Implications of Better PRGs for Permutation Branching Programs

Dean Doron
Department of Computer Science, Ben-Gurion University of the Negev, Beer-Sheva, Israel

William M. Hoza
Department of Computer Science, The University of Chicago, IL, USA

Abstract

We study the challenge of derandomizing constant-width standard-order read-once branching programs (ROBPs). Let $c \in [1, 2)$ be any constant. We prove that if there are explicit pseudorandom generators (PRGs) for width-6 length-n permutation ROBPs with error $1/n$ and seed length $\widetilde{O}(\log^c n)$, then there are explicit hitting set generators (HSGs) for width-4 length-n ROBPs with threshold $1/\operatorname{polylog}(n)$ and seed length $\widetilde{O}(\log^c n)$.

For context, there are known explicit PRGs that fool constant-width permutation ROBPs with error ε and seed length $O(\log n \cdot \log(1/\varepsilon))$ (Koucký, Nimbhorkar, and Pudlák STOC 2011; De CCC 2011; Steinke ECCC 2012). When $\varepsilon = 1/n$, there are known constructions of *weighted* pseudorandom generators (WPRGs) that fool polynomial-width permutation ROBPs with seed length $\widetilde{O}(\log^{3/2} n)$ (Pyne and Vadhan CCC 2021; Chen, Hoza, Lyu, Tal, and Wu FOCS 2023; Chattopadhyay and Liao ITCS 2024), but *unweighted* PRGs with seed length $o(\log^2 n)$ remain elusive. Meanwhile, for width-4 ROBPs, there are no known explicit PRGs, WPRGs, or HSGs with seed length $o(\log^2 n)$.

Our reduction can be divided into two parts. First, we show that explicit low-error PRGs for width-6 permutation ROBPs with seed length $\widetilde{O}(\log^c n)$ would imply explicit low-error PRGs for width-3 ROBPs with seed length $\widetilde{O}(\log^c n)$. This would improve Meka, Reingold, and Tal's PRG (STOC 2019), which has seed length $o(\log^2 n)$ only when the error parameter is relatively large. Second, we show that for any w, n, s, and ε, an explicit PRG for width-w ROBPs with error $0.01/n$ and seed length s would imply an explicit ε-HSG for width-$(w+1)$ ROBPs with seed length $O(s + \log n \cdot \log(1/\varepsilon))$.

2012 ACM Subject Classification Theory of computation → Pseudorandomness and derandomization

Keywords and phrases hitting set generators, pseudorandom generators, read-once branching programs

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2025.28

Category RANDOM

Related Version *Full Version*: https://eccc.weizmann.ac.il/report/2024/137/

Funding *William M. Hoza*: Part 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.

Acknowledgements We thank Omer Reingold and Avishay Tal for valuable discussions.

1 Introduction

One of the classic goals of computational complexity theory is to clarify the relationship between two fundamental computational resources: randomness and space complexity. The "$\mathsf{L} = \mathsf{BPL}$" conjecture asserts that if a language can be decided by a randomized algorithm that uses S bits of space and always halts, where $S \geq \log n$, then it can also be decided by a deterministic algorithm that uses $O(S)$ bits of space. A particularly satisfying way to prove $\mathsf{L} = \mathsf{BPL}$ would be to design a suitable *pseudorandom generator* (PRG), defined next.

© Dean Doron and William M. Hoza;
licensed under Creative Commons License CC-BY 4.0

▶ **Definition 1.1** (PRGs). *Let $\mathcal{F}$ be a class of functions $f\colon \{0,1\}^n \to \mathbb{R}$, let $\varepsilon > 0$, and let X be a distribution over $\{0,1\}^n$. We say that X* ε-fools *$\mathcal{F}$, or fools $\mathcal{F}$ with error ε, if*

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

Here $\mathbb{E}[f]$ is shorthand for $\mathbb{E}[f(U_n)]$, where U_n denotes the uniform distribution over $\{0,1\}^n$. An ε-PRG for $\mathcal{F}$*, or a PRG that fools $\mathcal{F}$ with error ε, is a function $G\colon \{0,1\}^s \to \{0,1\}^n$ such that $G(U_s)$ fools $\mathcal{F}$ with error ε. The parameter s is called the* seed length *of the PRG.*

To simulate a randomized algorithm, we would like a PRG that fools some model $\mathcal{F}$ that captures the behavior of the algorithm on a fixed input as a function of its random bits. In the case of space-bounded algorithms, we can take $\mathcal{F}$ to be the class of functions computable by polynomial-width standard-order read-once branching programs (ROBPs), defined below.

▶ **Definition 1.2** (ROBPs). *Let $w, n \in \mathbb{N}$. A* width-w length-n standard-order ROBP[1] *is a directed graph in which the vertices are arranged in layers $V_0, V_1, \ldots, V_n$ satisfying $|V_i| \leq w$ for every i. For every $i \in \{0, 1, \ldots, n-1\}$ and every $v \in V_i$, there are two outgoing edges leading from v to V_{i+1}, labeled 0 and 1 (the "0-edge" and the "1-edge" respectively). There is a designated "start vertex" $v_0 \in V_0$. Each input $x \in \{0,1\}^n$ defines a walk through the program: we start at v_0, and in step $i \in [n]$, we traverse the outgoing x_i-edge of our current vertex. In this manner, we arrive at a final vertex $v_n \in V_n$. Each vertex in V_n is labeled either "accept" or "reject," and the program accepts or rejects x based on the label of v_n. In this way, the program defines a function $f\colon \{0,1\}^n \to \{0,1\}$.*

For each fixed input, the output of a space-S algorithm can be computed by a width-w length-n standard-order ROBP that reads the n random bits used by the algorithm, where w and n are both bounded by $2^{O(S)}$. Consequently, if we could design an explicit[2] PRG for width-w length-n standard-order ROBPs with the optimal seed length $O(\log(wn/\varepsilon))$, then we could derandomize BPL by deterministically simulating the randomized algorithm on all possible outputs of the generator.

Decades ago, Nisan designed an explicit PRG for width-w length-n standard-order ROBPs with non-optimal seed length $O(\log(wn/\varepsilon) \cdot \log n)$ [28]. To this day, Nisan's PRG is the best PRG known for polynomial-width standard-order ROBPs.

1.1 Regular and Permutation Branching Programs

Because it has turned out to be extremely difficult to design better PRGs for polynomial-width standard-order ROBPs, researchers have investigated ROBPs with additional structural restrictions. Two of the most well-studied classes are *regular* ROBPs and *permutation* ROBPs, defined next.

▶ **Definition 1.3** (Regular and permutation ROBPs). *An ROBP with layers $V_0, V_1, \ldots, V_n$ is* regular *if every vertex $v \in V_1 \cup \cdots \cup V_n$ has precisely two incoming edges. If these two incoming edges have distinct labels (0 and 1) for each v, then we say that the program is a* permutation ROBP.

[1] The qualifier "standard-order" is often omitted; it emphasizes the fact that the program reads the input bits from left to right.

[2] I.e., the space complexity of the generator should be proportional to its seed length.

Braverman, Rao, Raz, and Yehudayoff showed that there is an explicit PRG that ε-fools width-w length-n standard-order *regular* ROBPs with seed length $\widetilde{O}(\log(w/\varepsilon) \cdot \log n)$ [5]. This beats Nisan's seed length [28] provided that $w \ll n$ and $\varepsilon \gg 1/n$. Their PRG construction consists of an appropriately-parameterized version of the so-called Impagliazzo-Nisan-Wigderson (INW) generator [21]. For constant-width standard-order *permutation* ROBPs, the seed length can be improved to $O(\log n \cdot \log(1/\varepsilon))$ [6, 22, 13, 33]. However, we emphasize that when $\varepsilon = 1/n$, there are still no known explicit PRGs with seed length $o(\log^2 n)$, even for the case of constant-width standard-order permutation ROBPs.

1.2 Width-2, Width-3, and Width-4 Branching Programs

There has also been some success at beating Nisan's PRG in the case of extremely narrow ROBPs, with no structural restrictions. In the 1990s, Saks and Zuckerman showed that any "small-bias distribution" [27] fools width-2 branching programs, and consequently, there is an explicit PRG that fools width-2 programs with the optimal seed length $O(\log(n/\varepsilon))$ [32, 2, 16, 23].

Width-3 programs turned out to be much more challenging. Nevertheless, Meka, Reingold, and Tal managed to construct an explicit PRG for width-3 standard-order ROBPs with seed length $\widetilde{O}(\log n \cdot \log(1/\varepsilon))$ [25]. Meka, Reingold, and Tal's PRG [25] follows the "iterated restrictions with early termination" paradigm. First, they show how to assign pseudorandom values to a pseudorandom subset of a width-3 ROBP's input variables in such a way that the expectation of the program is approximately preserved. Second, they show that after several such rounds of pseudorandom restrictions, the program *simplifies* with high probability, in the sense that it "almost" becomes a permutation ROBP. Finally, they show that the INW generator fools this "near-permutation program" with a good seed length.

For width-4 standard-order ROBPs, no explicit PRGs are known with seed length $o(\log^2 n)$. The width-4 case brings new challenges that are not present in the width-2 and width-3 cases. For example, width-4 ROBPs can compute read-once "$\mathsf{AND} \circ \mathsf{OR} \circ \mathsf{PARITY}$" formulas, which have been highlighted as a challenging case for the "iterated restrictions with early termination" paradigm [17]. These formulas are provably not "simple" enough to be fooled by the INW generator with seed length $o(\log^2 n)$ [6, 19], and they apparently do not get any "simpler" under restrictions, due to the layer of parity gates at the bottom. That being said, one can use small-bias distributions to ε-fool read-once $\mathsf{AND} \circ \mathsf{OR} \circ \mathsf{PARITY}$ formulas with a seed length of $O(\log n \cdot \log(1/\varepsilon))$.[3] This perhaps suggests that we can be optimistic about fooling all width-4 ROBPs with a similar seed length.

1.3 Hitting Set Generators

Another approach for making progress on the L vs. BPL problem is to look at relaxations of the PRG concept. The most famous such relaxation is the classic *hitting set generator* (HSG) concept, defined next.

▶ **Definition 1.4** (HSGs). *Let $\mathcal{F}$ be a class of functions $f\colon \{0,1\}^n \to \{0,1\}$ and let $\varepsilon > 0$. An* ε-hitting set for $\mathcal{F}$ *is a set $H \subseteq \{0,1\}^n$ such that for every $f \in \mathcal{F}$, if $\mathbb{E}[f] > \varepsilon$, then there is some $x \in H$ such that $f(x) = 1$. An* ε-HSG for $\mathcal{F}$ *is a function $G\colon \{0,1\}^s \to \{0,1\}^n$ such that the image of G is an ε-hitting set for $\mathcal{F}$.*

[3] After applying the parity gates, the resulting distribution still has small bias, and every δ-biased distribution fools read-once CNFs with error $\exp(-\Omega(\log(1/\delta)/\log n))$ [7, 14].

Every ε-PRG for a class $\mathcal{F}$ is also an ε-HSG, but not vice versa. By working through the definitions, one can easily show that explicit HSGs for polynomial-width ROBPs with seed length $O(\log n)$ would imply $\mathsf{L} = \mathsf{RL}$, where RL is the variant of BPL in which we only allow algorithms with one-sided error. In fact, it turns out that such HSGs would imply $\mathsf{L} = \mathsf{BPL}$ [11, 29]. Constructing HSGs is thus a route to proving $\mathsf{L} = \mathsf{BPL}$ that is potentially easier than constructing PRGs. However, despite the extra flexibility of the HSG definition, we would like to highlight the fact that there are currently no known explicit HSGs for width-4 standard-order ROBPs with seed length $o(\log^2 n)$.

1.4 Our Contributions

In this work, we establish new connections between the problems discussed above. Our main result says that if we could construct improved explicit PRGs for constant-width standard-order permutation ROBPs in the low-error regime, then we would get improved explicit HSGs for width-4 standard-order ROBPs.

▶ **Theorem 1.5** (PRGs for width-6 permutation programs $\implies$ HSGs for width-4 programs)**.** *Let $c \in [1, 2)$ be any constant. Assume that for every $n \in \mathbb{N}$, there exists an explicit $(1/n)$-PRG for width-6 length-n standard-order permutation ROBPs with seed length $\widetilde{O}(\log^c n)$. Then for every $n \in \mathbb{N}$ and every $\varepsilon \in (0, 1)$, there exists an explicit ε-HSG for width-4 length-n standard-order ROBPs with seed length $\widetilde{O}(\log^c n + \log n \cdot \log(1/\varepsilon))$.*

For context, prior work has shown that improved PRGs for *regular* ROBPs of *polynomial* width would have huge implications [31, 3, 24]. Indeed, Lee, Pyne, and Vadhan showed that every function computable by a width-w length-n standard-order ROBP can also be computed by a standard-order regular ROBP of width $O(wn)$ [24]. Our new reduction shows that improved PRGs for standard-order *permutation* ROBPs of *constant* width (a much weaker model) would already have significant consequences.

Our reduction can be divided into two parts, each of which is interesting in its own right. First, we revisit Meka, Reingold, and Tal's reduction from the problem of fooling width-3 ROBPs to the problem of fooling "near-permutation" programs [25]. We show a reduction to the problem of fooling programs that *exactly* satisfy the permutation condition, albeit of width 6 instead of 3:

▶ **Theorem 1.6** (Fooling width-6 permutation programs $\implies$ fooling width-3 programs)**.** *There exists a constant $C > 0$ such that the following holds. Assume that for every $n \in \mathbb{N}$ and $\delta \in (0, 1)$, there exists an explicit δ-PRG for width-6 length-n standard-order permutation ROBPs with seed length $s(n, \delta)$. Then, for every $n \in \mathbb{N}$ and $\varepsilon \in (0, \frac{1}{\log \log n})$, there exists an explicit ε-PRG for width-3 length-n standard-order ROBPs with seed length $s(n, \delta) + \widetilde{O}(\log(n/\varepsilon))$, where $\delta = \varepsilon^{C \cdot \log \log(n/\varepsilon)}$.*

Given Theorem 1.6, we can recover Meka, Reingold, and Tal's $\widetilde{O}(\log n \cdot \log(1/\varepsilon))$ seed length for fooling width-3 standard-order ROBPs [25] (up to log log factors) by plugging in the state-of-the-art seed length $s(n, \delta) = O(\log n \cdot \log(1/\delta))$ for fooling constant-width standard-order permutation ROBPs [22, 13, 33]. More importantly, Theorem 1.6 shows that hypothetical improved PRGs for constant-width permutation ROBPs would translate to improved PRGs for width-3 ROBPs.

Second, we show how to convert low-error PRGs for width-w ROBPs into HSGs for width-$(w + 1)$ ROBPs, for any w:

▶ **Theorem 1.7** (PRGs for width-w programs $\implies$ HSGs for width-$(w+1)$ programs)**.** *Let $\alpha > 0$ be a constant such that $e^{\alpha-1} + \alpha < 1/2$,[4] and let $w \in \mathbb{N}$ be any constant. Assume that for every $n \in \mathbb{N}$, there exists an explicit PRG for width-w standard-order ROBPs with error α/n and seed length $s(n)$. Then for every $n \in \mathbb{N}$ and every $\varepsilon > 0$, there exists an explicit ε-HSG for width-$(w+1)$ standard-order ROBPs with seed length $O(s(n) + \log n \cdot \log(1/\varepsilon))$.*

Conceivably, one could hope to strengthen Theorem 1.7 to the point that it could be iterated. By induction on width, such a strengthened version could imply explicit HSGs for all constant-width standard-order ROBPs with seed length $o(\log^2 n)$. We find this idea exciting, even if it is admittedly quite speculative.

Our main result (Theorem 1.5) readily follows from Theorems 1.6 and 1.7, as we now briefly explain.

Proof of Theorem 1.5, given Theorems 1.6 and 1.7. If we truncate a δ-PRG for width-6 standard-order permutation ROBPs, then we get another δ-PRG for width-6 standard-order permutation ROBPs (of shorter length). Therefore, the assumption of Theorem 1.5 implies the assumption of Theorem 1.6 with $s(n, \delta) = \widetilde{O}(\log^c(n/\delta))$. Therefore, by Theorem 1.6, for every $n \in \mathbb{N}$, there exists an explicit PRG that fools width-3 standard-order ROBPs with error $0.01/n$ and seed length $s'(n) = \widetilde{O}(\log^c(n/\delta))$, where $\delta = (0.01/n)^{C \cdot \log\log(n^2/0.01)}$. Simplifying, we have $s'(n) = \widetilde{O}(\log^c n)$. Applying Theorem 1.7 completes the proof. ◀

1.5 Context: Weighted Pseudorandom Generators

Theorems 1.5–1.7 are especially interesting in light of a recent line of work on *weighted pseudorandom generators* (WPRGs), defined next.

▶ **Definition 1.8** (WPRGs)**.** *Let $\mathcal{F}$ be a class of functions $f\colon \{0,1\}^n \to \mathbb{R}$ and let $\varepsilon > 0$. An ε-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*

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

Weighted PRGs were introduced by Braverman, Cohen, and Garg [4]. The key innovation in Definition 1.8 is that the weights $\rho(x)$ can be negative. One can show that WPRGs are "intermediate" between PRGs and HSGs.

The WPRG paradigm has turned out to be especially helpful in low-error regimes. For example, recent work has shown how to construct explicit WPRGs that fool the following models with error $1/n$ and seed length $\widetilde{O}(\log^{3/2} n)$:

- Constant-width standard-order regular ROBPs [10, 8, 9, 12].
- Width-3 standard-order ROBPs [10, 8].
- "Unbounded-width" standard-order permutation ROBPs [30, 10, 8, 12].

In contrast, in all three of the cases listed above, there are no known explicit unweighted PRGs with error $1/n$ and seed length $o(\log^2 n)$.[5]

Unfortunately, our new reductions (Theorems 1.5–1.7) require unweighted PRGs. If our reductions could be adapted to work with negative weights, then we would get an explicit HSG for width-4 standard-order ROBPs with seed length $\widetilde{O}(\log^{3/2} n + \log n \cdot \log(1/\varepsilon))$.

[4] This is satisfied when $\alpha < \alpha_* \approx 0.095$.

[5] In fact, in the case of unbounded-width standard-order permutation ROBPs, unweighted PRGs with error $1/n$ and seed length $o(\log^2 n)$ provably do not exist [18].

1.6 Techniques

Our reductions can be broken down into several smaller parts; see Figure 1. One common theme in our proofs is that, following prior work [15, 3, 26], we study modified versions of the ROBP model in which it is possible to accept or reject "early," before reaching the end of the computation. We find it most convenient to work with the following models.

▶ **Definition 1.9** (∧-programs and ∨-programs). *An* ∧-program *is a standard-order ROBP in which every edge is labeled as either "accepting" or "rejecting" (in addition to the usual binary edge label). Given an input x, if every edge that the ROBP traverses is an accepting edge, then we say that the ∧-program accepts x; otherwise, we say that the ∧-program rejects x. An* ∨-program *is defined similarly, except that an ∨-program accepts if it traverses at least one accepting edge and it rejects otherwise.*

An ∧-program or an ∨-program of width w can trivially be simulated by a standard-order ROBP of width $w + 1$, but note that (a) the difference between width w and width $w + 1$ is crucial in this work, and (b) we are concerned with structural properties such as the permutation condition that are not preserved by such a trivial simulation.

1.6.1 Near-Permutation Programs vs. Exact Permutation Programs

As discussed previously, Meka, Reingold, and Tal reduced the problem of fooling width-3 ROBPs to the problem of fooling "near-permutation" programs [25]. In more detail, by "near-permutation" programs, we refer to programs that are approximated in a certain sense by programs with *few colliding layers*. (A "colliding layer" is a layer that violates the definition of a permutation ROBP.) Later, Chen, Hoza, Lyu, Tal, and Wu improved the number of colliding layers in this reduction [10].

For the proof of Theorem 1.6, our main observation is that any width-w program with t colliding layers can be written as a sum of w^t many width-w permutation ∧-programs. We prove this by a simple "guess and check" argument; see Lemma 3.3 for details.

The permutation ∧-program model is stronger than the standard-order permutation ROBP model, but previous work by Bogdanov, Hoza, Prakriya, and Pyne implies that to fool width-w permutation ∧-programs, it suffices to fool width-$(2w)$ standard-order permutation ROBPs [3]. Thus, combining their work with our observation, we see that PRGs for width-6 standard-order permutation ROBPs automatically fool width-3 programs that have a few colliding layers.

The proof of Theorem 1.6 requires some additional technical details, mainly to handle approximation errors that arise in Meka, Reingold, and Tal's framework [25]. See Section 3 for details.

1.6.2 Using a PRG for Width w to Design an HSG for Width $w + 1$

Let X be a distribution that fools width-w ROBPs with error $0.01/n$. The first step in the proof of Theorem 1.7 is to prove that $\mathrm{Supp}(X)$ is a 0.4-hitting set for width-w ∧-programs.

To explain the intuition, let us consider a special case of width-w ∧-programs, namely, a function of the form $f(x) = f_1(x) \wedge \cdots \wedge f_t(x)$, where $f_1, \ldots, f_t$ are width-w standard-order ROBPs on disjoint variables. Since we are aiming to construct a 0.4-hitting set, let us assume that $\mathbb{E}[f] > 0.4$. Letting $\delta_i = 1 - \mathbb{E}[f_i]$, we have

$$\mathbb{E}[f] = \prod_{i=1}^{t}(1 - \delta_i) \leq \prod_{i=1}^{t} e^{-\delta_i} = e^{-\sum_{i=1}^{t} \delta_i},$$

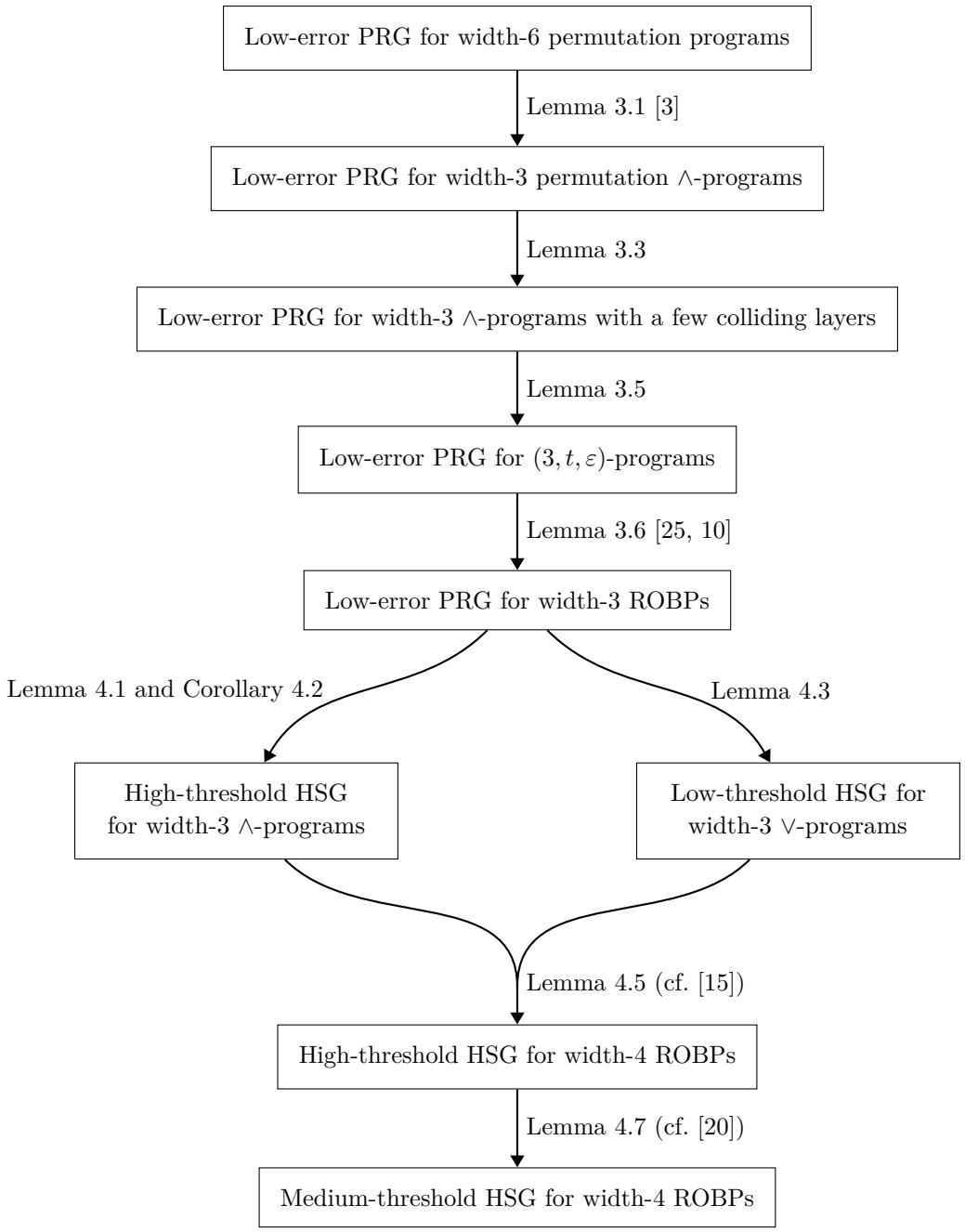

Figure 1 The reductions in this paper. Many of the reductions are slightly stronger than what the diagram suggests. For example, rather than a low-error PRG, Lemma 4.3 actually requires merely a low-threshold HSG for width-w ROBPs. Explicit constructions of such generators are known unconditionally in the $w = 3$ case [15]. On the other hand, we highlight the fact that Corollary 4.2 requires an *unweighted* PRG with *low error*. If we could eliminate either one of these requirements, then we could construct explicit HSGs for width-4 ROBPs with seed length $o(\log^2 n)$.

so $\sum_{i=1}^{t} \delta_i \leq \ln(1/\mathbb{E}[f]) < 0.92$. Therefore,

$$\begin{aligned}\Pr[f(X)=0] = \Pr[f_1(X)=0 \vee \cdots \vee f_t(X)=0] &\leq \sum_{i=1}^{t} \Pr[f_i(X)=0] \quad \text{(Union bound)} \\ &\leq \sum_{i=1}^{t} (\delta_i + 0.01/n) \\ &\leq 0.01 + \sum_{i=1}^{t} \delta_i \\ &< 0.93.\end{aligned}$$

Thus, indeed, $\mathrm{Supp}(X)$ hits f.[6]

Now suppose more generally that f is a width-w $\wedge$-program. In this case, we let δ_i be the probability that f traverses a rejecting edge in step i. By a probabilistic construction (Lemma 4.1), we show that f can be modified to ensure that $\sum_{i=1}^{n} \delta_i \leq \ln(1/\mathbb{E}[f])$. Consequently, $\mathrm{Supp}(X)$ is a 0.4-hitting set for width-w $\wedge$-programs by a similar "union bound" calculation as above. The probabilistic modification of f is arguably the most novel component of our reductions.

For the next step in the proof of Theorem 1.7, we build on Gopalan, Meka, Reingold, Trevisan, and Vadhan's techniques [15]. In our terminology, they show how to convert an explicit $(\frac{\varepsilon^2}{2n})$-HSG for width-$w$ $\wedge$-programs into an explicit ε-HSG for width-$(w+1)$ standard-order ROBPs [15, Theorem 6.5]. They used this reduction to construct a near-optimal HSG for width-3 standard-order ROBPs. We cannot apply their reduction directly, because it requires a *low-threshold* HSG and we merely have a 0.4-HSG. Therefore, we extend their reduction. We show that an explicit $\varepsilon_\wedge$-HSG for width-w $\wedge$-programs can be combined with an explicit $\varepsilon_\vee$-HSG for width-w $\vee$-programs to produce an explicit $(\varepsilon_\wedge + \varepsilon_\vee)$-HSG for width-$(w+1)$ standard-order ROBPs (Lemma 4.5). One can easily show that $\mathrm{Supp}(X)$ is a 0.01-hitting set for width-w $\vee$-programs (Lemma 4.3), so we get a 0.41-HSG for width-$(w+1)$ standard-order ROBPs.

Finally, we use an adaptation of Hoza and Zuckerman's error reduction technique [20] to construct an ε-HSG for width-$(w+1)$ standard-order ROBPs for any ε. This error reduction step blows up the seed length by a factor of $O(\log(1/\varepsilon))$, but we can get a final seed length of $O(s + \log n \cdot \log(1/\varepsilon))$ instead of $O(s \cdot \log(1/\varepsilon))$ by using the standard "sampler trick." See Section 4 for details.

1.7 Organization

After some preliminaries in Section 2, we prove Theorem 1.6 in Section 3, and then we prove Theorem 1.7 in Section 4.

2 Preliminaries

2.1 Subprograms, Substrings, Hitting Sets, and Restrictions

Let f be an ROBP with layers $V_0, V_1, \ldots, V_n$. For each $i \in \{0, 1, \ldots, n\}$ and each $u \in V_i$, we define the "subprogram" $f_{u\to}$ to be a version of f in which u is effectively the new start vertex. More precisely, $f_{u\to}$ is a length-n program on the layers $V_0', V_1', \ldots, V_i', V_{i+1}, \ldots, V_n$, where $|V_0'| = \cdots = |V_i'| = 1$. All transitions in the first i steps lead to u, and after that, the program is identical to f. We furthermore define $p_{u\to} = \mathbb{E}[f_{u\to}]$.

[6] Note that this argument breaks down if we try to replace X with a WPRG, because the union bound no longer holds.

If $x \in \{0,1\}^n$ and $1 \le i \le j \le n$, we use the notation $x_{i\dots j}$ to denote the string $x_i x_{i+1} \dots x_j$.

Let $f\colon \{0,1\}^n \to \{0,1\}$ and $H \subseteq \{0,1\}^n$. We say that H "hits" f if there is some $x \in H$ such that $f(x) = 1$.

A *restriction* is a string $\rho \in \{0,1,\star\}^n$. If $f\colon \{0,1\}^n$ and $\rho \in \{0,1,\star\}^n$, then we define the *restricted function* $f|_\rho\colon \{0,1\}^n \to \{0,1\}$ by the rule $f|_\rho(x) = f(x')$, where, for every $i \in [n]$,

$$x'_i = \begin{cases} \rho_i & \text{if } \rho_i \neq \star \\ x_i & \text{if } \rho_i = \star. \end{cases}$$

2.2 Probability

We rely on the following standard lemma from the theory of PRGs.

▶ **Lemma 2.1** (Sandwiching lemma). *Let $f_-, f, f_+\colon \{0,1\}^n \to \mathbb{R}$ and let X be a distribution over $\{0,1\}^n$. Assume that $f_-(x) \le f(x) \le f_+(x)$ for every x; assume that $\mathbb{E}[f_+ - f_-] \le \varepsilon$; and assume that X fools f_- and f_+ with error δ. Then X fools f with error $\varepsilon + \delta$.*

Proof.

$$\mathbb{E}[f(X)] \le \mathbb{E}[f_+(X)] \le \mathbb{E}[f_+] + \delta \le \mathbb{E}[f] + \varepsilon + \delta,$$

and

$$\mathbb{E}[f(X)] \ge \mathbb{E}[f_-(X)] \ge \mathbb{E}[f_-] - \delta \ge \mathbb{E}[f] - \varepsilon - \delta.$$ ◀

We also rely on the following elementary inequality.

▶ **Lemma 2.2** (Reverse union bound). *Let $E_1, E_2, \dots, E_n$ be events, and let $p = \Pr[E_1 \wedge \dots \wedge E_n]$. Then*

$$\sum_{i=1}^{n} \Pr[\neg E_i \mid E_1, E_2, \dots, E_{i-1}] \le \ln(1/p).$$

Proof. Let $\delta_i = \Pr[\neg E_i \mid E_1, E_2, \dots, E_{i-1}]$. Then

$$p = \prod_{i=1}^{n} (1 - \delta_i) \le \prod_{i=1}^{n} \exp(-\delta_i) = \exp\left(-\sum_{i=1}^{n} \delta_i\right),$$

and hence $\sum_{i=1}^{n} \delta_i \le \ln(1/p)$. ◀

3 Conditional Low-Error PRGs for Width-3 ROBPs

3.1 Fooling Programs That Have a Few Colliding Layers

The first step in the proof of Theorem 1.6 is the following lemma from the work of Bogdanov, Hoza, Prakriya, and Pyne [3, Lemma 20].

▶ **Lemma 3.1** (Fooling permutation $\wedge$-programs [3]). *Let X be a distribution over $\{0,1\}^n$. If X fools width-$(2w)$ standard-order permutation ROBPs with error ε, then X fools width-w permutation $\wedge$-programs with error 2ε.*

Technically, the model that Bogdanov, Hoza, Prakriya, and Pyne study ("unanimity programs") [3] is slightly weaker than our $\wedge$-program model, because they require the two outgoing edges of each vertex to be either both accepting or both rejecting. However, their proof applies equally well to both models.

Combining the assumption of Theorem 1.6 with Lemma 3.1 gives us a PRG for width-3 standard-order permutation $\wedge$-programs. The next step is to show that we can fool width-3 standard-order $\wedge$-programs that have a few *colliding layers*, defined next.

▶ **Definition 3.2** (Colliding layer). *Let f be an ROBP with layers $V_0, \ldots, V_n$. We say that layer i is* colliding, *where $i \in \{0, 1, \ldots, n-1\}$, if there exist two edges going from V_i to V_{i+1} with the same binary label leading to the same vertex.*

▶ **Lemma 3.3** (Fooling $\wedge$-programs with a few colliding layers). *Let f be a width-w length-n $\wedge$-program with at most t colliding layers. Then f can be written as a sum of w^t many width-w permutation $\wedge$-programs.*

Proof. Let the layers of f be $V_0, \ldots, V_n$. Let $\{i_1, i_2, \ldots, i_t\}$ be the set of colliding layers. For every $\vec{v} = (v_{i_1}, \ldots, v_{i_t}) \in V_{i_1} \times V_{i_2} \times \cdots \times V_{i_t}$, we will modify f to construct a width-w standard-order permutation $\wedge$-program $f_{\vec{v}}$. The construction is as follows.

1. Let $R = (V_{i_1} \setminus \{v_{i_1}\}) \cup \cdots \cup (V_{i_t} \setminus \{v_{i_t}\})$.
2. Redirect the outgoing edges of vertices in R in such a way that there are no remaining collisions, and re-label them as "rejecting" edges.

Observe that $f_{\vec{v}}(x) = 1$ if and only if $f(x)$ visits the vertices $v_{i_1}, \ldots, v_{i_t}$ and accepts. Consequently, $f = \sum_{\vec{v}} f_{\vec{v}}$. ◀

Next, we show how to fool width-3 standard-order ROBPs that are *well-approximated* by a suffix with few colliding layers. More precisely, we will fool $(3, t, \varepsilon)$-programs, defined below.

▶ **Definition 3.4** ((w, t, ε)-program). *Let f be a length-n standard-order ROBP with layers $V_0, V_1, \ldots, V_n$. We say that f is a* (w, t, ε)-program *if f has width at most w and there exists an $i \in \{0, 1, \ldots, n\}$ such that:*

1. *There are at most t colliding layers among layers $i, i+1, \ldots, n-1$.*
2. *If we sample $x \sim U_n$, then*

$$\Pr_x[\exists u, v \in V_i \text{ such that } f_{u\to}(x) \neq f_{v\to}(x)] \leq \varepsilon.$$

▶ **Lemma 3.5** (Fooling $\wedge$-programs with t colliding layers $\implies$ fooling (w, t, ε)-programs). *Let X be a distribution over $\{0,1\}^n$ and let $w \geq 3$. If X fools width-$\binom{w}{2}$ $\wedge$-programs that have at most t colliding layers with error δ, then X fools (w, t, ε)-programs with error $2w \cdot (\varepsilon + \delta)$.*

Proof. Let f be a (w, t, ε)-program, and let i be as in Definition 3.4. Let u_* be any element of V_i. For each $v \in V_i \setminus \{u_*\}$, define

$$h_v(x) = f_{u_*\to}(x) \oplus f_{v\to}(x).$$

We will use the following two approximators to sandwich f:

$$f_+ = f_{u_*\to} + \sum_{v \in V_i \setminus \{u_*\}} h_v$$
$$f_- = f_{u_*\to} - \sum_{v \in V_i \setminus \{u_*\}} h_v,$$

where the summation is over $\mathbb{R}$. For every $x \in \{0,1\}^n$, we have

$$f_-(x) \le f(x) \le f_+(x).$$

To see this, note that f on x either reaches u_*, or else it reaches some $v \neq u_*$, and then we can consider the h_v term. Furthermore,

$$\mathbb{E}[f_+ - f_-] = 2 \sum_{v \in V_i \setminus \{u_*\}} \mathbb{E}[h_v] \le 2w\varepsilon.$$

Thus, indeed, f is sandwiched between f_- and f_+ with error $2w\varepsilon$.

Now we will prove that X fools f_- and f_+. The program $f_{u_* \to}$ can be computed by a width-w standard-order ROBP with at most t colliding layers. The model of width-$\binom{w}{2}$ $\wedge$-programs with at most t colliding layers is only more general, so X fools $f_{u_* \to}$ with error δ. Now fix $v \in V_i \setminus \{u_*\}$; we will show that X fools h_v.

We can construct an $\wedge$-program g that computes h_v as follows.

- Each vertex of g represents an unordered pair of vertices $\{u', v'\}$ in the corresponding layer of f.
- Let $b \in \{0,1\}$ and let u'', v'' be the vertices reached from u', v' by following the outgoing b-edge. If $u'' \neq v''$, then the outgoing b-edge of $\{u', v'\}$ leads to $\{u'', v''\}$ and is an accepting edge. If $u'' = v''$, then the outgoing b-edge of $\{u', v'\}$ can lead to an arbitrary vertex and is a rejecting edge.

Observe that g has width $\binom{w}{2}$. Furthermore, every colliding layer of g is also a colliding layer of f. Therefore, g has at most t colliding layers. Consequently, X fools h_v with error δ. Therefore, X fools f_+ and f_- with error $w\delta$. Applying Lemma 2.1 completes the proof. ◀

3.2 Reducing the Number of Colliding Layers in a Width-3 Program

To complete the proof of Theorem 1.6, we will take Meka, Reingold, and Tal's approach [25]: we will apply a few pseudorandom restrictions, and we will argue that with high probability, the restricted program is "almost" a permutation program. In particular:

▶ **Lemma 3.6** (Restrictions for width-3 programs [25, 10]). *For every $n \in \mathbb{N}$ and $\varepsilon > 0$, there exists an explicit restriction generator $\mathcal{R}\colon \{0,1\}^s \to \{0,1,\star\}^n$, with $s = \widetilde{O}(\log(n/\varepsilon))$, such that if f is a width-3 standard-order ROBP and we sample $\rho \sim \mathcal{R}(U_s)$:*

1. *The restriction ρ preserves the expectation of f up to error ε. That is,*

$$\left| \mathop{\mathbb{E}}_{\rho,U}[f|_\rho(U)] - \mathbb{E}[f] \right| \le \varepsilon,$$

   where U is a uniform random string that is independent of ρ.
2. *With probability $1-\varepsilon$ over ρ, the restricted function $f|_\rho$ can be computed by a $(3,t,\varepsilon)$-program where $t = O((\log(1/\varepsilon) + \log\log\log n) \cdot \log\log(n/\varepsilon))$.*

(See Section 2 for the definition of $f|_\rho$.) In Meka, Reingold, and Tal's original analysis [25], the number of colliding layers (t) is $\mathrm{poly}(1/\varepsilon) \cdot \log\log n$ (and they use a slightly different notion of approximation). Later, Chen, Hoza, Lyu, Tal, and Wu refined their analysis [10]. Their bound on the number of colliding layers was roughly $\log^5(1/\varepsilon)$, but it turns out that merely fiddling with some parameters in their analysis is enough to prove the superior $\widetilde{O}(\log(1/\varepsilon) \cdot \log\log n)$ bound stated in Lemma 3.6. We explain some details below.

Proof sketch of Lemma 3.6. We assume that the reader is familiar with Chen, Hoza, Lyu, Tal, and Wu's analysis [10], and we merely explain what changes to make to their proofs.

- In their "Claim 7.25," we should not assume C is a constant. Their proof works for any C, provided we choose a suitable value $p = 1/\operatorname{poly}(C)$.
- In their "Claim 7.30," we should choose $C = 16\ln^{1/4}(1/\delta)$ rather than $C = 32$. After making this change, there is no longer any need to assume $\ell \geq \log\ln(1/\delta)$; the claim holds provided $\ell \geq 4$.
- In their "Lemma 7.31," we should not assume p is a constant. The lemma holds for any p that is a power of $1/2$, with a seed length of

$$s = \widetilde{O}(w \cdot \log(n/\delta) \cdot \log(1/p)).$$

- In their "Claim 7.35," instead of taking p to be a small enough constant, we should pick $p = 1/\operatorname{polylog}(1/\delta)$. In the proof, we should pick $C = 16\ln^{1/4}(1/\delta)$. We should define $\ell_j = \max\{4, \ell_{j-1}/2\}$ instead of $\ell_j = \ell_{j-1}/2$. That way, at the end, we reach $\ell = 4$, $m = C^\ell = O(\log(1/\delta))$, and $r = O(\log(1/\delta) \cdot \log\log(n/\delta))$. Applying this modified version of Claim 7.35 with $\delta = \min\{\varepsilon^4, 1/(\log\log n)^5\}$ completes the proof of Lemma 3.6. ◀

Proof of Theorem 1.6. Let G be the assumed δ-PRG. Our new PRG samples a restriction ρ using Lemma 3.6 with error $\varepsilon/16$, then samples a pseudorandom string X from G, and finally uses X to fill in the stars of ρ. The seed length is clearly $s(n,\delta) + \widetilde{O}(\log(n/\varepsilon))$.

To prove that this works, let f be a width-3 length-n standard-order ROBP. By Lemma 3.6, ρ preserves the expectation of f to within $\varepsilon/16$. Furthermore, except with probability $\varepsilon/16$, the restricted function $f|_\rho$ can be computed by a $(3, t, \varepsilon/16)$-program where $t = O((\log(1/\varepsilon) + \log\log\log n) \cdot \log\log(n/\varepsilon))$. Since we have assumed that $\varepsilon < \frac{1}{\log\log n}$, the bound simplifies to $t = O(\log(1/\varepsilon) \cdot \log\log(n/\varepsilon))$.

By Lemmas 3.1, 3.3, and 3.5, the distribution X fools width-3 standard-order permutation $\wedge$-programs with error 2δ; it fools width-3 standard-order $\wedge$-programs that have at most t colliding layers with error $2\delta \cdot 3^t$; and it fools $(3, t, \varepsilon/16)$-programs with error $6 \cdot (2\delta \cdot 3^t + \varepsilon/16)$. (Note that $\binom{3}{2} = 3$.)

Summing up, our PRG fools f with error $\varepsilon/16 + \varepsilon/16 + 6 \cdot (2\delta \cdot 3^t + \varepsilon/16) = \varepsilon/2 + \delta \cdot 12 \cdot 3^t$. If we choose the constant C in the definition of δ large enough, then $\delta \cdot 12 \cdot 3^t \leq \varepsilon/2$, completing the proof. ◀

4 Conditional HSGs for Width-4 ROBPs

4.1 Hitting Width-w $\wedge$-Programs

As discussed in Section 1.6, the first step in the proof of Theorem 1.7 is to show that in a $\wedge$-program f, without loss of generality, the expected number of rejecting edges traversed under a uniform random input is at most $\ln(1/\mathbb{E}[f])$. Constant factors are important here; see Figure 2. We prove it by using the probabilistic method to reroute the rejecting edges; the details follow.

▶ **Lemma 4.1** (Expected number of rejecting edges traversed in an $\wedge$-program)**.** *For every width-w $\wedge$-program f, there exists another width-w $\wedge$-program g with the following two properties.*

1. *g computes the same function as f.*
2. *When g reads a uniform random input, the expected number of rejecting edges that are traversed is at most $\ln(1/\mathbb{E}[f])$.*

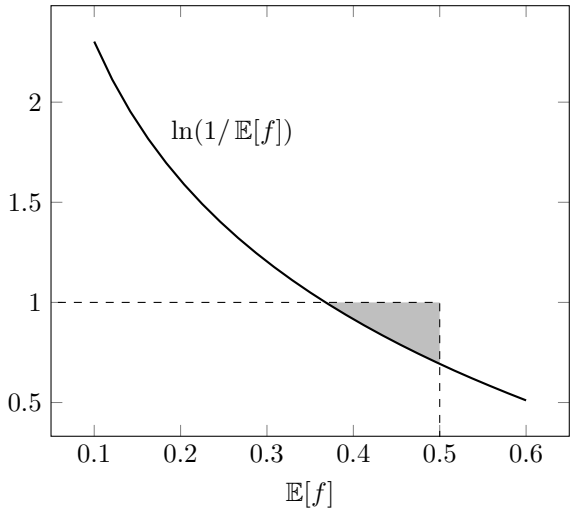

Figure 2 In Lemma 4.1, we care about the base of the logarithm in the bound $\ln(1/\mathbb{E}[f])$; a bound of $\log_2(1/\mathbb{E}[f])$ would not be good enough. The reason is that down the road, to perform error reduction, we will need a $(1/2 - \Omega(1))$-HSG (see Lemma 4.7). To construct such an HSG, we will use the fact that the bound $\ln(1/\mathbb{E}[f])$ in Lemma 4.1 is strictly less than one even when $\mathbb{E}[f]$ is slightly less than $1/2$, since $e > 2$.

Proof. Let $V_0, V_1, \ldots, V_n$ be the layers of vertices of f. For each string $x \in \{0,1\}^{\leq n}$, let $f[x]$ be the vertex that f reaches after reading x. For each $i \in \{0, 1, \ldots, n\}$, let A_i be the set of $x \in \{0,1\}^i$ such that when f reads x, every edge it traverses is an accepting edge. Furthermore, abusing notation, let A_i denote the uniform distribution over the set A_i. We construct g using the probabilistic method as follows.

1. For each $i \in [n]$ independently, we sample a string $X_i^* \sim A_i$.
2. Let g be f, except that for every $i \in [n]$, all rejecting edges from V_{i-1} to V_i are redirected to point to $f[X_i^*]$.

Since we only redirected rejecting edges, g computes the same function as f.

We will prove by induction that for every $i \in \{0, 1, \ldots, n\}$, we have $g[U_i] \sim f[A_i]$, where U_i denotes a uniform random i-bit string that is independent of the randomness used to construct g. When $i = 0$, this is trivial. Now, assuming it holds for i, we can sample $g[U_{i+1}]$ by the following steps.

1. Sample X uniformly at random from A_i (note that $f[X] \sim g[U_i]$).
2. Sample $Y \in \{0,1\}$ uniformly at random.
3. If $XY \notin A_{i+1}$, sample $X_{i+1}^* \sim A_{i+1}$ and output $f[X_{i+1}^*]$.
4. If $XY \in A_{i+1}$, output $f[XY]$.

Conditioned on the event $XY \notin A_{i+1}$, clearly $g[U_{i+1}] \sim f[A_{i+1}]$. Meanwhile, conditioned on the event that $XY \in A_{i+1}$, the string XY is in fact *uniformly* distributed over A_{i+1}, and hence once again $g[U_{i+1}] \sim f[A_{i+1}]$. This completes the inductive step.

Consequently, if we sample $Z \sim U_n$, then the expected number of rejecting edges traversed by $g(Z)$ is

$$\begin{aligned}\sum_{i=1}^{n} \Pr[g(Z) \text{ traverses a rejecting edge in step } i] &= \sum_{i=1}^{n} \Pr[Z_{1\ldots i} \notin A_i \mid Z_{1\ldots i-1} \in A_{i-1}] \\ &\leq \ln(1/\mathbb{E}[f])\end{aligned}$$

by Lemma 2.2. The best case is at least as good as the average case, so there is some fixing of g such that the expected number of rejecting edges traversed by $g(Z)$ with respect to the randomness of Z alone is at most $\ln(1/\mathbb{E}[f])$. ◀

▶ **Corollary 4.2** (PRGs for width-w ROBPs are HSGs for width-w $\wedge$-programs). *Let $w, n \in \mathbb{N}$ and $\beta \in (0,1)$, and let $\varepsilon_\wedge = e^{\beta-1}$. For every width-$w$ length-n $\wedge$-program f with $\mathbb{E}[f] > \varepsilon_\wedge$, there is a set $\mathcal{A}_f$ of width-w standard-order ROBPs such that the following hold.*
1. $|\mathcal{A}_f| = n$.
2. *If X is a distribution over $\{0,1\}^n$ that (β/n)-fools $\mathcal{A}_f$, then* $\mathrm{Supp}(X)$ *hits f.*
In particular, if X fools all width-w length-n standard-order ROBPs with error β/n, then $\mathrm{Supp}(X)$ *hits f.*

Proof. Let g be the program from Lemma 4.1. For each $i \in [n]$, there is a width-w standard-order ROBP g_i such that $g_i(x) = 1$ if and only if $g(x)$ traverses a rejecting edge in step i. Let $\mathcal{A}_f = \{g_1, g_2, \ldots, g_n\}$. Then

$$\begin{aligned}\Pr[f(X) = 0] = \Pr[g_1(X) = 1 \vee \cdots \vee g_n(X) = 1] \leq \sum_{i=1}^{n} \mathbb{E}[g_i(X)] &\leq \sum_{i=1}^{n} (\mathbb{E}[g_i] + \beta/n) \\ &\leq \ln(1/\mathbb{E}[f]) + \beta \\ &< 1.\end{aligned}$$ ◀

4.2 Hitting Width-w $\vee$-Programs

▶ **Lemma 4.3** (HSGs for width-w ROBPs are also HSGs for width-w $\vee$-programs). *For every width-w length-n $\vee$-program f, there is a width-w length-n standard-order ROBP g_f such that $g_f \leq f$ and $\mathbb{E}[g_f] \geq \mathbb{E}[f]/n$. Consequently, for every $H \subseteq \{0,1\}^n$ and every $\varepsilon_\vee > 0$, if H is an $(\varepsilon_\vee/n)$-hitting set for width-w length-n standard-order ROBPs, then H is also an $\varepsilon_\vee$-hitting set for width-w length-n $\vee$-programs.*

Proof. For each $i \in [n]$, there is a width-w standard-order ROBP f_i such that $f_i(x) = 1$ if and only if $f(x)$ traverses an accepting edge in step i. If we sample $X \sim U_n$, then

$$\mathbb{E}[f(X)] = \mathbb{E}[f_1(X) \vee f_2(X) \vee \cdots \vee f_n(X)] \leq \sum_{i=1}^{n} \mathbb{E}[f_i],$$

so there is some i such that $\mathbb{E}[f_i] \geq \mathbb{E}[f]/n$. We can let $g_f = f_i$. ◀

4.3 Hitting Width-$(w+1)$ Programs

We define the following associative operation on hitting sets.

▶ **Definition 4.4** (The $\circledast$ operation). *For any two sets $H, H' \subseteq \{0,1\}^n$, we define*

$$H \circledast H' = \{x_1 x_2 \ldots x_i y_{i+1} y_{i+2} \ldots y_n : x \in H, y \in H', 0 \leq i \leq n\}.$$

Furthermore, we define

$$H^{\circledast t} = \underbrace{H \circledast H \circledast \cdots \circledast H}_{t \text{ copies}}.$$

Note that $|H^{\circledast t}| \leq (n \cdot |H|)^t$.

As discussed in Section 1.6, Gopalan, Meka, Reingold, Trevisan, and Vadhan showed how to convert an $(\frac{\varepsilon^2}{2n})$-hitting set H for width-w $\wedge$-programs into an ε-hitting set H' for width-$(w+1)$ standard-order ROBPs [15]. In particular, they showed that one can take $H' = \{0^n\} \circledast H$. Extending their techniques, we now show that if $H_\vee$ and $H_\wedge$ are an $\varepsilon_\vee$-hitting set for width-w $\vee$-programs and an $\varepsilon_\wedge$-hitting set for width-w $\wedge$-programs, respectively, then $\{0^n\} \circledast H_\vee \circledast H_\wedge$ is an $(\varepsilon_\vee + \varepsilon_\wedge)$-hitting set for width-$(w+1)$ standard-order ROBPs.

▶ **Lemma 4.5** (Simulating width-$(w+1)$ programs using $\vee$-programs and $\wedge$-programs of width w)**.** *Let $\varepsilon_\vee, \varepsilon_\wedge > 0$. For every width-$(w+1)$ standard-order ROBP f with $\mathbb{E}[f] > \varepsilon_\vee + \varepsilon_\wedge$, there is a set $\mathcal{B}_f^\vee$ of width-w length-n $\vee$-programs and a set $\mathcal{B}_f^\wedge$ of width-w length-n $\wedge$-programs such that the following hold.*

1. *$|\mathcal{B}_f^\vee| = w+1$ and $|\mathcal{B}_f^\wedge| \le n$.*
2. *$\mathbb{E}[a] > \varepsilon_\vee$ for every $a \in \mathcal{B}_f^\vee$ and $\mathbb{E}[h] > \varepsilon_\wedge$ for every $h \in \mathcal{B}_f^\wedge$.*
3. *For every $z \in \{0,1\}^n$, for every $H_\vee \subseteq \{0,1\}^n$ such that $H_\vee$ hits $\mathcal{B}_f^\vee$, and for every $H_\wedge \subseteq \{0,1\}^n$ such that $H_\wedge$ hits $\mathcal{B}_f^\wedge$, the set $\{z\} \circledast H_\vee \circledast H_\wedge$ hits f.*

In particular, if $H_\vee$ is an $\varepsilon_\vee$-hitting set for width-w $\vee$-programs and $H_\wedge$ is an $\varepsilon_\wedge$-hitting set for width-w $\wedge$-programs, then $\{0^n\} \circledast H_\vee \circledast H_\wedge$ hits f.

Proof. Let $V_0, V_1, \dots, V_n$ be the layers of f. We assume without loss of generality that $|V_i| > 1$ for every i and $|V_n| = 2$, with one accepting vertex and one rejecting vertex in V_n.

For every $t \in \{0, 1, \dots, n\}$, there is some "good vertex" $u_t \in V_t$ such that $p_{u_t \to} \ge \mathbb{E}[f] > \varepsilon_\vee + \varepsilon_\wedge$. Let $i_* + 1$ be the smallest value such that there is some "bad vertex" $v_{i_*+1} \in V_{i_*+1}$ satisfying $p_{v_{i_*+1} \to} \le \varepsilon_\vee$. The existence of this first bad vertex implies that for every $t \in \{i_* + 2, \dots, n\}$, there is another bad vertex $v_t \in V_t$ satisfying $p_{v_t \to} \le \varepsilon_\vee$.

In brief, our strategy is as follows. We will take the first i_* steps arbitrarily; there is no risk of hitting a bad vertex during this time. Then, we will use $H_\vee$ to reach one of the good vertices (u_t). Afterward, we will use $H_\wedge$ to avoid the bad vertices (v_t).

In more detail, for each $v \in V_{i_*}$, define $a_v \colon \{0,1\}^n \to \{0,1\}$ by letting $a_v(x) = 1$ if and only if there is a value $j \in \{i_*, \dots, n\}$ such that $f_{v\to}(x)$ visits the good vertex u_j. Note that we think of $f_{v\to}$ as a function on n bits that ignores its first i_* many input bits. Observe that a_v can be computed by a width-w length-n $\vee$-program. (We can delete the vertices $u_{i_*}, u_{i_*+1}, \dots, u_n$, redirect their incoming edges arbitrarily, and mark those edges as accepting edges. Layers $0, 1, \dots, i_* - 1$ can have width 1, because that phase of the computation is trivial.) The definition of i_* implies that $\mathbb{E}[f_{v\to}] > \varepsilon_\wedge$. The final good vertex u_n is the accepting vertex, so $\mathbb{E}[a_v] > \varepsilon_\vee$. Let $\mathcal{B}_f^\vee = \{a_v : v \in V_{i_*}\}$.

Furthermore, for each $j \in \{i_*, i_* + 1, \dots, n\}$, define $h_j \colon \{0,1\}^n \to \{0,1\}$ by letting $h_j(x) = 1$ if and only if $f_{u_j \to}(x)$ avoids all of the bad vertices $v_{j+1}, v_{j+2}, \dots, v_n$. Observe that h_j can be computed by a width-w length-n $\wedge$-program. (We can delete the vertices $v_{j+1}, v_{j+2}, \dots, v_n$, redirect their incoming edges arbitrarily, and mark those edges as rejecting edges. Layers $0, \dots, j$ can have width 1, because that phase of the computation is trivial.)

Furthermore,

$$\begin{aligned}
\mathbb{E}[h_j] &= \Pr_{x\sim U_n}[f_{u_j\to}(x) \text{ does not visit } v_{j+1},\dots,v_n] \\
&= \Pr_{x\sim U_n}[f_{u_j\to}(x)=1 \text{ and } f_{u_j\to}(x) \text{ does not visit } v_{j+1},\dots,v_n] \\
&= p_{u_j\to} - \Pr_{x\sim U_n}[f_{u_j\to}(x)=1 \text{ and } f_{u_j\to}(x) \text{ visits at least one of } v_{j+1},\dots,v_n] \\
&= p_{u_j\to} - \sum_{t=j+1}^{n-1} \Pr[f_{u_j\to}(x)=1 \text{ and } f_{u_j\to}(x) \text{ visits } v_t \text{ but not } v_{j+1},\dots,v_{t-1}] \\
&\geq p_{u_j\to} - \sum_{t=j+1}^{n-1} \Pr[f_{u_j\to}(x) \text{ visits } v_t \text{ but not } v_{j+1},\dots,v_{t-1}]\cdot\varepsilon_\vee \qquad (1) \\
&\geq p_{u_j\to} - \varepsilon_\vee \\
&> \varepsilon_\wedge.
\end{aligned}$$

(Equation (1) holds because the probability that $f_{u_j\to}(x)$ accepts conditioned on visiting v_t without visiting $v_{j+1},\dots,v_{t-1}$ is precisely $p_{v_t\to}$, which is at most $\varepsilon_\vee$.) Let $\mathcal{B}_f^\wedge = \{h_{i_*},\dots,h_n\}$.

Finally, let $z\in\{0,1\}^n$ and let $H_\vee, H_\wedge \subseteq \{0,1\}^n$ be sets that hit $\mathcal{B}_f^\vee$ and $\mathcal{B}_f^\wedge$ respectively. We must show that $\{z\}\circledast H_\vee \circledast H_\wedge$ hits f. Since $H_\vee$ hits $\mathcal{B}_f^\vee$, there is some $x\in H_\vee$ and some j_* such that $f(z_{1\dots i_*}x_{i_*+1\dots n})$ visits u_{j_*}. Since $H_\wedge$ hits $\mathcal{B}_f^\wedge$, there is some $y\in H_\wedge$ such that $h_{j_*}(y)=1$, i.e., $f_{u_{j_*}\to}(y)$ avoids all of the bad vertices $v_{j+1},\dots,v_n$. Consequently,

$$f(z_{1\dots i_*}x_{i_*+1\dots j_*}y_{j_*+1\dots n}) = 1. \qquad \blacktriangleleft$$

4.4 Error Reduction

At this point, given an (α/n)-PRG for width-w programs, we have shown how to construct a $(1/2-\Omega(1))$-HSG for width-$(w+1)$ programs. To construct an ε-HSG where ε is small, we will use a version of an error reduction technique introduced by Hoza and Zuckerman [20]. We rely on the following lemma, which shows how to split a branching program into two subprograms, each of which has much higher acceptance probability.

▶ **Lemma 4.6** (Increasing a branching program's acceptance probability)**.** *Let $w,n\in\mathbb{N}$, let $K\in(1,\infty)$, and let f be a width-$(w+1)$ length-n standard-order ROBP with $\mathbb{E}[f]=p\leq 1/K$. Let V be the set of vertices in f and let S be the set of vertices $v\in V$ such that $p_{v\to}\geq Kp$. For each $x\in\{0,1\}^n$, let $g(x)\in\{0,1\}$ indicate whether $f(x)$ visits at least one vertex in S. Then:*

1. *The function g can be computed by a width-$(w+1)$ standard-order ROBP.*
2. $\mathbb{E}[g] > 1/(2K)$.

Proof sketch. Observe that if S intersects a layer of f, then S also intersects all subsequent layers of f. Therefore, to construct a program that computes g, we can redirect all outgoing edges from vertices in S to other vertices in S, and then in the final layer, label the vertices in S as accepting and all other vertices as rejecting. Finally, the fact that $\mathbb{E}[g] > 1/(2K)$ was proven by Bogdanov, Hoza, Prakriya, and Pyne [3, Lemma 25].[7] ◀

Given Lemma 4.6, the following error reduction lemma readily follows.

[7] Bogdanov, Hoza, Prakriya, and Pyne state the bound as $\mathbb{E}[g]\geq 1/(2K)$ [3], but their proof establishes the strict inequality $\mathbb{E}[g] > 1/(2K)$.

▶ **Lemma 4.7** (Error reduction). *Let $w, n \in \mathbb{N}$, let $\gamma \in (0, 1/2)$, let $\varepsilon \in (0,1)$, and let $t = \lceil \frac{\ln(1/\varepsilon)}{2\gamma} \rceil$. For every width-$(w+1)$ length-n standard-order ROBP f such that $\mathbb{E}[f] > \varepsilon$, there is a set $\mathcal{C}_f$ of width-$(w+1)$ length-n standard-order ROBPs such that the following hold.*

1. *$|\mathcal{C}_f| \le (w+1) \cdot n$.*
2. *$\mathbb{E}[g] > 1/2 - \gamma$ for every $g \in \mathcal{C}_f$.*
3. *For any set $H \subseteq \{0,1\}^n$, if H hits $\mathcal{C}_f$, then $H^{\circledast t}$ hits f.*

In particular, if H is a $(1/2-\gamma)$-hitting set for width-$(w+1)$ length-n standard-order ROBPs, then $H^{\circledast t}$ hits f.

Proof. Let $K = \frac{1}{1-2\gamma}$. Let $V_0, V_1, \dots, V_n$ be the layers of f. For every vertex $u \in V_0 \cup \dots \cup V_{n-1}$, we will define a function $g_u \colon \{0,1\}^n \to \{0,1\}$. The definition is as follows.

- First, suppose $p_{u\to} > 1 - 2\gamma$. In this case, we let $g_u = f_{u\to}$.
- Now, suppose $p_{u\to} \le 1 - 2\gamma$. Let S be the set of vertices v in layers beyond u such that $p_{v\to} \ge K \cdot p_{u\to}$. Let $g_u(x)$ indicate whether $f_{u\to}(x)$ visits at least one vertex in S.

Lemma 4.6 ensures that $\mathbb{E}[g_u] > 1/(2K) = 1/2 - \gamma$ and g_u can be computed by a width-$(w+1)$ standard-order ROBP. Let $\mathcal{C}_f = \{g_u : u \in V_0 \cup \dots \cup V_{n-1}\}$.

Now let H be a set that hits $\mathcal{C}_f$. We must show that $H^{\circledast t}$ hits f. By inductively applying the definition of $\mathcal{C}_f$, we see that there is some $x \in H^{\circledast t}$ such that $f(x)$ visits some vertex v such that $p_{v\to} \ge \min\{1, K^t \cdot \varepsilon\}$. By our choices of K and t, we have

$$\varepsilon \cdot K^t \ge \varepsilon \cdot \left(\frac{1}{1-2\gamma}\right)^{\ln(1/\varepsilon)/(2\gamma)} \ge \varepsilon \cdot \left(\frac{1}{e^{-2\gamma}}\right)^{\ln(1/\varepsilon)/(2\gamma)} = 1,$$

so $p_{v_t\to} = 1$. Therefore, $f(x) = 1$. ◀

4.5 The Sampler Trick

Combining the lemmas above yields the following.

▶ **Theorem 4.8** (If X fools width w, then $\mathrm{Supp}(X)^{\circledast k}$ hits width $w+1$). *Let $w \in \mathbb{N}$ and $\beta > 0$ be constants, and assume $e^{\beta-1} + \beta < 1/2$. For every $n \in \mathbb{N}$ and $\varepsilon > 0$, there is a value $k = O(\log(1/\varepsilon))$ such that for every width-$(w+1)$ length-n standard-order ROBP such that $\mathbb{E}[f] > \varepsilon$, there is a set $\mathcal{D}_f$ of width-w length-n standard-order ROBPs such that the following hold.*

1. *$|\mathcal{D}_f| \le O(n^3)$.*
2. *If X is a distribution over $\{0,1\}^n$ that fools $\mathcal{D}_f$ with error β/n, then $\mathrm{Supp}(X)^{\circledast k}$ hits f.*

In particular, if X fools all width-w standard-order ROBPs with error β/n, then $\mathrm{Supp}(X)^{\circledast k}$ hits f.

Proof. Let $\varepsilon_\wedge = e^{\beta-1}$, let $\varepsilon_\vee = \beta$, let $\gamma = \frac{1}{2} - \varepsilon_\wedge - \varepsilon_\vee > 0$, and let $t = \lceil \frac{\ln(1/\varepsilon)}{2\gamma} \rceil$. Let $\mathcal{C}_f$ be the set from Lemma 4.7. For each $g \in \mathcal{C}_f$, let $\mathcal{B}_g^\vee, \mathcal{B}_g^\wedge$ be the sets from Lemma 4.5. Let $\mathcal{B}_\vee = \bigcup_{g \in \mathcal{C}_f} \mathcal{B}_g^\vee$ and $\mathcal{B}_\wedge = \bigcup_{g \in \mathcal{C}_f} \mathcal{B}_g^\wedge$. For every $a \in \mathcal{B}_\vee$, let g_a be the program from Lemma 4.3. For every $h \in \mathcal{B}_\wedge$, let $\mathcal{A}_h$ be the set from Corollary 4.2. Let $\mathcal{D}_f = \{g_a : a \in \mathcal{B}_\vee\} \cup \bigcup_{h \in \mathcal{B}_\wedge} \mathcal{A}_h$.

Let us confirm the cardinality bound. We have $|\mathcal{C}_f| \le O(n)$. For each $g \in \mathcal{C}_f$, we have $|\mathcal{B}_g^\vee| = w+1$ and $|\mathcal{B}_g^\wedge| \le n$; therefore, $|\mathcal{B}_\vee| \le O(wn)$ and $|\mathcal{B}_\wedge| \le O(n^2)$. For each $h \in \mathcal{B}_\wedge$, we have $|\mathcal{A}_h| = n$; therefore, $|\mathcal{D}_f| \le O(wn) + O(n^3) = O(n^3)$ since w is a constant.

Now suppose X is a distribution over $\{0,1\}^n$ that fools $\mathcal{D}_f$ with error β/n. By Corollary 4.2, $\mathrm{Supp}(X)$ is an $\varepsilon_\wedge$-hitting set for $\mathcal{B}_\wedge$. By Lemma 4.3, $\mathrm{Supp}(X)$ is an $\varepsilon_\vee$-hitting set for $\mathcal{B}_\vee$. Therefore, by Lemma 4.5, $\mathrm{Supp}(X)^{\circledast 3}$ is a $(1/2-\gamma)$-hitting set for $\mathcal{C}_f$. Finally, applying Lemma 4.7, we conclude that $(\mathrm{Supp}(X)^{\circledast 3})^{\circledast t}) = \mathrm{Supp}(X)^{\circledast 3t}$ hits f. ◀

At this point, we are nearly done with the proof of Theorem 1.7. Applying Theorem 4.8 directly would lead to a seed length of $O(s(n) \cdot \log(1/\varepsilon))$. To achieve the superior seed length $O(s(n) + \log n \cdot \log(1/\varepsilon))$, we use the so-called "sampler trick." This trick, which was perhaps first used in the context of PRGs by Armoni [1], is based on the concept of an *averaging sampler*, defined as follows.

▶ **Definition 4.9** (Sampler). *Let $\varepsilon, \delta \in (0,1)$. An (ε,δ)-sampler is a function $\mathsf{Samp}\colon \{0,1\}^\ell \times \{0,1\}^d \to \{0,1\}^s$ such that for every function $\phi\colon \{0,1\}^s \to \{0,1\}^n$, we have*

$$\Pr_{x\sim U_\ell}[|\mathbb{E}[\phi] - \mathbb{E}[\phi(\mathsf{Samp}(x,U_s))]| \le \varepsilon] \ge 1-\delta.$$

We need an explicit sampler with good parameters. The following recent construction, by Xun and Zuckerman [34], is more than sufficient for our purposes.

▶ **Theorem 4.10** ([34]). *For every $s \in \mathbb{N}$ and every $\varepsilon, \delta \in (0,1)$, there exists an explicit (ε,δ)-sampler $\mathsf{Samp}\colon \{0,1\}^\ell \times \{0,1\}^d \to \{0,1\}^s$ with $\ell = O(s+\log(1/\delta))$ and $d = O(\log(1/\varepsilon) + \log\log(1/\delta))$.*

Proof of Theorem 1.7. Let $G\colon \{0,1\}^s \to \{0,1\}^n$ be the given PRG. Let $\alpha' > 0$ be a constant small enough that $e^{\alpha+\alpha'-1} + \alpha + \alpha' < 1/2$,[8] and let $\beta = \alpha + \alpha'$. Let $\varepsilon_{\mathsf{Samp}} = \alpha'/n$, and let $\mathsf{Samp}\colon \{0,1\}^\ell \times \{0,1\}^d \to \{0,1\}^s$ be the $(\varepsilon_{\mathsf{Samp}}, \delta_{\mathsf{Samp}})$-sampler from Theorem 4.10, where δ_{Samp} will be specified later.

Let $k = O(\log(1/\varepsilon))$ be the value from Theorem 4.8. For each $x \in \{0,1\}^\ell$, let $G_x = G(\mathsf{Samp}(x, U_d))$. Our hitting set is given by

$$H = \bigcup_{x\in\{0,1\}^\ell} \mathrm{Supp}(G_x)^{\circledast k}.$$

Let us prove that this works. Let f be a width-$(w+1)$ standard-order ROBP such that $\mathbb{E}[f] > \varepsilon$. Let $\mathcal{D}_f$ be the set from Theorem 4.8. For each $g \in \mathcal{D}_f$, we define $\phi_g\colon \{0,1\}^s \to \{0,1\}$ by $\phi_g(z) = g(G(z))$. We choose $\delta_{\mathsf{Samp}} = \Theta(1/n^3)$ such that $\delta_{\mathsf{Samp}} < 1/|\mathcal{D}_f|$. That way, by the sampler definition and the union bound, there exists some $x_* \in \{0,1\}^\ell$ such that for every $g \in \mathcal{D}_f$, we have

$$|\,\mathbb{E}[\phi_g] - \mathbb{E}[\phi_g(\mathsf{Samp}(x_*, U_s))]| \le \varepsilon_{\mathsf{Samp}} = \alpha'/n.$$

Since G fools g with error α/n, we have $|\,\mathbb{E}[\phi_g] - \mathbb{E}[g]| \le \alpha/n$. Therefore, G_{x_*} fools g with error β/n. By Theorem 4.8, it follows that $\mathrm{Supp}(G_{x_*})^{\circledast k}$ hits f.

The seed length of our HSG is $\ell + k \cdot (d + \log n)$, which is $O(s + \log n \cdot \log(1/\varepsilon))$ as promised. ◀

References

1 Roy Armoni. On the derandomization of space-bounded computations. In *Proceedings of the 2nd Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, pages 47–59, 1998. doi:10.1007/3-540-49543-6_5.

2 Andrej Bogdanov, Zeev Dvir, Elad Verbin, and Amir Yehudayoff. Pseudorandomness for width-2 branching programs. *Theory Comput.*, 9:283–292, 2013. doi:10.4086/toc.2013.v009a007.

[8] This is satisfied provided $\alpha + \alpha' < \alpha_* \approx 0.095$.

3 Andrej Bogdanov, William M. Hoza, Gautam Prakriya, and Edward Pyne. Hitting Sets for Regular Branching Programs. In *Proceedings of the 37th Computational Complexity Conference (CCC)*, pages 3:1–3:22, 2022. doi:10.4230/LIPIcs.CCC.2022.3.

4 Mark Braverman, Gil Cohen, and Sumegha Garg. Pseudorandom pseudo-distributions with near-optimal error for read-once branching programs. *SIAM J. Comput.*, 49(5):STOC18–242–STOC18–299, 2020. doi:10.1137/18M1197734.

5 Mark Braverman, Anup Rao, Ran Raz, and Amir Yehudayoff. Pseudorandom generators for regular branching programs. *SIAM J. Comput.*, 43(3):973–986, 2014. doi:10.1137/120875673.

6 Joshua Brody and Elad Verbin. The coin problem, and pseudorandomness for branching programs. In *Proceedings of the 51st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 30–39, 2010. doi:10.1109/FOCS.2010.10.

7 Suresh Chari, Pankaj Rohatgi, and Aravind Srinivasan. Improved algorithms via approximations of probability distributions. *J. Comput. System Sci.*, 61(1):81–107, 2000. doi:10.1006/jcss.1999.1695.

8 Eshan Chattopadhyay and Jyun-Jie Liao. Recursive Error Reduction for Regular Branching Programs. In *15th Innovations in Theoretical Computer Science Conference (ITCS)*, pages 29:1–29:20, 2024. doi:10.4230/LIPIcs.ITCS.2024.29.

9 Ben Chen and Amnon Ta-Shma. Better weighted pseudorandom generators against low weight read-once branching programs. ECCC preprint TR25-067, 2025. URL: https://eccc.weizmann.ac.il/report/2025/067/.

10 Lijie Chen, William M. Hoza, Xin Lyu, Avishay Tal, and Hongxun Wu. Weighted pseudorandom generators via inverse analysis of random walks and shortcutting. In *Proceedings of the 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1224–1239, 2023. doi:10.1109/FOCS57990.2023.00072.

11 Kuan Cheng and William M. Hoza. Hitting sets give two-sided derandomization of small space. *Theory Comput.*, 18:Paper No. 21, 32, 2022. doi:10.4086/toc.2022.v018a021.

12 Kuan Cheng and Ruiyang Wu. Weighted pseudorandom generators for read-once branching programs via weighted pseudorandom reductions. ECCC preprint TR25-027, 2025. URL: https://eccc.weizmann.ac.il/report/2025/027/.

13 Anindya De. Pseudorandomness for permutation and regular branching programs. In *26th Annual Conference on Computational Complexity (CCC)*, pages 221–231, 2011. doi:10.1109/CCC.2011.23.

14 Anindya De, Omid Etesami, Luca Trevisan, and Madhur Tulsiani. Improved pseudorandom generators for depth 2 circuits. In *Proceedings of the 14th International Workshop on Randomization and Computation (RANDOM)*, pages 504–517, 2010. doi:10.1007/978-3-642-15369-3_38.

15 Parikshit Gopalan, Raghu Meka, Omer Reingold, Luca Trevisan, and Salil Vadhan. Better pseudorandom generators from milder pseudorandom restrictions. In *Proceedings of the 53rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 120–129, 2012. doi:10.1109/FOCS.2012.77.

16 Pooya Hatami and William Hoza. Paradigms for unconditional pseudorandom generators. *Found. Trends Theor. Comput. Sci.*, 16(1-2):1–210, 2024. doi:10.1561/0400000109.

17 William M. Hoza. Recent progress on derandomizing space-bounded computation. *Bull. Eur. Assoc. Theor. Comput. Sci. EATCS*, 138:114–143, 2022. URL: http://bulletin.eatcs.org/index.php/beatcs/article/view/728.

18 William M. Hoza, Edward Pyne, and Salil Vadhan. Pseudorandom Generators for Unbounded-Width Permutation Branching Programs. In *12th Innovations in Theoretical Computer Science Conference (ITCS)*, pages 7:1–7:20, 2021. doi:10.4230/LIPIcs.ITCS.2021.7.

19 William M. Hoza, Edward Pyne, and Salil Vadhan. Limitations of the Impagliazzo–Nisan–Wigderson pseudorandom generator against permutation branching programs. *Algorithmica*, 2024. doi:10.1007/s00453-024-01251-2.

20 William M. Hoza and David Zuckerman. Simple optimal hitting sets for small-success **RL**. *SIAM J. Comput.*, 49(4):811–820, 2020. doi:10.1137/19M1268707.

21 Russell Impagliazzo, Noam Nisan, and Avi Wigderson. Pseudorandomness for network algorithms. In *Proceedings of the 26th Annual Symposium on Theory of Computing (STOC)*, pages 356–364, 1994. doi:10.1145/195058.195190.

22 Michal Koucký, Prajakta Nimbhorkar, and Pavel Pudlák. Pseudorandom generators for group products [extended abstract]. In *Proceedings of the 43rd ACM Symposium on Theory of Computing (STOC)*, pages 263–272, 2011. doi:10.1145/1993636.1993672.

23 Vinayak M. Kumar. New Pseudorandom Generators and Correlation Bounds Using Extractors. In *Proceedings of the 16th Innovations in Theoretical Computer Science Conference (ITCS)*, volume 325, pages 68:1–68:23, 2025. doi:10.4230/LIPIcs.ITCS.2025.68.

24 Chin Ho Lee, Edward Pyne, and Salil Vadhan. On the Power of Regular and Permutation Branching Programs. In *Proceedings of the 27th International Conference on Randomization and Computation (RANDOM)*, pages 44:1–44:22, 2023. doi:10.4230/LIPIcs.APPROX/RANDOM.2023.44.

25 Raghu Meka, Omer Reingold, and Avishay Tal. Pseudorandom generators for width-3 branching programs. In *Proceedings of the 51st Annual Symposium on Theory of Computing (STOC)*, pages 626–637, 2019. doi:10.1145/3313276.3316319.

26 Augusto Modanese. Pseudorandom generators for sliding-window algorithms. arXiv preprint 2301.07384, 2023. doi:10.48550/arXiv.2301.07384.

27 Joseph Naor and Moni Naor. Small-bias probability spaces: efficient constructions and applications. *SIAM J. Comput.*, 22(4):838–856, 1993. doi:10.1137/0222053.

28 Noam Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992. doi:10.1007/BF01305237.

29 Edward Pyne, Ran Raz, and Wei Zhan. Certified hardness vs. randomness for log-space. In *Proceedings of the 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 989–1007, 2023. doi:10.1109/FOCS57990.2023.00061.

30 Edward Pyne and Salil Vadhan. Pseudodistributions That Beat All Pseudorandom Generators (Extended Abstract). In *Proceedings of the 36th Computational Complexity Conference (CCC)*, pages 33:1–33:15, 2021. doi:10.4230/LIPIcs.CCC.2021.33.

31 Omer Reingold, Luca Trevisan, and Salil Vadhan. Pseudorandom walks on regular digraphs and the **RL** vs. **L** problem. In *Proceedings of the 38th Annual Symposium on Theory of Computing (STOC)*, pages 457–466, 2006. doi:10.1145/1132516.1132583.

32 Michael Saks and David Zuckerman, 1995. Unpublished.

33 Thomas Steinke. Pseudorandomness for permutation branching programs without the group theory. ECCC preprint TR12-083, 2012. URL: https://eccc.weizmann.ac.il/report/2012/083/.

34 Zhiyang Xun and David Zuckerman. Near-optimal averaging samplers and matrix samplers. ECCC preprint TR24-097, 2024. URL: https://eccc.weizmann.ac.il/report/2024/097/.