THEORY OF COMPUTING, Volume 18 (21), 2022, pp. 1–32
www.theoryofcomputing.org

SPECIAL ISSUE: CCC 2020

Hitting Sets Give Two-Sided Derandomization of Small Space

Kuan Cheng* William M. Hoza†

Received July 19, 2020; Revised June 21, 2022; Published September 20, 2022

Abstract. A *hitting set* is a "one-sided" variant of a pseudorandom generator (PRG), naturally suited to derandomizing algorithms that have one-sided error. We study the problem of using a given hitting set to derandomize algorithms that have *two-sided* error, focusing on space-bounded algorithms. For our first result, we show that if there is a log-space hitting set for polynomial-width read-once branching programs (ROBPs), then not only does $\mathbf{L} = \mathbf{RL}$ hold, but $\mathbf{L} = \mathbf{BPL}$ as well. This answers a question raised by Hoza and Zuckerman (SICOMP 2020).

Next, we consider constant-width ROBPs. We show that if there are log-space hitting sets for constant-width ROBPs, then given black-box access to a constant-width ROBP f, it is possible to deterministically estimate $\mathbb{E}[f]$ to within $\pm\epsilon$ in space

A conference version of this paper appeared in the Proceedings of the 35th Computational Complexity Conference, 2020 [11].

*This research was conducted while the author was a postdoctoral researcher at the University of Texas at Austin, supported by a Simons Investigator Award (#409864, David Zuckerman).

†Most of this work was done while the author was a graduate student at the University of Texas at Austin, supported by the NSF GRFP under Grant DGE-1610403 and by a Harrington Fellowship from UT Austin. Part of this work was done while the author was visiting the Simons Institute for the Theory of Computing.

ACM Classification: F.1.2, F.1.3

AMS Classification: 68Q10, 68Q15, 68Q87

Key words and phrases: derandomization, pseudorandomness, hitting set, branching programs, space complexity

 DOI: 10.4086/toc.2022.v018.a021

$O(\log(n/\epsilon))$. Unconditionally, we give a deterministic algorithm for this problem with space complexity $O(\log^2 n + \log(1/\epsilon))$, slightly improving over previous work.

Finally, we investigate the limits of this line of work. Perhaps the strongest reduction along these lines one could hope for would say that for every explicit hitting set generator, there is an explicit PRG with similar parameters. In the setting of constant-width ROBPs over a large alphabet, we prove that establishing such a strong reduction is at least as difficult as constructing a good PRG outright. Quantitatively, we prove that if the strong reduction holds, then for every constant $\alpha > 0$, there is an explicit PRG for constant-width ROBPs with seed length $O(\log^{1+\alpha} n)$. Along the way, unconditionally, we construct an improved hitting set for ROBPs over a large alphabet.

1 Introduction

Suppose some decision problem can be solved by an efficient randomized algorithm. That's good, but an efficient deterministic algorithm would be even better. We would therefore like to deterministically analyze the acceptance probability of the randomized algorithm on a given input. An ambitious approach to derandomization is to try to design a suitable *pseudorandom generator* (PRG).

Definition 1.1. Let $\mathcal{F}$ be a class of functions $f\colon \{0,1\}^n \to \{0,1\}$. An *$\epsilon$-PRG* for $\mathcal{F}$ is a function $G\colon \{0,1\}^s \to \{0,1\}^n$ such that for every $f \in \mathcal{F}$, $\left|\mathbb{E}[f] - \mathbb{E}_{X\in\{0,1\}^s}[f(G(X))]\right| \leq \epsilon$.

Let n be the number of random bits used by the randomized algorithm, and ensure that $\mathcal{F}$ can compute the action of the randomized algorithm on its random bits. By iterating over all "seeds" $x \in \{0,1\}^s$ and plugging $G(x)$ into the randomized algorithm, we can get an estimate of its acceptance probability with additive error at most ϵ.

Unfortunately, designing efficient PRGs has proved to be extremely difficult. Constructing a *hitting set* is sometimes less difficult.

Definition 1.2. Let $\mathcal{F}$ be a class of functions $f\colon \{0,1\}^n \to \{0,1\}$. An *$\epsilon$-hitting set* for $\mathcal{F}$ is a set $H \subseteq \{0,1\}^n$ such that for every $f \in \mathcal{F}$ with $\mathbb{E}[f] \geq \epsilon$, there is some $x \in H$ such that $f(x) = 1$.

The image of any ϵ-PRG is clearly an ϵ'-hitting set for every $\epsilon' > \epsilon$. By iterating over all strings in an ϵ-hitting set, we can at least distinguish acceptance probability 0 from acceptance probability $\geq \epsilon$. This is already sufficient for derandomizing some algorithms (namely, those with "one-sided error"). In this paper, we investigate the possibility of using a hitting set in a nontrivial way to obtain an estimate of the acceptance probability with a small additive error, just like what a PRG would have provided.

This possibility was previously studied in the context of derandomizing time-bounded algorithms. Several proofs have been discovered showing that if there is a polynomial-time hitting set for size-n circuits, then $\mathbf{P} = \mathbf{BPP}$ [3, 9, 4, 13]. In Section 5 we provide yet another proof of this theorem; our short proof is arguably simpler than all previous proofs. However, the focus of our paper is derandomizing space-bounded algorithms.

1.1 Derandomizing log-space algorithms

The behavior of a small-space algorithm as a function of its random bits can be modeled by a *read-once*[1] *branching program* (ROBP). A width-w length-n ROBP is a directed graph consisting of $n+1$ layers with w vertices per layer. There is a designated "start vertex" v_{start} in the first layer. Every vertex not in the last layer has two outgoing edges labeled 0 and 1 leading to the next layer. An n-bit input string naturally identifies a path through the graph by reading from left to right. The program accepts or rejects this string depending on whether the path ends at the designated "accept vertex" v_{acc} in the last layer.

Recall that **BPL** and **RL** are the classes of languages that can be decided by randomized log-space algorithms that always halt with two-sided and one-sided error respectively. A log-space hitting set[2] for polynomial-width ROBPs would immediately imply $\mathbf{L} = \mathbf{RL}$. For our first result, we show that such a hitting set would also imply $\mathbf{L} = \mathbf{BPL}$.

Theorem 1.3. *Assume that for every $n \in \mathbb{N}$, there is a $\frac{1}{2}$-hitting set for width-n, length-n ROBPs that can be computed in space $O(\log n)$. Then* $\mathbf{L} = \mathbf{BPL}$.

1.2 Motivation: Recent work on hitting sets

Theorem 1.3 is especially interesting in light of recent constructions of improved hitting sets for ROBPs [8, 19, 10]. The best PRG known for polynomial-width ROBPs is still Nisan's PRG [25], which has seed length

$$O(\log^2 n + \log n \log(1/\epsilon)) .$$

Until recently, Nisan's PRG also provided the best hitting set for polynomial-width ROBPs. Using sophisticated and novel techniques, Braverman, Cohen, and Garg obtained a hitting set with space complexity

$$\widetilde{O}(\log^2 n + \log(1/\epsilon)) ,$$

which is an improvement when ϵ is very small [8].

Actually, Braverman, Cohen, and Garg constructed something better than a hitting set, called a *weighted pseudorandom generator* (WPRG) or alternatively a *pseudorandom pseudodistribution generator*.

Definition 1.4 ([8])**.** Let $\mathcal{F}$ be a class of functions $f\colon \{0,1\}^n \to \{0,1\}$. An *$\epsilon$-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 $f \in \mathcal{F}$,

$$\left| \mathbb{E}[f] - \mathop{\mathbb{E}}_{X \in \{0,1\}^s}[f(G(X)) \cdot \rho(X)] \right| \le \epsilon .$$

[1]Because space-bounded algorithms only have read-once access to their random bits, it does not seem possible to adapt the existing derandomizations of **BPP** using a hitting set to the **BPL** case.

[2]When we say "a log-space hitting set," we mean a family of hitting sets $H_n \subseteq \{0,1\}^n$ such that given input $n \in \mathbb{N}$, the set H_n can be enumerated in space $O(\log n)$. For such a family, $|H_n| \le \text{poly}(n)$.

A WPRG can be used to estimate $\mathbb{E}[f]$ to within $\pm\epsilon$, provided that G and ρ can both be computed efficiently. The concept of a WPRG generalizes the concept of a PRG, which is the special case $\rho(x) \equiv 1$. In turn, if (G, ρ) is an ϵ-WPRG, then the image of G is an ϵ'-hitting set for every $\epsilon' > \epsilon$. So a WPRG is *intermediate* between a hitting set and a genuine PRG.

After Braverman, Cohen, and Garg's work [8], Hoza and Zuckerman gave a simpler construction of an ϵ-hitting set for polynomial-width ROBPs, with the slightly improved space complexity $O(\log^2 n + \log(1/\epsilon))$ [19]. Their construction is weaker in that it does not provide a WPRG. Theorem 1.3 bridges the gap between the two concepts somewhat: by Theorem 1.3, any generic hitting set can be used for two-sided derandomization, which was the main strength of a WPRG over a hitting set in the first place.

(Independently of our work, Chattopadhyay and Liao gave an improved WPRG construction with seed length $\widetilde{O}(\log^2 n) + O(\log(1/\epsilon))$ [10]. Further WPRG constructions were devised after the preliminary version of our work [11] appeared, leading to a seed length of $O(\log^2 n + \log(1/\epsilon))$ [12, 28, 17], which matches the parameters of Hoza and Zuckerman's hitting set for polynomial-width ROBPs [19].)

1.3 The constant-width setting

However, there is a weakness of Theorem 1.3. A PRG or a WPRG would provide a *black-box* derandomization, whereas the algorithm of Theorem 1.3 is not black-box. This weakness is especially acute when we consider the constant-width case. Given a constant-width ROBP f directly as input, it is trivial to compute $\mathbb{E}[f]$ with high accuracy, so the algorithm of Theorem 1.3 is meaningless. Nevertheless, constant-width ROBPs can compute many interesting functions, and it is a major open challenge to design improved PRGs, WPRGs, or hitting sets for constant-width ROBPs. (For width 2, optimal PRGs are known [7]. For width 3, the current best PRG has seed length $\widetilde{O}(\log n \log(1/\epsilon))$ [24]. The best hitting sets for width 3 are superior, with space complexity $\widetilde{O}(\log(n/\epsilon))$ for small ϵ [16] or $O(\log n)$ for large enough ϵ [29]. For width 4, the state of the art is simply the best results for polynomial-width ROBPs mentioned above [25, 19, 17].)

To address this weakness of Theorem 1.3, we abstract the "black-box" feature of PRGs and WPRGs in the following definition.

Definition 1.5. Let $\mathcal{F}$ be a class of functions $f\colon \{0,1\}^n \to \{0,1\}$. A *deterministic ϵ-sampler* for $\mathcal{F}$ is a deterministic oracle algorithm A that outputs a real number such that for every $f \in \mathcal{F}$,

$$|A^f - \mathbb{E}[f]| \leq \epsilon\,.$$

The concept of a deterministic sampler generalizes that of a WPRG, because given a WPRG (G, ρ) with seed length s, one can set $A^f = 2^{-s}\sum_x f(x)\rho(x)$. In the other direction, deterministic samplers imply hitting sets.

Proposition 1.6. *Identify 0 with the constant 0 function on $\{0,1\}^n$, and assume $0 \in \mathcal{F}$. Let A be a deterministic ϵ-sampler for $\mathcal{F}$, and let $H \subseteq \{0,1\}^n$ be the set of points where A^0 queries its oracle. Then for every $\epsilon' > 2\epsilon$, H is an ϵ'-hitting set for $\mathcal{F}$.*

Proof. Let $f \in \mathcal{F}$ satisfy $\mathbb{E}[f] > 2\epsilon$. Since $|A^0 - 0| \le \epsilon$ and $|A^f - \mathbb{E}[f]| \le \epsilon$, $A^0 \neq A^f$. Therefore, A^f must query f at some point $x \in f^{-1}(1)$. The *first* such query must be at a point $x \in H$. □

The deterministic sampler model captures some prior derandomization algorithms. In particular, all known derandomizations of **BPP** using a hitting set [3, 9, 4, 13], including our new derandomization in Section 5, can be interpreted as constructing deterministic samplers for circuits. For our second result, we prove the analogous reduction for constant-width ROBPs, i. e., we show that one can generically "upgrade" log-space hitting sets for such programs into log-space deterministic samplers. (See Figure 1.)

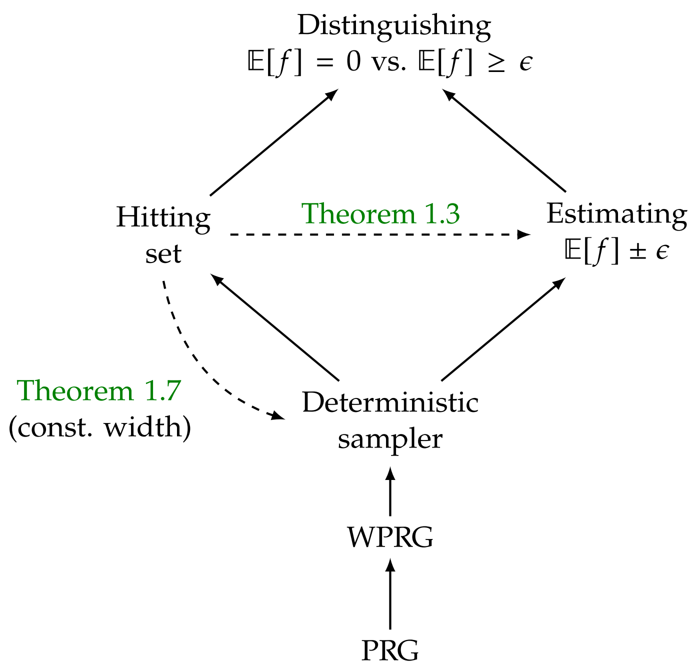

Figure 1: The relationships between different derandomization goals. The solid arrows are implications that are immediate from the definitions and hold for essentially any class $\mathcal{F}$ (possibly with some loss in ϵ). The dashed arrows are theorems in this paper, holding for ROBPs specifically.

Theorem 1.7. *Assume that for every constant w, for all $n \in \mathbb{N}$, there is a $\frac{1}{2}$-hitting set for width-w length-n ROBPs that can be computed in space $O(\log n)$. Then for every constant w, for all $n \in \mathbb{N}$ and all $\epsilon > 0$, there is a deterministic ϵ-sampler for width-w length-n ROBPs that runs in space $O(\log(n/\epsilon))$.*

More generally, given a $\frac{1}{2}$-hitting set for width-w' length-n' ROBPs for suitable values $w' = O(w^2)$ and $n' = O(n^2/\epsilon)$, we show how to construct a deterministic ϵ-sampler for width-w length-n ROBPs. If the hitting set has space complexity s, then our deterministic sampler has space complexity $O(s + w \log(n/\epsilon))$. See Theorem 3.1 and Corollary 3.7.

The proof of Theorem 1.7 uses different techniques than that of Theorem 1.3. Because of the factor of w in our deterministic sampler's space complexity, our sampler becomes meaningless when w is large. Thus, Theorem 1.3 and Theorem 1.7 are incomparable.

We also obtain a new *unconditional* deterministic sampler. When ϵ is moderate, the best deterministic sampler for constant-width ROBPs is simply from Nisan's PRG [25], which gives a sampler with space complexity $O(\log^2 n + \log n \log(1/\epsilon))$. When ϵ is small, using prior work, the best deterministic sampler for constant-width ROBPs was from Braverman, Cohen, and Garg's WPRG [8] (space complexity $\widetilde{O}(\log^2 n + \log(1/\epsilon)))$. The concurrent work by Chattopadhyay and Liao [10] gives a slightly better WPRG, and hence a slightly better deterministic sampler (space complexity $\widetilde{O}(\log^2 n) + O(\log(1/\epsilon)))$. By applying the reduction underlying Theorem 1.7 to the hitting set of Hoza and Zuckerman [19], we achieve a slightly better bound.

Theorem 1.8 (Unconditional sampler)**.** *For every constant w, for all $n \in \mathbb{N}$ and all $\epsilon > 0$, there is a deterministic ϵ-sampler for width-w length-n ROBPs running in space $O(\log^2 n + \log(1/\epsilon))$.*

In light of Theorem 1.8, when it comes to deterministic samplers, there is now a slight gap between the state of the art for polynomial-width ROBPs vs. the state of the art for width-w ROBPs with w a large constant. In other words, Theorem 1.8 is a case where we can take advantage of narrowness. There is no such gap when it comes to PRGs, WPRGs, or hitting sets. (As mentioned previously, work subsequent to this paper constructed WPRGs for polynomial-width ROBPs with seed length $O(\log^2 n + \log(1/\epsilon))$ [12, 28, 17], generalizing Theorem 1.8 and closing the aforementioned gap between polynomial-width ROBPs and constant-width ROBPs.)

1.4 Implications of a stronger reduction

Theorem 1.7 raises the question of whether we can go even further and upgrade any hitting set into a genuine PRG. In the time-bounded setting, this is indeed possible via the "hardness vs. randomness" paradigm.[3] Also, in the context of low-degree polynomials, Bogdanov showed how to convert any hitting set with a certain density property into a PRG [6]. Can a similar reduction be proven for small-space models?

We focus on the setting of constant-width ROBPs over a large alphabet. (An *ROBP over the alphabet* Σ computes a function $f : \Sigma^n \to \{0,1\}$; each vertex not in the last layer has $|\Sigma|$ outgoing edges labeled with the symbols in Σ.) We prove that *if* for every explicit hitting set in this setting, there is an explicit PRG with similar parameters, then there is in fact an explicit PRG for constant-width binary ROBPs with seed length $O(\log^{1+\alpha} n)$, where $\alpha > 0$ is an arbitrarily small constant. See Theorem 4.3 for the precise statement.

Our result is similar to a theorem by Hoza and Umans [18]. Like us, Hoza and Umans showed that if PRGs are equivalent to a seemingly weaker notion, then the equivalence itself can be used to construct a good PRG. Hoza and Umans focused on the distinction between PRGs and non-black-box derandomization, whereas we focus on the distinction between PRGs and hitting sets.

[3] If for every n there is a hitting set for size-n circuits computable in poly(n) time, then there is a language in **E** that requires circuits of size $2^{\Omega(n)}$. A major achievement in complexity theory was to show that assuming such a language exists, for every n, there is a polynomial-time logarithmic-seed PRG for size-n circuits [21].

1.4.1 Interpretation

Like any conditional theorem, Theorem 4.3 has both a positive and a negative interpretation.[4] According to the negative interpretation, Theorem 4.3 shows that it would be difficult to establish a general reduction from PRGs to hitting sets. After all, it's as difficult as constructing a good PRG for constant-width ROBPs, which is a challenge that researchers have been struggling with for decades. In this sense, Theorem 4.3 provides an "excuse" for the fact that Theorem 1.3 and Theorem 1.7 do not provide genuine PRGs.

We feel that the negative interpretation is more realistic, but there is also a sensible positive interpretation. According to the positive interpretation, our work provides a new approach to constructing improved PRGs or hitting sets for constant-width ROBPs. One "merely" needs to bridge the gap between deterministic samplers and PRGs. This could be done in one of two ways. One could improve Theorem 1.7 so that it concludes with a PRG instead of a deterministic sampler. Alternatively, one could improve the construction of Theorem 4.3 so that rather than relying on the equivalence of hitting sets and PRGs, it merely relies on the equivalence of hitting sets and deterministic samplers. (In exchange, presumably the conclusion would merely be a deterministic sampler rather than a true PRG, but that would still be a breakthrough.)

1.5 Overview of techniques

Let us first fix some notation. Let U_n denote the uniform distribution over $\{0,1\}^n$. For two strings x, y, let $x \circ y$ denote the concatenation of x with y. Suppose an ROBP f is clear from context. If u and v are vertices, let $p_{u\to v}$ be the probability that a random walk starting at u reaches v. We use the shorthand $p_{\to v} = p_{v_{\text{start}}\to v}$ and $p_{u\to} = p_{u\to v_{\text{acc}}}$. We use V_i to denote the set of vertices in the i-th layer of the ROBP, where $i \in \{0, 1, \ldots, n\}$.

1.5.1 Techniques for Theorem 1.3

We begin by outlining the proof of Theorem 1.3 (on derandomizing **BPL**). To derandomize **BPL**, it suffices to show that given a width-n length-n ROBP f, one can estimate $\mathbb{E}[f]$ to within a small additive error in log space. We do this using a hitting set H for width-(n^c) length-(n^c) ROBPs, where c is a large enough constant.

Each $x \in H$ is a string of length n^c. We think of x as a list of many shorter strings. Specifically, for every vertex v in f, the string x provides $\text{poly}(n)$ "sample inputs" associated with v. We compute the fraction $\widehat{p}_{\to v}$ of those sample inputs that lead to v. The hope is that

$$\forall v,\ \widehat{p}_{\to v} \approx p_{\to v}\,. \tag{1.1}$$

Of course we cannot directly verify Equation (1.1), since we do not know the values $p_{\to v}$. Instead, our algorithm looks for an $x \in H$ such that the estimates $\widehat{p}_{\to v}$ are *locally consistent*, i. e., for every

[4]Throughout this discussion, we will ignore the issue of alphabet size, to simplify matters. The proof of Theorem 1.7 does generalize well to the large-alphabet case.

$i \in [n]$ and every $v \in V_i$,

$$\widehat{p}_{\to v} \approx \sum_{u \in V_{i-1}} \widehat{p}_{\to u} \cdot p_{u \to v} \,.$$

Having found such an $x \in H$, we output the corresponding value $\widehat{p}_{\to v_{\mathrm{acc}}}$.

To establish the correctness of our algorithm, we must show two assertions. First, if the estimates $\widehat{p}_{\to v}$ pass the local consistency test, then $\mathbb{E}[f] \approx \widehat{p}_{\to v_{\mathrm{acc}}}$. Second, there is always a string $x \in H$ that passes the local consistency test.

To show the first assertion, we bound $\sum_{v \in V_i} |\widehat{p}_{\to v} - p_{\to v}|$ by induction on i. Because of the structure of the ROBP, the error accumulates mildly, only growing by a factor that is approximately the size of f.

For the second assertion, we use the hitting property of H. At first glance, it might seem that the assertion is immediate. After all, a random x certainly passes the local consistency test with high probability, and the local consistency test can be computed in small space. Unfortunately, however, that computation involves reading the bits of x multiple times, whereas H is merely guaranteed to hit *read-once* branching programs.

To deal with this issue, we notice that if x were chosen at random, then with high probability, it would satisfy Equation (1.1). Furthermore, there exists a width-(n^c) length-(n^c) ROBP f' that determines whether its input x satisfies Equation (1.1). The values $p_{\to v}$ are all hard-coded into f'. There is no need to algorithmically construct f'; the mere fact that it exists implies the existence of an $x \in H$ that satisfies Equation (1.1). Satisfying Equation (1.1) readily implies that x also passes the local consistency test, completing the proof of the second assertion.

1.5.2 Techniques for Theorem 1.7

The proof of Theorem 1.7 (on deterministic samplers) uses different techniques. Let f be a constant-width ROBP. To estimate $\mathbb{E}[f]$, we use the assumed hitting set H and queries to f to infer an approximation of the graph structure of f in some sense.

In more detail, let H_* be the set of all prefixes of strings in H, and let $H_i = H_* \cap \{0,1\}^i$. We use the hitting set H in two different ways. First, we identify each prefix $x \in H_*$ with the vertex that is reached when f reads x (so prefixes of length i correspond to vertices in layer i). In this way, we can think of f as a program over the vertex set H_*, which we have direct access to.

Second, we use H to merge some of these "vertices." We define an equivalence relation on $\{0,1\}^i$ by the rule

$$x \sim y \iff \forall z \in H_{n-i}, f(x \circ z) = f(y \circ z) \,.$$

Note that equivalence can be tested by making queries to f. If two "vertices" $x, y \in H_i$ are equivalent, then we think of them as a single "merged" vertex. (This merging operation is similar to a randomized learning algorithm by Gopalan, Klivans, and Meka [15]. Note that their algorithm is not space-efficient.) The number of distinct equivalence classes – hence the number of vertices per layer after merging – is bounded by the width of f.

Given the (approximate) graph structure of f, we work our way backward through the program starting with the final layer. We inductively compute the acceptance probability $p_{v\to}$ from each vertex v using the formula $p_{v\to} = \frac{1}{2}(p_{v_0\to} + p_{v_1\to})$, where v_0 and v_1 are the two out-neighbors of v. This allows us to estimate $\mathbb{E}[f]$, i. e., the acceptance probability from the start vertex $p_{v_{\text{start}}\to}$.

For the analysis, we must contend with two types of errors. First, there might be some vertices in f that are never visited when f reads strings in H. Such vertices are effectively lost when we think of f as a program over the vertex set H_*. Intuitively, these errors are tolerable, because the hitting property of H implies that for each such vertex v, the probability of reaching v is small, say $p_{\to v} \leq 1/\text{poly}(n)$, and hence v contributes little to $\mathbb{E}[f]$. Second, there might be pairs of distinct vertices u, v in some layer of f such that starting at u and reading a string $x \in H$ always leads to the same outcome (accept or reject) as starting at v and reading the same string x. Such vertices are effectively merged by our algorithm, even though they are distinct. Intuitively, these errors are also tolerable, because the hitting property of H implies that $p_{u\to} \approx p_{v\to}$ for such pairs.

Let us highlight one of the challenges in implementing the plan described above. In our approximation of the graph structure of f, the vertex set is H_* (modulo the equivalence relation $\sim$), but what exactly are the edges? Intuitively, if we are at a vertex $x \in H_i$ and we read a bit $b \in \{0, 1\}$, we would like to move to the vertex $x \circ b$. The main difficulty is that such a rule is not well-defined, because $x \sim y$ does not necessarily imply $x \circ b \sim y \circ b$. We overcome this difficulty by showing that it suffices to use whichever representative $x \in H_i$ maximizes the acceptance probability.

1.5.3 Techniques for Theorem 4.3

Recall that to prove Theorem 4.3, we must (conditionally) construct a PRG with seed length $O(\log^{1+\alpha} n)$, where $\alpha > 0$ is an arbitrarily small constant. For simplicity, in this overview, we will focus on the case $\alpha = 1/2$, i. e., seed length $O(\log^{3/2} n)$. Recall also that we are focusing on the constant-width case.

The starting point of the construction is the INW PRG, which ϵ-fools constant-width ROBPs over the alphabet $\{0, 1\}^t$ with seed length $O(t + \log(n/\epsilon) \log n)$ [20]. (Nisan's PRG [25] does not achieve the same optimal dependence on t.) Next, we present a reduction, showing how to convert a PRG with moderate error into a hitting set with very small threshold (Theorem 4.4). Hoza and Zuckerman gave a similar reduction [19], but their reduction only applies to binary ROBPs (the case $t = 1$). Our reduction is based on a more sophisticated variant of a key lemma in Hoza and Zuckerman's work [19].

Applying our new reduction to the INW generator, we unconditionally obtain an improved hitting set. The best previous hitting sets had space complexity $O(t + \log^2 n + \log(1/\epsilon) \log n)$ [20] or $O(t \log n + \log^2 n + \log(1/\epsilon))$ [19]. Our new hitting set (Corollary 4.8) achieves the "best of both worlds," with space complexity $O(t + \log^2 n + \log(1/\epsilon))$.

The next step in the proof of Theorem 4.3 is to apply the assumption of Theorem 4.3, converting our hitting set into a PRG. The final step is to use traditional "seed recycling" techniques to trade the excellent dependence on ϵ for an improved dependence on n. Briefly, starting with a length-n ROBP over the alphabet $\{0, 1\}^t$, we first use a randomized sampler [14] to reduce the alphabet size to poly(n). Then we divide our length-n ROBP of interest into blocks of length $m = 2^{\sqrt{\log n}}$. We can fool each chunk to within error $1/\mathrm{poly}(n)$ using a seed of length $O(\log^2 m + \log n) = O(\log n)$. Using the randomized sampler again, this allows us to effectively pay $O(\log n)$ truly random bits and reduce the length of the branching program by a factor of m. After repeating this process $\sqrt{\log n}$ times, the length is reduced to a constant, and we have paid a total of $O(\log^{3/2} n)$ truly random bits.

(To achieve seed length $O(\log^{1+\alpha} n)$, we start the whole process over again and iterate roughly $1/\alpha$ times. This iterative strategy is similar to the work of Hoza and Umans [18], but the specific reductions are different.)

1.6 Related work

We have already referenced most of the work related to this paper, such as work on derandomizing **BPP** using a hitting set [3, 9, 4, 13]. However, a couple of additional papers deserve mention.

1.6.1 BPL $\subseteq$ ZP*L

Our derandomization of **BPL** given a hitting set is similar to Nisan's unconditional proof that **BPL** $\subseteq$ **ZP*L** [26]. To estimate the acceptance probability of a width-n length-n ROBP f, Nisan, like us, interprets a string $x \in \{0,1\}^{\mathrm{poly}(n)}$ as a list of sample inputs, which he uses to compute estimates of $p_{\to v}$ for each vertex v. Nisan's algorithm picks x at random, and then in a similar fashion as our algorithm, performs certain "local tests" at each vertex to verify that the sample inputs are trustworthy. Nisan's local tests can be computed in small space given two-way access to x, and passing the local tests implies that the estimates are close to the corresponding true probabilities. Our local consistency test also satisfies these properties, and indeed, one can obtain an alternative proof that **BPL** $\subseteq$ **ZP*L** from our analysis. However, a technical point is that we use *fresh samples* for each vertex, whereas Nisan uses one set of n-bit sample inputs for all the vertices. This crucial distinction is how we are able to ensure the existence of a polynomial-width *read-once* branching program that verifies Equation (1.1). Unfortunately, using fresh samples breaks Nisan's local tests, hence our new local consistency test.

1.6.2 Deterministically simulating BPL with very low error

Recall that the current best unconditional hitting sets for polynomial-width ROBPs [8, 19, 10, 12, 28, 17] are superior to the state-of-the-art PRGs [25] in the small-ϵ regime. In this work, we present an algorithm for estimating the acceptance probability of a **BPL** algorithm given a hitting set for ROBPs. Taken together, these two facts draw attention to the problem of estimating the acceptance probability of a **BPL** algorithm to within $\pm\epsilon$ in the small-ϵ regime.

Unfortunately, our work does not imply an improved algorithm for this problem.[5] The good news is that Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan recently tackled this same problem with different techniques [1]. They show how to deterministically

[5]To estimate the acceptance probability of a **BPL** algorithm to within $\pm\epsilon$, our algorithm relies on a $\frac{1}{2}$-hitting set for ROBPs of length $\mathrm{poly}(n/\epsilon)$ rather than an ϵ-hitting set for ROBPs of length $\mathrm{poly}(n)$, so we are not able to get any advantage from the recent unconditional hitting set constructions [8, 19, 10, 12, 28, 17].

estimate the acceptance probability of a **BPL** algorithm to within $\pm\epsilon$ in space $O(\log^{3/2} n + \log n \log\log(1/\epsilon))$ [1].

1.7 Outline of this paper

In Section 2, we present our derandomization of **BPL** given a hitting set for polynomial-width ROBPs. In Section 3, we present our deterministic sampler for constant-width ROBPs given a hitting set. In Section 4, we present our theorem on the limitations of this line of work. Finally, in Section 5, we present our derandomization of **BPP** given a hitting set for polynomial-size circuits.

2 Derandomizing BPL given a hitting set

In this section, we show that the acceptance probability of an arbitrary polynomial-width ROBP can be approximated within a small bias in small space, given a certain hitting set. Theorem 1.3 will follow from this.

Theorem 2.1. *Assume there is a $\frac{1}{2}$-hitting set H for width-w' length-n' ROBPs that can be computed in space s. Then the acceptance probability of a given width-w length-n ROBP f can be approximated within a bias $\pm\epsilon$, in space $O(s + \log\frac{wn}{\epsilon})$.*

Here $w' = \left\lceil 9\frac{w^3n^2\log(wn)}{\epsilon^2}\right\rceil$, $n' = \left\lceil 5\frac{w^3n^4\log(wn)}{\epsilon^2}\right\rceil$.

Strictly speaking, Theorem 2.1 ought to be phrased in terms of *families* of ROBPs, to make the space bounds meaningful. That is, we assume there is an algorithm that constructs a $\frac{1}{2}$-hitting set for width-w length-n ROBPs, given w and n as inputs, running in space $s(w, n)$. Then given inputs f, ϵ, Theorem 2.1 should be understood to say that we can estimate $\mathbb{E}[f]$ to within $\pm\epsilon$ in space $O(s(w', n') + \log(wn/\epsilon))$.

We are most interested in the case that ϵ is a small constant, but we remark that when ϵ is very small, the parameters of Theorem 2.1 could be improved by applying the recent amplification technique by Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan [1].

We first give the derandomization and then give the analysis.

2.1 Derandomization based on a local consistency test

For $x \in \{0,1\}^{n'}$, we interpret it as a concatenation of wn "segments" – one segment for each vertex in $V_1 \cup \cdots \cup V_n$. For each $i \in [n]$ and each $v \in V_i$, the segment corresponding to v consists of a concatenation of t "sample strings" of length i, where t is a power of two satisfying $t \geq 4(\frac{wn}{\epsilon})^2 \log(wn)$. Let $\widehat{p}_{\to v}(x)$ be the fraction of strings that lead to v from the start vertex, among these t sample strings for v. When x is clear, we simply denote it as $\widehat{p}_{\to v}$. Also, for $v \in V_0$, we let $\widehat{p}_{\to v} = 1$ if $v = v_{\mathrm{start}}$ and $\widehat{p}_{\to v} = 0$ otherwise.

The derandomization conducts a local consistency test $\mathsf{Test} : \{0,1\}^{n'} \to \{0,1\}$ for every $x \in H$ as follows. For all $i \in [n]$, for all $v \in V_i$, check if

$$\left|\widehat{p}_{\to v} - \left(\sum_{u \in V_{i-1}} \widehat{p}_{\to u} \cdot p_{u \to v}\right)\right| \leq \left(1 + \sum_{u \in V_{i-1}} p_{u \to v}\right)\epsilon' , \tag{2.1}$$

where $\epsilon' = \frac{\epsilon}{2wn}$. If x passes the checks for all v, then $\mathsf{Test}(x) = 1$, otherwise it is 0.

Finally we find an $x \in H$ that passes Test, and output $\widehat{p}_{\to v_{\mathrm{acc}}}(x)$ as the approximation of $\mathbb{E}[f]$.

2.2 Analysis

We now define the "sample verification" function f' of f. For each $x \in \{0,1\}^{n'}$, we set $f'(x) = 1$ if and only if for every vertex v in f,

$$\left|\widehat{p}_{\to v} - p_{\to v}\right| \leq \epsilon' . \tag{2.2}$$

We stress that our derandomization algorithm does not require computing f'; we define f' only for the sake of analysis.

Lemma 2.2. *f' can be computed by a width-w' length-n' ROBP.*

Proof. For each $i \in [n]$ and each $v \in V_i$, we construct an ROBP f'_v which simulates f on each sample string in v's segment and counts how many lead to v. It stores a state of f and a counter value, for a total width of $w \cdot (t+1)$ and a total length $i \cdot t$. f'_v accepts if and only if the counter value is in $[p_{\to v}t - \frac{\epsilon}{2wn}t, p_{\to v}t + \frac{\epsilon}{2wn}t]$.

To construct f', we take the conjunction of f'_v, over all v in f. Note that this is a conjunction of ROBPs over disjoint variables. So we can easily see that f' can be computed by an ROBP with width at most $w(t+1) + 1 \leq \left\lceil 9\frac{w^3 n^2 \log(wn)}{\epsilon^2}\right\rceil$, length at most $tw\sum_{i=1}^{n} i \leq \left\lceil 5\frac{w^3 n^4 \log(wn)}{\epsilon^2}\right\rceil$. □

Lemma 2.3. *The acceptance probability of f' is at least $\frac{1}{2}$.*

Proof. By the construction of f', for each v of f, there are t uniform random samples. For each sample string, the probability that it leads to v from v_{start} in f is $p_{\to v}$. Hence the expected number of samples leading to v from v_{start} is $p_{\to v}t$. So by Hoeffding's inequality, $\Pr[|\widehat{p}_{\to v}t - p_{\to v}t| \geq \frac{\epsilon}{2wn}t] \leq 2 \cdot 2^{-2\log(wn)} \leq \frac{2}{(wn)^2}$. There are wn vertices that need to be tested in f. (For $v \in V_0$, the estimate $\widehat{p}_{\to v}$ is always exactly correct.) Thus by a union bound,

$$\Pr\left[\forall v, \left|\widehat{p}_{\to v} - p_{\to v}\right| \leq \frac{\epsilon}{2wn}\right] \geq 1 - \frac{2}{wn} .$$

This is at least $\frac{1}{2}$ when considering n to be at least some large enough constant. So by the definition of f', its acceptance probability is at least $\frac{1}{2}$. □

Lemma 2.4. *For every $x \in \{0,1\}^{n'}$, if $f'(x) = 1$ then $\mathsf{Test}(x) = 1$.*

Proof. For every $i \in [n]$, every $v \in V_i$,

$$p_{\to v} = \sum_{u \in V_{i-1}} p_{\to u} p_{u \to v} \,, \tag{2.3}$$

by the structure of ROBP. So

$$\begin{aligned}
\left| \widehat{p}_{\to v} - \sum_{u \in V_{i-1}} \widehat{p}_{\to u} p_{u \to v} \right| &= \left| \widehat{p}_{\to v} - p_{\to v} + p_{\to v} - \sum_{u \in V_{i-1}} \widehat{p}_{\to u} p_{u \to v} \right| \\
&= \left| \widehat{p}_{\to v} - p_{\to v} + \sum_{u \in V_{i-1}} p_{\to u} p_{u \to v} - \sum_{u \in V_{i-1}} \widehat{p}_{\to u} p_{u \to v} \right| && \text{(Equation (2.3))} \\
&\le \left| \widehat{p}_{\to v} - p_{\to v} \right| + \sum_{u \in V_{i-1}} \left| p_{\to u} - \widehat{p}_{\to u} \right| p_{u \to v} && \text{(Triangle Inequality)} \\
&\le \left(1 + \sum_{u \in V_{i-1}} p_{u \to v} \right) \epsilon' \,. && \text{(Equation (2.2))} \quad \square
\end{aligned}$$

Lemma 2.5. *For every $x \in \{0,1\}^{n'}$, if* $\mathsf{Test}(x) = 1$ *then* $\left| \widehat{p}_{\to v_{\mathrm{acc}}} - p_{\to v_{\mathrm{acc}}} \right| \le \epsilon$.

Proof. We use induction to show that for the i-th layer of f,

$$\sum_{v \in V_i} \left| \widehat{p}_{\to v} - p_{\to v} \right| \le 2wi\epsilon' \,.$$

For the base case, when $i = 0$, it's trivially true since we set $\widehat{p}_{\to v} = p_{\to v}$ for each $v \in V_0$. For the induction case, assume the hypothesis is true for layer i. Consider layer $i+1$.

$$\begin{aligned}
&\sum_{v\in V_{i+1}} \left|\widehat{p}_{\to v} - p_{\to v}\right| \\
&= \sum_{v\in V_{i+1}} \left|\widehat{p}_{\to v} - \sum_{u\in V_i} p_{\to u} p_{u\to v}\right| && \text{(Equation (2.3))} \\
&= \sum_{v\in V_{i+1}} \left|\widehat{p}_{\to v} - \sum_{u\in V_i} \widehat{p}_{\to u} p_{u\to v} + \sum_{u\in V_i} \widehat{p}_{\to u} p_{u\to v} - \sum_{u\in V_i} p_{\to u} p_{u\to v}\right| \\
&\le \sum_{v\in V_{i+1}} \left(\left|\widehat{p}_{\to v} - \sum_{u\in V_i} \widehat{p}_{\to u} p_{u\to v}\right| + \left|\sum_{u\in V_i} \widehat{p}_{\to u} p_{u\to v} - \sum_{u\in V_i} p_{\to u} p_{u\to v}\right|\right) && \text{(Triangle Inequality)} \\
&\le \sum_{v\in V_{i+1}} \left|\widehat{p}_{\to v} - \sum_{u\in V_i} \widehat{p}_{\to u} p_{u\to v}\right| + \sum_{v\in V_{i+1}} \sum_{u\in V_i} p_{u\to v} \left|\widehat{p}_{\to u} - p_{\to u}\right| && \text{(Triangle Inequality)} \\
&\le \sum_{v\in V_{i+1}} \left(1 + \sum_{u\in V_i} p_{u\to v}\right)\epsilon' + \sum_{v\in V_{i+1}} \sum_{u\in V_i} p_{u\to v} \left|\widehat{p}_{\to u} - p_{\to u}\right| && (\mathsf{Test}(x) = 1) \\
&= 2w\epsilon' + \sum_{u\in V_i} \left|\widehat{p}_{\to u} - p_{\to u}\right| && (2.4) \\
&\le 2w\epsilon' + 2wi\epsilon' && \text{(Induction hypothesis)} \\
&= 2w\cdot(i+1)\cdot\epsilon'\,.
\end{aligned}$$

Here Equation (2.4) is due to structures of ROBPs. Note that

$$\sum_{v\in V_{i+1}} \sum_{u\in V_i} p_{u\to v} = \sum_{u\in V_i} \sum_{v\in V_{i+1}} p_{u\to v} = w\,.$$

Also due to the same reasoning,

$$\sum_{v\in V_{i+1}} \sum_{u\in V_i} p_{u\to v}\left|\widehat{p}_{\to u} - p_{\to u}\right| = \sum_{u\in V_i} \sum_{v\in V_{i+1}} p_{u\to v}\left|\widehat{p}_{\to u} - p_{\to u}\right| = \sum_{u\in V_i} \left|\widehat{p}_{\to u} - p_{\to u}\right|.$$

As a result, for the last layer,

$$\left|\widehat{p}_{\to v_{\mathrm{acc}}} - p_{\to v_{\mathrm{acc}}}\right| \le \sum_{v\in V_n} \left|\widehat{p}_{\to v} - p_{\to v}\right| \le 2wn\epsilon' = \epsilon\,. \qquad \square$$

Lemma 2.6. *The derandomization is in space $O(s + \log\frac{wn}{\epsilon})$.*

Proof. Since H is computable in space s, for every $x \in H$ we can output any specified bit of it in space $O(s + \log n')$. So when considering the space for computing $\mathsf{Test}(x)$ and $\widehat{p}_{\to v}(x)$, we can just regard x as an input string and only consider working space.

Given vertex v in f, we first consider the space for computing $\widehat{p}_{\to v}$. By the definition of $\widehat{p}_{\to v}$, we can locate the starting position of the t samples for v, taking space $O(\log \frac{wn}{\epsilon})$. From there, we read the t samples one by one. For each sample, we run f from v_{start} to the layer of v to test if the sample leads to v. We use a counter c to record the number of samples leading to v. Then compute $\widehat{p}_{\to v}$ as c/t. Since t is a power of two, we can store this number exactly, with no rounding errors. So this step takes space $O(\log(wn)) + O(\log t) = O(\log \frac{wn}{\epsilon})$. Thus the whole computation is in space $O(\log \frac{wn}{\epsilon})$.

Next we consider Test. By the definition of Test, for every $i \in [n]$, for each vertex $v \in V_i$, we only need to compute $\widehat{p}_{\to v}$, $\sum_{u \in V_{i-1}} \widehat{p}_{\to u} \cdot p_{u \to v}$ and then test Equation (2.1). This again takes space $O(\log \frac{wn}{\epsilon})$. (Note that the algorithm for $\mathsf{Test}(x)$ reads the bits of x multiple times, because the value $\widehat{p}_{\to v}$ is used when testing Equation (2.1) for v itself and then again when testing Equation (2.1) for v's out-neighbors.)

So the overall space of the derandomization is $O(s + \log \frac{wn}{\epsilon})$. □

Proof of Theorem 2.1. Given a width-w length-n ROBP f, by Lemma 2.2, the function f' can be computed by a width-w' length-n' ROBP. By Lemma 2.3, the acceptance probability of f' is at least $1/2$. Since H is a $\frac{1}{2}$-hitting set for width-w' length-n' ROBPs, there exists $x \in H$ such that $f'(x) = 1$. So by Lemma 2.4, there is an $x \in H$ such that $\mathsf{Test}(x) = 1$. Hence we can exhaustively search though H to find an x which passes Test. Further, by Lemma 2.5, for each such x, $|\widehat{p}_{\to v_{\mathrm{acc}}}(x) - p_{\to v_{\mathrm{acc}}}| \le \epsilon$. This shows the derandomization outputs the desired approximation for $p_{\to v_{\mathrm{acc}}}$.

By Lemma 2.6, the derandomization can be done in space $O(s + \log \frac{wn}{\epsilon})$. □

Theorem 1.3 is directly implied from Theorem 2.1. The proof is straightforward by applying the well-known transformation between log-space computations and ROBPs.

3 Deterministic samplers for constant-width ROBPs

In this section, we will show how to use hitting sets to construct deterministic samplers for constant-width ROBPs, thereby proving Theorem 1.7. Most of the work will go toward establishing the following reduction, which is meaningful even for slightly super-constant width.

Theorem 3.1. *Let $w, n \in \mathbb{N}$ and let $\epsilon > 0$. Assume there is an $(\frac{\epsilon}{2n})$-hitting set H for width-$(\binom{w}{2} + 1)$ length-n ROBPs computable in space s. Then there is a deterministic ϵ-sampler for width-w length-n ROBPs that runs in space $O(s + w \log(n/\epsilon))$.*

Like Theorem 2.1, Theorem 3.1 technically ought to be phrased in terms of families of ROBPs. We should also clarify the model of space-bounded oracle algorithm. We assume that the sampler has write-only access to a "query tape" where it can write down an n-bit query string (the query string does not count against the sampler's space complexity). The sampler can then enter a special "query" state, which returns the result of the query into the algorithm's state and clears the query tape. This natural model was perhaps first studied by Ladner and Lynch [22].

3.1 Setting up the reduction

Toward proving Theorem 3.1, we begin by setting up some notation. For any ROBP f and a string $x \in \{0,1\}^{\leq n}$, let $v_f(x)$ be the vertex reached when f reads x. Furthermore, define

$$p_f(x) = \mathbb{E}[f(x \circ U_{n-|x|})]\,,$$

i. e., $p_f(x) = p_{v_f(x)\rightarrow}$.

Now, let $H \subseteq \{0,1\}^n$ be an ϵ_H-hitting set for width-$(\binom{w}{2}+1)$ ROBPs. For $i \leq n$, let H_i be the set of i-bit prefixes of strings in H, i. e., $H_i = \{x_1 x_2 \dots x_i : x \in H\}$. One can verify that H_i is an ϵ_H-hitting set for width-$(\binom{w}{2}+1)$ length-i ROBPs.

Let f be the width-w ROBP to which we have oracle access. Let λ denote the empty string. Our goal is to estimate $p_f(\lambda)$. For each $i \leq n$, define an equivalence relation $\sim$ on $\{0,1\}^i$ by the rule

$$x \sim y \iff \forall z \in H_{n-i}, f(x \circ z) = f(y \circ z)\,.$$

Lemma 3.2. *If $x \sim y$, then $|p_f(x) - p_f(y)| < \epsilon_H$.*

Proof. Let $i = |x| = |y|$. Define $g\colon \{0,1\}^{n-i} \rightarrow \{0,1\}$ by

$$g(z) = f(x \circ z) \oplus f(y \circ z)\,.$$

The function $g(z)$ can be computed by an ROBP of width $\binom{w}{2}+1$: we have one state in g for each unordered pair of states in f to run the computations $f(x \circ z), f(y \circ z)$ in parallel, along with one additional $\perp$ state in g to indicate that the two computations converged to the same state. If $|p_f(x) - p_f(y)| \geq \epsilon_H$, then H_{n-i} hits g, hence $x \nsim y$. □

Let $[x]$ denote the equivalence class of x, so $[x] \subseteq \{0,1\}^{|x|}$. Our deterministic sampler will be based on numbers $\widetilde{p}_f([x]) \in [0,1]$ for each equivalence class $[x]$. The definition of $\widetilde{p}_f$ will ensure that $\widetilde{p}_f([x]) \approx p_f(x)$ for typical values of x, although there might be some anomalous values of x where $\widetilde{p}_f([x]) \not\approx p_f(x)$.

The definition of $\widetilde{p}_f([x])$ is inductive. For the base case, when $x \in \{0,1\}^n$, define $\widetilde{p}_f([x]) = f(x)$. This is well-defined, because $x \sim y \implies f(x) = f(y)$. For the inductive step, suppose $x \in \{0,1\}^i$ with $i < n$. Define

$$\widetilde{p}_f([x]) = \max_{x' \in H_i \cap [x]} \left(\frac{1}{2}\widetilde{p}_f([x' \circ 0]) + \frac{1}{2}\widetilde{p}_f([x' \circ 1]) \right)\,, \tag{3.1}$$

with the convention that $\widetilde{p}_f([x]) = 0$ if $H_i \cap [x] = \varnothing$. Our sampler will output[6] $\widetilde{p}_f([\lambda])$. (In Section 3.3, we will explain in more detail how to compute $\widetilde{p}_f([\lambda])$ in a space-efficient manner.)

[6]Actually the sampler's output differs slightly from $\widetilde{p}_f([\lambda])$ due to rounding errors.

3.2 Correctness

The upper bound on $\widetilde{p}_f([x])$ is straightforward:

Claim 3.3. *For every i, for every $x \in \{0,1\}^{n-i}$,*

$$\widetilde{p}_f([x]) \leq p_f(x) + i\epsilon_H\,.$$

Proof. We proceed by induction on i. In the base case $i = 0$, $\widetilde{p}_f([x]) = f(x) = p_f(x)$. For the inductive step $i > 0$, we consider two cases. If $H_i \cap [x] = \varnothing$, then $\widetilde{p}_f([x]) = 0$ and the claim is trivial. Otherwise, there is some $x' \in H_i \cap [x]$ such that

$$\begin{aligned}
\widetilde{p}_f([x]) &= \frac{1}{2}\widetilde{p}_f([x' \circ 0]) + \frac{1}{2}\widetilde{p}_f([x' \circ 1]) && \text{(Equation (3.1))}\\
&\leq \frac{1}{2}p_f(x' \circ 0) + \frac{1}{2}p_f(x' \circ 1) + (i-1)\epsilon_H && \text{(Induction)}\\
&= p_f(x') + (i-1)\epsilon_H\\
&< p_f(x) + i\epsilon_H && \text{(Lemma 3.2.)}
\end{aligned}$$

□

The lower bound is a little more subtle. If u is a vertex in layer i of f, we say that u is *H-reachable* if there is some $x \in H_i$ with $v_f(x) = u$. Otherwise, we say that u is *H-unreachable*. Let $\widetilde{f}$ be a width-$(w+1)$ ROBP obtained from f by replacing all H-unreachable nodes with reject nodes.[7]

Claim 3.4. *For every i, for every $x \in \{0,1\}^{n-i}$,*

$$\widetilde{p}_f([x]) \geq p_{\widetilde{f}}(x)\,.$$

Proof. We proceed by induction on i. In the base case $i = 0$, $\widetilde{p}_f([x]) = f(x) \geq \widetilde{f}(x) = p_{\widetilde{f}}(x)$. For the inductive step $i > 0$, we consider two cases. If f visits some H-unreachable node when it reads x, then $p_{\widetilde{f}}(x) = 0$ and the claim is trivial. Therefore, assume that when f reads x, every node visited is H-reachable. Then there is some $x' \in H_{n-i}$ such that $v_f(x) = v_f(x')$. Of course when f reads x', every node visited is H-reachable, so

$$v_{\widetilde{f}}(x') = v_f(x') = v_f(x) = v_{\widetilde{f}}(x)\,.$$

Therefore,

$$\begin{aligned}
p_{\widetilde{f}}(x) = p_{\widetilde{f}}(x') &= \frac{1}{2}p_{\widetilde{f}}(x' \circ 0) + \frac{1}{2}p_{\widetilde{f}}(x' \circ 1)\\
&\leq \frac{1}{2}\widetilde{p}_f([x' \circ 0]) + \frac{1}{2}\widetilde{p}_f([x' \circ 1]) && \text{(Induction)}\\
&\leq \widetilde{p}_f(x) && \text{(Equation (3.1).)}
\end{aligned}$$

(The last inequality uses the fact that $v_f(x) = v_f(x')$ and hence $x \sim x'$.) □

[7] More precisely, add an extra node to each layer labeled $\perp$. In layers prior to the final layer, both outgoing edges from $\perp$ lead to $\perp$, and both outgoing edges from H-unreachable nodes lead to $\perp$. In the final layer, H-unreachable nodes and $\perp$ are reject nodes.

Corollary 3.5. $|\widetilde{p}_f([\lambda]) - \mathbb{E}[f]| \leq n \cdot \epsilon_H$.

Proof. By Claim 3.3,

$$\widetilde{p}_f([\lambda]) \leq p_f(\lambda) + n \cdot \epsilon_H = \mathbb{E}[f] + n \cdot \epsilon_H .$$

In the other direction, by Claim 3.4,

$$\widetilde{p}_f([\lambda]) \geq p_{\widetilde{f}}(\lambda) = \mathbb{E}[\widetilde{f}] .$$

Define $g \colon \{0,1\}^n \to \{0,1\}$ by

$$g(x) = 1 \iff \text{when } f \text{ reads } x, \text{ an } H\text{-unreachable node is visited.}$$

Then g can be computed by a width-$(w+1)$ ROBP by a construction very similar to that of $\widetilde{f}$. By construction, g rejects every string in H. Therefore, $\mathbb{E}[g] < \epsilon_H$. Furthermore, $g(x) = 0 \implies f(x) = \widetilde{f}(x)$. Therefore, $|\,\mathbb{E}[\widetilde{f}] - \mathbb{E}[f]| < \epsilon_H$, so $\widetilde{p}_f([\lambda]) > \mathbb{E}[f] - \epsilon_H$. ◻

3.3 Efficiently computing $\widetilde{p}_f([\lambda])$

To complete the proof of Theorem 3.1, we just need to show how to efficiently compute $\widetilde{p}_f([\lambda])$. This is fairly straightforward from the definitions; the details follow.

Proof of Theorem 3.1. Say a string $x \in H_i$ is a *representative* if it is the lexicographically first element of $[x] \cap H_i$. Let $x^{(i,1)}, x^{(i,2)}, \ldots$ be an enumeration of the representatives in H_i in lexicographic order. Given i, j, and oracle access to f, one can compute $x^{(i,j)}$ in space $O(s)$.

Our sampler works its way backward through the branching program, starting at layer n and ending with layer 0. The sampler stores data about layer i and uses it when processing layer $i-1$. Specifically, the data stored regarding layer i consists of a list of numbers $p_{i,1}, p_{i,2}, \ldots,$ with the interpretation $p_{i,j} = \widetilde{p}_f([x^{(i,j)}])$, or rather $p_{i,j} \approx \widetilde{p}_f([x^{(i,j)}])$ due to rounding error.

For layer n, we can compute this value exactly by setting $p_{i,j} = f(x^{(i,j)})$. Given these values for layer $i+1$, define $\overline{p}_f(x)$ for each $x \in \{0,1\}^{i+1}$ by

$$\overline{p}_f(x) = \begin{cases} p_{i+1,j} & \text{if } j \text{ is such that } x^{(i+1,j)} \sim x \\ 0 & \text{if no such } j \text{ exists.} \end{cases}$$

(Note that j is unique if it exists, and note that $\widetilde{p}_f([x]) = 0$ if such a j does not exist.) Then, we compute each value $p_{i,j}$ by the rule

$$p_{i,j} := \max_{x' \in H_i \cap [x^{(i,j)}]} \left(\frac{1}{2}\overline{p}_f(x' \circ 0) + \frac{1}{2}\overline{p}_f(x' \circ 1) \right) , \tag{3.2}$$

with the convention that $p_{i,j} = 0$ if $H_i \cap [x^{(i,j)}] = \varnothing$.

The sampler performs the arithmetic in Equation (3.2) to within $\lceil \log(2n/\epsilon) \rceil$ bits of precision. This ensures that the rounding error is not too large in each step; by induction, $|p_{i,j} - \widetilde{p}_f([x^{(i,j)}])| \leq \frac{\epsilon(n-i)}{2n}$. The sampler outputs $p_{0,1}$, which is within ϵ of $\mathbb{E}[f]$ by Corollary 3.5, since $\epsilon_H = \frac{\epsilon}{2n}$.

The number of vertices in each layer of f is at most w, so the number of equivalence classes in $\{0,1\}^i$ is also at most w. Therefore, there are at most w representatives in H_i, and hence there are only w numbers $p_{i,j}$ being stored for each layer. Storing those numbers for the layer currently being processed and the layer most recently processed takes $O(w \log(n/\epsilon))$ bits of space, so overall, the space complexity of the sampler is $O(s + w \log(n/\epsilon))$ as claimed. □

Interestingly, the sampler of Theorem 3.1 can be implemented to be non-adaptive, because it only queries f at strings of the form $x \circ y$ or $x \circ b \circ z$, where $x \in H_i$, $y \in H_{n-i}$, $b \in \{0,1\}$, and $z \in H_{n-i-1}$.

3.4 Applying the reduction

Proof of Theorem 1.8. Hoza and Zuckerman constructed an $\left(\frac{\epsilon}{2n}\right)$-hitting set H even for *polynomial*-width ROBPs that can be computed in space $O(\log^2 n + \log(1/\epsilon))$ [19]. Combining this result with Theorem 3.1 immediately proves Theorem 1.8. □

To prove Theorem 1.7, we must first amplify the assumed $\frac{1}{2}$-hitting set to get an $\left(\frac{\epsilon}{2n}\right)$-hitting set. This is straightforward, although we must pay a small penalty in terms of width, length, and cardinality.

Lemma 3.6. *Suppose H is a $\frac{1}{2}$-hitting set for width-$(w+1)$ length-(nm) ROBPs. Divide each string $x \in H$ into blocks of length n, $x = x^{(1)} \circ x^{(2)} \circ \cdots \circ x^{(m)}$. Let $H' = \{x^{(i)} : x \in H, i \in [m]\}$. Then H' is a $(\frac{1}{m})$-hitting set for width-w length-n ROBPs.*

Proof. Let f be a width-w length-n ROBP with $\mathbb{E}[f] \geq 1/m$. Define $g \colon (\{0,1\}^n)^m \to \{0,1\}$ by

$$g(x^{(1)} \circ \cdots \circ x^{(m)}) = \bigvee_{i \in [m]} f(x^{(i)}) .$$

Then g can be computed by a width-$(w+1)$ ROBP. Furthermore,

$$\mathbb{E}[g] = 1 - (1 - \mathbb{E}[f])^m \geq 1 - \left(1 - \frac{1}{m}\right)^m > \frac{1}{2} .$$

Therefore, H hits g, hence H' hits f. □

Combining Lemma 3.6 with Theorem 3.1 proves the following corollary, which immediately implies Theorem 1.7.

Corollary 3.7. *Let $w, n \in \mathbb{N}$ and let $\epsilon > 0$. Let $w' = \binom{w}{2} + 2$ and $n' = n \cdot \lceil 2n/\epsilon \rceil$. Assume there is a $\frac{1}{2}$-hitting set H for width-w' length-n' ROBPs computable in space s. Then there is a deterministic ϵ-sampler for width-w length-n ROBPs that runs in space $O(s + w \log(n/\epsilon))$.*

4 Using a hypothetical stronger reduction to construct PRGs

To directly compare hitting sets and PRGs, it is convenient to address the strings in the hitting set using a *hitting set generator* (HSG).

Definition 4.1. An ϵ*-HSG* for $\mathcal{F}$ is a function $G\colon \{0,1\}^s \to \{0,1\}^n$ such that $G(\{0,1\}^s)$ is an ϵ-hitting set for $\mathcal{F}$.

In our theorem statements so far, we have been somewhat informal with the distinction between an individual generator vs. a family of generators. Since the result we discuss in this section is more "meta" than our other results, we will make a precise definition for clarity's sake.

Definition 4.2. Let $s(n,t,\epsilon)$ be a space-constructible[8] function. An *explicit PRG (HSG) family* for width-w large-alphabet ROBPs with seed length s is a uniform algorithm G that takes as input the parameters n,t,ϵ and a string $y \in \{0,1\}^{s(n,t,\epsilon)}$ and outputs a string $G_{n,t,\epsilon}(y) \in \{0,1\}^{tn}$, interpreted as a length-n string over the alphabet $\{0,1\}^t$. The algorithm runs in space $O(s(n,t,\epsilon))$, and for each fixed n,t,ϵ, we require that $G_{n,t,\epsilon}$ is an ϵ-PRG (ϵ-HSG) for width-w length-n ROBPs over the alphabet $\{0,1\}^t$.

The assumption of Theorem 4.3 below says that hitting sets can be upgraded into PRGs with essentially no loss: the width parameter remains the same, and the seed length only increases by a constant factor, for any arbitrary setting of n,t,ϵ. This is only for simplicity's sake. The proof would still go through even if the parameters deteriorated a little when moving from hitting sets to PRGs.

Theorem 4.3. *Let w be a constant. Assume that for every space-constructible function $s(n,t,\epsilon)$, if there exists an explicit HSG family for width-w large-alphabet ROBPs with seed length s, then there exists a space-constructible function $s'(n,t,\epsilon) = O(s(n,t,\epsilon))$ and an explicit PRG family for width-w large-alphabet ROBPs with seed length s'. Then for every constant $\alpha > 0$, there exists an explicit PRG family for width-w large-alphabet ROBPs with seed length $s_\alpha(n,t,\epsilon)$, where*

$$s_\alpha(n,t,\epsilon) = O(t + \log(n/\epsilon)\log^\alpha n)\,.$$

4.1 From PRGs with moderate error to HSGs with tiny threshold

As outlined in Section 1.5.3, the proof of Theorem 4.3 is based on two reductions. The first reduction shows how to convert any PRG with inverse polynomial error into an ϵ-HSG for any ϵ, and the second reduction shows how to convert a PRG with a good dependence on ϵ into a PRG with a good dependence on n.

In this section, we present the first reduction. In the regime $n \geq w$, our reduction is a generalization of Hoza and Zuckerman's reduction [19] to the large-alphabet case $t \gg 1$.

Theorem 4.4. *Let $w,n,t \in \mathbb{N}$ and let $\epsilon > 0$. Assume there is a $(\frac{1}{2w^3n^2})$-PRG G for width-w length-n ROBPs over the alphabet $\{0,1\}^t$, with seed length and space complexity bounded by s. Then there is an ϵ-hitting set H for width-w length-n ROBPs over the alphabet $\{0,1\}^t$, computable in space $O(s+t+\log(wn/\epsilon))$.*

[8] I. e., given n,t,ϵ, the value $s(n,t,\epsilon)$ can be computed in space $O(s(n,t,\epsilon))$.

(Just like Theorem 2.1 and Theorem 3.1, Theorem 4.4 technically ought to be phrased in terms of families of ROBPs.)

4.1.1 Construction of the hitting set H

Our hitting set H relies on a hitting set H_{rect} for *combinatorial rectangles* [23]. Recall that a combinatorial rectangle over alphabet Γ of dimension r is a function $g\colon \Gamma^r \to \{0,1\}$ of the form $g(x_1,\dots,x_r) = g_1(x_1) \wedge \cdots \wedge g_r(x_r)$. Without loss of generality, assume $\epsilon < \frac{1}{w^2n^2}$ and $s \geq t$. The algorithm to enumerate H is as follows.

1. For all $r \in \left\{1, 2, \dots, \left\lfloor \frac{\log(1/\epsilon)}{\log(wn)} \right\rfloor \right\}$:
   (a) Let $H_{\text{rect}} \subseteq (\{0,1\}^s)^{2r-1}$ be an ϵ^4-hitting set for combinatorial rectangles over alphabet $\{0,1\}^s$ of dimension $2r-1$.
   (b) For all sequences $(x_1, y_1, x_2, y_2, \dots, x_{r-1}, y_{r-1}, x_r) \in H_{\text{rect}}$ and for all sequences of nonnegative integers $(n_1,\dots,n_r)$ satisfying $n_1 + n_2 + \cdots + n_r = n - r + 1$, output the (nt)-bit string

$$(G(x_1)|_{n_1 t}) \circ (y_1|_t) \circ (G(x_2)|_{n_2 t}) \circ (y_2|_t) \circ \cdots \circ (y_{r-1}|_t) \circ (G(x_r)|_{n_r t})\,. \tag{4.1}$$

In Equation (4.1), the notation $y|_t$ denotes the t-bit prefix of the bitstring y. The key difference between our construction and Hoza and Zuckerman's original hitting set construction [19] is the presence of the strings y_i, which do not pass through the PRG G.

4.1.2 Proof of correctness

Hoza and Zuckerman's reduction was based on a simple structural lemma for ROBPs [19, Lemma 1]. Toward proving the correctness of H, we will now prove a new variant of that lemma, applicable to ROBPs over a large alphabet. For two vertices u, v in an ROBP f, write $u \rightsquigarrow v$ if there is an edge from u to v. Let $\rightsquigarrow^*$ be the reflexive transitive closure of $\rightsquigarrow$, i. e., $u \rightsquigarrow^* v$ if $u = v$ or there is a path from u to v.

The way to think about the following Lemma 4.5 is to suppose that one is choosing a route from u to v_{acc}. Lemma 4.5 suggests two vertices $v \rightsquigarrow u'$ that one could visit on the way. Item 2 says that it is not difficult to find v. Item 3 says that if one can make it to u', it will be quite a bit easier to find v_{acc} from there. Item 4 says that overall, visiting v and u' is only a mild detour.

In general, in any ROBP over the alphabet Σ, if $v \rightsquigarrow u'$, then $p_{v\to u'} \geq 1/|\Sigma|$. In Hoza and Zuckerman's lemma [19, Lemma 1], they assume $\Sigma = \{0,1\}$, and they use the fact that therefore $p_{v\to u'} \geq 1/2$. At a high level, the reason we need a new structural lemma is that if Σ is large, $p_{v\to u'}$ might be small. Indeed, observe that Lemma 4.5 does not guarantee any lower bound on $p_{v\to u'}$.

Lemma 4.5. *Let f be a width-w, length-n ROBP over any alphabet. Let u be a vertex in f, and assume $0 < p_{u\to} \leq \frac{1}{wn}$. Then there is a pair of vertices (v, u') in f such that:*

1. $u \rightsquigarrow^* v \rightsquigarrow u'$.

2. $p_{u \to v} \geq \frac{1}{w^3 n^2}$.

3. $p_{u' \to} \geq wn \cdot p_{u \to}$.

4. $p_{u \to v} \cdot p_{v \to u'} \cdot p_{u' \to} \geq \frac{p_{u \to}}{w^2 n}$.

Proof. Suppose some pair (v, u') satisfies Item 1, but it violates Item 4. For such a pair, if we take a random walk from u, the probability that we visit v, u', and v_{acc} is less than $\frac{p_{u \to}}{w^2 n}$. The number of such pairs is at most $w^2 n$, so by the union bound, when we start at u and read random bits, the probability that we visit *any* such pair and v_{acc} is less than $p_{u \to}$. Therefore, there is some path from u to v_{acc} that never visits such a pair.

Let u' be the first vertex along that path that satisfies Item 3. (Such a u' exists, because if nothing else we can let $u' = v_{\mathrm{acc}}$.) Let v be the vertex immediately preceding u' in the path. (This makes sense, because $p_{u \to} < wn \cdot p_{u \to}$, so $u' \neq u$.) This pair clearly satisfies Item 1, Item 3, and Item 4; all that remains is to verify Item 2. Indeed,

$$\begin{aligned} \frac{p_{u \to}}{p_{u \to v}} &\leq w^2 n \cdot p_{v \to u'} \cdot p_{u' \to} && \text{(Item 4)} \\ &\leq w^2 n \cdot p_{v \to} \\ &< w^2 n \cdot wn \cdot p_{u \to}, \end{aligned}$$

where the last inequality holds because u' is the *first* vertex in the path satisfying Item 3, and v precedes u', so v must not satisfy Item 3. Rearranging completes the proof. □

Corollary 4.6. *Let $0 < \epsilon \leq \frac{1}{wn}$. Let f be a width-w, length-n ROBP over any alphabet with $\mathbb{E}[f] \geq \epsilon$. Then there is a sequence of vertices*

$$v_{\mathrm{start}} = u_1 \rightsquigarrow^* v_1 \rightsquigarrow u_2 \rightsquigarrow^* v_2 \rightsquigarrow \cdots \rightsquigarrow u_r \rightsquigarrow^* v_r = v_{\mathrm{acc}}$$

such that:

1. *For every i, $p_{u_i \to v_i} \geq \frac{1}{w^3 n^2}$.*

2. $r \leq \frac{\log(1/\epsilon)}{\log(wn)}$.

3. $p_{u_1 \to v_1} \cdot p_{v_1 \to u_2} \cdot p_{u_2 \to v_2} \cdots p_{v_{r-1} \to u_r} \cdot p_{u_r \to v_r} \geq \epsilon^3$.

Proof. We define the sequence inductively, starting with $u_1 = v_{\mathrm{start}}$. Assume we've defined $u_1, v_1, u_2, v_2, \ldots, u_i$. If $p_{u_i \to} \geq \frac{1}{w^3 n^2}$, then set $r = i$, set $v_i = v_{\mathrm{acc}}$, and terminate the sequence. Otherwise, let (v_i, u_{i+1}) be the vertices provided by plugging $u = u_i$ into Lemma 4.5.

Item 1 of Lemma 4.5 implies that $u_i \rightsquigarrow^* v_i$ and $v_i \rightsquigarrow u_{i+1}$. Item 1 is guaranteed by Item 2 of Lemma 4.5 and the termination condition. By Item 3 of Lemma 4.5, $p_{u_{i+1} \to} \geq wn \cdot p_{u_i \to}$, which implies Item 2. Finally, iteratively applying Item 4 of Lemma 4.5 shows that

$$p_{u_1 \to v_1} \cdot p_{v_1 \to u_2} \cdot p_{u_2 \to v_2} \cdots p_{v_{r-1} \to u_r} \cdot p_{u_r \to v_r} \geq \frac{p_{u_1 \to}}{(w^2 n)^r} \geq \epsilon^3,$$

i. e., Item 3 holds. □

We are now ready to complete the proof of correctness of our hitting set H.

Claim 4.7. *If f is a width-w length-n ROBP over the alphabet $\{0,1\}^t$ with $\mathbb{E}[f] \geq \epsilon$, then $f^{-1}(1) \cap H \neq \emptyset$.*

Proof. Let $u_1 \rightsquigarrow^* v_1 \rightsquigarrow \cdots \rightsquigarrow u_r \rightsquigarrow^* v_r$ be the sequence of vertices guaranteed by Corollary 4.6. Let n_i be the distance from u_i to v_i. Let $g\colon (\{0,1\}^s)^{2r-1} \to \{0,1\}$ be the following combinatorial rectangle:

$$g(x_1, y_1, x_2, y_2, \ldots, x_{r-1}, y_{r-1}, x_r) = 1 \iff$$
$$\forall i \in [r],\ G(x_i)|_{n_i t} \text{ leads from } u_i \text{ to } v_i \ \text{ and } \ \forall i \in [r-1],\ y_i|_t \text{ leads from } v_i \text{ to } u_{i+1}\,.$$

By Item 1 of Corollary 4.6, $p_{u_i \to v_i} \geq \frac{1}{w^3 n^2}$. Since G has error $\frac{1}{2w^3n^2}$,

$$\Pr[G(U) \text{ leads from } u_i \text{ to } v_i] \geq \frac{1}{2} p_{u_i \to v_i}\,.$$

Therefore, by Item 3 of Corollary 4.6, $\mathbb{E}[g] \geq \epsilon^3 \cdot 2^{-r} \geq \epsilon^4$. Therefore, there is some sequence $(x_1, y_1, \ldots, y_{r-1}, x_r) \in H_{\text{rect}}$ that hits g. By construction, the corresponding element of H is accepted by f. ◻

4.1.3 Efficiency

Proof of Theorem 4.4. To complete the proof of Theorem 4.4, let us analyze the space complexity of H. We use an ϵ^4-hitting set for combinatorial rectangles H_{rect} constructed by Linial, Luby, Saks, and Zuckerman [23]. Specifically, we use their simple "low-dimension" construction [23, Section 5]. In general, for combinatorial rectangles over alphabet Γ of dimension r, that hitting set can be enumerated in space $O(r + \log(|\Gamma|/\epsilon))$. In our case, the space complexity bound becomes $O(s + \log(1/\epsilon))$.

The number r can be stored using $O(\log\log(1/\epsilon))$ bits of space. The integers $n_1, \ldots, n_r$ can be straightforwardly stored using $O(r \log n) = O(\log(1/\epsilon))$ bits of space. Thus, overall, the space complexity of enumerating H is $O(s + \log(1/\epsilon))$. (Recall that we assumed without loss of generality that $\epsilon < \frac{1}{w^2n^2}$ and $s \geq t$.) ◻

4.2 Application: Unconditional improved hitting sets for large-alphabet ROBPs

As outlined in Section 1.5.3, plugging the classic INW generator [20] into the reduction of Theorem 4.4 already gives something interesting: an improved hitting set for large-alphabet ROBPs, even of polynomial width.

Corollary 4.8. *Let $w, n, t \in \mathbb{N}$ and let $\epsilon > 0$. There is an ϵ-hitting set H for width-w length-n ROBPs over the alphabet $\{0,1\}^t$, computable in space $O(t + \log(wn)\log n + \log(1/\epsilon))$.*

4.3 Trading a good dependence on ϵ for a good dependence on n

Recall that to prove Theorem 4.3, we must (conditionally) construct a PRG with a good dependence on n. So far, unconditionally, Theorem 4.4 has provided us with an HSG with a good dependence on ϵ. The assumption of Theorem 4.3 allows us to convert that HSG into a PRG with the same seed length, $O(t + \log^2 n + \log(1/\epsilon))$ (for width w, a constant). In this section, we show how to convert that PRG into another PRG with seed length $O(t+\log^{3/2} n+\log(1/\epsilon)\sqrt{\log n})$, i. e., we improve the dependence on n at the expense of a worse dependence on ϵ. That follows from setting $\alpha = 1/2$ in the following more general reduction.

Lemma 4.9. *Let $\alpha \in (0, 1)$ be a constant. Let $w, n, t \in \mathbb{N}$ and $\epsilon > 0$. Define $m = \left\lceil 2^{(\log n)^{1-\alpha}} \right\rceil$ and $d = \lceil C \log(n/\epsilon)\rceil$, where C is an appropriate constant. Assume there is an $(\frac{\epsilon}{4n})$-PRG G for width-w length-m ROBPs over the alphabet $\{0, 1\}^d$ with seed length and space complexity bounded by s. Then there is an ϵ-PRG G' for width-w length-n ROBPs over the alphabet $\{0, 1\}^t$ with seed length and space complexity $O(t + s \cdot \log^\alpha n + \log(wn/\epsilon))$.*

As usual, Lemma 4.9 should technically be phrased in terms of families of ROBPs. As suggested in Section 1.5.3, the proof of Lemma 4.9 is an application of traditional seed-recycling techniques, similar to classic constructions of PRGs for space-bounded computation [25, 20, 27]. Our construction and analysis are especially similar to Armoni's work [5].

One difference is that we use randomized samplers rather than extractors for convenience; in this respect, our construction is similar to a variant of the INW generator [20] described by Braverman, Cohen, and Garg [8] as a warm-up to their main construction. In particular, we rely on the following randomized sampler by Goldreich and Wigderson [14].

Theorem 4.10 ([14, Lemma 6.6])**.** *For all $t \in \mathbb{N}$, $\delta > 0$, there exists a function $\mathsf{Samp}_{t,\delta} \colon \{0, 1\}^t \times \{0, 1\}^{O(\log(1/\delta))} \to \{0, 1\}^t$ such that for any[9] function $f \colon \{0, 1\}^t \to [0, 1]$,*

$$\Pr_x\left[\left|\mathbb{E}_y[f(\mathsf{Samp}_{t,\delta}(x, y))] - \mathbb{E}[f]\right| \le \delta\right] \ge 1 - \delta\,.$$

Furthermore, given t, δ, x, y as inputs, $\mathsf{Samp}_{t,\delta}(x, y)$ can be computed in space $O(t)$.

We will recursively use the following basic PRG, which stretches $t + dn$ bits to tn bits where $d = O(\log(1/\delta))$. It might be helpful to think of the case $t = 100d$.

Lemma 4.11. *Let t, δ be arbitrary, and let $\mathsf{Samp}_{t,\delta} \colon \{0, 1\}^t \times \{0, 1\}^d \to \{0, 1\}^t$ be the randomized sampler of Theorem 4.10. Define $G_0 \colon \{0, 1\}^t \times (\{0, 1\}^d)^n \to (\{0, 1\}^t)^n$ by*

$$G_0(x, z_1 \circ \cdots \circ z_n) = \mathsf{Samp}_{t,\delta}(x, z_1) \circ \cdots \circ \mathsf{Samp}_{t,\delta}(x, z_n)\,.$$

Then G_0 fools width-w length-n ROBPs over the alphabet $\{0, 1\}^t$ with error $2\delta w^2 n$.

[9] Goldreich and Wigderson analyze the case that f is $\{0, 1\}$-valued, but the $[0, 1]$-valued case automatically follows with only a quadratic loss in δ.

The proof of Lemma 4.11 is straightforward, and we omit it. When reading the proof of Lemma 4.9, it might be helpful to keep in mind that all "x" variables are strings of length t, all "y" variables are strings of length s, and all "z" variables are strings of length d.

Proof of Lemma 4.9. Define $n_i = n/m^i$. For simplicity, we ignore rounding issues, i. e., we assume that n_i is an integer and that $m = 2^{(\log n)^{1-\alpha}}$ exactly. Let $\delta = \frac{\epsilon}{8w^2 n}$, and let $d = O(\log(wn/\epsilon))$ be the length of the second input to the function $\mathsf{Samp}_{t,\delta}$ of Theorem 4.10. We will recursively define a sequence of PRGs

$$G_i \colon \{0,1\}^t \times (\{0,1\}^s)^i \times (\{0,1\}^d)^{n_i} \to (\{0,1\}^t)^n .$$

The base case $i = 0$ is the basic PRG of Lemma 4.11:

$$G_0(x, z_1 \circ \cdots \circ z_n) = \mathsf{Samp}_{t,\delta}(x, z_1) \circ \cdots \circ \mathsf{Samp}_{t,\delta}(x, z_n) .$$

For the inductive step $i > 0$, we define

$$\begin{aligned} &G_i(x, y_1 \circ \cdots \circ y_i, z_1 \circ \cdots \circ z_{n_i}) \\ &= G_{i-1}(x, y_1 \circ \cdots \circ y_{i-1}, G(\mathsf{Samp}_{s,\delta}(y_i, z_1)) \circ \cdots \circ G(\mathsf{Samp}_{s,\delta}(y_i, z_{n_i}))) , \end{aligned}$$

where $G \colon \{0,1\}^s \to \{0,1\}^{dm}$ is the given PRG. To analyze these generators, let f be a width-w length-n ROBP over the alphabet $\{0,1\}^t$. For each i and each fixing of $x, y_1, \ldots, y_i$, define

$$\begin{aligned} g^{(x,y_1,\ldots,y_i)}(z_1 \circ \cdots \circ z_{n_i}) &= f(G_i(x, y_1 \circ \cdots \circ y_i, z_1 \circ \cdots \circ z_{n_i})) \\ h^{(x,y_1,\ldots,y_i)}(y'_1 \circ \cdots \circ y'_{n_{i+1}}) &= f(G_i(x, y_1 \circ \cdots \circ y_i, G(y'_1) \circ \cdots \circ G(y'_{n_{i+1}})) . \end{aligned}$$

These functions are related to one another by the rules

$$h^{(x,y_1,\ldots,y_i)}(y'_1 \circ \cdots \circ y'_{n_{i+1}}) = g^{(x,y_1,\ldots,y_i)}(G(y'_1) \circ \cdots \circ G(y'_{n_{i+1}})) \tag{4.2}$$

$$g^{(x,y_1,\ldots,y_i)}(z_1 \circ \cdots \circ z_{n_i}) = h^{x,y_1,\ldots,y_{i-1}}(\mathsf{Samp}_{s,\delta}(y_i, z_1) \circ \cdots \circ \mathsf{Samp}_{s,\delta}(y_i, z_{n_i})) . \tag{4.3}$$

This shows by induction on i that each g function can be computed by a width-w ROBP over the alphabet $\{0,1\}^d$ and each h function can be computed by a width-w ROBP over the alphabet $\{0,1\}^s$.

Let us now show by induction on i that G_i fools f with error $(2\delta w^2 + \epsilon_G) \cdot \sum_{j=0}^{i} n_j$, where ϵ_G is the error of G. The base case $i = 0$ is already established by Lemma 4.11. For the inductive step, we have

$$\begin{aligned}
&\mathop{\mathbb{E}}_{\substack{x\\ y_1,\dots,y_i\\ z_1,\dots,z_{n_i}}}[f(G_i(x, y_1 \circ \cdots \circ y_i, z_1 \circ \cdots \circ z_{n_i}))] \qquad (4.4)\\
&= \mathop{\mathbb{E}}_{\substack{x\\ y_1,\dots,y_i\\ z_1,\dots,z_{n_i}}}[g^{(x,y_1,\dots,y_i)}(z_1 \circ \cdots \circ z_{n_i})]\\
&= \mathop{\mathbb{E}}_{\substack{x\\ y_1,\dots,y_i\\ z_1,\dots,z_{n_i}}}[h^{(x,y_1,\dots,y_{i-1})}(\mathsf{Samp}_{s,\delta}(y_i, z_1) \circ \cdots \circ \mathsf{Samp}_{s,\delta}(y_i, z_{n_i}))] && \text{(Equation (4.3))}\\
&\le \mathop{\mathbb{E}}_{\substack{x\\ y_1,\dots,y_{i-1}\\ y'_1,\dots,y'_{n_i}}}[h^{(x,y_1,\dots,y_{i-1})}(y'_1 \circ \cdots \circ y'_{n_i})] + 2\delta w^2 n_i && \text{(Lemma 4.11)}\\
&= \mathop{\mathbb{E}}_{\substack{x\\ y_1,\dots,y_{i-1}\\ y'_1,\dots,y'_{n_i}}}[g^{(x,y_1,\dots,y_{i-1})}(G(y'_1) \circ \cdots \circ G(y'_{n_i}))] + 2\delta w^2 n_i && \text{(Equation (4.2))}\\
&\le \mathop{\mathbb{E}}_{\substack{x\\ y_1,\dots,y_{i-1}\\ z_1,\dots,z_{n_{i-1}}}}[g^{(x,y_1,\dots,y_{i-1})}(z_1 \circ \cdots \circ z_{n_{i-1}})] + (2\delta w^2 + \epsilon_G) \cdot n_i\\
&= \mathop{\mathbb{E}}_{\substack{x\\ y_1,\dots,y_{i-1}\\ z_1,\dots,z_{n_{i-1}}}}[f(G_{i-1}(x, y_1 \circ \cdots \circ y_{i-1}, z_1 \circ \cdots \circ z_{n_{i-1}}))] + (2\delta w^2 + \epsilon_G) \cdot n_i\,.
\end{aligned}$$

The lower bound on the quantity in Equation (4.4) follows the same argument. Let $r = \log^{\alpha} n$ and $G' = G_r$. Then G' fools f with error

$$(2\delta w^2 + \epsilon_G) \cdot \sum_{i=0}^{r} n_i \le (2\delta w^2 + \epsilon_G) \cdot n \cdot \sum_{i=0}^{\infty} m^{-i} \le 2n \cdot (2\delta w^2 + \epsilon_G) \le \epsilon\,.$$

Furthermore, the seed length of G' is $t + rs + d$ as claimed, and the space complexity of G' is clearly also $O(t + rs + d)$. □

4.4 Putting things together to prove Theorem 4.3

Proof of Theorem 4.3. We will show by induction on a that for each constant $a \in \mathbb{N}$, there is an explicit PRG family for width-w ROBPs with seed length $O(t + \log(n/\epsilon)\log^{1/a} n)$. The base case $a = 1$ holds unconditionally – this is the seed length of the classic INW generator [20].

For the inductive step, suppose $a > 1$. Let t, n, ϵ be arbitrary; we will construct an ϵ-PRG for width-w length-n ROBPs over the alphabet $\{0,1\}^t$. Define $\alpha = 1/a$. Let $m = 2^{(\log n)^{1-\alpha}}$ and $d = C\log(n/\epsilon)$, as in Lemma 4.9.

By induction, there is a $(\frac{1}{2w^3m^2})$-PRG G for width-w length-m ROBPs over the alphabet $\{0,1\}^d$, with seed length and space complexity bounded by $O(d + \log^{1+\frac{1}{a-1}} m)$. Now, by the choice of m,

$$\log^{1+\frac{1}{a-1}} m = (\log n)^{(1-\frac{1}{a})\cdot(1+\frac{1}{a-1})} = \log n\,,$$

so G has seed length and space complexity bounded by $O(\log(n/\epsilon))$. Plugging G into Theorem 4.4, we get an $(\frac{\epsilon}{4n})$-hitting set H for width-w length-m ROBPs over the alphabet $\{0,1\}^d$, computable in space $O(\log(n/\epsilon))$. Now we use our assumption to convert H into a PRG G' with exactly the same parameters. Finally, plugging G' into Lemma 4.9 gives the desired PRG. □

5 Derandomizing BPP given a hitting set

In this section, as mentioned in Section 1, we present an alternative proof for the known theorem that optimal explicit hitting sets for circuits would imply **P** = **BPP**.

Theorem 5.1 ([3])**.** *Assume that for every $s, n \in \mathbb{N}$, there is a $\frac{1}{2}$-hitting set $H_{s,n}$ for size-s circuits on n input bits that can be computed in time poly(s,n). Then* **P** = **BPP***.*

Proof. By naïve amplification, we may assume that the randomized algorithm has failure probability 2^{-N}, where N is the *input* length. Let C be a size-n circuit on n input bits describing the action of this algorithm on its random bits, so $n = \text{poly}(N)$ and we are trying to distinguish the cases $\mathbb{E}[C] \leq 2^{-N}$ vs. $\mathbb{E}[C] \geq 1 - 2^{-N}$. Our algorithm accepts if and only if there exists $x \in H_{n^c,n}$ such that for all $y \in H_{3n,n}$, $C(x \oplus y) = 1$. Here, c is a suitable constant that will become clear later. The runtime is clearly poly(N).

For the correctness proof, first suppose $\mathbb{E}[C] \leq 2^{-N}$. For any fixed x, the function $y \mapsto \neg C(x \oplus y)$ has expectation at least $1 - 2^{-N}$ and can be computed by a circuit of size $3n$. Therefore, there is some $y \in H_{3n,n}$ such that $C(x \oplus y) = 0$, and hence our algorithm rejects. Conversely, suppose $\mathbb{E}[C] \geq 1 - 2^{-N}$. Consider sampling $x \in \{0,1\}^n$ and $y \in H_{3n,n}$ uniformly at random. Since x is uniform, $\mathbb{E}_{x,y}[\neg C(x \oplus y)] \leq 2^{-N}$. By Markov's inequality,

$$\Pr_{x \in \{0,1\}^n}\left[\mathop{\mathbb{E}}_{y \in H_{3n,n}}[\neg C(x \oplus y)] < 2 \cdot 2^{-N}\right] > 1/2\,.$$

Since $H_{3n,n}$ can be computed in polynomial time, $|H_{3n,n}| \leq \text{poly}(N)$. Therefore, when N is sufficiently large,

$$\mathop{\mathbb{E}}_{y \in H_{3n,n}}[\neg C(x \oplus y)] < 2 \cdot 2^{-N} \implies \mathop{\mathbb{E}}_{y \in H_{3n,n}}[\neg C(x \oplus y)] = 0\,.$$

Therefore,

$$\Pr_{x \in \{0,1\}^n}[\forall y \in H_{3n,n}, C(x \oplus y) = 1] > 1/2\,.$$

Given input x, the predicate $\forall y \in H_{3n,n}, C(x \oplus y) = 1$ can be computed by a circuit C' of size n^c for some suitable constant c. Therefore, there is some $x \in H_{n^c,n}$ such that $C'(x) = 1$. □

6 Directions for further research

In this paper, we have shown that hitting sets for **RL** would derandomize **BPL**. Constructing a hitting set is the most natural way to prove **L** = **RL**, but there are also other approaches. In general, does **L** = **RL** imply **L** = **BPL**? In the polynomial-time setting, the "promise" variant of this question has been answered in the affirmative, i. e., **prP** = **prRP** $\implies$ **P** = **BPP** [9]. Does **prL** = **prRL** imply **L** = **BPL**? Or relaxing the challenge even further, does **L** = **NL** imply **L** = **BPL**?

We gave two different algorithms for estimating the expectation of an ROBP given a hitting set, one suited for $w = \mathrm{poly}(n)$ (Theorem 2.1) and one suited for $w = O(1)$ (Theorem 3.1). What about the case $n = \mathrm{polylog}(w)$, i. e., $w = 2^{n^{\Theta(1)}}$? Unconditionally, there are optimal hitting sets known in this regime [2, 19]. Given such an ROBP f as input, is it possible to compute $\mathbb{E}[f] \pm \frac{1}{w}$ in space $O(\log w)$? (The Nisan-Zuckerman PRG [27] achieves seed length $O(\log w)$ in this regime, but only for moderate error $\epsilon \gg \frac{1}{w}$.) An affirmative answer would imply that any space-s decision algorithm that uses n random bits could be simulated by another space-$O(s)$ algorithm using only $O(n/s^c)$ random bits, where c is an arbitrarily large constant.

Recently, Meka, Reingold, and Tal constructed a PRG for width-3 ROBPs with seed length $\widetilde{O}(\log n \log(1/\epsilon))$ [24]. This is near-optimal when ϵ is not too small, but for $\epsilon = 1/n$ it is worse than Nisan's PRG [25]. On the other hand, there is an explicit hitting set for width-3 ROBPs with near-optimal seed length $\widetilde{O}(\log(n/\epsilon))$ [16]. Can one construct an explicit deterministic sampler for width-3 ROBPs with near-optimal space complexity? Unfortunately, to produce a deterministic sampler for width-3 ROBPs, Theorem 3.1 would require a hitting set for width-4 ROBPs.

Assuming the existence of a log-space hitting set for polynomial-width ROBPs, is it possible to construct a log-space deterministic sampler for polynomial-width ROBPs?

Recall that WPRGs are superior to deterministic samplers (see Figure 1). Is it possible to improve Theorem 1.7 so that it concludes with a WPRG rather than a mere deterministic sampler?

7 Acknowledgments

We thank David Zuckerman, Dean Doron, and Pooya Hatami for helpful discussions and for comments on drafts of this paper. We thank Adam Klivans for bringing his work with Gopalan and Meka [15] to our attention. We thank anonymous referees for helpful comments.

References

[1] AMIRMAHDI AHMADINEJAD, JONATHAN KELNER, JACK MURTAGH, JOHN PEEBLES, AARON SIDFORD, AND SALIL VADHAN: High-precision estimation of random walks in small space. In *Proc. 61st FOCS*, pp. 1295–1306. IEEE Comp. Soc., 2020. [doi:10.1109/FOCS46700.2020.00123, arXiv:1912.04524] 10, 11

[2] MIKLÓS AJTAI, JÁNOS KOMLÓS, AND ENDRE SZEMERÉDI: Deterministic simulation in LOGSPACE. In *Proc. 19th STOC*, pp. 132–140. ACM Press, 1987. [doi:10.1145/28395.28410] 28

[3] ALEXANDER E. ANDREEV, ANDREA E. F. CLEMENTI, AND JOSÉ D. P. ROLIM: A new general derandomization method. *J. ACM*, 45(1):179–213, 1998. Preliminary version in ICALP'96. [doi:10.1145/273865.273933] 2, 5, 10, 27

[4] ALEXANDER E. ANDREEV, ANDREA E. F. CLEMENTI, JOSÉ D. P. ROLIM, AND LUCA TREVISAN: Weak random sources, hitting sets, and BPP simulations. *SIAM J. Comput.*, 28(6):2103–2116, 1999. Preliminary version in FOCS'97. [doi:10.1137/S0097539797325636, ECCC:TR97-011] 2, 5, 10

[5] ROY ARMONI: On the derandomization of space-bounded computations. In *Proc. 2nd Internat. Workshop on Randomization and Computation (RANDOM'98)*, pp. 47–59. Springer, 1998. [doi:10.1007/3-540-49543-6_5] 24

[6] ANDREJ BOGDANOV: Pseudorandom generators for low degree polynomials. In *Proc. 37th STOC*, pp. 21–30. ACM Press, 2005. [doi:10.1145/1060590.1060594] 6

[7] ANDREJ BOGDANOV, ZEEV DVIR, ELAD VERBIN, AND AMIR YEHUDAYOFF: Pseudorandomness for width-2 branching programs. *Theory of Computing*, 9(7):283–292, 2013. [doi:10.4086/toc.2013.v009a007, ECCC:TR09-070] 4

[8] MARK BRAVERMAN, GIL COHEN, AND SUMEGHA GARG: Pseudorandom pseudo-distributions with near-optimal error for read-once branching programs. *SIAM J. Comput.*, 49(5):242–299, 2020. Preliminary version in STOC'18. [doi:10.1137/18M1197734, ECCC:TR17-161] 3, 4, 6, 10, 24

[9] HARRY BUHRMAN AND LANCE FORTNOW: One-sided versus two-sided error in probabilistic computation. In *Proc. 16th Symp. Theoret. Aspects of Comp. Sci. (STACS'99)*, pp. 100–109. Springer, 1999. [doi:10.1007/3-540-49116-3_9] 2, 5, 10, 28

[10] ESHAN CHATTOPADHYAY AND JYUN-JIE LIAO: Optimal error pseudodistributions for read-once branching programs. In *Proc. 35th Comput. Complexity Conf. (CCC'20)*, pp. 25:1–27. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2020. [doi:10.4230/LIPIcs.CCC.2020.25, arXiv:2002.07208, ECCC:TR20-069] 3, 4, 6, 10

[11] KUAN CHENG AND WILLIAM M. HOZA: Hitting sets give two-sided derandomization of small space. In *Proc. 35th Comput. Complexity Conf. (CCC'20)*, pp. 10:1–25. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2020. [doi:10.4230/LIPIcs.CCC.2020.10, ECCC:TR20-016] 1, 4

[12] GIL COHEN, DEAN DORON, OREN RENARD, ORI SBERLO, AND AMNON TA-SHMA: Error reduction for weighted PRGs against read once branching programs. In *Proc. 36th Comput. Complexity Conf. (CCC'21)*, pp. 22:1–17. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2021. [doi:10.4230/LIPIcs.CCC.2021.22, ECCC:TR21-020] 4, 6, 10

[13] ODED GOLDREICH, SALIL VADHAN, AND AVI WIGDERSON: Simplified derandomization of BPP using a hitting set generator. In ODED GOLDREICH, editor, *Studies in Complexity and Cryptography*,

pp. 59–67. Springer, 2011. Preliminary version in RANDOM'99. [doi:10.1007/978-3-642-22670-0_8, ECCC:TR00-004] 2, 5, 10

[14] ODED GOLDREICH AND AVI WIGDERSON: Tiny families of functions with random properties: a quality-size trade-off for hashing. *Random Struct. Algor.*, 11(4):315–343, 1997. Preliminary version in STOC'94. [doi:10.1002/(SICI)1098-2418(199712)11:4<315::AID-RSA3>3.3.CO;2-D, ECCC:TR94-002] 9, 24

[15] PARIKSHIT GOPALAN, ADAM R. KLIVANS, AND RAGHU MEKA: Learning functions of halfspaces using prefix covers. In *Proc. 25th Ann. Conf. on Learning Theory (COLT'12)*, pp. 15.1–15.10. Springer, 2012. PMLR. 8, 28

[16] PARIKSHIT GOPALAN, RAGHU MEKA, OMER REINGOLD, LUCA TREVISAN, AND SALIL VADHAN: Better pseudorandom generators from milder pseudorandom restrictions. In *Proc. 53rd FOCS*, pp. 120–129. IEEE Comp. Soc., 2012. [doi:10.1109/FOCS.2012.77, arXiv:1210.0049, ECCC:TR12-123] 4, 28

[17] WILLIAM M. HOZA: Better pseudodistributions and derandomization for space-bounded computation. In *Proc. 25th Internat. Conf. on Randomization and Computation (RANDOM'21)*, pp. 28:1–23. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2021. [doi:10.4230/LIPIcs.APPROX/RANDOM.2021.28, ECCC:TR21-048] 4, 6, 10

[18] WILLIAM M. HOZA AND CHRIS UMANS: Targeted pseudorandom generators, simulation advice generators, and derandomizing logspace. *SIAM J. Comput.*, 51(2):281–304, 2022. Preliminary version in STOC'17. [doi:10.1137/17M1145707, arXiv:1610.01199] 6, 10

[19] WILLIAM M. HOZA AND DAVID ZUCKERMAN: Simple optimal hitting sets for small-success **RL**. *SIAM J. Comput.*, 49(4):811–820, 2020. Preliminary version in FOCS'18. [doi:10.1137/19M1268707, ECCC:TR18-063] 3, 4, 6, 9, 10, 19, 20, 21, 28

[20] RUSSELL IMPAGLIAZZO, NOAM NISAN, AND AVI WIGDERSON: Pseudorandomness for network algorithms. In *Proc. 26th STOC*, pp. 356–364. ACM Press, 1994. [doi:10.1145/195058.195190] 9, 23, 24, 26

[21] RUSSELL IMPAGLIAZZO AND AVI WIGDERSON: P = BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *Proc. 29th STOC*, pp. 220–229. ACM Press, 1997. [doi:10.1145/258533.258590] 6

[22] RICHARD E. LADNER AND NANCY A. LYNCH: Relativization of questions about log space computability. *Math. Sys. Theory*, 10(1):19–32, 1976. [doi:10.1007/BF01683260] 15

[23] NATHAN LINIAL, MICHAEL LUBY, MICHAEL SAKS, AND DAVID ZUCKERMAN: Efficient construction of a small hitting set for combinatorial rectangles in high dimension. *Combinatorica*, 17(2):215–234, 1997. Preliminary version in STOC'93. [doi:10.1007/BF01200907] 21, 23

[24] RAGHU MEKA, OMER REINGOLD, AND AVISHAY TAL: Pseudorandom generators for width-3 branching programs. In *Proc. 51st STOC*, pp. 626–637. ACM Press, 2019. [doi:10.1145/3313276.3316319, arXiv:1806.04256, ECCC:TR18-112] 4, 28

[25] NOAM NISAN: Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992. Preliminary version in STOC'90. [doi:10.1007/BF01305237] 3, 4, 6, 9, 10, 24, 28

[26] NOAM NISAN: On read-once vs. multiple access to randomness in logspace. *Theoret. Comput. Sci.*, 107(1):135–144, 1993. Preliminary version in SCT'90. [doi:10.1016/0304-3975(93)90258-U] 10

[27] NOAM NISAN AND DAVID ZUCKERMAN: Randomness is linear in space. *J. Comput. System Sci.*, 52(1):43–52, 1996. Preliminary version in STOC'93. [doi:10.1006/jcss.1996.0004] 24, 28

[28] EDWARD PYNE AND SALIL VADHAN: Pseudodistributions that beat all pseudorandom generators (extended abstract). In *Proc. 36th Comput. Complexity Conf. (CCC'21)*, pp. 33:1–15. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2021. [doi:10.4230/LIPIcs.CCC.2021.33, ECCC:TR21-019] 4, 6, 10

[29] JIŘÍ ŠÍMA AND STANISLAV ŽÁK: A polynomial-time construction of a hitting set for read-once branching programs of width 3. *Fundamenta Informaticae*, 184(4):307–354, 2021. Preliminary versions in CSR'11 and SOFSEM'12. [doi:10.3233/fi-2021-2101, arXiv:2101.01151, ECCC:TR10-088] 4

AUTHORS

Kuan Cheng
Assistant professor
Center on Frontiers of Computing Studies
Peking University
Beijing, China
ckkcdh@hotmail.com
https://sites.google.com/site/ckkcdh/

William M. Hoza
Postdoctoral fellow
Simons Institute for the Theory of Computing
University of California, Berkeley
Berkeley, CA, USA
williamhoza@berkeley.edu
https://williamhoza.com/tcs/

ABOUT THE AUTHORS

KUAN CHENG is an Assistant Professor in the Center on Frontiers of Computing Studies at Peking University. He got his Ph. D. from Johns Hopkins University in 2019, advised by Xin Li. After that, he was a postdoc hosted by David Zuckerman, at the University of Texas at Austin. He truly had a wonderful time doing research in these two places. Many new ideas, experiences, and happy collaborations were achieved during these years. Kuan has a broad interest in theoretical computer science, including randomness in computation, coding theory, highly efficient algorithms, learning theory, etc. In his spare time, he likes playing soccer, basketball, and skiing. Sometimes he can even play these by himself for a long time. Besides, he likes driving randomly to beautiful places around his living place.

WILLIAM M. HOZA is a postdoctoral fellow in Avishay Tal's group at the University of California, Berkeley through the Simons Institute for the Theory of Computing. He received his B. S. from the California Institute of Technology in 2016, where he received valuable mentorship from Leonard Schulman and Chris Umans. He received his Ph. D. from the University of Texas at Austin in 2021, where he was advised by David Zuckerman. He is especially interested in pseudorandomness and derandomization. As stated on his personal website, he likes pineapple on pizza, and he likes the Star Wars prequel trilogy, but he does not like chocolate.