Paradigms for Unconditional Pseudorandom Generators

Pooya Hatami[1] and William Hoza[2]

[1] *The Ohio State University, USA; pooyahat@gmail.com*
[2] *The University of Chicago, USA; williamhoza@uchicago.edu*

ABSTRACT

This is a survey of unconditional *pseudorandom generators* (PRGs). A PRG uses a short, truly random seed to generate a long, "pseudorandom" sequence of bits. To be more specific, for each restricted model of computation (e.g., bounded-depth circuits or read-once branching programs), we would like to design a PRG that "fools" the model, meaning that every function computable in the model behaves approximately the same when we plug in pseudorandom bits from the PRG as it does when we plug in truly random bits. In this survey, we discuss four major paradigms for designing PRGs:

- We present several PRGs based on k-wise uniform generators, small-bias generators, and simple combinations thereof, including proofs of Viola's theorem on fooling low-degree polynomials [242] and Braverman's theorem on fooling $\mathbf{AC}^0$ circuits [36].
- We present several PRGs based on "recycling" random bits to take advantage of communication bottlenecks, such as the Impagliazzo-Nisan-Wigderson generator [131].

Pooya Hatami and William Hoza (2024), "Paradigms for Unconditional Pseudorandom Generators", Foundations and Trends® in Theoretical Computer Science: Vol. 16, No. 1-2, pp 1–210. DOI: 10.1561/0400000109.

- We present connections between PRGs and computational hardness, including the Nisan-Wigderson framework for converting a hard Boolean function into a PRG [183].
- We present PRG frameworks based on random restrictions, including the "polarizing random walks" framework [49].

We explain how to use these paradigms to construct PRGs that work *unconditionally*, with no unproven complexity-theoretic assumptions. The PRG constructions use ingredients such as finite field arithmetic, expander graphs, and randomness extractors. The analyses use techniques such as Fourier analysis, sandwiching approximators, and simplification-under-restrictions lemmas.

1

Introduction

To make random choices, it would be useful to have an unlimited supply of "truly random" bits: unbiased and independent coin flips. What can we do if we only have a few truly random bits? A *pseudorandom generator* (PRG) uses a small amount of true randomness, called the "seed," to generate a long sequence that appears to be completely random (even though it isn't). PRGs are ubiquitous in computing theory and practice. The basic motivation is that we think of randomness as a scarce computational resource, akin to time or space, so whenever we get our hands on some random bits, we want to stretch them as far as possible.

To model PRGs mathematically, we consider some "observer," modeled as a function f. Let U_n denote the uniform distribution over $\{0,1\}^n$. We would like to "fool" f in the following sense.

Definition 1.1 (Fooling)**.** Suppose $f\colon \{0,1\}^n \to \{0,1\}$ is a function, X is a probability distribution over $\{0,1\}^n$, and $\varepsilon > 0$. We say that X *fools* f *with error* ε, or ε*-fools* f, if

$$|\Pr[f(X) = 1] - \Pr[f(U_n) = 1]| \leq \varepsilon.$$

More generally, we can consider a real-valued function $f\colon \{0,1\}^n \to \mathbb{R}$. In this case, we say that X fools f with error ε if

$$|\,\mathbb{E}[f(X)] - \mathbb{E}[f(U_n)]| \leq \varepsilon.$$

If $\varepsilon = 0$, we say that X *perfectly* fools f.

Remark 1.1. As a shorthand, we often identify the function f with the random variable $f(U_n)$. For example, instead of $\mathbb{E}[f(U_n)]$, we simply write $\mathbb{E}[f]$.

Definition 1.1 says that although X might not be uniform, X and U_n are nevertheless *indistinguishable*, at least from f's perspective. Conversely, if X does *not* ε-fool f, we refer to f as a "distinguisher" for X. A PRG's job is to use a few truly random bits to sample a distribution that fools f.

Definition 1.2 (PRGs)**.** Suppose $f\colon \{0,1\}^n \to \mathbb{R}$ and $G\colon \{0,1\}^s \to \{0,1\}^n$ are functions and $\varepsilon > 0$. We say that G is an *ε-PRG for f* if $G(U_s)$ fools f with error ε. In this case, we also say that G fools f with error ε (see Figure 1.1.)

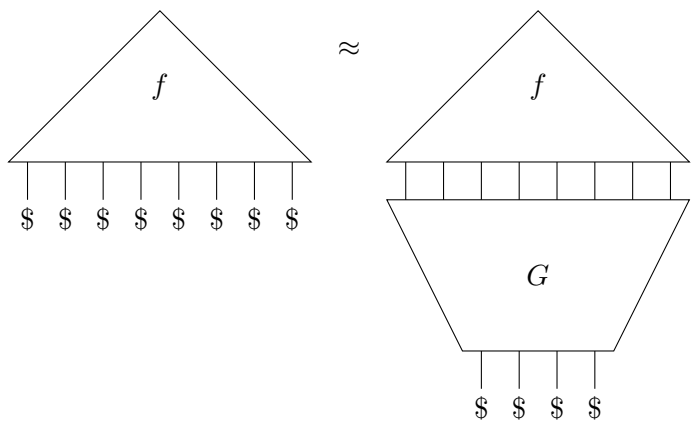

Figure 1.1: A PRG (G) uses a few truly random bits (depicted here using \$ symbols) to sample a pseudorandom string that is indistinguishable from a truly random string, from the perspective of the observer (f).

The parameter s is called the *seed length* of the PRG; we would like s to be as small as possible. Throughout this text, the parameter "n" will always denote the number of pseudorandom bits we are generating.

1.1 Whom Shall We Fool? Three Approaches to PRGs

An unavoidable fact of life is that for any nontrivial PRG, there exists a function that is not fooled by the PRG.

Claim 1.1 (Impossibility of fooling all functions)**.** Let $G\colon \{0,1\}^s \to \{0,1\}^n$ where $s < n$. There exists some $f\colon \{0,1\}^n \to \{0,1\}$ such that G does not 0.49-fool f.

Proof. Let f be the indicator function for the image of G. Then $\mathbb{E}[f(G(U_s))] = 1$, whereas $\mathbb{E}[f] \leq 1/2$ because $s < n$. □

In light of Claim 1.1, the best we can hope for is generating bits that fool some large sets of observers but not all of them. After all, as Avi Wigderson says, *randomness is in the eye of the beholder* [248].

Definition 1.3 (PRG for a class of functions)**.** Let $n \in \mathbb{N}$, let $\mathcal{F}$ be a class of functions $f\colon \{0,1\}^n \to \mathbb{R}$, let $G\colon \{0,1\}^s \to \{0,1\}^n$ be a function, and let $\varepsilon > 0$. We say that G is an *ε-PRG for $\mathcal{F}$* if G fools every $f \in \mathcal{F}$ with error ε.

Which observers shall we fool? The study of PRGs can be crudely divided into three approaches based on three possible answers:

1. Everyday non-adversarial applications.
2. All efficient observers.
3. Restricted models of computation.

We discuss these three approaches in Sections 1.1.1 to 1.1.3.

1.1.1 PRGs for everyday non-adversarial applications

In practice, when programmers want randomness, they invoke some type of `random()` method provided by the computing environment. Under the hood, these `random()` methods typically involve several components, each of which might be quite sophisticated. When practitioners speak of "pseudorandom number generators" or "random number generators," they are usually referring to the entire randomness system as a whole,

including whatever techniques are used to produce an initial seed. For example, the system might derive a seed from the current time of day, even though such a seed is rather predictable. As another example, the system might use hardware random number generators based on thermal noise measurements.

In this text, we sidestep the important issue of producing a seed, along with many other issues that are important in practice. We focus on the challenge of stretching a truly random seed out to a long pseudorandom string. In our terminology, this is the job of a PRG (see Definition 1.2). A PRG is thus one of multiple components of a practical randomness system. For example, Java's `Math.random()` method currently uses a type of PRG called a *linear congruential generator*. For such a PRG, the seed is a random number $X_0 \in \{0, 1, \ldots, M-1\}$, and the output sequence is $(X_1, X_2, X_3, \ldots)$, where

$$X_{i+1} = a \cdot X_i + b \bmod M$$

for some parameters M, a, b. Meanwhile, Python's `random.random()` method uses an algorithm called the "Mersenne twister" [169], and major web browsers currently use a PRG in the "xorshift+ family" [240] to implement Javascript's `Math.random()` function.

Why these PRGs are unsatisfactory

Practitioners use these randomness systems for both casual applications (e.g., video games) and serious applications (e.g., scientific simulations). However, for a generic randomized algorithm, there is *no firm mathematical guarantee* that the outputs will be reliable when the algorithm is executed using one of these practical randomness systems. The methods that practitioners typically use to run randomized algorithms must be considered *heuristics.*

To be clear, a lot of work goes into designing high-quality practical randomness systems. Designers strive to ensure that these systems can be safely used in any application that "comes up naturally" in practice. The system is only deemed acceptable for everyday use when it passes a great number of creative statistical tests, such as those in the TestU01 family [147].

These statistical tests are valuable, but there is a wide gap between the statistical tests and a typical randomized algorithm. The designers behind practical systems such as Java's `Math.random()` method wisely do not claim that they work in *adversarial* scenarios, so these systems are considered unsuitable for cryptography. This is true even if we focus solely on the PRG component of these systems. Furthermore, sometimes programs "accidentally" distinguish pseudorandom numbers from truly random numbers. There are quite a few documented cases in which PRGs have been shown to cause inaccurate scientific simulations [68], [69], [88], [89], [107], [139], [174], [187]! One must imagine that other cases have gone unnoticed.

To a theoretician, this state of affairs is deeply unsatisfactory. Yes, modern practical PRGs seem to almost always work well in practice, but we don't have a mathematically rigorous explanation for *why* these systems work. It's not even clear what precisely the goal is. (Mathematically, how can we make a distinction between "adversarially-designed" programs and "naturally-occurring" programs?) By theoreticians' standards, the success of practical PRGs is largely a mystery.

1.1.2 PRGs for all efficient observers

One of the great ideas in the theory of computing is the concept of a PRG that fools *all computationally efficient* observers. Given such a PRG and a truly random seed, we would be able to execute any randomized algorithm that is actually worth executing. (After all, there's no point running a program if one won't even survive long enough to see the output!) Such a PRG could also be used in cryptographic settings, because we can safely assume that eavesdroppers and hackers only have so much computational power.[1]

For example, the Blum-Blum-Shub (BBS) generator [27] uses a short seed to randomly select a suitable modulus M and a number

[1]There is a subtle distinction here. In the context of randomized algorithms, it's okay if the PRG itself uses a little more time than the algorithms that we are trying to fool. On the other hand, in the context of cryptography, we want an efficiently-computable PRG that fools all efficient adversaries, including those that use polynomially more time than the PRG uses.

$X_0 \in \{1, 2, \ldots, M-1\}$, and then it outputs the sequence $(X_1 \bmod 2, X_2 \bmod 2, X_3 \bmod 2, \ldots)$ where

$$X_{i+1} = X_i^2 \bmod M.$$

This PRG is reminiscent of linear congruential generators, but the similarity is only superficial. It is believed that the BBS generator fools polynomial-time algorithms.

Why these PRGs are also (currently) unsatisfactory

Fooling all efficient observers is a well-defined and well-motivated *goal*. Unfortunately, nobody knows how to prove that some efficiently-computable PRG actually *has* this marvelous property.

To be clear, there is a substantial body of "evidence" indicating that such PRGs exist. For example, Blum *et al.* [27] showed that their generator fools all polynomial-time observers, under the plausible-but-unproven assumption that there is no good algorithm for the "quadratic residuosity problem". There are many other examples of PRGs that fool all polynomial-time observers under reasonable cryptographic or complexity-theoretic assumptions [28], [83], [119], [134], [144], [183], [236], [249].[2] For practical cryptography, software developers tend to use PRGs that are not even supported by rigorous *conditional* proofs of correctness, but rather are supported by heuristic and intuitive arguments.

There is a genuine possibility that these PRGs are not secure. In one infamous incident, the U.S. National Institute of Standards and Technology (NIST) recommended using a PRG called "`Dual_EC_DRBG`." The PRG was designed by the U.S. National Security Agency (NSA), and allegedly, they *intentionally designed it to be insecure* for surveillance purposes [189].

Once again, to a theoretician, this state of affairs is not satisfactory. There is genuine room for doubt about *whether* known PRGs work, and perhaps more importantly, even if they do work, we don't have a good

[2] Note that some of these PRGs use somewhat more time than the observers they fool, and hence are suitable for simulating randomized algorithms but not for cryptography (cf. Footnote 1).

explanation for *why* they work. Conditional proofs can be considered partial explanations at best. The problem of designing PRGs that unconditionally fool all efficient observers is very challenging, with connections to deep topics such as the famous **P** vs. **NP** problem (see Section 4.1).

1.1.3 PRGs for restricted models of computation

The main topic of this text is a third approach to PRGs. In this third approach, we identify an interesting and well-defined *restricted model of computation*. Then we design PRGs that fool the chosen model of computation (unconditionally – with no unproven assumptions) and try to optimize the seed length of the PRG.

A toy example might clarify the idea. Let us design a PRG

$$G\colon \{0,1\}^2 \to \{0,1\}^3$$

that fools every observer f that only looks at two of the three output bits. This problem is not completely trivial, because we don't know which two bits f will observe. Nevertheless, the problem can be solved by defining

$$G(u_1, u_2) = (u_1, u_2, u_1 \oplus u_2),$$

where $\oplus$ denotes the XOR operation. When u_1 and u_2 are chosen uniformly at random, the three output bits are correlated, but any two of the bits are independent and uniform random.

Unconditional PRGs can be constructed for much richer and more interesting restricted models of computation. We are especially interested in fooling models of computation that have a "complexity theory" flavor, i.e., we want the output of the PRG to appear random to any observer that is "sufficiently efficient" in some sense. Arguably, the two most important models in this field are *constant-depth circuits* ($\mathbf{AC}^0$, see Definition 2.13) and *read-once branching programs* (ROBPs, see Definition 1.5).

The value of these PRGs

Could PRGs for restricted models ever be directly used in practical applications? Potentially. PRGs for restricted models can be used to

simulate randomized algorithms without significantly distorting their behavior, provided that the algorithms in question are "sufficiently efficient" in the appropriate sense. (See Section 1.5 for more details.)

Admittedly, it's a bit unrealistic to imagine the PRGs studied in the theoretical literature being implemented on actual computers, because it is hard to compete with the practical PRGs discussed in Section 1.1.1. Instead, the study of PRGs for restricted models has a much grander and broader purpose: these PRGs help to uncover the mysteries of the theory of computing, and hence are invaluable from a scientific perspective.

We briefly elaborate on some of the applications of PRGs within the theory of computing in Section 1.5. Apart from any application, we hope to convince the reader that PRGs for restricted models are interesting in their own right.

1.2 Overview of this Text

In this work, we survey some of the most important frameworks and techniques for constructing unconditional PRGs for restricted models of computation. We focus on four major PRG paradigms:

- In Section 2, we present k-wise uniform generators, small-bias generators, and simple combinations thereof.
- In Section 3, we present PRGs that "recycle" randomness to take advantage of communication bottlenecks, such as the Impagliazzo-Nisan-Wigderson generator [131].
- In Section 4, we present connections between PRGs and computational hardness, including the Nisan-Wigderson framework for converting a hard Boolean function into a PRG [183].
- In Section 5, we present methods for constructing PRGs based on (pseudo)random restrictions, including the relatively recent "polarizing random walks" framework [49].

Along the way, as needed, we introduce the computational models that we fool (decision trees, circuits, branching programs, etc.) and tech-

niques for analyzing PRGs (Fourier analysis, sandwiching approximators, simplification-under-restriction lemmas, etc.)

The literature on unconditional PRGs is vast, and this survey is far from exhaustive. (For example, we do not discuss the important line of work on fooling *linear threshold functions* [78], [104], [173], [195].) Instead, we hope that this work serves as a suitable *introduction* to the field of unconditional PRGs, preparing the reader to study new and old papers on PRGs and make their own contributions.

The results that we cover include both classic and recent works. Besides covering the most important *principles* of PRG design and analysis, we also made sure to include expositions of many of the most important *examples* of unconditional PRGs, such as Viola's [242] PRG for low-degree polynomials, Braverman's [36] theorem that limited independence fools $\mathbf{AC}^0$, and Forbes and Kelley's [91] relatively recent PRG for arbitrary-order ROBPs.

This text is primarily expository. However, we couldn't help but include a few novel theorems and proofs. For example, we present a new proof of Braverman's theorem (Section 2.6), and we present a new improvement to the polarizing random walks framework in the low-error regime (Section 5.1.4). We also highlight plenty of important open problems regarding PRGs for restricted models of computation.

Many wonderful prior expository works [15], [97], [165], [175], [185], [238] and lecture notes [45]–[47], [215], [216], [218], [219], [229], [233], [244], [253] include some coverage of unconditional PRGs. However, none of them has quite the same focus as our work, so we feel that our work fills a gap.

In the rest of this section, we discuss some additional basic issues related to the concept of a PRG, paving the way for the PRG constructions in subsequent sections.

1.3 The Generic Probabilistic Existence Proof

For many classes $\mathcal{F}$, including classes defined by standard nonuniform computational models (such as decision trees, circuits, branching programs, etc.), there is a totally generic argument showing that there exist PRGs that fool $\mathcal{F}$ with a small seed length.

Proposition 1.1 (Nonexplicit PRGs)**.** Let $\mathcal{F}$ be a class of functions $f\colon \{0,1\}^n \to \{0,1\}$. For every $\varepsilon > 0$, there exists an ε-PRG for $\mathcal{F}$ with seed length $\log\log|\mathcal{F}| + 2\log(1/\varepsilon) + O(1)$.

Proof. Pick a function $G\colon \{0,1\}^s \to \{0,1\}^n$ uniformly at random. Consider any arbitrary $f \in \mathcal{F}$. For each seed y, the value $f(G(y))$ is a random bit satisfying

$$\mathop{\mathbb{E}}_{G}[f(G(y))] = \mathop{\mathbb{E}}_{U_n}[f(U_n)].$$

Furthermore, as y ranges over all 2^s possible seeds, these random variables $f(G(y))$ are independent. Therefore, by Hoeffding's inequality,

$$\Pr_{G}\left[\left|\mathbb{E}[f] - 2^{-s}\sum_{y\in\{0,1\}^s} f(G(y))\right| > \varepsilon\right] \leq 2e^{-2\varepsilon^2 2^s}.$$

By the union bound, the probability that G fails to ε-fool $\mathcal{F}$ is bounded by $2|\mathcal{F}|e^{-2\varepsilon^2 2^s}$. For $s = \log\log|\mathcal{F}| + 2\log(1/\varepsilon) + O(1)$, this probability is less than 1, i.e., there exists a G that does ε-fool $\mathcal{F}$. □

In a typical case – e.g., if $\mathcal{F}$ is the set of all circuits of size at most n – each function $f \in \mathcal{F}$ can be described using $\mathrm{poly}(n)$ bits, i.e., $|\mathcal{F}| \leq 2^{\mathrm{poly}(n)}$. In this case, the PRG guaranteed by Proposition 1.1 has seed length $O(\log(n/\varepsilon))$.

1.4 Explicitness

Proposition 1.1 has a major weakness: it does not guarantee that the PRG is *efficiently computable*. The proof of Proposition 1.1 is in some sense "nonconstructive." Ideally, we want an *algorithm* for sampling from a pseudorandom distribution, and we want the algorithm to be reasonably efficient with respect to randomness *and* more conventional complexity measures simultaneously.

Definition 1.4 (Explicitness)**.** A PRG $G\colon \{0,1\}^s \to \{0,1\}^n$ is *explicit* if it can be computed in time $\mathrm{poly}(n)$.

One could consider alternative standards of explicitness. We could require that each individual output bit can be computed in time $\mathrm{polylog}\, n$,

or that the PRG runs in space $O(\log n)$, or that each bit can be computed in $\mathbf{AC}^0$, or any number of other conditions. The truth is, there is no "one true definition" of explicitness. The appropriate definition depends on what one hopes to gain from the PRG; see Section 1.5.

In this text, we will stick with Definition 1.4 for concreteness, but when we present PRG constructions, we will generally not bother carefully verifying the runtime bound. Instead, we will focus on making the construction clear to the reader.

1.4.1 Families of PRGs

Definition 1.4 refers to the time complexity of a PRG. To meaningfully speak of time complexity, we technically ought to be considering a whole *family* of PRGs. The convention in this line of work is to keep the family implicit. For example, a theorem might say something like the following.

> For all $n, m \in \mathbb{N}$ and all $\varepsilon > 0$, there exists an explicit ε-PRG for size-m decision trees on n input bits with seed length $O(\log(m/\varepsilon) + \log\log n)$.

(See Section 2.3.3.) Translating into more precise language, the same theorem can be restated as follows.

> There exists a randomized algorithm $\mathcal{G}$ satisfying the following.
>
> 1. Given input parameters n, m, ε, the algorithm $\mathcal{G}$ outputs a string $\mathcal{G}(n, m, \varepsilon) \in \{0,1\}^n$.
> 2. For all n, m, ε, the output distribution $\mathcal{G}(n, m, \varepsilon)$ fools size-m decision trees with error ε.
> 3. $\mathcal{G}(n, m, \varepsilon)$ uses at most $O(\log(m/\varepsilon) + \log\log n)$ random bits and runs in time $\mathrm{poly}(n)$.

There is something potentially troubling about this "translation" process. The quantifiers got flipped! In the informal theorem statement,

we say "for all n, m, ε, there exists an explicit PRG," but strictly speaking, we mean that there exists a single algorithm $\mathcal{G}$ that works for all n, m, ε simultaneously! Is this "flipped quantifiers" convention wise?

Let us make an analogy with big-O notation. Recall, e.g., the famous planar separator theorem:

> For all $n \in \mathbb{N}$, for every n-vertex planar graph, there exists a set of $O(\sqrt{n})$ vertices such that removing those vertices splits the graph into connected components with at most $2n/3$ vertices each.

If we wanted to be more rigorous, we ought to flip the quantifiers and write something like the following:

> There exists a function $f\colon \mathbb{N} \to \mathbb{N}$ such that $f \in O(\sqrt{n})$ and for all $n \in \mathbb{N}$, for every n-vertex planar graph, there exists a set of $f(n)$ vertices such that removing those vertices splits the graph into connected components with at most $2n/3$ vertices each.

We don't bother with such careful language because it obscures more than it clarifies. The important thing is that the expression "$O(\sqrt{n})$" tells how the number of removed vertices scales with the universally quantified parameter n. Analogously, when we say "there exists an explicit PRG," the word "explicit" tells how the *computational complexity* of the PRG scales with the parameters.

1.4.2 The default conjecture: Explicit PRGs exist

For each "reasonable" class $\mathcal{F}$, the standard conjecture is that there exists an *explicit* PRG with essentially the same seed length as the generic nonexplicit bound (Proposition 1.1). Oftentimes, this conjecture can be supported with evidence in the form of conditional constructions. For example, consider the class $\mathcal{F}$ of all CNF formulas of size at most n. The nonexplicit PRG has seed length $O(\log(n/\varepsilon))$. Under plausible complexity-theoretic assumptions, there is indeed an explicit PRG for

all size-n Boolean circuits (whether CNF formulas or not) with seed length $O(\log(n/\varepsilon))$ [134].

Even without a compelling conditional construction, the "default" conjecture would be that for natural families of functions a probabilistic existence proof can be matched by an explicit construction. The main challenge is to find the explicit construction. Typically, such a PRG would be optimal, i.e., one can unconditionally prove a seed length lower bound matching the nonexplicit bound to within a constant factor.[3] For example, every PRG for size-n CNF formulas (explicit or not) must have seed length at least $\Omega(\log(n/\varepsilon))$.

1.5 Applications of PRGs

PRGs for restricted models have many applications. We will not attempt to exhaustively list these applications, nor even to properly survey them. We will, however, briefly describe some of the most important applications. Hopefully, this brief discussion of applications will serve to motivate the main topic of this text, which is the construction and analysis of PRGs.

1.5.1 Simulating randomized algorithms

One of the most natural applications of PRGs is to simulate a randomized algorithm using only a few truly random bits (the seed of the PRG). Let A be a randomized algorithm that we would like to simulate. In order to simulate A without significantly distorting its behavior, what property should our PRG have?

For simplicity, let us assume that A is a decision algorithm, i.e., it outputs a bit. Let $A(a, x)$ denote the output value of A when the input is a and the random bits are x. For each input a, we can define a function $f_a\colon \{0,1\}^n \to \{0,1\}$, where n is the number of random bits that A uses,[4] by the rule $f_a(x) = A(a, x)$. That is, f_a describes the behavior of A on input a as a function of its random bits. Definition 1.2

[3] For a counterexample, see the work of Hoza *et al.* [127].

[4] For simplicity, we assume that n is determined by a rather than varying based on the random bits. This is a "Monte Carlo" algorithm rather than a "Las Vegas" algorithm.

implies that if $G\colon \{0,1\}^s \to \{0,1\}^n$ is an PRG that fools f_a with error ε, then G can be used to simulate A without changing its acceptance probability by more than ε:

$$|\Pr[A(a, U_n) = 1] - \Pr[A(a, G(U_s)) = 1]| \leq \varepsilon.$$

Thus, if we wish to design a PRG to simulate A, we should study the computational complexity of the functions f_a.

Simulating randomized polynomial-time algorithms

One important case is when A is a *polynomial-time* randomized algorithm, corresponding to the complexity class **BPP**. In this case, the following claim says that the functions f_a can be computed by *polynomial-size Boolean circuits*.[5]

Claim 1.2 (PRGs for circuits can be used to simulate **BPP**)**.** Let A be a randomized decision algorithm and let a be an input. Let n be the number of random bits that A uses on input a and define $f_a\colon \{0,1\}^n \to \{0,1\}$ by the rule $f_a(x) = A(a,x)$. Let T be the running time of A on input a and assume $T \geq |a|$. Then f_a can be computed by a Boolean circuit of size $\mathrm{poly}(T)$.[6]

Proof. The function $A(a,x)$ can be computed by a Boolean circuit of size $\mathrm{poly}(T)$ that reads both a and x [190]. When we fix the "a" portion of the input bits to arbitrary values, what remains is a circuit of size $\mathrm{poly}(T)$ operating on x. □

Claim 1.2 implies that if $G\colon \{0,1\}^s \to \{0,1\}^n$ fools circuits of size $\mathrm{poly}(T)$, then G can be used to simulate time-T randomized algorithms. The running time of this simulation is essentially T plus the running time of G, so for this application, the appropriate "explicitness" condition is that G can be computed quickly, e.g., in time $\mathrm{poly}(T)$. Unfortunately, as discussed previously, the challenge of designing explicit PRGs for general Boolean circuits is extremely difficult.

[5] Recall that a Boolean circuit is a network of AND, OR, and NOT gates.

[6] Again, we assume for simplicity that n and T are determined by a rather than varying based on the random bits (cf. Footnote 4).

Remark 1.2 (Nonuniformity)**.** The Boolean circuit model is a *nonuniform* model, i.e., each individual Boolean circuit operates on inputs of some fixed length. The reader might find it counterintuitive that we seek PRGs for circuits in order to simulate *uniform* randomized polynomial-time algorithms (i.e., the one randomized algorithm can handle inputs of any arbitrary length). The concept of *advice* might be helpful [141]. Recall that a family of polynomial-size circuits (one circuit for each input length) is equivalent to a polynomial-time algorithm with a polynomial amount of advice: data that is trustworthy but that depends only on the input length. In our setting, the input a to the polynomial-time algorithm A can be viewed as advice that A uses to try to distinguish between truly random bits and the output of a PRG. We want to simulate A correctly even on a *worst-case* input a, and hence we want a PRG that fools an adversarial polynomial-time observer with advice, i.e., a Boolean circuit.

Simulating randomized log-space algorithms

Another important case is when A is a *log-space* randomized algorithm, corresponding to the complexity class **BPL**. In this case, for each input a, the function f_a can be computed by a *polynomial-width standard-order read-once branching program* (ROBP), defined next.

Definition 1.5 (Standard-order read-once branching programs)**.** A length-n *standard-order read-once branching program* (standard-order ROBP) f consists of a directed layered multigraph with $n+1$ layers, $V_0, \ldots, V_n$. For every $i < n$, each vertex $v \in V_i$ has two outgoing edges leading to V_{i+1}, one labeled 0 and the other labeled 1. Vertices in V_n have zero outgoing edges. There is a designated "start vertex" $v_{\text{start}} \in V_0$. An input $x \in \{0,1\}^n$ selects a path $(v_0, v_1, \ldots, v_n)$ through the graph: the path starts at $v_0 = v_{\text{start}}$, and upon reaching a vertex $v_i \in V_i$, the bit x_{i+1} specifies which outgoing edge to use. There is a designated set of "accept vertices" $V_{\text{accept}} \subseteq V_n$, and $f(x) = 1$ if $v_n \in V_{\text{accept}}$ and $f(x) = 0$ otherwise. The *width* of the program is the maximum number of vertices in a single layer (see Figure 1.2).

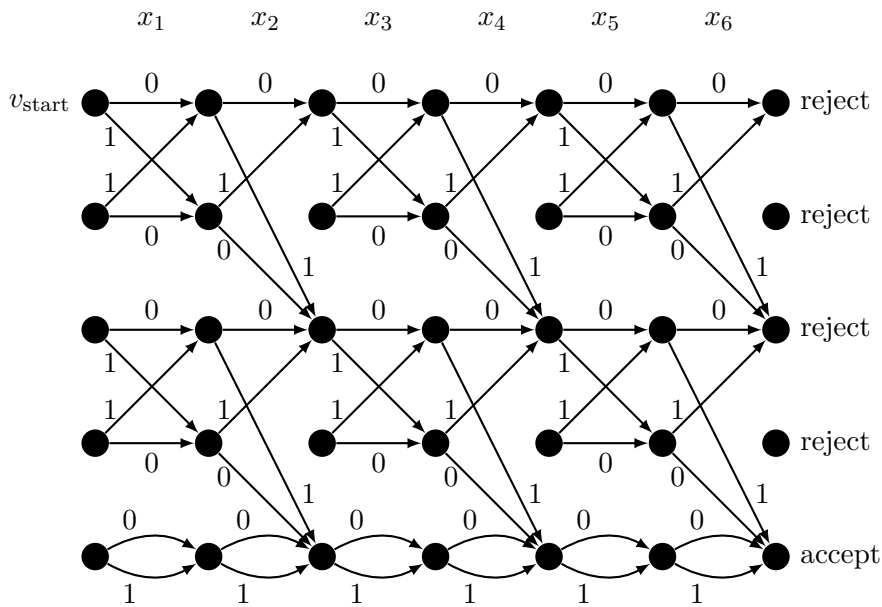

Figure 1.2: A width-5 length-6 standard-order ROBP computing the function $f(x) = \mathsf{MAJ}(x_1 \oplus x_2, x_3 \oplus x_4, x_5 \oplus x_6)$.

Claim 1.3 (PRGs for ROBPs can be used to simulate **BPL**)**.** Let A be a randomized decision algorithm and let a be an input. Let n be the number of random bits that A uses on input a and define $f_a \colon \{0,1\}^n \to \{0,1\}$ by the rule $f_a(x) = A(a,x)$. Let S be the number of bits of space used by A on input a and assume $S \geq \log |a|$. Then f_a can be computed by a standard-order ROBP of width $2^{O(S)}$.

Proof. We think of A as a Turing machine with an input tape, a work tape, and a random tape. Each vertex in the program corresponds to a configuration of A, consisting of the contents of its work tape, the location of the input tape and work tape read heads, and the internal state of A. An edge (u,v) labeled $b \in \{0,1\}$ indicates that if we run A on input a starting at configuration u until its next coin toss, and if that coin toss outcome is b, then the machine's configuration immediately following the coin toss is v. □

Remark 1.3 (The read-once property)**.** In general, a log-space algorithm with a polynomial amount of advice is equivalent to a polynomial-size branching program that might read its bits many times (see Definition 5.16). Nevertheless, we get a *read-once* branching program in Claim 1.3. The reason is that we are focusing on the behavior of the algorithm *as a function of its random bits*. An algorithm in the standard **BPL** model only has read-once access to its random tape: the algorithm cannot go back and re-read old random bits. (If one is computing using a single fair coin, then one cannot ask the coin what the outcome of the first toss was after tossing it a second time.)

Remark 1.4 (ROBP terminology)**.** In the pseudorandomness literature, standard-order ROBPs are often referred to as simply "ROBPs." This practice is a bit misleading, since the definition is *not* simply "a branching program that is read-once." Indeed, in addition to being read-once, we are assuming that the program is *oblivious*, meaning that the variable queried in time step i depends only on i, and more specifically, we are assuming that the branching program follows the *standard variable ordering*, meaning that in time step i, the program queries the variable x_i. (The branching program in the proof of Claim 1.3 indeed reads its input bits in the standard order, because without loss of generality, the algorithm A reads its read-once random tape from left to right.) Unsurprisingly, many papers outside the pseudorandomness literature use terms like "read-once branching program" to refer to more general models that are not necessarily even oblivious [17], [21], [22], [201], [247]. In this text, for clarity, we use the more verbose term "standard-order ROBP" to emphasize the variable ordering assumption.[7]

Claim 1.3 implies that if $G\colon \{0,1\}^s \to \{0,1\}^n$ fools standard-order ROBPs of width $2^{O(S)}$, then G can be used to simulate space-S randomized algorithms. For this application, the appropriate "explicitness" condition is that G can be computed in low space – perhaps space $O(S)$. More precisely, the space complexity of the deterministic simulation is essentially S plus the space complexity of computing $G(y)$ given *one-way read-only access* to the seed y.

[7] Hoza used the same verbose terminology in some other recent expository work [125].

Simulating other types of randomized algorithms

One can consider numerous other classes of randomized algorithms, as well as specific important randomized algorithms. In each case, if we wish to replace the truly random bits with pseudorandom bits, then the question we must answer is, what is the algorithm doing as a function of its random bits? If, for each fixed input a, the algorithm's behavior can be described by a function of "sufficiently low complexity" applied to its random bits, then we can design a PRG that fools such "low-complexity" functions and use it to simulate the algorithm. Because of the presence of the worst-case input a, the appropriate complexity measure will generally be captured by some *nonuniform* model of computation.

1.5.2 Derandomizing algorithms

If we use a PRG to simulate a randomized algorithm in the most natural possible way (as discussed above), we are still using a small amount of randomness, namely the truly random seed of the PRG. However, in many cases it is possible to eliminate this small amount of randomness, leading to a completely deterministic simulation. The most straightforward way to do this is to exhaustively try all possible seeds.

Claim 1.4 (Trying all seeds and taking a majority vote)**.** Let A be a randomized decision algorithm, let a be an input, and let n be the number of random bits that A uses on input a.[8] Let $\varepsilon > 0$ and assume that A succeeds with probability greater than $1/2+\varepsilon$, i.e., there is some "correct answer" $b \in \{0,1\}$ such that

$$\Pr[A(a, U_n) = b] > 1/2 + \varepsilon.$$

Define $f_a\colon \{0,1\}^n \to \{0,1\}$ by the rule $f_a(x) = A(a,x)$. Let $G\colon \{0,1\}^s \to \{0,1\}^n$ be a PRG that ε-fools f_a. Then

$$\mathsf{MAJ}_{y\in\{0,1\}^s}(A(a, G(y))) = b.$$

Proof. First, suppose $b = 1$. The definition of fooling implies that

$$\mathbb{E}[A(a, G(U_s))] = \mathbb{E}[f(G(U_s))] \geq \mathbb{E}[f] - \varepsilon > 1/2 + \varepsilon - \varepsilon = 1/2.$$

[8] Again, we assume for simplicity that n is determined by a.

Therefore, $A(a, G(y)) = 1$ for a majority of seeds y. Now suppose instead that $b = 0$. The fact that G fools f_a with error ε implies that G also fools $1 - f_a$ with error ε, because for any distribution X, we have

$$|\mathbb{E}[1-f_a(X)]-\mathbb{E}[1-f_a]| = |1-\mathbb{E}[f_a(X)]-1+\mathbb{E}[f_a]| = |\mathbb{E}[f_a]-\mathbb{E}[f_a(X)]|.$$

Therefore, by our previous analysis applied to $1 - A(a, x)$, we see that $A(a, G(y)) = 0$ for a majority of seeds y. □

Claim 1.4 implies, for example, that if $G\colon \{0,1\}^s \to \{0,1\}^n$ fools standard-order ROBPs of width $2^{O(S)}$, then we can use it to *deterministically* simulate randomized space-S decision algorithms. The space complexity of this deterministic simulation is essentially S, plus s, plus the space complexity of computing $G(y)$. Thus, for this application, the appropriate "explicitness" condition is that G can be computed in low space – perhaps space $O(s)$. In particular, for this application, there is no significant benefit to constructing a PRG with space complexity $o(s)$, because in the end we are going to use s bits of space to iterate through all possible seeds anyway.

The standard nonconstructive argument (Proposition 1.1) implies that there exists a nonexplicit ε-PRG for width-w length-n standard-order ROBPs with seed length $O(\log(wn/\varepsilon))$. Furthermore, the standard definition of **BPL** implies that randomized log-space algorithms have polynomial running time, and hence they use at most polynomially many random bits. Consequently, if we can design a PRG for standard-order ROBPs with seed length $O(\log(wn/\varepsilon))$ and space complexity $O(\log(wn/\varepsilon))$, then it will follow that $\mathbf{L} = \mathbf{BPL}$. That is, such a PRG would imply that randomized algorithms have at most a constant-factor advantage over deterministic algorithms in terms of space complexity. This would be a profound conclusion about the intrinsic relationship between randomness and memory as computational resources.

So far, optimal constructions of explicit PRGs for ROBPs are not known, but we do have "pretty good" constructions (see, e.g., Section 3.2). Furthermore, there are many partial derandomization results known for space-bounded computation, building on the theory of PRGs for ROBPs (in nontrivial ways). For example, it has been shown that randomized space-S algorithms can be simulated deterministically in

space slightly less than $S^{3/2}$ [124], [206]. The challenge of constructing optimal PRGs for standard-order ROBPs is an exciting and central open problem in the study of unconditional PRGs.

Other applications

We have briefly discussed the most straightforward applications of PRGs, namely simulating randomized algorithms using little or no randomness. We now give a small sample of less straightforward applications.

- Ironically, it turns out that PRGs are sometimes useful for *designing randomized algorithms*. For example, PRGs for space-bounded computation are often used in the design of randomized streaming algorithms using a technique first introduced by Indyk [136].
- Unconditional PRGs for restricted models have applications to "hardness amplification within **NP**" [105], [121], [160].
- Unconditional PRGs for restricted models have applications in the area of "meta-complexity." It turns out that PRGs can be used to rule out certain types of "natural proofs" of strong circuit lower bounds [199] or to show that certain models of computation cannot solve the "Minimum Circuit Size Problem" [137]. For these applications, the "correct" definition of explicitness is that for each fixed seed $y \in \{0,1\}^s$, there is a small Boolean circuit C_y such that for every $i \in [n]$, we have $C_y(i) = G(y)_i$.

1.6 Beyond PRGs: Hitting Set Generators and More

For the sake of context, in this section we briefly describe some relaxations of the PRG definition. The main motivation behind studying these relaxations is that constructing PRGs is challenging. These "generalized PRGs" are sometimes easier to construct, and yet they suffice for some (but not all) of the applications of PRGs. We only give a short overview of these concepts, since our main focus is true PRGs.

The most well-known "generalized PRG" concept is a *hitting set generator* (HSG).

Definition 1.6 (HSGs)**.** Suppose $\mathcal{F}$ is a class of functions $f\colon \{0,1\}^n \to \{0,1\}$. An *$\varepsilon$-HSG* for $\mathcal{F}$ is a function $G\colon \{0,1\}^s \to \{0,1\}^n$ such that for every $f \in \mathcal{F}$, if $\mathbb{E}[f] \geq \varepsilon$, then there exists some x such that $f(G(x)) = 1$.

An HSG is a "one-sided PRG." HSGs have been studied since the 1980s [3] if not earlier. HSGs can be used to derandomize algorithms that have one-sided error, simply by trying all seeds. In some contexts, HSGs can also be used (in nontrivial ways) to derandomize algorithms that have two-sided error [11], [12], [40], [61], [100].

A few years ago, [37] introduced a different generalization of PRGs, called *weighted PRGs* (WPRGs).[9]

Definition 1.7 (WPRG)**.** Suppose $\mathcal{F}$ is a class of functions $f\colon \{0,1\}^n \to \mathbb{R}$. An *$\varepsilon$-WPRG* for $\mathcal{F}$ is a pair (G, ρ), where $G\colon \{0,1\}^s \to \{0,1\}^n$ and $\rho\colon \{0,1\}^s \to \mathbb{R}$, such that for every $f \in \mathcal{F}$, we have

$$\left| \mathop{\mathbb{E}}_{U \sim U_s} [f(G(U)) \cdot \rho(U)] - \mathbb{E}[f] \right| \leq \varepsilon.$$

Thus, WPRGs generalize PRGs because we consider sparse *linear* combinations of the outputs of f rather than sparse *convex* combinations of the outputs of f. Several recent works have exploited this extra flexibility to construct WPRGs with better parameters than known PRGs [37], [52], [70], [124], [193].

Yet another generalization of PRGs is the concept of a *deterministic sampler*.

Definition 1.8 (Deterministic sampler)**.** Suppose $\mathcal{F}$ is a class of functions $f\colon \{0,1\}^n \to \mathbb{R}$. An *$\varepsilon$-deterministic sampler* for $\mathcal{F}$ is a deterministic oracle algorithm A that makes queries to a function $f \in \mathcal{F}$ and outputs a number $A^f \in \mathbb{R}$ such that $|A^f - \mathbb{E}[f]| \leq \varepsilon$.

The deterministic sampler model isolates a key feature of PRGs, which is that they are useful even if we merely have black-box access to the function f. Deterministic samplers have been discussed (by name)

[9] In Braverman, Cohen, and Garg's [37] original paper, they speak of "pseudorandom pseudo-distributions." The "weighted PRG" terminology was introduced later, by Cohen *et al.* [70].

in a few recent works [61], [191], [194]. Several older algorithms can also be understood as deterministic samplers [11], [12], [40], [100], [132].

One can show that these four concepts form a hierarchy:

$$\text{PRG} \implies \text{WPRG} \implies \text{deterministic sampler} \implies \text{HSG}.$$

Thus, PRGs (our focus in this text) are the most desirable of the four.

2

Limited Independence and Small-Bias Generators

In this section, we study "k-wise uniform" generators, "small-bias" generators, and simple combinations thereof. What these PRG constructions have in common is that they are closely related to *error correcting codes*. Prior knowledge of coding theory is not necessary to understand the PRGs. The constructions of these PRGs are fairly elementary, but we emphasize that the analyses are interesting and not always trivial. We will build up to showing that these simple PRGs can fool moderately powerful classes of functions, such as bounded-depth circuits and low-degree polynomials over $\mathbb{F}_2$.

2.1 Limited Independence

2.1.1 Pairwise uniform bits

For our first PRG, let us fool the first nontrivial case of *juntas*.

Definition 2.1 (Juntas)**.** A function f on $\{0,1\}^n$ is a *k-junta* if f only depends on at most k variables, i.e.,

$$f(x) = g(x_{i_1}, \ldots, x_{i_k})$$

for some indices $i_1, \ldots, i_k \in [n]$ and some function g.

Fooling 1-juntas is trivial. Indeed, let $G\colon \{0,1\} \to \{0,1\}^n$ be the PRG with seed length 1 given by $G(b) = (b, b, b, \ldots, b)$. Then G perfectly fools every 1-junta, because for any i, the i-th bit in the output of the PRG is a uniform bit.

Let us consider the case of 2-juntas. Fooling one specific 2-junta, such as the function $f(x) = x_7 \wedge x_{13}$, is still trivial: using a 2-bit seed, we can sample $X_7, X_{13} \in \{0,1\}$ uniformly and independently at random and set $X_i = 0$ for all $i \notin \{7, 13\}$. The challenge is to construct a single PRG that fools all 2-juntas simultaneously. In other words, the challenge is that when we design the PRG, we don't know in advance which two bits are relevant.

Theorem 2.1 (Pairwise uniform bits)**.** For every $n \in \mathbb{N}$, there is an explicit PRG that perfectly fools 2-juntas on n bits with seed length $\lfloor \log n \rfloor + 1$.

A distribution X that perfectly fools 2-juntas is also called a *pairwise uniform* distribution, because every two bits of X are uniform over $\{0,1\}^2$. In practice, people often use the alternative phrase "pairwise independent." This practice is a little sloppy, because it doesn't clarify the marginal distributions of the individual coordinates of X.

A generator for $n = 3$ was described in Section 1.1.3. The solution for larger n is a natural generalization.

Proof of Theorem 2.1. Let $s = \lfloor \log n \rfloor + 1$, and let $I_1, \ldots, I_n$ be arbitrary nonempty subsets of $[s]$, where we use that $2^{\lfloor \log n \rfloor + 1} - 1 \geq n$. The PRG $G\colon \{0,1\}^s \to \{0,1\}^n$ is given by

$$G(y) = \left(\bigoplus_{i \in I_1} y_i, \ldots, \bigoplus_{i \in I_n} y_i \right). \tag{2.1}$$

To prove that this works, consider sampling $Y \in \{0,1\}^s$ uniformly at random. First, observe that $\bigoplus_{i \in I} Y_i$ is uniform random for any nonempty set $I \subseteq [n]$, so each individual output bit of $G(Y)$ is distributed uniformly over $\{0,1\}$. Now let $j, k \in [n]$ be two distinct indices, and define

$$A = \bigoplus_{i \in I_j} Y_i, \qquad B = \bigoplus_{i \in I_k} Y_i,$$

so $(G(Y)_j, G(Y)_k) = (A, B)$. Observe that

$$A \oplus B = \bigoplus_{i \in I_j \Delta I_k} Y_i,$$

and $I_j \Delta I_k$ is nonempty. Therefore, A is uniform random, B is uniform random, and $A \oplus B$ is uniform random. One can show that it follows that (A, B) is distributed uniformly over $\{0,1\}^2$. Therefore, for any function f that only depends on x_j and x_k, the random variables $f(G(U_s))$ and $f(U_n)$ are identically distributed. □

The seed length in Theorem 2.1 is precisely optimal [8], [66], i.e., every pairwise uniform generator has a seed length of at least $\lfloor \log n \rfloor + 1$.

Pairwise uniform bits, and the more general concept of pairwise independence, have many applications in complexity theory and beyond; see Luby and Wigderson's work [165] for a survey.

2.1.2 *k*-wise uniform bits

For our next PRG, let us fool the class of k-juntas for any k, i.e., we will construct a *k-wise uniform* distribution.

Theorem 2.2 (k-wise uniform bits)**.** For every $n, k \in \mathbb{N}$, there is an explicit PRG that perfectly fools k-juntas on n bits with seed length $O(k \log n)$.

Proof. Let $\mathbb{F}_q$ be a finite field with at least n elements. Let $\mathcal{P}$ be the set of univariate polynomials over $\mathbb{F}_q$ of degrees less than k. Let $z_1, \ldots, z_k \in \mathbb{F}_q$ be distinct. In preparation for defining the PRG, define $H \colon \mathcal{P} \to \mathbb{F}_q^k$ by

$$H(p) = (p(z_1), \ldots, p(z_k)).$$

The function H is injective, because if $H(p) = H(p')$, then $p - p'$ is a polynomial with at least k zeroes of degree less than k, hence $p = p'$. Furthermore, $|\mathcal{P}| = |\mathbb{F}_q^k| = q^k$, since a polynomial $p \in \mathcal{P}$ can be specified by k coefficients from $\mathbb{F}_q$. Therefore, H is bijective, and hence if $P \in \mathcal{P}$ is sampled uniformly at random, $H(P)$ is a uniform random vector.

Now let $z_1, \ldots, z_n \in \mathbb{F}_q$ be distinct, and define $G \colon \mathcal{P} \to \mathbb{F}_q^n$ by

$$G(p) = (p(z_1), \ldots, p(z_n)).$$

By the above analysis, when $P \in \mathcal{P}$ is sampled uniformly at random, any k coordinates of $G(P)$ are independent and uniform random.

All that remains is to bridge the gap between field elements and bits. Let q be a power of two, so that field elements can be naturally encoded as bitstrings. The seed of our PRG describes a polynomial $p \in \mathcal{P}$ by giving the encodings of its k coefficients; this requires $k \log q = k \cdot \lceil \log n \rceil$ bits if we pick q to be the smallest power of two that is at least n. The output of our PRG is the sequence of first bits of the encodings of the coordinates of $G(p)$. □

The seed length in Theorem 2.2 is optimal up to a constant factor for moderate values of k. More precisely, the optimal seed length is $\Theta(k \cdot \log(n/k))$ [8], [62], [66], which is a slight improvement over Theorem 2.2 when $k \geq n^{1-o(1)}$. Even when k is small, the constant factor in the seed length of Theorem 2.2 can be improved by roughly a factor of two [8].

2.1.3 Perfectly fooling shallow decision trees

Next, let us fool shallow *decision trees*, which generalize juntas.

Definition 2.2 (Decision trees)**.** A *decision tree* over $\{0,1\}^n$ is a tree, where each internal node is labeled with a variable x_i and has two children, the two edges leading from an internal node to its children are labeled 0 and 1, and each leaf is labeled with an output value (0 or 1). A decision tree computes a function $f\colon \{0,1\}^n \to \{0,1\}$ by walking from root to leaf according to the values of the variables queried (see Figure 2.1).

Every k-junta can be computed by a depth-k decision tree. To fool decision trees, rather than constructing a new PRG from scratch, we'll show that every PRG for k-juntas automatically fools depth-k decision trees – even though such a tree might compute a function that depends on far more than k variables. This is a common pattern in PRG design: first one designs a PRG for a relatively simple class of functions, and then one proves that such a PRG automatically fools a more sophisticated class of functions.

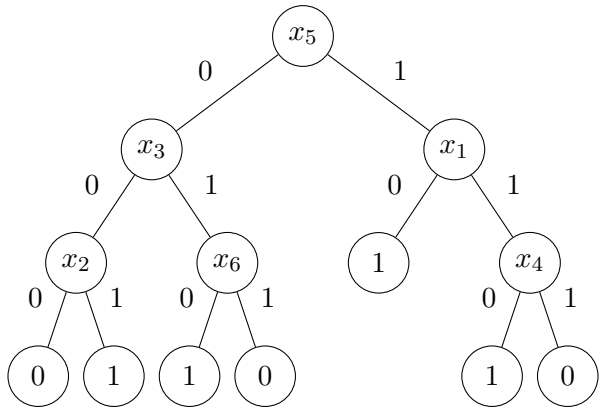

Figure 2.1: A depth-3 decision tree. Note that the function it computes depends on all 6 variables.

Proposition 2.1 (Perfect PRGs for shallow decision trees)**.** Let $n, k \in \mathbb{N}$ and let X be a k-wise uniform distribution over $\{0,1\}^n$. Then X perfectly fools depth-k decision trees. Consequently, there is an explicit PRG with seed length $O(k \log n)$ that perfectly fools depth-k decision trees on n variables.

Proof. Let f be a depth-k decision tree. Let A be the set of accepting leaves of f, i.e., leaves that are labeled 1. For each leaf $u \in A$, define $f_u \colon \{0,1\}^n \to \{0,1\}$ by letting $f_u(x) = 1$ if and only if f arrives at u when it reads x. Note that f_u is a k-junta, because its value only depends on the variables queries on the path from the root to u. Furthermore, we can express f as

$$f(x) = \sum_{u \in A} f_u(x).$$

Therefore, by linearity of expectation,

$$\begin{aligned}\mathbb{E}[f(X)] = \mathbb{E}\left[\sum_{u \in A} f_u(X)\right] = \sum_{u \in A} \mathbb{E}[f_u(X)] &= \sum_{u \in A} \mathbb{E}[f_u] \\ &= \mathbb{E}\left[\sum_{u \in A} f_u\right] \\ &= \mathbb{E}[f].\end{aligned}$$

□

The simple technique in the proof above is quite valuable. Let us abstract it out and generalize it to the case of imperfect PRGs.

Lemma 2.3 (Triangle Inequality for PRG Errors)**.** Let $f_1, \ldots, f_k \colon \{0,1\}^n \to \mathbb{R}$ be functions, let $\lambda_0, \ldots, \lambda_k \in \mathbb{R}$, and let $f(x) = \lambda_0 + \sum_{i=1}^{k} \lambda_i \cdot f_i(x)$. Let X be a distribution over $\{0,1\}^n$, and assume that X fools f_i with error ε_i for each i. Then X fools f with error ε, where

$$\varepsilon = \sum_{i=1}^{k} |\lambda_i| \cdot \varepsilon_i.$$

Proof.

$$\begin{aligned} |\mathbb{E}[f(X)] - \mathbb{E}[f]| &= \left| \sum_{i=1}^{k} \lambda_i \cdot \mathbb{E}[f_i(X)] - \sum_{i=1}^{k} \lambda_i \cdot \mathbb{E}[f_i] \right| \\ &\qquad \text{(Linearity of expectation)} \\ &\leq \sum_{i=1}^{k} |\lambda_i| \cdot |\,\mathbb{E}[f_i(X)] - \mathbb{E}[f_i]| \\ &\qquad \text{(Standard triangle inequality)} \\ &\leq \sum_{i=1}^{k} |\lambda_i| \cdot \varepsilon_i \end{aligned}$$

because X fools f_i with error ε_i. □

2.1.4 Connection with coding theory: Dual codes

For readers with a background in coding theory, the constructions of pairwise and k-wise uniform generators might have felt familiar. Indeed, the constructions are closely related to the Hadamard code and the Reed-Solomon code, respectively. For the sake of those readers who have some familiarity with coding theory, we will now describe a general elegant characterization of exactly which linear codes induce k-wise uniform distributions. Recall that a linear code over $\mathbb{F}_2^n$ is a subspace $C \subseteq \mathbb{F}_2^n$, and its *dual code* is defined as

$$C^{\perp} = \{x \in \mathbb{F}_2^n \colon \forall y \in C, \langle x, y \rangle = 0\},$$

where $\langle \cdot, \cdot \rangle$ is the standard dot product over $\mathbb{F}_2^n$. The minimum distance of a code is the smallest Hamming distance between two distinct codewords. For a linear code C, this distance coincides with the smallest Hamming weight among all nonzero codewords of C.

Proposition 2.2 (Connection between k-wise uniformity and coding theory)**.** Let $C \subseteq \mathbb{F}_2^n$ be a linear subspace, and sample X uniformly at random from C. Then X is k-wise uniform if and only if $C^\perp$ has minimum distance at least $k+1$.

Proof. First, suppose X is k-wise uniform. Let $x \in \mathbb{F}_2^n$ be a nonzero vector with Hamming weight at most k. Then $\langle x, X \rangle$ is a uniform random bit, so there is certainly some $y \in C$ such that $\langle x, y \rangle = 1$. Therefore, $x \notin C^\perp$. Since $C^\perp$ is a subspace, it follows that $C^\perp$ has minimum distance at least $k+1$.

Conversely, suppose $C^\perp$ has minimum distance at least $k+1$, and consider any k distinct indices $i_1, \ldots, i_k \in [n]$. Let $s = \dim(C)$, and let $M \in \mathbb{F}_2^{n \times s}$ be a matrix with image C and rows $M_1, \ldots, M_n$. Let $x \in \mathbb{F}_2^n$ be an arbitrary nonzero vector supported on the indices $i_1, \ldots, i_k$. Then x has Hamming weight at most k, so $x \notin C^\perp$, i.e., there is some $z \in \mathbb{F}_2^s$ such that

$$0 \neq \langle x, Mz \rangle = \sum_{i=1}^{n} x_i \cdot \langle M_i, z \rangle = \left\langle \sum_{i=1}^{n} x_i M_i, z \right\rangle.$$

Therefore, $\sum_{i=1}^n x_i M_i \neq 0$. Since x was arbitrary, this shows that $M_{i_1}, \ldots, M_{i_k}$ are linearly independent. Define

$$M' = \begin{bmatrix} M_{i_1} \\ M_{i_2} \\ \vdots \\ M_{i_k} \end{bmatrix}.$$

Since row rank is equal to column rank, there are k linearly independent columns of M' with indices in some set $J = \{j_1, \ldots, j_k\}$. Therefore, when $z \in \mathbb{F}_2^s$ is chosen uniformly at random, $M'z$ is a uniform random element of $\mathbb{F}_2^k$. To see this, note that for any $x \in \mathbb{F}_2^k$ and any fixing of all z_j with $j \notin J$, there is a unique choice of $z_J \in \mathbb{F}_2^k$ for which $M'z = x$. It follows that X is k-wise uniform. □

Proposition 2.2 provides a generic recipe for constructing k-wise uniform distributions from error correcting codes. Recall that the *redundancy* of a code is the difference between the block length and the message length. A binary linear code with block length n, minimum distance $k+1$, and redundancy s induces a k-wise uniform generator with seed length s and output length n.

2.2 Small-bias Distributions

2.2.1 Fooling parities of variables

For our first imperfect PRG, let us fool *parity functions*.

Definition 2.3 (Parity functions)**.** A *parity function* is a function $f\colon \{0,1\}^n \to \{0,1\}$ of the form $f(x) = \bigoplus_{i\in S} x_i$ for some set $S \subseteq [n]$.

Equivalently, we can think of f as a map $\mathbb{F}_2^n \to \mathbb{F}_2$. Then f is a parity function if and only if $f(x) = \langle a, x\rangle$ for some fixed vector a, where $\langle\cdot,\cdot\rangle$ is the usual inner product, i.e., $\langle a, x\rangle = \sum_{i=1}^n a_i \cdot x_i$. Sometimes it is more convenient to work with $\{\pm 1\}$-valued functions, in which case parity functions become *character functions*.

Definition 2.4 (Character functions)**.** Let $n \in \mathbb{N}$, and let $S \subseteq [n]$. The *character function* of S, denoted $\chi_S\colon \{0,1\}^n \to \{\pm 1\}$, is defined by

$$\chi_S(x) = \prod_{i\in S} (-1)^{x_i}.$$

Note that $f\colon \{0,1\}^n \to \{0,1\}$ is a parity function if and only if $(-1)^f$ is a character function. Since $(-1)^f = 1 - 2f$, it follows from the triangle inequality that fooling character functions with error ε is equivalent to fooling parity functions with error $\varepsilon/2$.

Definition 2.5 (Bias)**.** An *ε-biased distribution* over $\{0,1\}^n$ is a distribution that ε-fools character functions. Equivalently, a distribution is ε-biased if it $(\varepsilon/2)$-fools parity functions. An *ε-biased generator* is an ε-PRG for character functions.

Theorem 2.4 (Small-bias generators [178], [188])**.** For every $n \in \mathbb{N}, \varepsilon > 0$, there is an explicit ε-biased generator with output length n and seed length $O(\log(n/\varepsilon))$.

Remark 2.1. Ideally, we want to design PRGs for interesting and powerful models of computation. The reader might feel that "parity functions" is hardly a "model of computation" at all, and the utility of ε-biased generators is unclear. However, we will see later that any distribution that fools parity functions with sufficiently low error also fools many more interesting models. Furthermore, ε-biased generators are building blocks in many more powerful PRGs.

There are many possible ways to prove Theorem 2.4. The first proofs were discovered by by Naor and Naor [178] and by Peralta [188] independently. We'll present a simpler construction due to Alon *et al.* [9].

Proof of Theorem 2.4. Let $q = n/\varepsilon$, and assume without loss of generality that q is a power of two. As vector spaces over $\mathbb{F}_2$, identify $\mathbb{F}_q$ with $\mathbb{F}_2^{\log q}$. Our PRG $G\colon \mathbb{F}_q \times \mathbb{F}_q \to \{0,1\}^n$ is defined by

$$(G(y,z))_i = \langle y, z^i \rangle.$$

To prove that this works, let $f\colon \{0,1\}^n \to \{0,1\}$ be a nonzero parity function, say $f(x) = \bigoplus_{i \in S} x_i$. Then doing arithmetic in $\mathbb{F}_2$,

$$f(G(y,z)) = \sum_{i \in S} \langle y, z^i \rangle = \left\langle y, \sum_{i \in S} z^i \right\rangle.$$

Define $g(z) = \sum_{i \in S} z^i$. Then g is a nonzero polynomial in $\mathbb{F}_q[z]$ of degree at most n, and $f(G(y,z)) = \langle y, g(z) \rangle$. When z is a root of g, obviously $f(G(y,z)) = 0$. On the other hand, when z is not a root of g, if we sample $Y \in \mathbb{F}_q$ uniformly at random, $f(G(Y,z))$ is a uniform random bit. Therefore, when we sample $Y, Z \in \mathbb{F}_q$ independently and uniformly at random,

$$\mathop{\mathbb{E}}_{Y,Z}[f(G(Y,Z))] = \frac{1}{2} \cdot \Pr_Z[g(Z) \neq 0] \in \left[\frac{1}{2} - \frac{n}{2q}, \frac{1}{2}\right].$$

Since $\mathbb{E}[f] = \frac{1}{2}$, our PRG G fools parity functions with error $n/(2q) = \varepsilon/2$, and hence it fools character functions with error ε. □

2.2.2 A better seed length for parities of few variables

The seed length $O(\log(n/\varepsilon))$ in Theorem 2.4 is asymptotically optimal [9]. However, we can achieve a better seed length for parities of *just a few variables*, i.e., functions that are simultaneously parity functions and juntas.

Definition 2.6 (k-wise ε-bias)**.** Let X be a distribution over $\{0,1\}^n$. We say that X is *k-wise ε-biased* if it ε-fools every character function χ_S for which $|S| \leq k$. Similarly, a k-wise ε-biased generator is an ε-PRG for character functions χ_S that satisfy $|S| \leq k$.

Theorem 2.5 (k-wise ε-biased generators [178])**.** For every $n, k \in \mathbb{N}$ and every $\varepsilon > 0$, there is an explicit k-wise ε-biased generator with output length n and seed length $O(\log(k/\varepsilon) + \log\log n)$.

Proof. Let $G\colon \{0,1\}^s \to \{0,1\}^n$ be a k-wise uniform generator that is also a linear transformation when we think of it as a map between vector spaces, $G\colon \mathbb{F}_2^s \to \mathbb{F}_2^n$. (One can verify that the k-wise uniform generator that we constructed to prove Theorem 2.2 is indeed a linear transformation.) Let $Y = G'(U_{s'})$ where $G' : \{0,1\}^{s'} \to \{0,1\}^s$ is an ε-biased generator. We will show that $G(Y)$ fools parities of at most k bits. Indeed, let $f(x) = \sum_{i \in S} x_i$, where $x \in \mathbb{F}_2^n$ and $|S| \leq k$. Let $M \in \mathbb{F}_2^{n \times s}$ be the matrix representation of G, with rows $M_1, \ldots, M_n \in \mathbb{F}_2^s$. Then for any $y \in \mathbb{F}_2^s$,

$$f(G(y)) = \sum_{i \in S} \langle M_i, y \rangle = \sum_{i \in S} \sum_{j=1}^{s} M_{ij} y_j = \sum_{j=1}^{s} \left(\sum_{i \in S} M_{ij} \right) y_j.$$

This is a parity function of the variables $y_1, \ldots, y_s$. Therefore, since Y is ε-biased, $|\mathbb{E}[f(G(Y))] - \mathbb{E}[f(G(U))]| \leq \varepsilon/2$. Furthermore, since G is k-wise uniform and f is a k-junta, $\mathbb{E}[f(G(U))] = \mathbb{E}[f]$. Therefore, $G(Y)$ is k-wise ε-biased. To achieve the promised seed length, we can plug in the constructions of Theorems 2.2 and 2.4 for G and G' respectively. □

Once again, the seed length of Theorem 2.5 is optimal up to constant factors.

2.2.3 Connection with coding theory: Nearly balanced codes

In Section 2.1.4, we saw a connection between k-wise uniform distributions and error correcting codes that "explains" our constructions of k-wise uniform generators (Theorems 2.1 and 2.2). Now we discuss a similar connection between ε-biased distributions and error correcting codes.

Suppose $C \subseteq \mathbb{F}_2^m$ is a linear code. It is generally desirable for C to have a large minimum weight. Small-biased distributions are equivalent to codes C that also have a small *maximum* weight. Specifically, we say that C is ε-*balanced* if every nonzero $x \in C$ has relative Hamming weight $\frac{1}{2} \pm \varepsilon$.

Proposition 2.3 (Nearly balanced code $\iff$ small-bias distribution)**.** Let $M \in \mathbb{F}_2^{m \times n}$ be a linear transformation, and let C be the image of M, i.e., $C = \{Ma : a \in \{0,1\}^n\}$. Sample X uniformly at random from the rows of M, so $X \in \{0,1\}^n$. Then C is ε-balanced if and only if X is (2ε)-biased.

Proof. For any nonzero "message" $a \in \{0,1\}^n$, the relative Hamming weight of Ma is the fraction of rows M_i of M such that $\langle a, M_i \rangle = 1$, i.e., $\Pr[\langle a, X \rangle = 1]$. □

A nearly balanced code that stretches an n-bit message to an m-bit codeword corresponds to a small-bias generator that stretches a $(\log m)$-bit seed to an n-bit pseudorandom string. In both problems, it is desirable to minimize m. A natural way to construct a nearly balanced code is to concatenate the Hadamard code with the Reed-Solomon code. Through Proposition 2.3, that gives an explicit ε-biased generator similar to the PRG we constructed to prove Theorem 2.4. The two constructions are not quite identical. Both have seed length $O(\log(n/\varepsilon))$, so the coding-theory perspective gives an alternative proof of Theorem 2.4.

Because of the connection between small-bias distributions and nearly balanced codes, even constant-factor improvements in the seed length of small-bias generators are interesting. Note that a constant factor in the seed length translates to a constant factor in the *exponent* of

the codeword length! The seed length in Theorem 2.4 is $2\log(n/\varepsilon)+O(1)$. For moderate ε, the best small-bias generator is a construction by Ta-Shma [217] with seed length[1]

$$\log n + 2\log(1/\varepsilon) + \widetilde{O}(\log^{2/3}(1/\varepsilon)).$$

This seed length is extremely close to the nonconstructive bound of $\log n + 2\log(1/\varepsilon) + O(1)$ (Proposition 1.1), as well as to the lower bound of $\log n + 2\log(1/\varepsilon) - \log\log(1/\varepsilon) - O(1)$ [9]. Ta-Shma's seed length translates to an ε-balanced code that stretches messages of length n to codewords of length $n/\varepsilon^{2+o(1)}$.

Open Problem 2.1 (Optimal small-bias generators up to an *additive* constant in the seed length)**.** Construct an explicit ε-biased generator with seed length $\log n + 2\log(1/\varepsilon) + O(1)$, and hence an explicit ε-balanced code that stretches messages of length n to codewords of length $O(n/\varepsilon^2)$.

2.3 Analysis Technique: Fourier L_1 Bounds

2.3.1 Basic Fourier analysis

PRGs for character functions (i.e., small-bias distributions) are especially important because character functions are the basic "building blocks" out of which all other functions $f\colon \{0,1\}^n \to \mathbb{R}$ can be assembled.

Proposition 2.4 (The Fourier expansion)**.** Every function $f\colon \{0,1\}^n \to \mathbb{R}$ can be uniquely written as a linear combination of characters, i.e.,

$$f(x) = \sum_{S\subseteq[n]} \widehat{f}(S)\cdot\chi_S(x), \tag{2.2}$$

where $\widehat{f}(S) \in \mathbb{R}$.

Proof. The space of all functions $f\colon \{0,1\}^n \to \mathbb{R}$ is a vector space, isomorphic to $\mathbb{R}^{2^n}$. Define an inner product on this space by

$$\langle f, g\rangle = \mathop{\mathbb{E}}_{U\sim U_n}[f(U)\cdot g(U)].$$

[1] Here, we are ignoring rounding issues. That is, the domain size S of Ta-Shma's generator is not necessarily a power of two, and when we say "seed length" we simply mean $\log_2 S$.

With respect to this inner product, the 2^n character functions are orthonormal. Therefore, they form a basis. □

The decomposition of Equation (2.2) is called the *Fourier expansion* of f, and the numbers $\widehat{f}(S)$ are called the *Fourier coefficients* of f. The Fourier expansion of f can reveal important information about f. For example, by linearity of expectation,

$$\mathbb{E}[f] = \sum_{S \subseteq [n]} \widehat{f}(S) \cdot \mathbb{E}[\chi_S] = \widehat{f}(\varnothing). \tag{2.3}$$

We refer the reader to [76], [185] for an introduction to the Fourier analysis of functions on the Boolean cube.

2.3.2 Almost k-wise uniform bits

Let us use Fourier analysis to obtain another PRG for k-juntas. For moderate error, its seed length is superior to that of the k-wise uniform generator that we saw before (Theorem 2.2). The following theorem is a form of "Vazirani's XOR Lemma."

Theorem 2.6 (Almost k-wise uniform generator [178])**.** If X is a k-wise δ-biased distribution over $\{0,1\}^n$, then X fools $[-1,1]$-valued k-juntas with error $\delta \cdot 2^{k/2}$. Consequently, for every $k, n \in \mathbb{N}$ and $\varepsilon > 0$, there is an explicit ε-PRG for $[-1,1]$-valued k-juntas with seed length $O(k + \log(1/\varepsilon) + \log\log n)$.

A distribution X that fools all $\{0,1\}$-valued k-juntas with error ε is also called an *ε-almost k-wise uniform* distribution.[2] An equivalent condition is that every k coordinates of X are ε-close to U_k in total variation distance. In practice, people often use the alternative phrase "ε-almost k-wise independent."

The proof of Theorem 2.6 is based on bounding the magnitude of Fourier coefficients.

Definition 2.7 (Fourier L_1 norm)**.** Let $f\colon \{0,1\}^n \to \mathbb{R}$. The *Fourier L_1 norm of f*, also known as the *Fourier algebra norm* or *spectral norm* of

[2] Warning: Occasionally, the same "ε-almost" terminology refers to some other measure of the extent to which X fails to be perfectly k-wise uniform [1].

f, denoted $L_1(f)$, is the sum of absolute values of Fourier coefficients of f:

$$L_1(f) = \sum_{S \subseteq [n]} |\widehat{f}(S)|.$$

Lemma 2.7 (Universal Fourier L_1 bound)**.** For any function $f\colon \{0,1\}^n \to [-1,1]$, we have $L_1(f) \le 2^{n/2}$.

Proof. By the Cauchy-Schwarz inequality, $L_1(f) \le \sqrt{2^n \cdot \sum_{S \subseteq [n]} \widehat{f}(S)^2}$. Furthermore, for any function $f\colon \{0,1\}^n \to \mathbb{R}$,

$$\mathbb{E}[f(U_n)^2] = \langle f, f \rangle = \sum_{S,T \subseteq [n]} \widehat{f}(S) \cdot \widehat{f}(T) \cdot \langle \chi_S, \chi_T \rangle = \sum_{S \subseteq [n]} \widehat{f}(S)^2. \quad (2.4)$$

(Equation (2.4) is called *Parseval's theorem.*) In our case, f is $[-1,1]$-valued, so $\mathbb{E}[f(U_n)^2] \le 1$ and hence $\sum_{S \subseteq [n]} \widehat{f}(S)^2 \le 1$. □

Lemma 2.8 (Fourier L_1 bound $\implies$ fooled by small-bias)**.** Let $f\colon \{0,1\}^n \to \mathbb{R}$. If X is ε-biased, then X fools f with error $\varepsilon \cdot L_1(f)$.

Proof. This is a special case of the Triangle Inequality for PRG Errors (Lemma 2.3). □

Proof of Theorem 2.6. Let $f\colon \{0,1\}^n \to [-1,1]$ be a k-junta, i.e., $f(x) = g(x_{i_1}, \ldots, x_{i_k})$ for some function $g\colon \{0,1\}^k \to [-1,1]$, where $i_1, \ldots, i_k$ are distinct. By Lemmas 2.7 and 2.8, the distribution $(X_{i_1}, \ldots, X_{i_k})$ fools g with error $\delta \cdot 2^{k/2}$, and hence X fools f with the same error. The final seed length follows from Theorem 2.5 by choosing $\delta = \varepsilon \cdot 2^{-k/2}$. □

2.3.3 Fooling bounded-size decision trees

Recall that in Section 2.1.3, we showed that k-wise uniform generators, with seed length $O(k \log n)$, perfectly fool depth-k decision trees. As another application of Fourier L_1 bounds, let's design another PRG for bounded-depth decision trees with a better seed length (although this time the error will be nonzero). More generally, we will consider decision trees of unbounded depth but bounded *size*. The size of a decision tree is the number of leaves (see Figure 2.2).

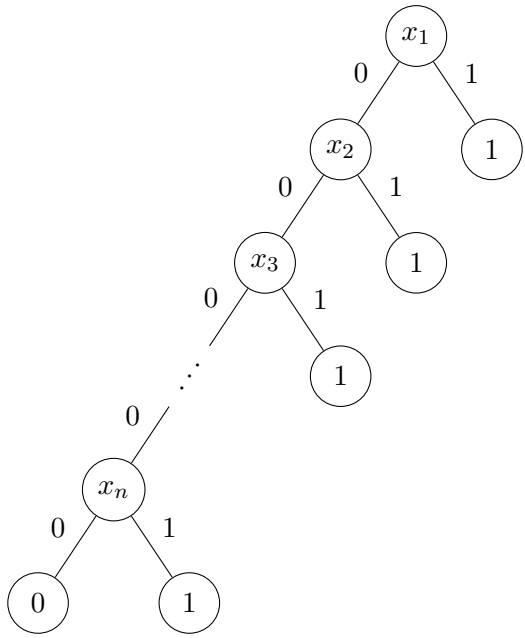

Figure 2.2: A decision tree computing the OR function on n bits. Note that the *size* of this decision tree is $n + 1$, which is relatively low, whereas the *depth* of this decision tree is maximal, which is unavoidable for the OR function.

Proposition 2.5 (PRG for bounded-size decision trees)**.** If X is a δ-biased distribution over $\{0,1\}^n$, then X fools size-m decision trees with error $m\delta$. Consequently, for every $n, m \in \mathbb{N}$ and $\varepsilon > 0$, there is an explicit ε-PRG for size-m decision trees on n input bits with seed length $O(\log(mn/\varepsilon))$.

Note that Proposition 2.5 implies a PRG for depth-k decision trees with seed length $O(k + \log(n/\varepsilon))$, because a depth-$k$ decision tree always has size at most 2^k. The proof of Proposition 2.5 is similar to the construction of almost k-wise uniform generators: we will bound the Fourier L_1 norm of size-m decision trees. We start with the special case of *conjunctions of literals*.

Proposition 2.6 (Fourier L_1 bound for conjunctions of literals)**.** Suppose $f\colon \{0,1\}^n \to \{0,1\}$ is a conjunction of literals, i.e.,

$$f(x) = \bigwedge_{i \in S} \ell_i$$

where $S \subseteq [n]$ and each ℓ_i is either x_i or $\neg x_i$. Then $L_1(f) = 1$.

Proof. There is a convenient formula for the Fourier coefficients of any function f:

$$\mathop{\mathbb{E}}_{U \sim U_n}[f(U) \cdot \chi_S(U)] = \langle f, \chi_S \rangle = \sum_{T \subseteq [n]} \widehat{f}(T) \cdot \langle \chi_T, \chi_S \rangle = \widehat{f}(S). \quad (2.5)$$

In the case $f = \mathsf{NOR}_n$ (the n-input NOR function), we get

$$\widehat{\mathsf{NOR}_n}(S) = \mathop{\mathbb{E}}_{U \sim U_n}[\mathsf{NOR}_n(U) \cdot \chi_S(U)] = 2^{-n}.$$

Therefore, $L_1(\mathsf{NOR}_n) = 2^n \cdot 2^{-n} = 1$. More generally, consider any conjunction of literals f. Without loss of generality, we may assume that all n variables appear in f. Consequently, there is some string $a \in \{0,1\}^n$ such that $f(x) = \mathsf{NOR}_n(x + a)$, where $+$ is the bitwise XOR operation. Therefore, by Equation (2.5), for each $S \subseteq [n]$,

$$\begin{aligned} \widehat{f}(S) = \mathop{\mathbb{E}}_{U}[\mathsf{NOR}_n(U + a) \cdot \chi_S(U)] &= \mathop{\mathbb{E}}_{U}[\mathsf{NOR}_n(U) \cdot \chi_S(U + a)] \\ &= \chi_S(a) \cdot \widehat{\mathsf{NOR}_n}(S) \\ &= \pm 2^{-n}. \end{aligned}$$

(In general, negating variables can only change the signs of Fourier coefficients, not the absolute values.) Therefore, $L_1(f) = 2^n \cdot 2^{-n} = 1$. □

Corollary 2.9 (Fourier L_1 bound for decision trees)**.** If f is a size-m decision tree, then $L_1(f) \leq m$.

Proof. One can verify that the Fourier $L_1(f)$ norm truly is a norm, i.e., $L_1(f + g) \leq L_1(f) + L_1(g)$ and $L_1(\lambda f) = |\lambda| \cdot L_1(f)$. Let f be a size-m decision tree. Just like in the proof of Proposition 2.1, we can write $f = \sum_{u \in A} f_u$, where A is the set of accepting leaves and $f_u(x)$ indicates whether x leads to u. Each f_u is a conjunction of literals. Therefore, $L_1(f) \leq \sum_{u \in A} L_1(f_u) \leq m$. □

Combining Theorems 2.4 and 2.9 and Lemma 2.8 completes the proof of Proposition 2.5. When $m < n$, one can improve the seed length to $O(\log(m/\varepsilon) + \log\log n)$ using m-wise δ-biased generators.

Proposition 2.5 extends to the more powerful model of *parity decision trees*, which are decision trees in which each internal node may query an arbitrary parity function of the input [146]. The reason is that we can write such a tree f as $f(x) = g(h_1(x), \ldots, h_m(x))$ where g is a size-m standard decision tree and $h_1, \ldots, h_m$ are parity functions. Consequently,

$$f(x) = \sum_{S \subseteq [n]} \widehat{g}(S) \cdot (-1)^{\sum_{i \in S} h_i(x)}.$$

For each fixed S, the function $(-1)^{\sum_{i \in S} h_i(x)}$ is a character function, so it has Fourier L_1 norm 1, and hence $L_1(f) \leq L_1(g) \leq m$.

2.3.4 Fooling width-2 branching programs

For a final application of Fourier L_1 bounds, let us obtain a PRG for *width-2 branching programs* (see Figure 2.3). Branching programs are one of the oldest sequential models of computation. For general size-m branching programs, which model computing with $\log m$ bits of memory, the current best PRG has seed length roughly $\sqrt{m}$ (see Section 5.6). We can do much better in the special case of width-2 branching programs, which model computing with a *single bit of memory* and a clock. The precise definition follows.

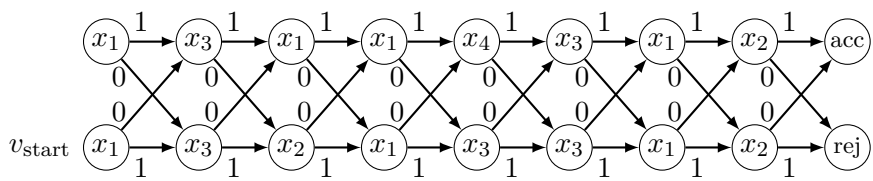

Figure 2.3: A width-2 length-8 branching program computing the function $f\colon \{0,1\}^4 \to \{0,1\}$ defined by $f(x) = 1 \iff |x| = 2$, where $|x|$ denotes Hamming weight. Note that this function cannot be computed by a width-2 *read-once* branching program. This example is derived from work by Borodin *et al.* [35].

Definition 2.8 (Bounded-width branching programs)**.** A *width-w length-m branching program* f is a directed (multi)graph with $m+1$ layers $V_0, \ldots, V_m$ of w vertices each. For $i \in [m]$, each vertex $v \in V_{i-1}$ is labeled with an index $j_v \in [n]$ and has two outgoing edges labeled 0 and 1 leading to vertices in V_i. There is a designated "start vertex" $v_{\text{start}} \in V_0$ and a designated set of "accepting vertices" $V_{\text{accept}} \subseteq V_m$.[3] Given an input $x \in \{0,1\}^n$, the program starts at v_{start}, and in each step, having reached a vertex v, the program queries x_{j_v} and traverses the corresponding outgoing edge. Eventually, the program reaches a vertex $v \in V_m$, and $f(x) = 1 \iff v \in V_{\text{accept}}$.

Branching programs of width even slightly larger than 2 are surprisingly powerful. For example, Barrington's theorem [19] says that the majority function can be computed by constant-width polynomial-length branching programs, and more generally, constant-width polynomial-length branching programs can compute exactly the functions in non-uniform $\mathbf{NC}^1$ (i.e., logarithmic-depth bounded-fan-in Boolean formulas).

On the other hand, width-2 branching programs are relatively weak, which allows us to fool them with a short seed. The following theorem is attributed to unpublished 1995 work of Saks and Zuckerman (see also the work of Bogdanov *et al.* [29]).

Theorem 2.10 (PRGs for width-2 branching programs)**.** If X is a δ-biased distribution over $\{0,1\}^n$, then X fools width-2 length-m branching programs with error $\delta \cdot (m+1)/2$. Consequently, for every $n, m \in \mathbb{N}$ and $\varepsilon > 0$, there is an explicit ε-PRG for width-2 length-m branching programs with seed length $O(\log(mn/\varepsilon))$.

One can show that every width-2 branching program on n variables can be simulated by a width-2 branching program of length $m = O(n^2)$ [35], so the seed length in Theorem 2.10 actually simplifies to $O(\log(n/\varepsilon))$. When Theorem 2.10 is mentioned in the literature, it is sometimes indicated that we should assume that the branching program is read-once [29], [105], [128], but such an assumption is not necessary. Once again, we will prove Theorem 2.10 by proving a Fourier L_1 bound.

[3] Note that we do not allow the branching program to halt prior to reaching layer m. This type of program is sometimes referred to as a "strict" width-w program [35].

Lemma 2.11 (Fourier L_1 bound for width-2 branching programs)**.** If f is a width-2 length-m branching program, then $L_1(f) \leq m/2 + 1/2$.

Proof. Let $F(x) = (-1)^{f(x)}$. For each vertex v in f, let $f_{\to v}(x)$ indicate whether $f(x)$ visits v, and let $F_{\to v}(x) = (-1)^{f_{\to v}(x)}$. We will prove by induction on m that $L_1(F) \leq m$. For the base case $m = 1$, the function F is a 1-junta, i.e., $F(x) = (-1)^{x_i}$ or $F(x) = (-1)^{1-x_i}$ or $F(x) = 1$ or $F(x) = -1$. In each case, $L_1(F) = 1$. Now, for the inductive step, let $V_{m-1} = \{u, v\}$. Then there exist 1-juntas $\phi_u, \phi_v \colon \{0,1\}^n \to \{\pm 1\}$ such that

$$\begin{aligned} F(x) &= f_{\to u}(x) \cdot \phi_u(x) + f_{\to v}(x) \cdot \phi_v(x) \\ &= \left(\frac{1}{2} - \frac{1}{2} \cdot F_{\to u}(x)\right) \cdot \phi_u(x) + \left(\frac{1}{2} + \frac{1}{2} \cdot F_{\to u}(x)\right) \cdot \phi_v(x) \\ &= \frac{1}{2} \cdot F_{\to u}(x) \cdot (\phi_v(x) - \phi_u(x)) + \frac{1}{2} \cdot (\phi_u(x) + \phi_v(x)) \,. \end{aligned}$$

Now, $L_1(\phi_u) = L_1(\phi_v) = 1$, and by induction, $L_1(F_{\to u}) \leq m - 1$. Furthermore, one can verify that the L_1 norm is submultiplicative, i.e., $L_1(g \cdot h) \leq L_1(g) \cdot L_1(h)$. Therefore,

$$\begin{aligned} L_1(F) &\leq \frac{1}{2} \cdot L_1(F_{\to u}) \cdot (L_1(\phi_v) + L_1(\phi_u)) + \frac{1}{2}(L_1(\phi_u) + L_1(\phi_v)) \\ &\leq m - 1 + 1 \\ &= m, \end{aligned}$$

completing the induction. Finally, $f(x) = \frac{1}{2} - \frac{1}{2}F(x)$, so $L_1(f) \leq \frac{1}{2} + \frac{m}{2}$. □

Theorem 2.10 follows by combining Lemmas 2.8 and 2.11 and Theorem 2.4. Analogously to the situation with decision trees, when $m < n$, one can improve the seed length to $O(\log(m/\varepsilon) + \log\log n)$ using $(2m)$-wise δ-biased generators.

2.4 Viola's Generator for Low-degree $\mathbb{F}_2$-polynomials

In Section 2.2 we saw a simple construction of an explicit small-bias generator, i.e., a PRG that fools all $\mathbb{F}_2$-linear functions with logarithmic seed length. We've discussed connections between small-bias generators

and coding theory and some simple applications of small-bias generators. As a natural generalization, let us construct PRGs for quadratic or higher degree polynomials.

Remark 2.2 (Polynomials over $\mathbb{F}_2$ vs. polynomials over $\mathbb{R}$)**.** Over the reals, every degree-d polynomial is perfectly fooled by d-wise uniform generators. However, in this section, we are working over $\mathbb{F}_2$. Thus, a low-degree polynomial is a function of the form PARITY $\circ$ AND where the AND gates have low fan-in. In this setting, k-wise uniformity is not a good approach. For instance, the uniform distribution over all strings with even Hamming weight is $(n-1)$-wise uniform, and yet it does not even fool degree-1 polynomials (parity functions).

The problem of designing PRGs for low-degree $\mathbb{F}_2$-polynomials seemed to be much harder than constructing small-bias generators or k-wise uniform generators. For a long time, even for constant degree, the best construction known was a PRG by Luby, Veličković, and Wigderson [164] with seed length $2^{O(\sqrt{\log n})}$. Over a decade later, a new line of work [32], [157] led to Viola's elegant proof [242] that simply summing d independent copies of small-bias generators gives a PRG for degree-d polynomials.

Theorem 2.12 (PRG for low-degree $\mathbb{F}_2$-polynomials [242])**.** Let $Y_1, \ldots, Y_d$ be independent δ-biased random variables over $\mathbb{F}_2^n$ where $\delta \leq 1/2$. Then $Y_1 + \cdots + Y_d$ fools degree-d $\mathbb{F}_2$-polynomials with error $4 \cdot (\delta/2)^{1/2^{d-1}}$. Consequently, for every $n, d \in \mathbb{N}$ and $\varepsilon > 0$, there is an explicit ε-PRG for degree-d $\mathbb{F}_2$-polynomials in n variables with seed length $O(d \cdot \log n + d \cdot 2^d \cdot \log(1/\varepsilon))$.

For context, it is easy to show that a sum of independent small-bias random variables is "more pseudorandom" than a single small-bias random variable in the sense that it has smaller bias (see below). Theorem 2.12 says that not only does the sum have smaller bias, it also fools higher-degree polynomials.

Observation 2.1 (XORing decreases bias)**.** Let $Y_1, \ldots, Y_d$ be independent δ-biased random variables distributed over $\mathbb{F}_2^n$. Then $\sum_{i=1}^d Y_i$ is (δ^d)-biased.

Proof. For every nonempty $S \subseteq [n]$, we have

$$\left|\mathbb{E}\left[\chi_S\left(\sum_{i=1}^{d} Y_i\right)\right]\right| = \left|\prod_{i=1}^{d} \mathbb{E}[\chi_S(Y_i)]\right| \leq \delta^d. \qquad \square$$

2.4.1 Directional derivatives

The proof of Theorem 2.12 is based on the notion of *directional derivatives* over $\mathbb{F}_2$, defined below.

Definition 2.9 (Directional derivative)**.** Let $f\colon \mathbb{F}_2^n \to \mathbb{F}_2$ and $y \in \mathbb{F}_2^n$. The *directional derivative* $\partial_y f$ is defined by

$$\partial_y f(x) = f(x+y) + f(x).$$

If $\mathcal{F}$ is a class of functions $f\colon \mathbb{F}_2^n \to \mathbb{F}_2$, we define

$$\partial\mathcal{F} = \{\partial_y f : f \in \mathcal{F}, y \in \mathbb{F}_2^n\}.$$

To fool low-degree polynomials, our strategy will be to show how to convert PRGs for $\partial\mathcal{F}$ into PRGs for $\mathcal{F}$, where $\mathcal{F}$ is any "reasonable" class. Formally, the only requirement on $\mathcal{F}$ is that is "closed under shifts," as defined below.

Definition 2.10 (Closure under shifts)**.** For a function f on $\mathbb{F}_2^n$ and a vector $y \in \mathbb{F}_2^n$, we define the *shift* f^{+y} by the formula $f^{+y}(x) = f(x+y)$. Let $\mathcal{F}$ be a class of functions f on $\mathbb{F}_2^n$. We say that $\mathcal{F}$ is *closed under shifts* if for every $f \in \mathcal{F}$ and every $y \in \mathbb{F}_2^n$, we have $f^{+y} \in \mathcal{F}$.

Lemma 2.13 (PRG for $\partial\mathcal{F} \implies$ PRG for $\mathcal{F}$)**.** Let $\mathcal{F}$ be a class of functions $f\colon \mathbb{F}_2^n \to \mathbb{F}_2$ that is closed under shifts. Suppose W fools $\partial\mathcal{F}$ with error γ, Y is δ-biased, and Y is independent of W. Then $W + Y$ fools $\mathcal{F}$ with error $\sqrt{2\gamma} + \delta/2$.

In general, $\partial\mathcal{F}$ seems to be "more complicated" than $\mathcal{F}$ itself, so Lemma 2.13 might not sound particularly useful. However, $\partial\mathcal{F}$ is "simpler" than $\mathcal{F}$ in one respect, namely degree:

Observation 2.2 (Differentiation decreases degree)**.** Let $d \geq 1$, let $f\colon \mathbb{F}_2^n \to \mathbb{F}_2$ be a degree-d polynomial, and let $y \in \mathbb{F}_2^n$. Then $\partial_y f$ is a degree-$(d-1)$ polynomial.

Thus, we will be able to prove Theorem 2.12 by applying Lemma 2.13 inductively.

2.4.2 The XOR of two independent copies of an arbitrary Boolean function

The proof of Lemma 2.13 (the reduction from fooling $\mathcal{F}$ to fooling $\partial\mathcal{F}$) relies on the following lemma, which explains how to use small-bias distributions to "recycle" randomness and thereby fool a certain class of functions.

Lemma 2.14 (Using small-bias distributions to fool $g(x) \cdot g(y)$)**.** Let n be an even positive integer, let $g\colon \{0,1\}^{n/2} \to \{\pm 1\}$, and let $f(x,y) = g(x) \cdot g(y)$. Let U and Y be independent, where $U \sim U_{n/2}$ and Y is an ε-biased random variable over $\mathbb{F}_2^{n/2}$. Then $(U, U+Y)$ fools f with error ε.

Proof. Define $F\colon \{0,1\}^{n/2} \to [-1,1]$ by

$$F(x) = \mathop{\mathbb{E}}_{U}[g(U) \cdot g(U+x)],$$

and note that $\mathbb{E}[F] = \mathbb{E}[f]$. Let $U' \sim U_{n/2}$ be independent of U. Then for any $S \subseteq [n]$, by Equation (2.5),

$$\begin{aligned}
\widehat{F}(S) &= \mathop{\mathbb{E}}_{U,U'}[g(U) \cdot g(U+U') \cdot \chi_S(U')] \\
&= \mathop{\mathbb{E}}_{U,U'}[g(U) \cdot g(U') \cdot \chi_S(U+U')] \\
&= \left(\mathop{\mathbb{E}}_{U}[g(U) \cdot \chi_S(U)]\right)^2 \\
&= \widehat{g}(S)^2.
\end{aligned}$$

Therefore, $L_1(F) = \sum_S \widehat{g}(S)^2 \le 1$ by Parseval's theorem. Consequently, Y fools F with error ε by Lemma 2.8, and hence $(U, U+Y)$ fools f with error ε. □

Remark 2.3 (Characterizing small bias)**.** One can show the following converse to Lemma 2.14: If $U \sim U_{n/2}$, Y is independent, and $(U, U+Y)$ fools all functions of the form $f(x,y) = g(x) \cdot g(y) \in \{\pm 1\}$ with error ε, then Y is ε-biased. Thus, the condition in Lemma 2.14 gives an alternative *characterization* of small-bias distributions.

Remark 2.4 (Connection to expander graphs)**.** The seed length for sampling the distribution $(U, U+Y)$ that appears in Lemma 2.14 is $n/2 + O(\log(n/\varepsilon))$. Lemma 2.14 generalizes to the case that g has bounded variance, $\mathrm{Var}[g] \le 1$, rather than being $\{\pm 1\}$-valued. This generalization is closely related to the notion of a *spectral expander*. In Section 3.1.1, we will discuss spectral expanders in more detail, and in particular we will discuss PRGs for such tests with the improved seed length $n/2 + O(\log(1/\varepsilon))$.

2.4.3 The reduction from fooling $\mathcal{F}$ to fooling $\partial\mathcal{F}$

To prove Lemma 2.13, we will consider two cases based on the extent to which $f \in \mathcal{F}$ is balanced. For a function $f\colon \mathbb{F}_2^n \to \mathbb{F}_2$, define

$$\mathrm{imbalance}(f) = \left|\mathbb{E}\left[(-1)^{f(U_n)}\right]\right| = 2 \cdot \left|\mathbb{E}[f] - \frac{1}{2}\right|.$$

(In the literature, this quantity is often referred to as the "bias" of f. We use the term "imbalance" instead to avoid confusion with small-bias distributions.) We begin with the case that f is close to balanced.

Lemma 2.15 (Fooling well-balanced functions)**.** Let $\mathcal{F}$ be a class of functions $f\colon \mathbb{F}_2^n \to \mathbb{F}_2$ that is closed under shifts. Suppose W fools $\partial\mathcal{F}$ with error γ, Y is δ-biased, and Y is independent of W. Then $W + Y$ fools each $f \in \mathcal{F}$ with error $\mathrm{imbalance}(f) + \sqrt{\gamma/2} + \delta/2$.

Proof. First observe that

$$\begin{aligned}
|\mathbb{E}[f(W+Y)] - \mathbb{E}[f(U_n)]| &= \frac{1}{2} \cdot \left|\mathbb{E}\left[(-1)^{f(W+Y)}\right] - \mathbb{E}\left[(-1)^{f(U_n)}\right]\right| \\
&\le \frac{1}{2} \cdot \left|\mathbb{E}\left[(-1)^{f(W+Y)}\right]\right| + \frac{1}{2} \cdot \mathrm{imbalance}(f).
\end{aligned}$$

Thus, it suffices to bound $|\mathbb{E}[(-1)^{f(W+Y)}|$. By Jensen's inequality,

$$\begin{aligned}
\left(\mathop{\mathbb{E}}_{W,Y}\left[(-1)^{f(W+Y)}\right]\right)^2 &\le \mathop{\mathbb{E}}_{W}\left[\left(\mathop{\mathbb{E}}_{Y}\left[(-1)^{f(W+Y)}\right]\right)^2\right] \\
&= \mathop{\mathbb{E}}_{W,Y,Y'}\left[(-1)^{f(W+Y)+f(W+Y')}\right],
\end{aligned}$$

where Y' is an independent copy of Y. For any fixed y, the function $f^{+y}(x) \stackrel{\mathrm{def}}{=} f(x+y)$ is in $\mathcal{F}$ because $\mathcal{F}$ is closed under shifts. Furthermore,

for fixed y, y', the function $g(x) \stackrel{\text{def}}{=} f(x+y)+f(x+y')$ is in $\partial\mathcal{F}$, because $g = \partial_{y+y'} f^{+y}$. Therefore, the assumption on W gives

$$\mathop{\mathbb{E}}_{W,Y,Y'}\left[(-1)^{f(W+Y)+f(W+Y')}\right] \le \mathop{\mathbb{E}}_{\substack{Y,Y',\\ U\sim U_n}}\left[(-1)^{f(U+Y)+f(U+Y')}\right] + 2\gamma.$$

Finally, observing that $(U+Y, U+Y')$ is identically distributed to $(U, U+Y+Y')$, we have

$$\begin{aligned}\mathop{\mathbb{E}}_{\substack{Y,Y',\\ U\sim U_n}}\left[(-1)^{f(U+Y)+f(U+Y')}\right] &= \mathop{\mathbb{E}}_{\substack{Y,Y',\\ U\sim U_n}}\left[(-1)^{f(U)+f(U+Y+Y')}\right] \\ &\le \text{imbalance}(f)^2 + \delta^2,\end{aligned}$$

where the last inequality follows from Lemma 2.14 and the fact that $Y+Y'$ is (δ^2)-biased. In summary, we have shown that

$$\begin{aligned}&|\mathbb{E}[f(W+Y)] - \mathbb{E}[f(U_n)]| \\ &\qquad \le \frac{1}{2}\cdot\left|\mathop{\mathbb{E}}_{W,Y}\left[(-1)^{f(W+Y)}\right]\right| + \frac{1}{2}\cdot\text{imbalance}(f) \\ &\qquad \le \frac{1}{2}\cdot\sqrt{\text{imbalance}(f)^2+\delta^2+2\gamma} + \frac{1}{2}\cdot\text{imbalance}(f) \\ &\qquad \le \text{imbalance}(f) + \frac{\delta}{2} + \sqrt{\frac{\gamma}{2}}.\end{aligned}$$

□

Now we move on to the case that f is significantly imbalanced. In this case, W alone (rather than $W+Y$) already fools f.

Lemma 2.16 (Fooling imbalanced functions)**.** Let $\mathcal{F}$ be any class of functions $f\colon \mathbb{F}_2^n \to \mathbb{F}_2$. Suppose W fools $\partial\mathcal{F}$ with error γ. Then W fools each $f \in \mathcal{F}$ with error $\gamma\cdot\text{imbalance}(f)^{-1}$.

Proof. Let U and U' be two independent copies of U_n. Then

$$\begin{aligned}&\text{imbalance}(f)\cdot\left|\mathbb{E}\left[(-1)^{f(W)}\right] - \mathbb{E}\left[(-1)^{f(U)}\right]\right| \\ &\qquad = \left|\mathbb{E}\left[(-1)^{f(W)+f(U)}\right] - \mathbb{E}\left[(-1)^{f(U)+f(U')}\right]\right| \\ &\qquad = \left|\mathbb{E}\left[(-1)^{f(W)+f(W+U)}\right] - \mathbb{E}\left[(-1)^{f(U')+f(U'+U)}\right]\right| \\ &\qquad \le 2\gamma,\end{aligned}$$

where the last inequality is due to the fact that for any fixing of U, the function $\partial_U f$ is fooled by W. □

Now we combine the two cases to complete the proof of the reduction.

Proof of Lemma 2.13. For any $f \in \mathcal{F}$ and any fixing of Y, the function $f^{+Y}(x) \stackrel{\text{def}}{=} f(x+Y)$ is in $\mathcal{F}$, and $\text{imbalance}(f^{+Y}) = \text{imbalance}(f)$. Therefore, Lemma 2.16 implies that $W + Y$ fools f with error $\gamma \cdot \text{imbalance}(f)^{-1}$. Combining with Lemma 2.15 shows that $W + Y$ fools f with error

$$\min\left\{\gamma \cdot \text{imbalance}(f)^{-1}, \text{imbalance}(f) + \sqrt{\gamma/2} + \delta/2\right\} \leq \sqrt{2\gamma} + \delta/2,$$

where the last inequality follows by case analysis based on whether $\text{imbalance}(f) \leq \sqrt{\gamma/2}$. □

2.4.4 Inductive analysis of low-degree polynomials

Proof of Theorem 2.12. By Lemma 2.13 and Observation 2.2, for every i, the random variable $\sum_{j=1}^{i} Y_j$ fools degree-i polynomials with error ε_i, where $\varepsilon_1 = \delta/2$ and $\varepsilon_{i+1} = \sqrt{2\varepsilon_i} + \delta/2$. Since $\delta \leq 1/2$, we get $\varepsilon_{i+1} \leq \sqrt{2\varepsilon_i} + \sqrt{\delta/2}/2$. Since $\varepsilon_i \geq \delta/2$, we get $\varepsilon_{i+1} \leq (\sqrt{2} + 1/2) \cdot \sqrt{\varepsilon_i} \leq 2\sqrt{\varepsilon_i}$. It follows that

$$\varepsilon_d \leq 4 \cdot (\delta/2)^{1/2^{d-1}}.$$

The seed length bound follows by choosing $\delta = 2 \cdot (\varepsilon/4)^{2^{d-1}}$ and using the small-bias generator construction of Theorem 2.4. □

When d is constant, the seed length in Theorem 2.12 is optimal. However, the generator becomes trivial when $d = \Theta(\log n)$.

Open Problem 2.2 (PRGs for logarithmic-degree polynomials)**.** Design an explicit nontrivial PRG for $\mathbb{F}_2$-polynomials of degree $\log n$.

Open Problem 2.2 is closely related to the challenge of proving better *correlation bounds* against polynomials; see Viola's survey [246].

2.4.5 Application: Width-2 branching programs that read several bits at a time

Studying low-degree polynomials is natural enough from a mathematical perspective, but what about from a computing perspective? The reader

might find it strange to think of polynomials as a computational model. However, we will now show that PRGs for low-degree polynomials imply PRGs for other models of a more "computational" nature, which demonstrates the importance of Viola's PRG. In particular, we can fool *compositions with juntas*, provided that the outer function has bounded Fourier L_1 norm.

Definition 2.11 (Compositions with juntas)**.** Let $f\colon \{0,1\}^r \to \mathbb{R}$. For each $n, d \in \mathbb{N}$, we define $f \circ \mathsf{JUNTA}_{n,d}$ to be the class of all functions $g\colon \{0,1\}^n \to \mathbb{R}$ of the form

$$g(x) = f(\phi_1(x), \ldots, \phi_r(x)),$$

where each ϕ_i is a d-junta on n bits. If $\mathcal{F}$ is a class of functions $f\colon \{0,1\}^r \to \mathbb{R}$, then we define $\mathcal{F} \circ \mathsf{JUNTA}_{n,d} = \bigcup_{f \in \mathcal{F}} f \circ \mathsf{JUNTA}_{n,d}$.

Lemma 2.17 (PRGs for compositions with juntas)**.** Suppose X is a distribution over $\{0,1\}^n$ that fools degree-d polynomials over $\mathbb{F}_2^n$ with error ε, and let $f\colon \{0,1\}^r \to \mathbb{R}$. Then X fools $f \circ \mathsf{JUNTA}_{n,d}$ with error $2\varepsilon \cdot L_1(f)$.

Proof. If $g(x) = f(\phi_1(x), \ldots, \phi_r(x))$, then by the Fourier expansion of f, we have

$$g(x) = \sum_{S \subseteq [r]} \widehat{f}(S) \cdot (-1)^{\sum_{i \in S} \phi_i(x)}.$$

The summation in the exponent may be performed modulo 2. If each ϕ_i is a d-junta, then each ϕ_i can be computed by a degree-d polynomial over $\mathbb{F}_2$, hence $\sum_{i \in S} \phi_i(x) \bmod 2$ is also a degree-d polynomial over $\mathbb{F}_2$. Therefore, X fools $(-1)^{\sum_{i \in S} \phi_i(x)}$ with error 2ε. The lemma follows by the Triangle Inequality for PRG Errors. □

Probably the most interesting example is when we take $\mathcal{F}$ to be the class of width-2 length-m branching programs on $2m$ input bits (see Definition 2.8). Then $\mathcal{F} \circ \mathsf{JUNTA}_{n,d}$ is precisely the class of functions computable by a variant model of width-2 length-m branching programs in which *the program reads d bits at a time*, i.e., each vertex v is labeled by a set of indices $J_v \subseteq [n]$ with $|J_v| = d$ and has 2^d outgoing edges

corresponding to the possible values of the input substring x_{J_v}. For this model, Theorem 2.12 and Lemmas 2.11 and 2.17 imply a seed length of $O(d \cdot \log n + d \cdot 2^d \cdot \log(m/\varepsilon))$. This was shown by Bogdanov *et al.* [29]. (They assume that the program is "oblivious" in the sense that $J_u = J_v$ if u and v are in the same layer, but such an assumption is not necessary: because of the way we defined $\mathcal{F}$, a program in $\mathcal{F}$ can query a distinct variable at each of its $2m$ vertices.)

2.5 Analysis Technique: Sandwiching Approximators

2.5.1 The sandwiching lemma

Suppose we wish to show that a distribution X fools some class $\mathcal{F}$. A common approach has two steps:

1. Prove that X fools some "simpler" class $\mathcal{F}_{\text{simp}}$.
2. Prove a "transfer theorem," saying that every distribution that fools $\mathcal{F}_{\text{simp}}$ also fools $\mathcal{F}$ (possibly with some loss in the error parameter).

The second step requires showing that $\mathcal{F}_{\text{simp}}$ can "simulate" $\mathcal{F}$ in some sense. For example, several times, we have shown that every function in some class of interest in can be written as a *linear combination* of "simpler" functions:

- Every depth-k decision tree can be written as a sum of k-juntas (Proposition 2.1).
- Every Boolean function can be written as a linear combination of parity functions (Proposition 2.4).
- Every width-2 branching program that reads several bits at a time can be written as a linear combination of low-degree polynomials over $\mathbb{F}_2$ (Section 2.4.5).

In each case, the Triangle Inequality for PRG Errors gives us our desired transfer theorem. (The final error depends on the magnitude of the coefficients in the linear combination.) In this section, we present a

second method for proving a "transfer theorem" stating that every distribution that fools $\mathcal{F}_{\text{simp}}$ also fools $\mathcal{F}$.

Suppose X is a distribution that fools $\mathcal{F}_{\text{simp}}$, and suppose that $\mathcal{F}_{\text{simp}}$ *approximately* simulates $\mathcal{F}$ in some sense. For example, suppose that for every $f \in \mathcal{F}$, there is an $f' \in \mathcal{F}_{\text{simp}}$ such that $\mathbb{E}[|f - f'|]$ is small. Unfortunately, it does not immediately follow that X fools $\mathcal{F}$: although f and f' behave similarly under the uniform distribution, it isn't clear whether they behave similarly under the pseudorandom distribution X. A technique for getting around this issue is to establish a stronger form of approximation called *sandwiching*.

Definition 2.12 (Sandwiching)**.** Let $f, f_\ell, f_u \colon \{0,1\}^n \to \mathbb{R}$. We say that f is δ-*sandwiched* between f_ℓ and f_u if $f_\ell \le f \le f_u$ and $\mathbb{E}[f_u - f_\ell] \le \delta$. In this case, we refer to f_ℓ and f_u as "sandwichers" or "sandwiching approximators" for f.

Lemma 2.18 (Sandwiching Lemma)**.** Suppose f is δ-sandwiched between f_ℓ and f_u, and suppose X fools f_ℓ and f_u with error ε. Then X fools f with error $\varepsilon + \delta$.

Proof.

$$\mathbb{E}[f(X)] \le \mathbb{E}[f_u(X)] \le \mathbb{E}[f_u] + \varepsilon \le \mathbb{E}[f] + \varepsilon + \delta$$
$$\mathbb{E}[f(X)] \ge \mathbb{E}[f_\ell(X)] \ge \mathbb{E}[f_\ell] - \varepsilon \ge \mathbb{E}[f] - \varepsilon - \delta.$$

□

Thus, we have two techniques for showing that every distribution that fools one class $\mathcal{F}_{\text{simp}}$ also fools another class $\mathcal{F}$: the Triangle Inequality for PRG Errors and the Sandwiching Lemma. It turns out that *these are the only two techniques that are ever necessary.* That is, if every distribution that fools $\mathcal{F}_{\text{simp}}$ also fools $\mathcal{F}$, then that fact can be proven by sandwiching each $f \in \mathcal{F}$ between linear combinations of functions in $\mathcal{F}_{\text{simp}}$. See Appendix A for details.

2.5.2 Using k-wise uniform generators to fool size-m decision trees

To illustrate the sandwiching technique, let us return to the decision tree model. Recall that we showed that k-wise uniform generators fool

depth-k decision trees (Proposition 2.1), and then later we showed that small-bias generators fool size-m decision trees (Proposition 2.5). The latter model generalizes the former by taking $m = 2^k$. We now show that k-wise uniform generators also fool bounded-*size* decision trees.

Proposition 2.7 (Limited independence fools bounded-size decision trees)**.** If X is a k-wise uniform distribution, then X fools size-m decision trees with error $m \cdot 2^{-k}$.

Proof. Let f be a size-m decision tree. Define a depth-k decision tree f_ℓ by starting with f and replacing each internal node at depth exactly k with a leaf labeled 0 (and deleting all of its descendants). Similarly, define f_u by replacing each internal node at depth k with a leaf labeled 1. Let us show that f is δ-sandwiched between f_ℓ and f_u, for $\delta = m \cdot 2^{-k}$.

Clearly $f_\ell \le f \le f_u$. For each "new" leaf u of f_ℓ or f_u (i.e., u was not a leaf in f), the probability of reaching u on a uniform random input is precisely 2^{-k}. The number of new leaves is the number of internal nodes of f at depth k, which is at most m. Therefore, by the union bound, $\mathbb{E}[f_u - f_\ell] \le m \cdot 2^{-k}$.

The Sandwiching Lemma completes the proof, because X fools f_ℓ and f_u with error 0 (see Section 2.1.3). □

Proposition 2.7 implies that using k-wise uniform generators, we can ε-fool size-m decision trees using a seed of length $O(\log(m/\varepsilon) \cdot \log n)$. This seed length is *inferior* to the seed length that we obtained previously using small-bias generators, which was $O(\log(mn/\varepsilon))$ (see Proposition 2.5). However, sometimes it is useful to understand the effect of specific classes of distributions, such as k-wise uniform distributions, on a given model of computation.

2.5.3 Small-bias distributions fool read-once $\mathbf{AC^0}$

For a more sophisticated example of a sandwiching argument, let us consider "$\mathbf{AC^0}$ circuits," i.e., bounded-depth Boolean circuits of unbounded fan-in.

Definition 2.13 ($\mathbf{AC^0}$ circuits)**.** An $\mathbf{AC^0}$ *circuit* is a directed acyclic graph where every input node is labeled by a literal (x_i or $\neg x_i$) or a

constant (0 or 1), every internal node ("gate") is labeled by $\wedge$ or $\vee$, and there is exactly one output node. The in-degrees (also called fan-ins) of $\wedge$ or $\vee$ gates are not bounded. The *size* of the circuit is the total number of its $\wedge$ and $\vee$ gates. The *depth* of the circuit is the length of its longest directed path.

Traditionally, the expression "$\mathbf{AC}^0$" refers to the *complexity class* consisting of all languages that can be decided by constant-depth polynomial size families of unbounded-fan-in circuits. As suggested by Definition 2.13, we will instead adopt the convenient convention of speaking of "size-m depth-d $\mathbf{AC}^0$ circuits," where m is not necessarily $\text{poly}(n)$ and d is not necessarily $O(1)$. That being said, $m = \text{poly}(n)$ and $d = O(1)$ is the parameter regime in which we are most interested.

Later, we will present PRGs for general $\mathbf{AC}^0$ circuits (see Sections 2.6, 4.2, 5.1 and 5.3). For now, let us focus on fooling the *read-once* version of $\mathbf{AC}^0$, a substantially easier problem. A *read-once* $\mathbf{AC}^0$ *formula* is an $\mathbf{AC}^0$ circuit in which every variable appears at most once and the underlying graph structure is a tree. See Figure 2.4.

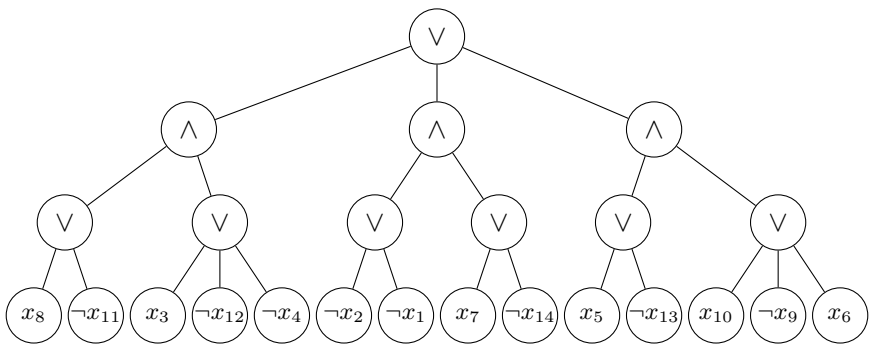

Figure 2.4: A depth-3 read-once $\mathbf{AC}^0$ formula.

Theorem 2.19 (Small-bias fools read-once $\mathbf{AC}^0$)**.** For every $n, d \in \mathbb{N}$ and $\varepsilon > 0$ with $d \geq 2$, there is a value $\delta = \exp(-\Theta(\log n)^{d-1} \cdot \log(1/\varepsilon))$ such that if X is a δ-biased distribution over $\{0,1\}^n$, then X fools depth-d read-once $\mathbf{AC}^0$ formulas with error ε. Consequently, there is an explicit ε-PRG for depth-d read-once $\mathbf{AC}^0$ formulas with seed length $O(\log n)^{d-1} \cdot \log(1/\varepsilon)$.

When $d = 2$ (read-once CNFs and DNFs) and ε is constant, the seed length of Theorem 2.19 is $O(\log n)$, which is optimal. For larger d or smaller ε, the seed length is not optimal: the optimal seed length would be $O(\log(n/\varepsilon))$, independent of depth (note we always have $d \leq n$). That being said, a benefit of Theorem 2.19 is the simplicity of the PRG itself. See Section 5.5 for a discussion of more sophisticated PRGs for read-once $\mathbf{AC}^0$ with better seed lengths.

The case $d = 2$ of Theorem 2.19 was first explicitly stated and proven by De *et al.* [75]. It also readily follows [151, Appendix A] from earlier work by Chari *et al.* [44]. It seems that the case $d > 2$ does not appear in the literature, but the argument for $d > 2$ is a straightforward generalization of the argument for $d = 2$.

The proof of Theorem 2.19 works by repeatedly applying the following lemma.

Lemma 2.20 (PRG for depth $d \implies$ PRG for depth $d+1$)**.** Suppose a distribution X fools depth-d read-once $\mathbf{AC}^0$ formulas with error ε, where $d \geq 1$. Then X fools depth-$(d+1)$ read-once $\mathbf{AC}^0$ formulas with error $\exp\left(-\Omega\left(\frac{\log(1/\varepsilon)}{\log n}\right)\right)$.

Proof. Let f be a depth-$(d+1)$ read-once $\mathbf{AC}^0$ formula. By merging gates and introducing dummy gates of fan-in 1, we may assume that f alternates between layers of $\wedge$ gates and layers of $\vee$ gates. Assume for now that the output gate of f is $\vee$, so we can write $f(x) = f_1(x) \vee \cdots \vee f_m(x)$. Define the *weight* of such a formula to be the expected number of terms satisfied on a uniform random input, i.e., $\mathrm{Weight}(f) = \sum_{i=1}^{m} \mathbb{E}[f_i]$. As a first step, we will show that for every even positive integer k, the distribution X fools f with error

$$\varepsilon \cdot (2m)^k + (e \cdot \mathrm{Weight}(f)/k)^k. \tag{2.6}$$

To prove it, let us use the inclusion-exclusion principle to compute $f(x)$. For each positive integer r, define $\psi_r \colon \{0,1\}^n \to \mathbb{R}$ by

$$\psi_r(x) = \sum_{t=1}^{r} (-1)^{t-1} \sum_{\substack{S \subseteq [m] \\ |S|=t}} \bigwedge_{i \in S} f_i(x).$$

Since k is even, $\psi_k \le f \le \psi_{k-1}$, and we claim that

$$\mathbb{E}[\psi_{k-1} - \psi_k] \le (e \cdot \text{Weight}(f)/k)^k. \tag{2.7}$$

Indeed, if $k > m$, then Equation (2.7) holds because $\psi_{k-1} \equiv \psi_k \equiv f$, and meanwhile if $k \le m$, then

$$\begin{aligned}\mathbb{E}[\psi_{k-1} - \psi_k] = \sum_{\substack{S \subseteq [m] \\ |S|=k}} \prod_{i \in S} \mathbb{E}[f_i] &\le \binom{m}{k} \cdot \left(\frac{\sum_{i=1}^m \mathbb{E}[f_i]}{m}\right)^k \\ &\le \left(\frac{em}{k}\right)^k \cdot \left(\frac{\text{Weight}(f)}{m}\right)^k \\ &= (e \cdot \text{Weight}(f)/k)^k.\end{aligned}$$

(The first inequality follows from Maclaurin's inequality.) Thus, f is sandwiched between ψ_k and ψ_{k-1}. Furthermore, since the top gate of each f_i is $\wedge$, each function $\bigwedge_{i \in S} f_i(x)$ is a depth-d read-once $\mathbf{AC}^0$ formula. Therefore, by the Triangle Inequality for PRG Errors, X fools ψ_r with error δ_r where

$$\delta_r = \varepsilon \cdot \sum_{t=1}^{r} \binom{m}{t} = \varepsilon \cdot \binom{m+r-1}{r} \le \varepsilon \cdot (m+r)^r.$$

Since $\psi_r = \psi_m$ for all $r \ge m$, it follows that X fools ψ_r with error $\varepsilon \cdot (2m)^r$. Therefore, by the sandwiching lemma (Lemma 2.18), X fools f with the error given by Equation (2.6).

Now let

$$k_* = \frac{\log(1/\varepsilon)}{2\log(2m)},$$

or to be more precise, let k_* be the smallest even positive integer that is at least the above value. We split into two cases. For the first case, suppose $\text{Weight}(f) \le k_*/(2e)$. Then we achieve error

$$\varepsilon \cdot (2m)^{k_*} + 2^{-k_*} = \exp\left(-\Omega\left(\frac{\log(1/\varepsilon)}{\log m}\right)\right).$$

Since f is read-once, $m \le n$, so this error value is sufficient to establish the lemma. For the second case, suppose $\text{Weight}(f) > k_*/(2e)$. Let

$f'(x) = f_1(x) \vee \cdots \vee f_{m'}(x)$, where m' is the largest value such that $\text{Weight}(f') \le k_*/(2e)$. Then $f' \le f \le 1$, and

$$\mathbb{E}[1 - f'] = \prod_{i=1}^{m'} (1 - \mathbb{E}[f_i]) \le e^{-\text{Weight}(f')} \le e^{-(\frac{k_*}{2e} - 1)}.$$

Therefore, f is δ-sandwiched between f' and 1, where

$$\delta = \exp\left(-\Omega\left(\frac{\log(1/\varepsilon)}{\log m}\right)\right).$$

Furthermore, X fools f' with error

$$\varepsilon \cdot (2m')^{k_*} + 2^{-k_*} \le \varepsilon \cdot (2m)^{k_*} + 2^{-k_*} = \exp\left(-\Omega\left(\frac{\log(1/\varepsilon)}{\log m}\right)\right),$$

and obviously X fools 1 with error 0, so another application of the sandwiching lemma completes the proof in this case.

Finally, suppose the output gate of f is $\wedge$. Then $\neg f$ can be computed by a depth-$(d+1)$ read-once formula where the output gate is $\vee$. Therefore, X fools $\neg f$, and hence it fools f with the same error. □

Remark 2.5. More generally, we can consider any class $\mathcal{F}$ of Boolean functions on n bits. (The interesting case is when $\mathcal{F}$ is *not* closed under complement.) Let $\mathsf{AND} \diamond \mathcal{F}$ denote the "read-once composition" of AND with $\mathcal{F}$, i.e., the class of functions of the form $f(x) = \bigwedge_{i=1}^{t} f_i(x)$ where $f_1, \ldots, f_t \in \mathcal{F}$ and $f_1, \ldots, f_t$ depend on disjoint parts of the input. Define $\mathsf{OR} \diamond \mathcal{F}$ similarly. The proof of Lemma 2.20 shows that if X fools $\mathsf{AND} \diamond \mathcal{F}$ with error ε, then X fools $\mathsf{OR} \diamond \mathcal{F}$ with error $\exp(-\Omega(\log(1/\varepsilon)/\log n))$.

Proof of Theorem 2.19. By Proposition 2.6, if f is a depth-1 read-once $\mathbf{AC}^0$ formula, then either $L_1(f) \le 1$ or $L_1(\neg f) \le 1$. Either way, every δ-biased distribution fools f with error δ. This is the base case of an induction on d, where Lemma 2.20 is the inductive step. □

We can also consider read-k depth-d $\mathbf{AC}^0$ circuits for $k > 1$. Servedio and Tan studied the case $d = 2$ [210], improving on previous work by Klivans *et al.* [143]. Both works show that small-bias distributions fool read-k CNFs and DNFs; in the case of polynomial-size DNFs, Servedio and Tan's analysis [210] leads to a seed length of $\text{poly}(k, \log(1/\varepsilon)) \cdot \log n$. The case of larger depth $d > 2$ is open.

Open Problem 2.3 (PRGs for read-twice $\mathbf{AC}^0$ circuits)**.** Design an explicit PRG for read-twice depth-d $\mathbf{AC}^0$ circuits with a better seed length than the state-of-the-art PRG for general depth-d $\mathbf{AC}^0$ circuits [166].

2.6 Braverman's Theorem: Limited Independence Fools $\mathbf{AC}^0$

In Section 2.5.3, we presented a PRG for read-once $\mathbf{AC}^0$ formulas. In this section, we will present a PRG for general (read-many) $\mathbf{AC}^0$ circuits (see Figure 2.5). In particular, we will show that every k-wise uniform generator fools constant-depth polynomial-size $\mathbf{AC}^0$ circuits for a suitable $k = \text{polylog}(n)$. This was first conjectured by Linial and Nisan [156]. Two decades later, it was proved to be true for depth-2 circuits by Bazzi [20] and a simpler proof of this was discovered by Razborov [200]. Building on this line of work, Braverman [36] proved that k-wise independence for polylogarithmic k fools $\mathbf{AC}^0$ circuits. The parameters were subsequently improved by Tal [228] and Harsha and Srinivasan [114], leading to the following.

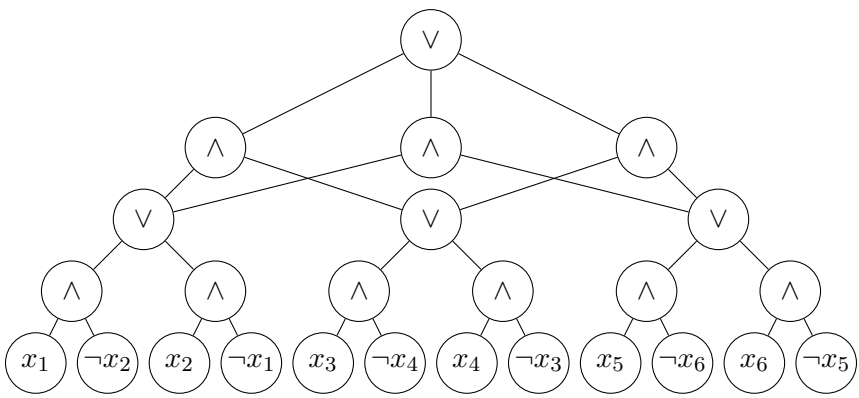

Figure 2.5: A depth-4 size-13 $\mathbf{AC}^0$ circuit computing the function $f(x) = \mathsf{MAJ}(x_1 \oplus x_2, x_3 \oplus x_4, x_5 \oplus x_6)$.

Theorem 2.21 (Braverman's theorem [36], [114], [228])**.** For every $n, m, d \in \mathbb{N}$ and $\varepsilon > 0$, there is a value $k = (\log m)^{O(d)} \cdot \log(1/\varepsilon)$ such that if X is a k-wise uniform distribution over $\{0,1\}^n$, then X fools size-m

depth-d $\mathbf{AC}^0$ circuits with error ε. Consequently, there is an explicit ε-PRG for depth-d size-m $\mathbf{AC}^0$ circuits with seed length

$$(\log m)^{O(d)} \cdot \log n \cdot \log(1/\varepsilon).$$

For context, Braverman's theorem represents neither the first nor the best unconditional PRG for $\mathbf{AC}^0$ known. Instead, the advantage of Braverman's theorem is that k-wise uniformity is a particularly *simple and general* PRG construction. That being said, the *analysis* is quite nontrivial, as we will see.

We will present two proofs of Braverman's theorem. First, we present a novel proof that is arguably simpler[4] than previous proofs, but it does not give the best parameters – the value of k will be slightly worse than what Theorem 2.21 promises. Then, we will present the known state-of-the-art proof [36], [114], [228].

2.6.1 LMN polynomials

Observe that every degree-k polynomial over the reals is perfectly fooled by k-wise uniform generators. To prove that k-wise uniform generators fool $\mathbf{AC}^0$ circuits, our approach will be to show that $\mathbf{AC}^0$ circuits are sandwiched between degree-k polynomials. Our starting point is the Linial-Mansour-Nisan theorem [155] and its subsequent improvements [33], [117], [228], which show that $\mathbf{AC}^0$ circuits can indeed be *approximated* by bounded low-degree polynomials in the L_2 norm.

Theorem 2.22 (LMN polynomials [228])**.** Let $f\colon \{0,1\}^n \to \{0,1\}$ be computable by a size-m depth-d $\mathbf{AC}^0$ circuit and let $\gamma > 0$. There exists $\widetilde{f}\colon \{0,1\}^n \to \mathbb{R}$ such that:

1. (L_2 approximation) We have
$$\left\|f - \widetilde{f}\right\|_2^2 \stackrel{\text{def}}{=} \mathop{\mathbb{E}}_{x \sim U_n}\left[\left|f(x) - \widetilde{f}(x)\right|^2\right] \leq \gamma. \tag{2.8}$$
2. (Low-degree) The degree of $\widetilde{f}$ as a polynomial over $\mathbb{R}$ is bounded by
$$\deg(\widetilde{f}) \leq O(\log m)^{d-1} \cdot \log(1/\gamma).$$

[4] In particular, the new proof does not require "probabilistic polynomials" for $\mathbf{AC}^0$ circuits.

3. (Bounded) For every $x \in \{0,1\}^n$, we have

$$\left|\widetilde{f}(x)\right| \leq 2^{O(\log m)^{d-1} \cdot \log(1/\gamma) \cdot \log\log m}. \tag{2.9}$$

The proof of Theorem 2.22 uses *random restrictions and switching lemmas* to analyze the *Fourier spectrum* of $\mathbf{AC}^0$ circuits.[5] We will not study the proof of Theorem 2.22 here. Instead, we will take Theorem 2.22 for granted, and use it to show that $\mathbf{AC}^0$ circuits are sandwiched by low-degree polynomials.

2.6.2 Operations on functions with low-degree sandwichers

We begin with a couple of lemmas about sandwiching by low-degree polynomials.

Lemma 2.23 (If f and g have low-degree sandwichers, then $f+g$ has low-degree sandwichers)**.** Suppose that $f\colon \{0,1\}^n \to \mathbb{R}$ is ε-sandwiched by polynomials of degree k and $g\colon \{0,1\}^n \to \mathbb{R}$ is δ-sandwiched by polynomials of degree k. Then the sum $f+g$ is $(\varepsilon+\delta)$-sandwiched by polynomials of degree k.

We omit the simple proof.

Lemma 2.24 (If f has low-degree sandwichers and g is a bounded low-degree polynomial, then $f \cdot g$ has low-degree sandwichers)**.** Suppose that $f\colon \{0,1\}^n \to \mathbb{R}$ is ε-sandwiched by polynomials of degree k. Let $g\colon \{0,1\}^n \to [-L, L]$, and let $h(x) = f(x) \cdot g(x)$. Then h is $O(\varepsilon \cdot L)$-sandwiched by polynomials of degree $k + \deg(g)$.

Proof. Let f_ℓ, f_u be the sandwichers for f. Suppose first that g is $[0, L]$-valued. In this case, for every $x \in \{0,1\}^n$,

$$f_\ell(x) \cdot g(x) \leq h(x) \leq f_u(x) \cdot g(x),$$

and

$$(f_u(x) - f_\ell(x)) \cdot g(x) \leq L \cdot (f_u(x) - f_\ell(x)),$$

[5] The polynomial $\widetilde{f}$ is defined by dropping all but the lowest-degree Fourier coefficients of f. The bounds Equations (2.8) and (2.9) follow from a bound on the L_2 tail of f [228, Theorem 1] and a bound on the L_1 growth of f [228, Theorem 37].

showing that h is $(\varepsilon \cdot L)$-sandwiched between $f_\ell \cdot g$ and $f_u \cdot g$, each of which has degree $k + \deg(g)$.

Next, suppose that g is $[-L, 0]$-valued. The previous argument shows that $-fg$ is $(\varepsilon \cdot L)$-sandwiched by polynomials of degree $k + \deg(g)$, and therefore so is fg by negating *and swapping* the sandwichers.

Finally, consider the general case that g is $[-L, L]$-valued. Write $g = -L + g'$, where g' is $[0, 2L]$-valued. Then

$$h = -L \cdot f + f \cdot g'.$$

By our previous analyses, $-L \cdot f$ is $(\varepsilon \cdot L)$-sandwiched by polynomials of degree k, and $f \cdot g'$ is $(2\varepsilon \cdot L)$-sandwiched by polynomials of degree $k + \deg(g') = k + \deg(g)$. By Lemma 2.23, it follows that h is $(3\varepsilon \cdot L)$-sandwiched by polynomials of degree $k + \deg(g)$. □

2.6.3 Low-degree sandwichers for $\mathbf{AC}^0$ circuits

We are now prepared to show that $\mathbf{AC}^0$ circuits are sandwiched by low-degree polynomials, and hence they are fooled by k-wise uniform distributions.

Theorem 2.25 ($\mathbf{AC}^0$ circuits are sandwiched by low-degree polynomials)**.** Let $m, d \in \mathbb{N}$ and $\varepsilon > 0$. Every size-m depth-d $\mathbf{AC}^0$ circuit f is ε-sandwiched by polynomials of degree $(\log m)^{O(d^2)} \cdot \log(1/\varepsilon)$.

By the Sandwiching Lemma, Theorem 2.25 implies Braverman's Theorem (Theorem 2.21), albeit with $k = (\log m)^{O(d^2)} \cdot \log(1/\varepsilon)$ instead of $k = (\log m)^{O(d)} \cdot \log(1/\varepsilon)$.

Proof of Theorem 2.25. We will show by induction on d that f has ε-sandwiching polynomials of degree

$$(c \log m)^{(d^2-d)/2} \cdot (\log \log m)^{d-1} \cdot \lceil \log(m/\varepsilon) \rceil$$

for a suitable constant c. First, consider the base case $d = 1$, and assume without loss of generality that f is an AND of m literals.

- If $m < \log(1/\varepsilon)$, then f can be computed exactly by a polynomial of degree m, so we are done.

- If $m \geq \log(1/\varepsilon)$, then our upper sandwicher is the product of the first $\lceil \log(1/\varepsilon) \rceil$ literals and our lower sandwicher is the constant 0 function.

Now, for the inductive step, suppose f has depth $d > 1$. Assume without loss of generality that the top gate of f is OR, say $f = \bigvee_{i=1}^{m} f_i$. We reason similarly to Bazzi [20] and Razborov [200]. For each $i \in [m]$, let $g_i = \bigwedge_{j=1}^{i-1}(\neg f_i)$, so that

$$f = \sum_{i=1}^{m} f_i \cdot g_i.$$

Each function g_i can be computed by a size-m depth-d $\mathbf{AC}^0$ circuit; let $\widetilde{g}_i$ be the corresponding polynomial approximation from Theorem 2.22 with error parameter $\gamma = \varepsilon/(2m^3)$. We define

$$h = \sum_{i=1}^{m} f_i \cdot \widetilde{g}_i$$

$$f_\ell = f - (f-h)^2 \tag{2.10}$$

$$f_u = f + (f-h)^2 \cdot \left(\left(\sum_{i=1}^{m} f_i \right) - f \right). \tag{2.11}$$

The plan is, we will show that f is sandwiched between f_ℓ and f_u, and then we will use our induction hypothesis to show that f_ℓ and f_u are sandwiched by low-degree polynomials. Consequently, f itself is sandwiched by low-degree polynomials.

From the definitions, clearly $f_\ell \leq f \leq f_u$. Furthermore,

$$\begin{aligned}
\mathbb{E}[f_u - f_\ell] &= \mathbb{E}\left[(f-h)^2 \cdot \left(1 - f + \sum_{i=1}^{m} f_i \right) \right] \\
&\leq m \cdot \mathbb{E}[(f-h)^2] \\
&= m \cdot \mathbb{E}\left[\left(\sum_{i=1}^{m} f_i \cdot (g_i - \widetilde{g}_i) \right)^2 \right] \\
&\leq m^2 \cdot \sum_{i=1}^{m} \mathbb{E}[f_i^2 \cdot (g_i - \widetilde{g}_i)^2] \\
&\leq m^3 \cdot \gamma = \varepsilon/2.
\end{aligned}$$

Now we turn to showing that f_ℓ and f_u are themselves sandwiched by low-degree polynomials. For the first step, we claim that

$$f_\ell = 1 - (1-h)^2 \tag{2.12}$$

$$f_u = 1 + (1-h)^2 \cdot \left(\left(\sum_{i=1}^{m} f_i \right) - 1 \right). \tag{2.13}$$

Indeed, when $f(x) = 1$, this is clear, since we have simply substituted 1 for each appearance of f in Equations (2.10) and (2.11). Meanwhile, when $f(x) = 0$, Equations (2.12) and (2.13) still hold, because $f_i(x) = 0$ for every i and therefore $\sum_{i=1}^{m} f_i(x) = 0$ and $h(x) = 0$.

Next, we will plug the definition of h into Equations (2.12) and (2.13) and expand. For convenience, define $f_0 = \widetilde{g}_0 = 1$. That way, we get

$$f_\ell = \sum_{i=0}^{m} \sum_{j=0}^{m} c_{i,j} \cdot f_i \cdot f_j \cdot \widetilde{g}_i \cdot \widetilde{g}_j$$

$$f_u = \sum_{i=0}^{m} \sum_{j=0}^{m} \sum_{k=0}^{m} c_{i,j,k} \cdot f_i \cdot f_j \cdot f_k \cdot \widetilde{g}_j \cdot \widetilde{g}_k,$$

where $|c_{i,j}| \leq 1$ and $|c_{i,j,k}| \leq 1$. For simplicity, let us focus on a single term from the expansion of f_u, namely a term of the form $c_{i,j,k} \cdot f_i \cdot f_j \cdot f_k \cdot \widetilde{g}_j \cdot \widetilde{g}_k$. By Equation (2.9), for every x, we have $|c_{i,j,k} \cdot \widetilde{g}_j(x) \cdot \widetilde{g}_k(x)| \leq L$ where

$$L = 2^{O(\log m)^{d-1} \cdot \log(m/\varepsilon) \cdot \log\log m}.$$

Now, each subcircuit f_i has an AND gate on top, so the product $f_i \cdot f_j \cdot f_k$ can be computed by a size-m depth-$(d-1)$ $\mathbf{AC}^0$ circuit. Therefore, by induction, for every $\delta > 0$, the product $f_i \cdot f_j \cdot f_k$ is δ-sandwiched by polynomials of degree D where

$$D = (c \log m)^{((d-1)^2 - (d-1))/2} \cdot (\log\log m)^{d-2} \cdot \lceil \log(m/\delta) \rceil.$$

We select $\delta = \Theta(\frac{\varepsilon}{Lm^3})$. That way, Lemma 2.24 ensures that the term $c_{i,j,k} \cdot f_i \cdot f_j \cdot f_k \cdot \widetilde{g}_j \cdot \widetilde{g}_k$ is $\left(\frac{\varepsilon}{4(m+1)^3}\right)$-sandwiched by polynomials of degree $D + \deg(\widetilde{g}_j \cdot \widetilde{g}_k)$. Therefore, by Lemma 2.23, f_u as a whole (and similarly f_ℓ as well) is $(\varepsilon/4)$-sandwiched by polynomials of degree $D + \deg(\widetilde{g}_j \cdot \widetilde{g}_k)$.

Consequently, f is ε-sandwiched by polynomials of degree $D+\deg(\widetilde{g}_j \cdot \widetilde{g}_k)$. All that remains is to simplify the degree bound:

$$\begin{aligned} &D + \deg(\widetilde{g}_j \cdot \widetilde{g}_k) \\ &\quad = (c \log m)^{((d-1)^2-(d-1))/2} \cdot (\log \log m)^{d-2} \cdot \lceil \log(m/\delta) \rceil \\ &\qquad\quad + O(\log m)^{d-1} \cdot \log(1/\gamma) \\ &\quad = (c \log m)^{((d-1)^2-(d-1))/2} \cdot O(\log m)^{d-1} \cdot (\log \log m)^{d-1} \cdot \log(m/\varepsilon) \\ &\quad \leq (c \log m)^{(d^2-d)/2} \cdot (\log \log m)^{d-1} \cdot \lceil \log(m/\varepsilon) \rceil, \end{aligned}$$

provided that we choose the constant c large enough. (Note that $\frac{(d-1)^2-(d-1)}{2} + d - 1 = \frac{d^2-d}{2}$.) □

2.6.4 Improved parameters via probabilistic polynomials

So far, we have shown that k-wise uniform generators fool size-m depth-d $\mathbf{AC}^0$ circuits where $k = (\log m)^{O(d^2)} \cdot \log(1/\varepsilon)$. Next, we will show how to improve the exponent from $O(d^2)$ to $O(d)$.[6] The improvement relies on a line of work constructing *probabilistic real polynomials* for $\mathbf{AC}^0$ circuits [23], [36], [114], [230], starting with two independent papers by Beigel *et al.* [23] and Tarui [230] (see also, e.g., work by Razborov [198], Smolensky [223], and Toda and Ogiwara [231]). These works show that for every $\mathbf{AC}^0$ circuit f, there is a distribution F over low-degree polynomials such that for each fixed input $x \in \{0,1\}^n$, with high probability over $\widetilde{f} \sim F$, we have the exact equality $\widetilde{f}(x) = f(x)$. In our setting, we will actually be thinking of the input x as random, which allows us to fix some $\widetilde{f} \in \mathrm{Supp}(F)$ that agrees with f with high probability over the choice of input. Furthermore, even in the "bad case" that $f(x) \neq \widetilde{f}(x)$, the constructions still have some guarantees. The best parameters known are achieved by Harsha and Srinivasan [114], who prove the following:

Theorem 2.26 (BRS-Tarui polynomials [114])**.** Let $f\colon \{0,1\}^n \to \{0,1\}$ be computable by a depth-d size-m $\mathbf{AC}^0$ circuit, let $\delta > 0$, and let D

[6] Note that for certain small values of d such as $d = 3$, the parameters from the first proof are actually superior to the parameters from the second proof. We thank Avishay Tal for pointing this out (personal communication).

be a distribution over $\{0,1\}^n$. There exist a polynomial $\widetilde{f}\colon \{0,1\}^n \to \mathbb{R}$ and an "error function" $E\colon \{0,1\}^n \to \{0,1\}$ such that

- $\mathbb{E}[E(D)] \leq \delta$, and if $E(x) = 0$, then $f(x) = \widetilde{f}(x)$. (Hence, $\Pr_{x\sim D}\left[f(x) \neq \widetilde{f}(x)\right] \leq \delta$.)
- $\deg(\widetilde{f}) \leq (\log m)^{O(d)} \log(1/\delta)$.
- $\|\widetilde{f}\|_\infty \stackrel{\text{def}}{=} \max_x |\widetilde{f}(x)|$ satisfies the bound

$$\|\widetilde{f}\|_\infty \leq \exp\left((\log m)^{O(d)} \log(1/\delta)\right).$$

- E can be computed by an $\mathbf{AC}^0$ circuit of size[7] $m^{O(1)}$ and depth $d + O(1)$.

Note that Theorem 2.26 provides a low-degree approximation over an arbitrary input distribution, unlike LMN polynomials (Theorem 2.22) which are specific to the uniform distribution. The constructions of probabilistic polynomials for $\mathbf{AC}^0$ [23], [36], [114], [230] rely on Valiant and Vazirani's *isolation lemma* [239].

Braverman's original proof that k-wise uniformity fools $\mathbf{AC}^0$ circuits for polylogarithmic k was based on a clever combination of LMN polynomials and BRS-Tarui polynomials. We present (a version of) that proof below to prove Theorem 2.21.

Lemma 2.27 (Sandwichers for $\mathbf{AC}^0$ with better parameters)**.** Let $f\colon \{0,1\}^n \to \{0,1\}$ be computable by a depth-d size-m $\mathbf{AC}^0$ circuit, let $\delta > 0$, and let D be a distribution over $\{0,1\}^n$. There exist polynomials $p_\ell, p_u\colon \{0,1\}^n \to \mathbb{R}$ of degree $(\log m)^{O(d)} \log(1/\delta)$ and a function $E\colon \{0,1\}^n \to \{0,1\}$ such that f is δ-sandwiched between $p_\ell - E$ and $p_u + E$, and furthermore,

$$\mathbb{E}[E(U_n)] \leq \delta \text{ and } \mathbb{E}[E(D)] \leq \delta. \tag{2.14}$$

[7] In Harsha and Srinivasan's work [114], the size bound is stated as $(m \log(1/\delta))^{O(1)}$. We may assume without loss of generality that $\log(1/\delta) < m$, because f can be computed exactly by a degree-m polynomial (namely its Fourier expansion).

To be clear, f is sandwiched between $p_\ell - E$ and $p_u + E$ *with respect to the uniform distribution* (see Definition 2.12). The only part of the conclusion that says something about the arbitrary distribution D is the bound $\mathbb{E}[E(D)] \leq \delta$.

Proof. Let $D' = \frac{1}{2}(D + U_n)$, i.e., D' is D with probability $1/2$ and U_n with probability $1/2$. Let $\widetilde{f}$ be the BRS-Tarui polynomial for f from Theorem 2.26 and let E be the corresponding error function with respect to the distribution D' with error $\mathbb{E}[E(D')] \leq \delta/24$. Note that

$$\mathbb{E}[E(D')] = \frac{1}{2}\,\mathbb{E}[E(D)] + \frac{1}{2}\,\mathbb{E}[E(U_n)] \leq \frac{\delta}{24},$$

so $\mathbb{E}[E(D)] \leq \delta/12$ and $\mathbb{E}[E(U_n)] \leq \delta/12$, proving Equation (2.14).

Recall that the error function E can be computed by an $\mathbf{AC}^0$ circuit of size $m^{O(1)}$ and depth $d + O(1)$ (see Theorem 2.26). Therefore, we may apply Theorem 2.22 to get an LMN polynomial $\widetilde{E}$ that satisfies $\|E - \widetilde{E}\|_2^2 \leq \gamma$ for an error parameter γ that will be specified later. Define three more approximations to f by the formulas

$$\begin{aligned} \phi &= 1 - (1 - f) \cdot (1 - E) = f \vee E \\ \widetilde{\phi} &= 1 - (1 - \widetilde{f}) \cdot (1 - \widetilde{E}) \\ p_u &= \left(1 - (1 - \widetilde{f}) \cdot (1 - \widetilde{E})\right)^2 = \left(\widetilde{\phi}\right)^2. \end{aligned} \tag{2.15}$$

We must show that $p_u + E$ is an upper sandwicher for f. By a case analysis, let us prove that the following two bounds hold (pointwise):

$$f \leq p_u + E \tag{2.16}$$

$$p_u \leq f + 2E + 2 \cdot \left(\widetilde{\phi} - \phi\right)^2. \tag{2.17}$$

- (Case 1) Suppose $f(x) > E(x)$, i.e., $E(x) = 0$ and $f(x) = 1$. Then $\widetilde{f}(x) = f(x) = 1$, so $p_u(x) = 1$ as well, which implies Equations (2.16) and (2.17).
- (Case 2) Suppose $f(x) \leq E(x)$, i.e., $E(x) = 1$ or $f(x) = 0$. Then Equation (2.16) holds because p_u is non-negative. Furthermore, $\phi(x) = E(x)$ in this case, so

  $$p_u(x) = \left(E(x) + \widetilde{\phi}(x) - \phi(x)\right)^2 \leq 2E(x)^2 + 2 \cdot \left(\widetilde{\phi}(x) - \phi(x)\right)^2,$$

  which implies Equation (2.17) because $E(x)^2 = E(x)$.

Now, because of the factor of $1 - E$ in Equation (2.15), we have the identity $\phi = 1 - (1 - \widetilde{f}) \cdot (1 - E)$, so Equation (2.17) becomes

$$p_u \leq f + 2E + 2 \cdot \left((1 - \widetilde{f}) \cdot (E - \widetilde{E})\right)^2.$$

Therefore,

$$\begin{aligned}\mathbb{E}[p_u + E - f] &\leq 3\,\mathbb{E}[E] + 2\,\mathbb{E}\left[\left((1 - \widetilde{f}) \cdot (E - \widetilde{E})\right)^2\right] \\ &\leq \delta/4 + 2 \cdot (1 + \|\widetilde{f}\|_\infty)^2 \cdot \|E - \widetilde{E}\|_2^2 \\ &\leq \delta/4 + 2 \cdot (1 + \|\widetilde{f}\|_\infty)^2 \cdot \gamma.\end{aligned}$$

By choosing $\gamma = \delta/(8 \cdot (1 + \|\widetilde{f}\|_\infty)^2)$, we get $\mathbb{E}[p_u + E - f] \leq \delta/2$.

Next, let us bound the degree of p_u. Recall that Theorems 2.22 and 2.26 give the bounds

$$\begin{aligned}\deg(\widetilde{f}) &\leq (\log m)^{O(d)} \log(1/\delta) \\ \|\widetilde{f}\|_\infty &\leq \exp\left((\log m)^{O(d)} \log(1/\delta)\right) \\ \deg(\widetilde{E}) &\leq O(\log m)^{d+O(1)} \log(1/\gamma) = (\log m)^{O(d)} \log(1/\delta).\end{aligned}$$

Therefore, $\deg(p_u) \leq 2\deg(\widetilde{f}) + 2\deg(\widetilde{E}) \leq (\log m)^{O(d)} \log(1/\delta)$.

To summarize, we have shown that every size-m depth-d $\mathbf{AC}^0$ circuit f can be upper-sandwiched by $p_u + E$ where p_u is a low-degree polynomial and E is a Boolean function with low expectation under both U_n and D. The class of size-m depth-d $\mathbf{AC}^0$ circuits is closed under complementation, so $1 - f$ has an upper sandwicher of the same form. Therefore, f can be *lower*-sandwiched by $p_\ell - E'$ where p_ℓ is a low-degree polynomial and E' is a Boolean function with low expectation under both U_n and D. If we use $1 - \widetilde{f}$ as our BRS-Tarui polynomial for $1 - f$ in the above argument, then we can furthermore ensure $E' \equiv E$. □

Braverman's theorem follows readily from Lemma 2.27:

Proof of Theorem 2.21. Let D be a k-wise independent distribution, where k is the bound on the degrees of the low-degree polynomials p_u and p_ℓ from Lemma 2.27 with $\delta = \varepsilon/2$. By Lemma 2.27, the circuit f is $(\varepsilon/2)$-sandwiched between $p_\ell - E$ and $p_u + E$, where E is $(\varepsilon/2)$-fooled by D. Since p_u and p_ℓ are degree-k polynomials, they are perfectly fooled by D. Therefore, by the sandwiching lemma, f is ε-fooled by D. □

Intriguingly, although the proof of Theorem 2.21 is a sandwiching argument, the sandwichers are apparently *not* low-degree polynomials. They are of the form $p_\ell - E$ and $p_u + E$, where p_ℓ and p_u are low-degree polynomials, but the "error function" E does not seem to be a low-degree polynomial. (Furthermore, the sandwichers depend on the pseudorandom distribution D.)

In a formal sense, sandwiching polynomials are the only tool one ever needs to prove that k-wise uniform generators fool some class of functions (see Appendix A). However, the proof of Braverman's theorem demonstrates that in practice, it is wise to "think outside the box" and consider other, more creative arguments. The technique of designing low-complexity error indicator functions, along the lines of Theorem 2.26, has turned out to be useful in other PRG problems [81], [120], [171].

It is an open problem to improve the parameters of Braverman's theorem even further. What is the optimal k such that every k-wise uniform generator fools depth-d size-m $\mathbf{AC}^0$ circuits with error ε? There are counterexamples showing that $k = \Omega((\log m)^{d-1} \log(1/\varepsilon))$ [163], but that still leaves a significant gap between the lower and upper bounds.

Open Problem 2.4 (Improved parameters for Braverman's theorem)**.** Show that for every $m, d \in \mathbb{N}$ and $\varepsilon > 0$, there exists a value

$$k = (\log m)^{d+O(1)} \log(1/\varepsilon)$$

such that every k-wise uniform distribution fools depth-d size-m $\mathbf{AC}^0$ circuits with error ε.

Explicit PRGs for $\mathbf{AC}^0$ circuits with seed length $(\log m)^{d+O(1)} \log(1/\varepsilon)$ are already known (see Section 5.3); the question is whether a generic k-wise uniform generator does the job.

3

Recycling Random Bits

In this section, we will present a few PRG constructions based on the paradigm of *recycling* random bits. In its simplest form, the idea is that we start by sampling $n/2$ *truly* random bits X. Then we "mix in" a few more truly random bits in some way, producing $n/2$ additional bits Y. Our final output is the concatenation (X, Y).

This type of approach tends to make the most sense if there is a "communication bottleneck" between the part of the computation that processes X and the part of the computation that processes Y. Indeed, our first instantiation of this paradigm will be a PRG for two-party communication protocols in the next section.

3.1 PRGs for Two-party Communication Protocols

In this section, we will present a PRG that fools two-party interactive communication protocols.

Definition 3.1 (Two-party communication protocol)**.** Let n be an even positive integer. In a *two-party protocol on n bits with communication cost m*, Alice holds $x \in \{0,1\}^{n/2}$ and Bob holds $y \in \{0,1\}^{n/2}$. They communicate interactively; in each round, the identity of the speaker

is a function of all the bits transmitted so far, and the content of the message is a function of the bits transmitted and that party's input (x or y). After at most m bits have been transmitted in total, the protocol terminates, and both parties output the same bit $f(x, y)$.

Are these protocols deterministic, or are they randomized? Both, in a sense: the protocol is deterministic after fixing x and y, but we are thinking of x and y as random bits and seeking to replace them with pseudorandom bits. The seed length in the following theorem, due to Impagliazzo *et al.* [131], is optimal.

Theorem 3.1 ([131]). For every $n, m \in \mathbb{N}$ and every $\varepsilon > 0$, there is an explicit ε-PRG for two-party protocols on n bits with communication cost m with seed length $\frac{n}{2} + O(m + \log(1/\varepsilon))$.

It is worth noting that the above theorem depends only on the total cost of communication and holds regardless of the number or structure of the rounds of communication in the protocol.

We will prove Theorem 3.1 by using one of the most useful facts about communication protocols, which is that the leaves of a protocol correspond to two-dimensional *combinatorial rectangles*. This will allow us to reduce the task of fooling communication protocols to fooling combinatorial rectangles, and we will achieve the latter task via known constructions of *expander graphs*. We begin by discussing expander graphs.

3.1.1 Expander graphs from a PRG perspective

Doing justice to the topic of expanders is beyond the scope of this work. For a thorough treatment, see, e.g., Hoory *et al.*'s survey [123], Goldreich's survey [98], Lubotzky's survey [162], or Vadhan's monograph [238]. That being said, let us at least review the definition.

Definition 3.2 (Spectral expanders). Let G be a regular undirected graph on N vertices with transition probability matrix $M \in [0,1]^{N \times N}$. Let the eigenvalues of M be $1 = \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N$. We say G is an *ε-spectral expander* if $|\lambda_i| \leq \varepsilon$ for $i = 2, 3, \ldots, N$.

As mentioned earlier, the reason we are discussing expanders is so we can use them to construct PRGs. Looking at Definition 3.2, it is not necessarily obvious that there is any connection between expanders and PRGs. However, it turns out that there is an alternative and equivalent definition of expanders in the language of PRGs. We focus on the special case that the number of vertices is a power of two, because that's the most natural scenario from the PRG perspective, but the following lemma generalizes in the natural way to an arbitrary number of vertices.

Lemma 3.2 (PRG characterization of spectral expander graphs)**.** Let n be an even positive integer, and let $\mathcal{F}$ be the class of functions $f\colon \{0,1\}^n \to \mathbb{R}$ of the form $f(x,y) = g(x) \cdot h(y)$ where $g, h\colon \{0,1\}^{n/2} \to \mathbb{R}$ satisfy $\mathrm{Var}[g] \leq 1$ and $\mathrm{Var}[h] \leq 1$. Let G be a regular undirected graph on the vertex set $\{0,1\}^{n/2}$, sample a uniform random vertex X in G, and sample a uniform random neighbor Y of X. For every $\varepsilon > 0$, the following are equivalent.

1. G is an ε-spectral expander (see Definition 3.2).
2. The distribution (X,Y) fools $\mathcal{F}$ with error ε.

Proof. Let M be the transition probability matrix of G. We will identify functions mapping $\{0,1\}^{n/2} \to \mathbb{R}$ with column vectors in the space $\mathbb{R}^{2^{n/2}}$ endowed with the inner product $\langle g, h\rangle = \mathbb{E}_{U \sim U_{n/2}}[g(U) \cdot h(U)]$ and the norm $\|g\| = \sqrt{\langle g, g\rangle}$.

(1 $\implies$ 2) Fix an arbitrary $f \in \mathcal{F}$, say $f(x,y) = g(x) \cdot h(y)$ where $\mathrm{Var}[g] \leq 1$ and $\mathrm{Var}[h] \leq 1$. First, suppose $\mathbb{E}[g] = \mathbb{E}[h] = 0$. Then

$$|\,\mathbb{E}[f(X,Y)]| = |\langle g, Mh\rangle| \leq \|g\| \cdot \|Mh\|$$

by the Cauchy-Schwarz inequality. Since G is regular, the all-ones vector is an eigenvector of M with eigenvalue $\lambda_1 = 1$. Since $\mathbb{E}[h] = 0$, the vector h is orthogonal to the all-ones vector. By the spectral theorem for real symmetric matrices, it follows that h is a linear combination of eigenvectors *other* than the all-ones vector, so by Definition 3.2, $\|Mh\| \leq \varepsilon \cdot \|h\|$. Therefore,

$$|\,\mathbb{E}[f(X,Y)]| \leq \varepsilon \cdot \|g\| \cdot \|h\| = \varepsilon \cdot \sqrt{\mathrm{Var}[g]} \cdot \sqrt{\mathrm{Var}[h]} \leq \varepsilon.$$

Now for the general case, write $g = \mathbb{E}[g] + \overline{g}$ and $h = \mathbb{E}[h] + \overline{h}$, so

$$f(x,y) = \mathbb{E}[g] \cdot \mathbb{E}[h] + \overline{g}(x) \cdot \overline{h}(y) + \mathbb{E}[g] \cdot \overline{h}(y) + \mathbb{E}[h] \cdot \overline{g}(x).$$

Since G is regular, each marginal distribution X and Y is uniform over $\{0,1\}^{n/2}$. Therefore, $\mathbb{E}[\overline{g}(X)] = \mathbb{E}[\overline{h}(Y)] = 0$. Furthermore, $\mathrm{Var}[\overline{g}] = \mathrm{Var}[g] \leq 1$ and $\mathrm{Var}[\overline{h}] = \mathrm{Var}[h] \leq 1$, so

$$|\,\mathbb{E}[f(X,Y)] - \mathbb{E}[f]| = |\,\mathbb{E}[\overline{g}(X) \cdot \overline{h}(Y)]| \leq \varepsilon.$$

($2 \implies 1$) Let g be a unit eigenvector of M with eigenvalue λ and assume that g is not parallel to the all-ones vector. By the spectral theorem for symmetric matrices, it follows that g is orthogonal to the all-ones vector, i.e., $\mathbb{E}[g] = 0$. Since g has unit norm as a vector, we have $\mathrm{Var}[g] = 1$. Therefore,

$$|\lambda| = |\langle Mg, g\rangle| = |\,\mathbb{E}[g(X) \cdot g(Y)]| \leq \varepsilon. \qquad \square$$

The seed length required to sample the distribution (X,Y) in Lemma 3.2 is governed by the *degree* of the expander graph. There exist explicit expanders with degree $\mathrm{poly}(1/\varepsilon)$:

Theorem 3.3 (Explicit expanders)**.** For every $n \in \mathbb{N}$ and every $\varepsilon > 0$, there exists an ε-spectral expander with vertex set $\{0,1\}^n$ and degree $D = \mathrm{poly}(1/\varepsilon)$. The expander is "strongly explicit," i.e., given a vertex x and a value $i \in [D]$, the i^{th} neighbor of x can be computed in time $\mathrm{poly}(n, \log(1/\varepsilon))$.

To learn about various approaches for proving Theorem 3.3, see the expository works on expander graphs mentioned previously [98], [123], [162], [238]. In this text, we will simply take Theorem 3.3 for granted.

Theorem 3.3 translates to a seed length of $n/2 + O(\log(1/\varepsilon))$ in Lemma 3.2. Pushing to the extreme, a "Ramanujan graph" is an ε-spectral expander of degree D where $\varepsilon = 2\sqrt{D-1}$, which is essentially the best possible [6], [92], [179] and which translates to a seed length of $n/2 + 2\log(1/\varepsilon) + O(1)$. There is a lot of work on the problems of proving existence of and explicitly constructing Ramanujan graphs and "near-Ramanujan" graphs [7], [25], [34], [67], [73], [93], [161], [167], [168], [176], [177].

Lemma 3.2 analyzes the distribution (X, Y), where X is a uniform random vertex in an expander and Y is a uniform random neighbor of X. More generally, we can consider taking a *random walk* through an expander graph, starting from a uniform random vertex. The pseudorandomness properties of such walks have been intensively studied in many works, including many recent works [10], [64], [71], [72], [102], [103], [108], [217]. For an introduction to these topics, see the expository works mentioned previously [98], [123], [162], [238].

3.1.2 Combinatorial rectangles and the Expander Mixing Lemma

One of the most famous facts about expander graphs is the Expander Mixing Lemma. In the language of PRGs, the Expander Mixing Lemma is the "1 $\implies$ 2" direction of Lemma 3.2, specialized to the case that g and h are Boolean-valued. That is, the Expander Mixing Lemma explains how to use expander graphs to fool *two-dimensional combinatorial rectangles*, defined next.

Definition 3.3 (Combinatorial rectangles)**.** Let n be a multiple of d and let $f\colon \{0,1\}^n \to \{0,1\}$. We say that f is a *d-dimensional combinatorial rectangle* if there are functions $f_1, \dots, f_d\colon \{0,1\}^{n/d} \to \{0,1\}$ such that

$$f(x^{(1)}, \dots, x^{(d)}) = \prod_{i=1}^{d} f_i(x^{(i)}).$$

Lemma 3.4 (Expander Mixing Lemma)**.** Let n be an even positive integer and let G be an ε-spectral expander on vertex set $\{0,1\}^{n/2}$. Sample a uniform random vertex X, then sample a uniform random neighbor Y of X. Then the distribution (X, Y) fools two-dimensional combinatorial rectangles with error $\varepsilon/4$.

The proof uses the following elementary bound.

Fact 3.1 (Popoviciu's inequality on variances)**.** Let $a < b$ be real numbers and let X be an $[a, b]$-valued random variable. Then $\mathrm{Var}[X] \le (b-a)^2/4$.

Proof of Lemma 3.4. Let f be a two-dimensional combinatorial rectangle, say $f(x, y) = g(x) \cdot h(y)$. We apply Lemma 3.2 to the functions $g/\sqrt{\mathrm{Var}[g]}$ and $h/\sqrt{\mathrm{Var}[h]}$ (each of which has variance 1). This shows

that (X, Y) fools f with error $\varepsilon \cdot \sqrt{\mathrm{Var}[g]} \cdot \sqrt{\mathrm{Var}[h]}$. Finally, by Fact 3.1, we have $\sqrt{\mathrm{Var}[g]} \cdot \sqrt{\mathrm{Var}[h]} \leq \sqrt{1/4} \cdot \sqrt{1/4} = 1/4$. □

Corollary 3.5 (Optimal PRG for two-dimensional combinatorial rectangles)**.** For every $n \in \mathbb{N}$ and every $\varepsilon > 0$, there is an explicit ε-PRG for two-dimensional combinatorial rectangles on n bits with seed length $\frac{n}{2} + O(\log(1/\varepsilon))$.

Proof. Immediate from Lemma 3.4 and Theorem 3.3. □

There is a large body of work designing PRGs for high-dimensional combinatorial rectangles [14], [86], [105], [106], [113], [148], [154], [159], [243]. Near-optimal constructions are known [105], [106], [148], but it is still an open problem to get the optimal seed length.

Open Problem 3.1 (Optimal PRGs for high-dimensional combinatorial rectangles)**.** Design an explicit PRG for d-dimensional combinatorial rectangles with seed length $O(n/d + \log(1/\varepsilon) + \log\log n)$.

For this section, however, it suffices to focus on the two-dimensional case. A two-dimensional combinatorial rectangle f can be computed by a communication protocol with a particularly simple structure: Alice computes $g(x)$ and sends it to Bob, and then Bob computes $h(x)$ and multiplies. Given Corollary 3.5, we are now ready to fool *general* two-party communication protocols by using the standard decomposition of communication protocols into combinatorial rectangles.

Proof of Theorem 3.1. Without loss of generality, we may assume that the parties always transmit exactly m bits in total. Define $\pi\colon \{0,1\}^n \to \{0,1\}^m$ by letting $\pi(x, y)$ be the sequence of bits transmitted when Alice holds x and Bob holds y. For each possible transcript $z \in \{0,1\}^m$, define functions $g_z, h_z\colon \{0,1\}^{n/2} \to \{0,1\}$ by the rule

$$g_z(x) = 1 \iff \exists y, \pi(x, y) = z$$
$$h_z(y) = 1 \iff \exists x, \pi(x, y) = z.$$

Clearly if $\pi(x, y) = z$, then $g_z(x) = h_z(y) = 1$. Conversely, we claim that if $g_z(x) = h_z(y) = 1$, then $\pi(x, y) = z$. To see why, assume by induction that the first i bits of $\pi(x, y)$ agree with z, and without loss

of generality assume that Alice is the next speaker after those i bits have been transmitted. Since $g_z(x) = 1$, there is some y' such that $\pi(x, y') = z$. Alice chooses which bit to speak based only on her private input (x) and the partial transcript so far (which agrees with z). All of this data is the same regardless of whether Bob holds y or y'. Therefore, since Alice would transmit z_{i+1} if Bob held y' ($\pi(x, y') = z$), she must transmit z_{i+1} when Bob holds y.

Since both parties output $f(x, y)$, there is some set $A \subseteq \{0,1\}^m$ such that $f(x, y) = 1 \iff \pi(x, y) \in A$. Therefore,

$$f(x, y) = \sum_{z \in A} g_z(x) \cdot h_z(y),$$

a linear combination of two-dimensional combinatorial rectangles. Therefore, by the Triangle Inequality for PRG Errors, every δ-PRG for two-dimensional combinatorial rectangles fools f with error $|A|\delta \leq 2^m\delta$. Picking $\delta = 2^{-m}\varepsilon$ and applying Corollary 3.5 completes the proof. □

3.2 The INW Generator for Standard-order ROBPs

Recall that in Section 1.5, we defined the *standard-order ROBP* model (Definition 1.5). We showed that PRGs for standard-order ROBPs can be used to simulate randomized space-bounded computation. In this section, we will present the Impagliazzo-Nisan-Wigderson (INW) generator [131], which is one of the most influential unconditional PRG constructions, and we will prove that it fools standard-order ROBPs, albeit with a non-optimal seed length.

The INW PRG is based on recycling seeds *recursively*. After constructing a PRG with output length $n/2$, we will use two correlated seeds to generate two pseudorandom strings of length $n/2$ and concatenate them to get a string of length n. To argue that the INW PRG works, first we argue that two independent seeds would work well, and then we argue that two correlated seeds is almost as good as two independent seeds.

3.2.1 Concatenating two independent pseudorandom strings

In this section, we show that the concatenation of two independent pseudorandom strings is itself pseudorandom. This argument is generic and applies to many different models of computation, not just ROBPs.

Lemma 3.6 (Concatenating independent pseudorandom strings)**.** Let n be an even positive integer, let $f\colon \{0,1\}^n \to \mathbb{R}$, and let X_L, X_R be independent random variables distributed over $\{0,1\}^{n/2}$. Let $\mathcal{F}$ be the class of all restrictions of f to $n/2$ bits. If X_L and X_R fool $\mathcal{F}$ with error ε, then (X_L, X_R) fools f with error 2ε.

Proof. Sample $U \sim U_{n/2}$ so that X_L, X_R, U are mutually independent. Then

$$\begin{aligned}\mathop{\mathbb{E}}_{X_L,X_R}[f(X_L,X_R)] = \mathop{\mathbb{E}}_{X_L}\left[\mathop{\mathbb{E}}_{X_R}[f(X_L,X_R)]\right] &= \mathop{\mathbb{E}}_{X_L}\left[\mathop{\mathbb{E}}_{U}[f(X_L,U)] \pm \varepsilon\right] \\ &= \mathop{\mathbb{E}}_{U}\left[\mathop{\mathbb{E}}_{X_L}[f(X_L,U)]\right] \pm \varepsilon \\ &= \mathbb{E}[f] \pm 2\varepsilon. \qquad \square\end{aligned}$$

3.2.2 Recycling seeds using a PRG for two-dimensional rectangles

The main lemma of the INW generator allows us to double the output length of a PRG by paying a small *additive* penalty in terms of the seed length.

Lemma 3.7 (Recycling seeds for standard-order ROBPs)**.** Let n be an even positive integer. Suppose $G\colon \{0,1\}^s \to \{0,1\}^{n/2}$ is an ε-PRG for width-w length-$(n/2)$ standard-order ROBPs. Let (Y_L, Y_R) be a distribution over $\{0,1\}^{2s}$ that δ-fools two-dimensional combinatorial rectangles. Then $(G(Y_L), G(Y_R))$ fools width-w length-n standard-order ROBPs with error $2\varepsilon + w\delta$.

To prove Lemma 3.7, we introduce some convenient standard notation for *subprograms* of ROBPs.

Definition 3.4 (Subprograms of ROBPs)**.** Let f be a standard-order ROBP with layers $V_0, \ldots, V_n$. Let $u \in V_i$ and $S \subseteq V_j$ where $i \le j$. We let $f_{u \to S}$ denote the subprogram from u to S, i.e., the standard-order ROBP

on layers $V_i, \ldots, V_j$ with start vertex u and accept vertices S. We use the shorthand $f_{u\to v} = f_{u\to\{v\}}$, $f_{\to v} = f_{v_{\text{start}}\to v}$, and $f_{u\to} = f_{u\to V_{\text{accept}}}$.

Proof of Lemma 3.7. Let f be a width-w length-n standard-order ROBP. Define $g\colon \{0,1\}^{2s} \to \{0,1\}$ by composing f with two independent copies of G, i.e.,

$$g(y_L, y_R) = f(G(y_L), G(y_R)).$$

By Lemma 3.6, using two independent seeds fools f with error 2ε, i.e., $|\mathbb{E}[f] - \mathbb{E}[g]| \le 2\varepsilon$. Now let us compare two independent seeds to the two correlated seeds (Y_L, Y_R). For any $x_L, x_R \in \{0,1\}^{n/2}$, we can write

$$f(x_L, x_R) = \sum_{v\in V_{n/2}} f_{\to v}(x_L) \cdot f_{v\to}(x_R).$$

Consequently, if we define $g_v(y_L, y_R) = f_{\to v}(G(y_L)) \cdot f_{v\to}(G(y_R))$, then

$$g(y_L, y_R) = \sum_{v\in V_{n/2}} g_v(y_L, y_R).$$

Each function g_v is a two-dimensional combinatorial rectangle on $2s$ bits. Therefore, (Y_L, Y_R) fools g_v with error δ, so by the Triangle Inequality for PRG Errors, (Y_L, Y_R) fools g with error $w\delta$. Therefore,

$$|\mathbb{E}[f] - \mathbb{E}[g(Y_L, Y_R)]| \le 2\varepsilon + w\delta. \qquad \square$$

Remark 3.1 (ROBPs and communication protocols)**.** The proof of Lemma 3.7 can be understood as a special case of our analysis of communication protocols in Section 3.1. If Alice is given $y_L \in \{0,1\}^s$ and Bob is given $y_R \in \{0,1\}^s$, then they can compute $g(y_L, y_R) := f(G(y_L), G(y_R))$ using $\log w$ bits of communication. This implies that (Y_L, Y_R) fools f by the same argument we used to prove Theorem 3.1.

Theorem 3.8 (The INW generator for standard-order ROBPs [131])**.** For every $n, w \in \mathbb{N}$ and every $\varepsilon > 0$, there is an explicit ε-PRG for width-w length-n standard-order ROBPs with seed length $O(\log(wn/\varepsilon) \cdot \log n)$.

Proof. Assume for simplicity that n is a power of two. Let $\delta = \frac{\varepsilon}{wn}$. We inductively construct a sequence of PRGs $G_i\colon \{0,1\}^{s_i} \to \{0,1\}^{2^i}$ for $i = 0, 1, \ldots, \log n$. We start with the trivial PRG $G(y) = y$ where $|y| = 1$.

Now let $i > 0$ and suppose we have constructed $G_1, \ldots, G_{i-1}$. The PRG G_i uses its seed to sample $Y_L, Y_R \in \{0,1\}^{s_{i-1}}$ such that (Y_L, Y_R) fools two-dimensional combinatorial rectangles with error δ. Then G_i outputs $(G_{i-1}(Y_L), G_{i-1}(Y_R))$.

By Lemma 3.7, G_i fools width-w standard-order ROBPs with error ε_i for every i, where $\varepsilon_0 = 0$ and $\varepsilon_i = 2\varepsilon_{i-1} + \frac{\varepsilon}{n}$ for $i > 0$. This implies that $\varepsilon_i = (2^i - 1) \cdot \frac{\varepsilon}{n}$ for every i, and hence $\varepsilon_{\log n} < \varepsilon$. Meanwhile, by Corollary 3.5, we have $s_i = s_{i-1} + O(\log(1/\delta))$. Therefore, $s_{\log n} = O(\log(1/\delta) \cdot \log n) = O(\log(wn/\varepsilon) \cdot \log n)$. □

The seed length in Theorem 3.8 was actually already achieved by Nisan [181] prior to Impagliazzo *et al.*'s work [131]. Nisan's PRG [181] follows a fairly similar intuition as the INW generator, but the details are different. The INW generator has some advantages over Nisan's generator; most importantly, the INW generator has turned out to be more flexible and amenable to analysis in other models. (We will see an example in Section 3.3.)

The optimal seed length for fooling width-w length-n standard-order ROBPs would be $O(\log(wn/\varepsilon))$. Designing optimal or near-optimal PRGs for standard-order ROBPs is one of the biggest open problems in the unconditional theory of PRGs. Some of the work on this problem focuses on the case that the width of the program is very small. In Section 2.3.4, we saw that small-bias generators fool width-2 branching programs with seed length $O(\log(n/\varepsilon))$. Explicit PRGs for width-3 standard-order ROBPs are known with seed length $\widetilde{O}(\log n \cdot \log(1/\varepsilon))$ [171]. However, for width-4 ROBPs, no PRG constructions are known with a seed length better than that of the INW generator.

Open Problem 3.2 (Better PRGs for width-4 ROBPs)**.** Design an explicit 0.1-PRG for width-4 length-n standard-order ROBPs with seed length $o(\log^2 n)$.

A candidate PRG for ROBPs, suggested by Reingold and Vadhan [151], [172], is to take a sum (i.e., bitwise XOR) of a few independent small-bias distributions. Recall Viola's proof that a sum of d small-bias distributions fools degree-d polynomials over $\mathbb{F}_2$ (see Section 2.4). Perhaps a similar PRG can work for constant-width ROBPs.

Open Problem 3.3 (ROBPs and sums of small-bias distributions)**.** Prove or disprove the suggestion that a sum of $O(1)$ small-bias distributions fools constant-width standard-order ROBPs with near-logarithmic seed length.

Amazingly, it is consistent with current knowledge that simply summing two small-bias distributions fools *polynomial*-width standard-order ROBPs with optimal seed length $O(\log(n/\varepsilon))$.

Due to the difficulty of constructing improved PRGs, much of the recent research on pseudorandomness for ROBPs has focused on constructing relaxations of PRGs such as HSGs and WPRGs (defined in Section 1.6) [37], [52]–[54], [70], [105], [124], [128], [193], [220]. Another line of work seeks PRGs for *restricted classes* of ROBPs; we will discuss an example in the next section.

3.3 The BRRY Generator for Standard-order Regular ROBPs

As discussed in the previous section, it is still an open problem to design an explicit PRG for constant-width standard-order ROBPs with seed length $o(\log^2 n)$. However, we can get a better seed length for many interesting special cases, including *regular* programs. A standard-order ROBP is called *regular* if every vertex of the program (except those in the first layer) has in-degree 2; see Figure 3.1.

3.3.1 Reduction from the general case to the regular case

Before presenting the PRG for regular programs, let us discuss the motivation behind trying to fool this particular class of branching programs. One compelling reason to study regular programs is that there is a reduction from the general case to the regular case. We will prove the following version of this reduction, which is a special case of a recent result by Lee *et al.* [150] and an improvement over prior reductions [30], [203].

Proposition 3.1 (Regular programs can simulate non-regular programs)**.** Let $n, w \in \mathbb{N}$ and let $f\colon \{0,1\}^n \to \{0,1\}$. If f can be computed by a standard-order ROBP of width w, then f can be computed by a standard-order regular ROBP of width $O(wn)$.

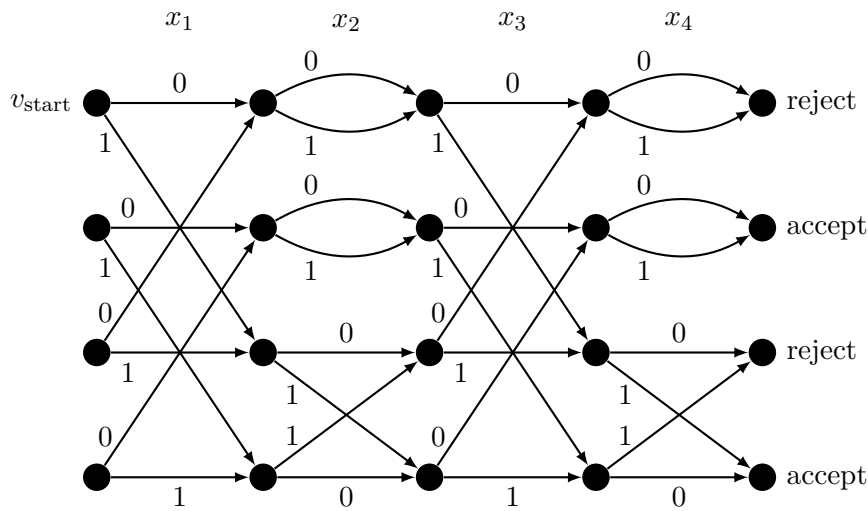

Figure 3.1: Let n be an even positive integer. The function $f(x) = x_1x_2 \oplus x_3x_4 \oplus \cdots \oplus x_{n-1}x_n$ can be computed by a width-4 standard-order regular ROBP. The case $n = 4$ is shown above.

Proof. We will make the layers regular one at a time, starting at the beginning of the program and finishing at the end. To be precise, we will show by induction on $i \in \{0, \ldots, n\}$ that there exists a standard-order ROBP g^i with layers $V_0^i, \ldots, V_n^i$ such that the following hold.

1. The program g^i computes the function f.
2. Every vertex in $V_1^i \cup \cdots \cup V_i^i$ has precisely two incoming edges.
3. We have $|V_0^i| = |V_1^i| = \cdots = |V_i^i| \leq w \cdot (i+1)$.
4. We have $|V_{i+1}^i| = |V_{i+2}^i| = \cdots = |V_n^i| = w$.

In the base case $i = 0$, this is trivial, because we can take g^0 to be the width-w standard-order ROBP computing f. Now, for the inductive step, let $i \in [n]$, assume that we have constructed g^{i-1}, and let us modify it to construct g^i.

For each vertex $v \in V_i^{i-1}$, let e_v be the number of edges leading into v. To construct g^i, we replace v with a collection S_v of $\lceil e_v/2 \rceil$ vertices. We distribute the incoming edges of v among the vertices in S_v in such

a way that each has at most two incoming edges, and we duplicate the outgoing edges of v at each vertex in S_v.

Due to rounding, each set S_v might have one vertex that only has one incoming edge at this point. To deal with these "leftover vertices," observe that the total number of edges leading into V_i^{i-1} (i.e., $\sum_v e_v$) is even, so the total number of leftover vertices must also be even. We can therefore introduce dummy vertices in $V_0^i, V_1^i, \ldots, V_{i-1}^i$ with dummy outgoing edges in such a way that every vertex in $V_1^i \cup \cdots \cup V_i^i$ has exactly two incoming edges.

By this construction, we have

$$\begin{aligned} |V_i^i| = \sum_{v \in V_i^{i-1}} \lceil e_v/2 \rceil &\leq \sum_{v \in V_i^{i-1}} (e_v/2 + 1) \\ &= |V_i^{i-1}| + |V_{i-1}^{i-1}| \\ &\leq w + w \cdot i \\ &= w \cdot (i+1). \end{aligned}$$

Furthermore, the fact that every vertex in $V_1^i \cup \cdots \cup V_i^i$ has exactly two incoming edges implies that $|V_0^i| = \cdots = |V_i^i|$. □

3.3.2 Improved analysis of the INW PRG for low-weight programs

Having demonstrated the importance of standard-order regular ROBPs, we now present the following result by Braverman *et al.* [38], which shows how to fool this model with a seed length of $\widetilde{O}(\log n)$ in the constant-width regime.

Theorem 3.9 (The BRRY generator for standard-order regular ROBPs [38])**.** For every $w, n \in \mathbb{N}$ and $\varepsilon > 0$, there is an explicit ε-PRG for width-w length-n standard-order regular ROBPs with seed length

$$O\left(\log n \cdot (\log w + \log\log n + \log(1/\varepsilon))\right).$$

As we will see, the PRG *construction* is actually the same INW generator that we discussed in Section 3.2 (albeit with different parameters). The improvement comes from a better analysis.

For intuition, let us briefly summarize the INW analysis that we seek to improve (see Section 3.2). In each round of the INW construction,

we use a δ-spectral expander to recycle the seed from the previous round. The analysis essentially argues that we pay an error δ at each vertex in the program, so the total error is δwn. We therefore chose $\delta = \frac{\varepsilon}{wn}$. In each of the $\log n$ rounds of the construction, the seed length increases by an additive $O(\log(1/\delta))$, leading to the final seed length of $O(\log(wn/\varepsilon) \cdot \log n)$.

Braverman *et al.* [38] had the insight that this analysis is overly pessimistic in some cases, because it treats every vertex in the program as "important." In reality, sometimes a vertex v is "unimportant," in the sense that its two out-neighbors $v[0], v[1]$ have almost the same acceptance probabilities: $\mathbb{E}[f_{v[0]\to}] \approx \mathbb{E}[f_{v[1]\to}]$. After reaching such a v, it doesn't matter much whether we read a high-quality random bit or a low-quality random bit, because it doesn't matter much whether we go to $v[0]$ or $v[1]$. Rather than contributing a penalty of δ to the overall error bound, intuitively we might hope that the error at such a vertex is closer to $\delta \cdot |\mathbb{E}[f_{v[0]\to}] - \mathbb{E}[f_{v[1]\to}]|$.

To formalize this intuition, we rely on a generalization of ROBPs that output real values instead of Boolean ones, called *read-once evaluation programs* (ROEPs).

Definition 3.5 (Read-once evaluation programs)**.** A length-n standard-order read once evaluation program (standard-order ROEP) f has the same graph structure as a length-n standard-order ROBP, but in the final layer, instead of a set of accept vertices $V_{\text{accept}} \subseteq V_n$, it has a value $q_v \in \mathbb{R}$ assigned to each vertex $v \in V_n$. Each input $x \in \{0,1\}^n$ defines a path from v_{start} to a vertex $v \in V_n$ as usual, which determines the output of the program $f(x) = q_v$. Thus, f computes a function $f\colon \{0,1\}^n \to \mathbb{R}$.

Standard-order ROBPs are a special case of standard-order ROEPs, where $q_v = 1$ if $v \in V_{\text{accept}}$ and $q_v = 0$ if $v \in V_n \setminus V_{\text{accept}}$. If v is a vertex in an ROEP f, then the subprogram $f_{\to v}$ is an ROBP while the subprogram $f_{v\to}$ is an ROEP. We extend the notation q_v to the case that $v \notin V_n$ by the rule $q_v = \mathbb{E}[f_{v\to}]$, i.e., q_v is the expected label of the vertex reached by starting at v and taking a random walk to V_n. The "importance" of a vertex in an ROEP is captured by the following definition.

Definition 3.6 (Weight in an ROEP)**.** Let f be a standard-order ROEP with layers $V_0, \ldots, V_n$. Let $v \in V_i$ with $i < n$. Let $v[0]$ and $v[1]$ be the two out-neighbors of v. The *weight* of v, denoted $\mathrm{Weight}(v)$, is defined to be

$$\mathrm{Weight}(v) = |q_{v[0]} - q_{v[1]}|.$$

The weight of f is the sum of the weights of the vertices,[1] i.e.,

$$\mathrm{Weight}(f) = \sum_{i=0}^{n-1} \sum_{v \in V_i} \mathrm{Weight}(v).$$

In Fourier-analytic terms, we have the following formula[2] [202]:

$$\mathrm{Weight}(f) = 2 \sum_{i=1}^{n} \sum_{v \in V_{i-1}} \left| \widehat{f_{v\to}}(\{i\}) \right|.$$

Clearly, we always have $\mathrm{Weight}(v) \le 1$ and hence $\mathrm{Weight}(f) \le wn$. Now let us show that the INW generator fools low-weight programs with a shorter seed.

Theorem 3.10 (The BRRY generator for low-weight ROEPs [38])**.** For every $w, n, m \in \mathbb{N}$ and $\varepsilon > 0$, there is an explicit ε-PRG for width-w length-n standard-order ROEPs f that satisfy $\mathrm{Weight}(f) \le m$ with seed length

$$O(\log n \cdot (\log(wm/\varepsilon) + \log\log n)).$$

In the original paper, to prove Theorem 3.10, Braverman *et al.* [38] analyzed the INW generator in terms of *randomness extraction*, similar to the analysis of the Nisan-Zuckerman generator (see Section 3.4). Here, we will show how to carry out the analysis more directly, using the PRG characterization of spectral expanders (Lemma 3.2). The first step, like the original INW analysis, is to analyze two independent seeds. We use the following refinement of Lemma 3.6.

[1] In the original paper, Braverman *et al.* [38] considered edge weights instead of vertex weights. The two definitions are equivalent.

[2] To properly interpret the formula, we should think of the variables of $f_{v\to}$ as being numbered $i, i+1, i+2, \ldots, n$, so its Fourier coefficients are $\widehat{f_{v\to}}(S)$ for $S \subseteq \{i, i+1, \ldots, n\}$.

Lemma 3.11 (Refined analysis of the concatenation of independent pseudorandom strings)**.** Let n be an even positive integer, let $f\colon \{0,1\}^n \to \mathbb{R}$, and let X_L, X_R be independent random variables distributed over $\{0,1\}^{n/2}$. Define $f_L\colon \{0,1\}^{n/2} \to \mathbb{R}$ by

$$f_L(x) = \mathbb{E}[f(x, U_{n/2})],$$

and let $\mathcal{F}_R$ be the class of all functions $f_R\colon \{0,1\}^{n/2} \to \mathbb{R}$ of the form

$$f_R(x) = f(a, x)$$

where $a \in \{0,1\}^{n/2}$. If X_L fools f_L with error ε_L and X_R fools $\mathcal{F}_R$ with error ε_R, then (X_L, X_R) fools f with error $\varepsilon_L + \varepsilon_R$.

Proof. Sample $U \sim U_{n/2}$ so that X_L, X_R, U are mutually independent. Then

$$\begin{aligned}
\mathop{\mathbb{E}}_{X_L,X_R}[f(X_L,X_R)] = \mathop{\mathbb{E}}_{X_L}\left[\mathop{\mathbb{E}}_{X_R}[f(X_L,X_R)]\right] &= \mathop{\mathbb{E}}_{X_L}\left[\mathop{\mathbb{E}}_{U}[f(X_L,U)] \pm \varepsilon_R\right] \\
&= \mathop{\mathbb{E}}_{X_L}[f_L(X_L)] \pm \varepsilon_R \\
&= \mathbb{E}[f_L] \pm (\varepsilon_L + \varepsilon_R) \\
&= \mathbb{E}[f] \pm (\varepsilon_L + \varepsilon_R). \qquad \square
\end{aligned}$$

Next, we argue that the outputs of a very-low-weight ROEP fall in a small interval.

Lemma 3.12 (Low weight $\implies$ bounded range)**.** For every standard-order ROEP f and every input x, we have

$$|f(x) - \mathbb{E}[f]| \le \mathrm{Weight}(f)/2.$$

Proof. Let $v_{\mathrm{start}} = v_0, v_1, \ldots, v_n$ be the sequence of vertices visited in the computation $f(x)$. Then

$$\begin{aligned}
|f(x) - \mathbb{E}[f]| = |q_{v_n} - q_{v_0}| \le \sum_{i=1}^{n} |q_{v_i} - q_{v_{i-1}}| &= \sum_{i=1}^{n} \mathrm{Weight}(v_{i-1})/2 \\
&\le \mathrm{Weight}(f)/2. \qquad \square
\end{aligned}$$

Finally, we are ready to analyze two correlated seeds. Recall that the INW generator is based on a key lemma (Lemma 3.7) that says,

if G fools width-w length-$(n/2)$ ROBPs with error ε and (X, Y) is a random edge in a δ-spectral expander, then $(G(X), G(Y))$ fools width-w length-n ROBPs with error $2\varepsilon + w\delta/4$. We will prove the following more refined lemma that allows us to avoid the critical factor of two.

Lemma 3.13 (Recycling randomness for low-weight programs)**.** Suppose $G\colon \{0,1\}^s \to \{0,1\}^{n/2}$ fools every width-w length-$(n/2)$ ROEP f with error $\varepsilon \cdot \mathrm{Weight}(f)$. Fix some δ-spectral expander on the vertex set $\{0,1\}^s$, and sample a uniform random vertex X and a uniform random neighbor Y. Then the distribution $(G(X), G(Y))$ fools every width-w length-n ROEP f with error $(\varepsilon + w\delta/4) \cdot \mathrm{Weight}(f)$.

Proof. Let f be a width-w length-n ROEP. Let Weight_L and Weight_R denote the weights of the left half and right half of f, respectively, so $\mathrm{Weight}(f) = \mathrm{Weight}_L + \mathrm{Weight}_R$. Similarly to the proof of Lemma 3.7, we can write

$$f(x, y) = \sum_{v \in V_{n/2}} f_{\to v}(x) \cdot f_{v \to}(y).$$

Fix a vertex $v \in V_{n/2}$. Define $g(a) = f_{\to v}(G(a)) \in \{0,1\}$ and $h(b) = f_{v\to}(G(b)) \in \mathbb{R}$. By Lemma 3.12, the outputs of h fall in an interval of length $\mathrm{Weight}(f_{v\to}) = \mathrm{Weight}_R$. Therefore, by Lemma 3.2 and Fact 3.1, we have

$$\begin{aligned} |\,\mathbb{E}[g(X) \cdot h(Y)] - \mathbb{E}[g]\,\mathbb{E}[h]| &\le \delta \cdot \sqrt{\mathrm{Var}[g]} \cdot \sqrt{\mathrm{Var}[h]} \\ &\le \delta \cdot \frac{1}{2} \cdot \frac{\mathrm{Weight}_R}{2}. \end{aligned}$$

Therefore, if we let U, U' be independent uniform seeds, then by the triangle inequality,

$$\begin{aligned} |\,\mathbb{E}[f(G(U), G(U'))] - \mathbb{E}[f(G(X), G(Y))]| &\le \sum_{v \in V_{n/2}} \frac{\delta \cdot \mathrm{Weight}_R}{4} \\ &= \frac{w\delta \cdot \mathrm{Weight}_R}{4}. \end{aligned}$$

To bound $|\,\mathbb{E}[f(G(U), G(U'))] - \mathbb{E}[f]|$, we use Lemma 3.11. The function f_L that appears in that lemma is precisely the left half of f, a standard-order ROEP of weight Weight_L. Therefore, G fools f_L with error $\varepsilon \cdot$

Weight_L. Meanwhile, each function f_R considered in that lemma is a subprogram of the form $f_{v\to}$ for some $v \in V_{n/2}$, hence a standard-order ROEP of weight Weight_R. Therefore, G fools f_R with error $\varepsilon \cdot \text{Weight}_R$. Thus, Lemma 3.11 guarantees that

$$|\mathbb{E}[f(G(U), G(U'))] - \mathbb{E}[f]| \le \varepsilon \cdot \text{Weight}_L + \varepsilon \cdot \text{Weight}_R .$$

Therefore,

$$\begin{aligned} |\mathbb{E}[f(G(X), G(Y))] - \mathbb{E}[f]| &\le \varepsilon \cdot \text{Weight}(f) + \frac{w\delta \cdot \text{Weight}_R}{4} \\ &\le (\varepsilon + w\delta/4) \cdot \text{Weight}(f). \end{aligned}$$

□

Just like the original INW generator, we can use Lemma 3.13 to inductively construct a PRG for width-w length-n standard-order ROEPs. We start with a trivial PRG outputting a single bit. Then, in each of $\log n$ steps, we use a δ-spectral expander to recycle the seed and thereby double the output length. The final error is $\frac{1}{4} \cdot w\delta \cdot \text{Weight}(f) \cdot \log n$. To ε-fool standard-order ROEPs of weight at most m, we may set $\delta = \frac{4\varepsilon}{wm \log n}$. If we use a sparse expander (see Theorem 3.3), then in each step, the seed length of our PRG increases by an additive $O(\log(1/\delta))$ bits. Thus, the final seed length is $O(\log n \cdot \log(1/\delta))$, which is

$$O(\log n \cdot (\log(wm/\varepsilon) + \log\log n)),$$

completing the proof of Theorem 3.10.

3.3.3 Regular programs have low weight

Recall that our original goal was to design an improved PRG for standard-order regular ROBPs (Theorem 3.9). To apply the analysis of low-weight programs, Braverman *et al.* [38] showed that width-w standard-order regular ROBPs have weight at most $O(w^2)$.[3] The original

[3] In Braverman *et al.*'s definition of a regular ROBP [38], they only allowed a single accept vertex ($|V_{\text{accept}}| = 1$), hence their weight bound was $O(w)$ rather than $O(w^2)$. For the purpose of Theorem 3.9, it makes no difference whether we allow multiple accept vertices, because an ε-PRG for the single-accept-vertex model is also an (εw)-PRG for the multiple-accept-vertex model. However, there are other contexts in which bounding the number of accept vertices makes a huge difference [30], [127], [149], [192]–[194].

proof has a "combinatorial" flavor and is based on studying a certain pebble game. Here we present a more "analytic" interpretation of their argument.

Lemma 3.14 (Regular programs have bounded weight)**.** If f is a length-n standard-order regular ROEP, then

$$\mathrm{Weight}(f) \le 2 \sum_{u,v \in V_n} |q_u - q_v|.$$

In particular, if f is a width-w standard-order regular ROBP, then $\mathrm{Weight}(f) \le 2w^2$.

Proof. Recall that for a vertex $v \in V_i$, where $i \le n$, we defined $q_v = \mathbb{E}[f_{v\to}]$. For each i, let $D_i = \sum_{u,v\in V_i} |q_u - q_v|$. For $i < n$, let $\mathrm{Weight}_i = \sum_{u\in V_i} \mathrm{Weight}(u)$. For a vertex u, we let $u[0]$ and $u[1]$ denote the two out-neighbors of u. If we sample $X, Y \in \{0,1\}$ uniformly at random, then

$$\begin{aligned}
\frac{1}{2}\mathrm{Weight}_i + D_i &= \frac{1}{2}\sum_{u\in V_i} |q_{u[0]} - q_{u[1]}| + \sum_{\substack{u,v\in V_i \\ u\neq v}} |q_u - q_v| \\
&= \sum_{u\in V_i} \mathop{\mathbb{E}}_{X,Y}\left[|q_{u[X]} - q_{u[Y]}|\right] + \sum_{\substack{u,v\in V_i \\ u\neq v}} \left|\mathop{\mathbb{E}}_{X}\left[q_{u[X]}\right] - \mathop{\mathbb{E}}_{Y}\left[q_{v[Y]}\right]\right| \\
&\le \sum_{u\in V_i} \mathop{\mathbb{E}}_{X,Y}\left[|q_{u[X]} - q_{u[Y]}|\right] + \sum_{\substack{u,v\in V_i \\ u\neq v}} \mathop{\mathbb{E}}_{X,Y}\left[|q_{u[X]} - q_{v[Y]}|\right] \\
&= \sum_{u,v\in V_i} \mathop{\mathbb{E}}_{X,Y}\left[|q_{u[X]} - q_{v[Y]}|\right] \\
&= \sum_{u,v\in V_{i+1}} \frac{\deg^-(u)\cdot\deg^-(v)}{4} \cdot |q_u - q_v| \\
&= D_{i+1},
\end{aligned}$$

where the inequality follows from the triangle inequality and the final step uses the fact that f is regular. Rearranging, we have shown $\frac{1}{2}\mathrm{Weight}_i \le D_{i+1} - D_i$. Summing over $i < n$ completes the proof. □

Combining Lemma 3.14 and Theorem 3.10 completes the proof of Theorem 3.9.

Besides regular ROBPs, another well-studied class of branching programs is *permutation ROBPs*, which are regular ROBPs with the additional assumption that for each vertex v, the two incoming edges of v have distinct labels. Braverman *et al.*'s paper [38] is one of a great number of papers studying pseudorandomness for regular and permutation ROBPs [2], [30], [38], [39], [49], [53], [54], [56], [74], [127], [145], [149], [192]–[194], [202], [224]. We will revisit permutation ROBPs in Section 5.2.

Let us highlight one open problem regarding these topics. As mentioned in Section 3.2, Meka *et al.* [171] designed a PRG for width-3 standard-order ROBPs with seed length $\widetilde{O}(\log n \cdot \log(1/\varepsilon))$. Roughly speaking, their approach involves reducing to the permutation case and then applying the INW generator. The INW generator has a $\log n \cdot \log(1/\varepsilon)$ term in its seed length (even for permutation branching programs [192]), so Meka *et al.* [171] get no improvement over Nisan's PRG when the error is $1/n$. This motivates the following problem.

Open Problem 3.4 (Low-error PRGs for width-3 standard-order permutation ROBPs)**.** Design an explicit PRG for width-3 standard-order permutation ROBPs with error $1/n$ and seed length $o(\log^2 n)$.

Although Open Problem 3.4 is unsolved, we do at least have good explicit constructions of low-error *hitting set generators and weighted PRGs* for permutation ROBPs, regular ROBPs, and width-3 ROBPs [30], [53], [54], [105], [193].

3.4 The Nisan-Zuckerman Generator for Short, Wide ROBPs

In the previous section, we focused on width-w length-n standard-order ROBPs where $w \ll n$. In this section, let us study the opposite regime, i.e., $w \gg n$. One reason to study this regime is that it corresponds to derandomizing space-bounded algorithms that only use a little bit of randomness in the first place.

Nisan and Zuckerman designed a PRG with *optimal* seed length $O(\log w)$ for the case that $n = \operatorname{polylog} w$ and the error parameter is moderate [184]. In contrast, the INW generator's seed length is $\Theta(\log w \cdot \log\log w)$ for such parameters.

Theorem 3.15 (The Nisan-Zuckerman generator)**.** Let $\nu > 0$ and $c \geq 1$ be constants. For all $w \in \mathbb{N}$, there is an explicit PRG for width-w length-$(\log^c w)$ standard-order ROBPs with seed length $O(\log w)$ and error $2^{-\log^{1-\nu} w}$.

Recall that when we say the PRG is "explicit," we mean that its time complexity is polynomial in its output length (Definition 1.4). In the case of Theorem 3.15, this means that the time complexity of the PRG is polylog(w). As usual in this text, we will refrain from carefully verifying this time complexity bound. However, it will turn out to be useful to analyze the *space complexity* of the PRG. As we will see, the Nisan-Zuckerman PRG can be computed using $O(\log w)$ bits of space. Using the connection between randomized space-bounded algorithms and ROBPs (Claims 1.3 and 1.4), one can show the following striking corollary: If a decision problem can be solved by a randomized algorithm using S bits of space and poly(S) random bits, then it can also be solved by a deterministic algorithm using $O(S)$ bits of space [184].

3.4.1 Randomness extractors

The proof of Theorem 3.15 uses seeded *randomness extractors* (introduced by Nisan and Zuckerman [184]) to recycle randomness. Informally, a randomness extractor is a tool for converting an "imperfect" source of randomness into near-uniform random bits. The following definition specifies our model of imperfect randomness.

Definition 3.7 (k-Source [65], [252])**.** The *min-entropy* of a distribution X, denoted $\mathrm{H}_{\min}(X)$, is defined by

$$\mathrm{H}_{\min}(X) = \min_{x \in \mathrm{supp}(X)} \log\left(\frac{1}{\Pr[X = x]}\right).$$

We say that a random variable X on $\{0,1\}^t$ is a *k-source* if $\mathrm{H}_{\min}(X) \geq k$, that is, for every $x \in \{0,1\}^t$, $\Pr[X = x] \leq 2^{-k}$.

(Min-entropy is also sometimes denoted $\mathrm{H}_\infty(X)$.) And how shall we define extractors? It would be natural to define an extractor to be a function $\mathsf{Ext}\colon \{0,1\}^t \to \{0,1\}^m$ such that for every k-source X, the distribution $\mathsf{Ext}(X)$ is statistically close to U_m. Unfortunately, that

natural definition would be too strong, because of the following simple impossibility proof.

Proposition 3.2 (Impossibility of seedless randomness extraction)**.** Let $\mathsf{Ext}\colon \{0,1\}^t \to \{0,1\}$ be any function. There exists a $(t-1)$-source X and a value $b \in \{0,1\}$ such that $\Pr[\mathsf{Ext}(X) = b] = 1$.

Proof. Since $\{0,1\}^t = \mathsf{Ext}^{-1}(0) \cup \mathsf{Ext}^{-1}(1)$, there is some b such that $|\mathsf{Ext}^{-1}(b)| \geq 2^{t-1}$. Let X be uniformly distributed over $\mathsf{Ext}^{-1}(b)$. □

There are multiple approaches for evading Proposition 3.2. Our approach will be to allow the extractor to use a small truly random *seed*, in addition to the k-source X. To make this precise, let $\mathrm{d_{TV}}(\cdot,\cdot)$ denote total variation distance.

Definition 3.8 (Seeded Randomness Extractor)**.** A function $\mathsf{Ext}\colon \{0,1\}^t \times \{0,1\}^d \to \{0,1\}^m$ is a (k,ε)-*extractor* if for every k-source X,

$$\mathrm{d_{TV}}(\mathsf{Ext}(X, U_d), U_m) \leq \varepsilon,$$

where U_d is independent of X.

Intuitively, extractors and PRGs both produce some type of random bits, but they are incomparable:

- The output of an extractor fools *all* tests $f\colon \{0,1\}^m \to \{0,1\}$ (this is equivalent to being close to uniform in total variation distance), whereas the output of a PRG only fools *some* tests.
- On the other hand, an extractor requires a seed *and* an imperfect source of randomness, whereas a PRG only requires a seed.

It is also possible to view extractors as a special case of PRGs,[4] but we will not use that connection in this text.

[4] Let $\mathsf{Ext}\colon \{0,1\}^t \times \{0,1\}^d \to \{0,1\}^m$ be a function, sample $X \sim U_t$, and let $Y = \mathsf{Ext}(X, U_d)$, where X and U_d are independent. Let 2^k be a positive integer and let $\mathcal{F}$ be the class of functions $f\colon \{0,1\}^{t+m} \to \{0,1\}$ of the form $f(x,y) = g(x) \cdot h(y)$, where $g\colon \{0,1\}^t \to \{0,1\}$, $h\colon \{0,1\}^m \to \{0,1\}$, and $\mathbb{E}[g] = 2^{k-t}$. Then Ext is a (k,ε)-extractor if and only if (X,Y) fools $\mathcal{F}$ with error $\varepsilon \cdot 2^{k-t}$ [238, Proposition 6.21].

There is a rich and beautiful theory of seeded randomness extractors that goes beyond the scope of this text. See, for example, Nisan and Ta-Shma's survey [182], Shaltiel's surveys [211], [212], or Vadhan's monograph [238]. We will take for granted an explicit construction due to Guruswami *et al.* [109], or rather a space-optimized version by Kane *et al.* [140].

Theorem 3.16 ([109], [140])**.** For every $k \leq t$ and $\varepsilon > 0$, there is an (k, ε)-extractor $\mathsf{Ext} : \{0,1\}^t \times \{0,1\}^d \to \{0,1\}^m$ with $m \geq k/2$ and $d = O(\log(t/\varepsilon))$. Furthermore, given (x, y, k) as input, $\mathsf{Ext}(x, y)$ can be computed in space $O(t + \log(1/\varepsilon))$.

Toward proving Theorem 3.15, we will first study a PRG of the following form:

$$G(x, y_1, \ldots, y_\ell) = (\mathsf{Ext}(x, y_1), \ldots, \mathsf{Ext}(x, y_\ell)). \tag{3.1}$$

We will then compose this PRG with itself to prove Theorem 3.15. Here's the intuition behind Equation (3.1). Let $(X, Y_1, \ldots, Y_\ell)$ be a uniform random seed. After the ROBP reads some prefix $(\mathsf{Ext}(X, Y_1), \ldots, \mathsf{Ext}(X, Y_i))$, it only "remembers" $\log w$ bits of information about what it has seen. We will set $|X| = 3 \log w$. Since the ROBP only "knows" $\log w$ bits about X, the random variable X should still have $2 \log w$ bits of entropy "from the ROBP's perspective." Therefore, $\mathsf{Ext}(X, Y_{i+1})$ should appear nearly uniform to the ROBP.

The most elegant way to formalize this intuition is to use the concept of conditional min-entropy introduced by Dodis *et al.* [79].

Definition 3.9 (Conditional min-entropy)**.** Let X and A be jointly distributed random variables. The *conditional min-entropy* of X given A is

$$\widetilde{\mathrm{H}}_{\min}(X \mid A) = \log \left(\frac{1}{\mathbb{E}_{a \sim A}[\max_{x \in \mathrm{supp}(X)} \Pr[X = x \mid A = a]]} \right).$$

Conditional min-entropy can be interpreted in terms of strategies for guessing X after seeing A [238, Problem 6.7]. Conditional min-entropy satisfies the following intuitive "chain rule" first proven by Dodis *et al.* [79, Lemma 2.2].

Lemma 3.17 (Chain rule for min-entropy)**.** If $|\operatorname{supp}(A)| \le w$, then

$$\widetilde{\mathrm{H}}_{\min}(X \mid A) \ge \mathrm{H}_{\min}(X) - \log w.$$

If X has a lot of min-entropy given A, then intuitively, we should expect that $\mathsf{Ext}(X, U_d)$ looks uniform even given A. This intuition is correct, as expressed by the following lemma, by Vadhan [238]. (See also a similar earlier lemma by Dodis *et al.* [79, Lemma 2.3].)

Lemma 3.18 (Extracting from sources with high conditional min-entropy [238, Problem 6.8])**.** Let $\mathsf{Ext}\colon \{0,1\}^t \times \{0,1\}^d \to \{0,1\}^m$ be a (k,ε)-extractor. If $\widetilde{\mathrm{H}}_{\min}(X \mid A) \ge k$, then

$$\mathrm{d}_{\mathrm{TV}}\Big((\mathsf{Ext}(X, U_d), A), (U_m, A)\Big) \le 3\varepsilon.$$

(Here U_d is independent of (X, A) and U_m is independent of A.)

Finally, we will need the following standard "data processing inequality," which says that applying a function – even a randomized function – can only make two distributions closer.

Lemma 3.19 (Data processing inequality for total variation distance)**.** Let A and $\widetilde{A}$ be random variables over the same space. Let R be independent of both A and $\widetilde{A}$, and let f be any function. Then

$$\mathrm{d}_{\mathrm{TV}}\left(f(A,R), f(\widetilde{A},R)\right) \le \mathrm{d}_{\mathrm{TV}}\left(A, \widetilde{A}\right).$$

3.4.2 Using extractors to fool standard-order ROBPs

We are now ready to analyze the PRG of Equation (3.1). We will show that it achieves the following parameters.

Lemma 3.20 (One iteration of the Nisan-Zuckerman generator)**.** Let $w, n \in \mathbb{N}$ with $n \ge \log w$, and let $\varepsilon > 0$. There is an explicit ε-PRG for width-w length-n standard-order ROBPs with seed length

$$O\left(\log w + \frac{n \log(n/\varepsilon)}{\log w}\right).$$

Furthermore, the generator can be computed by an algorithm that reads the seed once from left to right and runs in space $O(\log(wn/\varepsilon))$.

Proof. Let $\ell = \frac{n}{\log w}$. Let $\varepsilon' = \frac{\varepsilon}{3\ell}$, and let $\mathsf{Ext}\colon \{0,1\}^{3\log w} \times \{0,1\}^d \to \{0,1\}^{\log w}$ be the $(2\log w, \varepsilon')$-extractor of Theorem 3.16, so $d = O(\log(n/\varepsilon))$. The PRG G is given by Equation (3.1), which we repeat below for convenience:

$$G(x, y_1, \ldots, y_\ell) = (\mathsf{Ext}(x, y_1), \ldots, \mathsf{Ext}(x, y_\ell)).$$

The seed length and efficiency claims are clear.

As for correctness, our job is to show that G fools every width-w length-n standard-order ROBP. It will be convenient to group the n layers into ℓ blocks of $\log w$ layers each. This can be viewed as a width-w length-ℓ ROBP *over the alphabet* $\{0,1\}^{\log w}$, i.e., each vertex has w outgoing edges labeled with all strings in $\{0,1\}^{\log w}$. Let f be such a program, with layers $V_0, \ldots, V_\ell$. For $i < n$, $a \in V_i$, and $r \in \{0,1\}^{\log w}$, let $a[r]$ denote the vertex reached from a by traversing the outgoing edge with label r.

Sample $X, Y_1, \ldots, Y_\ell, R_1, \ldots, R_\ell$ independently and uniformly at random, where $X \in \{0,1\}^{3\log w}$, $Y_i \in \{0,1\}^d$, and $R_i \in \{0,1\}^{\log w}$. Let $A_0, \ldots, A_\ell$ be the sequence of vertices reached when f reads the truly random bits $R_1, \ldots, R_\ell$, i.e., $A_0 = v_{\text{start}}$ and

$$A_i = A_{i-1}[R_i].$$

Similarly, let $\widetilde{R}_i = \mathsf{Ext}(X, Y_i)$, and let $\widetilde{A}_0, \ldots, \widetilde{A}_\ell$ be the sequence of vertices reached when f reads the pseudorandom bits $\widetilde{R}_1, \ldots, \widetilde{R}_\ell$, i.e., $\widetilde{A}_0 = v_{\text{start}}$ and

$$\widetilde{A}_i = \widetilde{A}_{i-1}[\widetilde{R}_i].$$

We will prove by induction on i that $\mathrm{d_{TV}}\left(A_i, \widetilde{A}_i\right) \le 3\varepsilon' i$. The base case $i = 0$ is trivial. Now fix $i > 0$. By the triangle inequality, we have

$$\begin{aligned}\mathrm{d_{TV}}\left(A_i, \widetilde{A}_i\right) &= \mathrm{d_{TV}}\left(A_{i-1}[R_i], \widetilde{A}_{i-1}[\widetilde{R}_i]\right) \\ &\le \mathrm{d_{TV}}\left(A_{i-1}[R_i], \widetilde{A}_{i-1}[R_i]\right) + \mathrm{d_{TV}}\left(\widetilde{A}_{i-1}[R_i], \widetilde{A}_{i-1}[\widetilde{R}_i]\right).\end{aligned}$$

Applying the data processing inequality (Lemma 3.19) to each term, we get

$$\mathrm{d_{TV}}\left(A_i, \widetilde{A}_i\right) \le \mathrm{d_{TV}}\left(A_{i-1}, \widetilde{A}_{i-1}\right) + \mathrm{d_{TV}}\left((\widetilde{A}_{i-1}, R_i), (\widetilde{A}_{i-1}, \widetilde{R}_i)\right).$$

The first term is at most $3\varepsilon' \cdot (i-1)$ by induction. As for the second term, the chain rule for min-entropy (Lemma 3.17) implies that

$$\widetilde{\mathrm{H}}_{\min}(X \mid \widetilde{A}_{i-1}) \geq 2 \log w.$$

Therefore, Lemma 3.18 guarantees that the second term is at most $3\varepsilon'$. Summing up, we get $\mathrm{d_{TV}}\left(A_i, \widetilde{A}_i\right) \leq 3\varepsilon' i$ as claimed. Therefore,

$$\begin{aligned} \left|\mathbb{E}[f] - \mathbb{E}[f(G(U_{3\log w + \ell d})]\right| &= \left|\Pr[A_\ell \in V_{\text{accept}}] - \Pr\left[\widetilde{A}_\ell \in V_{\text{accept}}\right]\right| \\ &\leq \mathrm{d_{TV}}\left(A_\ell, \widetilde{A}_\ell\right) \\ &\leq \varepsilon. \end{aligned}$$

□

Lemma 3.20 implies Theorem 3.15 when c is small, such as $c = 1.5$. To handle larger c, we compose the generator of Lemma 3.20 with itself. More details are below.

Proof of Theorem 3.15. Let $\varepsilon = 2^{-\log^{1-\nu} w}$. Let $w_1 = w$ and $n_1 = \log^c w$, and let G_1 be the ε-PRG of Lemma 3.20 for width-w_1 length-n_1 ROBPs. This generator G_1 has seed length $n_2 = O(\log^{c-\nu} w)$. Furthermore, G_1 can be computed by an algorithm that reads the seed once from left to right and runs in space $O(\log w)$. It follows that for any width-w standard-order ROBP f, the function $f_2(x) \stackrel{\text{def}}{=} f(G_1(x))$ can be computed by an ROBP of width $w_2 = \mathrm{poly}(w)$. Therefore, let G_2 be the ε-PRG of Lemma 3.20 for width-w_2 length-n_2 standard-order ROBPs, with seed length $n_3 = O(\log^{c-2\nu} w)$. The composition $G_1 \circ G_2$ fools f with error 2ε. Once again, $f(G_1(G_2(x)))$ can be computed by a standard-order ROBP of width $w_3 = \mathrm{poly}(w)$. Continuing in this way for $O(c/\nu) = O(1)$ steps, we obtain an explicit $O(\varepsilon)$-PRG for f with seed length $O(\log w)$. □

An interesting feature of the proof of Theorem 3.15 is that the *efficiency* of the PRG of Lemma 3.20 is a key part of the proof of *correctness* of the final PRG.

The Nisan-Zuckerman generator and the INW generator are both based on "recycling" part of the seed so it can be reused to generate more pseudorandom bits. The strength of the Nisan-Zuckerman generator is

that we use fewer than $\log w$ fresh random bits for the recycling process, which enables us to achieve a nontrivial stretch with an overall seed length of $O(\log w)$. On the other hand, the INW generator has a better seed length when n (the number of pseudorandom bits) is large.

After Nisan and Zuckerman's work [184], Armoni developed their techniques further and designed a PRG that "interpolates" between the INW generator and the Nisan-Zuckerman generator [13]. Armoni's seed length was slightly improved later [140] by plugging in extractors that were developed after Armoni's work [109] (and analyzing their space complexity). At the extremes $n \geq w$ and $n \leq \text{polylog}(w)$, Armoni's generator has the same seed length as the INW generator and the Nisan-Zuckerman generator, respectively. However, in the intermediate regime $\text{polylog}(w) \ll n \ll w$, Armoni's generator (as optimized by Kane *et al.* [140]) is the best PRG known for ROBPs. It outperforms the INW generator in this regime by a factor of $\log \log w$ (for moderate error), and this slight improvement (combined with other techniques) later led to the current best unconditional derandomization of space-bounded computation [124].

The Nisan-Zuckerman generator does not have optimal error. Improving the error is an appealing open problem.

Open Problem 3.5 (Improving the error of the Nisan-Zuckerman generator)**.** For some function $n = \omega(\log w)$, design an explicit PRG for width-w length-n standard-order ROBPs with seed length $O(\log w)$ and error $1/w$.

The "hitting set generator" analogue of Open Problem 3.5 has been solved [3], [128].

4

PRGs and Hardness

From an algorithm-design perspective, an explicit PRG construction is a "positive" theorem – an *upper bound* on the resources needed to sample a distribution that fools such-and-such class. An explicit PRG can be used as a building block in a larger algorithm.

On the other hand, from a complexity perspective, a construction of an explicit PRG G fooling a model $\mathcal{F}$ provides a concrete example of a task that $\mathcal{F}$ *cannot* do. A PRG construction is thus a "negative" theorem – an impossibility result – a *lower bound* on the resources required to distinguish the generator's output from the uniform distribution.

In this section, we will explore connections between PRGs and more traditional "lower bound" notions. First, in Section 4.1, we will investigate the "PRGs as lower bounds" viewpoint in more detail and discuss its implications for the prospect of PRG design. Then, in Section 4.2, we will go the other direction, i.e., we will show how to construct PRGs from suitable lower bounds.

4.1 PRGs as High-quality Lower Bounds

4.1.1 PRGs imply hard Boolean functions

We begin by showing that a PRG that fools $\mathcal{F}$ can be converted into a Boolean function that cannot be computed by $\mathcal{F}$. For the simplest version of this reduction, we consider a PRG with one-bit stretch.

Proposition 4.1 (PRG $\implies$ hard function)**.** Let $n \in \mathbb{N}$, and let $\mathcal{F}$ be a class of functions $f\colon \{0,1\}^n \to \{0,1\}$. Let $G\colon \{0,1\}^{n-1} \to \{0,1\}^n$ be an ε-PRG for $\mathcal{F}$ where $\varepsilon < 1/2$. Define $h\colon \{0,1\}^n \to \{0,1\}$ by

$$h(x) = 1 \iff \text{there exists } y \text{ such that } G(y) = x.$$

Then $h \notin \mathcal{F}$.

Proof. The generator G does not ε-fool h, because $\mathbb{E}[h(G(U_{n-1}))] = 1$ whereas $\mathbb{E}[h(U_n)] \leq \frac{2^{n-1}}{2^n} = \frac{1}{2}$. □

Note that if the PRG G is explicit, then the hard function h in Proposition 4.1 is "somewhat explicit," namely $h \in \mathbf{NP}$.

We will refine Proposition 4.1 in two respects. First, if G has a seed length s that is significantly smaller than n, then we can modify our hard function so it is a function of approximately s bits instead of a function of n bits. This is an improvement, because when we prove hardness results, we want to say that computing h requires a large amount of resources (time, space, etc.) *compared to the input length.* To carry out this improvement, we make the mild assumption that the class $\mathcal{F}$ is "closed under restrictions" as defined below.

Definition 4.1 (Closure under restrictions)**.** Let $\mathcal{F}$ be a class of functions f on $\{0,1\}^n$. We say that $\mathcal{F}$ is *closed under restrictions* if the following holds. Let $f \in \mathcal{F}$, let $i \in [n]$, and let $b \in \{0,1\}$. Define $g(x) = f(x^{(i \to b)})$, where $x^{(i \to b)}$ denotes the string obtained from x by replacing the i-th bit with b. Then $g \in \mathcal{F}$.

We will also refine Proposition 4.1 in a second way. The conclusion of Proposition 4.1 is a "worst-case" lower bound, i.e., it merely asserts that there is no function in $\mathcal{F}$ that correctly computes the hard function h on *all* inputs. We will replace this worst-case lower bound with an

"average-case" lower bound, which is stronger. As formalized below, an average-case lower bound asserts that no function in $\mathcal{F}$ can correctly compute the hard function on significantly more than *half* of the inputs (with respect to some distribution).

Definition 4.2 (Average-case hardness)**.** Let $\mathcal{F}$ be a class of functions $f\colon \{0,1\}^r \to \{0,1\}$, let $h\colon \{0,1\}^r \to \{0,1\}$, and let D be a distribution over $\{0,1\}^r$. We say that h is ε*-hard for* $\mathcal{F}$ *with respect to* D if for every $f \in \mathcal{F}$,

$$\left| \Pr_{X \sim D}[f(X) = h(X)] - \frac{1}{2} \right| \leq \varepsilon.$$

(Note that in Definition 4.2, we assume that the success probability is neither significantly more than $1/2$, nor significantly *less* than $1/2$. This is just for convenience. The two bounds are equivalent if $\mathcal{F}$ is closed under complement.) The following refined reduction, first formalized by Viola [242], shows that PRGs imply average-case hardness.

Proposition 4.2 (PRG $\implies$ average-case hardness [242])**.** Let $\mathcal{F}$ be a class of functions $f\colon \{0,1\}^n \to \{0,1\}$ that is closed under restrictions. Let $G\colon \{0,1\}^s \to \{0,1\}^n$ be an ε-PRG for $\mathcal{F}$. Let $r = s + \lceil \log(1/\varepsilon) \rceil$ and assume that $r \leq n$. Define $h\colon \{0,1\}^r \to \{0,1\}$ by

$$h(x) = 1 \iff \text{there exist } y, z \text{ such that } G(y) = (x, z).$$

Let $\mathcal{F}'$ be the class of functions $f\colon \{0,1\}^r \to \{0,1\}$ of the form $f(x) = f_0(x, a)$ where $f_0 \in \mathcal{F}$ and $a \in \{0,1\}^{n-r}$. Let $D = \frac{1}{2}U_r + \frac{1}{2}G(U_s)_{1\ldots r}$, i.e., the distribution D is a balanced convex combination of the distributions U_r and $G(U_s)_{1\ldots r}$.[1] Then h is ε-hard for $\mathcal{F}'$ with respect to D.

Proof. Sample $U \sim U_r$ and $U' \sim U_s$. Fix any $f \in \mathcal{F}'$, say $f(x) = f_0(x, a)$. Then

$$\begin{aligned}
&\Pr_{X \sim D}[f(X) = h(X)] \\
&\quad = \frac{1}{2}\Pr[f(U) = h(U)] + \frac{1}{2}\Pr[f(G(U')_{1\ldots r}) = h(G(U')_{1\ldots r})] \\
&\quad = \frac{1}{2}\Pr[f(U) = h(U)] + \frac{1}{2}\Pr[f(G(U')_{1\ldots r}) = 1]
\end{aligned}$$

[1] We write $x_{1\ldots r}$ to denote the first r bits of x.

$$\begin{aligned}
&\leq \frac{1}{2}\Pr[f(U)=0] + \frac{1}{2}\Pr[h(U)=1] + \frac{1}{2}\left(\Pr[f(U)=1]+\varepsilon\right)\\
&= \frac{1}{2} + \frac{\varepsilon + \mathbb{E}[h]}{2}\\
&\leq \frac{1}{2} + \varepsilon,
\end{aligned}$$

where the first inequality uses the fact that $\mathcal{F}$ is closed under restrictions and hence the distribution $G(U')_{1\ldots r}$ fools $\mathcal{F}'$ with error ε. Since G also fools complements of functions in $\mathcal{F}$, we also get the reverse inequality $\Pr_{X\sim D}[f(X)=h(X)] \geq \frac{1}{2}-\varepsilon$. □

Proposition 4.2 demonstrates that there is a *hierarchy of lower bounds*:

$$\text{PRG} \implies \begin{pmatrix}\text{Average-Case}\\ \text{Lower Bound}\end{pmatrix} \implies \begin{pmatrix}\text{Worst-Case}\\ \text{Lower Bound}\end{pmatrix}. \tag{4.1}$$

A PRG construction is a lower bound that is particularly strong, qualitatively speaking. Admittedly, the average-case lower bound in Proposition 4.2 is with respect to a certain (explicitly-sampleable) *non-uniform* distribution D, whereas traditionally, we seek average-case lower bounds with respect to the uniform distribution. However, it turns out that in many cases the two types of average-case lower bound are essentially equivalent [55].

4.1.2 The lack-of-lower-bounds barrier

Looking at Equation (4.1), an optimist might hope to use PRGs to prove new lower bounds. In practice, however, *lower bounds come first.* Therefore, a *lack* of known lower bounds for a particular model can be considered a type of *barrier* to constructing PRGs for that model.

For example, let $\mathcal{F}$ be the class of all functions $f\colon \{0,1\}^n \to \{0,1\}$ that can be computed by Boolean circuits of size $n^{\log n}$. Nonexplicitly, there exists an ε-PRG that fools $\mathcal{F}$ with seed length $O(\log^2 n + \log(1/\varepsilon))$ (Proposition 1.1). If we could construct an *explicit* 0.49-PRG that fools $\mathcal{F}$ with seed length $n-1$, then by Proposition 4.1, we would also be able to construct a function $h \in \mathbf{NP}$ on n bits with circuit complexity greater than $n^{\log n}$. This would be a huge breakthrough in circuit complexity,

and in particular it would imply $\mathbf{P} \neq \mathbf{NP}$. The conventional wisdom is that one should not try to design an unconditional PRG for $\mathcal{F}$ until *after* proving the corresponding circuit lower bounds.

The good news, as we have seen already in Sections 2 and 3, is that this "lack-of-lower-bounds barrier" still leaves plenty of room for a rich theory of unconditional PRGs. After all, highly nontrivial lower bounds are *already known* for many interesting classes, and hence we can try to design PRGs with matching parameters. For example, for $\mathbf{AC}^0$ circuits (see Definition 2.13), the state-of-the-art lower bounds are as follows.

Theorem 4.1 (Parity is hard for $\mathbf{AC}^0$ circuits [118], [129])**.** For every $m, d \in \mathbb{N}$ and $\varepsilon > 0$, there exists a value $r = O(\log^{d-1} m \cdot \log(1/\varepsilon))$ such that the parity function on r bits is ε-hard for depth-d size-m $\mathbf{AC}^0$ circuits with respect to the uniform distribution.

In light of Theorem 4.1, we may reasonably hope to design an explicit ε-PRG for depth-d size-m $\mathbf{AC}^0$ circuits with seed length as low as

$$O(\log^{d-1} m \cdot \log(1/\varepsilon)). \tag{4.2}$$

(Assume for simplicity that $d \geq 2$ and $m \geq n$, where n is the number of pseudorandom bits.) Indeed, recall that Braverman's theorem (Section 2.6) implies a fairly similar seed length of $\log^{O(d)} m \cdot \log(1/\varepsilon)$. Furthermore, as we will discuss in Section 5.3, explicit PRGs for $\mathbf{AC}^0$ are known with better seed lengths, getting very close to the bound of Equation (4.2). The optimal seed length would be $O(\log(m/\varepsilon))$, independent of d, but we should probably not expect to go below $O(\log^{d-1} m \cdot \log(1/\varepsilon))$ until *after* improving the known lower bounds for $\mathbf{AC}^0$ (Theorem 4.1).

As another example, let us consider standard-order ROBPs. Here the situation is better, because optimal lower bounds are known:

Proposition 4.3 (Inner product is hard for standard-order ROBPs)**.** For each even positive integer $2r \in \mathbb{N}$, let $\mathsf{IP}_{2r} \colon \{0,1\}^{2r} \to \{0,1\}$ denote the function

$$\mathsf{IP}_{2r}(x, y) = \sum_{i=1}^{r} x_i y_i \bmod 2.$$

For every $w \in \mathbb{N}$ and $\varepsilon > 0$, there exists a value $r = O(\log(w/\varepsilon))$ such that IP_{2r} is ε-hard for width-w standard-order ROBPs with respect to the uniform distribution.

Proof. For every width-w length-$(2r)$ standard-order ROBP f, there is a communication protocol in which Alice holds x, Bob holds y, they communicate $1 + \lceil \log w \rceil$ bits, and then they output $f(x, y)$. (Alice simulates the first half of f and sends the state to Bob; Bob simulates the second half of f and sends the output bit to Alice.) The proposition follows by plugging in classic lower bounds on the average-case communication complexity of IP_{2r} (e.g., see Rao and Yehudayoff's text [196, Theorem 5.6]). □

Thus, for standard-order ROBPs, there is no lack-of-lower-bounds barrier, and it is perfectly reasonable to try to design optimal PRGs.

4.2 The Nisan-Wigderson Framework

In the previous section, we saw that a PRG implies a hard decision problem. In this section, we will discuss a famous method due to Nisan and Wigderson [183] for going the other way: converting a hard decision problem into a PRG. Thus, we will establish a fairly tight connection between PRGs and lower bounds (although there are losses that are important in some cases, so the connection is not a perfect equivalence).

4.2.1 Constructing a PRG from a Hard Function

Let $h\colon \{0,1\}^r \to \{0,1\}$ be a candidate "hard function." Let $s > r$, and let $S_1, \ldots, S_n$ be a family of r-subsets of $[s]$, i.e., $S_i \subseteq [s]$ and $|S_i| = r$. The Nisan-Wigderson generator $G\colon \{0,1\}^s \to \{0,1\}^n$ is given by

$$G(x) = (h(x_{S_1}), \ldots, h(x_{S_n})), \tag{4.3}$$

where $x_S = x_{i_1} x_{i_2} \ldots x_{i_r}$ when $S = \{i_1 < i_2 < \cdots < i_r\}$. In words, the generator applies the hard function to n different substrings of the seed.

We will prove that this construction works under certain assumptions on h and $S_1, \ldots, S_n$. Intuitively, to ensure that the output bits of G appear to be independent, we should apply the hard function h to

inputs that are almost "unrelated." We will achieve this property by requiring that the sets of indices $S_1, \ldots, S_n$ are "nearly" disjoint.

Definition 4.3 (Nearly disjoint sets)**.** We say that sets $S_1, \ldots, S_n$ are *k-nearly disjoint*[2] if $|S_i \cap S_j| \leq k$ for all distinct $i, j \in [n]$.

Meanwhile, the function h should be hard to compute, even on average (see Definition 4.2). To fool a function f, we will assume that h is hard for compositions of f with arbitrary k-juntas (see Definition 2.11). Under these assumptions, the Nisan-Wigderson generator achieves the following parameters.

Theorem 4.2 (Nisan-Wigderson reduction)**.** Let $f\colon \{0,1\}^n \to \{0,1\}$. Suppose $h\colon \{0,1\}^r \to \{0,1\}$ is ε-hard for $f \circ \mathsf{JUNTA}_{r,k}$ with respect to the uniform distribution, and suppose that $S_1, \ldots, S_n$ are k-nearly disjoint r-subsets of $[s]$ for some $s > r$. Then the Nisan-Wigderson generator G given by Equation (4.3) fools f with error $\varepsilon \cdot n$.

4.2.2 Analysis: Unpredictability

The proof of Theorem 4.2 is based on the problem of *predicting* the next bit of a pseudorandom string after seeing the first few bits. A truly random string is completely unpredictable.

Definition 4.4 (Unpredictability)**.** Let X be a distribution over $\{0,1\}^n$, let $f\colon \{0,1\}^n \to \{0,1\}$, and let $\varepsilon > 0$. We say that X is *ε-unpredictable for f* if for every $i \in [n]$ and every $a \in \{0,1\}^{n-i+1}$, we have

$$\left|\Pr[f(X_1, X_2, \ldots, X_{i-1}, a) = X_i] - \frac{1}{2}\right| \leq \varepsilon.$$

Equivalently, X fools the test $x \mapsto f(x_1, x_2, \ldots, x_{i-1}, a) \oplus x_i$ with error ε. We say that a generator $G\colon \{0,1\}^s \to \{0,1\}^n$ is ε-unpredictable for f if $G(U_s)$ is ε-unpredictable for f.

In some of the early literature, something like Definition 4.4 is actually taken to be the definition of a PRG [28], [214]. As a first step toward proving Theorem 4.2, let us show that the Nisan-Wigderson generator is unpredictable.

[2] A family of nearly disjoint sets, all of the same size, is also known as a *design* or a *partial design* or a *partial Steiner system* or a *packing*.

Lemma 4.3 (The NW PRG is unpredictable)**.** Under the assumptions of Theorem 4.2, the Nisan-Wigderson generator G is ε-unpredictable for f.

Proof. Fix $i \in [n]$ and $a \in \{0,1\}^{n-i+1}$. Sample a seed $Y \in \{0,1\}^s$ uniformly at random, and let $X = G(Y)$. Then

$$\begin{aligned}
&\left|\Pr[f(X_1,\dots,X_{i-1},a) = X_i] - \frac{1}{2}\right| \\
&\quad= \left|\mathop{\mathbb{E}}_{Y_{[s]\setminus S_i}}\left[\Pr_{Y_{S_i}}[f(h(Y_{S_1}),\dots,h(Y_{S_{i-1}}),a) = h(Y_{S_i})]\right] - \frac{1}{2}\right| \\
&\quad\le \mathop{\mathbb{E}}_{Y_{[s]\setminus S_i}}\left[\left|\Pr_{Y_{S_i}}[f(h(Y_{S_1}),\dots,h(Y_{S_{i-1}}),a) = h(Y_{S_i})] - \frac{1}{2}\right|\right].
\end{aligned}$$

Consider any arbitrary fixing of $Y_{[s]\setminus S_i}$ and let $Z = Y_{S_i}$. For each $j < i$, since we fixed $Y_{[s]\setminus S_i}$ and $|S_i \cap S_j| \le k$, there is some k-junta ϕ_j such that $h(Y_{S_j}) = \phi_j(Z)$. Therefore,

$$\begin{aligned}
&\left|\Pr_{Y_{S_i}}[f(h(Y_{S_1}),\dots,h(Y_{S_{i-1}}),a) = h(Y_{S_i})] - \frac{1}{2}\right| \\
&\quad= \left|\Pr_{Z}[f(\phi_1(Z),\dots,\phi_{i-1}(Z),a) = h(Z)] - \frac{1}{2}\right| \\
&\quad\le \varepsilon,
\end{aligned}$$

because h is ε-hard for $f \circ \mathsf{JUNTA}_{r,k}$ (note that each bit of a can trivially be computed by a 0-junta.) □

To complete the proof of Theorem 4.2, we will relate the "predictor" model to the standard "distinguisher" model. We will show that if a distribution is unpredictable for f, then it also fools f, with a factor of n loss in the error parameter. This lemma is attributed to Yao.

Lemma 4.4 (Unpredictable $\implies$ Pseudorandom)**.** Let X be a distribution over $\{0,1\}^n$ and let $f\colon \{0,1\}^n \to \{0,1\}$. If X is ε-unpredictable for f, then X fools f with error $\varepsilon \cdot n$.

Proof. Let $R \sim U_n$ be independent of X. Define hybrid distributions $D_0, D_1, \dots, D_n$ by

$$D_i = X_1 X_2 \dots X_i R_{i+1} R_{i+2} \dots R_n,$$

so D_i consists of i pseudorandom bits followed by $n-i$ truly random bits. By the triangle inequality,

$$\begin{aligned}|\,\mathbb{E}[f(X)] - \mathbb{E}[f]| &= |\,\mathbb{E}[f(D_n)] - \mathbb{E}[f(D_0)]| \\ &\leq \sum_{i=1}^{n} |\,\mathbb{E}[f(D_i)] - \mathbb{E}[f(D_{i-1})]|.\end{aligned}$$

For each $i \in [n]$, we have

$$\mathbb{E}[f(D_i)] = \mathbb{E}[f(D_{i-1}) \mid R_i = X_i]$$

and

$$\mathbb{E}[f(D_{i-1})] = \frac{1}{2}\,\mathbb{E}[f(D_{i-1}) \mid R_i = X_i] + \frac{1}{2}\,\mathbb{E}[f(D_{i-1}) \mid R_i \neq X_i].$$

Therefore,

$$\begin{aligned}&|\,\mathbb{E}[f(D_i)] - \mathbb{E}[f(D_{i-1})]| \\ &\quad= \left|\frac{1}{2}\,\mathbb{E}[f(D_{i-1}) \mid R_i = X_i] - \frac{1}{2}\,\mathbb{E}[f(D_{i-1}) \mid R_i \neq X_i]\right| \\ &\quad= \left|\frac{1}{2}\,\mathbb{E}[f(D_{i-1}) \mid R_i = X_i] + \frac{1}{2}\,\mathbb{E}[\neg f(D_{i-1}) \mid R_i \neq X_i] - \frac{1}{2}\right| \\ &\quad= \left|\mathbb{E}[f(D_{i-1}) \oplus R_i \oplus X_i] - \frac{1}{2}\right| \\ &\quad\leq \mathop{\mathbb{E}}_{R}\left[\left|\mathop{\mathbb{E}}_{X}[f(D_{i-1}) \oplus R_i \oplus X_i] - \frac{1}{2}\right|\right].\end{aligned}$$

This is at most ε, because for any fixing of R, if we let $g(x) = f(x_1 \ldots x_{i-1} R_i \ldots R_n) \oplus R_i \oplus x_i$, then either g or $\neg g$ is testing whether f successfully predicts x_i given $x_1, \ldots, x_{i-1}$. Summing up, we get $|\,\mathbb{E}[f(X)] - \mathbb{E}[f]| \leq \varepsilon \cdot n$. □

4.2.3 A family of nearly disjoint sets

We have now shown how to construct a PRG given two ingredients: a hard function h and a family of nearly disjoint sets $S_1, \ldots, S_n$. The hard function must be tailored to the specific class of functions that we wish to fool, but constructing the family of nearly disjoint sets is a combinatorics problem that can be addressed separately. We will construct such a family using an argument by Erdős *et al.* [85].

Lemma 4.5 (Existence of nearly disjoint sets [85])**.** Let $k, r, n \in \mathbb{N}$ with $r > k \geq 1$. For a suitable value $s = O(n^{1/(k+1)} \cdot r^2/k)$, there exists a k-nearly disjoint family of r-subsets $S_1, \ldots, S_n \subseteq [s]$. Furthermore, given k, r, and n, the family can be deterministically constructed in time $\text{poly}(n, 2^s)$.

Proof. Construct $S_1, \ldots, S_n$ greedily. That is, for $i \in [n]$, having constructed $S_1, \ldots, S_{i-1}$, search exhaustively through all subsets of $[s]$ to find a set S_i of size r such that for every $j < i$, we have $|S_j \cap S_i| \leq k$. To prove that such a set exists, consider picking S_i uniformly at random from among all subsets of $[s]$ of size r. Then by the union bound,

$$\begin{aligned}\Pr_{S_i}[\exists j < i \text{ such that } |S_i \cap S_j| > k] &\leq \sum_{j=1}^{i-1} \Pr_{S_i}[|S_j \cap S_i| > k] \\ &= (i-1) \cdot \frac{\binom{r}{k+1} \cdot \binom{s-(k+1)}{r-(k+1)}}{\binom{s}{r}} \\ &< n \cdot \frac{\binom{r}{k+1}^2}{\binom{s}{k+1}},\end{aligned}$$

where the last step uses the identity $\binom{s}{r} \cdot \binom{r}{k+1} = \binom{s}{k+1} \cdot \binom{s-(k+1)}{r-(k+1)}$. Therefore, a suitable S_i is guaranteed to exist provided $n \leq \binom{s}{k+1}/\binom{r}{k+1}^2$. Choose the universe size to be $s = \lceil er^2 \cdot n^{1/(k+1)}/k \rceil$, because that way

$$n \leq \left(\frac{sk}{er^2}\right)^{k+1} < \frac{\binom{s}{k+1}}{\binom{r}{k+1}^2},$$

where the last step uses $0 < k < r \leq s$. □

Optimality Ignoring explicitness, what is the best possible universe size s in Lemma 4.5? Equivalently, given k, r, and s, what is the largest number n such that there exists a k-nearly disjoint family of r-subsets $S_1, \ldots, S_n \subseteq [s]$? This seems to be an open problem in combinatorics, even if we are only interested in rough asymptotics. The proof of Lemma 4.5 shows the existence of a family with $n \geq \binom{s}{k+1}/\binom{r}{k+1}^2$. Conversely, in every such family, each size-$(k+1)$ subset of $[s]$ is

contained in at most one S_i, so[3] $n \leq \binom{s}{k+1}/\binom{r}{k+1}$. This last bound can be slightly improved [94], [207], but it seems that there is still a significant gap.

Open Problem 4.1 (Optimal nearly disjoint sets)**.** For given values k, r, and s, determine (asymptotically) the maximum value n such that there exists a k-nearly disjoint family of r-subsets $S_1, \ldots, S_n \subseteq [s]$.

When k and r are constant, the problem has been solved: a famous theorem by Rödl [204] says that there exist families with $n = (1 - o(1)) \cdot \binom{s}{k+1}/\binom{r}{k+1}$. When k and r are growing parameters, however, the situation seems to be less clear. For example, when $r = k^2$ and $s = 100k^3$, the optimal value of n is somewhere between $2^{\Theta(k)}$ and $k^{\Theta(k)}$, but the true value seems to be unknown.

Efficiency The proof of Lemma 4.5 is simple, but it's somewhat unsatisfactory because of the exhaustive search. The universe size s corresponds to the seed length of the Nisan-Wigderson generator. The time complexity $\text{poly}(n, 2^s)$ in Lemma 4.5 is too high to get a strictly "explicit" PRG, except in the case $s = O(\log n)$, since our definition of explicitness (Definition 1.4) is that the runtime should be $\text{poly}(n)$. In the literature [115], [144], [183], there are several constructions of families of nearly disjoint sets that are "more explicit" than the construction of Lemma 4.5, but unfortunately, the parameters of these constructions are not quite as good.

Open Problem 4.2 (More efficient constructions of nearly disjoint sets)**.** Find a family of nearly disjoint sets that has the same parameters as Lemma 4.5 and that can be constructed in time $\text{poly}(n)$. (Assume $s < n$.)

4.2.4 Unconditional applications

To illustrate the Nisan-Wigderson framework, let us use the framework to design another PRG for $\mathbf{AC}^0$ circuits. Recall that the parity function

[3] In terms of universe size, we get $s \geq \Omega(n^{1/(k+1)} \cdot r)$. One can also prove $s \geq \Omega(\min\{r^2/k, nr\})$ using the inclusion-exclusion principle.

is hard for such circuits (Theorem 4.1). By plugging the parity function into the Nisan-Wigderson framework, we get a PRG with the following parameters.

Corollary 4.6 (Hardness-based PRG for $\mathbf{AC}^0$)**.** For any $n, m, d \in \mathbb{N}$ and $\varepsilon > 0$, there is an ε-PRG for depth-d size-m $\mathbf{AC}^0$ circuits on n input bits with seed length $s = \log^{2d+O(1)}(mn) \cdot \operatorname{polylog}(1/\varepsilon)$. The PRG can be computed in time $2^{O(s)}$.

Proof. Let $k = \lfloor \log n \rfloor$. By Theorem 4.1, for a suitable value

$$r = O(\log^d(mn) \cdot \log(n/\varepsilon)),$$

the parity function $h\colon \{0,1\}^r \to \{0,1\}$ is (ε/n)-hard for depth-$(d+1)$ size-$(m + n \cdot 2^{k+1})$ $\mathbf{AC}^0$ circuits. Let $S_1, \ldots, S_n \subseteq [s]$ be the k-nearly disjoint family of r-subsets from Lemma 4.5, and let G be the Nisan-Wigderson PRG given by Equation (4.3).

To prove the correctness of this PRG, let f be a depth-d size-m $\mathbf{AC}^0$ circuit on n input bits. For simplicity, we assume that the circuit alternates between layers of AND gates and OR gates. Without loss of generality, assume that the gates immediately above the inputs are OR gates.

Every k-junta can be computed by a DNF with 2^k terms. Therefore, every function in $f \circ \mathsf{JUNTA}_{r,k}$ can be computed by an $\mathbf{AC}^0$ circuit of depth $d+1$ and size $m + n \cdot 2^{k+1}$: replace each of the $2n$ literals in f with a DNF for the corresponding junta, and then merge the adjacent layers of OR gates. Therefore, h is (ε/n)-hard for $f \circ \mathsf{JUNTA}_{r,k}$, so Theorem 4.2 implies that G fools f with error ε. By Lemma 4.5, the seed length of our generator is $s = O(n^{1/(k+1)} \cdot r^2/k)$, which is $\log^{2d+O(1)}(mn) \cdot \log^2(1/\varepsilon)$ as claimed. □

Historically, the Nisan-Wigderson approach provided the first explicit PRG for constant-depth polynomial-size $\mathbf{AC}^0$ circuits with seed length $\operatorname{polylog} n$ [180]. The seed length in Theorem 4.6 is a little better than the seed length implied by Braverman's theorem (Section 2.6) in some cases, because the factor 2 in the exponent is better. Today, we can use other frameworks to construct PRGs for $\mathbf{AC}^0$ circuits with better seed lengths (see the discussion in Section 5.3.) However, the

Nisan-Wigderson framework remains a valuable, flexible approach for designing PRGs, especially for more powerful models of computation. For example, the Nisan-Wigderson framework has been used to construct unconditional PRGs for $\mathbf{AC}^0$ circuits augmented with a few gates that compute arbitrary threshold functions or symmetric functions [138], [158], [164], [208], [241]. Also, a line of work initiated by Trevisan [232] shows that there are connections between the Nisan-Wigderson framework and unconditional constructions of *randomness extractors* (see Definition 3.8).

4.3 Hardness-based PRGs beyond Nisan-Wigderson

In summary, the Nisan-Wigderson framework is a method for converting a hard function into a PRG. Starting from a function on r bits that is ε-hard for $f \circ \mathsf{JUNTA}_{r,k}$, we get a PRG for f with seed length $O(n^{1/(k+1)} \cdot r^2/k)$ and error $\varepsilon \cdot n$.

Many other PRG constructions, including those that we saw in Sections 2 and 3 and those that we will see in Section 5, are related to lower bounds in a less direct way. To design a PRG for a class $\mathcal{F}$, rather than using the mere fact that such-and-such lower bound holds, we can try to distill and develop the *insights* that were used to *prove* the lower bound. For example, as we saw in Section 2.6, Braverman's theorem relies on the LMN theorem (Theorem 2.22), which builds on Håstad's switching lemma [116], which was originally proven for the sake of proving lower bounds for $\mathbf{AC}^0$ [116]. As another example, the INW generator for ROBPs (Section 3.2) relies on the same "communication complexity" intuition that appears in the proof of optimal lower bounds for ROBPs (Proposition 4.3).

There are also methods other than the Nisan-Wigderson framework for generically converting hardness into randomness. These methods improve on the Nisan-Wigderson framework in fascinating and important ways. However, the known applications of these other methods are almost exclusively *conditional*. Since our focus is on unconditional PRGs, we will only briefly survey these other methods.

Optimizing the circuit-size blow-up Suppose we wish to design a PRG for size-n circuits of unbounded depth. The Nisan-Wigderson framework can produce such a PRG, given a function that is hard for circuits that are a little larger. Indeed, if f is a size-n circuit, then every function in $f \circ \mathsf{JUNTA}_{r,k}$ can be computed by a circuit of size $m = n \cdot 2^{O(k)}$. To avoid paying the severe $n^{1/(k+1)}$ penalty in the Nisan-Wigderson seed length, one can choose $k = \Theta(\log n)$, in which case $m = n^{1+\Theta(1)}$.

There is a line of work on improving the size complexity m, i.e., showing how to construct a PRG for size-n circuits given a function that is hard for (some type of) circuits of size $O(n)$ or even $(1+\alpha)n$ [60], [83], [115], [197].

Read-once models There is another "loss" in the Nisan-Wigderson reduction of a more qualitative nature. The Nisan-Wigderson framework is not well-suited for the important problem of fooling standard-order ROBPs. The reason is that every polynomial-size *read-many* branching program can be written in the form $f \circ \mathsf{JUNTA}_{r,1}$ where f is a polynomial-size standard-order ROBP. Read-many branching programs are vastly more powerful than ROBPs, and the Nisan-Wigderson framework does not give us any way to take advantage of the read-once condition.

Babai *et al.* [18] nevertheless designed an unconditional hardness-based PRG for standard-order ROBPs via a different reduction. (Their generator was superseded by superior PRGs such as Nisan's PRG [181] and the INW PRG [131], which we discussed in Section 3.2.)

The seed length compared to the domain size The seed length in the Nisan-Wigderson reduction is not ideal. Recall that a PRG with seed length s implies a hard function on approximately s bits (Proposition 4.2). Therefore, starting from a hard function on r bits, we can hope to construct a PRG with seed length roughly r, rather than the r^2 factor that appears in Lemma 4.5. Indeed, there is a line of work showing how to convert an appropriately-hard function on r bits into a PRG with seed length $O(r)$ or even $(1+\alpha)r$ [60], [83], [133], [213], [236].

Worst-case hardness assumptions Recall that in the Nisan-Wigderson framework, we rely on access to a function h that is hard on average, i.e., it is hard to compute h on even a $(1/2+\varepsilon)$-fraction of inputs. There is a line of work on constructing PRGs from *worst-case* hardness [60], [83], [133], [134], [213], [227], [236]. To highlight one example, Umans showed how to construct a PRG for size-n circuits given a function that cannot be computed (in the worst case) by circuits of size n^c for a suitable constant c [236]. If the hard function is on r bits, then the PRG has seed length $O(r)$, and given the truth table of the hard function and a seed, the PRG can be computed in time $2^{O(r)}$ [236].

Uniform hardness assumptions In the Nisan-Wigderson framework, we start with a function h on a finite domain that is hard for some concrete, "nonuniform" model of computation. There are also many known constructions of PRGs from *uniform* complexity-theoretic hardness assumptions such as $\mathbf{BPP} \neq \mathbf{EXP}$ [16], [41], [42], [59], [98], [135], [234].

The error parameter The Nisan-Wigderson reduction converts an ε-hard function into a PRG with error $\varepsilon \cdot n$. Fefferman *et al.* [87] have studied the problem of avoiding the factor-of-n blow-up.

One of the motivations for studying this problem is the challenge of designing better PRGs for $\mathbf{AC}^0[\oplus]$ circuits, i.e., $\mathbf{AC}^0$ circuits augmented with parity gates. For context, fairly strong lower bounds are known for this class. In particular, based on Razborov and Smolensky's work [198], [222], [223], one can show that for every $m, d \in \mathbb{N}$ and $\varepsilon > 0$, there is a value $r = \varepsilon^{-2} \cdot O(\log m)^{d-1}$ such that the majority function on r bits is ε-hard for depth-d size-m $\mathbf{AC}^0[\oplus]$ circuits, provided $r \leq m$ [90], [126]. Therefore, even if we treat the lack-of-lower-bounds barrier as a real barrier, we can hope to design PRGs for these circuits with polylogarithmic seed length (at least in the constant-error regime). So far, however, the best fully-explicit PRGs for these circuits have much larger seed length, very close to the trivial seed length of n bits [87].

Open Problem 4.3 (PRGs for $\mathbf{AC}^0[\oplus]$)**.** Design an explicit PRG that fools constant-depth polynomial-size $\mathbf{AC}^0[\oplus]$ circuits on n input variables with seed length $o(n)$.

Note that if we severely relax our standards of "explicitness," then there are known constructions of PRGs for $\mathbf{AC}^0[\oplus]$ with seed length $n^{o(1)}$ [57], [58].

Cryptographic PRGs The Nisan-Wigderson generator is unsuitable for cryptography, because computing the generator involves evaluating the "hard function" h. A cryptographic PRG cannot afford to evaluate a function that is hard for the adversary to compute, because a cryptographic PRG must fool all efficient adversaries – including those that use polynomially more time than the PRG itself uses.

Nevertheless, the paradigm of using some type of "hard function" to construct a PRG has been highly successful in the cryptographic setting [5], [27], [28], [77], [84], [95], [96], [99], [101], [110]–[112], [119], [122], [153], [170], [214], [237], [249]–[251]. Indeed, hardness-based cryptographic PRGs predate the Nisan-Wigderson framework. See, for example, Goldreich's expository works [96], [97] for an introduction.

5

Random Restrictions

In this section, we will present several constructions of PRGs based on *random restrictions*. A *restriction* is a string $R \in \{0,1,\star\}^n$. Intuitively, when $R_i = \star$, the interpretation is that R does not assign any value to the i-th variable. A restriction R can be *applied* to a function f via the following two definitions.

Definition 5.1 (Composition of restrictions)**.** If $R \in \{0,1,\star\}^n$ and $x \in \{0,1\}^n$, define the *composition* $R \circ x \in \{0,1,\star\}^n$ by

$$(R \circ x)_i = \begin{cases} R_i & \text{if } R_i \in \{0,1\} \\ x_i & \text{if } R_i = \star. \end{cases} \tag{5.1}$$

More generally, the same formula holds when x is an element of $\{0,1,\star\}^n$ rather than $\{0,1\}^n$, so $\circ$ is an associative binary operation on the space $\{0,1,\star\}^n$.

Definition 5.2 (Applying a restriction to a function)**.** Let f be a function on $\{0,1\}^n$, and let $R \in \{0,1,\star\}^n$ be a restriction. Then the *restricted function* $f|_R$ is the function on $\{0,1\}^n$ defined by

$$f|_R(x) = f(R \circ x).$$

Remark 5.1 (Order of restriction composition)**.** Some sources define the composition operator $\circ$ the other way around, so $f|_R(x) = f(x \circ R)$ rather than $f(R \circ x)$. Both conventions are reasonable. The motivation behind our convention is that one can identify a restriction $R \in \{0, 1, \star\}^n$ with the unique function $R\colon \{0,1\}^n \to \{0,1\}^n$ such that for every f and x, we have $f|_R(x) = f(R(x))$ (see Figure 5.1). Under this identification, the restriction composition operator $\circ$ is literally function composition.

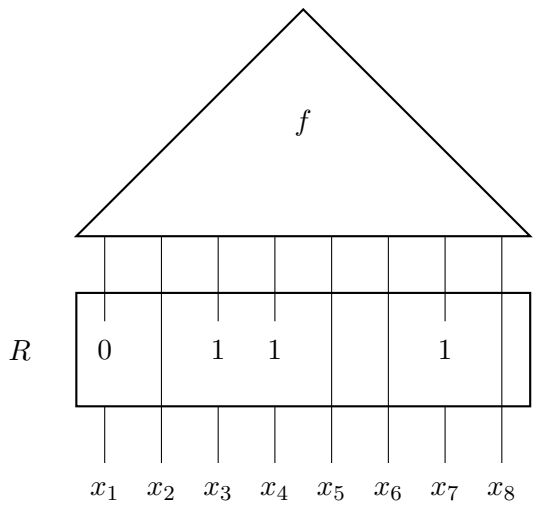

Figure 5.1: The restricted function $f|_R$ in the case $R = \{0, \star, 1, 1, \star, \star, 1, \star\}$. We still think of $f|_R$ as a function on the 8-variable domain $\{0,1\}^8$, but its output only depends on the values of four of those variables.

We will often consider *random* restrictions as defined below.

Definition 5.3 (Truly random restrictions)**.** For $p > 0$, let $\mathcal{R}_p$ denote a truly random restriction over $\{0, 1, \star\}^n$ with $\star$-probability p, i.e., the coordinates are independent, and each coordinate is

$$\begin{cases} \star & \text{with probability } p \\ 0 & \text{with probability } (1-p)/2 \\ 1 & \text{with probability } (1-p)/2. \end{cases}$$

(The parameter n will be clear from context.)

Random restrictions have been used in many areas of the theory of computing, perhaps starting with Subbotovskaya's pioneering work on De Morgan formulas [226]. The process of designing a PRG for a class $\mathcal{F}$ using restrictions can be divided into two main steps.

1. Prove a lemma that says that functions in $\mathcal{F}$ *simplify* in some sense under restrictions. We will refer to such a lemma as a "simplification-under-restrictions lemma." This first step requires an intimate understanding of the specific class $\mathcal{F}$.

2. Apply a generic *reduction* that says how to construct a PRG for any class satisfying a suitable simplification-under-restrictions lemma (and perhaps also satisfying some mild conditions such as closure properties). This second step is mainly about the abstract logic of restrictions and PRGs.

As we will see, this plan can be instantiated in multiple ways. There are *several different types* of simplification-under-restrictions lemmas:

- Are we considering truly random restrictions, or pseudorandom restrictions, or "partially pseudorandom" restrictions?
- What is our measure of "simplicity?"
- Does simplification occur with high probability, or does it merely occur "on average?"

Correspondingly, there are *several distinct reductions* from the problem of constructing PRGs to the problem of proving simplification under restrictions. We will discuss the *polarizing random walks* framework (Sections 5.1 and 5.2), the *iterated restrictions* paradigm (Sections 5.3 to 5.5), and the *Impagliazzo-Meka-Zuckerman* framework (Section 5.6). Perhaps these variations make the topic a bit confusing, but on the bright side, all this flexibility means that we have a rich toolkit for constructing PRGs.

5.1 PRGs from Polarizing Random Walks

In this section, we present our first reduction showing that if a class simplifies under random restrictions, then we get a PRG for that class.

This first reduction, based on "polarizing random walks," was introduced relatively recently by Chattopadhyay *et al.* [49]. It has the benefit that its assumptions are quite minimal, i.e., it is applicable in a relatively broad set of circumstances.

5.1.1 Simplification under truly random restrictions

Let $\mathcal{F}$ be a class of functions $f\colon \{0,1\}^n \to \{0,1\}$ that we wish to fool. Suppose that we have shown that functions in our class $\mathcal{F}$ simplify under random restrictions. Specifically, suppose we have identified a class $\mathcal{F}_{\text{simp}}$ of "simpler" functions and values $p, \delta > 0$ such that for each $f \in \mathcal{F}$, we have

$$\Pr[f|_{\mathcal{R}_p} \in \mathcal{F}_{\text{simp}}] \geq 1 - \delta.$$

For example, Rossman proved the following theorem [205] using Håstad's famous switching lemma [116] and his more recent "multi-switching" lemma [118].

Theorem 5.1 ($\mathbf{AC}^0$ simplifies under restrictions [205])**.** For every $n, m, d \in \mathbb{N}$, there is a value $p = 1/\Theta(\log m)^{d-1}$ such that if $f\colon \{0,1\}^n \to \{0,1\}$ is computable by a depth-d size-m $\mathbf{AC}^0$ circuit and $\delta > 0$, then with probability $1 - \delta$, the restricted function $f|_{\mathcal{R}_p}$ can be computed by a decision tree of depth at most $\log(1/\delta)$.

For this section, we will consider a class "simple" *if we can fool it with a short seed.* For example, we consider small-depth decision trees to be simple, because we can ε-fool decision trees of depth $\log(1/\delta)$ using a seed of length $O(\log(1/\delta)+\log(1/\varepsilon)+\log\log n)$ (see Section 2.3.3). In general, assuming that functions in $\mathcal{F}$ simplify to $\mathcal{F}_{\text{simp}}$ under restrictions, and assuming that we have a good PRG for $\mathcal{F}_{\text{simp}}$, how can we design a PRG for the original class $\mathcal{F}$? Our strategy will be to first design a relaxation of a PRG called a *fractional PRG*. Then, we will gradually transform the fractional PRG into a genuine PRG.

5.1.2 Fractional PRGs

For the purposes of this approach, it will be convenient to work with Boolean functions $f\colon \{-1,1\}^n \to \mathbb{R}$ instead of our usual domain $\{0,1\}^n$.

Recall that a PRG for a family of Boolean functions is a function $G\colon \{0,1\}^s \to \{-1,1\}^n$ such that $\mathbb{E}[f(G(U_s))] \approx \mathbb{E}[f]$ for every function f in the family. A *fractional PRG* is a relaxation of a PRG where we allow G to take values in the solid hypercube $[-1,1]^n$ as opposed to $\{-1,1\}^n$. For this to make sense, we would like to be able to "evaluate" $f : \{-1,1\}^n \to \mathbb{R}$ on arbitrary inputs from $[-1,1]^n$. One natural way to do this is by considering the multilinear extension of f. Recall that every Boolean function $f : \{-1,1\}^n \to \mathbb{R}$ has a unique Fourier expansion,

$$f(x) = \sum_{S \subseteq [n]} \widehat{f}(S) \prod_{i \in S} x_i.$$

The formula above can be evaluated at an arbitrary point $x \in \mathbb{R}^n$, which allows us to extend f to a multilinear polynomial $f\colon \mathbb{R}^n \to \mathbb{R}$. To understand how to interpret the value $f(x)$ for fractional vectors $x \in [-1,1]^n$, we make the following definition.

Definition 5.4 (Product distribution notation)**.** For $x \in [-1,1]^n$, let Π_x be the unique product distribution over $\{-1,1\}^n$ satisfying $\mathbb{E}[\Pi_x] = x$.

Fact 5.1 (Evaluation at fractional points)**.** Let $f\colon \{-1,1\}^n \to \mathbb{R}$ be any function. Extend f to the domain $[-1,1]^n$ via the Fourier expansion. Then for every $x \in [-1,1]^n$, we have

$$f(x) = \mathbb{E}[f(\Pi_x)].$$

More generally, for every product distribution X over $[-1,1]^n$, we have $\mathbb{E}[f(X)] = f(\mathbb{E}[X])$.

Proof. This is immediate from multilinearity and linearity of expectation. □

As a result, if f takes values in $\{-1,1\}$, then its multilinear extension is a map $[-1,1]^n \to [-1,1]$. Another useful corollary is that $f(0^n) = \mathbb{E}[f]$. Thus, a PRG can be seen as a method of sampling a distribution X over $\{-1,1\}^n$ such that $\mathbb{E}[f(X)] \approx f(0^n)$. We will define a fractional PRG by allowing the pseudorandom distribution to take values inside the cube.

Definition 5.5 (Fractional PRGs)**.** Let $f\colon \{-1,1\}^n \to \mathbb{R}$, and extend f to the domain $[-1,1]^n$ via the Fourier expansion. A random variable $X \in [-1,1]^n$ is said to *ε-fool f*, or *fool f with error ε*, if

$$|\,\mathbb{E}[f(X)] - f(0^n)| \le \varepsilon.$$

We say that X fools a family $\mathcal{F}$ of Boolean functions with error ε if X fools (the multilinear extension of) each function in $\mathcal{F}$ with error ε. A *fractional ε-PRG* for $\mathcal{F}$ is a function $G\colon \{0,1\}^s \to [-1,1]^n$ such that $G(U_s)$ fools $\mathcal{F}$ with error ε.

We can trivially fool *all* Boolean functions with error 0 and seed length 0 by simply outputting 0^n. However, our main motivation for defining fractional PRGs is as a means to constructing true PRGs, and our PRG construction will require some non-triviality conditions from the fractional PRG. In particular, we will require each coordinate of the fractional PRG to have variance bounded away from zero.

Definition 5.6 (Noticeability)**.** We say that a random variable $X \in [-1,1]^n$ is q-noticeable for a parameter $q \ge 0$, if for every $i \in [n]$, $\mathbb{E}[X_i^2] \ge q$. We say that a fractional PRG $G\colon \{0,1\}^s \to [-1,1]^n$ is q-noticeable if $G(U_s)$ is q-noticeable.

The following lemma shows that if a class $\mathcal{F}$ simplifies to another class $\mathcal{F}_{\mathrm{simp}}$ under restrictions, and we have a good PRG for $\mathcal{F}_{\mathrm{simp}}$, then we get a good *fractional* PRG for the original class $\mathcal{F}$, where the noticeability depends on the $\star$-probability of the restrictions.

Lemma 5.2 (Simplification implies fractional PRGs)**.** Let $\mathcal{F}$ and $\mathcal{F}_{\mathrm{simp}}$ be classes of functions $f\colon \{-1,1\}^n \to \{-1,1\}$. Let $p, \delta > 0$, and suppose that for each $f \in \mathcal{F}$, we have

$$\Pr[f|_{\mathcal{R}_p} \in \mathcal{F}_{\mathrm{simp}}] \ge 1 - \delta.$$

Let X be a distribution over $\{-1,1\}^n$ that ε-fools $\mathcal{F}_{\mathrm{simp}}$. Then pX is (p^2)-noticeable and fools $\mathcal{F}$ with error $\varepsilon + 2\delta$.

Proof. Clearly, we always have $(pX)_i^2 = p^2$, showing that pX is (p^2)-noticeable. Now fix $f \in \mathcal{F}$, and sample $R \sim \mathcal{R}_p$ independently of X.

For each fixed string $x \in \{\pm 1\}^n$, the composition $R \circ x$ is a product distribution over $\{\pm 1\}^n$, where

$$\mathbb{E}[(R \circ x)_i] = (1-p) \cdot 0 + p \cdot x_i = p \cdot x_i.$$

Therefore, by Fact 5.1, we have $\mathbb{E}[f(R \circ x)] = f(px)$. Consequently, $\mathbb{E}_X[f(pX)] = \mathbb{E}_{R,X}[f(R \circ X)]$. Clearly, $|\mathbb{E}_{R,X}[f(R \circ X)] - \mathbb{E}[f]| \leq \varepsilon + 2\delta$. □

By combining Lemma 5.2 and Theorem 5.1, we get a fractional PRG for $\mathbf{AC}^0$ with the following parameters.

Corollary 5.3 (Fractional PRGs for $\mathbf{AC}^0$)**.** For every $n, m, d \in \mathbb{N}$ and every $\varepsilon > 0$, there is an explicit q-noticeable fractional PRG that ε-fools depth-d size-m $\mathbf{AC}^0$ circuits with seed length $O(\log(1/\varepsilon) + \log\log n)$, where $q = 1/\Theta(\log m)^{2d-2}$.

5.1.3 From fractional PRGs to PRGs

In this section we will prove that fractional PRGs can be used to construct PRGs with small seed length as long as the fractional PRG has two useful properties: it has a small seed length and all its coordinates have noticeable variance.

Theorem 5.4 (Fractional PRG $\implies$ Standard PRG [49])**.** Suppose that $\mathcal{F}$ is a family of functions $f\colon \{-1,1\}^n \to \{-1,1\}$ that is closed under restrictions and shifts.[1] Assume that there exists an explicit q-noticeable fractional PRG for $\mathcal{F}$ with error ε and seed length s. Then there exists an explicit PRG for $\mathcal{F}$ with seed length $O(s \cdot q^{-1} \cdot \log(n/\varepsilon))$ and error $O(\varepsilon \cdot q^{-1} \cdot \log(n/\varepsilon))$.

For example, by combining Theorem 5.4 and Corollary 5.3, we get the following PRG for $\mathbf{AC}^0$.

Corollary 5.5 (PRG for $\mathbf{AC}^0$ based on simplification under truly random restrictions)**.** For every $n, m, d \in \mathbb{N}$ and $\varepsilon > 0$, there is an explicit ε-PRG for depth-d size-m $\mathbf{AC}^0$ circuits on n input bits with seed length

$$\widetilde{O}(\log m)^{2d-2} \cdot \widetilde{O}(\log(n/\varepsilon) \cdot \log(1/\varepsilon)).$$

[1] For functions on $\{\pm 1\}^n$, "closure under shifts" means that for every $f \in \mathcal{F}$ and every $y \in \{\pm 1\}^n$, the function $g(x) = f(x_1 y_1, x_2 y_2, \ldots, x_n y_n)$ is in $\mathcal{F}$.

The seed length in Theorem 5.5 is slightly better than the other PRGs for $\mathbf{AC}^0$ circuits that we have already seen (Braverman's theorem in Section 2.6, and the Nisan-Wigderson generator in Section 4.2). More importantly, this new PRG generalizes in some ways that the previously discussed PRGs do not. After all, the same approach works whenever a class simplifies under random restrictions. In fact, as we will discuss in Section 5.2, it even works in the more general setting of a class that simplifies "on average" under random restrictions.

Random walks

To prove Theorem 5.4, we will take a *random walk* through the solid hypercube $[-1,1]^n$. It is natural to take $Y^{(0)} = 0^n$ as the starting point, since $\mathbb{E}[f] = f(0^n)$. Our goal then is to define a random walk that converges quickly to the Boolean cube $\{-1,1\}^n$, while each step of the walk does not incur much error. We will define the steps by independent samples from the output distribution X of the fractional PRG.

To this end let $X^{(1)}, \ldots, X^{(t)}$ be t independent samples of X where t is to be determined later. A natural first step for the random walk is $Y^{(1)} = Y^{(0)} + X^{(1)}$, as it has the two useful properties:

1. $|\mathbb{E}[f(Y^{(1)})] - \mathbb{E}[f(Y^{(0)})]| \leq \varepsilon$, and
2. Each coordinate of $Y^{(1)}$ is likely closer to $\{-1,1\}$ due to p-noticeability.

It is tempting to continue this approach for all steps and in particular define $Y^{(j)} = Y^{(j-1)} + X^{(j)}$. This does not work, since we may already step out of the $[-1,1]^n$ cube, and in fact after some steps start getting farther and farther from $\{-1,1\}^n$. A slight modification that works is to normalize coordinates according to their distance from $\{-1,1\}^n$.

For two vectors $x, x' \in [-1,1]^n$, define $x \odot x' \in [-1,1]^n$ to be their coordinate-wise product. Moreover, for every vector $y \in [-1,1]^n$ define $\delta_y \in [0,1]^n$ to be the vector with i-th coordinate $(\delta_y)_i = 1 - |y_i|$, i.e., $(\delta_y)_i$ is the distance from $y_i \in [-1,1]$ to the Boolean endpoints $\{-1,1\}$. The vector δ_y defines dimensions of the largest subcube inside $[-1,1]^n$ centered at y. Using this notation, we can now define the random walk:

- $Y^{(0)} = 0^n$, and
- For $j > 0$, let $Y^{(j)} = Y^{(j-1)} + \delta_{Y^{(j-1)}} \odot X^{(j)}$.

We will show that this random walk quickly gets close to $\{-1, 1\}^n$. Still, there is a chance that the coordinates of $Y^{(t)}$ are never *exactly* integers. The final construction takes care of this by outputting the coordinate-wise signs of $Y^{(t)}$. To this end, for $x \in \mathbb{R}^n$ define $\mathrm{sign}(x) \in \{-1, 1\}^n$ to be the vector with i-th coordinate $\mathrm{sign}(x)_i = 1 \iff x_i > 0$.

The Generator G:

1. Let $X_1, \ldots, X_t$ be independent copies of X for a suitable value $t = O(q^{-1} \cdot \log(n/\varepsilon))$
2. Let $Y^{(0)} = 0^n$, and for $j > 0$ define
$$Y^{(j)} = Y^{(j-1)} + \delta_{Y^{(j-1)}} \odot X^{(j)}$$
3. Output $\mathrm{sign}(Y^{(t)})$

Analysis of the random walk

To prove the correctness of the generator G, we will prove that the random walk has three properties:

(a) Each step introduces little error: For every $f \in \mathcal{F}$ and $j \in [t]$, we have
$$\left|\mathbb{E}\left[f(Y^{(j)})\right] - \mathbb{E}\left[f(Y^{(j+1)})\right]\right| \leq \varepsilon.$$

(b) The walk *polarizes* with high probability:
$$\Pr[\|\delta_{Y^{(t)}}\|_\infty \leq \varepsilon/n] \geq 1 - \varepsilon.$$

(c) The final rounding operation introduces little error: For every $f \in \mathcal{F}$, conditioned on polarization, $|f(Y^{(t)}) - f(\mathrm{sign}(Y^{(t)}))| \leq \varepsilon$.

We prove these properties in the next three lemmas.

Lemma 5.6 (Steps incur small error)**.** Let $\mathcal{F}$ be a family of functions $f\colon \{-1,1\}^n \to \mathbb{R}$ that is closed under restrictions, and suppose $X \in [-1,1]^n$ fools $\mathcal{F}$ with error ε. Then for every $f \in \mathcal{F}$ and $y \in [-1,1]^n$,

$$|f(y) - \mathbb{E}[f(y + \delta_y \odot X)]| \le \varepsilon.$$

In particular, for every $j \in [t]$,

$$\left|\mathbb{E}\left[f(Y^{(j-1)})\right] - \mathbb{E}\left[f(Y^{(j)})\right]\right| \le \varepsilon.$$

Proof. Let $y \in [-1,1]^n$ be fixed. Sample a restriction $R \in \{-1,1,\star\}^n$, independent of X, where the coordinates of R are independent and distributed as follows:

$$R_i = \begin{cases} \operatorname{sign}(y_i) & \text{with probability } |y_i| \\ \star & \text{with probability } 1 - |y_i|. \end{cases}$$

We can extend the composition operation $R \circ x$ to the case of a vector $x \in [-1,1]^n$ in the natural way: we use x to fill in the $\star$ coordinates of R (see Equation (5.1)). That way, for each coordinate $i \in [n]$, we have

$$\mathop{\mathbb{E}}_{R}[(R \circ x)_i] = |y_i| \cdot \operatorname{sign}(y_i) + (1 - |y_i|) \cdot x_i = y_i + (1 - |y_i|) \cdot x_i,$$

and hence overall,

$$\mathop{\mathbb{E}}_{R}[R \circ x] = y + \delta_y \odot x.$$

By Fact 5.1, it follows that

$$\mathop{\mathbb{E}}_{R}[f|_R(x)] = \mathop{\mathbb{E}}_{R}[f(R \circ x)] = f\left(\mathop{\mathbb{E}}_{R}[R \circ x]\right) = f(y + \delta_y \odot x).$$

Consequently,

$$\begin{aligned} \left|f(y) - \mathop{\mathbb{E}}_{X}[f(y + \delta_y \odot X)]\right| &= \left|\mathop{\mathbb{E}}_{R}[f|_R(0^n)] - \mathop{\mathbb{E}}_{R,X}[f|_R(X)]\right| \\ &\le \mathop{\mathbb{E}}_{R}\left[\left|f|_R(0^n) - \mathop{\mathbb{E}}_{X}[f|_R(X)]\right|\right] \\ &\le \varepsilon, \end{aligned}$$

where the last step uses the fact that $\mathcal{F}$ is closed under restriction, hence X fools $f|_R$ with error ε for every fixing of R. □

Next, we will show that the random walk above converges to $\{-1,1\}^n$ quickly. For this argument, we assume that X is both q-noticeable for a large enough $q > 0$ and *symmetric* as defined below.

Definition 5.7 (Symmetric random variables)**.** Let X be a random variable distributed over $[-1,1]^n$. We say that X is *symmetric* if for every $x \in [-1,1]^n$, we have $\Pr[X = x] = \Pr[X = -x]$.

We can justify the symmetry assumption as follows. Starting from an arbitrary q-noticeable fractional ε-PRG G_{frac} for $\mathcal{F}$ with seed length s, we can define a new q-noticeable fractional PRG with seed length $s+1$ by the formula

$$G'_{\text{frac}}(x,b) = (-1)^b \cdot G_{\text{frac}}(x).$$

The distribution $G'_{\text{frac}}(U_{s+1})$ is symmetric, and because $\mathcal{F}$ is closed under shifts, $G'_{\text{frac}}(U_{s+1})$ still fools $\mathcal{F}$ with error ε. (This is the only place where we use the assumption that $\mathcal{F}$ is closed under shifts.)

The symmetry assumption is helpful because of the following lemma concerning the case $n = 1$.

Lemma 5.7. Let $X \in [-1,1]$ be a symmetric q-noticeable random variable. Then

$$\mathbb{E}\left[\sqrt{1-X}\right] \leq 1 - q/8.$$

In their original paper, Chattopadhyay *et al.* [49] observed that Lemma 5.7 follows immediately from the Taylor expansion of the function $\sqrt{1-x}$. We present an alternative argument below.

Proof of Lemma 5.7. Let $Y = |X|$, and sample $Z \in \{\pm 1\}$ independently of X. Then the product YZ is distributed identically to X. Furthermore, for each fixed value $y \in [0,1]$, we have

$$\left(\mathbb{E}\left[\sqrt{1-yZ}\right]\right)^2 = \left(\frac{\sqrt{1-y}+\sqrt{1+y}}{2}\right)^2 = \frac{1+\sqrt{1-y^2}}{2} \leq 1 - \frac{y^2}{4}.$$

Therefore,

$$\mathbb{E}\left[\sqrt{1-X}\right] = \mathop{\mathbb{E}}_{Y}\left[\mathop{\mathbb{E}}_{Z}\left[\sqrt{1-YZ}\right]\right] \leq \mathop{\mathbb{E}}_{Y}\left[\sqrt{1-Y^2/4}\right] \leq \mathop{\mathbb{E}}_{Y}[1 - Y^2/8]$$
$$\leq 1 - q/8. \qquad \square$$

Next, let us use Lemma 5.7 to show that coordinate-wise polarization happens with high probability. Indeed, looking ahead, the probability will be high enough to allow a union bound over all coordinates.

Lemma 5.8 (Polarization)**.** Let $A^{(1)}, \ldots, A^{(t)} \in [-1, 1]$ be independent symmetric q-noticeable random variables. Define $B^{(0)} = 0$, and for $j > 0$ define

$$B^{(j)} = B^{(j-1)} + (1 - |B^{(j-1)}|) \cdot A^{(j)}. \tag{5.2}$$

Then $\Pr[1 - |B^{(t)}| \geq e^{-tq/8}] \leq e^{-tq/16}$.

Proof. What happens to the distance $1-|B^{(\cdot)}|$ when we apply the update rule given in Equation (5.2)? If $\operatorname{sign}(A^{(j)}) = \operatorname{sign}(B^{(j-1)})$ (the "good case"), the distance decreases by a factor of $1 - |A^{(j)}|$. If $\operatorname{sign}(A^{(j)}) \neq \operatorname{sign}(B^{(j-1)})$ (the "bad case"), the distance might increase, but at most it increases by a factor of $1 + |A^{(j)}|$. Either way, for $j > 0$, we have

$$1 - |B^{(j)}| \leq (1 - |B^{(j-1)}|) \cdot (1 - A^{(j)} \cdot \operatorname{sign}(B^{(j-1)})).$$

We have assumed that $A^{(1)}, \ldots, A^{(j-1)}$ are symmetric. It follows that $B^{(j-1)}$ is also symmetric. Therefore, $|B^{(j-1)}|$ and $A^{(j)} \cdot \operatorname{sign}(B^{(j-1)})$ are independent. As a consequence,

$$\mathbb{E}\left[\sqrt{1 - |B^{(j)}|}\right] \leq \mathbb{E}\left[\sqrt{1 - |B^{(j-1)}|}\right] \cdot \mathbb{E}\left[\sqrt{1 - A^{(j)} \cdot \operatorname{sign}(B^{(j-1)})}\right].$$

The random variable $A^{(j)} \cdot \operatorname{sign}(B^{(j-1)})$ is symmetric and q-noticeable, so we may apply Lemma 5.7, giving us

$$\mathbb{E}\left[\sqrt{1 - |B^{(j)}|}\right] \leq \mathbb{E}\left[\sqrt{1 - |B^{(j-1)}|}\right] \cdot (1 - q/8).$$

By induction, this implies that

$$\mathbb{E}\left[\sqrt{1 - |B^{(t)}|}\right] \leq (1 - q/8)^t \leq e^{-qt/8}.$$

The lemma follows by Markov's inequality. □

Now we show that the final rounding step does not introduce too much error.

Lemma 5.9 (Rounding Error)**.** Let $f\colon \{-1,1\}^n \to \{-1,1\}$ be a function, and extend it to the domain $[-1,1]^n$ via the Fourier expansion. For every $y \in [-1,1]^n$,

$$|f(y) - f(\operatorname{sign}(y))| \le \sum_{i=1}^{n}(1-|y_i|) \le n \cdot \|\delta_y\|_\infty.$$

Proof. We have

$$\begin{aligned} |f(y) - f(\operatorname{sign}(y))| &= |\,\mathbb{E}[f(\Pi_y)] - f(\operatorname{sign}(y))| && \text{(Fact 5.1)} \\ &\le 2 \cdot \Pr[\Pi_y \neq \operatorname{sign}(y)] && \text{(since } \|f\|_\infty \le 1) \\ &\le 2 \cdot \sum_{i=1}^{n} \frac{1-|y_i|}{2}, \end{aligned}$$

where the final inequality follows by the union bound and the observation that the marginal distributions of Π_y are given by

$$(\Pi_y)_i = \begin{cases} \operatorname{sign}(y_i) & \text{with probability } \frac{1+|y_i|}{2} \\ -\operatorname{sign}(y_i) & \text{with probability } \frac{1-|y_i|}{2}. \end{cases}$$

□

We can now analyze G and complete the proof of Theorem 5.4. The output of the generator G is $\operatorname{sign}(Y^{(t)})$ for $t = 16\log(n/\varepsilon)/q$. The seed for G is determined by t independent samples from the fractional generator, and hence has seed-length $ts = O(s\log(n/\varepsilon)/q)$. Then, we can bound the error of the generator $\operatorname{sign}(Y^{(t)})$ as follows:

$$\begin{aligned} &|\,\mathbb{E}[f] - \mathbb{E}[f(\operatorname{sign}(Y^{(t)}))]| \\ &\quad\le |\,\mathbb{E}[f(\operatorname{sign}(Y^{(t)}))] - \mathbb{E}[f(Y^{(t)})]| + \sum_{j=1}^{t} |\,\mathbb{E}[f(Y^{(j)})] - \mathbb{E}[f(Y^{(j-1)})]| \\ &\quad\le |\,\mathbb{E}[f(\operatorname{sign}(Y^{(t)})) - f(Y^{(t)})]| + \varepsilon t \end{aligned}$$

by Lemma 5.6. Now let E denote the event that $\|\delta_{Y^{(t)}}\|_\infty \le e^{-tq/8}$ and note that $e^{-tq/8} \le \varepsilon/n$. Then

$$\begin{aligned} &|\,\mathbb{E}[f(\operatorname{sign}(Y^{(t)})) - f(Y^{(t)})]| + \varepsilon t \\ &\quad\le |\,\mathbb{E}[f(\operatorname{sign}(Y^{(t)})) - f(Y^{(t)}) \mid \mathrm{E}]| + 2\Pr[\mathrm{E}] + \varepsilon t \\ &\quad\le |\,\mathbb{E}[f(\operatorname{sign}(Y^{(t)})) - f(Y^{(t)}) \mid \mathrm{E}]| + 2n \cdot e^{-tq/16} + \varepsilon t \\ &\quad\le |\,\mathbb{E}[f(\operatorname{sign}(Y^{(t)})) - f(Y^{(t)}) \mid \mathrm{E}]| + \varepsilon(t+2) \end{aligned}$$

by Lemma 5.8. Finally, by Lemma 5.9, we get a final error bound of

$$(t+3)\varepsilon \leq O(\varepsilon \log(n/\varepsilon)/q).$$

5.1.4 A better reduction for the low-error regime

In this section, we present a more refined reduction, showing how to convert a fractional PRG into a standard PRG with slightly better parameters than what was achieved by Chattopadhyay *et al.* [49] (Theorem 5.4).

Theorem 5.10 (Fractional PRG $\implies$ Standard PRG, refined version)**.** Suppose that $\mathcal{F}$ is a family of functions $f\colon \{-1,1\}^n \to \{-1,1\}$ that is closed under restrictions and shifts. Assume that there exists an explicit q-noticeable fractional PRG for $\mathcal{F}$ with error ε and seed length s. Then there exists a PRG for $\mathcal{F}$ with seed length

$$O((s+\log(1/\varepsilon)) \cdot q^{-1} \cdot \log n)$$

and error $O(\varepsilon \cdot q^{-1} \cdot \log n)$, computable in $\mathrm{poly}(n, 1/\varepsilon)$ time.

For comparison, recall that in the conclusion of Theorem 5.4, the seed length is $O(s \cdot q^{-1} \cdot \log(n/\varepsilon))$ and the error is $O(\varepsilon \cdot q^{-1} \cdot \log(n/\varepsilon))$. We typically have $s \geq \log(1/\varepsilon)$, so the parameters of Theorem 5.10 are superior in the regime $\varepsilon < n^{-\omega(1)}$. Admittedly, the PRG is not fully explicit in this regime (the time complexity is greater than $\mathrm{poly}(n)$), but the PRG's time complexity is always at most exponential in the seed length, which is a sufficient "explicitness" condition for many applications of PRGs.

The idea behind the improvement is to replace the trivial rounding step. Instead of taking $O(q^{-1} \cdot \log(n/\varepsilon))$ steps of the random walk and arguing that *all* the coordinates of $Y^{(t)}$ are well-polarized (i.e., close to $\{-1,1\}$), we will take only $O(q^{-1} \cdot \log n)$ steps of the random walk and argue that *most* of the coordinates of $Y^{(t)}$ are well-polarized. Then, we will show how to approximately sample from the mostly-polarized product distribution $\Pi_{Y^{(t)}}$.

More precisely, our notion of being "mostly polarized" is that when we sample from $\Pi_{Y^{(t)}}$, with high probability, we get a vector that only disagrees with $\mathrm{sign}(Y^{(t)})$ in a few coordinates:

Definition 5.8 (Polarization). Let $y \in [-1,1]^n$, let $k \in \mathbb{N}$, and let $\delta > 0$. We say that y is (k,δ)-*polarized* if

$$\Pr[\Delta(\Pi_y, \operatorname{sign}(y)) \le k] \ge 1 - \delta,$$

where Δ denotes Hamming distance.

We now show that $O(q^{-1} \cdot \log n)$ steps of the random walk suffice to achieve $(2k,\varepsilon)$-polarization where $k = O(\log(1/\varepsilon)/\log n)$. For this argument, we assume that the coordinates of the output distribution X of the fractional PRG are k-wise independent. To justify this assumption, observe that we can replace X with $X \odot X'$, where $X' \in \{\pm 1\}^n$ is a k-wise independent distribution. Since $\mathcal{F}$ is closed under shifts, this distribution still fools $\mathcal{F}$ with error ε. This modification only increases the seed length of the fractional PRG by an additive $O(k \log n) = O(\log(1/\varepsilon))$ bits; this is the reason for the $s + \log(1/\varepsilon)$ term in the conclusion of Theorem 5.10.

Lemma 5.11 (Polarization, refined version). Let $k = \lceil \log(1/\varepsilon)/\log n \rceil$ and assume that the coordinates of X are k-wise independent.[2] There exists a value $t = O(q^{-1} \log n)$ such that with probability $1 - \varepsilon$, the vector $Y^{(t)}$ is $(2k,\varepsilon)$-polarized.

Proof. Let J be the set of coordinates where $Y^{(t)}$ is "poorly polarized," namely

$$J = \left\{ i \in [n] : 1 - |Y_i^{(t)}| \ge \frac{1}{n^2} \right\}.$$

By Lemma 5.8, there is a choice of $t = O(q^{-1} \log n)$ such that for each coordinate $i \in [n]$, we have $\Pr[i \in J] \le 1/n^2$. Therefore, for any set $S \subseteq [n]$ with $|S| = k$, we have

$$\Pr[S \subseteq J] \le n^{-2k}.$$

Thus by a union bound,

$$\Pr[|J| \ge k] \le \binom{n}{k} \cdot n^{-2k} \le n^{-k} \le \varepsilon.$$

Now, fix any $y \in [-1,1]^n$ such that $|\{i \in [n] : 1 - |y_i| \ge 1/n^2\}| < k$. Every such y is $(2k,\varepsilon)$-polarized, because

$$\Pr[\Delta(\Pi_y, \operatorname{sign}(y)) \ge 2k] \le \binom{n-k}{k} \cdot \left(\frac{1}{2n^2}\right)^k \le n^{-k} \le \varepsilon. \qquad \square$$

[2] In fact, it suffices for the coordinates to be ε^2-almost k-wise independent.

Next, as outlined before, we show that when y is $(2k, \varepsilon)$-polarized, we can approximately sample from Π_y. We use a naïve "brute force" approach.

Lemma 5.12 (Approximately sampling from well-polarized product distributions)**.** Let $y \in [-1,1]^n$, let $k \in \mathbb{N}$, and let $\delta > 0$. There is a randomized algorithm Sample such that $\text{Sample}(y, k, \delta)$ outputs a string in $\{-1,1\}^n$, and if y is (k, δ)-polarized, then the output distribution $\text{Sample}(y, k, \delta)$ is within total variation distance $O(\delta)$ of Π_y. Furthermore, $\text{Sample}(y, k, \delta)$ runs in time $\text{poly}(n^k, 1/\delta)$ and it uses $O(k \log n + \log(1/\delta))$ truly random bits.

Proof. Let $z = \frac{1}{2}\delta_y \in [0, 1/2]^n$. Let D be the product distribution over $\{0,1\}^n$ such that $\mathbb{E}[D] = z$. Note that if we could sample $x \sim D$, then we could sample Π_y by outputting the vector with i-th coordinate $(-1)^{x_i} \cdot \text{sign}(y_i)$. Thus, it suffices to show how to efficiently sample a vector from $\{0,1\}^n$ with a distribution close to D in total variation distance. To achieve this, it is helpful to think of a perfect sample from D as being produced by the following process.

Perfectly sampling D:

1. Pick $\rho \in [0,1]$ uniformly at random
2. Initialize $\mu := 0$
3. For $e \in \{0,1\}^n$:
   - 3.1 $\mu := \mu + \prod_{i:e_i=0} z_i \prod_{i:e_i=1}(1 - z_i)$
   - 3.2 If $\mu \geq \rho$ then halt and output e
4. Output 0^n

To *efficiently* sample from D (approximately), we make two changes. First, in the "for loop," we only iterate over $e \in B_k$, where $B_k = \{e \in \{0,1\}^n : \sum_i e_i \leq k\}$. By the assumption of (k, δ)-polarization, this change only introduces total variation error at most δ. Second,

we discretize ρ. That is, we sample $\rho \in \{\alpha, 2\alpha, 3\alpha, \ldots, 1\}$ uniformly at random, where $\alpha = 2\delta/|B_k|$. The additional total variation error introduced by this second change is at most $\frac{1}{2} \cdot |B_k| \cdot \alpha \leq \delta$, because the probability of outputting any particular $e \in B_k$ is affected by at most α. □

Given Lemmas 5.11 and 5.12, it follows that the PRG below proves Theorem 5.10, assuming that X is symmetric and its coordinates are k-wise independent.

The Generator G':

1. Let $t = O(q^{-1} \cdot \log(n))$ and $k = \lceil \log(1/\varepsilon)/\log n \rceil$
2. Let $X^{(1)}, \ldots, X^{(t)}$ be independent copies of X
3. Let $Y^{(1)} = 0^n$, and for $j > 0$ define
$$Y^{(j)} = Y^{(j-1)} + \delta_{Y^{(j-1)}} \odot X^{(j)}$$
4. Output Sample$(Y^{(t)}, 2k, \varepsilon)$

5.2 Analysis Technique: Fourier Growth Bounds

Let $\mathcal{F}$ be a class of functions $f\colon \{0,1\}^n \to \{0,1\}$ that we wish to fool, and let $\mathcal{F}_{\text{simp}}$ be a class of "simpler" functions that we know how to fool. In the previous section, we considered the case that every $f \in \mathcal{F}$ simplifies to $\mathcal{F}_{\text{simp}}$ with high probability under random restrictions, i.e., for some $p, \delta > 0$, we have

$$\Pr[f|_{\mathcal{R}_p} \in \mathcal{F}_{\text{simp}}] \geq 1 - \delta. \tag{5.3}$$

We presented the "polarizing random walks" framework, which shows that under this assumption, we can construct a PRG for $\mathcal{F}$.

In this section, we consider a more general setting, which is when $\mathcal{F}$ "simplifies on average" under restrictions. We explain the meaning of this condition in Section 5.2.1. Then, in Section 5.2.2, we present an example

of this condition – we show that bounded-width regular ROBPs satisfy such an "average-case" simplification-under-restrictions lemma. Finally, in Section 5.2.3, we show that the polarizing random walks framework still works under this weaker assumption, and consequently we get a PRG for the model of bounded-width "arbitrary-order permutation ROBPs."

5.2.1 The noise operator and simplification on average

The notion of "simplification on average" is based on the *noise operator*.

Definition 5.9 (Noise operator)**.** Let $f\colon \{0,1\}^n \to \mathbb{R}$ and $p > 0$. We define $T_p f\colon \{0,1\}^n \to \mathbb{R}$ by the equation

$$T_p f(x) = \mathbb{E}[f|_{\mathcal{R}_p}(x)].$$

T_p is called the "noise operator" with parameter p, because for each bit of x, with probability $1-p$, we replace the bit with a fresh random bit ("noise"). Intuitively, $T_p f$ is a "smoothed out" version of f, and smaller values of p correspond to more smoothing out.

We say that f *simplifies on average* under the random restriction $\mathcal{R}_p$ if the function $T_p f$ lies in some "simpler" class $\mathcal{F}_{\text{simp}}$. For example, let $\mathcal{F}$ be the class of parity functions. When we apply a random restriction $\mathcal{R}_p$, with high probability, no meaningful simplification occurs: the restriction of a parity function is another parity function (or its complement). However, parity functions *do* drastically simplify *on average* over restrictions. Indeed, if f is a parity function on k bits, then $T_p f$ is approximated by the constant $1/2$ function to within pointwise error p^{-k}.

For a more interesting example, let us return to the model of *bounded-width regular ROBPs*, which we studied previously in Section 3.3. These programs can compute parity functions, so once again, they do not meaningfully simplify under a typical *individual* restriction. However, we will show that these programs simplify on average under restrictions. Specifically, we will show that $T_p f$ is fooled by almost k-wise uniform distributions, where p and k are suitable parameters and f is any bounded-width regular ROBP. Our approach for proving this

simplification-under-restrictions lemma is to bound the *Fourier growth* of f, discussed next.

5.2.2 Fourier growth bounds for regular ROBPs

Recall the Fourier L_1 bound from Section 2.3, which is a simple Fourier-analytic way of measuring the "complexity" of a Boolean function. Fourier growth is a more refined complexity measure which takes into account the *degree* of Fourier coefficients. Specifically:

Definition 5.10 (Functions with bounded Fourier growth)**.** For $a, b \geq 1$, denote by $\mathcal{L}_1^n(a,b)$ the family of all n-variate Boolean functions $f : \{-1,1\}^n \to \{-1,1\}$ that satisfy

$$\sum_{\substack{S \subseteq [n] \\ |S|=d}} \left|\widehat{f}(S)\right| \leq a \cdot b^d,$$

for every $d \in [n]$. Define $\mathcal{L}_1(a,b) = \bigcup_{n \in \mathbb{N}} \mathcal{L}_1^n(a,b)$.

Remark 5.2 (Fourier L_2 tail bounds)**.** One can similarly define $\mathcal{L}_2^n(a,b)$ to be the family of all n-variate Boolean functions $f : \{-1,1\}^n \to \{-1,1\}$ that satisfy

$$\sum_{\substack{S \subseteq [n] \\ |S|=d}} \left|\widehat{f}(S)\right|^2 \leq a \cdot 2^{-d/b},$$

for every $d \in [n]$. Tal showed that $\mathcal{L}_2(a,b) \subseteq \mathcal{L}_1(a, O(b))$ [228]. The simple example of the PARITY function (i.e., $\prod_{i \in [n]} x_i$) shows that the reverse is not true. In other words, having bounded L_1 Fourier growth is a weaker assumption than having bounded L_2 Fourier tails.

Reingold *et al.* [202] were the first to prove a Fourier growth bound for regular ROBPs. Later, building on their work and work by Chattopadhyay *et al.* [51], Lee *et al.* [149] improved the bound. In this section, we will present the proof of the latter bound.

Theorem 5.13 (Fourier growth of regular ROBPs [149])**.** If f is a width-w standard-order[3] regular ROBP, then $f \in \mathcal{L}_1(1, w-1)$. That is, for every $d \geq 0$,

[3]The theorem holds more generally for the "arbitrary-order" model, in which the

$$\sum_{|S|=d} |\widehat{f}(S)| \leq (w-1)^d.$$

After we are done proving Theorem 5.13, we will show that Fourier growth bounds imply simplification on average under restrictions.

Level 1 Fourier coefficients

The first step of the proof of Theorem 5.13 is to bound the level-1 Fourier coefficients of f. As a shorthand, we write $\widehat{f}(i)$ rather than $\widehat{f}(\{i\})$. We will prove the following.

Lemma 5.14 (Bound on level-1 Fourier coefficients of regular ROBPs)**.** Let f be a width-w length-n standard-order regular ROBP. Then

$$\sum_{i=1}^{n} |\widehat{f}(i)| \leq \mathbb{E}[f] \cdot (w-1).$$

Proof. Let m be the number of rejecting vertices in the final layer, i.e., $m = w - |V_{\text{accept}}|$. We will show by induction on n that

$$\sum_{i=1}^{n} |\widehat{f}(i)| \leq \mathbb{E}[f] \cdot m. \tag{5.4}$$

The lemma will follow, because $m \leq w$ (and if $m = w$, then $f \equiv 0$ and the lemma is trivial).

In the base case $n = 0$, Equation (5.4) is trivial, so assume that $n > 0$. Let $V_0, \ldots, V_n$ be the layers of f. Partition $V_{n-1} = R \cup S \cup T$ based on the number of accepting edges, i.e.,

$$\begin{aligned} R &= \{v \in V_{n-1} : \mathbb{E}[f_{v\to}] = 0\} \\ S &= \{v \in V_{n-1} : \mathbb{E}[f_{v\to}] = 1/2\} \\ T &= \{v \in V_{n-1} : \mathbb{E}[f_{v\to}] = 1\}. \end{aligned}$$

ROBP reads the variables in an arbitrary permuted order (note that we still assume the ROBP is oblivious). The reason is that the quantity $\sum_{|S|=d} |\widehat{f}(S)|$ is invariant under variable permutations.

Observe that $m = |R| + \frac{1}{2}|S|$ because f is regular. For each $i < n$, we have

$$\begin{aligned}
\widehat{f}(i) &= \mathop{\mathbb{E}}_{x \sim U_n}[f(x) \cdot (-1)^{x_i}] \\
&= \mathop{\mathbb{E}}_{x_1,\ldots,x_{n-1}}\left[(-1)^{x_i} \cdot \mathop{\mathbb{E}}_{x_n}[f(x)]\right] \\
&= \mathop{\mathbb{E}}_{x_1,\ldots,x_{n-1}}\left[(-1)^{x_i} \cdot \left(f_{\to T}(x) + \frac{1}{2} f_{\to S}(x)\right)\right] \\
&= \widehat{f_{\to T}}(i) + \frac{1}{2}\widehat{f_{\to S}}(i).
\end{aligned}$$

Therefore, if we write $p_{\to A}$ as a shorthand for the probability of visiting a vertex in $A \subseteq V_{n-1}$ (namely, $p_{\to A} = \mathbb{E}[f_{\to A}]$), then we have

$$\begin{aligned}
\sum_{i=1}^{n-1} |\widehat{f}(i)| &\le \frac{1}{2}\sum_{i=1}^{n-1} |\widehat{f_{\to T}}(i)| + \frac{1}{2}\sum_{i=1}^{n-1} |\widehat{f_{\to T}}(i) + \widehat{f_{\to S}}(i)| \\
&= \frac{1}{2}\sum_{i=1}^{n-1} |\widehat{f_{\to T}}(i)| + \frac{1}{2}\sum_{i=1}^{n-1} |\widehat{f_{\to S \cup T}}(i)| \\
&\le \frac{1}{2}|R \cup S| \cdot p_{\to T} + \frac{1}{2}|R| \cdot (p_{\to S \cup T}) \qquad \text{(Induction)} \\
&= m \cdot p_{\to T} + |R| \cdot \frac{p_{\to S}}{2}.
\end{aligned}$$

Meanwhile, at $i = n$, we have

$$\begin{aligned}
|\widehat{f}(n)| &\le \mathop{\mathbb{E}}_{x_1,\ldots,x_{n-1}}\left[\left|\mathop{\mathbb{E}}_{x_n}[(-1)^{x_n} \cdot f(x)]\right|\right] \\
&= \frac{p_{\to S}}{2} \le \frac{|S|}{2} \cdot \frac{p_{\to S}}{2},
\end{aligned}$$

because the regularity of f implies that $|S|$ is even. Therefore, overall,

$$\begin{aligned}
\sum_{i=1}^{n} |\widehat{f}(i)| &\le m \cdot p_{\to T} + \left(|R| + \frac{|S|}{2}\right) \cdot \frac{p_{\to S}}{2} \\
&= m \cdot \left(p_{\to T} + \frac{p_{\to S}}{2}\right) \\
&= m \cdot \mathbb{E}[f].
\end{aligned}$$

□

Higher-level Fourier coefficients

To bound the higher-level Fourier coefficients, we rely on a notion of *local monotonicity* [39], [51]. For every vertex v of an ROBP f, define $p_{v\to} = \mathbb{E}[f_{v\to}]$.

Definition 5.11 (Local Monotonicity)**.** Let f be a standard-order ROBP with layers $V_0, \ldots, V_n$. We say that f is *locally monotone* if for each $i \in [n]$ and each vertex $u \in V_{i-1}$, if we let (u, v) be the outgoing 0-edge and (u, s) be the outgoing 1-edge, then $p_{s\to} \geq p_{v\to}$.

It is easy to see that if f is a locally monotone ROBP, then for every i, we have $\widehat{f}(i) \leq 0$.[4] We observe that every ROBP can be "locally monotonized" by relabeling its edges, and so local monotonicity is a property of the labeling and not the structure of the graph.

Observation 5.1 (Local Monotonization [51])**.** Let f be a standard-order ROBP with layers $V_0, \ldots, V_n$. There exists a locally monotone standard-order ROBP f' obtained by relabeling edges of f. Furthermore, for every $i \in [n]$ and every $v \in V_{i-1}$,

$$\widehat{f'_{v\to}}(i) = -\left|\widehat{f_{v\to}}(i)\right|.$$

Proof. First order the vertices in each layer according to $p_{v\to}$, breaking ties according to a predetermined fixed ordering. We start from the layer V_i for $i = n$ and move backwards. For every vertex $v \in V_i$ we relabel its outgoing edges if they do not satisfy the local monotonicity condition. Note that for each vertex v, the acceptance probability $p_{v\to}$ remains unchanged under this relabeling, and hence the ordering of the vertices within each layer remains the same. Furthermore, swapping the labels of the outgoing edges of a vertex $v \in V_{i-1}$ flips the sign of the Fourier coefficient $\widehat{f_{v\to}}(i)$ so that it is nonpositive. □

Note that if f is regular, then so is the local monotonization f'. We also rely on the following general formula for Fourier coefficients of standard-order ROBPs.

[4] In the literature, the reverse inequality is claimed [51], [149], but this seems to be a mistake.

Lemma 5.15. Let f be a standard-order ROBP with layers $V_0, \dots, V_n$. Let $i \in \{0, 1, \dots, n\}$, let $S \subseteq \{1, 2, \dots, i\}$, and let $T \subseteq \{i+1, i+2, \dots, n\}$. Then [5]

$$\widehat{f}(S \cup T) = \sum_{v \in V_i} \widehat{f_{\to v}}(S) \cdot \widehat{f_{v \to}}(T).$$

Proof.

$$\begin{aligned}
\widehat{f}(S \cup T) &= \mathop{\mathbb{E}}_{\substack{x \sim U_i \\ y \sim U_{n-i}}} [f(xy) \cdot \chi_{S \cup T}(xy)] \\
&= \mathop{\mathbb{E}}_{\substack{x \sim U_i \\ y \sim U_{n-i}}} \left[\left(\sum_{v \in V_i} f_{\to v}(x) \cdot f_{v \to}(y) \right) \cdot \chi_S(x) \cdot \chi_T(y) \right] \\
&= \sum_{v \in V_i} \widehat{f_{\to v}}(S) \cdot \widehat{f_{v \to}}(T). \qquad \square
\end{aligned}$$

Given Observation 5.1 and Lemma 5.15, we are prepared to bound the higher-level Fourier coefficients of a regular ROBP, using an inductive argument due to Chattopadhyay *et al.* [51].

Proof of Theorem 5.13. We will show by induction on d that

$$\sum_{|S|=d} |\widehat{f}(S)| \leq (w-1)^d \cdot \mathbb{E}[f].$$

Lemma 5.14 is the base case $d = 1$. For the inductive step, let the layers of f be $V_0, \dots, V_n$. Then

$$\begin{aligned}
\sum_{|S|=d+1} |\widehat{f}(S)| &= \sum_{i=1}^{n} \sum_{\substack{T \subseteq [i-1] \\ |T|=d}} |\widehat{f}(T \cup \{i\})| \\
&= \sum_{i=1}^{n} \sum_{\substack{T \subseteq [i-1] \\ |T|=d}} \left| \sum_{v \in V_{i-1}} \widehat{f_{\to v}}(T) \cdot \widehat{f_{v \to}}(i) \right| \qquad \text{(Lemma 5.15)} \\
&\leq \sum_{i=1}^{n} \sum_{\substack{T \subseteq [i-1] \\ |T|=d}} \sum_{v \in V_{i-1}} |\widehat{f_{\to v}}(T)| \cdot |\widehat{f_{v \to}}(i)|
\end{aligned}$$

[5] To properly interpret the formula, we think of the variables of $f_{v\to}$ as being numbered $i+1, i+2, \dots, n$, so its Fourier coefficients are $\widehat{f_{v\to}}(T)$ for $T \subseteq \{i+1, i+2, \dots, n\}$.

$$= \sum_{i=1}^{n} \sum_{v \in V_{i-1}} \left(\sum_{\substack{T \subseteq [i-1] \\ |T|=d}} |\widehat{f_{\to v}}(T)| \right) \cdot |\widehat{f_{v\to}}(i)|$$

$$\leq (w-1)^d \cdot \sum_{i=1}^{n} \sum_{v \in V_{i-1}} \mathbb{E}[f_{\to v}] \cdot |\widehat{f_{v\to}}(i)|$$

by induction. Now, to get rid of the absolute value signs, let f' be the local monotonization of f from Observation 5.1. Then continuing, we have

$$\begin{aligned}
\sum_{|S|=d+1} |\widehat{f}(S)| &\leq (w-1)^d \cdot \sum_{i=1}^{n} \sum_{v \in V_{i-1}} \mathbb{E}[f_{\to v}] \cdot |\widehat{f_{v\to}}(i)| \\
&= (w-1)^d \cdot \sum_{i=1}^{n} \sum_{v \in V_{i-1}} \mathbb{E}[f'_{\to v}] \cdot (-\widehat{f'_{v\to}}(i)) \\
&= (w-1)^d \cdot \sum_{i=1}^{n} - \mathop{\mathbb{E}}_{x \sim U_n} \left[\sum_{v \in V_{i-1}} f'_{\to v}(x) \cdot f'_{v\to}(x) \cdot (-1)^{x_i} \right] \\
&= (w-1)^d \cdot \sum_{i=1}^{n} -\widehat{f'}(i) \\
&\leq (w-1)^d \cdot (w-1) \cdot \mathbb{E}[f'] \\
&= (w-1)^{d+1} \cdot \mathbb{E}[f],
\end{aligned}$$

where the inequality holds by Lemma 5.14. □

Fourier growth bounds imply simplification on average

So far, we have shown that regular ROBPs have bounded Fourier growth. Next, we show that in general, functions with bounded Fourier growth simplify on average under restrictions, to the point that they can be fooled by k-wise small-bias distributions.

Proposition 5.1 (Bounded Fourier growth $\implies$ simplification on average under restrictions)**.** Let $a, b \geq 1$ and $f \in \mathcal{L}_1^n(a,b)$. Let $\varepsilon \in (0,1)$, and let $X \in \{-1,1\}^n$ be k-wise δ-biased,[6] where

$$\delta = \varepsilon/(2a) \quad \text{and } k = \lceil \log(2a/\varepsilon) \rceil.$$

[6] Since we are working over ± 1, the meaning of "k-wise δ-biased" is that for every nonempty set $S \subseteq [n]$ with $|S| \leq k$, we have $|\mathbb{E}[\prod_{i \in S} X_i]| \leq \delta$.

Let $p = 1/(2b)$. Then X fools $T_p f$ with error ε.

Proof. The Fourier expansion of $T_p f$ is given by

$$\begin{aligned} T_p f(x) = \mathop{\mathbb{E}}_{R \sim R_p}[f(R \circ x)] &= \sum_{S \subseteq [n]} \widehat{f}(S) \mathop{\mathbb{E}}_{R \sim R_p}[\chi_S(R \circ x)] \\ &= \sum_{S \subseteq [n]} \widehat{f}(S) \cdot p^{|S|} \cdot \chi_S(x). \end{aligned}$$

(Informally, the noise operator T_p "attenuates" the Fourier coefficients of f.) Noting that $\widehat{T_p f}(\varnothing) = \mathbb{E}[T_p f]$, we have

$$\begin{aligned} |\mathbb{E}[T_p f(X)] - \mathbb{E}[T_p f]| &= \left| \sum_{\substack{S \subseteq [n] \\ S \neq \varnothing}} p^{|S|} \widehat{f}(S) \, \mathbb{E}\left[\prod_{i \in S} X_i\right] \right| \\ &\leq \sum_{\substack{S \subseteq [n] \\ S \neq \varnothing}} p^{|S|} \left|\widehat{f}(S)\right| \cdot \left| \mathbb{E}\left[\prod_{i \in S} X_i\right] \right|. \end{aligned}$$

When $|S| \leq k$, we have $\left|\mathbb{E}\left[\prod_{i \in S} X_i\right]\right| \leq \delta$. When $|S| > k$, we use the trivial bound $\left|\mathbb{E}\left[\prod_{i \in S} X_i\right]\right| \leq 1$. Plugging these bounds into the above inequality, we get

$$\begin{aligned} |\mathbb{E}[T_p f(X)] - \mathbb{E}[T_p f]| &\leq \delta \cdot a \cdot \sum_{d=1}^{k} (pb)^d + a \cdot \sum_{d=k+1}^{n} (pb)^d \\ &\leq \delta a + 2^{-k} a \\ &\leq \varepsilon. \end{aligned}$$

□

Thus, in particular, bounded-width standard-order regular ROBPs simplify on average under restrictions.

5.2.3 Using Fourier growth bounds to obtain PRGs

Using the analysis in the previous section, let us now obtain a PRG for ROBPs. We already presented some PRGs for *standard-order* ROBPs in Section 3, such as the INW PRG. However, those PRGs rely heavily

on the fact that the order of the input variables of the ROBP is known to the generator. A more challenging scenario is when we wish to fool functions of the form

$$f(x) = g(x_{\pi(1)}, \ldots, x_{\pi(n)}),$$

where g is a width-w length-n standard-order ROBP and π is an *unknown* permutation of the coordinates $[n]$. We refer to such a function f as a width-w length-n *arbitrary-order ROBP* (see Figure 5.2). One motivation for fooling arbitrary-order ROBPs is that they can simulate other interesting classes of tests, such as read-once formulas [31]. In this section, we will focus on a subclass of ROBPs called *permutation ROBPs.*

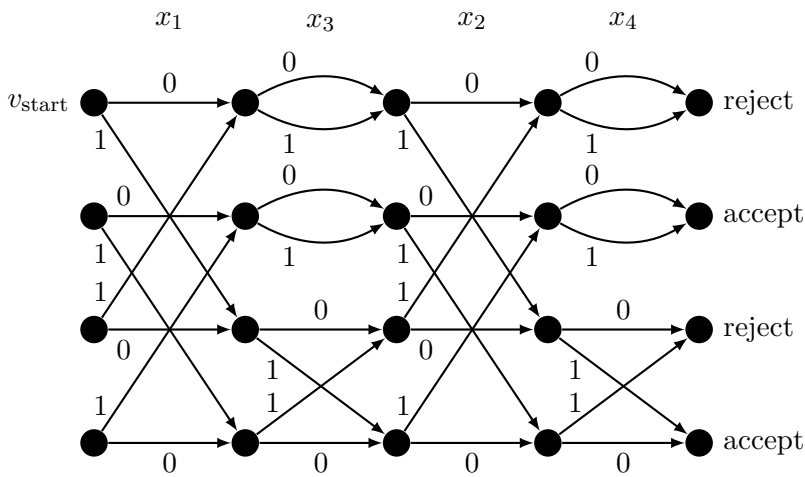

Figure 5.2: Define $f\colon \{0,1\}^{2n} \to \{0,1\}$ by the formula $f(x) = \bigoplus_{1 \le i \le j \le n} x_i \cdot x_{n+j}$. This function can be computed by a width-4 arbitrary-order permutation ROBP (the case $n = 2$ is shown above). In contrast, every *standard-order* ROBP computing f has width $2^{\Omega(n)}$. This can be shown by a communication complexity argument, reducing from the inner product mod 2 problem.

Definition 5.12 (Permutation ROBPs)**.** Let f be a length-n arbitrary-order ROBP with layers $V_0, \ldots, V_n$. We say that f is a *permutation ROBP* if for every $i \in [n]$ and $v \in V_i$, there are exactly two incoming edges to v (regularity), and those two edges have distinct labels (one is labeled 0 and the other is labeled 1).

In Section 3.3, we saw the BRRY generator, which fools constant-width *standard-order* regular ROBPs with seed length $\widetilde{O}(\log n)$. We will now show how to design another PRG for constant-width permutation ROBPs, which once again has seed length $\widetilde{O}(\log n)$, but this time the PRG works even in the more challenging arbitrary-order setting:

Theorem 5.16 (PRG for arbitrary-order permutation ROBPs [49], [149], [202])**.** For every $w, n \in \mathbb{N}$ and $\varepsilon > 0$, there is an ε-PRG for width-w length-n arbitrary-order permutation ROBPs that has seed length $\widetilde{O}(w^2 \cdot \log n \cdot \log(1/\varepsilon))$[7] and that is computable in time $\mathrm{poly}(n, 1/\varepsilon)$.

As a reminder, we showed in Section 5.2.2 that permutation ROBPs (and more generally regular ROBPs) simplify on average under restrictions. To prove Theorem 5.16, we now observe that such an average-case simplification-under-restrictions lemma is sufficient for the polarizing random walks framework.

Lemma 5.17 (Simplification on average implies fractional PRGs)**.** Let $\mathcal{F}$ be a class of functions $f\colon \{\pm 1\}^n \to \mathbb{R}$. Let $p, \varepsilon > 0$, let X be a distribution over $\{-1, 1\}^n$, and assume that for every $f \in \mathcal{F}$, the distribution X fools $T_p f$ with error ε. Then pX is (p^2)-noticeable and fools $\mathcal{F}$ with error ε.

Proof. The proof is exactly the same as the proof of Lemma 5.2, except that we replace the final step with the following:

$$\left| \mathop{\mathbb{E}}_{R,X}[f(R \circ X)] - \mathbb{E}[f] \right| = \left| \mathop{\mathbb{E}}_{X}[T_p f(X)] - \mathbb{E}[T_p f] \right| \leq \varepsilon. \qquad \square$$

Putting everything together gives us our PRG for bounded-width arbitrary-order permutation ROBPs:

Proof of Theorem 5.16. Let f be a width-w length-n arbitrary-order permutation ROBP. Such a program f satisfies the Fourier growth bound of Theorem 5.13 regardless of the order in which it reads the

[7] The specific seed length in Theorem 5.16 does not appear in prior work, but it does follow directly from prior work [49], [91], [149], [202]. In particular, if $\varepsilon > 1/n$, then it follows from Lee *et al.*'s work [149], whereas if $\varepsilon \leq 1/n$, then it follows from Forbes and Kelley's work [91]. Our refined polarizing random walks framework (Theorem 5.10) gives us a unified proof of Theorem 5.16 for all ε.

input variables, because the quantity $\sum_{|S|=d} |\widehat{f}(S)|$ is invariant under variable permutations.

Let $\gamma = \varepsilon/t$ for a suitable value $t = O(w^2 \log n)$. Let $\delta = \gamma/2$, let $k = \lceil \log(2/\gamma) \rceil$, and let $X \in \{-1,1\}^n$ be k-wise δ-biased. Let $p = \frac{1}{2\cdot(w-1)}$. Then by Theorem 5.13 and Proposition 5.1, X fools $T_p f$ with error γ. Therefore, by Lemma 5.17, pX is (p^2)-noticeable and fools f with error γ.

The distribution X can be explicitly sampled using a seed of length $s = O(\log(1/\gamma) + \log\log n)$ (see Theorem 2.5). Furthermore, the class of width-w permutation ROBPs is closed under restrictions and shifts. Therefore, by the refined polarizing random walks framework (Theorem 5.10), there is a PRG for such programs with error $O(\gamma \cdot p^{-2} \cdot \log n) = \varepsilon$ and seed length

$$O\left(s \cdot p^{-2} \cdot \log n\right) = O\left(w^2 \cdot \log n \cdot (\log(w/\varepsilon) + \log\log n)\right),$$

computable in time $\mathrm{poly}(n, 1/\varepsilon)$. □

To be clear, the Fourier growth bound (Theorem 5.13) works for all *regular* ROBPs, not merely permutation ROBPs. However, the class of width-w arbitrary-order regular ROBPs is unfortunately not closed under restrictions. It is therefore unclear how to apply the polarizing random walks framework to such programs.

Open Problem 5.1 (PRGs for arbitrary-order regular ROBPs)**.** Design a PRG for constant-width arbitrary-order *regular* ROBPs with seed length $o(\log^2 n)$.

Thankfully, the more restricted class consisting of width-w arbitrary-order *permutation* ROBPs is closed under restrictions, a fact which is crucial in the proof of Theorem 5.16.

More generally, by the same argument, one can use the polarizing random walks framework to construct a PRG for any class with bounded Fourier growth, provided that the class is closed under restrictions and shifts.[8]

[8] One can show that closure under shifts is not necessary. On the other hand, it seems quite challenging to handle classes that are not closed under restrictions.

Theorem 5.18 (PRG for functions with bounded Fourier growth)**.** For every n, a, b, ε, there is an explicit PRG G such that if $\mathcal{F} \subseteq \mathcal{L}_1(a, b)$ and $\mathcal{F}$ is closed under restrictions and shifts, then G fools $\mathcal{F}$ with error ε. Furthermore, G has seed length

$$O(b^2 \cdot \log n \cdot (\log(ab/\varepsilon) + \log\log n)).$$

When Reingold *et al.* [202] proved their Fourier growth bound for standard-order regular ROBPs, they also conjectured a Fourier growth bound for all standard-order ROBPs (whether regular or not). In particular, they conjectured that constant-width standard-order ROBPs are in $\mathcal{L}_1(\text{poly}(n), \text{polylog}(n))$. The width-3 case of this conjecture was proven by Steinke *et al.* [225], and then the general case was proven by Chattopadhyay *et al.* [51].

Theorem 5.19 (Fourier growth bound for ROBPs [51])**.** Suppose f is a width-w length-n standard-order ROBP. Then $f \in \mathcal{L}_1(1, O(\log n)^w)$. That is, for every $d \in [n]$,

$$\sum_{|S|=d} |\widehat{f}(S)| \leq O(\log n)^{wd}.$$

Combining Theorems 5.18 and 5.19 gives a PRG for constant-width *arbitrary-order* ROBPs with seed length $\text{polylog}(n)$. Later (Section 5.4), we will see a better PRG for this class that will make use of Theorem 5.19 in a more sophisticated way.

Motivated by the goal of constructing new PRGs for classes of functions such as $\mathbf{AC}^0[\oplus]$ and $\mathbb{F}_2$-polynomials (see Open Problems 2.2 and 4.3), there has been effort on two fronts with the latter having led to some success: 1. Prove reasonable Fourier tail bounds for the said classes, and 2. Construct fractional PRGs under more relaxed assumptions, for example, Fourier tail bounds only on few levels. Chattopadhyay *et al.* [50] showed how to construct a fractional PRG using only *second-level* Fourier bounds. Then, Chattopadhyay *et al.* [48] showed that better bounds can be achieved if bounds on higher Fourier levels are available, and interestingly, that fractional PRGs can be achieved even from bounds on $|\sum_{S:|S|=d} \widehat{f}(S)|$ where one can have cancellations, as opposed to L_1 bounds. These works show that certain improved bounds on the

Fourier tails of $\mathbb{F}_2$-polynomials will lead to new PRGs. On the other hand, Viola [245] showed that the conjectured Fourier tail bounds in these works is equivalent to new correlation bounds, perhaps hinting at the difficulty in the success of these approaches.

5.3 Fooling $\mathbf{AC}^0$ via the Ajtai-Wigderson Framework

Let $\mathcal{F}$ be a class of functions $f\colon \{0,1\}^n \to \{0,1\}$ that we wish to fool. In this section, we revisit the case that functions in $\mathcal{F}$ simplify *with high probability* under restrictions. That is, suppose that for some values $p, \delta > 0$, we have

$$\Pr[f|_{\mathcal{R}_p} \in \mathcal{F}_{\text{simp}}] \geq 1 - \delta, \tag{5.5}$$

and furthermore suppose that we already have an explicit PRG for the "simpler" class $\mathcal{F}_{\text{simp}}$, say with seed length s.

In Section 5.1, we presented the polarizing walks framework, and we showed how to use it to construct a PRG for the original class $\mathcal{F}$ with a seed length of roughly $p^{-2} \cdot s \cdot \log n$. In this section, we present an older PRG framework due to Ajtai and Wigderson [4]. The Ajtai-Wigderson framework achieves a better seed length, but it requires a stronger initial assumption. Roughly speaking, we will show that if it is possible to "derandomize the choice of $\star$ positions" in Equation (5.5), then we can fool $\mathcal{F}$ with a seed length of approximately $p^{-1} \cdot s \cdot \log n$. The distinction between p^{-2} and p^{-1} is quite important in many cases.

5.3.1 Simplification under partially pseudorandom restrictions

The Ajtai-Wigderson framework is based on *partially pseudorandom restrictions*, meaning that the set of $\star$ positions is pseudorandom, but the assigned values are truly random. To reason about these two components of a restriction separately, we must introduce notation for encoding restrictions using bitstrings. Many different notational approaches have been used for this encoding; unfortunately, it seems inevitable that the notation is somewhat cumbersome and clunky. We will take the approach of defining the following $\circledast$ operation.

Definition 5.13 (Encoding restrictions)**.** For $x, y \in \{0,1\}^n$, define $x \circledast y \in \{0,1,\star\}^n$ by

$$(x \circledast y)_i = \begin{cases} x_i & \text{if } y_i = 0 \\ \star & \text{if } y_i = 1. \end{cases}$$

(See Figure 5.3.)

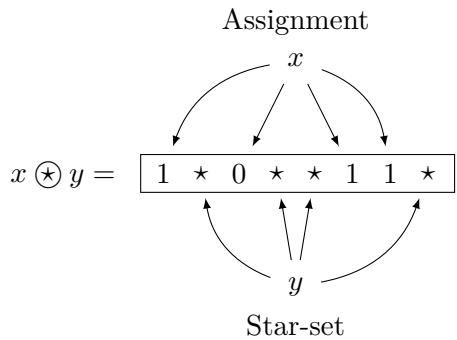

Figure 5.3: In the restriction $x \circledast y$, the string y indicates the $\star$ positions, and the string x assigns values to the non-$\star$ positions.

The assumption of the Ajtai-Wigderson framework is that we have a simplification-under-restrictions lemma of the form

$$\Pr[f|_{U \circledast Z} \in \mathcal{F}_{\text{simp}}] \geq 1 - \delta, \tag{5.6}$$

where the string of assigned bits $U \in \{0,1\}^n$ is distributed uniformly at random, the star set $Z \in \{0,1\}^n$ is independent of U, and Z can be sampled explicitly with a short seed. For example, for $\mathbf{AC}^0$, there is a line of work on proving *derandomized switching lemmas* [4], [142], [166], [209], [228], [235]. By combining Lyu's "derandomized multi-switching lemma" [166] with Rossman's arguments [205], one can prove the following.[9]

Lemma 5.20 (Simplification of $\mathbf{AC}^0$ under a partially pseudorandom restriction [166], [205])**.** For every n, m, d, δ, there is a random variable Z over $\{0,1\}^n$ that can be explicitly sampled using $\widetilde{O}(d \cdot \log(mn/\delta))$

[9] Note that Lyu actually proved a *fully* derandomized multi-switching lemma [166], but we only use the weaker version where the assigned bits are truly random.

truly random bits such that for any depth-d size-m $\mathbf{AC}^0$ circuit f on n input bits,

$$\Pr[\mathrm{DT}(f|_{U \circledast Z}) \leq O(\log(md/\delta))] \geq 1 - \delta.$$

Here, U is a uniform random n-bit string independent of Z, and $\mathrm{DT}(f|_{U \circledast Z})$ denotes the depth of the shallowest decision tree computing $f|_{U \circledast Z}$. Furthermore, for each coordinate i, the "star probability" is given by $\mathbb{E}[Z_i] = \Theta(1/\log m)^{d-1}$.

We will not study the proof of Lemma 5.20 here. Instead, we will focus on the logic of constructing a PRG *given* a statement such as Lemma 5.20. The only fact we will use about shallow decision trees is that they can be fooled with a short seed length. Indeed, in its simplest form, the Ajtai-Wigderson framework reduces the problem of constructing PRGs to the problem of proving statements like Equation (5.6) with the following parameters.

Theorem 5.21 (Simplification under partially-pseudorandom restrictions $\implies$ PRG [4])**.** Let $\mathcal{F}$ and $\mathcal{F}_{\mathrm{simp}}$ be classes of functions $f\colon \{0,1\}^n \to \{0,1\}$. Assume that $\mathcal{F}$ is closed under restrictions. Let Z be a random variable over $\{0,1\}^n$ that can be explicitly sampled using s truly random bits such that

$$\forall f \in \mathcal{F},\ \Pr[f|_{U \circledast Z} \in \mathcal{F}_{\mathrm{simp}}] \geq 1 - \delta$$

where $U \in \{0,1\}^n$ is uniform random and independent of Z. Assume that we can explicitly compute a value p such that for every $i \in [n]$, we have $\mathbb{E}[Z_i] \geq p$. Suppose also that there is an explicit δ-PRG for $\mathcal{F}_{\mathrm{simp}}$ with seed length s'. Then there is an explicit PRG for $\mathcal{F}$ with seed length $t \cdot (s + s')$ and error $O(t\delta)$, where $t = \lceil p^{-1} \ln(n/\delta) \rceil$.

For example, applying the Ajtai-Wigderson framework to Lemma 5.20 gives the following PRG for $\mathbf{AC}^0$.

Corollary 5.22 (PRG for $\mathbf{AC}^0$ via the Ajtai-Wigderson framework)**.** For every $n, m, d \in \mathbb{N}$ and $\varepsilon > 0$, there is an explicit ε-PRG for size-m depth-d $\mathbf{AC}^0$ circuits with seed length

$$O(\log m)^{d-1} \cdot \widetilde{O}(\log(mn/\varepsilon) \cdot \log(n/\varepsilon)).$$

When $m = \text{poly}(n)$ and $\varepsilon = 1/\text{poly}(n)$, the seed length in Theorem 5.22 is only $O(\log n)^{d+O(1)}$, which is superior to the $O(d)$ exponents in the seed lengths of the PRGs that we saw previously (Sections 2.6, 4.2 and 5.1). Trevisan and Xue were the first to achieve exponent $d + O(1)$ [235]. Several subsequent papers improved on their seed length [142], [166], [209], [228], and the current best seed length is achieved by Lyu [166]. Let d be constant and let $m \geq n$. Using a sophisticated variant of the Ajtai-Wigderson framework, Lyu designed an explicit ε-PRG for size-m depth-d $\mathbf{AC}^0$ circuits with seed length $\widetilde{O}(\log^{d-1} m \cdot \log(m/\varepsilon))$, which is quite close to the lack-of-lower-bounds barrier of $\Theta(\log^{d-1} m \cdot \log(1/\varepsilon))$ (see Section 4.1.2). The remaining gap between the two bounds is most pronounced for the case $d = 2$. The best PRGs for CNFs and DNFs have seed length $\widetilde{O}(\log(m/\varepsilon) \cdot \log m)$ [75], [228], whereas the lack-of-lower-bounds barrier allows for a seed length as low as $O(\log m \cdot \log(1/\varepsilon))$.

Open Problem 5.2 (Better PRGs for CNFs and DNFs)**.** Design an explicit PRG for polynomial-size CNFs and DNFs on n variables with error 0.1 and seed length $o(\log^2 n)$.

In the remainder of this section, we explain the Ajtai-Wigderson framework in its simplest form, i.e., we prove Theorem 5.21.

5.3.2 Restrictions that preserve expectation

The first step of the proof of Theorem 5.21 is to construct a *fully pseudorandom* restriction that *preserves the expectations* of functions in $\mathcal{F}$, as defined next.

Definition 5.14 (Preserving expectation)**.** Let $f\colon \{0,1\}^n \to \mathbb{R}$, and let R be a random variable distributed over $\{0,1,\star\}^n$. We say that R *preserves the expectation of f to within ε*, or *ε-preserves the expectation of f*, if

$$|\,\mathbb{E}[f|_R(U)] - \mathbb{E}[f]| \leq \varepsilon.$$

Here U is independent of R and distributed uniformly over $\{0,1\}^n$.

Recall that we are assuming that f simplifies under the partially-pseudorandom restriction $U \circledast Z$. To construct a restriction that preserves the expectation of f, we *replace the truly random bits with stars,*

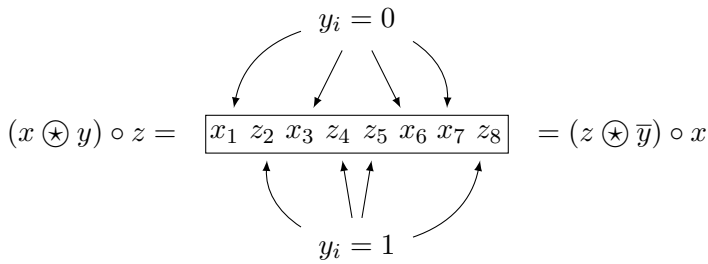

Figure 5.4: To compute $(x \circledast y) \circ z$, we use y to partition the coordinates into two parts. The coordinates in one part get their values from x, while the coordinates in the other part get their values from z. Thus, if we swap the roles of x and z *and* flip each bit of y, nothing changes. Here we depict the case $y = 01011001$.

and we replace the stars with pseudorandom bits. To explain further, for $y \in \{0,1\}^n$, let $\overline{y}$ denote the string obtained by flipping each bit of y, i.e., $\overline{y}_i = 1 - y_i$. Observe that the $\circledast$ operation (Definition 5.13) and the $\circ$ operation (Definition 5.1) satisfy the following "duality" condition:

Fact 5.2 (Restriction duality)**.** For any $x, y, z \in \{0,1\}^n$, we have

$$(x \circledast y) \circ z = (z \circledast \overline{y}) \circ x.$$

(See Figure 5.4.) As a consequence of Fact 5.2, whenever we have a simplification-under-restriction lemma with a derandomized set of $\star$ positions (Equation (5.6)), there is a related restriction that preserves expectations:

Lemma 5.23 (Simplification $\implies$ preserving expectation)**.** Let $\mathcal{F}_{\text{simp}}$ be a class of functions $f\colon \{0,1\}^n \to \{0,1\}$. Let X, Z, U be independent random variables taking values in $\{0,1\}^n$, where U is uniform and X fools $\mathcal{F}_{\text{simp}}$ with error ε. Let $f\colon \{0,1\}^n \to \{0,1\}$, and assume that

$$\Pr[f|_{U \circledast Z} \in \mathcal{F}_{\text{simp}}] \geq 1 - \delta.$$

Then $X \circledast \overline{Z}$ preserves the expectation of f to within $\varepsilon + \delta$.

Proof. We must show that $(X \circledast \overline{Z}) \circ U$ fools f. By Fact 5.2,

$$\begin{aligned} f((X \circledast \overline{Z}) \circ U) &= f((U \circledast Z) \circ X) \\ &= f|_{U \circledast Z}(X). \end{aligned}$$

By assumption, except with probability δ, $f|_{U \circledast Z} \in \mathcal{F}_{\text{simp}}$, and in this case, X fools the restricted function with error ε. □

Observe that if $U \circledast Z$ (the restriction that causes simplification) has $\star$-probability p, then $X \circledast \overline{Z}$ (the restriction that preserves expectations) has $\star$-probability $1 - p$. When we are trying to prove simplification under restrictions, we want the $\star$-probability to be as large as possible, whereas when we are trying to prove preservation of expectation, we want the $\star$-probability to be as small as possible.

5.3.3 Iterated restrictions

Lemma 5.20 provides us with a fully pseudorandom restriction R that preserves the expectation of $f \in \mathcal{F}$. Because R is fully pseudorandom, we can afford to sample it and apply it, so f becomes $f|_R$. The next lemma says that if we then sample another copy of R and further restrict the restricted function, we continue preserving its expectation.

Lemma 5.24 (Composing restrictions that preserve expectations)**.** Let $\mathcal{F}$ be a class of functions $f\colon \{0,1\}^n \to \mathbb{R}$ that is closed under restriction. Let $R^{(1)}, \ldots, R^{(t)}$ be independent random variables distributed over $\{0,1,\star\}^n$, assume that each $R^{(i)}$ preserves the expectation of every $f \in \mathcal{F}$ to within ε, and let $R = R^{(1)} \circ \cdots \circ R^{(t)}$. Then R preserves the expectation of every $f \in \mathcal{F}$ to within $\varepsilon \cdot t$.

Proof. Let us prove it by induction on t. The base case $t = 1$ is trivial. For the inductive step, let $R^{(<t)} = R^{(1)} \circ \cdots \circ R^{(t-1)}$ and assume that $R^{(<t)}$ preserves the expectation of every $f \in \mathcal{F}$ to within $\varepsilon \cdot (t-1)$. Fix some $f \in \mathcal{F}$, and let $F = f|_{R^{(<t)}}$. Since $\mathcal{F}$ is closed under restriction, we have $F \in \mathcal{F}$ with probability 1. Sample $U \in \{0,1\}^n$ uniformly at random. Then

$$
\begin{aligned}
&|\,\mathbb{E}[f|_R(U)] - \mathbb{E}[f]| \\
&\quad = |\,\mathbb{E}[f(R^{(<t)} \circ R^{(t)} \circ U)] - \mathbb{E}[f]| \\
&\quad = |\,\mathbb{E}[F|_{R^{(t)}}(U)] - \mathbb{E}[f]| \\
&\quad \le |\,\mathbb{E}[F|_{R^{(t)}}(U)] - \mathbb{E}[F(U)]| + |\,\mathbb{E}[F(U)] - \mathbb{E}[f]|
\end{aligned}
$$

$$\leq \mathop{\mathbb{E}}_{F}\left[\left|\mathop{\mathbb{E}}_{R^{(t)},U}[F|_{R^{(t)}}(U)] - \mathop{\mathbb{E}}_{U}[F(U)]\right|\right] + |\mathop{\mathbb{E}}[f|_{R^{(<t)}}(U)] - \mathbb{E}[f]|$$
$$\leq \mathop{\mathbb{E}}_{F}[\varepsilon] + \varepsilon \cdot (t-1) = \varepsilon \cdot t.$$

□

To get a full PRG, we will take t to be large enough that with high probability, the composed restriction R in Lemma 5.24 assigns values to *all* variables.

Lemma 5.25 (Preserving expectation $\implies$ PRG)**.** Let $\mathcal{F}$ be a class of functions $f\colon \{0,1\}^n \to \mathbb{R}$ that is closed under restrictions. Suppose there is a distribution R over $\{0,1,\star\}^n$ such that R can be explicitly sampled using s truly random bits, and R preserves the expectation of every $f \in \mathcal{F}$ to within δ, and for every coordinate i, $\Pr[R_i = \star] \leq 1 - p$. Then there is an explicit PRG for $\mathcal{F}$ with seed length $s \cdot t$ and error $O(t\delta)$, where $t = \lceil p^{-1} \ln(n/\delta) \rceil$.

Proof. Let $R^{\circ t} = R^{(1)} \circ \cdots \circ R^{(t)}$, where $R^{(1)}, \ldots, R^{(t)}$ are independent copies of R. Our PRG outputs the string $R^{\circ t} \circ 0^n$. This PRG clearly has seed length $s \cdot t$. By Lemma 5.24, the restriction $R^{\circ t}$ preserves the expectation of every $f \in \mathcal{F}$ to within error $t \cdot \delta$. Furthermore, the probability that there are any stars remaining in $R^{\circ t}$ is bounded by

$$\Pr[R^{\circ t} \notin \{0,1\}^n] \leq \sum_{i=1}^{n} \Pr[R_i^{\circ t} = \star] = \sum_{i=1}^{n} \Pr[R_i = \star]^t \leq \sum_{i=1}^{n} (1-p)^t \leq \delta.$$

Consequently, $R^{\circ t} \circ 0^n$ fools $\mathcal{F}$ with error $(t+1) \cdot \delta$. □

The Ajtai-Wigderson reduction, Theorem 5.21, stating that simplification under pseudorandom restrictions implies PRGs, follows from Lemma 5.23 (Simplification $\Rightarrow$ preserving expectation) and Lemma 5.25 (Preserving expectation $\Rightarrow$ PRG).

5.4 The Forbes-Kelley Generator for ROBPs

In Section 5.2, we presented a PRG for arbitrary-order *permutation* ROBPs. In this section, we present Forbes and Kelley's PRGs [91],

which fool *general* arbitrary-order ROBPs, without any permutation assumption. In the polynomial-width case, Forbes and Kelley achieve seed length $O(\log^3 n)$ (Theorem 5.28), and in the constant-width case, they achieve seed length $\widetilde{O}(\log^2 n)$ (Theorem 5.29).

These seed lengths constitute significant improvements over several earlier PRGs for arbitrary-order ROBPs [31], [51], [113], [130], [202], [225]. For polynomial-width programs, the best prior seed length was $\widetilde{O}(\sqrt{n})$ [202]. For width-w programs where $w = O(1)$, the best prior seed length was $\widetilde{O}(\log^{w+1} n)$ [51]. Forbes and Kelley's work implies the first PRG with polylogarithmic seed length for *read-once formulas with constant fan-in* over an arbitrary basis, since every such formula can be computed by a polynomial-width ROBP under some input order [31].

In terms of techniques, Forbes and Kelley's work builds on several earlier papers, especially work by Reingold *et al.* [202] and work by Haramaty *et al.* [113]. Forbes and Kelley's PRGs are based on a generalization of the Ajtai-Wigderson framework (Section 5.3). We begin by explaining the generalized framework, and then we will present Forbes and Kelley's PRGs.

5.4.1 Pseudorandomness plus noise

Let $\mathcal{F}$ be a class of functions that we wish to fool. Recall the first step of the Ajtai-Wigderson framework: we showed (Lemma 5.23) that if functions in $\mathcal{F}$ simplify with high probability under partially-pseudorandom restrictions, then we get a *fully pseudorandom* restriction that *preserves the expectation* of each $f \in \mathcal{F}$. We now refine that analysis to get an "if and only if" condition (actually three equivalent conditions).

Lemma 5.26 (Characterizing preservation of expectation)**.** Let X, Z, U be mutually independent random variables, where each is distributed over $\{0,1\}^n$ and U is uniform random. Let $f\colon \{0,1\}^n \to \mathbb{R}$ be a function and let $\varepsilon > 0$. The following are equivalent.

1. (Preservation of Expectation) The restriction $X \circledast Z$ preserves the expectation of f to within error ε, i.e.,

$$|\mathbb{E}[f|_{X \circledast Z}(U)] - \mathbb{E}[f]| \leq \varepsilon.$$

2. (Simplification on Average) The distribution X fools g with error ε, where
$$g(x) \stackrel{\text{def}}{=} \mathop{\mathbb{E}}_{Z,U}[f|_{U \circledast \overline{Z}}(x)]. \tag{5.7}$$

3. (Pseudorandomness Plus Noise) The distribution $X + (Z \wedge U)$ fools f with error ε, i.e.,
$$|\mathbb{E}[f(X + (Z \wedge U)] - \mathbb{E}[f]| \leq \varepsilon, \tag{5.8}$$
where $+$ denotes addition over $\mathbb{F}_2^n$ and $\wedge$ denotes coordinatewise conjunction.

Proof. (1 $\iff$ 2) By Fact 5.2, we have
$$f|_{X \circledast Z}(U) = f((U \circledast \overline{Z}) \circ X) = f|_{U \circledast \overline{Z}}(X).$$
Therefore, $\mathbb{E}[f|_{X \circledast Z}(U)] = \mathbb{E}[g(X)]$. Furthermore, because U is uniform random, we have $\mathbb{E}[g] = \mathbb{E}[f]$.

(2 $\iff$ 3) Because U is uniform random, the random variables $X + (Z \wedge U)$ and $(U \circledast \overline{Z}) \circ X$ are identically distributed. (In each case, we start with X and then we randomize the bits where $Z_i = 1$.) Therefore, $\mathbb{E}[f(X + (Z \wedge U)] = \mathbb{E}[g(X)]$. As mentioned previously, the fact that U is uniform random also implies that $\mathbb{E}[f] = \mathbb{E}[g]$. □

Recall that the *noise operator* with parameter p is defined by $T_p f(x) = \mathbb{E}[f|_{\mathcal{R}_p}(x)]$. The function g defined in Equation (5.7) can be understood as the result of applying a *partially derandomized noise operator* to f. Thus, Item 2 says that f *simplifies on average under partially-pseudorandom restrictions*. This condition is a combination of what we studied in Section 5.2 (simplification on average) and what we studied in Section 5.3 (simplification under partially-pseudorandom restrictions). A form of the perspective embodied by Item 2 was first studied by Gopalan *et al.* [105].

In this section, it will not be so useful to think in terms of "simplification." Instead, it will be more productive to reason about fooling f itself *using a partially-pseudorandom distribution*, as articulated in Item 3. Intuitively, establishing Equation (5.8) is easier than trying to design a PRG in one shot, because the helpful "noise vector" $Z \wedge U$

contributes a considerable amount of true randomness into the picture. This "pseudorandomness plus noise" perspective originates in work by Haramaty *et al.* [113]. We adopt this perspective for the rest of this section.

Once we obtain random variables $X, Z \in \mathbb{F}_2^n$ satisfying Equation (5.8), we can repeat the same procedure iteratively to get a PRG for $\mathcal{F}$. This is because $\mathcal{F}$ is closed under restrictions and for any fixed $x, z \in \mathbb{F}_2^n$, fooling the new function $g(y) = f(x + (z \wedge y))$ is equivalent to fooling $f|_{x \circledast z}$ (see Lemma 5.25). We now move on to explaining the Forbes-Kelley PRGs for arbitrary-order ROBPs, which are important examples of the "pseudorandomness plus noise" approach.

5.4.2 A Fourier decomposition lemma for ROBPs

The starting point for Forbes and Kelley's work is the following Fourier decomposition lemma for ROBPs. For simplicity, we assume that the ROBP uses the standard variable ordering (note that permuting variables just permutes Fourier coefficients in the obvious way).

Lemma 5.27 (Forbes-Kelley decomposition of ROBPs)**.** Let f be a length-n width-w standard-order ROBP with layers $V_0, V_1, \ldots, V_n$, and $k \geq 0$ be an arbitrary integer. Then

$$f(x) = L(x) + H(x),$$

where

$$L(x) = \sum_{S : |S| < k} \widehat{f}(S)\chi_S(x),$$

and

$$H(x) = \sum_{i=1}^{n} \sum_{v \in V_i} H_v(x) f_{v\to}(x),$$

where $H_v(x) = \sum_{S \subseteq [i] : |S| = k, i \in S} \widehat{f_{\to v}}(S)\chi_S(x)$.

Proof. Since $L(x)$ is exactly the degree $< k$ part of the Fourier expansion of f, we just need to show that

$$H(x) = \sum_{S \subseteq [n] : |S| \geq k} \widehat{f}(S)\chi_S(x).$$

For every S with $|S| \geq k$, let $i(S)$ denote the k-th smallest index that appears in S, let $S_L := S \cap [i(S)]$ be the indices in S up to $i(S)$, and let $S_R := S \setminus S_L$ be the remaining indices in S. Note that $|S_L| = k$ and $i(S) \in S_L$. We have

$$\widehat{f}(S) = \sum_{v \in V_{i(S)}} \widehat{f_{\to v}}(S_L) \cdot \widehat{f_{v\to}}(S_R),$$

where we used the fact that $f(x) = \sum_{i \in V_{i(S)}} f_{\to v}(x) \cdot f_{v\to}(x)$. Thus,

$$\begin{aligned}
\sum_{S \subseteq [n]: |S| \geq k} \widehat{f}(S)\chi_S(x) &= \sum_{i=1}^{n} \sum_{S: i(S)=i} \widehat{f}(S)\chi_S(x) \\
&= \sum_{i=1}^{n} \sum_{S: i(S)=i} \sum_{v \in V_i} \widehat{f_{\to v}}(S_L) \cdot \widehat{f_{v\to}}(S_R)\chi_{S_L}(x)\chi_{S_R}(x) \\
&= \sum_{i=1}^{n} \sum_{v \in V_i} \sum_{\substack{S_L \subseteq [i] \\ |S_L|=k \\ i \in S_L}} \sum_{S_R \subseteq [n] \setminus [i]} \widehat{f_{\to v}}(S_L) \cdot \widehat{f_{v\to}}(S_R)\chi_{S_L}(x)\chi_{S_R}(x) \\
&= \sum_{i=1}^{n} \sum_{v \in V_i} H_v(x) \sum_{S_R \subseteq [n] \setminus [i]} \widehat{f_{v\to}}(S_R)\chi_{S_R}(x) \\
&= \sum_{i=1}^{n} \sum_{v \in V_i} H_v(x) f_{v\to}(x) \\
&= H(x).
\end{aligned}$$

□

5.4.3 Pseudorandom restrictions that preserve the expectation of ROBPs

The key point of the construction will be to analyze ROBPs under bounded independent restrictions. In the proposition below, once again we assume for simplicity that the ROBP reads its variables in the standard order; this is without loss of generality because k-wise uniformity is preserved under variable permutations.

Proposition 5.2 (Preserving the expectation of ROBPs). Suppose $f : \{0,1\}^n \to \{0,1\}$ is computed by a width-w standard-order ROBP.

Suppose that X, Z, U are independent random variables, where X is $2k$-wise uniform, Z is k-wise uniform, and U is uniform. Then

$$\left| \mathop{\mathbb{E}}_{X,Z,U}[f(X + Z \wedge U)] - \mathbb{E}[f] \right| \leq \frac{wn}{2^{k/2}}.$$

Remark 5.3. For our current project of designing PRGs for ROBPs, the fact that Z is pseudorandom in Proposition 5.2 is not crucial. After all, if we take Z to be truly random, then Proposition 5.2 says that X fools $T_{1/2}f$, where T_p is the noise operator with parameter p. From there, the polarizing random walks framework gives a PRG (see Sections 5.1 and 5.2). Since p is a constant, the seed length from the polarizing random walks framework is the same as the seed length from the iterated restrictions framework for this case. (In general, the polarizing walks framework has a worse dependence on p.)

That being said, there are other benefits to the iterated restrictions paradigm besides seed length. In particular, the fact that Z is pseudorandom will be crucial in Section 5.5, which builds on this section using an "early termination" approach.

The intuition behind the proof is that the noise operator has the simplifying effect of significantly lowering the weight of the high degree terms, and the low degree terms are fooled by bounded independence.

Proof. The key is to utilize Lemma 5.27. We have

$$\begin{aligned}
&\left| \mathop{\mathbb{E}}_{X,Z,U}[f(X + Z \wedge U)] - \mathbb{E}[f] \right| \\
&\qquad = \left| \mathop{\mathbb{E}}_{X,Z,U}[L(X + Z \wedge U) + H(X + Z \wedge U)] - \widehat{f}(\varnothing) \right| \\
&\qquad = \left| \mathop{\mathbb{E}}_{X,Z,U}[H(X + Z \wedge U)] \right|,
\end{aligned}$$

where the last equality uses the fact that

$$\mathop{\mathbb{E}}_{X,Z,U}[L(X + Z \wedge U)] = \widehat{f}(\varnothing),$$

since $X + Z \wedge U$ is k-wise uniform. It is left to bound $|\mathbb{E}_{X,Z,U}[H(X + Z \wedge U)]|$. Note that,

$$\begin{aligned}
&\left|\mathop{\mathbb{E}}_{X,Z,U}[H(X+Z\wedge U)\right| \\
&\qquad= \left|\sum_{i=1}^{n}\sum_{v\in V_i}\mathop{\mathbb{E}}_{X,Z,U}[H_v(X+Z\wedge U)f_{v\to}(X+Z\wedge U)]\right| \\
&\qquad= \left|\sum_{i=1}^{n}\sum_{v\in V_i}\mathop{\mathbb{E}}_{X,Z}\left[\mathop{\mathbb{E}}_{U}[H_v(X+Z\wedge U)]\mathop{\mathbb{E}}_{U}[f_{v\to}(X+Z\wedge U)]\right]\right| \\
&\qquad\le \sum_{i=1}^{n}\sum_{v\in V_i}\mathop{\mathbb{E}}_{X,Z}\left[\left|\mathop{\mathbb{E}}_{U}[H_v(X+Z\wedge U)]\right|\right] \\
&\qquad\le nw\cdot\max_{v}\mathop{\mathbb{E}}_{X,Z}\left[\left|\mathop{\mathbb{E}}_{U}[H_v(X+Z\wedge U)]\right|\right],
\end{aligned}$$

where the first inequality uses the triangle inequality and the fact that $|\mathbb{E}_U[f_{v\to}(X+Z\wedge U)]|\le 1$. It is thus sufficient to bound the right-hand side. For every i and $v\in V_i$, we have

$$\begin{aligned}
&\mathop{\mathbb{E}}_{X,Z}\left[\left|\mathop{\mathbb{E}}_{U}[H_v(X+Z\wedge U)]\right|\right]^2 \\
&\le \mathop{\mathbb{E}}_{X,Z}\left[\left(\mathop{\mathbb{E}}_{U}[H_v(X+Z\wedge U)]\right)^2\right] \\
&= \mathop{\mathbb{E}}_{\substack{X,Z\\U,U'}}\left[\sum_{S,S'}\widehat{f_{\to v}}(S)\widehat{f_{\to v}}(S')\chi_S(X+Z\wedge U)\chi_{S'}(X+Z\wedge U')\right] \\
&= \sum_{S,S'}\widehat{f_{\to v}}(S)\widehat{f_{\to v}}(S')\mathop{\mathbb{E}}_{X,Z}\left[\mathop{\mathbb{E}}_{U}[\chi_S(X+Z\wedge U)]\mathop{\mathbb{E}}_{U'}[\chi_{S'}(X+Z\wedge U')]\right],
\end{aligned}$$

where the sums are over $S,S'\subseteq[i]$, such that $i\in S,S', |S|=|S'|=k$, and the first step is the Cauchy-Schwarz inequality. Now note that, whenever Z has a 1 coordinate in S or S', then $\mathbb{E}_U[\chi_S(X+Z\wedge U)]\,\mathbb{E}_{U'}[\chi_{S'}(X+Z\wedge U']=0$. Otherwise when Z is all zeros on both S and S' coordinates, we have $\chi_S(X+Z\wedge U)=\chi_S(X)$ and $\chi_{S'}(X+Z\wedge U')=\chi_{S'}(X)$. Now since X is $2k$-wise uniform, in this case we get

$$\begin{aligned}
&\mathop{\mathbb{E}}_{X}\left[\mathop{\mathbb{E}}_{U}[\chi_S(X+Z\wedge U)]\mathop{\mathbb{E}}_{U'}[\chi_{S'}(X+Z\wedge U')]\right] \\
&\qquad= \mathop{\mathbb{E}}_{X}[\chi_S(X)\cdot\chi_{S'}(X)] = \begin{cases}1 & \text{if } S=S'\\ 0 & \text{if } S\neq S'.\end{cases}
\end{aligned}$$

Putting these facts together, we get

$$\sum_{\substack{S,S' \subseteq [i]: \\ i \in S,S', \\ |S|=|S'|=k}} \widehat{f_{\to v}}(S)\widehat{f_{\to v}}(S') \mathop{\mathbb{E}}_{X,Z}\left[\mathop{\mathbb{E}}_{U}[\chi_S(X + Z \wedge U)] \mathop{\mathbb{E}}_{U'}[\chi_{S'}(X + Z \wedge U')]\right]$$

$$= \sum_{\substack{S \subseteq [i]: \\ i \in S, |S|=k}} \widehat{f_{\to v}}(S)^2 \Pr_Z[Z_i = 0 \ \forall i \in S]$$

$$= 2^{-k} \cdot \sum_{\substack{S \subseteq [i]: \\ i \in S, |S|=k}} \widehat{f_{\to v}}(S)^2 \leq 2^{-k},$$

where the second equality is due to Z being k-wise uniform, and the last inequality follows from Parseval's identity. □

Given Proposition 5.2, we can get a full ε-PRG for arbitrary-order ROBPs by choosing $k = O(\log(wn/\varepsilon))$ and applying the generic reduction Lemma 5.25. Using efficient constructions of k-wise and $2k$-wise uniform distributions, an individual restriction can be sampled explicitly using $O(k \log n) = O(\log n \cdot \log(\frac{nw}{\varepsilon}))$ truly random bits. Therefore, the overall seed length is as follows.

Theorem 5.28 (PRGs for arbitrary-order ROBPs [91])**.** For every $n, w \in \mathbb{N}$ and $\varepsilon > 0$, there is an explicit ε-PRG for width-w length-n arbitrary-order ROBPs with seed length $O(\log n \cdot \log(n/\varepsilon) \cdot \log(nw/\varepsilon))$. In particular, when $w = \text{poly}(n)$ and $\varepsilon = 1/\text{poly}(n)$, the seed length is $O(\log^3 n)$.

For small values of ε, a better seed length of $O(\log(nw/\varepsilon) \cdot \log^2 n)$ can be achieved by terminating the iterative restriction process after $O(\log n)$ restrictions instead of $O(\log(n/\varepsilon))$ restrictions and observing that the restricted function is an $O(\log(nw/\varepsilon))$-junta with high probability. See Forbes and Kelley's work for details [91].

5.4.4 A better generator for the small-width setting

Forbes and Kelley showed how to achieve a better seed length when w is small. The construction is similar, except that k-wise uniform distributions are replaced by small-bias distributions.[10]

[10] In Forbes and Kelley's work [91], they take the star-set T to be almost k-wise uniform; this is implied by the small-bias condition (see Theorem 2.6).

Proposition 5.3 (Preserving the expectation of ROBPs that have low level-k Fourier weight [91, Lemma 7.2])**.** Suppose $f\colon \{0,1\}^n \to \{0,1\}$ is computed by a width-w standard-order ROBP. Moreover, suppose that X and Z are independent β-biased random variables distributed over $\{0,1\}^n$. Let $k \in \mathbb{N}$, and let $L = \sum_{|S|=k} |\widehat{f}(S)|$. Then the restriction $X \circledast Z$ preserves the expectation of f to within

$$O\left(\left(2^{-k/2} + \sqrt{\beta} \cdot (L + 2^{k/4})\right) \cdot n \cdot w\right).$$

The proof of Proposition 5.3 is similar to the proof of Proposition 5.2, and we omit it. The point is that Proposition 5.3 allows us to take advantage of bounds on the Fourier growth of f. Plugging Theorem 5.19 into Proposition 5.3 and choosing $k = \Theta(\log(wn/\varepsilon))$ yields the following.

Proposition 5.4 (Preserving the expectation of low-width ROBPs)**.** For every $w, n \in \mathbb{N}$ and $\varepsilon > 0$, there is a value

$$\beta = 2^{-O(\log(wn/\varepsilon)\cdot w \cdot \log\log n)}$$

such that if X and Z are independent β-biased random variables distributed over $\{0,1\}^n$, then the restriction $X \circledast Z$ preserves the expectation of any width-w ROBP $f\colon \{0,1\}^n \to \{0,1\}$ to within ε.

Note that the restriction $X \circledast Z$ has $\star$-probability $1/2$, so we are assigning values to roughly half the input bits. Furthermore, when w is a constant, the restriction $X \circledast Z$ can be sampled using $\widetilde{O}(\log(n/\varepsilon))$ truly random bits. Iterating for $O(\log n)$ steps like before leads to the following PRG.

Theorem 5.29 ([91])**.** There is an explicit PRG for width-w length-n arbitrary-order ROBPs with error ε and seed length $O(w \cdot \log(nw/\varepsilon) \cdot \log n \cdot \log\log n)$.

5.5 PRGs for Read-once CNFs via Early Termination

Let $\mathcal{F}$ be a class of functions that we wish to fool. In Sections 5.3 and 5.4, our approach for fooling $\mathcal{F}$ was to first design a pseudorandom restriction R that approximately *preserves the expectations* of functions in $\mathcal{F}$. Then, we iteratively applied many copies of this restriction, assigning values

to more and more variables. If R assigns values to a p-fraction of the variables, then we perform roughly $p^{-1} \cdot \log n$ many iterations. Thus, if each copy of R costs s truly random bits, then this approach leads to a final seed length of roughly $p^{-1} \cdot s \cdot \log n$.

In this section, we develop a refined version of the iterated restrictions paradigm that can lead to an improved seed length in some cases. The idea is, we start by composing t copies of R, say $R' = R^{(1)} \circ \cdots \circ R^{(t)}$, where the number of iterations (t) is *significantly smaller* than $p^{-1} \cdot \log n$, such as perhaps $t = p^{-1} \cdot \log \log n$. Then, we prove that functions in $\mathcal{F}$ *simplify* under this composed restriction R'. That is, we show that

$$\forall f \in \mathcal{F},\ \Pr[f|_{R'} \in \mathcal{F}_{\text{simp}}] \geq 1 - \delta, \tag{5.9}$$

where $\mathcal{F}_{\text{simp}}$ is some class of "simpler" functions. Consequently, there is no need to continue sampling copies of R. Instead, we can use a PRG for $\mathcal{F}_{\text{simp}}$ to finish the job, taking advantage of the "simplicity" of the restricted function $f|_{R'}$. This leads to a final seed length of $s \cdot t + s'$, where s' is the seed length for fooling $\mathcal{F}_{\text{simp}}$.

Observe that the restriction R' in this framework is fully pseudorandom. Thus, the key challenge of this approach is that it requires proving a *fully-derandomized* simplification-under-restrictions lemma (Equation (5.9)). In Section 5.5.1, we show an example of a fully-derandomized simplification-under-restriction lemma, for the class of read-once CNF formulas. Then, in Section 5.5.2, we explain how to use that lemma to design a near-optimal PRG for such formulas.

5.5.1 Simplification of read-once CNFs under fully-pseudorandom restrictions

Recall that a CNF is a conjunction of *clauses*, each of which is a disjunction of literals. We will show that under a suitable pseudorandom restriction, a read-once CNF is likely to simplify, in the sense that the restricted formula only has a few remaining clauses. We begin with the following convenient definition, which generalizes the notion of γ-almost k-wise uniformity.

Definition 5.15 (k-wise γ-close distributions)**.** Let Σ be an alphabet, let $n \in \mathbb{N}$, and let X, Y be distributions over Σ^n. Let $k \in \mathbb{N}$ and $\gamma > 0$. We

say that X is *k-wise γ-close* to Y if for every $S \subseteq [n]$ with $|S| \leq k$, the substrings X_S and Y_S have total variation distance at most γ.

Recall that the *width* of a clause is the number of literals in the clause. A width-w CNF is a CNF in which each clause has width at most w. We begin by analyzing the effect of a "mild" pseudorandom restriction, i.e., a restriction that only assigns values to a constant fraction of the input variables. Intuitively, under such a restriction, the width of a read-once CNF should decrease by a constant factor, because in any given clause, a constant fraction of the variables should be assigned values. We show that this indeed occurs in all but a few "exceptional" clauses.

Lemma 5.30 (Simplification of read-once CNFs under mild fully-pseudorandom restrictions)**.** For every $w \in \mathbb{N}$ and $\delta > 0$, there is a value $k = O(w + \log(1/\delta))$ such that the following holds. Let f be a width-w read-once CNF. Let $\gamma = 2^{-4k}$, let $p \leq 2^{-10}$, and let $\widetilde{R}$ be a distribution over $\{0, 1, \star\}^n$ that is k-wise γ-close to $\mathcal{R}_p$. Then with probability $1 - \delta$, the restricted function $f|_{\widetilde{R}}$ can be computed by a read-once CNF of the form $f|_{\widetilde{R}} = f_{\text{narrow}} \wedge f_{\text{small}}$, where f_{narrow} is a read-once CNF of width at most $w/2$ and f_{small} is a read-once CNF with at most $O(\log^2(1/\delta))$ clauses.

To prove Lemma 5.30, we rely on the following tail bound, which we cite without proof.

Theorem 5.31 (Tail bound for almost k-wise independent random variables [82])**.** Let $q > 0$ and let $X \in \{0,1\}^m$, where $X_1, \ldots, X_m$ are independent random variables with $\mathbb{E}[X_i] \geq q$ for each i. Let k be an even positive integer, let $\gamma > 0$, and let $\widetilde{X}$ be k-wise γ-close to X. Then[11]

$$\Pr\left[\widetilde{X}_1 = \widetilde{X}_2 = \cdots = \widetilde{X}_m = 0\right] \leq \left(\frac{16k}{qm}\right)^{k/2} + 2k \cdot \gamma \cdot \left(\frac{1}{q}\right)^k.$$

[11] The statement in Doron *et al.*'s work [82] includes an extra assumption $k \leq qm$. This assumption is not necessary, because if $k > qm$, then the conclusion of the theorem is trivial.

The specific bound of Theorem 5.31 appears in work by Doron *et al.* [82] as a corollary of earlier work by Steinke *et al.* [225]; a similar bound appears in work by Celis *et al.* [43]. Now we prove Lemma 5.30 using Theorem 5.31.

Proof of Lemma 5.30. Let k be a multiple of w such that $32k \cdot 2^{-k/2} \leq \delta$. Let m be the number of clauses in f, say $f = C_1 \wedge \cdots \wedge C_m$. We consider three cases based on the value of m. For the first case, suppose that $m \leq 8^w$. For $j \in [m]$, let X_j be the indicator for the bad event that there are more than $w/2$ variables that are read by C_j and kept alive by R. Let f_{small} consist of all the clauses with $X_j = 1$ and let f_{narrow} consist of all the clauses with $X_j = 0$; our job is to show that f_{small} has few clauses with high probability. Let w_j be the number of variables that C_j reads, so $w_j \leq w$. If $\widetilde{R}$ is a *truly* random restriction $\widetilde{R} = \mathcal{R}_p$, then

$$\mathbb{E}[X_j] \leq \binom{w_j}{w/2} \cdot p^{w/2} \leq 2^w \cdot p^{w/2} \leq 2^{-4w}.$$

For any set $S \subseteq [m]$ of size $|S| = k/w$, the clauses $\{C_j\}_{j \in S}$ read a total of at most k variables, so when $\widetilde{R}$ is a *pseudorandom* restriction (namely k-wise γ-close to $\mathcal{R}_p$), we have

$$\Pr[X_S = 1^S] \leq \gamma + (2^{-4w})^{k/w} = 2 \cdot 2^{-4k}.$$

Therefore, by the union bound,

$$\Pr\left[\sum_j X_j \geq k/w\right] \leq \binom{m}{k/w} \cdot 2 \cdot 2^{-4k} \leq 8^{w \cdot k/w} \cdot 2 \cdot 2^{-4k} = 2 \cdot 2^{-k} \leq \delta/(16k).$$

Note that $k/w = O(1 + \log(1/\delta)/w) \leq O(\log(1/\delta))$, so indeed, with high probability, f_{small} has at most $O(\log(1/\delta))$ clauses.

Next, for the second case, suppose that $8^w < m \leq 8^w \cdot 16(k/w)$. In this case, write $f = f_1 \wedge \cdots \wedge f_{16(k/w)}$, where each f_j is a read-once CNF of width at most w with at most 8^w clauses. We apply the previous argument to each f_j. By the union bound, with probability $1 - \delta$, each $f_j|_{\widetilde{R}}$ can be written as $f^{(j)}_{\text{narrow}} \wedge f^{(j)}_{\text{small}}$, where $f^{(j)}_{\text{narrow}}$ has width at most $w/2$ and $f^{(j)}_{\text{small}}$ has at most $O(\log(1/\delta))$ clauses. Define

$f_{\text{narrow}} = \bigwedge_j f^{(j)}_{\text{narrow}}$ and $f_{\text{small}} = \bigwedge_j f^{(j)}_{\text{small}}$, and observe that f_{small} has at most $O(\log(1/\delta) \cdot k/w) \le O(\log^2(1/\delta))$ clauses.

For the final case, suppose that $m > 8^w \cdot 16(k/w)$. In this case, we will show that with high probability, the restricted function is identically zero. Let $Y = (Y_1, \dots, Y_m)$, where Y_j indicates whether the clause C_j is falsified by a truly random restriction, i.e., $Y_j = 1 \iff C_j|_{\mathcal{R}_p} \equiv 0$. Similarly, let $\widetilde{Y} = (\widetilde{Y}_1, \dots, \widetilde{Y}_m)$, where $\widetilde{Y}_j$ indicates whether $C_j|_{\widetilde{R}} \equiv 0$. Observe that $Y_1, \dots, Y_m$ are independent and $\mathbb{E}[Y_j] \ge (\frac{1-p}{2})^w \ge 4^{-w}$. Meanwhile, the vector $\widetilde{Y}$ is (k/w)-wise γ-close to Y. Therefore, by Theorem 5.31,

$$\begin{aligned}\Pr[f|_R \not\equiv 0] = \Pr[\widetilde{Y} = 0^m] &\le \left(\frac{16k/w}{4^{-w}m}\right)^{k/(2w)} + 2(k/w) \cdot \gamma \cdot 4^{w(k/w)} \\ &\le 2^{-k/2} + 2(k/w) \cdot 2^{-2k} \\ &\le \delta.\end{aligned}$$

□

Next, we analyze the effect of a pseudorandom restriction with a relatively small $\star$-probability (more similar to the traditional "switching lemma" regime). By applying Lemma 5.30 several times, we show that the number of clauses in a read-once CNF drastically decreases under such a restriction.

Corollary 5.32 (Simplification of read-once CNFs under fully-pseudorandom restrictions)**.** For every $w \in \mathbb{N}$ and $\delta > 0$, there is a value $k = O(w + \log(1/\delta))$ such that the following holds. Let f be a width-w read-once CNF. Let $\gamma = 2^{-4k}$, let $p \le 2^{-10}$, let $t = \lceil \log w \rceil$, and let $\widetilde{R}^{(1)}, \dots, \widetilde{R}^{(t)}$ be independent random variables, where each $\widetilde{R}^{(j)}$ is k-wise γ-close to $\mathcal{R}_p$. Then with probability $1 - \delta$, the restricted function $f|_{\widetilde{R}^{(1)} \circ \cdots \circ \widetilde{R}^{(t)}}$ can be computed by a read-once CNF with at most $\widetilde{O}(\log w \cdot \log^2(1/\delta))$ many clauses.

Proof sketch. We repeatedly apply Lemma 5.30 with failure probability δ/t. The first time, we apply it to f, which with high probability becomes $f_{\text{narrow}} \wedge f_{\text{small}}$. The second time, we apply it to f_{narrow}, which becomes $f'_{\text{narrow}} \wedge f'_{\text{small}}$. The third time, we apply it to f'_{narrow}, etc. After t

iterations, the width of the "narrow" part drops below 1, so only the "small" parts remain. All together, the "small" parts have $O(t \cdot \log^2(t/\delta))$ many clauses. □

5.5.2 Iterated restrictions with early termination

So far, we have shown that read-once CNFs simplify under fully-pseudorandom restrictions (Theorem 5.32). Next, we shall use Theorem 5.32 to design a PRG for read-once CNFs. Previously, we showed that δ-biased generators ε-fool read-once $\mathbf{AC}^0$ formulas (Section 2.5.3); this led to a seed length of $O(\log n \cdot \log(1/\varepsilon))$ for the depth-two case. Now we will show how to achieve near-optimal seed length $\widetilde{O}(\log(n/\varepsilon))$.

Theorem 5.33 (Near-optimal PRG for read-once depth-two formulas)**.** For every $n \in \mathbb{N}$ and $\varepsilon > 0$, there is an explicit ε-PRG for read-once CNFs and DNFs with seed length $\widetilde{O}(\log(n/\varepsilon))$.

Theorem 5.33 was first proven by Gopalan *et al.* [105]. (Compared to Gopalan *et al.*'s original proof [105], the analysis we present here is more similar to later work that considers more general models [82], [148].) Later, Doron *et al.* [81] achieved a slightly better seed length of $O(\log n) + \widetilde{O}(\log(1/\varepsilon))$. It remains an open problem to achieve the optimal seed length for this basic and fundamental class.

Open Problem 5.3 (Optimal PRGs for read-once depth-two formulas)**.** Construct an explicit PRG for read-once CNFs with seed length

$$O(\log(n/\varepsilon)).$$

As outlined at the beginning of this section, we prove Theorem 5.33 using the iterated restrictions paradigm. As the first (and main) step, let us aim for seed length $\widetilde{O}(w + \log(n/\varepsilon))$, where w is the width of the formula.

Lemma 5.34 (PRGs for bounded-width read-once CNFs)**.** For every $w, n \in \mathbb{N}$ and $\varepsilon > 0$, there is an explicit ε-PRG for width-w read-once CNFs on n input variables with seed length $\widetilde{O}(w + \log(n/\varepsilon))$.

Proof. The PRG outputs $R^{(1)} \circ \cdots \circ R^{(t)} \circ Z$, where each $R^{(i)}$ is distributed according to $X \circledast Y$, X and Y are β-biased, and Z is α-biased for suitable values

$$\begin{aligned} t &= O(\log w) \\ \beta &= 2^{-\Theta(w + \log(n/\varepsilon) \cdot \log\log n)} \\ \alpha &= 2^{-\Theta(\log(1/\varepsilon) \cdot \log\log(w/\varepsilon))}. \end{aligned}$$

Observe that the seed length is $O(t \log(n/\beta) + \log(1/\alpha))$, which is indeed $\widetilde{O}(w + \log(n/\varepsilon))$. Our remaining job is to prove correctness. Let $f\colon \{0,1\}^n \to \{0,1\}$ be a width-w read-once CNF, and let R be the composed restriction $R = R^{(1)} \circ \cdots \circ R^{(t)}$.

The first step of the correctness proof is to show that each individual restriction $R^{(i)}$ preserves the expectation of read-once CNFs. This is a straightforward consequence of Forbes and Kelley's work (see Section 5.4). Indeed, read-once CNFs can be simulated by width-3 arbitrary-order ROBPs. Therefore, by Proposition 5.4, each individual restriction $R^{(i)}$ preserves the expectation of f to within $\frac{\varepsilon}{3t}$, provided we choose $\beta < 2^{-c \cdot \log(n/\varepsilon) \cdot \log\log n}$ for a large enough constant c. Consequently, by Lemma 5.24, the composed restriction R preserves the expectation of f to within $\varepsilon/3$.

The next step is to show that read-once CNFs simplify under the composed restriction R. This is a straightforward consequence of our analysis in Section 5.5.1. Indeed, for every k, each individual restriction $R^{(i)}$ is k-wise $(2\beta \cdot 2^{k/2})$-close to $\mathcal{R}_{1/2}$ by Theorem 2.6. Therefore, a composition of *ten* restrictions such as $R^{(1)} \circ \cdots \circ R^{(10)}$ is k-wise $(20\beta \cdot 2^{k/2})$-close to $\mathcal{R}_p$ where $p = 2^{-10}$. Therefore, by Theorem 5.32, with probability $1 - \varepsilon/3$, the restricted function $f|_R$ is computable by a read-once CNF with at most m_* many clauses, where $m_* = \widetilde{O}(\log w \cdot \log^2(1/\varepsilon))$, provided we choose $t = 10\lceil \log w \rceil$ and $\beta < 2^{-c' \cdot (w + \log(1/\varepsilon))}$ for a large enough constant c'. Let E be the bad event that the restricted function is *not* computable by a read-once CNF with only m_* clauses.

The third step is to show that the small-bias distribution Z fools the simplified formula. This follows from our analysis in Section 2.5.3. In that section, we argued that α-bias generators fool read-once CNFs with error $\exp(-\Omega(\log(1/\alpha)/\log n))$. Looking at the proof more closely,

if the number of clauses is bounded by m_*, then the error is actually $\exp(-\Omega(\log(1/\alpha)/\log m_*))$. This error is at most $\varepsilon/3$ provided we choose a suitable value $\alpha = 2^{-\Theta(\log(1/\varepsilon)\cdot\log\log(w/\varepsilon))}$.

To wrap up the proof, let us compute the overall error of our PRG. Let U be a uniform random n-bit string. Then

$$\begin{aligned}
&|\,\mathbb{E}[f(R\circ Z)] - \mathbb{E}[f]| \\
&\qquad \le |\,\mathbb{E}[f(R\circ Z)] - \mathbb{E}[f(R\circ U)]| + |\,\mathbb{E}[f(R\circ U)] - \mathbb{E}[f]| \\
&\qquad \le \mathop{\mathbb{E}}_{R}\left[\left|\mathop{\mathbb{E}}_{Z}[f|_R(Z)] - \mathop{\mathbb{E}}_{U}[f|_R(U)]\right|\right] + \varepsilon/3 \\
&\qquad \le \mathop{\mathbb{E}}_{R}\left[\left|\mathop{\mathbb{E}}_{Z}[f|_R(Z)] - \mathop{\mathbb{E}}_{U}[f|_R(U)]\right| \mid \neg E\right] + 2\varepsilon/3 \\
&\qquad \le \varepsilon.
\end{aligned}$$

□

To complete the proof of Theorem 5.33, we need to eliminate the dependence on width from Lemma 5.34. We accomplish this by a sandwiching argument.

Lemma 5.35. Let f be a read-once CNF. For every $\varepsilon > 0$, f can be ε-sandwiched by read-once CNFs of width $\lceil\log(n/\varepsilon)\rceil$.

Proof. Define f_u by deleting from f all clauses of width greater than $\lceil\log(n/\varepsilon)\rceil$, and meanwhile define f_ℓ by deleting all but the first $\lceil\log(n/\varepsilon)\rceil$ literals from each clause. Clearly, $f_\ell \le f \le f_u$. Furthermore, a clause of width $\lceil\log(n/\varepsilon)\rceil$ has expectation at least $1 - \varepsilon/n$. A read-once CNF can have at most n clauses, so by the union bound,

$$\Pr_x[f_u(x) \ne f_\ell(x)] \le \varepsilon.$$ □

Theorem 5.33 is an immediate consequence of Lemmas 5.34, 5.35 and 2.18.

5.5.3 Discussion: Two types of simplification

Looking again at the construction and analysis, our near-optimal PRG for read-once CNFs is based on combining *two distinct* simplification-under-restrictions lemmas:

- The first step of the PRG is applying a Forbes-Kelley pseudorandom restriction R (with $\star$-probability $p = 1/2$). As discussed in Section 5.4.1, the fact that R preserves the expectation of every read-once CNF is equivalent to saying that every read-once CNF "simplifies on average" under a related partially-pseudorandom restriction (with $\star$-probability $1 - p = 1/2$).
- The second step of the PRG is arguing that read-once CNFs simplify with high probability under $R^{\circ t}$ for some $t = O(\log \log n)$. Note that $R^{\circ t}$ is a fully-pseudorandom restriction with $\star$-probability $1/\operatorname{polylog}(n)$.

This is an attribute of the early termination framework more generally. In general, when we're trying to prove the first simplification-under-restrictions lemma, two things are working in our favor: the relevant restriction is only *partially* pseudorandom, and it suffices to show simplification *on average*. On the other hand, when we're trying to prove the second simplification-under-restrictions lemma, we have something else going for us: the *$\star$-probability* is relatively low. The early termination framework's power comes from the fact that we get to *combine the advantages* of these two different settings.

In the decade since Gopalan *et al.* [105] introduced the early termination framework, it has proven to be a versatile and powerful approach to PRG design, especially in the regime of near-optimal seed length [80]–[82], [105], [148], [152], [171]. Recently, Lyu introduced a different "partition-based" refinement of the iterated restrictions framework, which is also based on showing simplification under purely-pseudorandom restrictions, and used it to design an improved PRG for $\mathbf{AC}^0$ circuits [166].

5.6 Fooling General Branching Programs via the IMZ Framework

Let $\mathcal{F}$ be a class of functions that we wish to fool. Let us suppose yet again that functions in $\mathcal{F}$ simplify with high probability under restrictions. That is, for some values $p, \delta > 0$, we have

$$\Pr[f|_{\mathcal{R}_p} \in \mathcal{F}_{\text{simp}}] \geq 1 - \delta, \tag{5.10}$$

where $\mathcal{F}_{\text{simp}}$ is some class of "simpler" functions.

In Section 5.1, we used the polarizing walks framework to design a PRG for $\mathcal{F}$ with a seed length of roughly $p^{-2} \cdot s$ (ignoring log factors), where s is the seed length of a PRG for $\mathcal{F}_{\text{simp}}$. Then, in Section 5.3, we showed that if it is possible to *partially derandomize* Equation (5.10), then we can improve the seed length to roughly $p^{-1} \cdot s$ using the Ajtai-Wigderson framework. In this section, we will present a framework due to Impagliazzo *et al.* [130] (the "IMZ framework"). Assuming that it is possible to *fully* derandomize Equation (5.10), the IMZ framework gives a PRG for $\mathcal{F}$ with a seed length of roughly $p^{-1} + r$, where r is the *description length* of functions in the "simpler" class $\mathcal{F}_{\text{simp}}$, i.e., $r = \log |\mathcal{F}_{\text{simp}}|$.

Observe that our measure of "simplicity" has changed. In all the previous sections, what mattered was the cost of *fooling* functions in $\mathcal{F}_{\text{simp}}$ (i.e., s), but now, what matters is the cost of *describing* functions in $\mathcal{F}_{\text{simp}}$ (i.e., r). Usually, the fooling cost s is much smaller than the description length r. (Indeed, ideally we hope for $s \approx \log r$; see Proposition 1.1.) Nevertheless, the IMZ framework is sometimes superior to the Ajtai-Wigderson framework, because the final seed length in the Ajtai-Wigderson framework is approximately the *product* $p^{-1} \cdot s$, whereas the final seed length in the IMZ framework is closer to the *sum* $p^{-1} + r$.

We emphasize that the IMZ framework requires a *fully-derandomized* simplification-under-restrictions lemma (just like the early termination framework that we discussed in Section 5.5). In Section 5.6.1, we will prove a fully-derandomized simplification-under-restrictions lemma for *general branching programs* where we only have a bound on the size of the program (i.e., the number of vertices). Then, in Section 5.6.2, we explain how to use such a lemma to construct a PRG.

5.6.1 Shrinkage of branching programs under fully-pseudorandom restrictions

In this section, we study general size-m branching programs, with no restriction on the width or the number of times each variable is read.

Definition 5.16 (Unrestricted branching programs)**.** A *size-m branching program* over n input variables is a directed acyclic graph with at most

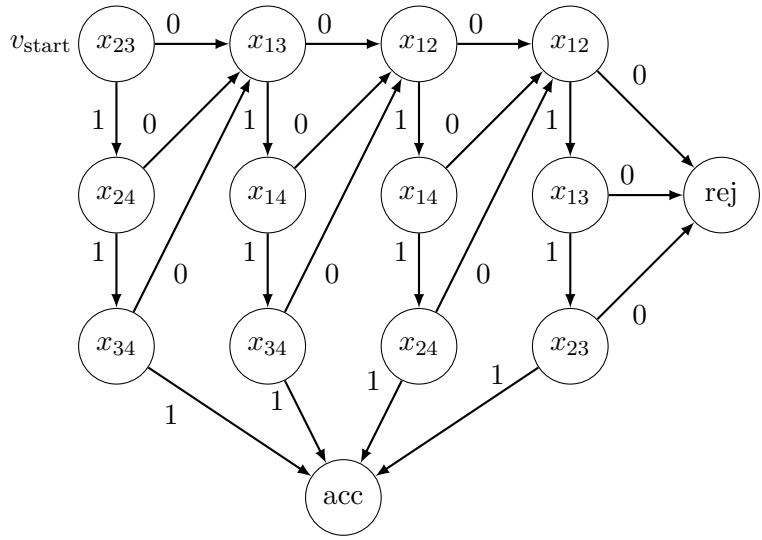

Figure 5.5: Let $n = m^2$, and let $x \in \{0,1\}^n = \{0,1\}^{m \times m}$ be the adjacency matrix of an undirected graph G. There is a branching program of size $O(m^3) = O(n^{1.5})$ that tests whether G has a triangle given x (the case $m = 4$ is shown above). In contrast, every *read-once* branching program computing this function must have width $2^{\Omega(n)}$, even if we allow arbitrary variable ordering. This follows from communication complexity lower bounds in the best-case partition model [186].

m vertices. Each non-sink vertex v is labeled with an index $j_v \in [n]$ has two outgoing edges labeled 0 and 1. A subset V_{accept} of the sink vertices are designated as "accepting vertices." Given an input $x \in \{0,1\}^n$, the program starts at a designated "start vertex" v_{start}, and in each step, having reached a vertex v, the program queries x_{j_v} and traverses the corresponding outgoing edge. Eventually, the program reaches a sink vertex v, and $f(x) = 1 \iff v \in V_{\text{accept}}$ (see Figure 5.5).

Let $\mathrm{BP}(f)$ denote the size of the smallest branching program computing f. One can easily show that branching programs *shrink* under truly random restrictions, in the sense that $\mathbb{E}[\mathrm{BP}(f|_{\mathcal{R}_p})] \leq p \cdot \mathrm{BP}(f)$. We will prove that similar shrinkage occurs *with high probability* rather than in expectation, and furthermore that it occurs under a *fully pseudorandom restriction*. In particular, our pseudorandom restriction distribution is as follows.

Definition 5.17 (k-wise independent restrictions)**.** Let R be a distribution over $\{0,1,\star\}^n$. We say that R is a *k-wise independent p-regular restriction* if the coordinates of R are k-wise independent, and the marginal distributions are given by

$$R_i = \begin{cases} \star & \text{with probability } p \\ 0 & \text{with probability } (1-p)/2 \\ 1 & \text{with probability } (1-p)/2. \end{cases}$$

Equivalently, R is k-wise 0-close to $\mathcal{R}_p$ (see Definition 5.15).

Lemma 5.36 (High-probability shrinkage of branching programs under fully-pseudorandom restrictions)**.** For every $\delta > 0$, there is a value $k = O(\log(1/\delta))$ such that for every $f\colon \{0,1\}^n \to \{0,1\}$ and every $p > 0$, if $R \in \{0,1,\star\}^n$ is a k-wise independent p-regular restriction, then

$$\Pr\left[\mathrm{BP}(f|_R) \le \lceil p \cdot \mathrm{BP}(f)\rceil \cdot 2^{O(\sqrt{\log(1/\delta)})}\right] \ge 1-\delta.$$

As discussed previously, what we really care about is the *description length* of $f|_R$, but this can be bounded in terms of the branching program size $\mathrm{BP}(f|_R)$. To prove Lemma 5.36, we rely on a tail bound for sums of k-wise independent random variables, which we cite without proof.[12]

Theorem 5.37 (Tail bound for sums of k-wise independent random variables [24, Lemma 2.3])**.** Let $X_1,\ldots,X_n \in [0,1]$ be k-wise independent, where $k \ge 4$ is an even integer, and let $X = \sum_{i=1}^n X_i$. Then for any $\Delta > 0$,

$$\Pr\left[|X - \mathbb{E}[X]| \ge \Delta\right] \le 8 \cdot \left(\frac{k\,\mathbb{E}[X] + k^2}{\Delta^2}\right)^{k/2}.$$

Proof of Lemma 5.36. Identify f with a branching program computing f of size $\mathrm{BP}(f)$. For $i \in [n]$, let m_i be the number of nodes in f that query x_i. Let H be the set of "heavy variables," namely $H = \{i : m_i > h\}$

[12] Here we are citing a bound due to Bellare and Rompel [24]. Skorski has shown an improvement to Bellare and Rompel's bound [221], but the improved bound is slightly more complicated and it makes no difference in our application, so we stick with Bellare and Rompel's simpler bound [24].

where $h = p \cdot \mathrm{BP}(f) \cdot 2^{\sqrt{\log(1/\delta)}}$. We first show that with high probability, few heavy variables are left alive. Indeed, since $k > \sqrt{\log(1/\delta)}$, we have

$$\begin{aligned}\Pr\left[|H \cap R^{-1}(\star)| \geq \sqrt{\log(1/\delta)}\right] &\leq \binom{|H|}{\sqrt{\log(1/\delta)}} \cdot p^{\sqrt{\log(1/\delta)}} \\ &\leq (p|H|)^{\sqrt{\log(1/\delta)}} \\ &\leq (p \cdot \mathrm{BP}(f)/h)^{\sqrt{\log(1/\delta)}} \\ &= \delta.\end{aligned}$$

Now let us consider the "light" variables. For each $i \notin H$, let $X_i = m_i \cdot \mathbb{1}[R_i = \star]/h \in [0,1]$. Let $X = \sum_{i \in H} X_i$, so $\mathbb{E}[X] \leq p \cdot \mathrm{BP}(f)/h < 1$. By Theorem 5.37 with $\Delta = O(\log(1/\delta))$, we have

$$\Pr[X \leq O(\log(1/\delta))] \geq 1 - \delta.$$

Our branching program for $f|_R$ begins by querying all the variables in $H \cap R^{-1}(\star)$ and storing all those values in memory. Then it simulates the branching program for f, skipping queries to variables in $H \cup R^{-1}(\{0,1\})$ since those values are known. The size of the branching program is

$$2^{|H \cap R^{-1}(\star)|} - 1 + 2^{|H \cap R^{-1}(\star)|} \cdot \sum_{i \in R^{-1}(\star) \setminus H} m_i < (1 + h \cdot X) \cdot 2^{|H \cap R^{-1}(\star)|},$$

which is bounded by $\lceil p \cdot \mathrm{BP}(f) \rceil \cdot 2^{O(\sqrt{\log(1/\delta)})}$ except with probability 2δ. Replacing δ with $\delta/2$ completes the proof. □

The parameters of Lemma 5.36 are perhaps a bit disappointing. Let $m = \mathrm{BP}(f)$. When $\delta < 2^{-\Theta(\log^2 m)}$, the lemma breaks down: it is not able to show that *any* shrinkage occurs with probability $1 - \delta$. Unfortunately, this is unavoidable. Indeed, by a standard counting argument, there exists a function f with $\mathrm{BP}(f) \geq m$ that only reads $k \stackrel{\text{def}}{=} O(\log m)$ of the input variables. For this function f, assuming $p \geq 1/m$, even under a truly random restriction $R = \mathcal{R}_p$, we have

$$\Pr[\mathrm{BP}(f|_R) = \mathrm{BP}(f)] \geq p^k \geq 2^{-O(\log^2 m)}.$$

Thus, in the regime $p \geq 1/m$ (which is the most interesting regime), one cannot prove a shrinkage lemma where the failure probability is exponentially small compared to m.

5.6.2 PRGs from fully-derandomized shrinkage lemmas

Now let us present the IMZ reduction.

Theorem 5.38 (Simplification under fully-pseudorandom restrictions $\implies$ PRG [130])**.** Let $\mathcal{F}$ and $\mathcal{F}_{\text{simp}}$ be classes of functions $f\colon \{0,1\}^n \to \{0,1\}$. Assume that $\mathcal{F}$ is closed under restrictions and shifts, and assume that $\mathcal{F}_{\text{simp}}$ contains the constant 0 function. Let $\delta > 0$, and let R be a random variable over $\{0,1,\star\}^n$ that can be explicitly sampled using q truly random bits such that

$$\forall f \in \mathcal{F},\ \Pr[f|_R \in \mathcal{F}_{\text{simp}}] \geq 1 - \delta.$$

Assume that we can explicitly compute a value p such that for every $i \in [n]$, we have $\Pr[R_i = \star] \geq p$. Let $r \in \mathbb{N}$, and assume that (a) $\log|\mathcal{F}_{\text{simp}}| \leq r$ and (b) there is an explicit δ-PRG for $\mathcal{F}_{\text{simp}}$ with seed length r.[13] Then there is an explicit PRG that fools $\mathcal{F}$ with error $O(t\delta)$ and seed length $O(t \cdot (q + \log(r/\delta)) + r)$, where $t = \lceil p^{-1} \ln(n/\delta) \rceil$.

We think of q and $\text{polylog}(nm/\delta)$ as "small," so the seed length in Theorem 5.38 is indeed approximately $p^{-1} + r$, as suggested at the beginning of this section. For branching programs, Theorem 5.38 implies the following PRG.

Corollary 5.39 (PRG for branching programs [130])**.** For any n, m, ε, there is an explicit ε-PRG for size-m branching programs with seed length $\sqrt{m} \cdot 2^{O(\sqrt{\log(m/\varepsilon)})} \cdot \text{polylog}\, n$.

When $\varepsilon = 1/\text{poly}(m)$ and $m \geq n$, the seed length in Theorem 5.39 is $m^{\frac{1}{2}+o(1)}$.

Proof sketch. Assume without loss of generality that $\log n \leq m \leq n^2$. Let p be the largest power of two with $p < 1/\sqrt{m}$. Let $t = \lceil p^{-1} \ln(n/\varepsilon) \rceil$ and let $\delta = \Theta(\varepsilon/t)$. Let R be a k-wise independent p-regular restriction where $k = O(\log(1/\delta))$. By a construction similar to the proof of Theorem 2.2, the distribution R can be sampled explicitly using q truly random bits, where

$$q = O(k \log(n/p)) = O(\log(m/\varepsilon) \log n).$$

[13] Given condition (a), condition (b) is relatively mild; note that the optimal seed length would be $O(\log(r/\delta))$ (see Proposition 1.1).

Let $\mathcal{F}_{\text{simp}}$ be the class of all branching programs of size at most m', where

$$m' = \lceil p \cdot m \rceil \cdot 2^{O(\sqrt{\log(1/\delta)})} = \sqrt{m} \cdot 2^{O(\sqrt{\log(m/\varepsilon)})}.$$

By Lemma 5.36, for every size-m branching program f, we have $\Pr[f|_R \in \mathcal{F}_{\text{simp}}] \geq 1 - \delta$. Let $r = O(m' \log n)$. Then $\log |\mathcal{F}_{\text{simp}}| \leq r$, and furthermore there is an explicit PRG that perfectly fools $\mathcal{F}_{\text{simp}}$ with seed length r, namely an m'-wise uniform generator. Therefore, by Theorem 5.38, there is an explicit PRG for size-m branching programs with error $O(t\delta) = \varepsilon$ and seed length

$$O(t \cdot (q + \log(r/\delta)) + r) = \sqrt{m} \cdot 2^{O(\sqrt{\log(m/\varepsilon)})} \cdot \text{polylog}(n). \qquad \square$$

In their original paper, Impagliazzo *et al.* [130] used the IMZ framework to design PRGs for a few additional classes, such as De Morgan formulas. Later, Hatami *et al.* [120] gave improved PRGs for branching programs and De Morgan formulas using variants of the IMZ framework. For branching programs, the improved seed length is $\sqrt{m} \cdot \text{polylog}(n/\varepsilon)$, which is close to the lack-of-lower-bounds barrier. See also the work of Cheraghchi *et al.* [63] for another variation on the IMZ framework.

Now let us get started proving the basic IMZ reduction (Theorem 5.38). The proof draws inspiration from the Nisan-Zuckerman generator (Section 3.4). The high-level intuition is that when we do a restriction, the restricted function cannot encode much information about the random bits we have used so far (since it can be succinctly described), and therefore we can use an extractor to recycle the random bits.

In detail, let $G\colon \{0,1\}^r \to \{0,1\}^n$ be a δ-PRG for $\mathcal{F}_{\text{simp}}$, let

$$\mathsf{Ext}\colon \{0,1\}^\ell \times \{0,1\}^d \to \{0,1\}^r$$

be an $(\ell - r, \delta)$-extractor, and let G' denote the following PRG.

1. Sample t independent copies $R^{(1)}, \ldots, R^{(t)}$ of the restriction R.
2. Sample $X, Y^{(1)}, \ldots, Y^{(t)}$ uniformly at random, let $Z^{(i)} = \mathsf{Ext}(X, Y^{(i)})$, and output
$$\sum_{i=1}^{t} R^{(i)} \circ G(Z^{(i)}),$$
where the sum is over $\mathbb{F}_2^n$.

Lemma 5.40 (Correctness of the IMZ reduction)**.** Under the assumptions of Theorem 5.38, the generator G' defined above fools $\mathcal{F}$ with error $O(t\delta)$.

Proof. The proof is a hybrid argument. Sample $U^{(1)}, \dots, U^{(t)} \in \{0,1\}^n$ independently and uniformly at random. Define hybrid distributions $H^{(0)}, \dots, H^{(t)}$ by

$$H^{(j)} = \left(\sum_{i=1}^{j} R^{(i)} \circ U^{(i)} \right) + \sum_{i=j+1}^{t} R^{(i)} \circ G(Z^{(i)}),$$

i.e., in the first j terms of the sum, we fill in the stars of $R^{(i)}$ using truly random bits instead of the pseudorandom bits $G(Z^{(i)})$. Fix some $f \in \mathcal{F}$ and some $j \in [t]$. Let us show that $\mathbb{E}[f(H^{(j-1)})] \approx \mathbb{E}[f(H^{(j)})]$.

There is a random variable B, independent of $U^{(j)}$ and $Y^{(j)}$, such that

$$H^{(j-1)} = B + R^{(j)} \circ G(Z^{(j)})$$
$$H^{(j)} = B + R^{(j)} \circ U^{(j)}.$$

(Note that B and X are *not* independent.) Define $f^{+B}(x) = f(x+B)$, define $F = f^{+B}|_{R^{(j)}}$, and define

$$F' = \begin{cases} F & \text{if } F \in \mathcal{F}_{\text{simp}} \\ \text{the 0 function} & \text{if } F \notin \mathcal{F}_{\text{simp}}. \end{cases}$$

By construction, $F' \in \mathcal{F}_{\text{simp}}$, so F' can be described using r bits. By Lemma 3.17, $\widetilde{\mathrm{H}}_{\min}(X \mid F') \geq \ell - r$. Therefore, by Lemma 3.18,

$$\mathrm{d}_{\mathrm{TV}}((Z^{(j)}, F'), (U, F')) \leq 3\delta,$$

where U is a uniform random r-bit string independent of F'. Applying a deterministic function can only make two distributions closer, so

$$|\,\mathbb{E}[F'(G(Z^{(j)}))] - \mathbb{E}[F'(G(U))]| \leq 3\delta.$$

Since $F' \in \mathcal{F}_{\text{simp}}$, the generator G fools F' with error δ, hence $|\,\mathbb{E}[F'(G(U))] - \mathbb{E}[F'(U^{(j)})]| \leq \delta$. Furthermore, with probability $1-\delta$, we have $F \equiv F'$. Therefore, overall,

$$|\,\mathbb{E}[F(G(Z^{(j)}))] - \mathbb{E}[F(U^{(j)})]| \leq 5\delta,$$

or equivalently,

$$|\,\mathbb{E}[f(H^{(j-1)})] - \mathbb{E}[f(H^{(j)})]| \leq 5\delta.$$

Clearly, $H^{(0)}$ is the output distribution of our PRG G'. By the triangle inequality, $H^{(0)}$ and $H^{(t)}$ are nearly indistinguishable to f. To complete the proof, let us show that $H^{(t)}$ is statistically close to uniform. Indeed, for each $j \in [n]$, we have

$$\Pr[\forall i \in [t],\ R_j^{(i)} \neq \star] \leq (1-p)^t \leq e^{-pt} \leq \delta/n.$$

Therefore, by the union bound, with probability at least $1-\delta$, we have $\bigcup_{i=1}^{t}(R^{(i)})^{-1}(\star) = [n]$. In this case, with repsect to the randomness of $U^{(1)}, \ldots, U^{(t)}$, the distribution $H^{(t)}$ is uniform. Therefore, overall, the total variation distance between $H^{(t)}$ and the uniform distribution is at most δ, and hence G' fools f with error $(5t+1)\delta$. □

To complete the proof of Theorem 5.38, we bound the seed length of G'.

Proof of Theorem 5.38. The restrictions $R^{(1)}, \ldots, R^{(t)}$ cost qt truly random bits in total. Using the GUV extractor (Theorem 3.16), the source length ℓ of the extractor Ext is $O(r)$, and its seed length is $d = O(\log(r/\delta))$. Therefore, the total seed length is $(q+d)t + \ell$, which is $O((q + \log(r/\delta)) \cdot t + r)$ as claimed. □

Acknowledgements

We thank Yevgeniy Dodis, Oded Goldreich, Avishay Tal, Salil Vadhan, Emanuele Viola, Avi Wigderson, David Zuckerman, and anonymous reviewers for helpful comments on drafts of this work.

Part of this work done while the second author was visiting the Simons Institute for the Theory of Computing. Part of this work was done while the second author was a graduate student at the University of Texas at Austin.

The version of record is available at: http://dx.doi.org/10.1561/0400000109

Appendices

A

Converse of the Sandwiching Lemma

Suppose we wish to show that every distribution that fools one class $\mathcal{F}_{\text{simp}}$ also fools another class $\mathcal{F}$. We presented two techniques for proving such a "transfer theorem":

1. The first technique is to express each $f \in \mathcal{F}$ as a linear combination of functions in $\mathcal{F}_{\text{simp}}$ and invoke the Triangle Inequality for PRG Errors.

2. The second technique is to sandwich each $f \in \mathcal{F}$ between functions in $\mathcal{F}_{\text{simp}}$ and invoke the Sandwiching Lemma.

As discussed in Section 2.5.1, we will now prove the following converse statement: If every distribution that fools $\mathcal{F}_{\text{simp}}$ also fools $\mathcal{F}$, then every $f \in \mathcal{F}$ is sandwiched between linear combinations of functions in $\mathcal{F}_{\text{simp}}$.

Theorem A.1 (Characterization of when fooling one class implies fooling another)**.** Let $n \in \mathbb{N}$, let $\mathcal{F}_{\text{simp}}$ be a finite class of functions $f\colon \{0,1\}^n \to \mathbb{R}$, and let $g\colon \{0,1\}^n \to \mathbb{R}$. Let $\varepsilon_0, \varepsilon > 0$ and suppose that every distribution X that fools $\mathcal{F}_{\text{simp}}$ with error ε_0 also fools g with error ε.

Then g is (2ε)-sandwiched between two functions $f_\ell, f_u \colon \{0,1\}^n \to \mathbb{R}$ of the form

$$f_\ell(x) = \lambda_\ell^{(0)} + \sum_{i=1}^{k_\ell} \lambda_\ell^{(i)} f_\ell^{(i)}(x) \tag{A.1}$$

$$f_u(x) = \lambda_u^{(0)} + \sum_{i=1}^{k_u} \lambda_u^{(i)} f_u^{(i)}(x), \tag{A.2}$$

where $k_\ell, k_u \in \mathbb{N}$, $\lambda_\ell^{(i)}, \lambda_u^{(i)} \in \mathbb{R}$, $f_\ell^{(i)}, f_u^{(i)} \in \mathcal{F}_{\text{simp}}$, and

$$\varepsilon_0 \cdot \sum_{i=1}^{k_\ell} |\lambda_\ell^{(i)}| \leq \varepsilon \tag{A.3}$$

$$\varepsilon_0 \cdot \sum_{i=1}^{k_u} |\lambda_u^{(i)}| \leq \varepsilon. \tag{A.4}$$

Conversely, if we start from the assumption that Equations (A.1) to (A.4) hold, then for any distribution X that fools $\mathcal{F}_{\text{simp}}$ with error ε_0, the Triangle Inequality for PRG Errors implies that X fools f_ℓ and f_u with error ε, and therefore the Sandwiching Lemma implies that X fools g with error 3ε. This recovers the assumption of Theorem A.1 up to a factor of three[1] in the error parameter. In this sense, Theorem A.1 shows that the Triangle Inequality for PRG Errors and the Sandwiching Lemma are "complete."

Before presenting the proof, let us elaborate on what the theorem says in two important special cases.

- Let $\mathcal{F}_{\text{simp}}$ be the class of Boolean k-juntas and let $\varepsilon_0 = 0$. Then Theorem A.1 says that a function is fooled by every k-wise uniform distribution if and only if the function can be sandwiched between two *low-degree real polynomials*. This was first shown by Bazzi [20] and, independently, by Benjamini *et al.* [26].

- Next, let $\mathcal{F}_{\text{simp}}$ to be the class of parity functions. Then Theorem A.1 essentially says that a function is fooled by every small-bias distribution if and only if the function can be sandwiched

[1] A more refined analysis, involving a more cumbersome version of the Sandwiching Lemma, gives a tight characterization without the extra factor of three.

between two functions with *low Fourier L_1 norm.*[2] This was first shown by De *et al.* [75].[3]

The general case seems to be folklore.

Proof of Theorem A.1. The proof uses linear programming duality. For each $f \in \mathcal{F}_{\text{simp}}$, define $\overline{f}\colon \{0,1\}^n \to \mathbb{R}$ by $\overline{f}(x) = f(x) - \mathbb{E}[f]$. Consider the following linear program in the variables $\{p_x\}_{x\in\{0,1\}^n}$:

$$\begin{aligned}
&\text{Maximize} \sum_{x\in\{0,1\}^n} p_x g(x),\\
&\text{subject to } p_x \geq 0 \text{ for all } x \in \{0,1\}^n\\
&\qquad \text{and} \sum_{x\in\{0,1\}^n} p_x = 1\\
&\qquad \text{and} \sum_{x\in\{0,1\}^n} p_x \overline{f}(x) \leq \varepsilon_0 \text{ for all } f \in \mathcal{F}_{\text{simp}}\\
&\qquad \text{and} -\sum_{x\in\{0,1\}^n} p_x \overline{f}(x) \leq \varepsilon_0 \text{ for all } f \in \mathcal{F}_{\text{simp}}.
\end{aligned}$$

The constraints say that the p_x variables are the probability mass function of some distribution that fools $\mathcal{F}_{\text{simp}}$ with error ε_0. The program is feasible, because if nothing else we can set $p_x = 2^{-n}$ (the uniform distribution). The objective function is the expectation of g under the distribution defined by the p_x variables, so the optimal value must be at most $\mathbb{E}[g] + \varepsilon$.

The dual linear program, in the variables z and $\{y_f^+, y_f^-\}_{f\in\mathcal{F}_{\text{simp}}}$, is as follows:

$$\begin{aligned}
&\text{Minimize } z + \varepsilon_0 \cdot \sum_{f\in\mathcal{F}_0} (y_f^+ + y_f^-),\\
&\text{subject to } y_f^+, y_f^- \geq 0 \text{ for all } f \in \mathcal{F}_{\text{simp}}\\
&\qquad \text{and } z + \sum_{f\in\mathcal{F}_0} \overline{f}(x) \cdot (y_f^+ - y_f^-) \geq g(x) \text{ for all } x \in \{0,1\}^n.
\end{aligned}$$

[2] Actually the quantity that matters is the sum of absolute values of the *nonempty* Fourier coefficients, whereas we included the empty Fourier coefficient in our definition of Fourier L_1 norm.

[3] Note that there is a minor mistake in the formulation by De *et al.* [75]: in their Proposition 2.7, the lower and upper sandwichers should be allowed to have different values of "l" and "δ."

By strong LP duality, the optimal value of this dual linear program is also at most $\mathbb{E}[g] + \varepsilon$. Observe that given a feasible solution to the dual linear program, if we subtract $\min\{y_f^+, y_f^-\}$ from y_f^+ and from y_f^-, then we get another feasible solution and the objective function can only decrease. Therefore, by setting $y_f = y_f^+ - y_f^-$, we obtain real numbers z^* and $\{y_f^*\}_{f \in \mathcal{F}_{\text{simp}}}$ such that

$$z^* + \varepsilon_0 \cdot \sum_{f \in \mathcal{F}_{\text{simp}}} |y_f^*| \leq \mathbb{E}[g] + \varepsilon, \text{ and}$$
$$z^* + \sum_{f \in \mathcal{F}_{\text{simp}}} \overline{f}(x) y_f^* \geq g(x) \text{ for all } x \in \{0,1\}^n.$$

Define

$$\begin{aligned} f_u(x) &= z^* + \sum_{f \in \mathcal{F}_{\text{simp}}} y_f^* \cdot \overline{f}(x) \\ &= \left(z^* - \sum_{f \in \mathcal{F}_{\text{simp}}} y_f^* \, \mathbb{E}[f] \right) + \sum_{f \in \mathcal{F}_{\text{simp}}} y_f^* \cdot f(x). \end{aligned}$$

Then f_u has the form given by Equation (A.2), and $f_u \geq g$. Furthermore, $\mathbb{E}[f_u] = z^*$, so

$$0 \leq \mathbb{E}[f_u - g] = z_* - \mathbb{E}[g] \leq \varepsilon - \varepsilon_0 \cdot \sum_{f \in \mathcal{F}_{\text{simp}}} |y_f^*|.$$

This shows that $\mathbb{E}[f_u - g] \leq \varepsilon$ and that Equation (A.4) holds.

Fooling g is equivalent to fooling $-g$, so the above also shows that there is some function f_ℓ of the form given by Equation (A.1) such that $-f_\ell \geq -g$, $\mathbb{E}[g - f_\ell] \leq \varepsilon$, and Equation (A.3) holds. Therefore, g is (2ε)-sandwiched between f_ℓ and f_u. □

B

List of PRGs

For reference, we conclude this text by listing the best explicit PRG constructions currently known for various models of computation, arranged by the model they fool. The list is not meant to be exhaustive; only a selection of important computational models are included. In each case, we only record a single state-of-the-art seed length, which in many cases is superior to the PRG constructions that we presented.

B.1 Circuit Models

In the list below, we use d to denote depth and m to denote size. Assume $d = O(1)$ and $m \geq n$.

- Conjunctions/disjunctions of literals
  - Seed length: $O(\log(1/\varepsilon) + \log\log n)$
  - Approach: k-wise δ-bias
  - Reference: Folklore
- $\mathbf{AC}^0$ circuits
  - Seed length: $\widetilde{O}(\log^{d-1} m \cdot \log(m/\varepsilon))$
  - Approach: Variant of the Ajtai-Wigderson framework
  - Reference: [166]
- Read-once CNFs/DNFs
  - Seed length: $O(\log n) + \widetilde{O}(\log(1/\varepsilon))$
  - Approach: Iterated restrictions with early termination
  - Reference: [81]
- Read-once $\mathbf{AC}^0$ formulas
  - Seed length: $\widetilde{O}(\log(n/\varepsilon))$
  - Approach: Iterated restrictions with early termination
  - References: [80], [82]
- De Morgan formulas
  - Seed length: $m^{1/3+o(1)} \cdot \mathrm{polylog}(1/\varepsilon)$
  - Approach: Variant of the IMZ framework
  - Reference: [120]
- Read-once De Morgan formulas
  - Seed length: $O(\log^2 n \cdot \log(n/\varepsilon))$
  - Approach: Iterated restrictions
  - Reference: [91]

B.2 Branching Program Models

In the list below, we use m to denote size and w to denote width. Assume $m \geq n$.

- Unrestricted branching programs
  - Seed length: $\sqrt{m} \cdot \text{polylog}(n/\varepsilon)$
  - Approach: Variant of the IMZ framework
  - Reference: [120]
- Width-2 branching programs that read d bits at a time
  - Seed length: $O(d \log n + d \cdot 2^d \cdot \log(m/\varepsilon))$
  - Approach: Sum of d δ-biased distributions
  - Reference: [29]
- Standard-order ROBPs with $w = 3$
  - Seed length: $\widetilde{O}(\log n \cdot \log(1/\varepsilon))$
  - Approach: Iterated restrictions with early termination
  - Reference: [171]
- Standard-order ROBPs with $4 \leq w \leq n$
  - Seed length: $O(\log(n/\varepsilon) \cdot \log n)$
  - Approach: Recycling seeds
  - References: [131], [181]
- Standard-order ROBPs with $w \gg n$
  - Seed length: $O\left(\frac{\log(w/\varepsilon) \cdot \log n}{\log \log w}\right)$
  - Approach: Recycling seeds
  - References: [13], [140]

- Standard-order regular ROBPs
  - Seed length: $\widetilde{O}(\log(w/\varepsilon) \cdot \log n)$
  - Approach: INW generator
  - Reference: [38]
- Standard-order permutation ROBPs with $w = O(1)$
  - Seed length: $O(\log n \cdot \log(1/\varepsilon))$
  - Approach: INW generator
  - References: [74], [145], [224]
- Arbitrary-order ROBPs
  - Seed length: $O(\log(wn/\varepsilon) \cdot \log^2 n)$
  - Approach: Iterated restrictions
  - Reference: [91]
- Arbitrary-order ROBPs with $w = O(1)$
  - Seed length: $\widetilde{O}(\log(n/\varepsilon) \cdot \log n)$
  - Approach: Iterated restrictions
  - Reference: [91]
- Arbitrary-order permutation ROBPs with $w = O(1)$
  - Seed length: $\widetilde{O}(\log n \cdot \log(1/\varepsilon))$
  - Approach: Polarizing random walks
  - Reference: [49]
- Decision trees, or more generally parity decision trees
  - Seed length: $O(\log(m/\varepsilon))$
  - Approach: δ-bias
  - Reference: [146]

B.3 Algebraic Models

- Parity functions
  - Seed length: $O(\log(n/\varepsilon))$
  - Approach: Balanced codes
  - References: [178], [217]
- Parities of at most k bits
  - Seed length: $O(\log(k/\varepsilon)) + \log\log n$
  - Approach: ε-biased seed for k-wise uniform generator
  - Reference: [178]
- Degree-d polynomials over $\mathbb{F}_2$
  - Seed length: $O(d\log n + d2^d \log(1/\varepsilon))$
  - Approach: Sum of d δ-biased distributions
  - Reference: [242]

B.4 Models Based on Locality

- $[-1, 1]$-valued k-juntas
  - Seed length: $O(k + \log(1/\varepsilon) + \log\log n)$
  - Approach: k-wise δ-bias
  - Reference: [178]
- Two-dimensional combinatorial rectangles
  - Seed length: $\frac{n}{2} + O(\log(1/\varepsilon))$
  - Approach: Random edge of expander
  - Reference: [131]
- d-dimensional combinatorial rectangles
  - Seed length: $\widetilde{O}(n/d + \log(1/\varepsilon) + \log\log n)$
  - Approach: Iterative alphabet reduction
  - Reference: [106]

- Two-party communication protocols with cost m
  - Seed length: $\frac{n}{2} + O(m + \log(1/\varepsilon))$
  - Approach: Random edge of expander
  - Reference: [131]

References

[1] S. Aaronson, "BQP and the polynomial hierarchy," in *Proc. 42nd Annual ACM Symposium on Theory of Computing (STOC)*, pp. 141–150, 2010. DOI: 10.1145/1806689.1806711.

[2] A. Ahmadinejad, J. Kelner, J. Murtagh, J. Peebles, A. Sidford, and S. Vadhan, "High-precision estimation of random walks in small space," in *Proc. 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 1295–1306, 2020. DOI: 10.1109/FOCS46700.2020.00123.

[3] M. Ajtai, J. Komlós, and E. Szemerédi, "Deterministic simulation in logspace," in *Proc. 19th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 132–140, 1987. DOI: 10.1145/28395.28410.

[4] M. Ajtai and A. Wigderson, "Deterministic simulation of probabilistic constant-depth circuits," *Advances in Computing Research – Randomness and Computation*, vol. 5, 1989, pp. 199–23.

[5] W. Alexi, B. Chor, O. Goldreich, and C. P. Schnorr, "RSA and Rabin functions: Certain parts are as hard as the whole," *SIAM J. Comput.*, vol. 17, no. 2, 1988, pp. 194–209. DOI: 10.1137/0217013.

[6] N. Alon, "Eigenvalues and expanders," *Combinatorica*, vol. 6, no. 2, 1986, pp. 83–96. DOI: 10.1007/BF02579166.

[7] N. Alon, "Explicit expanders of every degree and size," *Combinatorica*, vol. 41, no. 4, 2021, pp. 447–463. DOI: 10.1007/s00493-020-4429-x.

[8] N. Alon, L. Babai, and A. Itai, "A fast and simple randomized parallel algorithm for the maximal independent set problem," *J. Algorithms*, vol. 7, no. 4, 1986, pp. 567–583. DOI: 10.1016/0196-6774(86)90019-2.

[9] N. Alon, O. Goldreich, J. Håstad, and R. Peralta, "Simple constructions of almost k-wise independent random variables," *Random Structures & Algorithms*, vol. 3, no. 3, 1992, pp. 289–304. DOI: 10.1002/rsa.3240030308.

[10] E. Anand and C. Umans, "Pseudorandomness of the sticky random walk," *arXiv preprint arXiv:2307.11104*, 2023.

[11] A. E. Andreev, A. E. F. Clementi, and J. D. P. Rolim, "A new general derandomization method," *J. ACM*, vol. 45, no. 1, 1998, pp. 179–213. DOI: 10.1145/273865.273933.

[12] A. E. Andreev, A. E. F. Clementi, J. D. P. Rolim, and L. Trevisan, "Weak random sources, hitting sets, and BPP simulations," *SIAM J. Comput.*, vol. 28, no. 6, 1999, pp. 2103–2116. DOI: 10.1137/S0097539797325636.

[13] R. Armoni, "On the derandomization of space-bounded computations," in *Proc. 2nd International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, pp. 47–59, 1998. DOI: 10.1007/3-540-49543-6_5.

[14] R. Armoni, M. Saks, A. Wigderson, and S. Zhou, "Discrepancy sets and pseudorandom generators for combinatorial rectangles," in *Proc. 37th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 412–421, 1996. DOI: 10.1109/SFCS.1996.548500.

[15] S. Arora and B. Barak, *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009. DOI: 10.1017/CBO9780511804090.

[16] L. Babai, L. Fortnow, N. Nisan, and A. Wigderson, "BPP has subexponential time simulations unless EXPTIME has publishable proofs," *Comput. Complexity*, vol. 3, no. 4, 1993, pp. 307–318. DOI: 10.1007/BF01275486.

[17] L. Babai, P. Hajnal, E. Szemerédi, and G. Turán, "A lower bound for read-once-only branching programs," *J. Comput. System Sci.*, vol. 35, no. 2, 1987, pp. 153–162. DOI: 10.1016/0022-0000(87)90010-9.

[18] L. Babai, N. Nisan, and M. Szegedy, "Multiparty protocols, pseudorandom generators for logspace, and time-space trade-offs," *J. Comput. System Sci.*, vol. 45, no. 2, 1992, pp. 204–232. DOI: 10.1016/0022-0000(92)90047-M.

[19] D. A. Barrington, "Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1," *J. Comput. System Sci.*, vol. 38, no. 1, 1989, pp. 150–164. DOI: 10.1016/0022-0000(89)90037-8.

[20] L. M. J. Bazzi, "Polylogarithmic independence can fool DNF formulas," *SIAM J. Comput.*, vol. 38, no. 6, 2009, pp. 2220–2272. DOI: 10.1137/070691954.

[21] P. Beame, T. S. Jayram, and M. Saks, "Time-space tradeoffs for branching programs," *J. Comput. System Sci.*, vol. 63, no. 4, 2001, pp. 542–572. DOI: 10.1006/jcss.2001.1778.

[22] P. Beame, V. Liew, and M. Pătraşcu, "Finding the median (obliviously) with bounded space," in *Proc. 42nd International Colloquium on Automata, Languages and Programming (ICALP)*, pp. 103–115, 2015. DOI: 10.1007/978-3-662-47672-7_9.

[23] R. Beigel, N. Reingold, and D. A. Spielman, "The perceptron strikes back," in *Proc. 6th Annual IEEE Conference on Structure in Complexity Theory*, pp. 286, 287, 288, 289, 290, 291, Jul. 1991. DOI: 10.1109/SCT.1991.160270.

[24] M. Bellare and J. Rompel, "Randomness-efficient oblivious sampling," in *Proc. 35th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 276–287, 1994. DOI: 10.1109/SFCS.1994.365687.

[25] A. Ben-Aroya and A. Ta-Shma, "A combinatorial construction of almost-Ramanujan graphs using the zig-zag product," *SIAM J. Comput.*, vol. 40, no. 2, 2011, pp. 267–290. DOI: 10.1137/080732651.

[26] I. Benjamini, O. Gurel-Gurevich, and R. Peled, "On k-wise independent distributions and boolean functions," *arXiv:1201.3261*, 2012.

[27] L. Blum, M. Blum, and M. Shub, "A simple unpredictable pseudorandom number generator," *SIAM J. Comput.*, vol. 15, no. 2, 1986, pp. 364–383. DOI: 10.1137/0215025.

[28] M. Blum and S. Micali, "How to generate cryptographically strong sequences of pseudorandom bits," *SIAM J. Comput.*, vol. 13, no. 4, 1984, pp. 850–864. DOI: 10.1137/0213053.

[29] A. Bogdanov, Z. Dvir, E. Verbin, and A. Yehudayoff, "Pseudorandomness for width-2 branching programs," *Theory of Computing*, vol. 9, 2013, pp. 283–293. DOI: 10.4086/toc.2013.v009a007.

[30] A. Bogdanov, W. M. Hoza, G. Prakriya, and E. Pyne, "Hitting Sets for Regular Branching Programs," in *Proc. 37th Computational Complexity Conference (CCC)*, 3:1–3:22, 2022. DOI: 10.4230/LIPIcs.CCC.2022.3.

[31] A. Bogdanov, P. A. Papakonstantinou, and A. Wan, "Pseudorandomness for read-once formulas," in *FOCS*, R. Ostrovsky, Ed., pp. 240–246, IEEE, 2011.

[32] A. Bogdanov and E. Viola, "Pseudorandom bits for polynomials," *SIAM J. Comput.*, vol. 39, no. 6, 2010, pp. 2464–2486. DOI: 10.1137/070712109.

[33] R. B. Boppana, "The average sensitivity of bounded-depth circuits," *Inf. Process. Lett.*, vol. 63, no. 5, 1997, pp. 257–261. DOI: 10.1016/S0020-0190(97)00131-2.

[34] C. Bordenave, "A new proof of Friedman's second eigenvalue theorem and its extension to random lifts," *Ann. Sci. Éc. Norm. Supér. (4)*, vol. 53, no. 6, 2020, pp. 1393–1439. DOI: 10.24033/asens.245.

[35] A. Borodin, D. Dolev, F. E. Fich, and W. Paul, "Bounds for width two branching programs," *SIAM J. Comput.*, vol. 15, no. 2, 1986, pp. 549–560. DOI: 10.1137/0215040.

[36] M. Braverman, "Polylogarithmic independence fools AC^0 circuits," *J. ACM*, vol. 57, no. 5, 2010, 28:1–28:10. DOI: 10.1145/1754399.1754401.

[37] M. Braverman, G. Cohen, and S. Garg, "Pseudorandom pseudo-distributions with near-optimal error for read-once branching programs," *SIAM J. Comput.*, vol. 49, no. 5, 2020, STOC18-242–STOC18-299. DOI: 10.1137/18M1197734.

[38] M. Braverman, A. Rao, R. Raz, and A. Yehudayoff, "Pseudorandom generators for regular branching programs," *SIAM J. Comput.*, vol. 43, no. 3, 2014, pp. 973–986. DOI: 10.1137/120875673.

[39] J. Brody and E. Verbin, "The coin problem and pseudorandomness for branching programs," in *Proc. 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 30–39, 2010. DOI: 10.1109/FOCS.2010.10.

[40] H. Buhrman and L. Fortnow, "One-sided versus two-sided error in probabilistic computation," in *Proc. 16th Symposium on Theoretical Aspects of Computer Science (STACS)*, pp. 100–109, 1999. DOI: 10.1007/3-540-49116-3_9.

[41] J.-Y. Cai, A. Nerurkar, and D. Sivakumar, "Hardness and hierarchy theorems for probabilistic quasi-polynomial time," in *Proc. 31st Annual ACM Symposium on Theory of Computing (STOC)*, pp. 726–735, 1999. DOI: 10.1145/301250.301444.

[42] M. L. Carmosino, R. Impagliazzo, and M. Sabin, "Fine-Grained Derandomization: From Problem-Centric to Resource-Centric Complexity," in *Proc. 45th International Colloquium on Automata, Languages and Programming (ICALP)*, 27:1–27:16, 2018. DOI: 10.4230/LIPIcs.ICALP.2018.27.

[43] L. E. Celis, O. Reingold, G. Segev, and U. Wieder, "Balls and bins: Smaller hash families and faster evaluation," *SIAM J. Comput.*, vol. 42, no. 3, 2013, pp. 1030–1050. DOI: 10.1137/120871626.

[44] S. Chari, P. Rohatgi, and A. Srinivasan, "Improved algorithms via approximations of probability distributions," *J. Comput. System Sci.*, vol. 61, no. 1, 2000, pp. 81–107. DOI: 10.1006/jcss.1999.1695.

[45] E. Chattopadhyay, "Pseudorandomness and combinatorial constructions," 2018. URL: https://courses.cs.cornell.edu/cs6815/2018fa/.

[46] E. Chattopadhyay, "Pseudorandomness and combinatorial constructions," 2019. URL: https://courses.cs.cornell.edu/cs6815/2019fa/.

[47] E. Chattopadhyay, "Pseudorandomness and combinatorial constructions," 2022. URL: https://courses.cs.cornell.edu/cs6815/2022fa/.

[48] E. Chattopadhyay, J. Gaitonde, C. H. Lee, S. Lovett, and A. Shetty, "Fractional Pseudorandom Generators from Any Fourier Level," in *Proc. 36th Computational Complexity Conference (CCC)*, 10:1–10:24, 2021. DOI: 10.4230/LIPIcs.CCC.2021.10.

[49] E. Chattopadhyay, P. Hatami, K. Hosseini, and S. Lovett, "Pseudorandom generators from polarizing random walks," *Theory Comput.*, vol. 15, no. 1, 2019, pp. 1–26. DOI: 10.4086/toc.2019.v015a010.

[50] E. Chattopadhyay, P. Hatami, S. Lovett, and A. Tal, "Pseudorandom Generators from the Second Fourier Level and Applications to AC0 with Parity Gates," in *Proc. 10th Conference on Innovations in Theoretical Computer Science (ITCS)*, 22:1–22:15, 2018. DOI: 10.4230/LIPIcs.ITCS.2019.22.

[51] E. Chattopadhyay, P. Hatami, O. Reingold, and A. Tal, "Improved pseudorandomness for unordered branching programs through local monotonicity," in *Proc. 50th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 363–375, 2018. DOI: 10.1145/3188745.3188800.

[52] E. Chattopadhyay and J.-J. Liao, "Optimal error pseudodistributions for read-once branching programs," in *Proc. 35th Annual IEEE Conference on Computational Complexity (CCC)*, vol. 169, 25:1–25:27, 2020. DOI: 10.4230/LIPIcs.CCC.2020.25.

[53] E. Chattopadhyay and J.-J. Liao, *Recursive error reduction for regular branching programs*, ECCC preprint TR23-130, 2023. URL: https://eccc.weizmann.ac.il/report/2023/130/.

[54] L. Chen, W. M. Hoza, X. Lyu, A. Tal, and H. Wu, *Weighted pseudorandom generators via inverse analysis of random walks and shortcutting*, ECCC preprint TR23-114, 2023. URL: https://eccc.weizmann.ac.il/report/2023/114/.

[55] L. Chen, Z. Lu, X. Lyu, and I. C. Oliveira, "Majority vs. Approximate Linear Sum and Average-Case Complexity Below NC^1," in *Proc. 48th International Colloquium on Automata, Languages and Programming (ICALP)*, 51:1–51:20, 2021. DOI: 10.4230/LIPIcs.ICALP.2021.51.

[56] L. Chen, X. Lyu, A. Tal, and H. Wu, "New PRGs for Unbounded-Width/Adaptive-Order Read-Once Branching Programs," in *Proc. 50th International Colloquium on Automata, Languages and Programming (ICALP)*, vol. 261, 39:1–39:20, 2023. DOI: 10.4230/LIPIcs.ICALP.2023.39.

[57] L. Chen, X. Lyu, and R. R. Williams, "Almost-everywhere circuit lower bounds from non-trivial derandomization," in *Proc. 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 1–12, 2020. DOI: 10.1109/FOCS46700.2020.00009.

[58] L. Chen and H. Ren, "Strong average-case circuit lower bounds from nontrivial derandomization," *SIAM J. Comput.*, vol. 51, no. 3, 2022, STOC20-115–STOC20-173. DOI: 10.1137/20M1364886.

[59] L. Chen, R. D. Rothblum, R. Tell, and E. Yogev, "On exponential-time hypotheses, derandomization, and circuit lower bounds: Extended abstract," in *Proc. 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 13–23, 2020. DOI: 10.1109/FOCS46700.2020.00010.

[60] L. Chen and R. Tell, "Simple and fast derandomization from very hard functions: Eliminating randomness at almost no cost," in *Proc. 53rd Annual ACM Symposium on Theory of Computing (STOC)*, pp. 283–291, 2021. DOI: 10.1145/3406325.3451059.

[61] K. Cheng and W. M. Hoza, "Hitting sets give two-sided derandomization of small space," *Theory of Computing*, vol. 18, no. 21, 2022, pp. 1–32. DOI: 10.4086/toc.2022.v018a021.

[62] K. Cheng and X. Li, "Efficient document exchange and error correcting codes with asymmetric information," in *Proc. 2021 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 2424–2443, 2021. DOI: 10.1137/1.9781611976465.144.

[63] M. Cheraghchi, V. Kabanets, Z. Lu, and D. Myrisiotis, "Circuit lower bounds for MCSP from local pseudorandom generators," *ACM Trans. Comput. Theory*, vol. 12, no. 3, 2020, Art. 21, 27. DOI: 10.1145/3404860.

[64] R. Chiclana and Y. Peres, "A local central limit theorem for random walks on expander graphs," *arXiv preprint arXiv:2212.00958*, 2022.

[65] B. Chor and O. Goldreich, "Unbiased bits from sources of weak randomness and probabilistic communication complexity," *SIAM J. on Computing*, vol. 17, no. 2, 1988, pp. 230–261. DOI: 10.1137/0217015.

[66] B. Chor, O. Goldreich, J. Håstad, J. Friedman, S. Rudich, and R. Smolensky, "The bit extraction problem or t-resilient functions," in *Proc. 26th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 396–407, 1985. DOI: 10.1109/SFCS.1985.55.

[67] S. M. Cioabă and M. R. Murty, "Expander graphs and gaps between primes," *Forum Math.*, vol. 20, no. 4, 2008, pp. 745–756. DOI: 10.1515/FORUM.2008.035.

[68] T. H. Click, A. Liu, and G. A. Kaminski, "Quality of random number generators significantly affects results of monte carlo simulations for organic and biological systems," *Journal of Computational Chemistry*, vol. 32, no. 3, 2011, pp. 513–524. DOI: 10.1002/jcc.21638.

[69] P. Coddington, "Analysis of random number generators using monte carlo simulation," *International Journal of Modern Physics C*, vol. 05, no. 03, 1994, pp. 547–560. DOI: 10.1142/S0129183194000726.

[70] G. Cohen, D. Doron, O. Renard, O. Sberlo, and A. Ta-Shma, "Error reduction for weighted PRGs against read once branching programs," in *Proc. 36th Computational Complexity Conference (CCC)*, 22:1–22:17, 2021. DOI: 10.4230/LIPIcs.CCC.2021.22.

[71] G. Cohen, D. Minzer, S. Peleg, A. Potechin, and A. Ta-Shma, "Expander Random Walks: The General Case and Limitations," in *Proc. 49th International Colloquium on Automata, Languages and Programming (ICALP)*, 43:1–43:18, 2022. DOI: 10.4230/LIPIcs.ICALP.2022.43.

[72] G. Cohen, N. Peri, and A. Ta-Shma, "Expander random walks: A Fourier-analytic approach," in *Proc. 53rd Annual ACM Symposium on Theory of Computing (STOC)*, pp. 1643–1655, ACM, New York, 2021. DOI: 10.1145/3406325.3451049.

[73] M. B. Cohen, "Ramanujan graphs in polynomial time," in *Proc. 57th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 276–281, 2016. DOI: 10.1109/FOCS.2016.37.

[74] A. De, "Pseudorandomness for permutation and regular branching programs," in *Proc. 26th Annual IEEE Conference on Computational Complexity (CCC)*, pp. 221–231, 2011. DOI: 10.1109/CCC.2011.23.

[75] A. De, O. Etesami, L. Trevisan, and M. Tulsiani, "Improved pseudorandom generators for depth 2 circuits," in *Proc. 14th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, pp. 504–517, 2010. DOI: 10.1007/978-3-642-15369-3_38.

[76] R. De Wolf, "A brief introduction to fourier analysis on the boolean cube," *Theory of Computing*, 2008, pp. 1–20.

[77] A. Degwekar, V. Vaikuntanathan, and P. N. Vasudevan, "Fine-grained cryptography," in *Proc. 36th Annual International Cryptology Conference (CRYPTO)*, pp. 533–562, 2016. DOI: 10.1007/978-3-662-53015-3_19.

[78] I. Diakonikolas, P. Gopalan, R. Jaiswal, R. A. Servedio, and E. Viola, "Bounded independence fools halfspaces," *SIAM J. Comput.*, vol. 39, no. 8, 2010, pp. 3441–3462. DOI: 10.1137/100783030.

[79] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," *SIAM J. Comput.*, vol. 38, no. 1, 2008, pp. 97–139. DOI: 10.1137/060651380.

[80] D. Doron, P. Hatami, and W. M. Hoza, "Near-Optimal Pseudorandom Generators for Constant-Depth Read-Once Formulas," in *Proc. 34th Computational Complexity Conference (CCC)*, 16:1–16:34, 2019. DOI: 10.4230/LIPIcs.CCC.2019.16.

[81] D. Doron, P. Hatami, and W. M. Hoza, "Log-Seed Pseudorandom Generators via Iterated Restrictions," in *Proc. 35th Computational Complexity Conference (CCC)*, 6:1–6:36, 2020. DOI: 10.4230/LIPIcs.CCC.2020.6.

[82] D. Doron, R. Meka, O. Reingold, A. Tal, and S. Vadhan, "Pseudorandom Generators for Read-Once Monotone Branching Programs," in *Proc. 25th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, 58:1–58:21, 2021. DOI: 10.4230/LIPIcs.APPROX/RANDOM.2021.58.

[83] D. Doron, D. Moshkovitz, J. Oh, and D. Zuckerman, "Nearly optimal pseudorandomness from hardness," in *Proc. 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 1057–1068, 2020. DOI: 10.1109/FOCS46700.2020.00102.

[84] S. Egashira, Y. Wang, and K. Tanaka, "Fine-grained cryptography revisited," *J. Cryptology*, vol. 34, no. 3, 2021, Paper No. 23, 43. DOI: 10.1007/s00145-021-09390-3.

[85] P. Erdős, P. Frankl, and Z. Füredi, "Families of finite sets in which no set is covered by the union of r others," *Israel J. Math.*, vol. 51, no. 1-2, 1985, pp. 79–89. DOI: 10.1007/BF02772959.

[86] G. Even, O. Goldreich, M. Luby, N. Nisan, and B. Veličković, "Efficient approximation of product distributions," *Random Structures Algorithms*, vol. 13, no. 1, 1998, pp. 1–16. DOI: 10.1002/(SICI)1098-2418(199808)13:1<1::AID-RSA1>3.0.CO;2-W.

[87] B. Fefferman, R. Shaltiel, C. Umans, and E. Viola, "On beating the hybrid argument," *Theory Comput.*, vol. 9, 2013, pp. 809–843. DOI: 10.4086/toc.2013.v009a026.

[88] A. M. Ferrenberg, D. P. Landau, and Y. J. Wong, "Monte carlo simulations: Hidden errors from 'good' random number generators," *Phys. Rev. Lett.*, vol. 69, 23 Dec. 1992, pp. 3382–3384. DOI: 10.1103/PhysRevLett.69.3382.

[89] T. Filk, M. Marcu, and K. Fredenhagen, "Long range correlations in random number generators and their influence on monte carlo simulations," *Physics Letters B*, vol. 165, no. 1, 1985, pp. 125–130. DOI: 10.1016/0370-2693(85)90705-1.

[90] Y. Filmus, "Smolensky's lower bound," 2010. URL: https://yuvalfilmus.cs.technion.ac.il/Manuscripts/Smolensky.pdf.

[91] M. A. Forbes and Z. Kelley, "Pseudorandom generators for read-once branching programs, in any order," in *Proc. 59th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 946–955, 2018. DOI: 10.1109/FOCS.2018.00093.

[92] J. Friedman, "Some geometric aspects of graphs and their eigenfunctions," *Duke Math. J.*, vol. 69, no. 3, 1993, pp. 487–525. DOI: 10.1215/S0012-7094-93-06921-9.

[93] J. Friedman, "A proof of Alon's second eigenvalue conjecture and related problems," *Mem. Amer. Math. Soc.*, vol. 195, no. 910, 2008, pp. viii+100. DOI: 10.1090/memo/0910.

[94] Z. Füredi, "Matchings and covers in hypergraphs," *Graphs Combin.*, vol. 4, no. 2, 1988, pp. 115–206. DOI: 10.1007/BF01864160.

[95] O. Goldreich and L. A. Levin, "A hard-core predicate for all one-way functions," in *Proc. 21st Annual ACM Symposium on Theory of Computing (STOC)*, pp. 25–32, 1989. DOI: 10.1145/73007.73010.

[96] O. Goldreich, *Foundations of Cryptography Volume I: Basic Tools*. Cambridge University Press, 2001. DOI: 10.1017/CBO9780511546891.

[97] O. Goldreich, *A primer on pseudorandom generators*, vol. 55, ser. University Lecture Series. American Mathematical Society, Providence, RI, 2010, pp. x+114. DOI: 10.1090/ulect/055.

[98] O. Goldreich, "In a world of $\mathbf{P} = \mathbf{BPP}$," in *Studies in complexity and cryptography*, ser. Lecture Notes in Comput. Sci. Vol. 6650, Springer, Heidelberg, 2011, pp. 191–232. DOI: 10.1007/978-3-642-22670-0_20.

[99] O. Goldreich, H. Krawczyk, and M. Luby, "On the existence of pseudorandom generators," *SIAM J. Comput.*, vol. 22, no. 6, 1993, pp. 1163–1175. DOI: 10.1137/0222069.

[100] O. Goldreich, S. Vadhan, and A. Wigderson, "Simplified derandomization of BPP using a hitting set generator," in *Studies in Complexity and Cryptography*, ser. Lecture Notes in Computer Science, vol. 6650, Springer, Heidelberg, 2011, pp. 59–67. DOI: 10.1007/978-3-642-22670-0_8.

[101] S. Goldwasser, S. Micali, and P. Tong, "Why and how to establish a private code on a public network," in *Proc. 23rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 134–144, 1982. DOI: 10.1109/SFCS.1982.100.

[102] L. Golowich, "A new Berry-Esseen theorem for expander walks," in *Proc. 55th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 10–22, ACM, New York, 2023. DOI: 10.1145/3564246.3585141.

[103] L. Golowich and S. Vadhan, "Pseudorandomness of Expander Random Walks for Symmetric Functions and Permutation Branching Programs," in *Proc. 37th Computational Complexity Conference (CCC)*, 27:1–27:13, 2022. DOI: 10.4230/LIPIcs.CCC.2022.27.

[104] P. Gopalan, D. M. Kane, and R. Meka, "Pseudorandomness via the discrete Fourier transform," *SIAM J. Comput.*, vol. 47, no. 6, 2018, pp. 2451–2487. DOI: 10.1137/16M1062132.

[105] P. Gopalan, R. Meka, O. Reingold, L. Trevisan, and S. Vadhan, "Better pseudorandom generators from milder pseudorandom restrictions," in *Proc. 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 120–129, 2012. DOI: 10.1109/FOCS.2012.77.

[106] P. Gopalan and A. Yehudayoff, "Concentration for limited independence via inequalities for the elementary symmetric polynomials," *Theory Comput.*, vol. 16, 2020, Paper No. 17, 29. DOI: 10.4086/toc.2020.v016a017.

[107] P. Grassberger, "On correlations in 'good' random number generators," *Physics Letters A*, vol. 181, no. 1, 1993, pp. 43–46. DOI: 10.1016/0375-9601(93)91122-L.

[108] V. Guruswami and V. M. Kumar, "Pseudobinomiality of the Sticky Random Walk," in *Proc. 12th Conference on Innovations in Theoretical Computer Science (ITCS)*, vol. 185, 2021. DOI: 10.4230/LIPIcs.ITCS.2021.48.

[109] V. Guruswami, C. Umans, and S. Vadhan, "Unbalanced expanders and randomness extractors from Parvaresh-Vardy codes," *J. ACM*, vol. 56, no. 4, 2009, Art. 20, 34. DOI: 10.1145/1538902.1538904.

[110] I. Haitner, D. Harnik, and O. Reingold, "Efficient pseudorandom generators from exponentially hard one-way functions," in *Proc. 33rd International Colloquium on Automata, Languages and Programming (ICALP)*, pp. 228–239, 2006. DOI: 10.1007/11787006_20.

[111] I. Haitner, D. Harnik, and O. Reingold, "On the power of the randomized iterate," *SIAM J. Comput.*, vol. 40, no. 6, 2011, pp. 1486–1528. DOI: 10.1137/080721820.

[112] I. Haitner, O. Reingold, and S. Vadhan, "Efficiency improvements in constructing pseudorandom generators from one-way functions," *SIAM J. Comput.*, vol. 42, no. 3, 2013, pp. 1405–1430. DOI: 10.1137/100814421.

[113] E. Haramaty, C. H. Lee, and E. Viola, "Bounded independence plus noise fools products," *SIAM J. Comput.*, vol. 47, no. 2, 2018, pp. 493–523. DOI: 10.1137/17M1129088.

[114] P. Harsha and S. Srinivasan, "On polynomial approximations to AC^0," *Random Structures Algorithms*, vol. 54, no. 2, 2019, pp. 289–303. DOI: 10.1002/rsa.20786.

[115] T. Hartman and R. Raz, "On the distribution of the number of roots of polynomials and explicit weak designs," *Random Structures Algorithms*, vol. 23, no. 3, 2003, pp. 235–263. DOI: 10.1002/rsa.10095.

[116] J. Hastad, "Almost optimal lower bounds for small depth circuits," *Adv. Comput. Res.*, vol. 5, 1989, pp. 143–170. URL: https://www.csc.kth.se/~johanh/largesmalldepth.pdf.

[117] J. Håstad, "A slight sharpening of lmn," *Journal of Computer and System Sciences*, vol. 63, no. 3, 2001, pp. 498–508. DOI: 10.1006/jcss.2001.1803.

[118] J. Håstad, "On the correlation of parity and small-depth circuits," *SIAM J. Comput.*, vol. 43, no. 5, 2014, pp. 1699–1708. DOI: 10.1137/120897432.

[119] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby, "A pseudorandom generator from any one-way function," *SIAM J. Comput.*, vol. 28, no. 4, 1999, pp. 1364–1396. DOI: 10.1137/S0097539793244708.

[120] P. Hatami, W. M. Hoza, A. Tal, and R. Tell, "Fooling constant-depth threshold circuits," in *Proc. 62nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 104–115, 2022. DOI: 10.1109/FOCS52979.2021.00019.

[121] A. Healy, S. Vadhan, and E. Viola, "Using nondeterminism to amplify hardness," *SIAM Journal on Computing*, vol. 35, no. 4, 2006, pp. 903–931. DOI: 10.1137/S0097539705447281.

[122] T. Holenstein, "Pseudorandom generators from one-way functions: A simple construction for any hardness," in *Theory of cryptography*, ser. Lecture Notes in Comput. Sci. Vol. 3876, Springer, Berlin, 2006, pp. 443–461. DOI: 10.1007/11681878_23.

[123] S. Hoory, N. Linial, and A. Wigderson, "Expander graphs and their applications," *Bull. Amer. Math. Soc. (N.S.)*, vol. 43, no. 4, 2006, pp. 439–561. DOI: 10.1090/S0273-0979-06-01126-8.

[124] W. M. Hoza, "Better pseudodistributions and derandomization for space-bounded computation," in *Proc. 25th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, 28:1–28:23, 2021. DOI: 10.4230/LIPIcs.APPROX/RANDOM.2021.28.

[125] W. M. Hoza, "Recent progress on derandomizing space-bounded computation," *Bulletin of the EATCS*, no. 138, 2022, pp. 114–143. URL: https://eatcs.org/images/bulletin/beatcs138.pdf.

[126] W. M. Hoza, *A technique for hardness amplification against* AC^0, ECCC preprint TR23-176, 2023. URL: https://eccc.weizmann.ac.il/report/2023/176/.

[127] W. M. Hoza, E. Pyne, and S. Vadhan, "Pseudorandom Generators for Unbounded-Width Permutation Branching Programs," in *Proc. 12th Conference on Innovations in Theoretical Computer Science (ITCS)*, 7:1–7:20, 2021. DOI: 10.4230/LIPIcs.ITCS.2021.7.

[128] W. M. Hoza and D. Zuckerman, "Simple optimal hitting sets for small-success **RL**," *SIAM J. Comput.*, vol. 49, no. 4, 2020, pp. 811–820. DOI: 10.1137/19M1268707.

[129] R. Impagliazzo, W. Matthews, and R. Paturi, "A satisfiability algorithm for AC^0," in *Proc. 23rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 961–972, 2012. URL: https://dl.acm.org/doi/10.5555/2095116.2095193.

[130] R. Impagliazzo, R. Meka, and D. Zuckerman, "Pseudorandomness from shrinkage," *J. ACM*, vol. 66, no. 2, 2019, Art. 11, 16. DOI: 10.1145/3230630.

[131] R. Impagliazzo, N. Nisan, and A. Wigderson, "Pseudorandomness for network algorithms," in *Proc. 26th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 356–364, 1994. DOI: 10.1145/195058.195190.

[132] R. Impagliazzo, R. Shaltiel, and A. Wigderson, "Near-optimal conversion of hardness into pseudo-randomness," in *Proc. 40th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 181–190, 1999. DOI: 10.1109/SFFCS.1999.814590.

[133] R. Impagliazzo, R. Shaltiel, and A. Wigderson, "Reducing the seed length in the Nisan-Wigderson generator," *Combinatorica*, vol. 26, no. 6, 2006, pp. 647–681. DOI: 10.1007/s00493-006-0036-8.

[134] R. Impagliazzo and A. Wigderson, "P = BPP if E requires exponential circuits: Derandomizing the XOR lemma," in *Proc. 29th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 220–229, 1997. DOI: 10.1145/258533.258590.

[135] R. Impagliazzo and A. Wigderson, "Randomness vs time: Derandomization under a uniform assumption," *J. Comput. System Sci.*, vol. 63, no. 4, 2001, pp. 672–688. DOI: 10.1006/jcss.2001.1780.

[136] P. Indyk, "Stable distributions, pseudorandom generators, embeddings, and data stream computation," *J. ACM*, vol. 53, no. 3, 2006, pp. 307–323. DOI: 10.1145/1147954.1147955.

[137] V. Kabanets and J.-Y. Cai, "Circuit minimization problem," in *Proc. 32nd Annual ACM Symposium on Theory of Computing (STOC)*, pp. 73–79, 2000. DOI: 10.1145/335305.335314.

[138] V. Kabanets and Z. Lu, "Satisfiability and Derandomization for Small Polynomial Threshold Circuits," in *Proc. 22nd International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, 46:1–46:19, 2018. DOI: 10.4230/LIPIcs.APPROX-RANDOM.2018.46.

[139] C. Kalle and S. Wansleben, "Problems with the random number generator ranf implemented on the cdc cyber 205," *Computer Physics Communications*, vol. 33, no. 4, 1984, pp. 343–346. DOI: 10.1016/0010-4655(84)90139-5.

[140] D. M. Kane, J. Nelson, and D. P. Woodruff, *Revisiting norm estimation in data streams*, 2008. arXiv: 0811.3648 [cs.DS].

[141] R. M. Karp and R. J. Lipton, "Some connections between nonuniform and uniform complexity classes," in *Proc. 12th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 302–309, 1980. DOI: 10.1145/800141.804678.

[142] Z. Kelley, "An improved derandomization of the switching lemma," in *Proc. 53rd Annual ACM Symposium on Theory of Computing (STOC)*, pp. 272–282, 2021. DOI: 10.1145/3406325.3451054.

[143] A. R. Klivans, H. K. Lee, and A. Wan, "Mansour's conjecture is true for random DNF formulas," in *Proc. 23rd Conference on Learning Theory (COLT)*, pp. 368–380, 2010. URL: http://www.learningtheory.org/colt2010/papers/085Lee.pdf.

[144] A. R. Klivans and D. van Melkebeek, "Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses," *SIAM J. Comput.*, vol. 31, no. 5, 2002, pp. 1501–1526. DOI: 10.1137/S0097539700389652.

[145] M. Koucký, P. Nimbhorkar, and P. Pudlák, "Pseudorandom generators for group products," in *Proc. 43rd Annual ACM Symposium on Theory of Computing (STOC)*, pp. 263–272, 2011. DOI: 10.1145/1993636.1993672.

[146] E. Kushilevitz and Y. Mansour, "Learning decision trees using the Fourier spectrum," *SIAM J. Comput.*, vol. 22, no. 6, 1993, pp. 1331–1348. DOI: 10.1137/0222080.

[147] P. L'Ecuyer and R. Simard, "Testu01: A c library for empirical testing of random number generators," *ACM Trans. Math. Softw.*, vol. 33, no. 4, Aug. 2007. DOI: 10.1145/1268776.1268777.

[148] C. H. Lee, "Fourier bounds and pseudorandom generators for product tests," in *Proc. 34th Computational Complexity Conference (CCC)*, 7:1–7:25, 2019. DOI: 10.4230/LIPIcs.CCC.2019.7.

[149] C. H. Lee, E. Pyne, and S. Vadhan, "Fourier Growth of Regular Branching Programs," in *Proc. 26th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, 2:1–2:21, 2022. DOI: 10.4230/LIPIcs.APPROX/RANDOM.2022.2.

[150] C. H. Lee, E. Pyne, and S. Vadhan, "On the Power of Regular and Permutation Branching Programs," in *Proc. 27th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, 44:1–44:22, 2023. DOI: 10.4230/LIPIcs.APPROX/RANDOM.2023.44.

[151] C. H. Lee and E. Viola, "Some limitations of the sum of small-bias distributions," *Theory Comput.*, vol. 13, 2017, Paper No. 16, 23. DOI: 10.4086/toc.2017.v013a016.

[152] C. H. Lee and E. Viola, "More on bounded independence plus noise: Pseudorandom generators for read-once polynomials," *Theory Comput.*, vol. 16, 2020, Paper No. 7, 50. DOI: 10.4086/toc.2020.v016a007.

[153] L. A. Levin, "One way functions and pseudorandom generators," *Combinatorica*, vol. 7, no. 4, 1987, pp. 357–363. DOI: 10.1007/BF02579323.

[154] N. Linial, M. Luby, M. Saks, and D. Zuckerman, "Efficient construction of a small hitting set for combinatorial rectangles in high dimension," *Combinatorica*, vol. 17, no. 2, 1997, pp. 215–234. DOI: 10.1007/BF01200907.

[155] N. Linial, Y. Mansour, and N. Nisan, "Constant depth circuits, Fourier transform, and learnability," *Journal of the ACM*, vol. 40, no. 3, 1993, pp. 607–620. DOI: 10.1145/174130.174138.

[156] N. Linial and N. Nisan, "Approximate inclusion-exclusion," *Combinatorica*, vol. 10, no. 4, 1990, pp. 349–365. DOI: 10.1007/BF02128670.

[157] S. Lovett, "Unconditional pseudorandom generators for low degree polynomials," *Theory of Computing*, vol. 5, no. 1, 2009, pp. 69–82. DOI: 10.4086/toc.2009.v005a003.

[158] S. Lovett and S. Srinivasan, "Correlation bounds for poly-size AC^0 circuits with $n^{1-o(1)}$ symmetric gates," in *Proc. 15th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, pp. 640–651, 2011. DOI: 10.1007/978-3-642-22935-0_54.

[159] C.-J. Lu, "Improved pseudorandom generators for combinatorial rectangles," *Combinatorica*, vol. 22, no. 3, 2002, pp. 417–433. DOI: 10.1007/s004930200021.

[160] C.-J. Lu, S.-C. Tsai, and H.-L. Wu, "Improved hardness amplification in NP," *Theoret. Comput. Sci.*, vol. 370, no. 1-3, 2007, pp. 293–298. DOI: 10.1016/j.tcs.2006.10.009.

[161] A. Lubotzky, R. Phillips, and P. Sarnak, "Ramanujan graphs," *Combinatorica*, vol. 8, no. 3, 1988, pp. 261–277. DOI: 10.1007/BF02126799.

[162] A. Lubotzky, "Expander graphs in pure and applied mathematics," *Bull. Amer. Math. Soc. (N.S.)*, vol. 49, no. 1, 2012, pp. 113–162. DOI: 10.1090/S0273-0979-2011-01359-3.

[163] M. Luby and B. Veličković, "On deterministic approximation of DNF," *Algorithmica*, vol. 16, no. 4/5, 1996, pp. 415–433. DOI: 10.1007/BF01940873.

[164] M. Luby, B. Veličković, and A. Wigderson, "Deterministic approximate counting of depth-2 circuits," in *Proc. 2nd Israel Symposium on Theory and Computing Systems (ISTCS)*, pp. 18–24, 1993. DOI: 10.1109/ISTCS.1993.253488.

[165] M. Luby and A. Wigderson, "Pairwise independence and derandomization," *Foundations and Trends in Theoretical Computer Science*, vol. 1, no. 4, 2006, pp. 237–301. DOI: 10.1561/0400000009.

[166] X. Lyu, "Improved Pseudorandom Generators for AC^0 Circuits," in *Proc. 37th Computational Complexity Conference (CCC)*, 34:1–34:25, 2022. DOI: 10.4230/LIPIcs.CCC.2022.34.

[167] A. W. Marcus, D. A. Spielman, and N. Srivastava, "Interlacing families I: Bipartite Ramanujan graphs of all degrees," *Ann. of Math. (2)*, vol. 182, no. 1, 2015, pp. 307–325. DOI: 10.4007/annals.2015.182.1.7.

[168] G. A. Margulis, "Explicit group-theoretic constructions of combinatorial schemes and their applications in the construction of expanders and concentrators," *Problemy Peredachi Informatsii*, vol. 24, no. 1, 1988, pp. 51–60. URL: http://mi.mathnet.ru/eng/ppi/v24/i1/p51.

[169] M. Matsumoto and T. Nishimura, "Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator," *ACM Trans. Model. Comput. Simul.*, vol. 8, no. 1, Jan. 1998, pp. 3–30. DOI: 10.1145/272991.272995.

[170] N. Mazor and J. Zhang, "Simple constructions from (almost) regular one-way functions," in *Proc. 19th Theory of Cryptography Conference (TCC)*, pp. 457–485, 2021. DOI: 10.1007/978-3-030-90453-1_16.

[171] R. Meka, O. Reingold, and A. Tal, "Pseudorandom generators for width-3 branching programs," in *Proc. 51st Annual ACM Symposium on Theory of Computing (STOC)*, pp. 626–637, 2019. DOI: 10.1145/3313276.3316319.

[172] R. Meka and D. Zuckerman, "Small-bias spaces for group products," in *Proc. 13th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, pp. 658–672, 2009. DOI: 10.1007/978-3-642-03685-9_49.

[173] R. Meka and D. Zuckerman, "Pseudorandom generators for polynomial threshold functions," *SIAM J. Comput.*, vol. 42, no. 3, 2013, pp. 1275–1301. DOI: 10.1137/100811623.

[174] A. Milchev, K. Binder, and D. Heermann, "Fluctuations and lack of self-averaging in the kinetics of domain growth," *Zeitschrift für Physik B Condensed Matter*, vol. 63, no. 4, 1986, pp. 521–535. DOI: 10.1007/BF01726202.

[175] P. B. Miltersen, "Derandomizing complexity classes," in *Handbook of randomized computing, Vol. I, II*, ser. Comb. Optim. Vol. 9, Kluwer Acad. Publ., Dordrecht, 2001, pp. 843–941. DOI: 10.1007/978-1-4615-0013-1_19.

[176] S. Mohanty, R. O'Donnell, and P. Paredes, "Explicit near-Ramanujan graphs of every degree," in *Proc. 52nd Annual ACM Symposium on Theory of Computing (STOC)*, pp. 510–523, ACM, New York, 2020. DOI: 10.1145/3357713.3384231.

[177] M. Morgenstern, "Existence and explicit constructions of $q + 1$ regular Ramanujan graphs for every prime power q," *J. Combin. Theory Ser. B*, vol. 62, no. 1, 1994, pp. 44–62. DOI: 10.1006/jctb.1994.1054.

[178] J. Naor and M. Naor, "Small-bias probability spaces: Efficient constructions and applications," *SIAM J. Comput.*, vol. 22, no. 4, 1993, pp. 838–856. DOI: 10.1137/0222053.

[179] A. Nilli, "On the second eigenvalue of a graph," *Discrete Math.*, vol. 91, no. 2, 1991, pp. 207–210. DOI: 10.1016/0012-365X(91)90112-F.

[180] N. Nisan, "Pseudorandom bits for constant depth circuits," *Combinatorica*, vol. 11, no. 1, 1991, pp. 63–70. DOI: 10.1007/BF01375474.

[181] N. Nisan, "Pseudorandom generators for space-bounded computation," *Combinatorica*, vol. 12, no. 4, 1992, pp. 449–461. DOI: 10.1007/BF01305237.

[182] N. Nisan and A. Ta-Shma, "Extracting randomness: A survey and new constructions," *J. Comput. System Sci.*, vol. 58, no. 1, part 2, 1999, pp. 148–173. DOI: 10.1006/jcss.1997.1546.

[183] N. Nisan and A. Wigderson, "Hardness vs randomness," *J. Comput. Syst. Sci.*, vol. 49, no. 2, 1994, pp. 149–167. DOI: 10.1016/S0022-0000(05)80043-1.

[184] N. Nisan and D. Zuckerman, "Randomness is linear in space," *J. Comput. System Sci.*, vol. 52, no. 1, 1996, pp. 43–52. DOI: 10.1006/jcss.1996.0004.

[185] R. O'Donnell, *Analysis of Boolean Functions*. Cambridge University Press, 2014. DOI: 10.1017/CBO9781139814782.

[186] C. H. Papadimitriou and M. Sipser, "Communication complexity," *J. Comput. System Sci.*, vol. 28, no. 2, 1984, pp. 260–269. DOI: 10.1016/0022-0000(84)90069-2.

[187] G. Parisi and F. Rapuano, "Effects of the random number generator on computer simulations," *Physics Letters B*, vol. 157, no. 4, 1985, pp. 301–302. DOI: 10.1016/0370-2693(85)90670-7.

[188] R. Peralta, "On the randomness complexity of algorithms," *University of Wisconsin, Milwaukee CS Research Report TR 90-1*, 1990.

[189] N. Perlroth, "Government announces steps to restore confidence on encryption standards," *The New York Times*, 2013. URL: https://bits.blogs.nytimes.com/2013/09/10/government-announces-steps-to-restore-confidence-on-encryption-standards/ (accessed on 07/14/2021).

[190] N. Pippenger and M. J. Fischer, "Relations among complexity measures," *Journal of the ACM*, vol. 26, no. 2, 1979, pp. 361–381. DOI: 10.1145/322123.322138.

[191] E. Pyne, R. Raz, and W. Zhan, *Certified hardness vs. randomness for log-space*, ECCC preprint TR23-040, 2023. URL: https://eccc.weizmann.ac.il/report/2023/040/.

[192] E. Pyne and S. Vadhan, "Limitations of the Impagliazzo-Nisan-Wigderson pseudorandom generator against permutation branching programs," in *Proc. 27th International Computing and Combinatorics Conference (COCOON)*, pp. 3–12, 2021. DOI: 10.1007/978-3-030-89543-3_1.

[193] E. Pyne and S. Vadhan, "Pseudodistributions that beat all pseudorandom generators (extended abstract)," in *Proc. 36th Computational Complexity Conference (CCC)*, 33:1–33:15, 2021. DOI: 10.4230/LIPIcs.CCC.2021.33, Full version: ECCC preprint TR21-019.

[194] E. Pyne and S. Vadhan, "Deterministic approximation of random walks via queries in graphs of unbounded size," in *Proc. 5th Symposium on Simplicity in Algorithms (SOSA)*, pp. 57–67, 2022. DOI: 10.1137/1.9781611977066.5.

[195] Y. Rabani and A. Shpilka, "Explicit construction of a small ε-net for linear threshold functions," *SIAM J. on Computing*, vol. 39, no. 8, 2010, pp. 3501–3520. DOI: 10.1137/090764190.

[196] A. Rao and A. Yehudayoff, *Communication Complexity and Applications*. Cambridge University Press, 2020. DOI: 10.1017/9781108671644.

[197] R. Raz, O. Reingold, and S. Vadhan, "Extracting all the randomness and reducing the error in Trevisan's extractors," *J. Comput. System Sci.*, vol. 65, no. 1, 2002, pp. 97–128. DOI: 10.1006/jcss.2002.1824.

[198] A. A. Razborov, "Lower bounds on the size of bounded depth circuits over a complete basis with logical addition," *Math. Notes*, vol. 41, no. 4, 1987, pp. 333–338. DOI: 10.1007/BF01137685.

[199] A. A. Razborov and S. Rudich, "Natural proofs," *J. Comput. Syst. Sci.*, vol. 55, no. 1, 1997, pp. 24–35.

[200] A. Razborov, "A simple proof of Bazzi's theorem," *ACM Transactions on Computation Theory*, vol. 1, no. 1, 2009. DOI: 10.1145/1490270.1490273.

[201] A. A. Razborov, "Lower bounds for deterministic and nondeterministic branching programs," in *Proc. 8th International Conference on Fundamentals of Computation Theory (FCT)*, pp. 47–60, 1991. DOI: 10.1007/3-540-54458-5_49.

[202] O. Reingold, T. Steinke, and S. Vadhan, "Pseudorandomness for regular branching programs via Fourier analysis," in *Proc. 17th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, pp. 655–670, 2013. DOI: 10.1007/978-3-642-40328-6_45.

[203] O. Reingold, L. Trevisan, and S. Vadhan, "Pseudorandom walks on regular digraphs and the **RL** vs. **L** problem," in *Proc. 38th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 457–466, 2006. DOI: 10.1145/1132516.1132583.

[204] V. Rödl, "On a packing and covering problem," *European Journal of Combinatorics*, vol. 6, no. 1, 1985, pp. 69–78. DOI: 10.1016/S0195-6698(85)80023-8.

[205] B. Rossman, "Criticality of Regular Formulas," in *Proc. 34th Computational Complexity Conference (CCC)*, 1:1–1:28, 2019. DOI: 10.4230/LIPIcs.CCC.2019.1.

[206] M. Saks and S. Zhou, "$\mathrm{BP_HSPACE}(S) \subseteq \mathrm{DSPACE}(S^{3/2})$," *J. Comput. System Sci.*, vol. 58, no. 2, 1999, pp. 376–403. DOI: 10.1006/jcss.1998.1616.

[207] J. Schönheim, "On coverings," *Pacific J. Math.*, vol. 14, 1964, pp. 1405–1411. URL: http://projecteuclid.org/euclid.pjm/1103033815.

[208] R. A. Servedio and L.-Y. Tan, "Luby-Veličković-Wigderson revisited: Improved correlation bounds and pseudorandom generators for depth-two circuits," in *Proc. 22nd International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, 56:1–56:20, 2018. DOI: 10.4230/LIPIcs.APPROX-RANDOM.2018.56.

[209] R. A. Servedio and L.-Y. Tan, "Improved Pseudorandom Generators from Pseudorandom Multi-Switching Lemmas," in *Proc. 28th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, 45:1–45:23, 2019. DOI: 10.4230/LIPIcs.APPROX-RANDOM.2019.45.

[210] R. A. Servedio and L.-Y. Tan, "Pseudorandomness for read-k DNF formulas," in *Proc. 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 621–638, 2019. DOI: 10.1137/1.9781611975482.39.

[211] R. Shaltiel, "Recent developments in extractors," *Bulletin of the European Association for Theoretical Computer Science*, vol. 77, Jun. 2002, pp. 67–95.

[212] R. Shaltiel, "An introduction to randomness extractors," in *Proc. 38th International Colloquium on Automata, Languages and Programming (ICALP)*, pp. 21–41, 2011. DOI: 10.1007/978-3-642-22012-8_2.

[213] R. Shaltiel and C. Umans, "Simple extractors for all min-entropies and a new pseudorandom generator.," *J. ACM*, vol. 52, no. 2, 2005, pp. 172–216. DOI: 10.1145/1059513.1059516.

[214] A. Shamir, "On the generation of cryptographically strong pseudorandom sequences," *ACM Trans. Comput. Syst.*, vol. 1, no. 1, 1983, pp. 38–44. DOI: 10.1145/357353.357357.

[215] A. Ta-Shma, "Randomized algorithms and de-randomization," 2015. URL: http://www.cs.tau.ac.il/~amnon/Classes/2015-PRG/class.htm.

[216] A. Ta-Shma, "Expanders, pseudorandomness and derandomization," 2016. URL: http://www.cs.tau.ac.il/~amnon/Classes/2016-PRG/class.htm.

[217] A. Ta-Shma, "Explicit, almost optimal, epsilon-balanced codes," in *Proc. 49th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 238–251, 2017. DOI: 10.1145/3055399.3055408.

[218] A. Ta-Shma, "Space-bounded computation," 2018. URL: http://www.cs.tau.ac.il/~amnon/Classes/2018-Space/class.htm.

[219] A. Ta-Shma, "A first course in derandomization," 2019. URL: http://www.cs.tau.ac.il/~amnon/Classes/2019-Derandomization/class.htm.

[220] J. Šíma and S. Žák, "A polynomial-time construction of a hitting set for read-once branching programs of width 3," *Fund. Inform.*, vol. 184, no. 4, 2021, pp. 307–354. DOI: 10.3233/fi-2021-2101.

[221] M. Skorski, "Tight Chernoff-Like Bounds Under Limited Independence," in *Proc. 26th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, 15:1–15:14, 2022. DOI: 10.4230/LIPIcs.APPROX/RANDOM.2022.15.

[222] R. Smolensky, "On representations by low-degree polynomials," in *Proc. 34th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 130–138, 1993. DOI: 10.1109/SFCS.1993.366874.

[223] R. Smolensky, "Algebraic methods in the theory of lower bounds for Boolean circuit complexity," in *Proc. 19th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 77–82, 1987. DOI: 10.1145/28395.28404.

[224] T. Steinke, *Pseudorandomness for permutation branching programs without the group theory*, ECCC preprint TR12-083, 2012. URL: https://eccc.weizmann.ac.il/report/2012/083/.

[225] T. Steinke, S. Vadhan, and A. Wan, "Pseudorandomness and Fourier-growth bounds for width-3 branching programs," *Theory Comput.*, vol. 13, 2017, Paper No. 12. DOI: 10.4086/toc.2017.v013a012.

[226] B. A. Subbotovskaya, "Realizations of linear function by formulas using $+, \cdot, -$," *Doklady Akademii Nauk SSSR*, vol. 136:3, 1961, pp. 553–555. URL: http://mi.mathnet.ru/eng/dan/v136/i3/p553.

[227] M. Sudan, L. Trevisan, and S. Vadhan, "Pseudorandom generators without the xor lemma," *J. Comput. Syst. Sci.*, vol. 62, no. 2, 2001, pp. 236–266. DOI: 10.1006/jcss.2000.1730.

[228] A. Tal, "Tight Bounds on the Fourier Spectrum of AC0," in *Proc. 32nd Computational Complexity Conference (CCC)*, 15:1–15:31, 2017. DOI: 10.4230/LIPIcs.CCC.2017.15.

[229] A. Tal, "Pseudorandomness," 2021. URL: https://www.avishaytal.org/pseudorandomness.

[230] J. Tarui, "Probabilistic polynomials, AC^0 functions and the polynomial-time hierarchy," *Theoret. Comput. Sci.*, vol. 113, no. 1, 1993, pp. 167–183. DOI: 10.1016/0304-3975(93)90214-E.

[231] S. Toda and M. Ogiwara, "Counting classes are at least as hard as the polynomial-time hierarchy," *SIAM J. Comput.*, vol. 21, no. 2, 1992, pp. 316–328. DOI: 10.1137/0221023.

[232] L. Trevisan, "Extractors and pseudorandom generators," *J. ACM*, vol. 48, no. 4, 2001, pp. 860–879. DOI: 10.1145/502090.502099.

[233] L. Trevisan, "Pseudorandomness and combinatorial constructions," 2005. URL: https://web.archive.org/web/20150115081847/http://www.cs.berkeley.edu/~luca/pacc/.

[234] L. Trevisan and S. Vadhan, "Pseudorandomness and average-case complexity via uniform reductions," *Comput. Complexity*, vol. 16, no. 4, 2007, pp. 331–364. DOI: 10.1007/s00037-007-0233-x.

[235] L. Trevisan and T. Xue, "A derandomized switching lemma and an improved derandomization of AC0," in *Proc. 28th Annual IEEE Conference on Computational Complexity (CCC)*, pp. 242–247, 2013. DOI: 10.1109/CCC.2013.32.

[236] C. Umans, "Pseudo-random generators for all hardnesses," *J. of Computer and System Sciences*, vol. 67, no. 2, 2003, pp. 419–440. DOI: 10.1016/S0022-0000(03)00046-1.

[237] S. Vadhan and C. J. Zheng, "Characterizing pseudoentropy and simplifying pseudorandom generator constructions," in *Proc. 44th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 817–836, 2012. DOI: 10.1145/2213977.2214051.

[238] S. P. Vadhan, "Pseudorandomness," *Foundations and Trends in Theoretical Computer Science*, vol. 7, no. 1-3, 2012, pp. 1–336. DOI: 10.1561/0400000010.

[239] L. G. Valiant and V. V. Vazirani, "NP is as easy as detecting unique solutions," *Theoret. Comput. Sci.*, vol. 47, no. 1, 1986, pp. 85–93. DOI: 10.1016/0304-3975(86)90135-0.

[240] S. Vigna, "Further scramblings of Marsaglia's `xorshift` generators," *J. Comput. Appl. Math.*, vol. 315, 2017, pp. 175–181. DOI: 10.1016/j.cam.2016.11.006.

[241] E. Viola, "Pseudorandom bits for constant-depth circuits with few arbitrary symmetric gates," *SIAM J. Comput.*, vol. 36, no. 5, 2007, pp. 1387–1403. DOI: 10.1137/050640941.

[242] E. Viola, "The sum of D small-bias generators fools polynomials of degree D," *Comput. Complexity*, vol. 18, no. 2, 2009, pp. 209–217. DOI: 10.1007/s00037-009-0273-5.

[243] E. Viola, "Randomness buys depth for approximate counting," *Comput. Complexity*, vol. 23, no. 3, 2014, pp. 479–508. DOI: 10.1007/s00037-013-0076-6.

[244] E. Viola, "Special topics in complexity theory," 2017. URL: https://www.ccs.neu.edu/home/viola/classes/spepf17.html.

[245] E. Viola, "Fourier conjectures, correlation bounds, and majority," in *Proc. 48th International Colloquium on Automata, Languages and Programming (ICALP)*, 111:1–111:15, 2021. DOI: 10.4230/LIPIcs.ICALP.2021.111.

[246] E. Viola, *Correlation bounds against polynomials*, ECCC preprint TR22-142, 2022. URL: https://eccc.weizmann.ac.il/report/2022/142/.

[247] I. Wegener, *The complexity of Boolean functions*, ser. Wiley-Teubner Series in Computer Science. John Wiley & Sons, Inc., 1987, pp. xii+457. URL: https://dl.acm.org/doi/10.5555/35517.

[248] A. Wigderson, *Randomness and pseudorandomness*, IAS Institute Letter, 2009. URL: https://www.ias.edu/ideas/2009/wigderson-randomness-pseudorandomness.

[249] A. C. Yao, "Theory and applications of trapdoor functions," in *Proc. 23rd Annual ACM Symposium on Theory of Computing (STOC)*, pp. 80–91, 1982. DOI: 10.1109/SFCS.1982.45.

[250] Y. Yu, D. Gu, X. Li, and J. Weng, "The randomized iterate, revisited—almost linear seed length PRGs from a broader class of one-way functions," in *Proc. 12th Theory of Cryptography Conference (TCC)*, pp. 7–35, 2015. DOI: 10.1007/978-3-662-46494-6_2.

[251] Y. Yu, X. Li, and J. Weng, "Pseudorandom generators from regular one-way functions: New constructions with improved parameters," *Theoret. Comput. Sci.*, vol. 569, 2015, pp. 58–69. DOI: 10.1016/j.tcs.2014.12.013.

[252] D. Zuckerman, "General weak random sources," in *Proc. 31st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pp. 534–543, 1990. DOI: 10.1109/FSCS.1990.89574.

[253] D. Zuckerman, "Pseudorandomness and combinatorial constructions," 2001. URL: https://www.cs.utexas.edu/~diz/395T/01/.