm{poly}(2^m) = \mathrm{poly}(n, 2^t)$), and consider the field $\tilde{\mathbb{F}}_{2^m}$. To compute the polynomial power $s(x)^k$, we first evaluate the polynomial $s(x)$ at every element $\gamma_i, 1 \leq i < 2^m$, of $\tilde{\mathbb{F}}_{2^m}^\times$, just as in the proof of Theorem 5.2.4, part 1. Next, we compute $s(\gamma_1)^k, \ldots, s(\gamma_{2^m-1})^k$ by using exponentiation in the field $\tilde{\mathbb{F}}_{2^m}$ (which follows from part 2 of Corollary 5.2.6; alternatively, one can use Theorem 3.2 from [AAI+01].) Then, by the choice of m, the values $s(\gamma_1)^k, \ldots, s(\gamma_{2^m-1})^k$ uniquely determine $s(x)^k$, and thus we can interpolate in uniform $\mathbf{AC}^0[\oplus]$ to recover $A(x)$, using the inverse Fourier Transform just as in the proof of Theorem 5.2.4, part 1. □

Proof of Lemma 5.4.1. Denote the degree of $g(x)$ by $m \leq d$, and throughout we write $f(x) = x^n + a_{n-1}x^{n-1} + \cdots + a_0$ and $g(x) = x^m + b_{m-1}x^{n-1} + \cdots + b_0$ for $a_i, b_i \in \mathbb{F}_2$. The algorithm will proceed as follows:

(1) Construct $f_R(x) \stackrel{\text{def}}{=} a_0x^n + \cdots + a_{n-1}x + 1$ by reversing the coefficients of $f(x)$.

(2) Construct $g_R(x) \stackrel{\text{def}}{=} b_0 x^m + \cdots + b_{m-1}x + 1$ by reversing the coefficients of $g(x)$.

(3) Let $\tilde{f}_R(x) \stackrel{\text{def}}{=} 1 + (1 - f_R(x)) + (1 - f_R(x))^2 + \cdots + (1 - f_R(x))^{m-n}$. (Note that $\tilde{f}_R(x)$ is simply a truncation of the power series $f_R(x)^{-1} = 1 + (1 - f_R(x)) + (1 - f_R(x))^2 + \cdots$.)

(4) Compute $h(x) \stackrel{\text{def}}{=} \tilde{f}_R(x)g_R(x) = c_0 + c_1 x + c_2 x^2 + \cdots$, and then the coefficients of $q(x) = q_{m-n}x^{m-n} + \cdots + q_1 x + q_0$ can be read off as $q_i = c_{m-n-i}$, i.e. the reverse of the lowest $m - n + 1$ coefficients of $h(x)$.

(5) Once $q(x)$ has been computed, $r(x)$ can be found by computing $r(x) = g(x) - q(x)f(x)$.

Before proving the correctness of the algorithm, let us see why it can be performed by uniform $\mathbf{AC}^0[\oplus]$ circuits: Steps (1) and (2) are trivial. The computation of $(1 - f_R(x))^k$ for $0 \leq k \leq m - n$ follows from Lemma 5.4.2, and it is clear that the summation in step (3) only requires (unbounded fan-in) parity gates. Step (4) is trivial. Step (5) only requires polynomial multiplication which is easily seen to be in uniform $\mathbf{AC}^0[\oplus]$.

Now we establish the correctness of the algorithm. Note that $f_R(x) = x^n f(1/x)$, $g_R(x) = x^m g(1/x)$ and define $q_R(x) = x^{m-n}q(1/x)$ and $r_R(x) = x^{n-1}r(1/x)$. Thus we have

$$g(x) = q(x)f(x) + r(x)$$
$$g(1/x) = q(1/x)f(1/x) + r(1/x)$$
$$g_R(x) = q_R(x)f_R(x) + x^{m-n+1}r_R(x)$$

Hence $h(x) \stackrel{\text{def}}{=} \tilde{f}_R(x)g_R(x) = q_R(x)(\tilde{f}_R(x)f_R(x)) + x^{m-n+1}\tilde{f}_R(x)r_R(x)$. Note, however, that

$$\tilde{f}_R(x)f_R(x) = \tilde{f}_R(x)(1 - (1 - f_R(x))) = 1 + (1 - f_R(x))^{m-n+1},$$

and since the constant term of $f_R(x)$ is 0 (by the assumption that f has degree exactly n), we have that

$$\tilde{f}_R(x)f_R(x) = 1 + x^{m-n+1}t(x)$$

for some $t(x) \in \mathbb{F}_2[x]$. In particular,

$$h(x) \stackrel{\text{def}}{=} \tilde{f}_R(x)g_R(x) = q_R(x)(1 + x^{m-n+1}t(x)) + x^{m-n+1}\tilde{f}_R(x)r_R(x),$$

and it is clear that the lowest $m - n$ coefficients of $h(x)$ are the coefficients of $q_R(x)$ as claimed. □

Now we are prepared to prove Theorem 5.2.7. We restate the theorem for the reader's convenience.

Theorem 5.2.7 (restated)**.**

1. *For every constant $\epsilon > 0$, there exist uniform $\mathbf{AC}^0[\oplus]$ circuits of size $\mathrm{poly}(n, 2^{t^\epsilon})$ that perform the following: Given $f(x) \in \mathbb{F}_2[x]$ of degree n and $\alpha_1, \alpha_2, \ldots, \alpha_t \in \mathbb{F}_2[x]/(f(x))$, compute $\alpha_1 \cdot \alpha_2 \cdots \alpha_t \in \mathbb{F}_2[x]/(f(x))$.*

2. *There exist uniform $\mathbf{AC}^0[\oplus]$ circuits of size $\mathrm{poly}(n, 2^t)$ that perform the following: Given $f(x) \in \mathbb{F}_2[x]$ of degree n, $\alpha \in \mathbb{F}_2[x]/(f(x))$ and a t-bit integer k, compute $\alpha^k \in \mathbb{F}_2[x]/(f(x))$.*

Proof of Theorem 5.2.7. (1) It suffices to replace the use of Lemma 5.3.2 in the proof of part 1 of Theorem 5.2.4 with Lemma 5.4.1.

(2) Consider $\alpha \in \mathbb{F}_2[x]/(f(x))$ as a polynomial $\alpha(x) \in \mathbb{F}_2[x]$ of degree at most $n-1$. We may apply Lemma 5.4.2 to compute $\alpha(x)^k$ (which has degree at most $k \cdot (n-1) \leq \mathrm{poly}(n, 2^t)$) by $\mathbf{AC}^0[\oplus]$ circuits of size $\mathrm{poly}(n, 2^t)$, and then apply Lemma 5.4.1, to reduce it modulo $f(x)$, again by $\mathbf{AC}^0[\oplus]$ circuits of size $\mathrm{poly}(n, 2^t)$. □

Theorem 5.2.8 (restated)**.** *The problem of determining whether a given polynomial $f(x) \in \mathbb{F}_2[x]$ of degree n is irreducible, is $\mathrm{poly}(n)$-size $\mathbf{AC}^0[\oplus]$-reducible to the following problem: Given an irreducible polynomial $f(x) \in \mathbb{F}_2[x]$ of degree n, compute the conjugates $x, x^2, x^{2^2}, \ldots, x^{2^{n-1}} \pmod{f(x)}$.*

Proof. The reduction proceeds as follows on input $f(x) \in \mathbb{F}_2[x]$ of degree n:

(i) Use the oracle to try to compute $x, x^2, x^{2^2}, \ldots, x^{2^{n-1}} \pmod{f(x)}$. Call the resulting quantities $a_0, a_1, \ldots, a_{n-1}$.

(ii) Check that $a_0 = x$, that $a_{i+1} \equiv a_i^2 \pmod{f(x)}$ for all $0 \leq i \leq n-1$ and that $a_{n-1}^2 \equiv x \pmod{f(x)}$. Otherwise, return REDUCIBLE.

(iii) If

$$\prod_{\text{primes } p \mid n} \left(x^{2^{n/p}} - x \right) \equiv 0 \pmod{f(x)}$$

then return REDUCIBLE, otherwise return IRREDUCIBLE.

First we argue the correctness of the reduction. Since the analysis is similar to the approach from [Rab80] and [MS83], we will only give a sketch of the proof.

The proof will use the following basic facts from the theory of finite fields: (1) The roots of $x^{2^n} - x$ are precisely the elements of the field $\mathbb{F}_{2^n}$, each occurring with multiplicity 1. (2) $x^{2^n} - x$ is divisible by an irreducible polynomial $g(x)$ of degree m if and only if m divides n.

If $f(x)$ is irreducible, then the oracle call in step (i) will succeed, returning $a_i \equiv x^{2^i} \pmod{f(x)}$; step (ii) will succeed because of the assignment of the a_i's and because $a_{n-1}^2 \equiv (x^{2^{n-1}})^2 = x^{2^n} \equiv x \pmod{f(x)}$ for any irreducible $f(x)$ (since $x^{2^n} - x$ is divisible by every irreducible polynomial of degree n); finally, step (iii) succeeds because an irreducible $f(x)$ of degree n cannot divide $x^{2^m} - x$ for any $m < n$.

On the other hand, if $f(x)$ is reducible and steps (i) and (ii) succeed, then we know that $a_i \equiv x^{2^i} \pmod{f(x)}$, and moreover that $x^{2^n} \equiv x \pmod{f(x)}$. This guarantees that $f(x)$ divides $x^{2^n} - x$ and therefore that $f(x)$ is square-free and has all its roots in $\mathbb{F}_{2^n}$. Let $f(x) = h_1(x) \cdots h_\ell(x)$ be the factorization of f into distinct irreducibles $h_i(x)$. To show that step (iii) succeeds, it suffices, by the Chinese Remainder Theorem, to show that the product $\prod_{p|n}(x^{2^{n/p}} - x)$ is divisible by each $h_i(x)$. Fix an irreducible factor $h(x)$ of $f(x)$. The degree of $h(x)$ must divide n (since $f(x)$ divides $x^{2^n} - x$), and hence must divide some maximal proper divisor n/p of n. Therefore, $h(x)$ will divide $(x^{2^{n/p}} - x)$ for some prime $p \mid n$, and the product in part (iii) will be divisible by $h(x)$. This concludes the proof of correctness.

Next we argue that the reduction is computable by $\mathbf{AC}^0[\oplus]$ circuits of size $\mathrm{poly}(n)$: Step (i) is simply an oracle query. Each check in step (ii) can be accomplished in parallel using modular multiplication (which follows from the iterated product in Theorem 5.2.7 part 1, together with Lemma 5.4.1). Each term in the product from step (iii) can be computed using the oracle responses to compute $x^{2^{n/p}}$; since there are at most $O(\log n)$ primes dividing n, step (iii) is an iterated product of $O(\log n)$ polynomials which can be computed by $\mathbf{AC}^0[\oplus]$ circuits of size $\mathrm{poly}(n)$ by Theorem 5.2.7, provided that the list of primes p dividing n is known. This list of primes can either be hard-wired into the circuit (for a non-uniform reduction) or can be shown to be computable in uniform $\mathbf{AC}^0[\oplus]$ by a more complicated proof, which we omit. Finally, the answer can be reduced modulo $f(x)$ using Lemma 5.4.1. □

5.5 Proof of *AE* = *Dlogtime* uniform $\mathbf{TC}^0$

In this section we prove that *AE* = *Dlogtime* uniform $\mathbf{TC}^0$. First we exhibit a dual basis for $\tilde{\mathbb{F}}_{2^n}$. Recall that the *trace* function $\mathrm{tr} : \mathbb{F}_{2^n} \to \mathbb{F}_2$ is defined by $\mathrm{tr}(\alpha) \stackrel{\mathrm{def}}{=} \sum_{i=0}^{n-1} \alpha^{2^i}$. Also recall that two bases $(\alpha_0, \alpha_1, \ldots, \alpha_{n-1})$ and $(\beta_0, \beta_1, \ldots, \beta_{n-1})$ are *dual* if for every i, j we have that $\mathrm{tr}(\alpha_i \cdot \beta_j) = 1$ when $i = j$, while $\mathrm{tr}(\alpha_i \cdot \beta_j) = 0$ when $i \neq j$.

Lemma 5.5.1. *Let n be of the form $n = 2 \cdot 3^l$. Let $(\alpha_0, \alpha_1, \ldots, \alpha_{n-1}) = (1, x, \ldots, x^{n-1})$ be the standard basis for $\tilde{\mathbb{F}}_{2^n}$. Then $(\beta_0, \beta_1, \ldots, \beta_{n-1}) = (x^{3^l}, x^{3^l-1}, \ldots, x^{3^l-(n-1)})$ is the dual basis of $(\alpha_0, \alpha_1, \ldots, \alpha_{n-1})$.*

The proof of this lemma follows immediately from the next lemma.

Lemma 5.5.2. *Let $x \in \tilde{\mathbb{F}}_{2^n}$ be a root of $x^{2 \cdot 3^\ell} + x^{3^\ell} + 1$. Then, for any $0 \leq i < 3^{\ell+1}$, we have $tr(x^i) = 1$ if $i = 3^\ell$ or $i = 2 \cdot 3^\ell$, and $tr(x^i) = 0$ otherwise.*

Proof. Recall that $\mathrm{tr}(x^i) = \sum_{k=0}^{2 \cdot 3^\ell - 1} x^{i \cdot 2^k}$. Thus, if $i = 0$, then $\mathrm{tr}(x^i) = \mathrm{tr}(1) = \sum_{k=0}^{2 \cdot 3^\ell - 1} 1^{2^k} = (2 \cdot 3^\ell) \cdot 1 \equiv 0 \mod 2$.

Now suppose that $0 < i < 3^{\ell+1}$, and let 3^m be the largest power of 3 dividing i. We will show that $\mathrm{tr}(x^i) = 1$ if $m = \ell$ and $\mathrm{tr}(x^i) = 0$ otherwise.

Since 2 is a generator of $\mathbb{Z}_{3^t}^\times$ for any integer $t > 0$ (e.g., [van99] Lemma 1.1.27), and since x has order $3^{\ell+1}$ by Fact 5.3.1, we know that the exponent, $i \cdot 2^k$, of x will take on every value in the multiset $3^m\mathbb{Z}_{3^{\ell+1}}^\times$ (with multiplicities) as k ranges from 0 to $\varphi(3^{\ell+1}) - 1 = 2 \cdot 3^\ell - 1$. Therefore, we have

$$\mathrm{tr}(x^i) \equiv \sum_{\substack{r=0 \\ (r,3)=1}}^{3^{\ell+1}-1} x^{3^m \cdot r} = \sum_{r=0}^{3^{\ell+1}-1} x^{3^m \cdot r} - \sum_{r=0}^{3^\ell-1} x^{3^{m+1} \cdot r} = \frac{1 - (x^{3^m})^{3^{\ell+1}}}{1 - x^{3^m}} - \sum_{r=0}^{3^\ell-1} x^{3^{m+1} \cdot r} \equiv \sum_{r=0}^{3^\ell-1} x^{3^{m+1} \cdot r},$$

where we use that the denominator is non-zero because $m < \ell + 1$ and x has order $3^{\ell+1}$.

If $m = \ell$, then every term of the sum is 1, and so this is $3^\ell \equiv 1 \mod 2$. On the other hand, if $m < \ell$, then

$$\sum_{r=0}^{3^\ell-1} x^{3^{m+1} \cdot r} = \frac{1 - (x^{3^{m+1}})^{3^{\ell+1}}}{1 - x^{3^{m+1}}} = 0.$$

□

Theorem 5.2.10 (restated)**.** *AE = Dlogtime−uniform* $\mathbf{TC}^0$ *in the following sense:*

Let $f : \{0,1\}^n \to \{0,1\}^n$ *be in Dlogtime-uniform* $\mathbf{TC}^0$*. Then there is* $f' : \tilde{\mathbb{F}}_{2^n} \to \tilde{\mathbb{F}}_{2^n}$ *in AE such that for every* n *of the form* $2 \cdot 3^l$*, and for every* x *of length* n*,* $f(x) = f'(x)$*.*

Conversely, let $f : \tilde{\mathbb{F}}_{2^n} \to \tilde{\mathbb{F}}_{2^n}$ *be in AE. Then there is* $f' : \{0,1\}^n \to \{0,1\}^n$ *in Dlogtime-uniform* $\mathbf{TC}^0$ *such that for every* n *of the form* $2 \cdot 3^l$*, and for every* x *of length* n*,* $f(x) = f'(x)$*.*

Proof of Theorem 5.2.10. We use the following result from [FVB94]: *AE* is equivalent to the class of functions computed by *Dlogtime*-uniform *arithmetic* circuits of polynomial size and constant-depth. Where an arithmetic circuit is a circuit with gates for the constant field element x, unbounded fan-in sum, unbounded fan-in product, and single input conjugation (this gate computes $\alpha \to \alpha^{2^j}$ for $0 \le j \le n$). We refer the reader to Definition 2.1 in [FVB94] for more on arithmetic circuits. (While their equivalence is proved for a slightly different notion of *AE* and arithmetic circuit, it can be verified that it also applies to ours.)

AE $\subseteq$ *Dlogtime*−uniform $\mathbf{TC}^0$: By the equivalence above, it is enough to show that any function computed by a *Dlogtime*−uniform arithmetic circuits of polynomial size and constant-depth is computable in *Dlogtime*−uniform $\mathbf{TC}^0$. This follows by replacing the gates of the uniform circuits with the corresponding circuits as given by Theorem 5.2.3 (the iterated sum is not stated in the theorem but can be easily computed with XOR gates).

Dlogtime−uniform $\mathbf{TC}^0 \subseteq AE$: To understand the proof of this inclusion, we first need to discuss an issue about interpretations of bit strings as field elements. Throughout the work, and in particular in the statement of the theorem we are proving, we have interpreted a n-bit string as a field element in a field of size 2^n. Let us call this interpretation (1). Another possible interpretation, which we denote (2), is to interpret a n-bit string as a *tuple* of n field elements, the i-th field element being 0 or 1 according to the i-th bit in the string. Lemma 2.2 in [FVB94] proves the inclusion *Dlogtime*−uniform $\mathbf{TC}^0 \subseteq AE$ under interpretation (2) (to make sense of this one extends in the natural way the definition of AE to include functions mapping *tuples* of field elements to *tuples* of field elements). To prove the inclusion under interpretation (1), and thus concluding the proof of the theorem, we show how to convert back and forth between interpretations (1) and (2) in AE. Converting from (2) to (1) is relatively simple, and we omit the details that can be found in [FVB94]. To convert from (1) to (2), following [BFS92, FVB94], we use a *dual basis* for $\tilde{\mathbb{F}}_{2^n}$. Specifically, let $(\alpha_0, \alpha_1, \ldots, \alpha_{n-1}) = (1, x, \ldots, x^{n-1})$ be the standard basis for $\tilde{\mathbb{F}}_{2^n}$. In other words we view an input $(c_0, c_1, \ldots, c_{n-1}) \in \{0,1\}^n$ as the field element $\gamma = \sum_i c_i \alpha_i \in \tilde{\mathbb{F}}_{2^n}$. Now let $(\beta_0, \beta_1, \ldots, \beta_{n-1}) = (x^{3^l}, x^{3^l-1}, \ldots, x^{3^l-(n-1)})$ be the dual basis of $(\alpha_0, \alpha_1, \ldots, \alpha_{n-1})$ as given by Lemma 5.5.1. It follows from the definition of dual basis that $c_i = \mathrm{tr}(\beta_i \cdot \gamma)$. Therefore to convert from interpretation (1) to (2) is enough to note that $\mathrm{tr}(\beta_i \cdot \gamma)$ can be computed by a *Dlogtime*-uniform arithmetic circuit, and thus is in AE by the result from [FVB94] mentioned at the beginning of this proof. □

5.6 Proof of k-wise and ϵ-biased generator constructions

Theorem 5.2.13 (restated)**.**

1. *For every k and m there is a k-wise independent generator $G : \{0,1\}^s \to \{0,1\}^m$, with $s = O(k \log m)$ that is bitwise computable by uniform $\mathbf{AC}^0[\oplus]$ circuits of size $\mathrm{poly}(s, \log m) = \mathrm{poly}(s)$.*

2. *For every ϵ and m, there is an ϵ-biased generator $G : \{0,1\}^s \to \{0,1\}^m$ with $s = O(\log m + \log(1/\epsilon))$ that is bitwise computable by uniform $\mathbf{TC}^0$ circuits of size $\mathrm{poly}(s, \log m) = \mathrm{poly}(s)$.*

Proof of Theorem 5.2.13. (1) We use the following construction from [CG89, ABI86]. Let $h = O(\log m)$ be the smallest integer bigger than $\log(m)$ of the form $h = 2 \cdot 3^l$ for some l. The generator $G : \{0,1\}^s \to \{0,1\}^m$ is defined as

$$G(\alpha_0, \alpha_1, \ldots, \alpha_{k-1})_i \stackrel{\mathrm{def}}{=} \sum_{j<k} \alpha_j \cdot i^j, \text{ where } \alpha_0, \alpha_1, \ldots, \alpha_{k-1}, i \in \tilde{\mathbb{F}}_{2^h},$$

is a k-wise independent generator. This generator is computable by uniform $\mathbf{AC}^0[\oplus]$ circuits of size $\text{poly}(s, \log m)$ by Theorem 5.2.4.

(2) We use the following construction from [AGHP92]. Let $h = O(\log m + \log(2/\epsilon))$ be the smallest integer bigger than $\log(m) + \log(2/\epsilon)$ of the form $h = 2 \cdot 3^l$ for some l. The generator $G : \{0,1\}^s \to \{0,1\}^m$ defined as

$$G(\alpha, \beta)_i \stackrel{\text{def}}{=} \langle \alpha^i, \beta \rangle \text{ where } \alpha, \beta \in \tilde{\mathbb{F}}_{2^h},$$

is an ϵ-biased generator. (Where $\langle \cdot, \cdot \rangle$ denotes inner product mod 2.) This generator is computable by uniform $\mathbf{TC}^0$ circuits of size $\text{poly}(s, \log m)$ by Item 2 in Theorem 5.2.3. □

5.7 Open Problems

Given $\alpha \in \tilde{\mathbb{F}}_{2^n}$, can α^{-1} be computed by uniform $\mathbf{AC}^0[\oplus]$ circuits of size $\text{poly}(n)$?

Given an irreducible polynomial $f(x)$ of degree n and $\alpha \in \mathbb{F}_2[x]/(f(x))$, is it possible to compute α^{2^i} for any $i = \omega(\log n)$ by uniform $\mathbf{TC}^0$ circuits of size $\text{poly}(n)$? (cf. Lemma 5.3.3)? This is what limits our results about exponentiation in $\mathbb{F}_2[x]/(f(x))$.

Both of the above problems are also open for nonuniform circuits.

Bibliography

[AAI+01] Manindra Agrawal, Eric Allender, Russell Impagliazzo, Toniann Pitassi, and Steven Rudich. Reducing the complexity of reductions. *Computational Complexity*, 10(2):117–138, 2001.

[ABD+03] Eric Allender, Anna Bernasconi, Carsten Damm, Joachim von zur Gathen, Michael Saks, and Igor Shparlinski. Complexity of some arithmetic problems for binary polynomials. *Computational Complexity*, 12(1-2):23–47, 2003.

[ABI86] N. Alon, L. Babai, and A. Itai. A fast and simple randomized algorithm for the maximal independent set problem. *Journal of Algorithms*, 7:567–583, 1986.

[ABO84] Miklós Ajtai and Michael Ben-Or. A theorem on probabilistic constant depth computation. In *Proceedings of the 16th Annual ACM Symposium on Theory of Computing*, pages 471–474, April 30 – May 2 1984.

[AGHP92] Noga Alon, Oded Goldreich, Johan Håstad, and René Peralta. Simple constructions of almost k-wise independent random variables. *Random Structures & Algorithms*, 3(3):289–304, 1992.

[Ajt93] Miklós Ajtai. Approximate counting with uniform constant-depth circuits. In *Advances in computational complexity theory*, pages 1–20. Amererican Mathematical Society, 1993.

[AKS83] M. Ajtai, J. Komlos, and E. Szemeredi. An O(n log n) sorting network. *Combinatorica*, 3:1–19, 1983.

[AKS87] M. Ajtai, J. Komlos, and E. Szemeredi. Deterministic simulation in LOGSPACE. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*, pages 132–140, May 25–27 1987.

[And02] Ian Anderson. *Combinatorics of finite sets*. Dover Publications Inc., Mineola, NY, 2002. Corrected reprint of the 1989 edition.

[AR94] Noga Alon and Yuval Roichman. Random cayley graphs and expanders. *Random Structures & Algorithms*, 5:271–284, 1994.

[AS00] Noga Alon and Joel H. Spencer. *The Probabilistic Method*. Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley and Sons, Inc., 2000.

[BF90] Donald Beaver and Joan Feigenbaum. Hiding instances in multioracle queries. In *Proceedings of the 7th Annual Symposium on Theoretical Aspects of Computer Science*, volume 415 of *Lecture Notes in Computer Science*, pages 37–48. Springer, February 22–24 1990.

[BFL91] László Babai, Lance Fortnow, and Carsten Lund. Nondeterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1(1):3–40, 1991.

[BFNW93] László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3(4):307–318, 1993.

[BFS92] Joan Boyar, Gudmund Frandsen, and Carl Sturtivant. An arithmetic model of computation equivalent to threshold circuits. *Theoretical Computer Science*, 93(2):303–319, 1992.

[BGG93] M. Bellare, O. Goldreich, and S. Goldwasser. Randomness in interactive proofs. *Computational Complexity*, 4(1):319–354, 1993.

[BIS90] David A. Mix Barrington, Neil Immerman, and Howard Straubing. On uniformity within NC^1. *Journal of Computer and System Sciences*, 41(3):274–306, 1990.

[BL90] Michael Ben-Or and Nathan Linial. Collective coin-flipping. In Silvio Micali, editor, *Randomness and Computation*, pages 91–115. Academic Press, New York, 1990.

[BT03] Andrej Bogdanov and Luca Trevisan. On worst-case to average-case reductions for NP problems. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, October 11–14 2003.

[BYGW99] Z. Bar-Yossef, O. Goldreich, and A. Wigderson. Deterministic amplification of space-bounded probabilistic algorithms. In *Proceedings of the 14th Annual IEEE Conference on Computational Complexity*, pages 188–198, June 1999.

[CEG95] Ran Canetti, Guy Even, and Oded Goldreich. Lower bounds for sampling algorithms for estimating the average. *Information Processing Letters*, 53(1):17–25, 1995.

[CG89] Benny Chor and Oded Goldreich. On the power of two-point based sampling. *Journal of Complexity*, 5(1):96–106, March 1989.

[CGH+85] B. Chor, O. Goldreich, J. Hastad, J. Friedman, S. Rudich, and R. Smolensky. The bit extraction problem and t-resilient functions. In *Proceedings of the 26th Annual IEEE Symposium on Foundations of Computer Science*, pages 396–407, October 21–23 1985.

[CPS99] Jin-Yi Cai, A. Pavan, and D. Sivakumar. On the hardness of the permanent. In *Proceedings of the the 16th Annual Symposium on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science. Springer-Verlag, 1999.

[CW89] Aviad Cohen and Avi Wigderson. Dispersers, deterministic amplification, and weak random sources. In *Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science*, pages 14–19, October 30 – November 1 1989.

[Din95] I. H. Dinwoodie. A probability inequality for the occupation measure of a reversible Markov chain. *Annals of Applied Probability*, 5(1):37–43, 1995.

[Ebe89] W. Eberly. Very fast parallel polynomial arithmetic. *SIAM Journal on Computing*, 18(5):955–976, 1989.

[FF93] Joan Feigenbaum and Lance Fortnow. Random-self-reducibility of complete sets. *SIAM Journal on Computing*, 22(5):994–1005, October 1993.

[Fil91] J. A. Fill. Eigenvalue bounds on convergence to stationarity for nonreversible Markov chains with an application to the exclusion process. *Annals of Applied Probability*, 1:62–87, 1991.

[FK06] Lance Fortnow and Adam Klivans. Linear advice for randomized logarithmic space. In *Proceedings of the 23rd Annual Symposium on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science, pages 469 – 476. Springer, February 23–25 2006.

[FL96] Uriel Feige and Carsten Lund. On the hardness of computing the permanent of random matrices. *Computational Complexity*, 6(2):101–132, 1996.

[For03] Lance Fortnow. Balanced NP sets. *Computational Complexity Weblog*, 12 September 2003. `http://weblog.fortnow.com/archive/2003_09_07_archive.html`.

[FSS84] Merrick L. Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, April 1984.

[FVB94] Gudmund S. Frandsen, Mark Valence, and David A. Mix Barrington. Some results on uniform arithmetic circuit complexity. *Mathematical Systems Theory*, 27(2):105–124, 1994.

[GG81] O. Gabber and Z. Galil. Explicit construction of linear size superconcentrators. *Journal of Computer and System Sciences*, 22:407–420, 1981.

[GIL+90] Oded Goldreich, Russell Impagliazzo, Leonid A. Levin, Ramarathnam Venkatesan, and David Zuckerman. Security preserving amplification of hardness. In *Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science*, pages 318–326, 1990.

[Gil94] David Gillman. A Chernoff bound for random walks on expander graphs. In *Proceedings of the 34th Annual IEEE Symposium on Foundations of Computer Science*, pages 680–691, 1994.

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 25–32, May 15–17 1989.

[GNW95] Oded Goldreich, Noam Nisan, and Avi Wigderson. On Yao's XOR lemma. *Electronic Colloquium on Computational Complexity (ECCC)*, (TR95–050), March 1995. `http://www.eccc.uni-trier.de/eccc`.

[Gol97] Oded Goldreich. A sample of samplers - a computational perspective on sampling (survey). *Electronic Colloquium on Computational Complexity (ECCC)*, (TR97-020), May 1997.

[Gol99] Oded Goldreich. *Modern cryptography, probabilistic proofs and pseudorandomness*, volume 17 of *Algorithms and Combinatorics*. Springer-Verlag, Berlin, 1999.

[Gol01] Oded Goldreich. *Foundations of Cryptography. Volume 1 - Basic Techniques*. Cambridge University Press, Cambridge, 2001.

[GV04] Dan Gutfreund and Emanuele Viola. Fooling parity tests with parity gates. In *Proceedings of the 8th International Workshop on Randomization and Computation (RANDOM)*, volume 3122 of *Lecture Notes in Computer Science*, pages 381–392, August 22–24 2004.

[HAB02] William Hesse, Eric Allender, and David A. Mix Barrington. Uniform constant-depth threshold circuits for division and iterated multiplication. *Journal of Computer and System Sciences*, 65(4):695–716, 2002. Special issue on Complexity, 2001 (Chicago, IL).

[Hås87] Johan Håstad. *Computational limitations of small-depth circuits*. MIT Press, 1987.

[Hea06] Alexander Healy. Randomness-efficient sampling within $\mathbf{NC}^1$. In *Proceedings of the 10th International Workshop on Randomization and Computation (RANDOM)*, volume 4110 of *Lecture Notes in Computer Science*, pages 398 – 409, August 28–30 2006.

[HLW06] S. Hoory, N. Linial, and A. Wigderson. Expander graphs and their applications. *Bulletin of the American Mathematical Society*, 43(4):439–561, 2006.

[HV06] Alexander Healy and Emanuele Viola. Constant-depth circuits for arithmetic in finite fields of characteristic two. In *Proceedings of the 23rd Annual Symposium on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science, pages 672 – 683. Springer, February 23–25 2006.

[HVV04] Alexander Healy, Salil Vadhan, and Emanuele Viola. Using nondeterminism to amplify hardness. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing*, pages 192–201, June 13–15 2004. Invited to *SIAM Journal of Computing*, STOC Special Issue.

[HVV06] Alexander Healy, Salil Vadhan, and Emanuele Viola. Using nondeterminism to amplify hardness. *SIAM Journal on Computing*, 35(4):903–931, 2006. Special Issue on STOC 2004.

[Imp95a] Russell Impagliazzo. Hard-core distributions for somewhat hard problems. In *Proceedings of the 36th Annual IEEE Symposium on Foundations of Computer Science*, pages 538–545, October 23–25 1995.

[Imp95b] Russell Impagliazzo. A personal view of average-case complexity. In *Proceedings of the Tenth Annual Structure in Complexity Theory Conference*, pages 134–147. IEEE, 19–22 June 1995.

[IW97] Russell Impagliazzo and Avi Wigderson. $P = BPP$ if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 220–229, May 4–6 1997.

[IW01] Russell Impagliazzo and Avi Wigderson. Randomness vs time: derandomization under a uniform assumption. *Journal of Computer and System Sciences*, 63(4):672–688, 2001. Special issue on FOCS 98 (Palo Alto, CA).

[IZ89] Russell Impagliazzo and David Zuckerman. How to recycle random bits. In *Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science*, pages 248–253, October 30 – November 1 1989.

[Kah97] N. Kahale. Large deviation bounds for Markov chains. *Combinatorics, Probability and Computing*, 6(4):465–474, 1997.

[KKL88] Jeff Kahn, Gil Kalai, and Nathan Linial. The influence of variables on Boolean functions (extended abstract). In *Proceedings of the 29th Annual IEEE Symposium on Foundations of Computer Science*, pages 68–80, October 24–26 1988.

[KS03] Adam Klivans and Rocco A. Servedio. Boosting and hard-core sets. *Machine Learning*, 53(3):217–238, 2003.

[Kun74] H. T. Kung. On computing reciprocals of power series. *Numerical Math*, 22:341–348, 1974.

[Lǵ98] P. Lézaud. Chernoff-type bound for finite Markov chains. *Annals of Applied Probability*, 8(3):849–867, 1998.

[Lev86] Leonid A. Levin. Average case complete problems. *SIAM Journal on Computing*, 15(1):285–286, February 1986.

[Lip89] Richard Lipton. New directions in testing. In *Proceedings of DIMACS Workshop on Distributed Computing and Cryptography*, 1989.

[LP04] Carlos A. León and François Perron. Optimal Hoeffding bounds for discrete reversible Markov chains. *Annals of Applied Probability*, 14(2):958–970, 2004.

[LPS88] A. Lubotzky, R. Phillips, and P. Sarnak. Ramanujan graphs. *Combinatroica*, 8(3):261–277, 1988.

[LTW05] Chi-Jen Lu, Shi-Chun Tsai, and Hsin-Lung Wu. On the complexity of hardness amplification. In *Proceedings of the 20th Annual IEEE Conference on Computational Complexity*, June 12–15 2005.

[Mar73] G. A. Margulis. Explicit constructions of expanders. *Problemy Peredachi Informatssi; English translation, Problems of Information Transmission*, 9(4):71–80, 1973.

[Mih89] M. Mihail. Conductance and convergence of Markov chains: a combinatorial treatment of expanders. In *Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science*, pages 526–531, October 30 – November 1 1989.

[MNT90] Yishay Mansour, Noam Nisan, and Prasoon Tiwari. The computational complexity of universal hashing. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 235–243, May 14–16 1990.

[MO03] Elchanan Mossel and Ryan O'Donnell. On the noise sensitivity of monotone functions. *Random Structures & Algorithms*, 23(3), 2003.

[MR95] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[MS83] Moshe Morgensteren and Eli Shamir. Parallel algorithms for arithmetics, irreducibility and factoring of GFq-polynomials. Stanford University Technical Report STAN-CS-83-991, December 1983.

[Nis91] Noam Nisan. Pseudorandom bits for constant depth circuits. *Combinatorica*, 11(1):63–70, 1991.

[Nis92] Noam Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12, 1992.

[NN90] J. Naor and M. Naor. Small-bias probability spaces: efficient constructions and applications. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, pages 213–223, May 14–16 1990.

[NW94] Noam Nisan and Avi Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49(2):149–167, October 1994.

[O'D04] Ryan O'Donnell. Hardness amplification within NP. *Journal of Computer and System Sciences*, 69(1):68–94, August 2004.

[Rab80] Michael O. Rabin. Probabilistic algorithms in finite fields. *SIAM Journal on Computing*, 9(2):273–280, 1980.

[Raz87] Alexander A. Razborov. Lower bounds on the dimension of schemes of bounded depth in a complete basis containing the logical addition function. *Akademiya Nauk SSSR. Matematicheskie Zametki*, 41(4):598–607, 623, 1987.

[Rei86] J. Reif. Logarithmic depth circuits for algebraic functions. *SIAM Journal on Computing*, 15(1):231–242, 1986.

[Rei05] Omer Reingold. Undirected st-connectivity in log-space. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 376–385, May 21–24 2005.

[RVW02] 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, January 2002.

[Sak96] Michael Saks. Randomization and derandomization in space-bounded computation. In *Proceedings of the 11th Annual IEEE Conference on Computational Complexity*, pages 128–149, May 24–27 1996.

[SF93] Carl Sturtivant and Gudmund Skovbjerg Frandsen. The computational efficacy of finite-field arithmetic. *Theoretical Computer Science*, 112(2):291–309, 1993.

[Sha02] R. Shaltiel. Recent developments in explicit constructions of extractors. *Bulletin of the European Association for Theoretical Computer Science*, (77):67–95, 2002. Columns: Computational Complexity.

[Sie72] M. Sieveking. An algorithm for division of power series. *Computing*, 10:153–156, 1972.

[Smo87] Roman Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*, pages 77–82, May 25–27 1987.

[STV01] Madhu Sudan, Luca Trevisan, and Salil Vadhan. Pseudorandom generators without the XOR lemma. *Journal of Computer and System Sciences*, 62(2):236–266, 2001. Special issue on the Fourteenth Annual IEEE Conference on Computational Complexity (Atlanta, GA, 1999).

[SU00] Ronen Shaltiel and Christopher Umans. Simple extractors for all min-entropies and a new pseudo-random generator. In *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science*, October 14–17 2000.

[Tre03] Luca Trevisan. List decoding using the XOR lemma. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, October 11–14 2003.

[TV02] Luca Trevisan and Salil Vadhan. Pseudorandomness and average-case complexity via uniform reductions. In *Proceedings of the 17th Annual IEEE Conference on Computational Complexity*, pages 129–138, May 2002.

[Uma02] Christopher Umans. Pseudo-random generators for all hardnesses. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, May 19–21 2002.

[Val77] Leslie G. Valiant. Graph-theoretic arguments in low-level complexity. In *Mathematical foundations of computer science (Tatranská Lomnica, 1977)*, volume 53 of *Lecture Notes in Computer Science*, pages 162–176. Springer, Berlin, 1977.

[van99] J. H. van Lint. *Introduction to coding theory*. Springer-Verlag, Berlin, third edition, 1999.

[Vio04] Emanuele Viola. The complexity of constructing pseudorandom generators from hard functions. *Computational Complexity*, 13(3-4):147–188, 2004.

[Vio05a] Emanuele Viola. On constructing parallel pseudorandom generators from one-way functions. In *Proceedings of the 20th Annual IEEE Conference on Computational Complexity*, June 12–15 2005.

[Vio05b] Emanuele Viola. Pseudorandom bits for constant-depth circuits with few arbitrary symmetric gates. In *Proceedings of the 20th Annual IEEE Conference on Computational Complexity*, June 12–15 2005.

[Vol99] Heribert Vollmer. *Introduction to circuit complexity*. Springer-Verlag, Berlin, 1999.

[WX05] Avi Wigderson and David Xiao. A randomness-efficient sampler for matrix-valued functions and applications. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, October 22–25 2005. See also Electronic Colloquium on Computational Complexity (ECCC) Technical Report TR05-107, `http://eccc.hpi-web.de/eccc/`.

[WX06] Avi Wigderson and David Xiao. Derandomizing the AW matrix-valued Chernoff bound using pessimistic estimators and applications. *Electronic Colloquium on Computational Complexity (ECCC)*, (TR06-105), August 2006.

[Yao82] Andrew C. Yao. Theory and applications of trapdoor functions (extended abstract). In *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science*, pages 80–91, November 3–5 1982.

[Zuc97] David Zuckerman. Randomness-optimal oblivious sampling. *Random Structures & Algorithms*, 11(4):345–367, 1997.

[Zuc06] David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, May 21–23 2006.