Electronic Colloquium on Computational Complexity, Revision 5 of Report No. 27 (2025)

Weighted Pseudorandom Generators for Read-Once Branching Programs via Weighted Pseudorandom Reductions

Kuan Cheng * Ruiyang Wu †

Abstract

We study weighted pseudorandom generators (WPRGs) and derandomizations for read-once branching programs (ROBPs). Denote n and w as the length and the width of a ROBP. We have the following results.

For standard ROBPs, we give an explicit ε-WPRG with seed length

$$O\left(\frac{\log n \log(nw)}{\max\{1, \log\log w - \log\log n\}} + \log w\left(\log\log\log w - \log\log\max\left\{2, \frac{\log w}{\log \frac{n}{\varepsilon}}\right\}\right) + \log\frac{1}{\varepsilon}\right).$$

When $n = w^{o(1)}$, this is better than the WPRGs of [Hoz21, CDR+21, PV21, CL20]. Further as a direct application, we attain a WPRG for regular ROBPs with a better seed length than that of [CHL+23, CL24].

For permutation ROBPs with unbounded widths and single accept nodes, we give an explicit ε-WPRG with seed length

$$O\left(\log n\left(\log\log n + \sqrt{\log(1/\varepsilon)}\right) + \log(1/\varepsilon)\right),$$

improving [CHL+23]. A key difference to [CHL+23] is that this implies a WPRG with optimal seed length for short-wide ROBPs with multiple accept nodes. Specifically, after switching to multiple accept nodes in a standard way by replacing ε with ε/w, this gives a WPRG with optimal seed length $O(\log w)$ for $n = 2^{O(\sqrt{\log w})}$, and error $1/\operatorname{poly} w$. The only previous work attaining optimal seed lengths are Nisan-Zuckerman style PRGs [NZ96, Arm98] but they are only optimal for $n = \operatorname{poly}\log w, \varepsilon = 2^{-\log^{0.9} w}$.

We also give a new Nisan-Zuckerman style derandomization for regular ROBPs with width w, length $n = 2^{O(\sqrt{\log w})}$, and multiple accept nodes. We attain optimal space complexity $O(\log w)$ for arbitrary approximation error $\varepsilon = 1/\operatorname{poly} w$. When requiring the derandomization to be in $\mathbf{L}$, again the only previous result is by Nisan-Zuckerman style PRGs [NZ96, Arm98], which are only optimal for $n = \operatorname{poly}\log w, \varepsilon = 2^{-\log^{0.9} w}$. Also, if compared to [AKM+20, CHL+23, CL24], which can be viewed as Saks-Zhou style derandomizations, then for $n = 2^{O(\sqrt{\log w})}$ our derandomization not only improves the space complexity to optimal, but also substantially improves the time complexity from super polynomial to standard polynomial in w. Note that derandomizations of [AKM+20, CHL+23, CL24] has space complexity $S = O(\log(nw)\log\log(nw/\varepsilon))$ and time complexity exponential in S.

All our results are based on iterative weighted pseudorandom reductions, which can iteratively reduce fooling long ROBPs to fooling short ones.

*CFCS, School of Computer Science, Peking University. ckkcdh@pku.edu.cn.
†CFCS, School of Computer Science, Peking University. wuruiyang@stu.pku.edu.cn.

ISSN 1433-8092

1 Introduction.

Randomness is a fundamental resource in computation, but is it essential? A key conjecture in space-bounded computation is that randomized algorithms in the complexity class **BPL** can be efficiently simulated by deterministic logspace algorithms, i.e. $\mathbf{BPL} = \mathbf{L}$. A central approach toward addressing this conjecture is the derandomization of standard-order read-once branching programs (ROBPs), which is usually defined as the following.

Definition 1.1 (Read-once branching programs (ROBP))**.** *A read-once branching program f of length n, width w and alphabet size $|\Sigma| = 2^s$ is a directed acyclic graph with $n+1$ layers $V_0, \ldots, V_n$. For any layer V_i except V_n, each node $v \in V_i$ has 2^s outgoing edges to nodes in V_{i+1}. These edges are labeled by distinct symbols in Σ. There exists a unique start node $v_{start} \in V_0$ and a set of accept nodes $V_{accept} \subset V_n$. Given an input $x \in \Sigma^n$, the computation of $f(x)$ is defined as: $f(x) = 1$, if there exists a unique path $v_{start}, v_1, \ldots, v_n$ such that the edge between v_i and v_{i+1} is labeled by x_i, and $v_n \in V_{accept}$; $f(x) = 0$ otherwise.*

Every problem in **BPL** can be reduced to approximating $\mathbb{E}f$ for a corresponding ROBP f. A classical method for derandomizing ROBPs is constructing pseudorandom generators (PRGs).

Definition 1.2 (PRG)**.** *Let $\mathcal{F}$ be a class of ROBPs $f : (\{0,1\}^s)^n \to \{0,1\}$. An ε-PRG for $\mathcal{F}$ is a function $G : \{0,1\}^d \to (\{0,1\}^s)^n$ such that for every $f \in \mathcal{F}$, we have*

$$\left| \mathbb{E}_{x \in (\{0,1\}^s)^n} f(x) - \mathbb{E}_{r \in \{0,1\}^d} f(G(r)) \right| \leq \varepsilon.$$

*The input length d is called the seed length of the PRG. We say that G is **explicit** if it can be computed in space $O(d)$ and time* $\mathrm{poly}(d, n)$.

By the probabilistic method, one can show the existence of a non-uniform ε-PRG for standard-order ROBPs of length n, width w, and alphabet size 2^s, with an optimal seed length of $O(s + \log(nw/\varepsilon))$. However, constructing explicit PRGs that have short seed lengths turns out to be an exceptional challenge. In a seminal work, Nisan [Nis92a] constructed an explicit ε-PRG for ROBPs of length n, width w, and binary alphabet $\{0,1\}$, with a seed length of $O(\log n \log(nw/\varepsilon))$. Building on this, Saks and Zhou [SZ99] developed a celebrated algorithm to derandomize **BPL**, within $O(\log^{3/2} n)$ space deterministically. Nisan [Nis92b] also used [Nis92a] to show that $\mathbf{BPL} \subseteq \mathbf{SC}$. Impagliazzo, Nisan, and Wigderson [INW94] generalized the construction of [Nis92a] by using expanders, to fool more general models in network communication. In the meantime, for short ROBPs, Nisan and Zuckerman [NZ96] gave another remarkable PRG that has an optimal seed length for wide but short ROBPs, i.e. with seed length $O(\log w)$ when $n = \mathrm{poly}\log w, \varepsilon = 2^{-\log^{0.99} w}$. Armoni [Arm98] further extended [Nis92a, NZ96] to construct an improved PRG with seed length[1] $O\left(\frac{\log n \log(nw/\varepsilon)}{\max\{1, \log\log w - \log\log(n/\varepsilon)\}}\right)$.

1.1 Weighted pseudorandom generators.

Despite years of research, the challenging problem of constructing better PRGs for general ROBPs remains open. However, PRGs are not the only black-box method for derandomizations. A remarkable work by Braverman, Cohen, and Garg [BCG19] improves the seed length by introducing and constructing WPRGs.

[1] [Arm98] needs to use an extractor with seed length optimal up to constant factors which is discovered later than [Arm98], e.g. [GUV09].

Definition 1.3 (WPRG)**.** *Let $\mathcal{F}$ be a class of ROBPs $f : (\{0,1\}^s)^n \to \{0,1\}$. A W-bounded ε-WPRG for $\mathcal{F}$ is a function $(G,\sigma) : \{0,1\}^d \to (\{0,1\}^s)^n \times \mathbb{R}$ such that for every $f \in \mathcal{F}$, we have*

$$\left| \mathbb{E}_{x\in(\{0,1\}^s)^n} f(x) - \sum_{r\in\{0,1\}^d} \left[\frac{1}{2^d} \sigma(r) \cdot f(G(r)) \right] \right| \le \varepsilon,$$
$$\forall r, |\sigma(r)| \le W.$$

*The input length d is called the seed length of the WPRG. We say that (G,w) is **explicit** if it can be computed in space $O(d)$ and time* $\mathrm{poly}(d,n)$.

As shown in [BCG19], under this notion, the seed length can be significantly improved to $\widetilde{O}(\log n \log(nw) + \log \frac{1}{\varepsilon})$, i.e. for ε there is only an isolated addend. Chattopadhyay and Liao [CL20] further improved this construction to attain seed length $O(\log n \log(nw) \log\log(nw) + \log \frac{1}{\varepsilon})$. Then Cohen, Doron, Renard, Renard, Sberlo, and Ta-Shma [CDR+21], and also Pyne and Vadhan [PV21] gave black-box error reductions from large error PRGs to small error WPRGs. They both are based on a preconditioned Richardson Iteration, which was previously developed by Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan [AKM+20] for high precision derandomization of random walks on Eulerian graphs. Hoza [Hoz21] further improved the error reduction based WPRG constructions to attain seed length $O(\log n \log(nw) + \log(1/\varepsilon))$.

Following these new error reduction methods, there are several new progress on derandomizations. Hoza [Hoz21] improved the derandomization of Saks and Zhou [SZ99] to be $\mathbf{BPL} \subseteq \mathbf{DSPACE}\left(\frac{\log^{3/2} n}{\sqrt{\log\log n}}\right)$. Cohen, Doron, Sberlo, and Ta-Shma [CDST23], also Pyne and Putterman [PP23], showed that ROBPs with medium width $w = 2^{O(\sqrt{\log n})}$ can be derandomized in $\widetilde{O}(\log n)$ space. Cheng and Wang [CW24] showed that $\mathbf{BPL} \subseteq$ logspace-uniform $\mathbf{AC}^1$.

WPRG is stronger than Hitting Set Generator (HSG), while HSG is already a powerful tool in derandomization.

Definition 1.4 (HSG)**.** *Let $\mathcal{F}$ be a class of ROBPs $f : (\{0,1\}^s)^n \to \{0,1\}$. An ε-HSG for $\mathcal{F}$ is a function $H : \{0,1\}^d \to (\{0,1\}^s)^n$ such that for every $f \in \mathcal{F}$, if $\mathbb{E}_{x\in(\{0,1\}^s)^n} f(x) \ge \varepsilon$, then $\exists r \in \{0,1\}^d, f(H(r)) = 1$.*

*The input length d is called the seed length of the HSG. We say that H is **explicit** if it can be computed in space $O(d)$ and time* $\mathrm{poly}(d,n)$.

One can use HSG to find an accepting path when the acceptance probability is significant. Actually, HSG is more powerful than this. Cheng and Hoza [CH22] showed how to approximate the acceptance probabilities of ROBPs by HSGs for larger-size ROBPs. Pyne, Raz, and Zhan [PRZ23] further extended the method to give a deterministic sampler for such tasks. Constructing better HSGs is also a challenging task. Hoza and Zuckerman gave a HSG with seed length $O\left(\frac{\log n \log(nw)}{\max\{1, \log\log w - \log\log n\}} + \log(1/\varepsilon)\right)$ which is optimal when $n = \mathrm{poly}\log w$.

1.2 WPRG for short-wide standard ROBPs.

Nisan-Zuckerman PRG and Armoni's generator are better in seed length than Nisan's PRG or INW PRG for short-wide cases when the error is large. It is a natural question whether one can transform these PRGs to WPRGs to attain better seed lengths for smaller errors. A specific interesting case is $n = \mathrm{poly}\log w$, for which NZ PRG [NZ96] and Armoni's PRG [Arm98] have optimal seed lengths of $O(\log w)$ when $\varepsilon = 2^{-\log^{1-v} w}$, where v is any constant in $(0,1)$. However, when the error is smaller,

such as $\varepsilon = 1/\operatorname{poly}(w)$, the seed lengths of the WPRGs of [CDR+21, Hoz21, PV21] deteriorate to $O(\log w \log\log w)$, which has no advantage over those of the PRGs [Nis92a, INW94, Arm98].

In this paper, we make progress for answering the question by giving a new WPRG construction that has a better seed length for short-wide ROBPs. Specifically, we have the following result.

Theorem 1.5. *For every $n, w \in \mathbb{N}$, $\varepsilon \in (0,1)$, with $n \geq \log w$, there exists an explicit ε-WPRG with seed length*

$$O\left(\frac{\log n \log(nw)}{\max\{1, \log\log w - \log\log n\}} + \log w \left(\log\log\log w - \log\log\max\left\{2, \frac{\log w}{\log(n/\varepsilon)}\right\}\right) + \log(1/\varepsilon)\right)$$

for the class of ROBPs of width w, length n and alphabet $\{0,1\}$.

The seed length is better than [Hoz21, CDR+21, PV21, CL20], when $n \ll w$. For the Nisan-Zuckerman regime i.e. $n = \operatorname{poly}\log w$, $\varepsilon = 2^{-\log^{1-\alpha} n}$ for any constant $\alpha > 0$, the seed length of our Theorem 1.5 is $O(\log w)$, since the subtraction of the two triple-log terms becomes a constant. This matches the seed length of the Nisan-Zuckerman PRG [NZ96]. Further when n is larger or ε is smaller, our seed length is strictly better. Table 1 and Table 2 summarize the seed length of ε-PRGs and ε-WPRGs for short-wide general ROBPs.

Seed length	Type	Reference
$O\left(\frac{\log n \log(nw/\varepsilon)}{\log\log w - \log\log(n/\varepsilon)}\right)$	PRG	[Arm98, KNW08]
$\widetilde{O}(\log n \log(nw) + \log\frac{1}{\varepsilon})$	WPRG	[BCG19]
$O(\log n \log(nw) \log\log(nw) + \log\frac{1}{\varepsilon})$	WPRG	[CL20]
$O\left(\log n \log(nw) + \log(w/\varepsilon) \log\log_n(1/\varepsilon)\right)$	WPRG	[CDR+21]
$O(\log n \log(nw) + \log(1/\varepsilon))$	WPRG	[Hoz21]
$O\left(\frac{\log n \log(nw)}{\log\log w - \log\log n} + \log w \log\log\log w + \log(1/\varepsilon)\right)$	WPRG	Theorem 1.5

Table 1: Comparison of seed length of ε-WPRGs for general ROBPs of width w, length $n < w$, and alphabet $\{0,1\}$.

Seed length	Type	Reference	Note
$O(\log w)$	PRG	[NZ96]	$\varepsilon = 2^{-\log^{0.99} w}$
$O(\log(w/\varepsilon) \log\log w)$	PRG	[Arm98, INW94, Nis92a]	$\varepsilon \leq 1/\operatorname{poly}(w)$
$O(\log w \log\log w + \log(1/\varepsilon))$	WPRG	[Hoz21, CDR+21]	$\varepsilon \leq 1/\operatorname{poly}(w)$
$O(\log w)$	WPRG	Theorem 1.5	$\varepsilon = 2^{-\log^{0.99} w}$
$O(\log w \log\log\log w + \log(1/\varepsilon))$	WPRG	Theorem 1.5	$\varepsilon \leq 1/\operatorname{poly}(w)$
$O(\log w + \log(1/\varepsilon))$	PRG	folklore	Optimal; non-explicit

Table 2: Comparison of seed length of ε-WPRGs for short-wide ROBPs of width w, length $n = \operatorname{poly}(\log w)$, and alphabet $\{0,1\}$.

Another interesting point is that our construction deploys a new framework which iteratively applies weighted pseudorandom reductions. We will describe this in detail in later sections.

As a direct application of our WPRG, we also attain a better WPRG for regular ROBPs.

Definition 1.6 (Regular ROBP)**.** *A regular ROBP is a standard-order ROBP f, where for every $i \in [n]$, the bipartite graph induced by the nodes in layers V_{i-1} and V_i is a regular graph.*

Theorem 1.7. *For every $w, n \in \mathbb{N}$, and every $\varepsilon > 0$, there exists an explicit ε-WPRG with seed length*

$$O\left(\log n\left(\sqrt{\log(1/\varepsilon)} + \log w + \log\log n\right) + \log(1/\varepsilon)\right).$$

for regular ROBPs of width w, length n and binary alphabet.

This is better than that of Chen, Hoza, Lyu, Tal, and Wu [CHL+23] which has a seed length $\widetilde{O}\left(\log n\left(\sqrt{\log(1/\varepsilon)} + \log w + \log\log n\right) + \log(1/\varepsilon)\right)$, and matches an independent work of Chen and Ta-Shma [CTS25a] for the binary case. For larger alphabets, we can actually attain a seed length with a better dependence on the alphabet size than [CTS25a]. See our Theorem 3.19 for details. The construction of [CTS25a] combines the methods of [CHL+23] and Hoza [Hoz21]. Our construction is using our main technical theorem Theorem 3.2 to replace the INW PRG generating correlated seeds in [CHL+23], with an argument based on weighted pseudorandom reductions.

1.3 WPRG for unbounded-width permutation ROBPs with a single accept node.

Our framework works even better for constructing WPRGs against permutation ROBPs. Permutation ROBPs are interesting special ROBPs, where the transition functions between layers are permutations.

Definition 1.8 (permutation ROBP)**.** *A (standard-order) permutation ROBP is a standard-order ROBP f, where for every $i \in [n]$ and $x \in \{0,1\}^s$, the transition matrix from V_i to V_{i+1} through edges labeled by x, is a permutation matrix in $\mathbb{R}^{w\times w}$.*

Early work on PRGs for permutation ROBPs [BV10, De11, KNP11, Ste12, RSV13, CHHL19] focus on constant width cases. A remarkable line of recent studies develops PRG/WPRG/HSGs for unbounded-width permutation ROBPs with single accepting nodes [HPV21a, PV21, BHPP22, CHL+23]. Hoza, Pyne and Vadhan [HPV21a] showed an ε-PRG for unbounded-width permutation ROBPs with seed length $\widetilde{O}(\log n \log(1/\varepsilon))$. They also proved that any PRG for this class must have seed length $\widetilde{\Omega}(\log n \log(1/\varepsilon))$. For the WPRG case, Pyne and Vadhan [PV21] showed a ε-WPRG with seed length $\widetilde{O}(\log n\sqrt{\log(n/\varepsilon)} + \log(1/\varepsilon))$. This was improved by Chen, Hoza, Lyu, Tal, and Wu [CHL+23] to $O\left(\log n\sqrt{\log(1/\varepsilon)}\sqrt{\log\log(n/\varepsilon)} + \log(1/\varepsilon)\log\log(n/\varepsilon)\right)$.

We give an improved WPRG against permutation ROBPs as the following.

Theorem 1.9. *For every $n, s \in \mathbb{N}$ and $\varepsilon \in (0,1)$, there exists an explicit ε-WPRG with seed length*

$$O\left(s + \log n\left(\log\log n + \sqrt{\log(1/\varepsilon)}\right) + \log(1/\varepsilon)\right)$$

for the class of permutation ROBPs of length n and alphabet $\{0,1\}^s$ with a single accept node.

The comparison of our result with the previous results is shown in Table 3.

Seed length	Type	Reference	Note
$\widetilde{O}(\log n \log(1/\varepsilon))$	PRG	[HPV21a]	
$\widetilde{O}(\log n\sqrt{\log(n/\varepsilon)} + \log(1/\varepsilon))$	WPRG	[PV21]	
$O\left(\log n\sqrt{\log(1/\varepsilon)}\sqrt{\log\log(n/\varepsilon)} + \log(1/\varepsilon)\log\log(n/\varepsilon)\right)$	WPRG	[CHL+23]	
$O\left(\log n\left(\log\log n + \sqrt{\log(1/\varepsilon)}\right) + \log(1/\varepsilon)\right)$	WPRG	This work	
$\Omega(\log n \log(1/\varepsilon))$	PRG	[HPV21a]	lower bound

Table 3: Comparison of seed length of ε-PRGs and ε-WPRGs for unbounded-width permutation ROBPs of length n and alphabet $\{0,1\}$ with one accepting node.

Note that by replacing ε with ε/w, one can attain a WPRG for multiple accept nodes. In this way our result gives a WPRG with seed length $O\left(s + \log n\left(\log\log n + \sqrt{\log(w/\varepsilon)}\right) + \log(w/\varepsilon)\right)$. This reveals a substantial difference between our result and [CHL+23]. Notice that our seed length is optimal $O(\log w)$ for $n = 2^{O(\sqrt{\log w})}, \varepsilon = 1/\operatorname{poly} w$, i.e. it implies a Nisan-Zuckerman style PRG for short-wide permutation ROBPs. The only prior work attaining optimal seed length for such ROBPs are [NZ96, Arm98] but only work for $n = \operatorname{poly}\log w, \varepsilon = 2^{-\log^{0.9} w}$. It is not clear how to use [CHL+23] to achieve optimal seed length even for constant error, since when switching to multiple accept nodes, the only known way is to replace ε with ε/w.

We also remark that one may want to use the sampler technique of [Hoz21] on the WPRG of [CHL+23] to attain similar parameters to our Theorem 1.9. However, one barrier is that it is unclear if it is possible to attain samplers supporting the sv-approximation used in the analysis of [CHL+23], with desired parameters.

1.4 Derandomization for short-wide regular ROBPs.

Our framework also provides a new derandomization for regular ROBPs. There is an extensive body of work on constructing HSGs, WPRGs, and PRGs for regular ROBPs [BRRY14, BHPP22, De11, RSV13, CL24, CHL+23]. For derandomization, dedicated derandomization algorithms achieve better performance. Ahmadinejad, Kelner, Murtagh, Peebles, Sidford, and Vadhan [AKM+20] derandomized regular ROBPs of length n, width w and alphabet $\{0,1\}$ within error ε, using space $O(\log(nw)\log\log(nw/\varepsilon))$. Subsequently, Chen, Hoza, Lyu, Tal, and Wu [CHL+23] achieved the same result with a simplified algorithm. Chattopadhyay and Liao [CL24] constructed another alternate algorithm with the same space complexity. These derandomizations can be viewed as Sak-Zhou style derandomizations since they can work for full length with small space. However their time complexity is exponential in their space complexity, hence is not polynomial.

Our new derandomization, stated as the following, can be viewed as a Nisan-Zuckerman style derandomization, since it has optimal logspace complexity for short-wide ROBPs.

Theorem 1.10. *There exists an algorithm that for any input regular ROBP f of width w, length $n = 2^{O(\log^{1/2} w)}$, alphabet $\{0,1\}^s$, and for any input parameter $\varepsilon \geq 1/\operatorname{poly} w$, outputs an approximation for $\mathbb{E}[f]$ with addtive error ε. The algorithm uses $O(s + \log w)$ bits of space.*

Notice that the space complexity is optimal up to constant factors, placing the derandomization precisely in **L**, as long as the alphabet bit-length $s = O(\log w)$. Also notice that this implies our algorithm is in time $\operatorname{poly}(w)$. For derandomizing regular ROBPs within **L**, the only prior feasible methods are Nisan-Zuckerman generator [NZ96] and Armoni's generator [Arm98]. Our result significantly improves the length of the regular ROBPs that can be derandomized in **L** from $\operatorname{poly}\log w$ to $2^{O(\log^{1/2} w)}$ and also improves the precision from $2^{-\log^{0.99} w}$ to arbitrary $1/\operatorname{poly} w$.

If compared to the Saks-Zhou style derandomizations of [AKM+20, CHL+23, CL24], then our algorithm not only attains optimal space complexity $O(\log w)$, but also has a substantial improvement for time complexity from super-polynomial to standard polynomial, as long as $n = 2^{O(\log^{1/2} w)}$. Note that derandomizations of [AKM+20, CHL+23, CL24] are not in polynomial time even if $n = \text{poly}(\log w)$ and $\varepsilon = O(1)$.

1.5 Technical Overview

WPRGs for standard-order ROBPs We start by reviewing the Richardson iteration based error reduction [AKM+20, CDR+21, PV21] that attains low-error WPRGs from large-error PRGs.

The Richardson iteration based error reduction can be viewed as a procedure that produces high-precision approximations for iterated matrices multiplications, given low-precision approximations. Let $\{A_i\}_{i=1}^n \subseteq \mathbb{R}^{w\times w}$ be transition matrices, and $\{B_{i,j}\}_{0\le j<i\le n} \subseteq \mathbb{R}^{w\times w}$ be approximations satisfying $\|B_{i,j} - A_i \cdots A_{j+1}\| \le \varepsilon_0/(2(n+1))$. For any integer k, a standard method, e.g. [CDR+21], indicates there is a weighted sum $P = \sum_{i=1}^K \sigma_i B_{n_{i,1},n_{i,2}} \cdots B_{n_{i,k-1},n_{i,k}}$ of $K = n^{O(k)}$ terms such that

$$\|A_n \cdots A_1 - P\| \le \varepsilon_0^{k/2}(n+1),$$

where $\sigma_i \in \{-1, 0, 1\}$ and $\forall i$, the sequence $n_{i,j}, j \in [k]$ is an increasing sequence of indices in $[n]$.[2] To apply this to an ROBP f, one can instantiate the matrices as

- A_i: the stochastic matrix of f from layer $i-1$ to i.
- $B_{i,j} := \mathbb{E}_{x\sim\{0,1\}^s}\left[f^{j\to i}(\mathsf{PRG}_{j\to i}(x))\right]$, where $f^{j\to i}$ is the transition of f from layer j to layer i, and $\mathsf{PRG}_{j\to i}$ is an $\frac{\varepsilon_0}{2(n+1)}$-PRG with seed length s and outputting $i-j$ symbols.

This immediately yields a high-precision approximation of the expectation of the ROBP:

$$\left|\mathbb{E}_U[f(U)] - \sum_{i=1}^K \sigma_i \cdot \mathbb{E}_{x_1,\ldots,x_k\sim\{0,1\}^s} f\left(\mathsf{PRG}_{0\to n_{i,1}}(x_1), \ldots, \mathsf{PRG}_{n_{i,k-1}\to n_{i,k}}(x_k)\right)\right| \le \varepsilon_0^{k/2}(n+1).$$

Notice that this also immediately gives a WPRG for f with error $\varepsilon_0^{k/2}(n+1)$ by definition. However the k independent PRG callings cost too much randomness. [CDR+21] further reduces the randomness by using an INW generator to generate the seeds for the k independent PRG callings. But since INW does not have optimal seed length, this takes $O(\log k \log \frac{kw}{\varepsilon} + s)$ randomness which has an extra $O(\log k)$ factor in the main term. Using the sampler technique of [Hoz21], one can avoid this factor, but only when the starting error ε_0 is already $1/\operatorname{poly} w$. If the large error PRG has $\varepsilon_0 \gg 1/\operatorname{poly} w$, then this does not work as desired. For example, if the large-error PRG is the NZ PRG, then it is unclear how to use this to derive an optimal seed length for $\varepsilon = 1/\operatorname{poly} w$.

To address the problem, we introduce a new strategy. For each $i \in [K]$, one can view

$$f_i(y_1, \ldots, y_k) := f\left(\mathsf{PRG}_{0\to n_{i,1}}(y_1), \mathsf{PRG}_{n_{i,1}\to n_{i,2}}(y_2), \ldots, \mathsf{PRG}_{n_{i,k-1}\to n_{i,k}}(y_k)\right),$$

as a new ROBP with a shorter length k and a large alphabet, while the width is still w. To fool f, one only need to fool these shorter ROBPs. Based on this observation, our high-level idea is to iteratively apply the above procedure to reduce the ROBPs to even shorter ones until the length become a constant such that the reduced ROBPs can be fooled trivially.

We use the notion *weighted pseudorandom reduction* to describe the construction in details.

[2] See Appendix B for details.

Definition 1.11 (Weighted Pseudorandom Reduction)**.** *Let $\mathcal{B}_{n_0,s_0,w}$ be the class of length-n_0, width-w ROBPs over alphabet $\Sigma_0 = \{0,1\}^{s_0}$. Let $\mathcal{B}_{n_1,s_1,w}$ be the class of length-n_1, width-w ROBPs over alphabet $\Sigma_1 = \{0,1\}^{s_1}$. A (d, K, ε)-reduction from $\mathcal{B}_{n_0,s_0,w}$ to $\mathcal{B}_{n_1,s_1,w}$ is a tuple (R, σ), in which $\mathsf{R} : (\{0,1\}^{s_1})^{n_1} \times \{0,1\}^d \to (\{0,1\}^{s_0})^{n_0}$ and $\sigma : \{0,1\}^d \to \mathbb{R}$. Furthermore, for every $f \in \mathcal{B}_{n_0,s_0,w}$, we have:*

$$\left| \mathbb{E}_U f(U) - \frac{1}{2^d} \sum_{i \in \{0,1\}^d} \sigma(i) \mathbb{E}_{U'}[f(\mathsf{R}(U', i))] \right| \leq \varepsilon,$$
$$\forall i, |\sigma(i)| \leq K,$$
$$\forall i, f(\mathsf{R}(\cdot, i)) \in \mathcal{B}_{n_1,s_1,w}.$$

We call R the reduction function and σ the weight function.

Let the original length-n, width-w ROBP be f over alphabet Σ. For simplicity, now let the target error $\varepsilon = 1/\operatorname{poly} w$. We plan to construct a sequence of length reductions as the following. Let $n_0 = n$. The i-th length reduction, which is instantiated by the Richardson iteration based error reduction with a base $\frac{\varepsilon_i}{2(n_{i-1}+1)}$-PRG, is a (d_i, K_i, ε)-reduction from $\mathcal{B}_{n_{i-1},s_{i-1},w}$ to $\mathcal{B}_{n_i,s_i,w}$, where we take

$$n_i = O\left(\frac{\log(n_{i-1}/\varepsilon)}{\log(1/\varepsilon_i)}\right).$$

This may introduce error

$$\varepsilon_i^{n_i/2}(n_{i-1}+1) \leq \varepsilon,$$

as it is instantiated by the Richardson iteration based error reduction. Notice that as i goes up, one can decrease ε_i such that n_i can be smaller and smaller. One may doubt that a smaller ε_i may require a more precise base PRG, which may require a longer seed. However, the seed length depends on $n_{i-1}, w, \varepsilon_i$. As n_{i-1} is already decreased by the previous reduction, one can pick a smaller ε_i without increasing the seed length. Another issue is that the alphabet is accumulating quickly as the seed length of the base PRG will become the alphabet bit-length for the next reduction. Our solution is to apply an alpahbet reduction after each length reduction to retain a small alphabet. That is, we construct our reduction from $\mathcal{B}_{n_{i-1},s_{i-1},w}$ to $\mathcal{B}_{n_i,s_i,w}$ by composing a length reduction (from $\mathcal{B}_{n_{i-1},s_{i-1},w}$ to $\mathcal{B}_{n_i,s'_i,w}$) with a subsequent alphabet reduction (from $\mathcal{B}_{n_i,s'_i,w}$ to $\mathcal{B}_{n_i,s_i,w}$).

More concretely, the l length reductions mentioned above are instantiated by the Richardson iteration based error reduction with Armoni's generator [Arm98, KNW08] as base PRGs. For a length-n, width-w ROBP over $\Sigma = \{0,1\}^s$, we first choose $\varepsilon_0 = 2^{-\sqrt{\log w \log n}}$ and apply a ($d_0 = \log K_0, K_0, \varepsilon$)-reduction from $\mathcal{B}_{n,s,w}$ to $\mathcal{B}_{n_1,s'_1,w}$, in which $n_1 = O\left(\sqrt{\frac{\log w}{\log n}}\right)$, $K_0 = 2^{O\left(\sqrt{\log n \log w}\right)}$ and $s'_1 = O\left(s + \frac{\log n \log(nw/\varepsilon_0)}{\log\log w - \log\log(n/\varepsilon_0)}\right)$. For the subsequent $l-1$ reductions i.e. for $i = 1, 2 \ldots, l-1$, we setup a $(d_i = \log K_i, K_i, \varepsilon_i)$ reduction from $\mathcal{B}_{n_i,s_i,w}$ to $\mathcal{B}_{n_{i+1},s'_{i+1},w}$, where we assume $s_i = c \log w$ with c being a universal constant. We choose $\varepsilon_i = 2^{-(\log w)/n_i^{1/3}}$, and consequently $n_{i+1} = n_i^{1/3}$, $K_i = O\left(2^{n_i^{1/3} \log n_i}\right) = o\left(\log^{1/2} w\right)$ and $s'_{i+1} = O(s_i + \log w)$. We iterate for l steps until we reach $n_l = O(1)$. One can see that $l \leq O(\log\log\log w)$.

As mentioned before, for every i, after the ith length reduction, we apply an alphabet reduction to retain a small alphabet. Each alphabet reduction is realized by a one-level NZ generator. Assume we have a ROBP f' with length $m = n_i$ and alphabet bit-length s'_i. The one-level NZ generator which, given a source X of $d = O(s' + \log w)$ bits and a sequence of independent seeds $Y_1, \ldots, Y_m$, each of $s_i = O(\log(d/\varepsilon))$ bits, outputs $(\textsc{Ext}(X, Y_1), \textsc{Ext}(X, Y_2), \ldots, \textsc{Ext}(X, Y_m))$ where each $\textsc{Ext}(X, Y_i)$

is of binary length s'_i. Now fixing $X = x$, the function $f'(\textsc{Ext}(x, y_1), \textsc{Ext}(x, y_2), \ldots, \textsc{Ext}(x, y_m))$ can be viewed as an ROBP on input $(y_1, \ldots, y_m)$ with alphabet bit-length $s_i = O\left(\log \frac{s'_i + \log w}{\varepsilon}\right)$. One can check $s_i \le c \log w$ as $\varepsilon = 1/\operatorname{poly} w$, where c is a universal constant.

The seed length of our WPRG is the sum of the randomness of all reductions, together with the true randomness used to fool the final constant length ROBPs. For length reductions the seed length is $\sum_{i=0}^{l} d_i \le O(\log(nw))$. The first alphabet reduction contributes $O\left(s + \frac{\log n \log(nw)}{\max\{1, \log\log w - \log\log n\}}\right)$. The remaining alphabet reductions contribute $O(l \cdot \log w)$. The true randomness for final constant length ROBPs is $O(\log w)$ since the alphabet bit-length is $O(\log w)$.

Finally for even smaller errors ($\varepsilon \ll 1/\operatorname{poly}(w)$), we apply a variant of Hoza's sampler [Hoz21] technique, which can also be viewed as an extra level of weighted pseudorandom reduction.

WPRGs for unbounded-width permutation ROBPs with a single accept node We start by reviewing the recent WPRG given by [CHL+23] in the view of weighted pseudorandom reductions. Their WPRG combines the INW generator and a new matrix iteration method. To fool a unbounded-width permutation ROBP of length n and alphabet $\{0,1\}^s$ within error ε, they first prepare a base INW generator $\mathsf{PRG} : \{0,1\}^d \to (\{0,1\}^s)^n$ with moderate sv-error[3] $\tau = 2^{-O(\log\log n\sqrt{\log(1/\varepsilon)})}$. Define $\mathsf{PRG}_{j\to i}$ as the truncation of PRG to the first $i - j$ output symbols. Then they construct the ε-WPRG (G, σ) in the following form:

$$G(j, x_1, \ldots, x_m) = \mathsf{PRG}_{0\to n_{j,1}}(x_1), \mathsf{PRG}_{n_{j,2}\to n_{j,1}}(x_2), \ldots, \mathsf{PRG}_{n_{j,m}\to n_{j,m-1}}(x_m),$$
$$\sigma(j) = \sigma_j \cdot K,$$

where $j \in [K]$, $K = 2^{O(m)}$, $m = O\left(\log n \cdot \frac{\log(1/\varepsilon)}{\log(1/\tau)}\right)$, for each j the sequence $n_{j,k}, k \in [m]$ of indices being such that $0 \le n_{j,1} < n_{j,2} < \ldots < n_{j,m} \le n$, and $\sigma_j \in \{-1, 0, 1\}$. The seed length of PRG is $d = s + O(\log n \cdot (\log\log n + \log(1/\tau)))$. Note that for each $j \in [K]$, one can view $f(G(j, x_1, \ldots, x_m))$ as a new permutation ROBP on input $x_1, x_2, \ldots, x_m$. [CHL+23] applies an ε/K-INW generator to fool such reduced permutation ROBPs, which cost $d + O(\log m \cdot (\log\log m + \log(K/\varepsilon))$ bits of randomness. This step introduces some double-logarithmic factors.

To address this problem, we introduce the following new strategy based on weighted pseudorandom reductions. Let $\mathcal{P}_{n,s}$ denote the class of length-n unbounded-width permutation ROBPs over alphabet $\{0,1\}^s$. Then (G, σ) can be viewed as a $(\log K, K, \varepsilon)$-reduction from $\mathcal{P}_{n,s}$ to $\mathcal{P}_{m,d}$. Let our target error be ε. We setup l reductions as the following. The i-th reduction is a $(d_i = \log K_i, K_i, \varepsilon_i)$-reduction from $\mathcal{P}_{n_{i-1},s_{i-1}}$ to $\mathcal{P}_{n_i,s_i}$. For the first reduction, we set $\tau_1 = 2^{-O\left(\sqrt{\log(1/\varepsilon)}\right)}$ to obtain a $(\log K_1, K_1, \varepsilon/2)$-reduction from $\mathcal{P}_{n,s}$ to $\mathcal{P}_{n_1,s_1}$. One can see $n_1 = O\left(\log n \cdot \sqrt{\log(1/\varepsilon)}\right)$, $s_1 = s + O\left(\log n \cdot \left(\log\log n + \sqrt{\log(1/\varepsilon)}\right)\right)$ and $K_1 = \exp\left(O\left(\log n \cdot \sqrt{\log(1/\varepsilon)}\right)\right)$. For the subsequent $l - 1$ reductions, we set $\varepsilon' = (\varepsilon/(2K_1))^2$ and $\tau_i = 2^{-\frac{C\log(1/\varepsilon')}{\log^2 n_{i-1}}}$ to obtain a $(\log K_i, K_i, \varepsilon')$-reduction from $\mathcal{P}_{n_{i-1},s_{i-1}}$ to $\mathcal{P}_{n_i,s_i}$, where $C > 0$ is a universal constant. One can deduce that $n_i = O\left(\log n_{i-1} \frac{\log(1/\varepsilon')}{\log(1/\tau_i)}\right) = \log^3 n_{i-1}$, $s_i = s_{i-1} + O(\log n_{i-1} \cdot (\log\log n_{i-1} + \log(1/\tau_i))) = s_{i-1} + O\left(\frac{\log(1/\varepsilon')}{\log n_{i-1}}\right)$ and $K_i = 2^{O(\log^3 n_i)}$. We do the iteration until $n_l = O(1)$. So $l = o(\log\log n)$.

We emphasize that the alphabet bit-length increment from s_i to s_{i+1} is not significant. In fact $s_l = s_1 + O(\log(1/\varepsilon')) \cdot \sum_{i=1}^{l-1} \frac{1}{\log n_i} = s_1 + O(\log(1/\varepsilon'))$. Also notice that the overall weight $K_1 K_2 K_3 \ldots K_l$. Here $K_2 K_3 \ldots K_l$ is bound by $2^{O(\log^3 n_1)}$ since n_i decreases quickly. This is negligible

[3] See Definition 4.5.

compared to ε/K_1, which means that $\varepsilon' = (\varepsilon/(2K_1))^2$ is small enough to deduce an ε-WPRG. Finally, we calculate the seed length. The overall random bits used in our WPRG have two parts, the bits for the final ROBP of length n_l and alphabet $\{0,1\}^{s_l}$ and the randomness used in the weighted reductions. For the first part, we need $s_l \cdot n_l$ bits. For the second part, we need $\sum_i d_i = \log K_1 + \sum_{i=2}^{l} \log K_i = \log K_1 + O(\log^3 n_1)$ bits. Therefore, one can deduce the overall random bits used in our WPRG is $s_l \cdot n_l + \sum_i d_i = O(s + \log n \cdot (\log\log n + \sqrt{\log(1/\varepsilon)}) + \log(1/\varepsilon))$.

Derandomization for short-wide regular ROBPs To attain our derandomization, notice that if we only want to derandomize permutation ROBPs with multiple accept nodes then we can directly use our WPRG for such permutation ROBPs. Recall that the seed length is $O\left(s + \log n\left(\log\log n + \sqrt{\log(w/\varepsilon)}\right) + \log(w/\varepsilon)\right)$. Hence it is $O(s + \log w)$ if $n = 2^{O(\sqrt{\log w})}$, $\varepsilon = 1/\operatorname{poly} w$. In fact, for regular ROBPs, we essentially do the same thing, except that we apply some white-box techniques such that the framework of our WPRG for permutation ROBPs can also work for regular ROBPs.

To derandomize regular ROBPs with binary alphabet, we show that there is a logspace algorithm that can transform a regular ROBP into a permutation ROBP which has the same expectation (though it probably computes a different function). Then we approximation this expectation with our WPRG for permutation ROBPs.

The hard case is to derandomize regular ROBPs with larger alphabets. Let f be a regular ROBP with length n, width w, alphabet bit-length s. We apply l reductions as the following. For every $p \in [l]$, let $f_{i_1,\ldots,i_{p-1}}$ be a reduced regular ROBP of length n_{p-1}, with its stochastic matrices $\mathbf{M}_{l\ldots r}$ from its layer l to layer r, $\forall l, r \in [n_p]$. $f_{i_1,\ldots,i_{p-1}}$ is f if $p-1=0$. For every $0 \le l < r \le n_{p-1}$ one can construct a τ_p-sv-approximation $\widetilde{\mathbf{M}}_{l\ldots r}$ of $\mathbf{M}_{l\ldots r}$ by the derandomizing square method of [AKM+20, HPV21a, CHL+23], if $f_{i_1,\ldots,i_{p-1}}$ has a two-way labeling. A crucial point is that $\widetilde{\mathbf{M}}_{l\ldots r}$ corresponds to the stochastic matrix of a regular bipartite graph $\widetilde{G}_{l\ldots r}$ also with a two-way labeling, as long as $f_{i_1,\ldots,i_{p-1}}$ has a two-way labeling. Recall that a two-way labeling for an ROBP, defined by [RVW00, RV05], requires that every edge has two labels, each for one direction, such that for every vertex all its out-going labels are distinct, and all its in-coming labels are also distinct. And a two-way labeling naturally supports a rotation technique which, whenever the pseudorandom walk arrives at a new node v from u through an edge e, switches the label of e from the out-going label of u to the incoming label of v. In this way, though the program is only regular, it can still be fooled by a pseudorandom walk from [AKM+20, CHL+23] with such rotations. Using an error reduction polynomial of [CHL+23], one can decompose $\mathbf{M}_{0\ldots n_{p-1}}$ into a signed sum:

$$\mathbf{M}_{0\ldots n_{p-1}} \approx \sum_{i_p \in [K_p]} \sigma_{i_p}^{(p)} \cdot \prod_{t=1}^{n_p} \widetilde{\mathbf{M}}_{m_{i_p,t-1}^{(p)}\ldots m_{i_p,t}^{(p)}},$$

where $0 = m_{i_p,0}^{(p)} < m_{i_p,1}^{(p)} < \cdots < m_{i_p,n_p}^{(p)} = n_{p-1}$ are indices partitioning $f_{i_1,\ldots,i_{p-1}}$ into n_p segments, K_p denotes the number of terms, and $\sigma_{i_p}^{(p)} \in \{-1,0,1\}$ are signs. Each product in the sum corresponds to a concatenation of regular bigraphs such that the product can also be viewed as a regular ROBP $f_{i_1,\ldots,i_p}$ of length n_p:

$$f_{i_1,\ldots,i_p} = \widetilde{G}_{m_{i_p,0}^{(p)}\ldots m_{i_p,1}^{(p)}} \circ \widetilde{G}_{m_{i_p,1}^{(p)}\ldots m_{i_p,2}^{(p)}} \circ \cdots \circ \widetilde{G}_{m_{i_p,n_p-1}^{(p)}\ldots m_{i_p,n_p}^{(p)}}.$$

$f_{i_1,\ldots,i_p}$ again has a two-way labeling since it is the concatenation of bigraphs with two-way labelings. If consider all the reductions, then finally we can approximate $\mathbb{E}[f]$ with a signed sum of expectations

of those reduced ROBPs:

$$\mathbb{E}[f] \approx \sum_{i_1 \in [K_1], \ldots, i_l \in [K_l]} \sigma_{i_1}^{(1)} \cdots \sigma_{i_l}^{(l)} \cdot \mathbb{E}[f_{i_1,\ldots,i_l}],$$

where $\forall p \in [l], \forall i_p, \sigma_{i_p}^{(p)}$ is a corresponding sign from the p-th reduction. It turns out all relevant parameters (e.g. the length and alphabet of the reduced ROBPs, the signs, the seed length) can be essentially the same as those parameters of our WPRG for permutation ROBPs. Also one can see that since f has a two-way labeling then $\forall p \in [l]$, every reduced program $f_{i_1,\ldots,i_p}$ has a two-way labeling by the property mentioned above. So finally for each $f_{i_1,\ldots,i_l}$, whose length is $n_l = O(1)$, one can apply a true random walk using rotations supported by the two-way labeling, and the expectation can be approximated by a weighted sum of the results of these random walks. By a more detailed analysis, we show that these rotations are again space efficient so that the overall space complexity is only linear of the seed length.

2 Preliminaries.

2.1 Some notations.

For convenience of description, we denote $\mathcal{B}_{(n,s,w)}$ as the class of ROBPs with length n, alphabet size 2^s and width w. We denote $\mathcal{P}_{(n,s)}$ as the class of permutation ROBPs with unbounded width and only one accept node, where n is the length s is the bit-length of the alphabet. We denote $\mathcal{R}_{(n,s,w)}$ as the class of regular ROBPs with length n, alphabet size 2^s and width w.

Given an ROBP f, we denote $f^{[i,j]} : (\{0,1\}^s)^{j-i} \to \mathbb{R}^{w \times w}$ as the matrix representing the transition function of f between layers V_i and V_j. Specifically, the entry $[f^{[i,j]}(x)]_{u,v} = 1$ if there exists a path from node u in V_i to node v in V_j that is labeled by the string $x = x_1 \ldots x_{j-i}$. Otherwise, $[f^{[i,j]}(x)]_{u,v} = 0$.

We use $\circ$ to denote the composition of functions, i.e. for any two functions f, g where any output of g can be an input of f, we have that $f \circ g(x) := f(g(x))$.

2.2 Weighted pseudorandom reductions.

Here we state the definition of weighted pseudorandom reduction.

Definition 2.1 (Weighted Pseudorandom Reduction)**.** *Let $\mathcal{F}_0$ be a class of functions $(\{0,1\}^{s_0})^{n_0} \to \mathbb{R}$. Let $\mathcal{F}_{simp}$ be a class of functions $(\{0,1\}^{s_1})^{n_1} \to \mathbb{R}$. A (d, K, ε)-weighted pseudorandom reduction from $\mathcal{F}_0$ to $\mathcal{F}_{simp}$ is a tuple (R, σ), in which $\mathsf{R} : (\{0,1\}^{s_1})^{n_1} \times \{0,1\}^d \to (\{0,1\}^{s_0})^{n_0}$ and $\sigma : \{0,1\}^d \to \mathbb{R}$. Furthermore, for every $f \in \mathcal{F}_0$, we have:*

$$\left| \mathbb{E}_U f(U) - \frac{1}{2^d} \sum_{i \in \{0,1\}^d} \sigma(i) \mathbb{E}_{U'}[f(\mathsf{R}(U', i))] \right| \leq \varepsilon,$$
$$\forall i, |\sigma(i)| \leq K,$$
$$\forall i, f(\mathsf{R}(\cdot, i)) \in \mathcal{F}_{simp}.$$

We call R the reduction function and σ the weight function. In some cases, we use the notation $\mathsf{R}_i(\cdot) := \mathsf{R}(\cdot, i)$ for convenience.

We emphasize that throughout this work, both $\mathcal{F}_0$ and $\mathcal{F}_{simp}$ are some classes of ROBPs, i.e. all our reductions keep this read-once property which is crucial in our proof.

We will frequently use compositions of reductions.

Lemma 2.2 (Composition Lemma)**.** *Let $\mathcal{F}_0, \mathcal{F}_1, \mathcal{F}_2$ be classes of boolean functions within $\{0,1\}^{s_0} \to \mathbb{R}, \{0,1\}^{s_1} \to \mathbb{R}, \{0,1\}^{s_2} \to \mathbb{R}$ respectively. Let $(\mathsf{R}^{(1)}, \sigma^{(1)})$ be an explicit $(d_1, K_1, \varepsilon_1)$-weighted pseudorandom reduction from $\mathcal{F}_0$ to $\mathcal{F}_1$ and $(\mathsf{R}^{(2)}, \sigma^{(2)})$ be an explicit $(d_2, K_2, \varepsilon_2)$-weighted pseudorandom reduction from $\mathcal{F}_1$ to $\mathcal{F}_2$. Then the composition $(\mathsf{R}^{(1)} \circ \mathsf{R}^{(2)}, \sigma^{(1)} \cdot \sigma^{(2)})$:*

$$(\mathsf{R}^{(1)} \circ \mathsf{R}^{(2)})_{(i_1,i_2)}(x) := \mathsf{R}^{(1)}_{i_1}(\mathsf{R}^{(2)}_{i_2}(x)),$$
$$\sigma^{(1)} \cdot \sigma^{(2)}(i_1, i_2) := \sigma^{(1)}(i_1) \cdot \sigma^{(2)}(i_2),$$

is an explicit $(d_1 + d_2, K_1K_2, \varepsilon_1 + K_1\varepsilon_2)$-weighted pseudorandom reduction from $\mathcal{F}_0$ to $\mathcal{F}_2$.

Proof. Let $\mathsf{R} = \mathsf{R}^{(1)} \circ \mathsf{R}^{(2)}$ and $\sigma = \sigma^{(1)} \cdot \sigma^{(2)}$. For any $f \in \mathcal{F}_0$, we have:

$$\begin{aligned}
&\left| \mathbb{E}_U f(U) - \frac{1}{2^{d_1+d_2}} \sum_{i_1 \in \{0,1\}^{d_1}, i_2 \in \{0,1\}^{d_2}} \sigma(i_1, i_2) \mathbb{E}_{U_{s_2}} \left[f(\mathsf{R}_{(i_1,i_2)}(U_{s_2})) \right] \right| \\
&\leq \left| \mathbb{E}_U f(U) - \frac{1}{2^{d_1}} \sum_{i_1 \in \{0,1\}^{d_1}} \sigma^{(1)}(i_1) \mathbb{E}_{U_{s_1}} \left[f(\mathsf{R}^{(1)}_{i_1}(U_{s_1})) \right] \right| \\
&+ \frac{1}{2^{d_1}} \sum_{i_1 \in \{0,1\}^{d_1}} |\sigma^{(1)}(i_1)| \left| \mathbb{E}_{U_{s_1}} \left[f(\mathsf{R}^{(1)}_{i_1}(U_{s_1})) \right] - \frac{1}{2^{d_2}} \sum_{i_2 \in \{0,1\}^{d_2}} \sigma^{(2)}(i_2) \mathbb{E}_{U_{s_2}} \left[f(\mathsf{R}^{(2)}_{i_2}(U_{s_2})) \right] \right| \\
&\leq \varepsilon_1 + K_1\varepsilon_2.
\end{aligned}$$

Furthermore, $|\sigma(i_1, i_2)| \leq |\sigma^{(1)}(i_1)| \cdot |\sigma^{(2)}(i_2)| \leq K_1K_2$. For any i_1, i_2, $x \mapsto f(\mathsf{R}^{(1)}_{i_1}(x)) \in \mathcal{F}_1$, sp we can apply $\mathsf{R}^{(2)}_{i_2}$ to this mapping, therefore $x \mapsto f(\mathsf{R}_{(i_1,i_2)}(x)) \in \mathcal{F}_2$. The seed length is $d_1 + d_2$. The function is explicit since both components are explicit. □

Lemma 2.3 (Composition Lemma for multiple reductions)**.** *For any positive integer k, let $\mathcal{F}_0, \mathcal{F}_1, \ldots, \mathcal{F}_k$ be classes of boolean functions within $\{0,1\}^{s_0} \to \mathbb{R}, \{0,1\}^{s_1} \to \mathbb{R}, \ldots, \{0,1\}^{s_k} \to \mathbb{R}$ respectively. Let $(\mathsf{R}^{(1)}, \sigma^{(1)}), \ldots, (\mathsf{R}^{(k)}, \sigma^{(k)})$ be explicit $(d_1, K_1, \varepsilon_1), \ldots, (d_k, K_k, \varepsilon_k)$-weighted pseudorandom reductions from $\mathcal{F}_0$ to $\mathcal{F}_1, \ldots, \mathcal{F}_k$ respectively. Then the composition $(\mathsf{R}^{(1)} \circ \ldots \circ \mathsf{R}^{(k)}, \sigma^{(1)} \cdot \ldots \cdot \sigma^{(k)})$:*

$$(\mathsf{R}^{(1)} \circ \ldots \circ \mathsf{R}^{(k)})_{(i_1,\ldots,i_k)}(x) = \mathsf{R}^{(1)}_{i_1}(\ldots \mathsf{R}^{(k)}_{i_k}(x) \ldots),$$
$$\sigma^{(1)} \cdot \ldots \cdot \sigma^{(k)}(i_1, \ldots, i_k) = \sigma^{(1)}(i_1) \cdot \ldots \cdot \sigma^{(k)}(i_k),$$

is an explicit $(\sum_{i=1}^k d_i, \prod_{i=1}^k K_i, \sum_{i=1}^k \left(\prod_{j=1}^{i-1} K_j \right) \varepsilon_i)$-weighted pseudorandom reduction from $\mathcal{F}_0$ to $\mathcal{F}_k$.

Proof. We prove this by induction on k. The base case $k = 2$ is shown by Lemma 2.2. Suppose the statement holds for $k-1$, then $(\mathsf{R}^{(1)} \circ \ldots \circ \mathsf{R}^{(k-1)}, \sigma^{(1)} \cdot \ldots \cdot \sigma^{(k-1)})$ is an explicit $(\sum_{i=1}^{k-1} d_i, \prod_{i=1}^{k-1} K_i, \sum_{i=1}^{k-1} \left(\prod_{j=1}^{i-1} K_j \right) \varepsilon_i)$-weighted pseudorandom reduction from $\mathcal{F}_0$ to $\mathcal{F}_{k-1}$. Then we can apply Lemma 2.2 on $(\mathsf{R}^{(1)} \circ \ldots \circ \mathsf{R}^{(k-1)}, \sigma^{(1)} \cdot \ldots \cdot \sigma^{(k-1)})$ and $(\mathsf{R}^{(k)}, \sigma^{(k)})$, which gives the desired result. □

3 WPRG for standard ROBPs.

In this section, we provide a new construction of Weighted Pseudorandom Generator(WPRG) for read-once branching programs(ROBPs). The main idea is iteratively approximating the expectation of a long ROBP by a weighted sum of the expectations of much shorter ROBPs. Let f be a long ROBP, our reduction will follow the paradigm:

$$\left|\mathbb{E}f(U) - \frac{1}{2^d}\sum_{i\in 2^d}\sigma^{(i)}\mathbb{E}\left[f(\mathsf{R}^{(i)}(U))\right]\right| < \varepsilon,$$

where $f\circ\mathsf{R}^{(i)}$ could be computed by a much shorter ROBP for any fixed i and f.

We mainly use two types of weighted pseudorandom reductions: alphabet reduction and length reduction.

Alphabet Reduction: In an alphabet reduction (R,σ), each $\mathsf{R}^{(i)}$ is a $(\{0,1\}^{s'})^n \to (\{0,1\}^s)^n$ function that maps n short symbols to n long symbols. For any $f\in\mathcal{B}_{(n,s,w)}$, $f\circ\mathsf{R}^{(i)}$ can be computed in $\mathcal{B}_{(n,s',w)}$, where the length and width are unchanged but the alphabet bit-length is reduced.

Length Reduction: In a length reduction (R,σ), each $\mathsf{R}^{(i)}$ is a $(\{0,1\}^{s'})^k \to (\{0,1\}^s)^n$ function that maps k long symbols to $n \gg k$ short symbols. For any $f\in\mathcal{B}_{(n,s,w)}$, $f\circ\mathsf{R}^{(i)}$ can be computed in $\mathcal{B}_{(k,s',w)}$, where the length is reduced from n to k but the alphabet bit-length is increased to s'.

For the rest of this section, we are going to show the following lemma. The general strategy is to apply the two reductions alternately to reduce the length and maintain the width and alphabet bit-length.

Lemma 3.1. *For every $\varepsilon \geq 1/\operatorname{poly}(w)$, there exists an explicit ε-WPRG for $\mathcal{B}_{(n,s,w)}$ with seed length*

$$O\left(s + \frac{\log n\log(nw)}{\max\{1,\log\log w - \log\log n\}} + \log w\left(\log\log\min\{n,\log w\} - \log\log\max\left\{\frac{\log w}{\log\frac{n}{\varepsilon}},2\right\}\right)\right)$$

and weight $(8n)^{2\sqrt{\frac{\log w}{\log n}}}$.

We note that for smaller target errors, one can further use the sampler trick [Hoz21] with Lemma 3.1 to reduce the error from $1/\operatorname{poly}(w)$ to an arbitrary $\varepsilon > 0$, attaining the following theorem.

Theorem 3.2. [4] *For all integer n, s, w, there exists an explicit construction of a ε-WPRG for $\mathcal{B}_{(n,s,w)}$ with seed length*

$$s' + O\left(\frac{\log n\log(nw)}{\max\{1,\log\log w - \log\log n\}} + \log w\left(\log\log\min\{n,\log w\} - \log\log\max\left\{\frac{\log w}{\log\frac{n}{\varepsilon}},2\right\}\right) + \log(\frac{1}{\varepsilon})\right),$$

and weight $\operatorname{poly}(nw/\varepsilon)$. *Here* $s' = O\left(s\cdot\frac{\log(\frac{n}{\varepsilon})}{\log(nw)}\right)$.

Note that our main theorem Theorem 1.5 is a direct corollary of Theorem 3.2.

[4] Theorem 3.2 in a prior version of this paper stated that the seed length dependence on s is $O(s)$, but the proof here only supports the current version. We thank Ben Chen, Gil Cohen, Dean Doron, Yuval Khaskelberg, and Amnon Ta-Shma for pointing out this error. We give a revised construction in Appendix F to attain the optimal dependence on s.

3.1 Alphabet reduction.

The alphabet reduction reduces the alphabet bit-length s to a much smaller $O(\log w)$ and keeps the length n and width w unchanged. The alphabet reduction is achieved by using the Nisan-Zuckerman (NZ) PRG, we start with the construction of the PRG and its main ingredient, extractors.

Definition 3.3 (Extractor)**.** *A (k, ε)-extractor is a function $\textsc{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^k$ such that for any $X \in \{0,1\}^n$ with $H(X) \geq k$, the distribution of $\textsc{Ext}(X, U_d)$ is ε-close to the uniform distribution on $\{0,1\}^k$. Here $H(X) = \max_{x \in \{0,1\}^n} -\log \Pr[X = x]$ is the min-entropy of X, and U_d is the uniform distribution on $\{0,1\}^d$.*

Theorem 3.4 (Explicit Extractor from [GUV09])**.** *There exists a universal constant C_{GUV} that for all positive integer s and positive real ε, there is an explicit construction of a $(2s, \varepsilon/3)$-extractor $\textsc{Ext} : \{0,1\}^{3s} \times \{0,1\}^d \to \{0,1\}^s$ with seed length $d = C_{GUV} \cdot \log \frac{ns}{\varepsilon}$.*

We mainly use the one-level version of NZ PRG.

Lemma 3.5 (One-level NZ PRG[NZ96] with a large alphabet)**.** *Let $s \geq \log w$. Assume there exists a $(2s, \frac{\varepsilon}{3n})$-extractor $\textsc{Ext} : \{0,1\}^{3s} \times \{0,1\}^d \to \{0,1\}^s$. Let X and $Y_1, \ldots, Y_n$ be independent uniform random variables. Then the following construction*

$$\mathsf{NZ}(X, Y) = \textsc{Ext}(X, Y_1), \ldots, \textsc{Ext}(X, Y_n)$$

fools any $f \in \mathcal{B}_{(n,s,w)}$ with error at most ε.

The original proof of [NZ96] only considers the binary case. So we include a proof in Appendix A for completeness.

Now we construct the alphabet reduction. Given a ROBP $f \in \mathcal{B}_{(n,s,w)}$, we use $\mathbb{E} f(\mathsf{NZ}(X, Y))$ to approximate $\mathbb{E} f$. By fixing X, the computation of $f(\mathsf{NZ}(X, Y))$ can be done by a program of much smaller alphabet, that gives the following reduction.

Lemma 3.6 (Alphabet Reduction)**.** *For all positive integers w, n, s with $s \geq \log w$, there exists an explicit $(3s, 1, \varepsilon)$-weighted pseudorandom reduction from $\mathcal{B}_{(n,s,w)}$ to $\mathcal{B}_{(n, C_{GUV} \log \frac{ns}{\varepsilon}, w)}$.*

Proof. Let $\textsc{Ext} : \{0,1\}^{3s} \times \{0,1\}^d \to \{0,1\}^s$ be a $(2s, \frac{\varepsilon}{3n})$-extractor with seed length $d = C_{GUV} \cdot \log \frac{ns}{\varepsilon}$ from Theorem 3.4. Let NZ be defined as in Lemma 3.5. We define the reduction (R, σ):

$$\mathsf{R}_x(y) := \mathsf{NZ}(x, y)$$
$$\sigma_x := 1.$$

Now consider that we fix x and let y_i's be free. We need to prove that the reduction (R, σ) is a $(3s, 1, \varepsilon)$-weighted pseudorandom reduction.

Let $f \in \mathcal{B}_{(n,s,w)}$, then by Lemma 3.5,

$$|\mathbb{E} f - 2^{-3s} \sum_{x \in \{0,1\}^{3s}} \mathbb{E} f(\mathsf{NZ}(x, *))| = |\mathbb{E} f - \mathbb{E} f(\mathsf{NZ}(X, Y))| \leq \varepsilon.$$

Therefore, (R, σ) approximates f with error at most ε.

To prove that $f(\mathsf{NZ}(x, *)) \in \mathcal{B}_{(n,d,w)}$ for all $f \in \mathcal{B}_{(n,s,w)}$ and $x \in \{0,1\}^{3s}$, $y \in \{0,1\}^d$. We construct the ROBP $g_x := f(\mathsf{NZ}(x, *))$ as follows:

Let $V_0, \ldots, V_n$ be the layers of f, each consisting of w nodes. The ROBP g_x is also defined on $V_0, \ldots, V_n$. The labeled edge set of g_x is defined as follows:

$$E_{g_x} = \{(u, v, y) : u \in V_{i-1}, v \in V_i, y \in \{0,1\}^d, \exists \text{ an edge from } u \text{ to } v \text{ labeled by } \textsc{Ext}(x, y) \text{ in } f\}$$

The start node and accept nodes of G are the same as those in f. Then g_x is in $\mathcal{B}_{(n,d,w)}$ and $g_x(y_1, \ldots, y_n) = f(\textsc{Ext}(x, y_1), \ldots, \textsc{Ext}(x, y_n)) = f(\mathsf{NZ}(x, y))$. □

3.2 Length reduction framework from Richardson Iteration.

A length reduction reduces the length n to a much smaller k but may increase the alphabet bit-length. In this subsection, we construct a length reduction using the error reduction given by [AKM+20, CDR+21, PV21].

Lemma 3.7 (framework of the length reduction)**.** *For any positive integers n, s, w, positive odd integer k and positive real ε, assume there exists an explicit $\frac{\varepsilon}{(n+1)^2}$-PRG for $\mathcal{B}_{(n,s,w)}$ with seed length $d = d(n, s, w)$. Then there exists an explicit $(\log K, K, \varepsilon^{\frac{k+1}{2}} \cdot (n+1))$-weighted pseudorandom reduction from $\mathcal{B}_{(n,s,w)}$ to $\mathcal{B}_{(k,d,w)}$, where $K = (8n)^{k+1}$.*

To prove Lemma 3.7, we need the following error reduction.

Theorem 3.8 (Error Reduction based on Richardson Iteration [AKM+20][CDR+21][PV21])**.** *Let $\{A_i\}_{i=1}^n \subset \mathbb{R}^{w\times w}$ be a sequence of matrices. Let $\{B_{i,j}\}_{i,j=0}^n \subset \mathbb{R}^{w\times w}$ be a family of matrices such that for every $i+1<j$, $\|B_{i,j} - A_{i+1}\dots A_j\| \le \varepsilon/(2(n+1))$ for some submultiplicative norm $\|\cdot\|$, $\|A_i\| \le 1$ for all i and also $B_{i-1,i} = A_i$ for all i. Then for any odd $k \in \mathbb{N}$, there exists a $K = (8n)^{k+1}$, a set of indices $\{n_{i,j}\}_{i\in[K],j\in[k]}$ with $0 \le n_{i,1} \le \dots \le n_{i,k} = n$, and signs $\sigma_i \in \{-1,0,1\}, i \in [K]$ such that (We set $B_{i,i} = I$ for all i):*

$$\left\| A - \sum_{i\in[K]} \sigma_i \cdot B_{0,n_{i,1}} B_{n_{i,1},n_{i,2}} \dots B_{n_{i,k-1},n_{i,k}} \right\| \le \varepsilon^{(k+1)/2} \cdot (n+1).$$

A proof of Theorem 3.8 is in Appendix B.

Proof of Lemma 3.7. Let $k, K, n_{i,j}, \sigma_i'$ be as in Theorem 3.8, Let PRG be a $\varepsilon/(2(n+1))$-PRG for $\mathcal{B}_{(n,s,w)}$ with seed length d in the assumption. We define the reduction (R, σ) as follows:

$$\mathsf{R}_i(x_1, \dots, x_k) = \mathsf{PRG}(x_1)_{n_{i,1}}, \mathsf{PRG}(x_2)_{n_{i,2}-n_{i,1}}, \dots, \mathsf{PRG}(x_k)_{n_{i,k}-n_{i,k-1}}$$
$$\sigma_i = \sigma_i' K.$$

Here $\mathsf{PRG}(x_i)_c$ denotes the first c symbols of $\mathsf{PRG}(x_i)$ and each symbol is in $\{0,1\}^s$.

We will show that (R, σ) is a $(\log K, K, \varepsilon^{\frac{k+1}{2}} \cdot (n+1))$-weighted pseudorandom reduction from $\mathcal{B}_{(n,s,w)}$ to $\mathcal{B}_{(k,d,w)}$.

Let $f \in \mathcal{B}_{(n,s,w)}$. Recall that $f^{[i,j]}(x)$ denotes the transition matrix of f from layer i to layer j with input $x = (x_1, \dots, x_{j-i})$. Define $A_i = \mathbb{E}_{x\in\{0,1\}^s} f^{[i-1,i]}[x]$ and $B_{i,j} = \mathbb{E}_{x\in\{0,1\}^d} f^{[i,j]}[\mathsf{PRG}(x)_{j-i}]$. By the definition of the PRG, we have $\|B_{i,j} - A_{i+1}\dots A_j\|_\infty \le \varepsilon/(n+1)$.

Therefore, by Theorem 3.8,

$$\begin{aligned}
&\left|\mathbb{E}f - \frac{1}{K}\sum_{i=1}^{K}\sigma(i)\mathbb{E}_X f(\mathsf{R}_i(X))\right| \\
=&\left\|\mathbb{E}f^{[0,n]} - \sum_{i=1}^{K}\sigma_i \mathbb{E}f^{[0,n]}(\mathsf{PRG}(X_1)_{n_{i,1}}, \mathsf{PRG}(X_2)_{n_{i,2}-n_{i,1}}, \ldots, \mathsf{PRG}(X_k)_{n_{i,k}-n_{i,k-1}})\right\|_\infty \\
\leq&\left\|\mathbb{E}f^{[0,n]} - \sum_{i=1}^{K}\sigma_i \mathbb{E}f^{[0,n_{i,1}]}(\mathsf{PRG}(X_1)_{n_{i,1}})\mathbb{E}f^{[n_{i,1},n_{i,2}]}(\mathsf{PRG}(X_2)_{n_{i,2}-n_{i,1}})\cdots\right. \\
&\qquad\left.\cdots\mathbb{E}f^{[n_{i,k-1},n_{i,k}]}(\mathsf{PRG}(X_k)_{n_{i,k}-n_{i,k-1}})\right\|_\infty \\
=&\left\|A_1\ldots A_n - \sum_{i=1}^{K}\sigma_i B_{0,n_{i,1}}B_{n_{i,1},n_{i,2}}\ldots B_{n_{i,k-1},n_{i,k}}\right\|_\infty \\
\leq&\varepsilon^{\frac{k+1}{2}}\cdot(n+1).
\end{aligned}$$

The weight is $K = (8n)^{k+1}$ by Theorem 3.8, and so the seed length is $\log K = O(k\log n)$.

Finally, we need to show that $f \circ \mathsf{R}_i \in \mathcal{B}_{(k,d,w)}$ for all $f \in \mathcal{B}_{(n,s,w)}$ and $i \in [K]$. We construct the ROBP $g := f \circ \mathsf{R}_i$ as follows:

Let $V_0, \ldots, V_n$ be the layers of f, each consisting of w nodes. The ROBP g is defined on $V_0, V_{n_{i,1}}, \ldots, V_{n_{i,k}}$. The labelled edge set of g is defined as follows:

$$\begin{aligned}
E_g = \{&(u,v,x) : j \in [n], u \in V_{n_{i,j-1}}, v \in V_{n_{i,j}}, x \in \{0,1\}^d, \\
&\text{there exists a path from } u \text{ to } v \text{ in } f \text{ through } \mathsf{PRG}(x)_{n_{i,j}-n_{i,j-1}}\ \}
\end{aligned}$$

The start node and accept nodes of g are the same as that in f. Then g is in $\mathcal{B}_{(k,d,w)}$ and $g(x_1,\ldots,x_n) = f(\mathsf{PRG}(x_1)_{n_{i,1}}, \mathsf{PRG}(x_2)_{n_{i,2}-n_{i,1}}, \ldots, \mathsf{PRG}(x_k)_{n_{i,k}-n_{i,k-1}}) = f(\mathsf{R}_i(x_1,\ldots,x_k))$. □

3.3 Length reduction instantiated by Armoni's PRG.

We use Armoni's PRG for $\mathcal{B}_{(n,s,w)}$ to instantiate the framework of Section 3.2.

Theorem 3.9 (Armoni's PRG[Arm98, KNW08])**.** *For all positive integer n, s, w and positive real ε, there exists an explicit construction of a ε-PRG for $\mathcal{B}_{(n,s,w)}$ with seed length $O(s + \frac{\log n \log(nw/\varepsilon)}{\log\log w - \log\log(n/\varepsilon)})$. Here the big-O of the seed length hides a universal constant.*

We have two instantiations that have different parameters.

Lemma 3.10 (Length reduction 1)**.** *For any constant $a, c, C \in \mathbb{N}$, for all integer n, w, if $n \leq \log^c w$, then there exists an explicit $(\log K, K, 1/w^a)$-weighted pseudorandom reduction from $\mathcal{B}_{(n,C\log w,w)}$ to $\mathcal{B}_{(n^{1/c},C'\log w,w)}$, where $K \leq (8n)^{n^{1/c}+1}$, C' is a constant depending on c, C, a.*

Proof. Fix c, C to be constants. We use Armoni's PRG[5] for $\mathcal{B}_{(n,C\log w,w)}$ with error $\varepsilon_0 = \frac{2^{-\frac{a\log w}{n^{1/c}}}}{(n+1)^2}$. By Theorem 3.9, the seed length is $O\left(C\log w + \frac{a\log w\log\log w}{\log\log w/c + O(1)}\right) = C'\log w$, where C' depends on c, C, a. Then we can set $k = 2n^{1/c} \geq \frac{a\log w + \log n}{1/2\cdot\log(1/\varepsilon_0)}$ in Lemma 3.7 and invoke it to attain this lemma. □

[5]Readers can notice that for this setting, the Armoni's PRG that we use, should automatically become the Nisan-Zuckerman PRG.

Lemma 3.11 (Length reduction 2)**.** *For any constant $a \in \mathbb{N}$, for all integer n, s, w, there exists an explicit $(\log K, K, 1/w^a)$-weighted pseudorandom reduction from $\mathcal{B}_{(n,s,w)}$ to $\mathcal{B}_{\left(\log^{\frac{1}{2}} w, O\left(s+\frac{\log n \log(nw)}{\log\log w - \log\log n}\right), w\right)}$, where $K = (8n)^{\sqrt{\frac{\log w}{\log n}}+1}$.*

Proof. We use Armoni's PRG for $\mathcal{B}_{(n,s,w)}$ with error $\varepsilon_0 = \frac{2^{-2a\sqrt{\log n \log w}}}{(n+1)}$. By Theorem 3.9, the seed length is $O\left(s + \frac{\log n \log(nw)}{\log\log w - \log\log n}\right)$. Then we can set $k = \frac{a \log w + \log n}{1/2 \cdot \log(1/\varepsilon_0)} \leq \sqrt{\frac{\log w}{\log n}} \leq \log^{1/2} w$ in Lemma 3.7 and invoke it to attain this lemma. □

3.4 Combining the reductions.

We combine the reductions from Lemma 3.6, Lemma 3.10 and Lemma 3.11 to give the following reduction.

Lemma 3.12 (Main reduction)**.** *For all integer n, s, w, for all $\varepsilon \geq 1/\operatorname{poly}(w)$, there exists an explicit $\left(O\left(s + \frac{\log n \log(nw)}{\log\log w - \log\log n} + \log w(\log\log \min\{n, \log w\} - \log\log \max\left\{\frac{\log w}{\log \frac{n}{\varepsilon}}, 2\right\})\right), (8n)^{2\sqrt{\frac{\log w}{\log n}}}, \varepsilon\right)$-weighted pseudorandom reduction from $\mathcal{B}_{(n,s,w)}$ to $\mathcal{B}_{(O(1),O(\log w),w)}$.*

Proof. We set up the reductions as the following, which is also illustrated in Figure 1.

We have the following reductions:

Let $(\mathsf{R}^{(1)}, \sigma^{(1)})$ be a $(d_1, K_1, \varepsilon/4)$-weighted pseudorandom reduction from $\mathcal{B}_{(n,s,w)}$ to $\mathcal{B}_{(n_1,s_1,w)}$ given by Lemma 3.11, where

- $d_1 = O(\log w)$,
- $n_1 = \min\{n, \log^{1/2} w\}$,
- $s_1 = O(s + \frac{\log n \log(nw)}{\log\log w - \log\log n})$,
- $K_1 = (8n)^{\sqrt{\frac{\log w}{\log n}}+1}$.

Let $(\mathsf{R}^{(2)}, \sigma^{(2)})$ be a $(s_2, K_2, \varepsilon/(4K_1))$-weighted pseudorandom reduction from $\mathcal{B}_{(n_1,s_1,w)}$ to $\mathcal{B}_{(n_2,s_2,w)}$ given by Lemma 3.6, where

- $d_2 = O(s + \frac{\log n \log(nw)}{\log\log w - \log\log n})$,
- $n_2 = n_1$,
- $s_2 \leq 2C_{GUV} \log(1/\varepsilon)$,
- $K_2 = 1$.

Using Lemma 2.2, $(\mathsf{R}^{(1)} \circ \mathsf{R}^{(2)}, \sigma^{(1)} \cdot \sigma^{(2)})$ is a $(d_1 + d_2, K_1, \varepsilon/2)$ weighted pseudorandom reduction from $\mathcal{B}_{(n,s,w)}$ to $\mathcal{B}_{(\log^{1/2} w, 2C_{GUV}\log(1/\varepsilon), w)}$. We set $\varepsilon' = (\varepsilon/K_1)^2$ and continue the reduction with the following parameters:

$n_3 = \log^{1/2} w, n_{i+1} = n_i^{1/3}$ for $i = 3, \ldots, l-1$. $n_l = \frac{\log w}{\log 1/\varepsilon'} + 2 = \frac{\log w}{\log 1/\varepsilon + \sqrt{\log n \log w}} + 2$.

It is clear that

$$l = \log\log \min\{n, \log w\} - \log\log\left(\frac{\log w}{\log n/\varepsilon} + 2\right) + O(1).$$

For $i = 3, \ldots, l-1$, let $(\mathsf{R}^{(i)}, \sigma^{(i)})$ be a (d_i, K_i, ε')-weighted pseudorandom reduction from $\mathcal{B}_{(n_i, 2C_{GUV}\log(1/\varepsilon'), w)}$ to $\mathcal{B}_{(n_{i+1}, C_{large}\log w, w)}$ given by Lemma 3.10, setting constants $c = 3$, $C = \frac{2C_{GUV}\log(1/\varepsilon')}{\log w}$, $a = \log_w(1/\varepsilon')$ in Lemma 3.10, where

- $d_i = \log K_i = O(n_i^{1/2}\log n_i)$,
- $K_i = \exp(O(n_i^{1/2}\log n_i))$.

Notice that since $n_3 = \log^{1/2} w$ and n_i is monotonously decreasing, the condition $n_i \le \log^3 w$ is always satisfied, so the condition in Lemma 3.10 is always met.

For $i = 3, \ldots, l-2$, let $(\widehat{\mathsf{R}}^{(i)}, \widehat{\sigma}^{(i)})$ be a $(\widehat{d}_i, \widehat{K}_i, \varepsilon')$-weighted pseudorandom reduction from $\mathcal{B}_{(n_{i+1}, C_{large}\log w, w)}$ to $\mathcal{B}_{(n_{i+1}, 2C_{GUV}\log(1/\varepsilon'), w)}$ given by Lemma 3.6, where

- $\widehat{d}_i = O(\log w)$,
- $\widehat{K}_i = 1$.

We invoke Lemma 2.3 to compose the $2l-7$ reductions, which gives $(\mathsf{R}^{(3)} \circ \widehat{\mathsf{R}}^{(3)} \circ \mathsf{R}^{(4)} \circ \widehat{\mathsf{R}}^{(4)} \circ \ldots \circ \mathsf{R}^{(l-1)}, \sigma^{(3)} \cdot \widehat{\sigma}^{(3)} \cdot \sigma^{(4)} \cdot \widehat{\sigma}^{(4)} \cdot \cdots \cdot \sigma^{(l-1)})$. Also by Lemma 2.3, we know the reduction is a $(d^*, K^*, \varepsilon^*)$-weighted pseudorandom reduction from $\mathcal{B}_{(n,s,w)}$ to $\mathcal{B}_{(n_l, 2C_{GUV}\log(1/\varepsilon'), w)}$. The latter class is contained in $\mathcal{B}_{(O(1), C_{GUV}\log w, w)}$.The parameters $(d^*, K^*, \varepsilon^*)$ is shown as following:

- $d^* = \sum_{i=3}^{l-1} d_i = O(l\log w)$,
- $K^* = \prod_{i=3}^{l-1} K_i \cdot \widehat{K}_i = \exp(\sum_{i=3}^{l-1} n_i^{1/2}\log n_i) \le \exp(l\log^{1/3} w) < \exp(\log^{1/2} w)$,
- $\varepsilon^* = \sum_{i=3}^{l-1} \varepsilon' \cdot \prod_{j=3}^{i-1} K_j \cdot \widehat{K}_j \le l \cdot \varepsilon \cdot K < \varepsilon' \cdot \exp(\log^{1/2} w) < \varepsilon/(2K_1)$.

We invoke Lemma 2.2 again and compose the $(d_1 + d_2, K_1, \varepsilon/2)$-weighted pseudorandom reduction from $\mathcal{B}_{(n,s,w)}$ to $\mathcal{B}_{(\log^{1/3} w, 2C_{GUV}\log(1/\varepsilon), w)}$ with the $(d^*, K^*, \varepsilon^*)$-weighted pseudorandom reduction from $\mathcal{B}_{(\log^{1/3} w, 2C_{GUV}\log(1/\varepsilon), w)}$ to $\mathcal{B}_{(O(1), C_{GUV}\log w, w)}$. We get the desired reduction with the following parameters:

- Seed length: $s_1 + s_2 + s^* = O(s + \frac{\log n \log(nw)}{\log\log w - \log\log n} + \log w(\log\log\log w - \log\log\frac{\log w}{\log(n/\varepsilon)})$,
- Weight: $K_1 \cdot K^* \le (8n)^{2\sqrt{\frac{\log w}{\log n}}}$,
- Error: $\varepsilon/2 + K_1 \cdot \varepsilon/(2K_1) = \varepsilon$.

The lemma follows. □

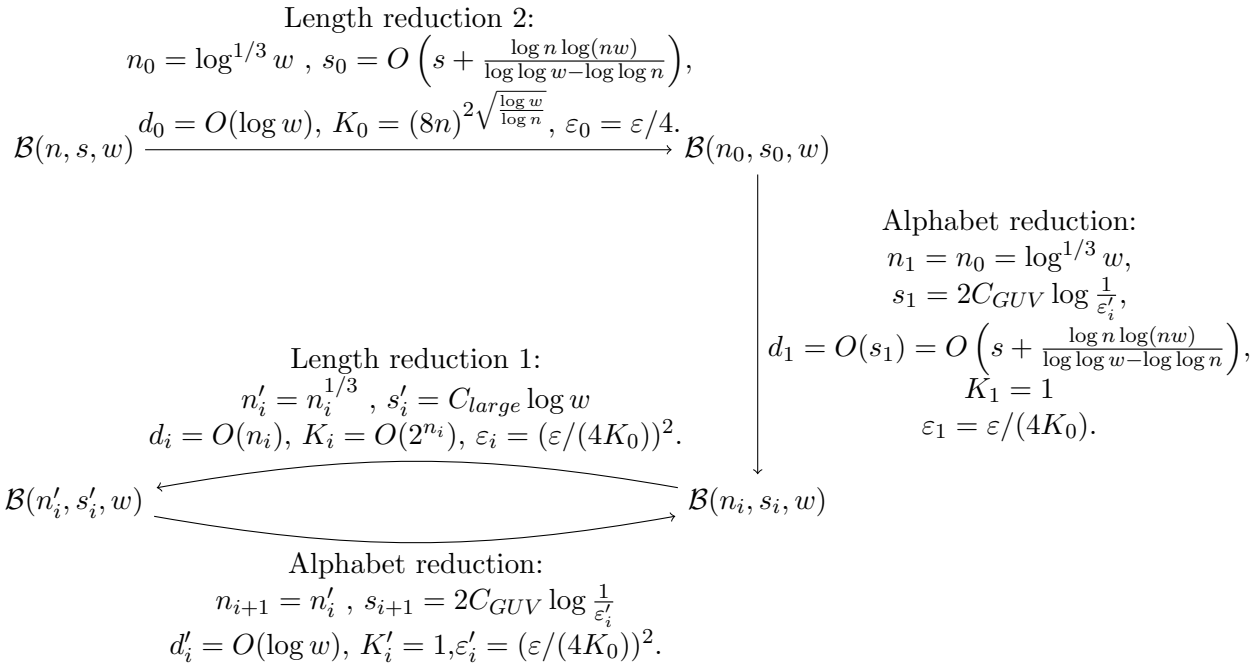


Figure 1: The recursion we use to iteratively reduce the ROBP in the construction of the WPRG Lemma 3.1. The arrows represent the reduction from an ROBP to simpler ROBPs.

The reduction naturally gives a WPRG, which is the PRG in Lemma 3.1

Lemma 3.13 (Lemma 3.1 restated)**.** *For all $C > 0, \varepsilon > 1/\operatorname{poly}(w)$, there exists an explicit ε-WPRG for $\mathcal{B}_{(n, C\log w, w)}$ with seed length $O\left(\frac{\log n \log(nw)}{\log\log w - \log\log n} + \log w \left(\log\log\log w - \log\log \max\left\{\frac{\log w}{\log \frac{n}{\varepsilon}}, 2\right\}\right)\right)$ and weight $(8n)^{2\sqrt{\frac{\log w}{\log n}}}$.*

Proof. Let (R, σ) be the reduction from Lemma 3.12. We construct the WPRG (G, w) with the same weight function w and G is defined as $G(x, i) = \mathsf{R}(x, i)$. By Lemma 3.12, to fool the original ROBP we only need to provide randomness for the seed of (R, σ) and also provide a random string x which can fool an arbitrary ROBP in $\mathcal{B}_{(O(1), C_{GUV}\log w, w)}$. We use true randomness for x and this only takes $O(\log w)$ random bits. Also the randomness used by (R, σ) is $O\left(\frac{\log n \log(nw)}{\log\log w - \log\log n} + \log w(\log\log\log w - \log\log \frac{\log w}{\log(n/\varepsilon)})\right)$. So the total seed length is as stated. The weight also directly follows from Lemma 3.12. □

3.5 Reducing error beyond $1/\operatorname{poly}(w)$.

To further reduce the error from $1/\operatorname{poly}(w)$ an arbitrary small ε, we use the sampler trick from [Hoz21]. The original technique is applied on PRGs to get a WPRG with a smaller error. Here we also use the technique but we apply it on a WPRG and get a WPRG with a smaller error.

Recall the definition of averaging samplers:

Definition 3.14 (Averaging Sampler)**.** *A (α,γ)-averaging sampler is a function* $\mathrm{Samp}:\{0,1\}^r\times\{0,1\}^p\to\{0,1\}^q$ *such that for every function* $f:\{0,1\}^q\to[-1,1]$*, it holds that:*

$$\Pr_{x\in\{0,1\}^r}\left[\left|2^{-p}\sum_{y\in\{0,1\}^p}f(\mathrm{Samp}(x,y))-\mathbb{E}[f]\right|\geq\alpha\right]\leq\gamma.$$

Lemma 3.15 ([CL20] Appendix B, [Gol11, RVW00])**.** *For all $\alpha>0,\gamma>0$, there exists an explicit (α,γ)-averaging sampler with seed length $r=q+O(\log(1/\alpha)+\log(1/\gamma))$ and $p=O(\log(1/\alpha)+\log\log(1/\gamma))$.*

We now give the construction of a WPRG for class $\mathcal{B}_{(n,s,w)}$ with small error ε. First, let $(G_0:\{0,1\}^d\to(\{0,1\}^s)^n,\sigma_0:\{0,1\}^d\to\mathbb{R})$ be a WPRG for $\mathcal{B}_{(n,s,w)}$, error $1/(2w\cdot(n+1)^2)$ and weight W, which is constructed from Lemma 3.1. We then set the parameters as follows:

- $k=\frac{\log(n/\varepsilon)}{\log(nw)}$,
- $\alpha=1/(W\cdot w^2\cdot(n+1)^2)$,
- $\gamma=\varepsilon/(2(2n)^k\cdot(W)^{k+1}\cdot w^2)$.

Let $\mathrm{Samp}:\{0,1\}^r\times\{0,1\}^p\to\{0,1\}^d$ be a (α,γ)-averaging sampler via Lemma 3.15. Let $K,n_{i,j},\sigma_i$ be as the corresponding parameters in Lemma 3.8, which depend on our k,n here. Our WPRG is constructed as follows:

$$\begin{cases} G(x,y_1,\ldots,y_k,i)=Z(x,y_1,i),Z(x,y_2,i),\ldots,Z(x,y_k,i),\\ \forall i\in[K],j\in[k],Z(x,y_j,i)=\begin{cases} G_0(\mathrm{Samp}(x,y_j))_{n_{i,j}-n_{i,j-1}} & if\ n_{i,j}-n_{i,j-1}>1,\\ (y_j)_s, & if\ n_{i,j}-n_{i,j-1}=1,\end{cases}\\ \sigma(x,y_1,\ldots,y_k,i)=\sigma_iK\cdot\prod_{j=1}^k\sigma_0(\mathrm{Samp}(x,y_j)),\end{cases}$$

where $x\in\{0,1\}^r$, each $y_j\in\{0,1\}^p\cup\{0,1\}^s$. $(y_j)_s$ is y_j truncated to the first s bits. Notice that in the Richardson-iteration error-reduction polynomial, each monomial alternates between approximate multi-step transition matrices and exact one-step transition matrices. The approximated transitions are attained by using the outputs of G_0, and the exact transitions are attained by using those $(y_j)_s$.

Now we show that the construction serves as a good WPRG for $\mathcal{B}_{(n,s,w)}$ with error ε.

Lemma 3.16. *For all n,s,w, assume there exists a W-bounded $1/(2w\cdot(n+1)^2)$-WPRG (G_0,σ_0) for $\mathcal{B}_{(n,s,w)}$ with seed length d, and a (α,γ)-averaging sampler* $\mathrm{Samp}:\{0,1\}^r\times\{0,1\}^p\to\{0,1\}^d$ *with α,γ as defined above. Then there exists a ε-WPRG for $\mathcal{B}_{(n,s,w)}$ with seed length $r+kp+ks$ and weight $\sigma^{\log(n/\varepsilon)/\log(nw)}\cdot\mathrm{poly}(nw/\varepsilon)$.*

The proof of Lemma 3.16 is deferred to Appendix C.

Combining the lemma with Lemma 3.1, we prove the main theorem of the section:

Theorem 3.17. *For all integer n,s,w, there exists an explicit construction of a ε-WPRG for $\mathcal{B}_{(n,s,w)}$ with seed length*

$$O\left(s\cdot\frac{\log(n/\varepsilon)}{\log(nw)}+\frac{\log n\log(nw)}{\log\log w-\log\log n}+\log w(\log\log\min\{n,\log w\}-\log\log\frac{\log w}{\log(n/\varepsilon)})+\log(1/\varepsilon)\right)$$

and weight $\mathrm{poly}(nw/\varepsilon)$.

Proof. Let (G_0, σ_0) be a $1/(2w \cdot (n+1)^2)$-WPRG for $\mathcal{B}_{(n,s,w)}$ with seed length

$$d = O\left(s + \frac{\log n \log(nw)}{\log\log w - \log\log n} + \log w(\log\log \min\{n, \log w\} - \log\log \frac{\log w}{\log(n/\varepsilon)})\right)$$

and weight $K = \text{poly}(nw/\varepsilon)$ given by Lemma 3.1.

Let Samp be a $(1/(2W \cdot w^2 \cdot (n+1)^2), \varepsilon/(2(2n)^k \cdot \sigma^k))$-averaging sampler with seed length $r = d + O(\log(nw/\varepsilon))$ and $p = O(\log(nw) + \log\log(1/\varepsilon))$ given by Lemma 3.15.

By Lemma 3.16, we get a ε-WPRG for $\mathcal{B}_{(n,s,w)}$ with seed length

$$\begin{aligned}&r + (2k+1)p + (2k+1)s\\ =&O\left(s \cdot \frac{\log(n/\varepsilon)}{\log(nw)} + \frac{\log n \log(nw)}{\log\log w - \log\log n} + \log w(\log\log \min\{n, \log w\} - \log\log \frac{\log w}{\log(n/\varepsilon)}) + \log(1/\varepsilon)\right)\end{aligned}$$

and weight $\text{poly}(nw/\varepsilon)$.

□

3.6 Application: a WPRG for regular branching program with a better seed length.

One application of our WPRG is to improve the WPRG for regular ROBPs by Chen, Hoza, Lyu, Tal, and Wu[CHL+23]. Their construction can be viewed as a weighted pseudorandom reduction from long regular ROBPs to short standard ROBPs with large alphabets. Then they fool the short ROBPs with an INW generator. We show that if one replaces this INW generator with our WPRG Theorem F.1 (or alternatively, by the WPRG of [CCD+26]) instead, then we can slightly improve the seed length.

First we recall the "Weight" of an ROBP defined in [BRRY14], which is different from the weight of a WPRG.

Definition 3.18 (Weight of ROBPs)**.** *Let $f \in \mathcal{B}_{n,s,w}$ be an ROBP. For every $v \in [n] \times [w]$, let $f^{v\to}$ denote the subsection of f starting at v and has the same accept nodes as f. Let E be the edges of f. We define the weight of f as:*

$$\mathsf{W}(f) = \sum_{(u,v)\in E} |\mathbb{E}[f^{v\to}] - \mathbb{E}[f^{u\to}]|\,.$$

Recall that we denote the class of all regular ROBPs of length n, width w and alphabet $\Sigma = \{0,1\}^s$ as $\mathcal{R}_{n,s,w}$. Our result can be stated as the following.

Theorem 3.19. *For every $w, n, s \in \mathbb{N}$, and every $\varepsilon > 0$, there exists an explicit ε-WPRG for $\mathcal{B}_{n,s,w}$ with seed length*

$$O\left(s + \log n\left(\sqrt{\log(1/\varepsilon)} + \log w + \log(\mathsf{W}(f)) + \log\log n\right) + \log(1/\varepsilon)\right).$$

Furthermore, the weight of this WPRG is bound by $\text{poly}(n^{1+\sqrt{\log(1/\varepsilon)}}, w)$.

One main result of [BRRY14] states that the weights of regular ROBPs over a binary alphabet are bounded by $O(w^2)$.

Lemma 3.20 (Lemma 6 of [BRRY14])**.** *For every $n, w \in \mathbb{N}$ and $f \in \mathcal{R}_{n,1,w}$, $\mathsf{W}(f) \le O(w^2)$.*

Combining Lemma 3.20 with Theorem 3.19 immediately gives Theorem 1.7.

Before we prove Theorem 3.19, we recall the following from [CHL+23, CL24]. In Section 6.3 of [CHL+23], the WPRG $(G, w) : [K] \times (\{0,1\}^s)^r \to \{0,1\}^n \times \mathbb{R}$ for $\mathcal{R}_{n,1,w}$, has the form

$$G(i, y_1, y_2, \dots y_r) = (G_{i,1}(y_1), G_{i,2}(y_2), \dots, G_{i,r}(y_r))$$
$$\sigma(i, y_1, y_2, \dots y_r) = K \cdot \gamma_i,$$

where $i \in [K]$ and $\forall j \in [r], j \in \{0,1\}^s$. The parameters satisfy $K = n^{O(m)}, r = O(m \log n), \gamma_i \in \{-1, +1\}$, where $m = \Theta\left(\frac{\log(w/\varepsilon)}{\sqrt{\log(1/\varepsilon)}+\log w}\right) = O\left(\sqrt{\log(1/\varepsilon)}\right)$. $G_{i,j}$ is the PRG by [BRRY14] for $\mathcal{R}_{n,1,w}$ within target error $\tau = \min\left\{2^{-\sqrt{\log(1/\varepsilon)}}, 1/\operatorname{poly}(w, \mathsf{W}(f), \log n)\right\}$. Therefore, the length of each y is

$$s = O\left(\log n\left(\sqrt{\log(1/\varepsilon)} + \log w + \log(\mathsf{W}(f)) + \log\log n\right)\right).$$

We view the WPRG as a weighted pseudorandom reduction from regular ROBPs to standard ROBPs by fixing i and let $y_1, \dots, y_r$ be free. Moreover, this WPRG can naturally extend to large alphabet cases although not explicitly stated in [CHL+23]. [CL24] gives a similar result. We restate both of their results as the following theorem, and we include a short proof in Appendix E for completeness.

Theorem 3.21 (Section 6.3 of [CHL+23], also Section 3 of [CL24])**.** *For all $n, w \in \mathbb{N}$ and $\varepsilon > 0$, there exists a $(\log K, K, \varepsilon)$-weighted pseudorandom reduction from $\mathcal{B}_{n,s,w}$ to $\mathcal{B}_{n_1,n_1,w}$, where $K = n^{O(\sqrt{\log(1/\varepsilon)})}, n_1 = O(\log n\sqrt{\log(1/\varepsilon)})$ and*

$$s_1 = O\left(s + \log n\left(\sqrt{\log(1/\varepsilon)} + \log w + \log(\mathsf{W}(f)) + \log\log n\right)\right).$$

Now prove Theorem 3.19.

Proof of Theorem 3.19. In the case of $w > n$, Theorem F.1 already gives a suitable WPRG. So we assume $w \le n$. Denote $f \in \mathcal{B}_{n,s,w}$ as the original regular ROBP. Let $(\mathsf{R}, \sigma) = \{(\mathsf{R}_i, \sigma_i)\}_{i\in[K]}$ be the $(\log K, K, \varepsilon/2)$ -reduction from $\mathcal{R}_{n,s,w}$ to $\mathcal{B}_{n_1,s_1,w}$, given by Theorem 3.21, where $K = n^{O(\sqrt{\log(1/\varepsilon)})}, n_1 = O(\log n\sqrt{\log(1/\varepsilon)})$, and

$$s_1 = O\left(s + \log n\left(\sqrt{\log(1/\varepsilon)} + \log w + \log(\mathsf{W}(f)) + \log\log n\right)\right).$$

Then we have

$$\left|\mathbb{E}[f] - \sum_{i\in[K]} \frac{\sigma_i}{K} \cdot \mathbb{E}_{x\in(\{0,1\}^{s_1})^{n_1}}\left[f(\mathsf{R}_i(x)\right]\right| \le \varepsilon/2,$$

in which $|\sigma_i| \le K$. Let $(G, \sigma') : \{0,1\}^d \to (\{0,1\}^{s_1})^{n_1} \times \mathbb{R}$ be the $\varepsilon/(2K)$-WPRG given by Theorem F.1. We have

$$\begin{aligned}
&\left|\mathbb{E}[f] - \sum_{i\in[K]} \frac{\sigma_i}{K} \cdot \sum_{y\in\{0,1\}^d} \frac{\sigma'(y)}{2^d}\left[f(\mathsf{R}_i(G(y))\right]\right| \\
\le &\left|\mathbb{E}[f] - \sum_{i\in[K]} \frac{\sigma_i}{K} \cdot \mathbb{E}_{x\in(\{0,1\}^{s_1})^{n_1}}\left[f(\mathsf{R}_i(x)\right]\right| + \sum_{i\in[K]} \frac{\sigma_i}{K}\left|\mathbb{E}_{x\in(\{0,1\}^s)^r}\left[f(\mathsf{R}_i(x)\right] - \sum_{y\in\{0,1\}^d} \frac{\sigma'(y)}{2^d}\left[f(\mathsf{R}_i(G(y))\right]\right| \\
\le &\varepsilon/2 + K \cdot \varepsilon/(2K) = \varepsilon.
\end{aligned}$$

Therefore $(\mathsf{R} \circ G, \sigma \cdot \sigma')$ is a ε-WPRG for f with seed length $d_{total} = \log K + d$.

By Theorem F.1, the $\varepsilon/(2K)$-WPRG requires

$$d = O\left(s_1 + \frac{\log n_1 \log(n_1 w)}{\max\{1, \log\log w - \log\log n_1\}} + \log w \log\log \min\{n_1, \log w\} + \log(K/\varepsilon)\right)$$

We use the assumption that $w \le n$, organize the terms and find

$$\begin{aligned} d_{total} =& O\left(s + \log n\left(\sqrt{\log(1/\varepsilon)} + \log w + \log(\mathsf{W}(f)) + \log\log n\right) + \log n_1 \log n + \log(K/\varepsilon)\right) \\ =& O\left(s + \log n\left(\sqrt{\log(1/\varepsilon)} + \log w + \log(\mathsf{W}(f)) + \log\log n\right) + \log(1/\varepsilon)\right), \end{aligned}$$

since $\log n_1 \log(n_1 w) = O(\log n_1 \log n), \log w \log\log \min\{n_1, \log w\} = O(\log n \log n_1)$.

Now we compute the weight, which is the upper bound of $|\sigma_i| \cdot |\sigma'(x)|$ over $i \in [K], x \in \{0,1\}^d$. By Theorem F.1, $|\sigma'(x)| \le \mathrm{poly}(r, w, \varepsilon/2K) = \mathrm{poly}(n^{1+\sqrt{\log(1/\varepsilon)}}, w)$. $|\sigma_i| \le K = n^{O(1+\sqrt{\log(1/\varepsilon)})}$. Therefore, the total weight is bound by $\mathrm{poly}(n^{1+\sqrt{\log(1/\varepsilon)}}, w)$.

□

4 WPRG for Permutation Read-once Branching Programs.

In this section we focus on WPRGs against permutation ROBPs. First we recall the definition of permutation ROBPs with only one accept node.

Definition 4.1 (Permutation ROBPs). *A ROBP $f \in \mathcal{B}_{(n,s,w)}$ is a permutation ROBP if for every $i \in [n]$, $x \in \{0,1\}^s$, the matrix $f^{[i-1,i]}(x)$ is a permutation matrix.*

We denote the class of permutation ROBPs with unbounded width and only one accept node by $\mathcal{P}_{(n,s)}$, where n is the length s is the bit-length of the alphabet.

Now we give our WPRG for $\mathcal{P}_{(n,s)}$.

Theorem 4.2. *[Restatement of Theorem 1.9] There exists an ε-WPRG for $\mathcal{P}_{(n,s)}$ with seed length $s + O(\log n(\log\log n + \sqrt{\log(1/\varepsilon)}) + \log(1/\varepsilon))$ and weight $2^{O(\log n\sqrt{\log(1/\varepsilon)})}$.*

Before we prove the theorem, we introduce some more definitions and lemmas.

Definition 4.3 (PSD norm). *Let A be a $w \times w$ positive semi-definite matrix. The PSD norm on $\mathbb{R}^w$ with respect to A is defined as $\|x\|_A = \sqrt{x^T A x}$.*

Definition 4.4 (sv-approximation [APP+23]). *Let $\widetilde{W}$ and W be two $w \times w$ doubly stochastic matrices. We say that $\widetilde{W}$ is a ε-singular-value approximation of W, denoted by $\widetilde{W} \overset{sv}{\approx}_\varepsilon W$, if for all $x, y \in \mathbb{R}^w$,*

$$\left|y^T(\widetilde{W} - W)x\right| \le \frac{\varepsilon}{4}\left(\|x\|^2_{I - W^T W} + \|y\|^2_{I - WW^T}\right).$$

Definition 4.5 (fooling with sv-error). *Let $(G, \sigma) : \{0,1\}^s \to \{0,1\}^n \times \mathbb{R}$ be a WPRG. Let $\mathcal{C}$ be a family of matrix valued functions $\mathbf{B} : \{0,1\}^s \to \mathbb{R}^{w \times w}$. We say that G fools $\mathcal{C}$ with ε-sv-error if for all $\mathbf{B} \in \mathcal{C}$, $\sum_{z \in \{0,1\}^s}[\frac{1}{2^s}\mathbf{B}(G(z)) \cdot \sigma(x)] \overset{sv}{\approx}_\varepsilon \mathbb{E}_{x \in \{0,1\}^n}[\mathbf{B}(x)]$.*

We describe the PRG against permutation ROBPs from [HPV21b]. In the following descriptions we regard ROBPs of $\mathcal{P}_{(n,s)}$ as matrix-valued functions.

Lemma 4.6 (Lemma 4.1 in [CHL+23], originally by [HPV21b]). *For all s, n, ε there exists a PRG (actually the INW generator) that fools $\mathcal{P}_{(n,s)}$ with ε-sv-error. The seed length is*

$$s + O(\log n(\log\log n + \log(1/\varepsilon))).$$

Remark. *We note that Lemma 4.1 in [CHL+23] only proves the case for $s = 2$, but their proof can naturally be generalized to handle arbitrary s. See Appendix D.*

Next we describe the key ingredient in previous error reductions for WPRGs against permutation ROBPs.

Lemma 4.7 (Claim 9.3 in [CHL+23]). *Let $n \in \mathbb{N}$, let $\tau \in \left(0, \frac{1}{64\log^2 n}\right)$, and let $G : \{0,1\}^d \to (\{0,1\}^s)^n$ be a PRG that fools $\mathcal{P}_{(n,s)}$ with sv-error τ. Let $k \in \mathbb{N}$, and let*

$$\alpha = \left(4 \cdot \sqrt{\tau} \cdot \log n\right)^{k+1}.$$

Then there exists a WPRG $G^{(k)}, \sigma^{(k)}$ that can be written in the form of

$$G^{(k)}(i, x_1, \ldots, x_r) = G(x_1)_{n_{i,1}}, G(x_2)_{n_{i,2}}, \ldots, G(x_r)_{n_{i,r}},$$
$$\sigma^{(k)}(i) = \gamma_i \cdot K.$$

where $K = n^{O(k)}, i \in [K], r = O(k\log n)$, $\gamma_i \in \{-1, 0, 1\}$ and $0 \le n_{i,j} \le n$, such that $G^{(k)}$ fools $\mathcal{P}_{(n,s)}$ with entrywise error α.

4.1 One level of the reduction.

We show that the previous lemma already gives a reduction:

Lemma 4.8. *For any integer n, k, s and any $\varepsilon > 0$, there exists $\tau = \Omega(\varepsilon^{\frac{2}{k+1}}/\log^2 n)$ such that there exists a $(O(k\log n), n^{O(k)}, \varepsilon)$-weighted pseudorandom reduction (R, σ) from $\mathcal{P}_{(n,s)}$ to $\mathcal{P}_{(O(k\log n), s+O(\log n(\log\log n+\log 1/\tau)))}$.*

Proof. Let G be the INW PRG that fools $\mathcal{P}_{(n,s)}$ with τ-sv-error, such that $\tau = \min\{\frac{\varepsilon^{2/(k+1)}}{16\log^2 n}, \frac{1}{64\log^2 n}\}$. Then the seed length is $d = s + O(\log n(\log\log n + \log 1/\tau))$ by Lemma 4.6.

We invoke Lemma 4.7 with G and k, which gives $(G^{(k)}, \sigma^{(k)})$. This WPRG fools $\mathcal{P}_{(n,s)}$ with entrywise error $(4 \cdot \sqrt{\tau} \cdot \log n)^{k+1} < \varepsilon$. We define the reduction $(\mathsf{R}, \sigma) : \mathsf{R}_i(x_1, ..., x_r) = G^{(k)}(i, x_1, ..., x_r), w_i = \sigma^{(k)}(i)$. By Lemma 4.7, the weight of this reduction is bounded by $K = n^{O(k)}$. The seed length of the reduction is $\log K = O(k\log n)$, since we need $\log K = O(k\log n)$ bits to choose i.

Given any $f \in \mathcal{P}_{(n,s)}$, notice that each $f \circ \mathsf{R}_i$ is a permutation ROBP of length $r = O(k\log n)$ and alphabet size $2^d = 2^{s+O(\log n(\log\log n+\log 1/\tau))}$. The reason is that the transition matrix of $f \circ \mathsf{R}_i$ from the $j-1$-th layer to the j-th layer on input $x \in \{0,1\}^d$ can be expressed as $(f \circ \mathsf{R}_i)^{[j-1,j]}(x)$. As $(f \circ \mathsf{R})^{[j-1,j]}(x) = f^{[n_{i,j-1}, n_{i,j}]}(G(x)_{n_{i,j}-n_{i,j-1}})$ and the latter is a product of $n_{i,j} - n_{i,j-1}$ permutation matrices, the transition matrix of $f \circ \mathsf{R}_i$ is also a permutation matrix. Hence $f \circ \mathsf{R}_i$ is a permutation ROBP. Also, note that each $f \circ \mathsf{R}_i$ only has one accept node since its accept node is the same as that of f.

Therefore, the reduction is a $(O(k\log n), O(n^k), \varepsilon)$-weighted pseudorandom reduction.

□

Setting $k = \sqrt{\log(1/\varepsilon)}$ in Lemma 4.8, we immediately have the following lemma:

Lemma 4.9. *For any integer n, s and any $\varepsilon > 0$, there exists a $(O(\log n\sqrt{\log(1/\varepsilon)}), 2^{O(\log n\sqrt{\log(1/\varepsilon)})}, \varepsilon)$-weighted pseudorandom reduction from $\mathcal{P}_{(n,s)}$ to $\mathcal{P}_{(O(\log n\sqrt{\log(1/\varepsilon)}),s+O(\log n(\sqrt{\log(1/\varepsilon)}+\log\log n)))}$.*

Setting $k = O(\log^2 n)$ in Lemma 4.8, we immediately have the following lemma:

Lemma 4.10. *For any integer n, s and $\varepsilon > 0$, there exists a $(O(\log^3 n), 2^{O(\log^3 n)}, \varepsilon)$-weighted pseudorandom reduction from $\mathcal{P}_{(n,s)}$ to $\mathcal{P}_{(\log^3 n,s+O(\frac{\log(1/\varepsilon)}{\log n}+\log n\log\log n))}$.*

4.2 Recursion of reductions.

We give a multi-level recursive procedure which gradually reduces the program from $\mathcal{P}_{(n,s)}$ to $\mathcal{P}_{(O(1),s+O(\log n(\log\log n+\sqrt{\log(1/\varepsilon)})))}$ with error ε.

We start by reducing the length of the permutation ROBP to $O(\log n\sqrt{\log(1/\varepsilon)})$ and the weight to $2^{O(\log n\sqrt{\log(1/\varepsilon)})}$ with error $\varepsilon/2$. Previous works use an INW PRG to fool the reduced ROBP, which leads to an extra double logarithmic factor in the seed length. Here we use iterative reductions instead to avoid this extra factor.

Lemma 4.11 (Reduction for permutation ROBP)**.** *For any integer n, s and any $\varepsilon > 0$, there exists a $\left(O\left(\log n\sqrt{\log(1/\varepsilon)}\right), 2^{\log n\sqrt{\log(1/\varepsilon)}}, O(\varepsilon)\right)$-weighted pseudorandom reduction from $\mathcal{P}_{(n,s)}$ to $\mathcal{P}_{\left(O(1),s+O\left(\log n\left(\log\log n+\sqrt{\log(1/\varepsilon)}\right)+\log(1/\varepsilon)\right)\right)}$.*

Proof. Let $(\mathsf{R}^{(0)}, \sigma^{(0)})$ be the $(d_0, K_0, \varepsilon/2)$-weighted pseudorandom reduction from $\mathcal{P}_{(n,s)}$ to $\mathcal{P}_{(n_1,s_1)}$ such that $n_1 = O(\log n\sqrt{\log(1/\varepsilon)})$ and $s_1 = s + O(\log n(\log\log n + \sqrt{\log(1/\varepsilon)}))$. This reduction is guaranteed by Lemma 4.9 and have the following parameters:

1. $d_0 = O(\log n\sqrt{\log(1/\varepsilon)})$,
2. $K_0 = 2^{O(\log n\sqrt{\log(1/\varepsilon)})}$.

Now we define a new parameter $\varepsilon' = \varepsilon/2K_0$. Let

$$n_0 = n, n_i = \log^3 n_{i-1}$$

for each $i = 1, \ldots, l$ and $n_l = 32768$. It is clear that $l \le \log\log n$.

For each $i = 1, \ldots, l$, let $(\mathsf{R}^{(i)}, \sigma^{(i)})$ be the $(d_i, K_i, (\varepsilon')^2)$-weighted pseudorandom reduction from $\mathcal{P}_{(n_i,s_i)}$ to $\mathcal{P}_{(n_{i+1},s_{i+1})}$, which is guaranteed by Lemma 4.10. The reduction has the following parameters:

1. $d_i = O(\log^3 n_i)$,
2. $K_i = 2^{\log^3 n_i}$,
3. $s_{i+1} = s_i + O(\frac{\log(1/\varepsilon')}{\log n_i} + \log n_i \log\log n_i) = s_i + O(\frac{\log(1/\varepsilon)}{\log n_i})$.

Using Lemma 2.3, we first composite $(\mathsf{R}^{(1)}, \sigma^{(1)}), \ldots, (\mathsf{R}^{(l)}, \sigma^{(l)})$ to get a $(d^*, K^*, \varepsilon^*)$-weighted pseudorandom reduction $(\mathsf{R}^{(*)}, \sigma^{(*)})$ from $\mathcal{P}_{(n_0,s_0)}$ to $\mathcal{P}_{(n_l,s_l)}$ with $\mathsf{R}^{(*)} = \mathsf{R}^{(1)} \circ \ldots \circ \mathsf{R}^{(l)}$ and $\sigma^{(*)} = \sigma^{(1)} \cdot \ldots \cdot \sigma^{(l)}$. We have the following parameters:

1. $d^* = d_1 + \ldots + d_l \le O(l \log^3 n_1) \le O(\log^{0.1}(n/\varepsilon))$,

2. $K^* = K_1 \cdot \ldots \cdot K_l \le 2^{O(l \cdot \log^3 n_1)} \le 2^{O(\log^{0.1}(n/\varepsilon))}$,

3. $\varepsilon^* = \sum_{i=1}^{l} (\varepsilon')^2 \cdot \prod_{j=0}^{i-1} K_j \le (\varepsilon')^2 \cdot K^* \cdot l \le \varepsilon'/2$,

4. $s_l = s_0 + \sum_{i=1}^{l} (s_i - s_{i-1}) = s_0 + \sum_{i=1}^{l} O(\frac{\log(1/\varepsilon')}{\log n_i}) = s_0 + O(\log 1/\varepsilon')$.

The first three items are attained by directly plugging in our parameters. The last item holds because n_i decreases rapidly. Note that $n_{i-1} = 2^{\sqrt[3]{n_i}}$, and when $n_i \ge 2^{15} = 32768$, we have $n_{i-1} \ge n_i^2$. When $n_l \ge 32768$, it holds that $\sum_{i=1}^{l} \frac{\log(1/\varepsilon')}{\log n_i} \le \log 1/\varepsilon' \cdot (1/\log n_l + 1/(2\log n_l) + 1/(4\log n_l) + \cdots + 1/(2^l \log n_l)) \le \log 1/\varepsilon'$.

Finally, we composite $(\mathsf{R}^{(*)}, \sigma^{(*)})$ with $(\mathsf{R}^{(0)}, \sigma^{(0)})$ to get the final reduction (R, σ) from $\mathcal{P}_{(n,s)}$ to $\mathcal{P}_{(O(1),s_l))}$. By Lemma 2.2, the parameters are:

1. $d = d^* + d_0 = O(\log n \sqrt{\log(1/\varepsilon)})$,

2. $K = K^* \cdot K_0 \le 2^{O(\log n \sqrt{\log(1/\varepsilon)})}$,

3. $\varepsilon = \varepsilon/2 + \varepsilon'/2 \cdot K_0 \le \varepsilon$,

4. $s = s_l = s_0 + O(\log 1/\varepsilon') = s + O(\log n(\log\log n + \sqrt{\log(1/\varepsilon)}) + \log(1/\varepsilon))$.

□

One can also see our reduction strategy for the above lemma through Figure 4.2.

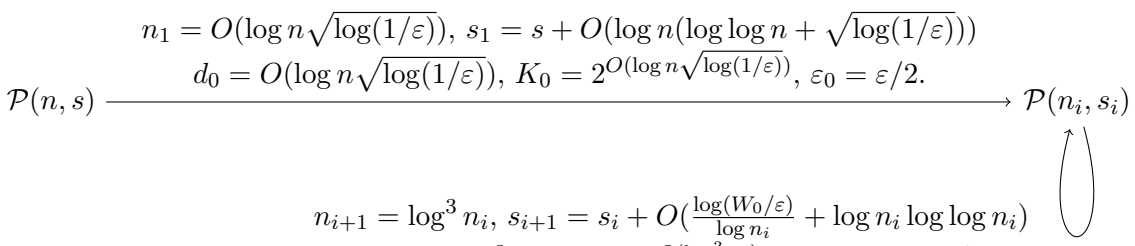


Figure 2: Reduction for permutation ROBPs in Lemma 4.11.

Now we show the main theorem of this section:

Theorem 4.12 (Theorem 4.2 restated)**.** *There exists an ε-WPRG for $\mathcal{P}_{(n,s)}$ with seed length $s + O(\log n(\log\log n + \sqrt{\log(1/\varepsilon)}) + \log(1/\varepsilon))$ and weight $2^{O(\log n \sqrt{\log(1/\varepsilon)})}$.*

Proof. Let (R, σ) be the reduction from Lemma 4.11. We construct the WPRG (G, σ) with the same weight function σ, and G is defined as $G(x, i) = \mathsf{R}(x, i)$. By Lemma 4.11, to fool the original ROBP we only need to provide randomness for the seed of (R, σ) and also provide a random string x which can fool an arbitrary ROBP in $\mathcal{P}_{(O(1),s+O(\log n(\log\log n+\sqrt{\log(1/\varepsilon)})+\log(1/\varepsilon)))}$. We can use another ε error INW generator from [HPV21a] to generate this constant length x and this only takes $s + O\left(\log n\left(\log\log n + \sqrt{\log(1/\varepsilon)}\right) + \log(1/\varepsilon)\right)$ random bits. Also the randomness used by (R, σ) is $O(\log n \sqrt{\log(1/\varepsilon)})$. So the total seed length is as stated. The weight also directly follows from Lemma 4.11.

□

It is well known that a WPRG can fool a Permutation ROBP with an arbitrary number of accept nodes and error ε if it can fool the program with only one accept node with error ε/w. Therefore, we immediately have the following corollary:

Corollary 4.13. *There exists an ε-WPRG for length n, width w, alphabet size (out degree) 2^s Permutation ROBPs having an arbitrary number of accept nodes, with seed length $s + O(\log n(\log\log n + \sqrt{\log(w/\varepsilon)}) + \log(w/\varepsilon))$ and weight $2^{O(\log n\sqrt{\log(w/\varepsilon)})}$.*

5 Derandomization for Regular Branching Programs.

In this section, we show our derandomization for short-wide regular ROBPs, which indicates that when $n \leq 2^{\log^{1/2} w}$ and $\varepsilon = 1/\operatorname{poly}(w)$, there is a $O(\log w)$ space derandomization, where the space complexity is optimal up to constant factors. We achieve this by constructing a more general derandomization algorithm as the following.

Theorem 5.1. *There exists an algorithm that for any input regular ROBP f of width w, length n, alphabet $\{0,1\}^s$, and any input parameter $\varepsilon > 0$ as input, outputs an approximation for $\mathbb{E}[f]$ with addtive error ε. The algorithm uses $O(s + \log n(\log\log n + \sqrt{\log(w/\varepsilon)}) + \log(w/\varepsilon))$ bits of space.*

Notice that our main theorem Theorem 1.10 is a direct corollary of Theorem 5.1. We remark that one may want to further improve the space dependence on ε by first using Theorem 5.1 and then calling the Richardson iteration method of [AKM+20]. This can be done, but our main purpose here is to get a derandomization in **L** exactly for short-wide ROBPs. If applying that additional operation, then the algorithm is not in **L** and also not in polynomial time.

Next as a warm-up, we show that binary regular ROBPs can be transformed in logspace to binary permutation ROBPs such that they have the same acceptance probability (but probably do not compute the same function). We are unaware of any previous work stating this transformation.

Lemma 5.2. *There is an algorithm which on inputting a regular ROBP P with length n width w and binary edge labels, outputs a permutation ROBP P' with length n width w and binary edge labels, such that P' has exactly the same acceptance probability as P (but may not computing the same function as P). The algorithm has workspace $O(\log(nw))$.*

Proof. P' has the same vertex set as that of P. The algorithm visits each layer i of P in the order $i = 0, 1, 2, \ldots, n$. For layer i, consider the subgraph S between V_i and V_{i+1}. The algorithm visits each node $v \in V_i$ in order. For each $v \in V_i$, it tests if v can be reached by any previously visited node of V_i in S. This can be done by revisiting each previous node v' and go through the cycle that v' is in. If none of the previous node can reach v, then do the following traverse in S. Starting from v, choose an edge with label 0, and then go to the corresponding neighbor. For each next node u visited, denote ℓ as the label of the adjacent edge through which we reach u. If the other adjacent edge e' of u has a label ℓ' such that $\ell' = \ell$, then flip the label ℓ'. After this we traverse through e' to go to the next neighbor. Go on doing this until it reaches v again.

Notice that starting from v the traverse can go through every node of $V_i \cup V_{i+1}$ reachable from v. Because for each next neighbor u, it can only be either v or a node which is not touched before in this traverse. In other words, this traverse is going through the cycle of S containing v. Also note that after this traverse, every node reachable from v, because of the flipping, can have different labels for its two adjacent edges. So after we visited every $v \in V_i$, every node of S reachable by nodes in V_i has its two adjacent edges with different labels because of the flips. Hence the graph with these new labels is a permutation. The acceptance probability maintains the same since the

edges are the same as before except that we flip some edge labels. After flippings, each node still has its two outgoing edges with different labels.

The traverse can be done in logspace since recording the index of a layer takes space $O(\log n)$, recording the index of a node in a layer takes space $O(\log w)$, and going through a cycle of S takes space $O(\log w)$.

□

After this transformation, we can run our WPRG of Corollary 4.13 to attain the desired derandomization for the binary case.

However for alphabets with size larger than 2, the problem turns out to be more complicated. In the rest of this section, we focus on the case with large alphabets.

5.1 More preliminaries.

We recall some standard definitions in the literature of [RVW00, RV05, AKM+20, CHL+23, CL24].

Definition 5.3 (Regular bigraph)**.** *A bigraph is a triple $G = (U, V, E)$, where U and V are two sets of vertices and $E \subseteq U \times V$ is a set of edges going from U to V. A bigraph is called d-regular if every vertex in U has d outgoing edges and every vertex in V has d incoming edges.*

The transition matrix of a regular bigraph is a the matrix $\mathbf{M} \in \mathbb{R}^{V \times U}$ such that $\mathbf{M}_{v,u}$ is the fraction of edges going from u that go to v.

Definition 5.4 (One-way labeling)**.** *A one-way labeling of a d-regular bigraph G assigns a label $i \in [d]$ to each edge in G such that for every vertex $u \in U$, the labels of the outgoing edges of u are distinct. If G has a one way labeling, we say that $G[u, i] = v$ if the outgoing edge of u that is labeled i goes to v.*

Definition 5.5 (Two-way labeling)**.** *A two-way labeling of a d-regular bigraph G is a labeling of the edges of G such that:*

- *Every edge (u, v) has two labels in $[d]$, the 'outgoing label' and the 'incoming label'.*
- *For every vertex $u \in U$, the outgoing labels of the outgoing edges of u are distinct.*
- *For every vertex $v \in V$, the incoming labels of the incoming edges of v are distinct.*

Definition 5.6 (Rotation map)**.** *Let $G = (U, V, E)$ be a d-regular bigraph with a two-way labeling. The rotation map of G is a function $\mathsf{Rot}_G : U \times [d] \to V \times [d]$ such that $\mathsf{Rot}_G(u, i) = (v, j)$ if there is an edge $(u, v) \in E$ with the outgoing label i and the incoming label j.*

Definition 5.7 (Derandomized Product)**.** *Let $G_1 = (U, V, E_1)$ and $G_2 = (V, T, E_2)$ be two d-regular bigraphs where G_1 has a two-way labeling and G_2 has a one-way labeling. Let $H = ([d], [d], E_H)$ be a c-regular bigraph with one way labeling. The derandomized product of G_1, G_2 and H is a $(c \cdot d)$-regular bigraph with one-way labeling denoted by $G_1 (\!\circledcirc\!)_H G_2$ defined as follows. To compute $(G_1 (\!\circledcirc\!)_H G_2)[v_0, (i_0, j_0)]$ for $v_0 \in U$ and $(i_0, j_0) \in [d] \times [c]$:*

- *Let $(v_1, i_1) = \mathsf{Rot}_{G_1}(v_0, i_0)$.*
- *Let $i_2 = H[i_1, j_0]$.*
- *Let $v_2 = G_2[v_1, i_2]$.*
- *Output $(G_1 (\!\circledcirc\!)_H G_2)[v_0, (i_0, j_0)] =: v_2$.*

Definition 5.8 (Two-way labeling of derandomized product)**.** *Let $G_1 = (U, V, E_1)$ and $G_2 = (V, T, E_2)$ be two d-regular bigraphs where both G_1 and G_2 have two-way labeling. Let $H = ([d], [d], E_H)$ be a c-regular bigraph with two-way labeling. We define the two-way labeling of the derandomized product $G_1 ⓟ_H G_2$ as follows. To compute the rotation map $\mathsf{Rot}_{G_1 ⓟ_H G_2}$ for any vertex $v_0 \in U$ and any pair $(i_0, j_0) \in [d] \times [c]$, we do the following:*

- *Let $(v_1, i_1) = \mathsf{Rot}_{G_1}(v_0, i_0)$.*
- *Let $(i_2, j_1) = \mathsf{Rot}_H(i_1, j_0)$.*
- *Let $(v_2, i_3) = \mathsf{Rot}_{G_2}(v_1, i_2)$.*
- *Output $\mathsf{Rot}_{G_1 ⓟ_H G_2}(v_0, (i_0, j_0)) = (v_2, (i_3, j_1))$.*

To ensure that the derandomized product behaves like the direct concatenation of the two bigraphs, we need H to be a spectral expander.

Definition 5.9 (spectral expander)**.** *Let $H = (U, V, E_H)$ be a d-regular bigraph with transition matrix $W_H \in \mathbb{R}^{U \times V}$, where $|U| = |V|$. Define $J \in \mathbb{R}^{U \times V}$ where $J_{i,j} = 1/|U|$ for all $i \in U, j \in V$. Then the spectral expansion of H is denoted by $\lambda(H)$ and defined as follows:*

$$\lambda(H) = \|W_H - J\|_2$$

We also need the definitions for regular branching programs:

Definition 5.10 (Regular Branching Program)**.** *Let f be a ROBP in $\mathcal{B}_{n,s,w}$. We call f a regular branching program if all vertices in the graph of f except the vertices in the first layer have precises 2^s incoming edges.*

A general regular branching program may not have a two-way labeling. If we equip it with a two-way labeling for each step of the transition, we call it a regular ROBP with two-way labeling.

Definition 5.11 (Regular ROBP with Two-way labeling)**.** *A regular ROBP with two-way labeling is a regular ROBP f and every edge in f has two labels, the 'outgoing label' and the 'incoming label'. For every vertex u in $V(f) \backslash V_n$, the outgoing labels of the outgoing edges of u are distinct. For every vertex v in $V(f) \backslash V_0$, the incoming labels of the incoming edges of v are distinct. We compute f according to the outgoing labels of the edges, i.e., for every $(x_1, \ldots, x_n) \in (\{0,1\}^s)^n$, we find the unique path $(v_0 = v_{start}, v_1, \ldots, v_n)$ in f such that for every $i \in [n]$, the edge (v_{i-1}, v_i) has the outgoing label x_i, and we output 1 iff v_n is an accepting vertex.*

We denote the class of all regular ROBPs with two-way labeling $\mathcal{R}^{tw}_{n,s,w}$.

Like bigraphs, we define $\mathsf{Rot}_f : ([w] \times [n]) \times \{0,1\}^s \to ([w] \times [n]) \times \{0,1\}^s$ such that $\mathsf{Rot}_f(u, \sigma) = (v, \sigma')$ if there is an edge (u, v) in f with the outgoing label σ and the incoming label σ'.

We recall the derandomization algorithm for regular branching programs in [CHL+23].

Let f be a regular ROBP with two-way labeling in $\mathcal{R}^{tw}_{n,s,w}$. For each $i \in [\log n]$, let $H_t = ([2^s \cdot c^{t-1}], [2^s \cdot c^t], E_{H_t})$ be a c-regular bigraph with $\lambda(H_t) \leq \lambda$. Define $\widetilde{G}_{j \to j+1} := (V_{t-1}, V_t, E_t)$, which is a 2^s-regular bigraph with two-way labeling. Define $E(SC_n) = \{(j, j + 2^t) : j \in [n - 2^t], t \in [\log n], 2^t \text{ is the largest power of 2 dividing } j\}$. For each $(j, j + 2^t) \in E(SC_n)$ recursively define the graph $\widetilde{G}_{j \to j+2^t}$ as follows:

$$\widetilde{G}_{j \to j+2^t} := \widetilde{G}_{j \to j+2^{t-1}} ⓟ_{H_i} \widetilde{G}_{j+2^{t-1} \to j+2^t}$$

The following lemma shows that the transition matrix of the final graph $\widetilde{G}_{0 \to n}$ is close to the product of the transition matrices of the intermediate graphs.

Lemma 5.12 (Lemma A.20 of [CHL+23]). *Let $n, d, c \in \mathbb{N}$. For each $t \in \{0, 1, \ldots, \log n\}$ and each $j \in [n - 2^t]$, let $\widetilde{G}_{j+2^{t-j}} = \left(V_j, V_{j+2^t}, E_{j+2^{t-j}}\right)$ be a $(d \cdot c^t)$-regular bigraph with a two-way labeling. Furthermore, let $\lambda \in \left(0, \frac{1}{6 \log^2 n}\right)$, and for each $t \in [\log n]$, let $H_t = \left([d \cdot c^{t-1}], [d \cdot c^{t-1}], E_{H_t}\right)$ be a c-regular bigraph with a one-way labeling satisfying $\lambda(H_t) \leq \lambda$. Assume also that for each $t \in [\log n]$ and each $j \in [n - 2^t]$, we have*

$$\widetilde{G}_{j \to j+2^t} = \widetilde{G}_{j \to j+2^{t-1}} \textcircled{p}_{H_t} \widetilde{G}_{j+2^{t-1} \to j+2^t}, \forall t \geq 1,$$

and

$$\widetilde{G}_{j \to j+1} = G_{j \to j+1},$$

(where the equation above merely denotes equality as graphs with one-way labelings). For each $(i, j) \in E(\mathsf{SC}_n)$, let $\widetilde{\mathbf{W}}_{j \leftarrow i}$ be the transition matrix of $\widetilde{G}_{i \to j}$, and let

$$\mathbf{W}_{j \leftarrow i} = \widetilde{\mathbf{W}}_{j \leftarrow j-1} \cdots \widetilde{\mathbf{W}}_{i+1 \leftarrow i}$$

Then

$$\widetilde{\mathbf{W}}_{n \leftarrow 0} \overset{sv}{\approx}_{11\lambda \cdot \log n} \mathbf{W}_{n \leftarrow 0}.$$

Theorem 5.13 (Claim A.23 of [CHL+23]). *The following algorithm computes*

$$\mathsf{Rot}_{\widetilde{G}_{0 \to n}}(v_0, (x, e_1, \ldots, e_{\log n}))$$

1. *For $i = 1$ to n:*
   (a) *Update $(v, x) \leftarrow \mathsf{Rot}_{\widetilde{G}_{i-1 \to i}}(v, x)$, so now $v \in V^{(i)}$.*
   (b) *If $i < n$:*
      i. *Let $t \in [\log n]$ be the smallest positive integer such that i is not a multiple of 2^t, i.e., the binary expansion of i has precisely $t - 1$ trailing zeroes.*
      ii. *Update $(x, e_1, \ldots, e_{t-1}) \leftarrow \mathsf{Rot}_{H_t}((x, e_1, \ldots, e_{t-1}), e_t)$.*
2. *Output (v, e).*

We also need the efficiently computable spectral expander H_t. The following lemma shows that such a spectral expander exists. The construction of H_t is based on Margulis-Gabber-Galil graphs [Mar73, GG81].

Lemma 5.14 (Space-efficient expanders, Lemma A.22 of [CHL+23]). *For every $d \in \mathbb{N}$ that is a power of two, for every $\lambda \in (0, 1)$, there is a bigraph $H = \left([d], [d], E_H\right)$ with a two-way labeling satisfying the following:*

- *$\lambda(H) \leq \lambda$.*
- *H is c-regular where c is a power of two and $c \leq \mathrm{poly}(1/\lambda)$.*
- *Rot_H can be evaluated in space that is linear in its input length, i.e., space $O(\log(d/\lambda))$.*

We use the following notation for the algorithm in Theorem 5.13 equipped with the efficiently computable spectral expanders.

Definition 5.15. *We define the algorithm* DerandWalk. *Let $n, s, w \in \mathbb{N}$, and $0 < \tau < \frac{1}{6 \log n}$. For every $t \in [\lceil \log n \rceil]$, let $c \in \mathbb{N}$ and $H_t = \left([2^s \cdot c^{t-1}], [2^s \cdot c^{t-1}], E_{H_t}\right)$ be a c-regular bigraph with a two-way labeling satisfying $\lambda(H_t) \leq \lambda$ constructed by Lemma 5.14. For any $f \in \mathcal{R}^{tw}_{n,s,w}$, $0 \leq l < r \leq n$, $u \in [w]$, $x \in \{0,1\}^s$ and $e_1, e_2 \ldots, e_{\lceil \log n \rceil} \in [c]$, we denote* $\mathrm{DerandWalk}(\tau, f, l, r, u, (x, e_1, \ldots, e_{\lceil \log n \rceil}))$ *as the algorithm in Theorem 5.13 instantiated with the l-th layer to the r-th layer of f and $\{H_t\}$, starting at node u with seed $(x, e_1, \ldots, e_{\lceil \log n \rceil})$, i.e.* $\mathrm{DerandWalk}(\tau, f, l, r, u, (x, e_1, \ldots, e_{\lceil \log n \rceil})) = \mathsf{Rot}_{\widetilde{G}_{l \to r}}(u, (x, e_1, \ldots, e_{\log n}))$, *where $\widetilde{G}_{l \to r}$ is the bigraph generated by Lemma 5.12 with the l-th layer to the r-th layer of f.*

We use the following error reduction polynomial in our algorithm, which gives a good sv-approximation for a sequence of transitions.

Theorem 5.16 (Recursion [CL24])**.** *Let $\{A_i\}_{i=1}^n \subset \mathbb{R}^{w \times w}$ be a sequence of doubly stochastic matrices. Let $\{B_{i,j}\}_{i,j=0}^n \subset \mathbb{R}^{w \times w}$ be a family of matrices such that $B_{i,j} \overset{sv}{\approx}_{\tau/(10 \log n)} A_{i+1} \ldots A_j$ Assuming that $B_{i-1,i} = A_i$ for all i. Then for any $k \in \mathbb{N}$, there exists $K = O((2n)^k)$, $t = O(k \log n)$, a set of indices $m_{i,j}$ for each $i \in [K], j \in [t]$ with $0 \leq m_{i,1} \leq \ldots \leq m_{i,t} = n$ and a set of signs $\sigma_i \in \{-1, 0, 1\}$ for each $i \in [K]$ such that:*

$$\sum_{i \in [K]} \sigma_i \cdot B_{0,m_{i,1}} B_{m_{i,1},m_{i,2}} \ldots B_{m_{i,t-1},m_{i,t}} \overset{sv}{\approx}_{\tau^k} A_1 A_2 \ldots A_n.$$

5.2 The derandomization for regular ROBPs with large alphabets.

Theorem 5.17 (Restatement of Theorem 5.1)**.** *There exists an algorithm that takes as input a regular ROBP and a parameter $\varepsilon > 0$, and outputs an approximation of the acceptance probability with additive error ε. Furthermore, if the regular ROBP has length n, width w and alphabet bit length s, then the algorithm uses $O(s + \log n(\log \log n + \sqrt{\log(w/\varepsilon)}) + \log(w/\varepsilon))$ bits of space.*

Let $f \in \mathcal{R}^{tw}_{n,w,s}$ be a regular ROBP with two-way labeling and let $\mathbf{M}_{l \ldots r}$ denote its transition matrix from layer l to layer r. For every $0 \leq l < r \leq n$, we invoke Lemma 5.12, which gives a regular bigraph $\widetilde{G}_{l \ldots r}$ with two-way labeling such that its transition matrix $\widetilde{\mathbf{M}}_{l \ldots r}$ is a τ-sv-approximation of $\mathbf{M}_{l \ldots r}$. Then by Theorem 5.16, for any $k \in \mathbb{N}$, we have $K = O((2n)^k)$ and a partition $0 = m_{i,0} < m_{i,1} < \cdots < m_{i,l} = n_0$ for every $i \in [K]$ and σ_i such that

$$\sum_{i \in [K]} \sigma_i \widetilde{\mathbf{M}}_{m_{i,0} \ldots m_{i,1}} \widetilde{\mathbf{M}}_{m_{i,1} \ldots m_{i,2}} \cdots \widetilde{\mathbf{M}}_{m_{i,l-1} \ldots m_{i,l}} \text{ approximates } \mathbf{M}_{0 \ldots n_0} \text{ with entry-wise error } \tau^k,$$

Given the summation, we can view it as a sum of regular ROBPs with two-way labelings. For every $i \in [K]$, we define $f_i \in \mathcal{R}^{tw}_{n_1,w,s_1}$ as the concatenation of the bipartite graphs

$$\widetilde{G}_{m_{i,0} \ldots m_{i,1}}, \widetilde{G}_{m_{i,1} \ldots m_{i,2}}, \ldots, \widetilde{G}_{m_{i,l-1} \ldots m_{i,l}},$$

which still has a two-way labeling. We set the starting vertex and accepting vertices of f_i to be the same as those of f. Then we obtain

$$\left| \mathbb{E}_x f(x) - \sum_{i \in [K]} \sigma_i \mathbb{E}_y f_i(y) \right| \leq \tau^k w.$$

The same procedure can be applied to each regular ROBP f_i, which gives shorter ROBPs f_{i_1,i_2} for every $i_2 \in [K_2]$. We repeat the procedure until we attain regular ROBPs $f_{i_1,\ldots,i_\ell}$ with constant length, which can be fooled by true randomness. We formalize the above into the following detailed proof.

Proof of Theorem 5.17. Let $f \in \mathcal{R}^{tw}_{n,w,s}$ be the regular ROBP to derandomize. Let $\ell \in \mathbb{N}$ be the number of iterations we need to perform and let τ_i, ε_i be the error parameters of the i-th iteration for every $i \in [\ell]$. The exact values of $\ell, \{\tau_i, \varepsilon_i\}_{i\in[\ell]}$ will be specified later. For each iteration $i \in [\ell]$, define the error reduction exponent $k_i = \lceil \log_{1/\tau_i}(2/\varepsilon_i) \rceil$, and we setup n_i, K_i iteratively. Let $n_0 = n$. Consider every $i \in [\ell]$. Let $n_i = t, K_i = K$, where $t = O(k_i \cdot \log n_{i-1})$ and $K = O((2n_{i-1})^{k_i})$ are the parameters whose existence is guaranteed by Theorem 5.16, when initiating Theorem 5.16 with $n = n_{i-1}, k = k_i, \tau = \tau_i$.

We iteratively define $f_{i_1,i_2,\ldots,i_p} \in \mathcal{R}^{tw}_{n_p,w,s_p}$ for every $i_1 \in [K_1], i_2 \in [K_2], \ldots, i_p \in [K_p]$ and $p \in \{0, 1, \ldots, \ell\}$ as follows:

1. When $p = 0$, $f_{i_1,i_2,\ldots,i_p}$ is defined to be f.

2. For every $p \in [\ell]$, assume we have defined $f_{i_1,i_2,\ldots,i_{p-1}}$.We denote $\mathbf{M}^{(i_1,i_2,\ldots,i_{p-1})}_{l\ldots r}$ to be the stochastic matrix of $f_{i_1,i_2,\ldots,i_{p-1}}$ from layer l to layer r. For every $0 \le l < r \le n_{p-1}$, let $\widetilde{G}^{(i_1,i_2,\ldots,i_{p-1})}_{l\ldots r}$ be a regular bigraph with a two-way labeling obtained by approximating $\mathbf{M}^{(i_1,i_2,\ldots,i_{p-1})}_{l\ldots r}$ by the algorithm of Lemma 5.12 with error parameter τ_p [6]. Let $\widetilde{\mathbf{M}}^{(i_1,i_2,\ldots,i_{p-1})}_{l\ldots r}$ be the stochastic matrix of $\widetilde{G}^{(i_1,i_2,\ldots,i_{p-1})}_{l\ldots r}$. By Theorem 5.16, for every $i_p \in [K_p]$ we have a partition $0 = m^{(p)}_{i_p,0} < m^{(p)}_{i_p,1} < \cdots < m^{(p)}_{i_p,n_p} = n_{p-1}$ and weights $\sigma^{(p)}_{i_p} \in \{-1, 0, 1\}$ such that

$$\sum_{i_p\in[K_p]} \sigma^{(p)}_{i_p} \widetilde{\mathbf{M}}^{(i_1,i_2,\ldots,i_{p-1})}_{m^{(p)}_{i_p,0}\ldots m^{(p)}_{i_p,1}} \widetilde{\mathbf{M}}^{(i_1,i_2,\ldots,i_{p-1})}_{m^{(p)}_{i_p,1}\ldots m^{(p)}_{i_p,2}} \cdots \widetilde{\mathbf{M}}^{(i_1,i_2,\ldots,i_{p-1})}_{m^{(p)}_{i_p,n_p-1}\ldots m^{(p)}_{i_p,n_p}} \text{ approximates } \mathbf{M}^{(i_1,i_2,\ldots,i_{p-1})}_{0\ldots n_{p-1}} \text{ with entry-wise error } \tau_p^{k_p}. \tag{1}$$

We define $f_{i_1,i_2,\ldots,i_p}$ to be the concatenation of $\widetilde{G}^{(i_1,i_2,\ldots,i_{p-1})}_{m^{(p)}_{i_p,0}\ldots m^{(p)}_{i_p,1}}, \widetilde{G}^{(i_1,i_2,\ldots,i_{p-1})}_{m^{(p)}_{i_p,1}\ldots m^{(p)}_{i_p,2}}, \ldots, \widetilde{G}^{(i_1,i_2,\ldots,i_{p-1})}_{m^{(p)}_{i_p,n_p-1}\ldots m^{(p)}_{i_p,n_p}}$, which also has a two-way labeling.

Having defined the regular ROBPs, we use $|\Sigma_p| = 2^{s_p}$ to denote the alphabet size of $f_{i_1,i_2,\ldots,i_p}$. Note that for every $i_1 \in [K_1], i_2 \in [K_2], \ldots, i_p \in [K_p]$ and $p \in \{0, 1, \ldots, \ell\}$, it holds that $f_{i_1,i_2,\ldots,i_p}$ always has the same alphabet size, since all such regular ROBPs are constructed from Theorem 5.13 with the same set of parameters.

Our derandomization algorithm computes

$$\sum_{i_1\in[K_1],i_2\in[K_2],\ldots,i_\ell\in[K_\ell]} \sigma^{(1)}_{i_1}\sigma^{(2)}_{i_2}\cdots\sigma^{(\ell)}_{i_\ell} \cdot \mathbb{E}_{x\in(\{0,1\}^{s_\ell})^{n_\ell}} f_{i_1,i_2,\ldots,i_\ell}(x).$$

The algorithm operates as follows, assuming we can access the ROBP $f_{i_1,i_2,\ldots,i_\ell}$ with two-way labeling:

1. Enumerate all tuples $(i_1, i_2, \ldots, i_\ell) \in [K_1] \times [K_2] \times \cdots \times [K_\ell]$ and every string $(x_1, x_2, \ldots, x_{n_\ell}) \in (\Sigma_\ell)^{n_\ell}$. For each tuple $(i_1, i_2, \ldots, i_\ell, x_1, x_2, \ldots, x_{n_\ell})$, we compute the following:
   (a) Let $u \in [w]$ be the starting vertex of $f_{i_1,i_2,\ldots,i_\ell}$.
   (b) For every $t \in [n_\ell]$, we compute the rotation map $u \leftarrow \mathsf{Rot}_{f_{i_1,i_2,\ldots,i_\ell}}(u, x_t)$. ($\mathsf{Rot}$ outputs a new pair (u, x_t), but we discard x_t.)
   (c) If u is an accepting vertex of $f_{i_1,i_2,\ldots,i_\ell}$, then add $\sigma^{(1)}_{i_1}\sigma^{(2)}_{i_2}\cdots\sigma^{(\ell)}_{i_\ell}/(2^{n_l\cdot s_l})$ to the result; otherwise, add 0 to the result.

[6] we always consider it as a length $2^{\lceil \log n_{p-1} \rceil}$ ROBP regardless of $r - l$

2. Return the result.

Since we do not have a direct access to $f_{i_1,i_2,\ldots,i_\ell}$, we need to compute it with a recursive program. For every $p \in [\ell]$ all tuples $(i_1, i_2, \ldots, i_p) \in [K_1] \times [K_2] \times \cdots \times [K_p]$ we compute $(u', x') \leftarrow \mathsf{Rot}_{f_{i_1,i_2,\ldots,i_p}}(u, x)$ with the following recursive program, under the assumption that we have already defined the program to compute $\mathsf{Rot}_{f_{i_1,i_2,\ldots,i_{p-1}}}(\cdot,\cdot)$:

1. If $p = 0$, then we compute $\mathsf{Rot}_f(u, x)$ directly with the original ROBP f.
2. Otherwise, assume $u = (a, b)$ is the a-th vertex in the b-th layer of the ROBP $f_{i_1,i_2,\ldots,i_p}$. Then we let $b' \leftarrow b + 1$ and compute $(a', x') \leftarrow \mathsf{Rot}_{\widetilde{G}^{(i_1,i_2,\ldots,i_{p-1})}_{m^{(p)}_{i_p,b} \ldots m^{(p)}_{i_p,b+1}}}(a, x)$ with DerandWalk from Definition 5.15:
   (a) Let $l \leftarrow m^{(p)}_{i_p,b}, r \leftarrow m^{(p)}_{i_p,b+1}$.
   (b) Let $(a', x') \leftarrow \mathrm{DerandWalk}(\tau_p, f_{i_1,i_2,\ldots,i_{p-1}}, l, r, a, x)$, during which the algorithm calls $\mathsf{Rot}_{f_{i_1,i_2,\ldots,i_{p-1}}}$.
   (c) Let $u' \leftarrow (a', b')$.
   (d) Output (u', x').

We now examine both the error bound and the complexity of the algorithm. First we bound the error of the algorithm. By the construction of $f_{i_1,i_2,\ldots,i_p}$, we have

$$\left| \mathbb{E}_x f_{i_1,i_2,\ldots,i_{p-1}}(x) - \sum_{i_p \in [K_p]} \sigma^{(p)}_{i_p} \mathbb{E}_y f_{i_1,i_2,\ldots,i_p}(y) \right| \le \tau_p^{k_p} w \le \varepsilon_p w$$

for every $i_1 \in [K_1], i_2 \in [K_2], \ldots, i_{p-1} \in [K_{p-1}]$. Using triangular inequalities, we have the following bound:

$$\begin{aligned}
&\left| \mathbb{E}_x f(x) - \sum_{i_1 \in [K_1], i_2 \in [K_2], \ldots, i_\ell \in [K_\ell]} \sigma^{(1)}_{i_1} \sigma^{(2)}_{i_2} \cdots \sigma^{(\ell)}_{i_\ell} \cdot \mathbb{E}_y f_{i_1,i_2,\ldots,i_\ell}(y) \right| \\
&\le \sum_{p=0}^{\ell-1} \sum_{i_1 \in [K_1], i_2 \in [K_2], \ldots, i_p \in [K_p]} \left| \mathbb{E}_x f_{i_1,i_2,\ldots,i_p}(x) - \sum_{i_{p+1} \in [K_{p+1}]} \sigma^{(p+1)}_{i_{p+1}} \mathbb{E}_y f_{i_1,i_2,\ldots,i_p,i_{p+1}}(y) \right| \\
&\le \varepsilon_1 w + K_1 \cdot \varepsilon_2 w + K_1 K_2 \cdot \varepsilon_3 w + \cdots + K_1 K_2 \cdots K_{\ell-1} \cdot \varepsilon_\ell w
\end{aligned}$$

For the complexity of the algorithm, notice that the recursive algorithm computing $\mathsf{Rot}_{f_{i_1,i_2,\ldots,i_p}}$ requires $(s_p - s_{p-1}) + \log n_p$ more bits than that of the algorithm computing $\mathsf{Rot}_{f_{i_1,i_2,\ldots,i_{p-1}}}$, since it only needs to store b_l, b_r and the current $e_1, e_2, \ldots, e_{\log\lceil n_{p-1} \rceil}$ during the computation of $((a, b), y) \leftarrow \mathsf{Rot}_{f_{i_1,i_2,\ldots,i_{p-1}}}((a, b), y)$. Given that the base case $p = 0$ requires $O(\log n_0 + s_0 + \log w)$ bits of space, the algorithm to compute $\mathsf{Rot}_{f_{i_1,i_2,\ldots,i_\ell}}$ requires $O(\sum_{p=0}^{\ell} \log n_p + s_\ell + \log w)$ bits of space. The main algorithm to compute the expectation of $f_{i_1,i_2,\ldots,i_\ell}$ requires $O(s_\ell + \sum_{p=0}^{\ell} \log K_p + \log w)$ bits to store the current tuple $(i_1, i_2, \ldots, i_\ell)$ and the current string $(x_1, x_2, \ldots, x_{n_\ell})$ and maintain the result. Therefore, the total space complexity of the algorithm is

$$O\left(\sum_{p=0}^{\ell} \log n_p + s_\ell + \log w + \sum_{p=0}^{\ell} \log K_p \right).$$

We conclude the proof by setting up the parameters $\ell, \{\tau_i, \varepsilon_i\}_{i\in[\ell]}$. We set $\varepsilon_1 = \varepsilon/(2w), k_1 = \lceil \log_{1/\tau_1}(2/\varepsilon_1)\rceil = \sqrt{\log(2w/\varepsilon)}$ and $\tau_1 = \min\{\frac{\varepsilon_1^{2/(k_1+1)}}{16\log^2 n_0}, \frac{1}{64\log^2 n_0}\}$. Then $n_1 = O(\log n\sqrt{\log(w/\varepsilon)}), K_1 = n^{O(\log n\sqrt{\log(w/\varepsilon)})}$. For every $i \in \{2,\dots,\ell-1\}$, we set $\varepsilon_i = \varepsilon/(2wK_1)^2, k_i = O(\log^2 n_{i-1})$ and $\tau_i = \min\{\frac{\varepsilon_{i-1}^{2/(k_i+1)}}{16\log^2 n_{i-1}}, \frac{1}{64\log^2 n_{i-1}}\}$. k_i is chosen such that $n_i = O(k_i \log n_{i-1}) = \log^3 n_{i-1}$ and $K_i = O((2n_{i-1})^{k_i}) = n^{O(\log^3 n_{i-1})}$. We set ℓ to be the smallest integer such that $n_\ell \le 32768$. We can easily verify that $\ell = o(\log\log n)$. Given the parameters, the construction of Lemma 5.12 gives $s_i = s_{i-1} + O(\log n_i(\log\log n_i + \log\tau_i))$ for every $i \in [\ell]$. Therefore, $s_1 = s + O(\log n(\log\log n + \sqrt{\log(w/\varepsilon)}))$ and $s_i = s_{i-1} + O(\frac{\log(wK_1/\varepsilon)}{\log n_{i-1}})$ for every other $i \in [\ell]$.

Plugging the parameters into the error bound, we have

$$\begin{aligned}
&\left|\mathbb{E}_x f(x) - \sum_{i_1\in[K_1], i_2\in[K_2],\dots,i_\ell\in[K_\ell]} \sigma_{i_1}\sigma_{i_2}\cdots\sigma_{i_\ell}\cdot\mathbb{E}_y f_{i_1,i_2,\dots,i_\ell}(y)\right| \\
\le&\varepsilon_1 w + K_1\cdot\varepsilon_2 w + K_1K_2\cdot\varepsilon_3 w + \cdots + K_1K_2\cdots K_{\ell-1}\cdot\varepsilon_\ell w \\
\le&\varepsilon/2 + \ell\cdot K_1K_2\cdots K_{\ell-1}\cdot\frac{\varepsilon}{2wK_1^2}\cdot w \le \varepsilon,
\end{aligned}$$

by noticing that $K_1K_2\cdots K_{\ell-1} \le K_1^2$.

Plugging the parameters also gives the space complexity of the algorithm as

$$\begin{aligned}
&O\left(\sum_{i=0}^{\ell}\log n_i + s_\ell + \log w + \sum_{i=0}^{\ell}\log K_i\right) \\
=&O\left(s + \log n\left(\log\log n + \sqrt{\log(w/\varepsilon)}\right) + \log(w/\varepsilon)\right).
\end{aligned}$$

□

References

[AKM+20] AmirMahdi Ahmadinejad, Jonathan Kelner, Jack Murtagh, John Peebles, Aaron Sidford, and Salil Vadhan. High-precision Estimation of Random Walks in Small Space. In 2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS), pages 1295–1306. IEEE, 2020.

[APP+23] AmirMahdi Ahmadinejad, John Peebles, Edward Pyne, Aaron Sidford, and Salil Vadhan. Singular Value Approximation and Sparsifying Random Walks on Directed Graphs. In 2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS), pages 846–854. IEEE, 2023.

[Arm98] Roy Armoni. On the Derandomization of Space-Bounded Computations. In Michael Luby, José D. P. Rolim, and Maria Serna, editors, Randomization and Approximation Techniques in Computer Science, pages 47–59, Berlin, Heidelberg, 1998. Springer.

[BCG19] Mark Braverman, Gil Cohen, and Sumegha Garg. Pseudorandom pseudo-distributions with near-optimal error for read-once branching programs. SIAM Journal on Computing, 49(5):STOC18–242, 2019.

[BHPP22] Andrej Bogdanov, William M. Hoza, Gautam Prakriya, and Edward Pyne. Hitting sets for regular branching programs. In 37th Computational Complexity Conference (CCC 2022). Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022.

[BRRY14] Mark Braverman, Anup Rao, Ran Raz, and Amir Yehudayoff. Pseudorandom Generators for Regular Branching Programs. SIAM Journal on Computing, 43(3):973–986, January 2014.

[BV10] Joshua Brody and Elad Verbin. The coin problem and pseudorandomness for branching programs. In 2010 IEEE 51st Annual Symposium on Foundations of Computer Science, pages 30–39. IEEE, 2010.

[CCD+26] Ben Chen, Gil Cohen, Dean Doron, Yuval Khaskelberg, and Amnon Ta-Shma. Improved error reduction for weighted PRGs. Technical Report TR26-064, Electronic Colloquium on Computational Complexity (ECCC), May 2026. Revision #3.

[CDR+21] Gil Cohen, Dean Doron, Oren Renard, Ori Sberlo, and Amnon Ta-Shma. Error Reduction for Weighted PRGs Against Read Once Branching Programs. In 36th Computational Complexity Conference, CCC 2021, page 22. Schloss Dagstuhl-Leibniz-Zentrum fur Informatik GmbH, Dagstuhl Publishing, 2021.

[CDST23] Gil Cohen, Dean Doron, Ori Sberlo, and Amnon Ta-Shma. Approximating Iterated Multiplication of Stochastic Matrices in Small Space. In Proceedings of the 55th Annual ACM Symposium on Theory of Computing, pages 35–45, Orlando FL USA, June 2023. ACM.

[CH22] Kuan Cheng and William M. Hoza. Hitting sets give two-sided derandomization of small space. Theory of Computing, 18(1):1–32, 2022.

[CHHL19] Eshan Chattopadhyay, Pooya Hatami, Kaave Hosseini, and Shachar Lovett. Pseudorandom generators from polarizing random walks. Theory of Computing, 15(1):1–26, 2019.

[CHL+23] Lijie Chen, William M. Hoza, Xin Lyu, Avishay Tal, and Hongxun Wu. Weighted Pseudorandom Generators via Inverse Analysis of Random Walks and Shortcutting. In 2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS), pages 1224–1239. IEEE, 2023.

[CL20] Eshan Chattopadhyay and Jyun-Jie Liao. Optimal Error Pseudodistributions for Read-Once Branching Programs. In 35th Computational Complexity Conference (CCC 2020). Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2020.

[CL24] Eshan Chattopadhyay and Jyun-Jie Liao. Recursive Error Reduction for Regular Branching Programs. In 15th Innovations in Theoretical Computer Science Conference (ITCS 2024). Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2024.

[CTS25a] Ben Chen and Amnon Ta-Shma. Better weighted pseudorandom generators against low weight read-once branching programs. In Electronic Colloquium on Computational Complexity (ECCC), 2025.

[CTS25b] Ben Chen and Amnon Ta-Shma. Simplifying Armoni's PRG. In Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques

(APPROX/RANDOM 2025), volume 353, pages 36:1–36:8, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[CW24] Kuan Cheng and Yichuan Wang. BPL $\subseteq$ L-AC^1. In 39th Computational Complexity Conference (CCC 2024). Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2024.

[De11] Anindya De. Pseudorandomness for permutation and regular branching programs. In 2011 IEEE 26th Annual Conference on Computational Complexity, pages 221–231. IEEE, 2011.

[GG81] Ofer Gabber and Zvi Galil. Explicit constructions of linear-sized superconcentrators. Journal of Computer and System Sciences, 22(3):407–420, 1981.

[Gol11] Oded Goldreich. A sample of samplers: A computational perspective on sampling. In Studies in complexity and cryptography: miscellanea on the interplay between randomness and computation, pages 302–332. Springer, 2011.

[GUV09] Venkatesan Guruswami, Christopher Umans, and Salil Vadhan. Unbalanced expanders and randomness extractors from Parvaresh–Vardy codes. Journal of the ACM, 56(4):1–34, June 2009.

[Hoz21] William M. Hoza. Better Pseudodistributions and Derandomization for Space-Bounded Computation. In Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021). Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

[HPV21a] 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 2021). Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2021.

[HPV21b] 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 2021). Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2021.

[INW94] Russell Impagliazzo, Noam Nisan, and Avi Wigderson. Pseudorandomness for network algorithms. In Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing - STOC '94, pages 356–364, Montreal, Quebec, Canada, 1994. ACM Press.

[KNP11] Michal Kouckỳ, Prajakta Nimbhorkar, and Pavel Pudlák. Pseudorandom generators for group products. In Proceedings of the forty-third annual ACM symposium on Theory of computing, pages 263–272, 2011.

[KNW08] Daniel M. Kane, Jelani Nelson, and David P. Woodruff. Revisiting Norm Estimation in Data Streams. arXiv preprint arXiv:0811.3648, 2008.

[Mar73] Grigorii Aleksandrovich Margulis. Explicit constructions of concentrators. Problemy Peredachi Informatsii, 9(4):71–80, 1973.

[Nis92a] Noam Nisan. Pseudorandom generators for space-bounded computation. Combinatorica, 12(4):449–461, December 1992.

[Nis92b] Noam Nisan. RL $\subseteq$ SC. In Proceedings of the twenty-fourth annual ACM symposium on Theory of computing, pages 619–623, 1992.

[NZ96] Noam Nisan and David Zuckerman. Randomness is Linear in Space. Journal of Computer and System Sciences, 52(1):43–52, February 1996.

[PP23] Aaron (Louie) Putterman and Edward Pyne. Near-Optimal Derandomization of Medium-Width Branching Programs. In Proceedings of the 55th Annual ACM Symposium on Theory of Computing, pages 23–34, 2023.

[PRZ23] Edward Pyne, Ran Raz, and Wei Zhan. Certified Hardness vs. Randomness for Log-Space. In 2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS), pages 989–1007. IEEE, 2023.

[PV21] Edward Pyne and Salil Vadhan. Pseudodistributions that beat all pseudorandom generators (extended abstract). In 36th Computational Complexity Conference (CCC 2021), volume 200, pages 33:1–33:15, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[RSV13] Omer Reingold, Thomas Steinke, and Salil Vadhan. Pseudorandomness for regular branching programs via fourier analysis. In International Workshop on Approximation Algorithms for Combinatorial Optimization, pages 655–670. Springer, 2013.

[RV05] Eyal Rozenman and Salil Vadhan. Derandomized Squaring of Graphs. In International Workshop on Approximation Algorithms for Combinatorial Optimization, pages 436–447. Springer, 2005.

[RVW00] Omer Reingold, Salil Vadhan, and Avi Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders and extractors. In Proceedings 41st Annual Symposium on Foundations of Computer Science, pages 3–13. IEEE, 2000.

[Ste12] Thomas Steinke. Pseudorandomness for permutation branching programs without the group theory. Technical Report TR12-083, Electronic Colloquium on Computational Complexity (ECCC), July 2012.

[SZ99] Michael Saks and Shiyu Zhou. $\mathrm{BP}_H\mathrm{SPACE}(S) \subseteq \mathrm{DSPACE}(S^{3/2})$. Journal of Computer and System Sciences, 58(2):376–403, April 1999.

[Vad12] Salil P. Vadhan. Pseudorandomness. Foundations and Trends® in Theoretical Computer Science, 7(1-3):1–336, 2012.

A A proof for Lemma 3.5.

We reprove the lemma for completeness.

Definition A.1 (total variation)**.** *Let A and B be two random variables defined on a common probability space X. The total variation distance between A and B is defined as*

$$d_{TV}(A,B) = \frac{1}{2}\sum_{x\in X} \left|\Pr[A=x] - \Pr[B=x]\right|.$$

Definition A.2 (Min-Entropy)**.** *Let X be a random variable. The min-entropy of X is defined as*

$$H_\infty(X) = -\log_2\left(\max_{x\in X}\Pr[X=x]\right).$$

Definition A.3 (Conditional Min-Entropy [Vad12] Problem 6.7)**.** *Let X and A be two random variables. The conditional min-entropy of X given Y is defined as*

$$\tilde{H}_\infty(X|Y) = -\log_2\left(\mathop{\mathbb{E}}_{y\in Y}\sup_{x\in X}\Pr[X=x|Y=y]\right).$$

We need the following lemmas.

Lemma A.4 (Chain rule of Conditional Min-Entropy [Vad12] Problem 6.7)**.** *If $|supp(A)| \le 2^s$, then $\widetilde{H}_\infty(X|A) \ge H_\infty(X) - s$.*

Lemma A.5 (Problem 6.8 of [Vad12])**.** *Let $Ext : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ be a (k,ε)-extractor. If $\tilde{H}_\infty(X|A) \ge k$, then*

$$d_{TV}\left((Ext(X,U_d),A),(U_m,A)\right) \le 3\varepsilon.$$

(Here U_m and U_d are independent uniform random variables of length m and d respectively.)

Lemma A.6 (Data Processing Inequality)**.** *Let X, Y be two random variables in the same probability space. Let A be a random variable that is independent of both X and Y. Let f be a function. Then*

$$d_{TV}(f(X,A),f(Y,A)) \le d_{TV}(X,Y).$$

Now we can prove the lemma.

Lemma A.7 (Lemma 3.5 restated)**.** *Let $s \ge \log w$. Assume there exists a $(2s, \frac{\varepsilon}{3n})$-extractor $\textsc{Ext} : \{0,1\}^{3s} \times \{0,1\}^d \to \{0,1\}^s$. Let X and $Y_1,\dots,Y_n$ be independent uniform random variables. Then the following construction*

$$\mathsf{NZ}(X,Y) = \textsc{Ext}(X,Y_1),\dots,\textsc{Ext}(X,Y_n)$$

fools any $f \in \mathcal{B}_{(n,s,w)}$ with error at most ε.

Proof. Let $f \in \mathcal{B}_{(n,s,w)}$. Let X and $Y_1,\dots,Y_n$ be independent uniform random variables. We use a hybrid argument. Define U_i to be independent uniform random variables of length s for each $i \in [n]$. Define $Z_i = \textsc{Ext}(X,Y_i)$ for each $i \in [n]$. Define R_i to be the random variable over V_i, which represents the distribution of the state of f on input $U_1,\dots,U_i$. Define $\widetilde{R}_i$ to be the random variable over V_i which represents the distribution of the state of f on input $Z_1,\dots,Z_i$. We will show that $d_{TV}(R_i,\widetilde{R}_i) \le \frac{i\cdot\varepsilon}{n}$ for all $i \in [n]$.

The base case $i = 0$ is trivial, as both R_0 and $\widetilde{R}_0$ represent the initial state of f. Assume the statement holds for $i-1$. For the case i, notice that $|\mathrm{supp}(\widetilde{R}_{i-1})| \le w \le 2^s$. By the chain rule Lemma A.4,

$$\tilde{H}_\infty(X|\widetilde{R}_{i-1}) \ge 2s.$$

Therefore Lemma A.5 implies

$$d_{TV}((Z_i,\widetilde{R}_{i-1}),(U_i,\widetilde{R}_{i-1})) \le \frac{\varepsilon}{n}.$$

By the data processing inequality, we have

$$d_{TV}((U_i, \widetilde{R}_{i-1}), (U_i, R_{i-1})) \leq d_{TV}(\widetilde{R}_{i-1}, R_{i-1}) \leq \frac{(i-1)\varepsilon}{n}.$$

Using the triangle inequality, we have

$$d_{TV}((Z_i, \widetilde{R}_{i-1}), (U_i, R_{i-1})) \leq \frac{i\varepsilon}{n}.$$

Denote the transition function of f from V_{i-1} to V_i as T_i. Then $\widetilde{R}_i = T_i(Z_i, \widetilde{R}_{i-1})$ and $R_i = T_i(U_i, R_{i-1})$. Using the data processing inequality again, we have

$$d_{TV}(R_i, \widetilde{R}_i) \leq d_{TV}((U_i, R_{i-1}), (Z_i, \widetilde{R}_{i-1})) \leq \frac{i\varepsilon}{n}.$$

Therefore, $d_{TV}(R_n, \widetilde{R}_n) \leq \varepsilon$ and the lemma is proved. □

B A proof for Theorem 3.8.

We prove Theorem 3.8 using the Richardson Iteration for completeness. The Richardson iteration is a method to obtain a finer approximation of L^{-1} from a coarse approximation B of L^{-1} and the invertible matrix L itself.

Lemma B.1. *Let $L \in \mathbb{R}^{m\times m}$ be an invertible matrix and $B \in \mathbb{R}^{m\times m}$ such that $\|B - L^{-1}\| \leq \varepsilon_0$ for a submultiplicative norm $\|\cdot\|$. For any non-negative integer k, define*

$$R(B, L, k) = \sum_{i=0}^{k} (I - BL)^i B$$

Then $\|L^{-1} - R(B, L, k)\| \leq \|L^{-1}\| \cdot \|L\|^{k+1} \cdot \varepsilon_0^{k+1}$.

Proof.

$$\begin{aligned}
\|L^{-1} - R(B, L, k)\| &= \left\| \left(I - \sum_{i=0}^{k} (I - BL)^i BL \right) L^{-1} \right\| \\
&= \left\| (I - BL)^{k+1} L^{-1} \right\| \\
&\leq \|I - BL\|^{k+1} \cdot \left\| L^{-1} \right\| \\
&\leq \left\| L^{-1} \right\| \cdot \|L\|^{k+1} \cdot \varepsilon_0^{k+1}
\end{aligned}$$

□

Theorem B.2 (Theorem 3.8 restated)**.** *Let $\{A_i\}_{i=1}^n \subset \mathbb{R}^{w\times w}$ be a sequence of matrices. Let $\{B_{i,j}\}_{i,j=0}^n \subset \mathbb{R}^{w\times w}$ be a family of matrices such that for every $i + 1 < j$, $\|B_{i,j} - A_{i+1} \dots A_j\| \leq \varepsilon/(2(n+1))$ for some submultiplicative norm $\|\cdot\|$, $\|A_i\| \leq 1$ for all i and also $B_{i-1,i} = A_i$ for all i. Then for any odd $k \in \mathbb{N}$, there exists a $K = (8n)^{k+1}$, a set of indices $\{n_{i,j}\}_{i\in[K], j\in[k]}$ with $0 \leq n_{i,1} \leq \dots \leq n_{i,k} = n$, and signs $\sigma_i \in \{-1, 0, 1\}, i \in [K]$ such that (We set $B_{i,i} = I$ for all i):*

$$\left\| A - \sum_{i\in[K]} \sigma_i \cdot B_{0,n_{i,1}} B_{n_{i,1},n_{i,2}} \dots B_{n_{i,k-1},n_{i,k}} \right\| \leq \varepsilon^{(k+1)/2} \cdot (n+1).$$

Proof. Define $L \in \mathbb{R}^{(n+1)w\times(n+1)w}$ and $B \in \mathbb{R}^{(n+1)w\times(n+1)w}$ as follows:

$$L = \begin{pmatrix} I & 0 & 0 & \dots & 0 & 0 \\ -A_1 & I & 0 & \dots & 0 & 0 \\ 0 & -A_2 & I & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & -A_n & I \end{pmatrix}, B = \begin{pmatrix} I & 0 & 0 & \dots & 0 & 0 \\ B_{0,1} & I & 0 & \dots & 0 & 0 \\ B_{0,2} & B_{1,2} & I & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ B_{0,n} & B_{1,n} & B_{2,n} & \dots & B_{n-1,n} & I \end{pmatrix}$$

This means that

$$L^{-1} = \begin{pmatrix} I & 0 & 0 & \dots & 0 & 0 \\ A_1 & I & 0 & \dots & 0 & 0 \\ A_1A_2 & A_2 & I & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ A_1A_2\dots A_n & A_2\dots A_n & A_3\dots A_n & \dots & A_n & I \end{pmatrix}$$

By assumption, there exists a submultiplicative norm $\|\cdot\|$ on $\mathbb{R}^{w\times w}$. We induce a submultiplicative norm on $\|\cdot\|_A$ on $\mathbb{R}^{(n+1)w\times(n+1)w}$: Let $M = (M_{i,j})_{i,j=1}^{n+1} \in \mathbb{R}^{(n+1)w\times(n+1)w}$, where each $M_{i,j} \in \mathbb{R}^{w\times w}$. Define $\|M\|_A = \max_{j\in[n+1]} \sum_{i=1}^{n+1} \|M_{i,j}\|$. It is easy to verify that $\|\cdot\|_A$ is a submultiplicative norm, and $\|M\|_A = \|M\|_1$ when $w = 1$.

Beacuse $\|B_{i,j} - A_{i+1}\dots A_j\| \le \varepsilon/(n+1)^2$ for all $i+1 < j$, we have $\|B - L^{-1}\|_A \le \varepsilon$. Also, Because $\|A_i\| \le 1$ for all i, we have $\|L\|_A \le 2$ and $\|L^{-1}\|_A \le n+1$.

Define $M = R(B, L, (k-1)/2)$, then

$$\|L^{-1} - M\|_A \le \|L^{-1}\|_A \cdot \|L\|_A^{\frac{k+1}{2}} \cdot \left(\frac{\varepsilon}{n+1}\right)^{\frac{k+1}{2}} \le \varepsilon^{(k+1)/2} \cdot (n+1).$$

Expand $M = (M_{i,j})_{i,j=1}^{n+1}$, where each $M_{i,j} \in \mathbb{R}^{w\times w}$. We foucus on $M_{1,n+1}$, which is a $w \times w$ matrix and $\|M_{1,n+1} - A_1A_2\dots A_n\| \le \|L^{-1} - M\|_A \le \varepsilon^{(k+1)/2} \cdot (n+1)$.

To express $M_{1,n+1}$, we define

$$\Delta_{i,j} = \begin{cases} B_{i,j-1}A_j - B_{i,j} & i < j, \\ 0 & i \ge j. \end{cases}$$

Then we have

$$M_{1,n+1} = B_{0,n} + \sum_{j=1}^{(k-1)/2} \sum_{0<r_1<\dots<r_j<n} \Delta_{0,r_1}\Delta_{r_1,r_2}\cdots\Delta_{r_{j-1},r_j}B_{r_j,n}.$$

Defining $M_{i,j}^{(0)} = B_{i,j-1}A_j = B_{i,j-1}B_{j-1,j}$ and $M_{i,j}^{(1)} = B_{i,j}$, we have

$$M_{1,n+1} = B_{0,n} + \sum_{j=1}^{(k-1)/2} \sum_{0<r_1<\dots<r_j<n} \sum_{t_1,\dots,t_j\in\{0,1\}} (-1)^{t_1+\dots+t_j} M_{0,r_1}^{(t_1)} M_{r_1,r_2}^{(t_2)} \dots M_{r_j,n}^{(t_j)}.$$

Encoding $j, r_1, \dots, r_j, t_1, \dots, t_j$ into a single index $i \in [K]$, rewrite $M_{0,r_1}^{(t_1)} M_{r_1,r_2}^{(t_2)} \dots M_{r_j,n}^{(t_j)}$ as $B_{0,n_{i,1}}B_{n_{i,1},n_{i,2}}\dots B_{n_{i,k-1},n_{i,k}}$ for some $n_{i,1} \le \dots \le n_{i,k} = n$, define $\sigma_i = (-1)^{t_1+\dots+t_j}$, we have the desired result.

Finally, we bound K by $\frac{k-1}{2} \cdot (n+1)^{(k-1)/2} \cdot 2^{(k-1)/2} \le (8n)^{k+1}$.

□

C A proof for Lemma 3.16.

In this section we provide a proof for Lemma 3.16, which is similar to that of [Hoz21]. We start by sampling a random matrix using a sampler, using a union bound.

Lemma C.1. *Let* $\mathrm{Samp} : \{0,1\}^p \times \{0,1\}^r \to \{0,1\}^q$ *be a* (α,γ)*-averaging sampler. Then for every matrix valued function* $f : \{0,1\}^q \to \mathbb{R}^{w\times w}$ *such that* $\|f\|_1 \le C$*, we have:*

$$\Pr_{x\in\{0,1\}^r}\left[\left\|2^{-p}\sum_{y\in\{0,1\}^p} f(\mathrm{Samp}(x,y)) - \mathbb{E}[f]\right\|_1 \ge C\alpha w\right] \le w^2\gamma.$$

Proof.

$$\begin{aligned}
&\Pr_{x\in\{0,1\}^r}\left[\left\|2^{-p}\sum_{y\in\{0,1\}^p} f(\mathrm{Samp}(x,y)) - \mathbb{E}[f]\right\|_1 \ge C\alpha w\right]\\
=&\Pr_{x\in\{0,1\}^r}\exists_{i\in[w]}\left[\sum_{j\in[w]}\left[2^{-p}\sum_{y\in\{0,1\}^p} f(\mathrm{Samp}(x,y))_{i,j} - \mathbb{E}[f]_{i,j}\right] \ge C\alpha w\right]\\
\le&\Pr_{x\in\{0,1\}^r}\exists_{i\in[w],j\in[w]}\left[\left|2^{-p}\sum_{y\in\{0,1\}^p} f(\mathrm{Samp}(x,y))_{i,j} - \mathbb{E}[f]_{i,j}\right| \ge C\alpha\right]\\
\le&\sum_{i\in[w],j\in[w]}\Pr_{x\in\{0,1\}^r}\left[\left|C^{-1}2^{-p}\sum_{y\in\{0,1\}^p} f(\mathrm{Samp}(x,y))_{i,j} - C^{-1}\mathbb{E}[f]_{i,j}\right| \ge \alpha\right]\\
\le&w^2\gamma.
\end{aligned}$$

□

Lemma C.2 (Lemma 3.16 restated)**.** *For all* n,s,w*, assume there exists a* W*-bounded* $1/(2w\cdot(n+1)^2)$*-WPRG* (G_0,w_0) *for* $\mathcal{B}_{(n,s,w)}$ *with seed length* d*, and a* (α,γ)*-averaging sampler* $\mathrm{Samp} : \{0,1\}^r\times\{0,1\}^p\to\{0,1\}^d$ *with* α,γ *as defined above. Then there exists a* ε*-WPRG for* $\mathcal{B}_{(n,s,w)}$ *with seed length* $r+kp+ks$ *and weight* $W^{\log(n/\varepsilon)/\log(nw)}\cdot\mathrm{poly}(nw/\varepsilon)$*.*

Proof. Take any $f\in\mathcal{B}_{(n,s,w)}$. Let $A_i = \mathbb{E}f^{[i-1,i]}(G_0(U))$ for $i\in[n]$. Define $B_{i,j} = \frac{1}{2^p}\sum_{z\in\{0,1\}^d} w_0(z)\cdot f^{[i,j]}(G(z))$. For any $x\in\{0,1\}^r$ and $0\le i<i+1<j\le n$, we define $B^x_{i,j} = \frac{1}{2^p}\sum_{y\in\{0,1\}^p} w_0(\mathrm{Samp}(x,y))\cdot f^{[i,j]}((G(\mathrm{Samp}(x,y)))_{j-i})$. For $j=i+1$, we define $B^x_{i,i+1} = A_i$ as the true transition matrix.

We call a seed x to be 'good' iff for all $i,j\in[n]$, $\|B_{i,j}-B^x_{i,j}\|_1\le 1/(2w\cdot(n+1))$. Otherwise, we call x to be 'bad'.

Since w_0 is W-bounded, $\|w_0(z)\cdot f^{[i,j]}(G(z))\|_1$ is at most W. Note that $W\alpha w\le 1/(2w\cdot(n+1)^2)$. By Lemma C.1, for any $i,j\in[n]$,

$$\Pr_{x\in\{0,1\}^r}[\|B_{i,j}-B^x_{i,j}\|_1 > 1/(2w\cdot(n+1))]\le w^2\gamma = \varepsilon/(2(2n)^k\cdot W^k).$$

By a union bound, the probability $\Pr_{x\in\{0,1\}^r}[x \text{ is bad}]$ is at most $\varepsilon/(2(2n)^k\cdot W^k)$.

For a good seed x, we have $\|A_i\dots A_j - B^x_{i,j}\|_1 \le \|A_i\dots A_j - B_{i,j}\|_1 + \|B_{i,j}-B^x_{i,j}\|_1 \le 1/(w\cdot(n+1)^2)$. By Theorem 3.8, we have:

$$\left\|A_1\cdots A_n - \sum_{i=1}^{k}\sigma_i B^x_{0,i_1}B^x_{i_1,i_2}\dots B^x_{i_{k-1},i_k}\right\|_1 \le ((n+1)w)^{-k}\cdot(n+1)\le\varepsilon/2.$$

For a bad seed, we can upper-bound $\|B^x_{i,j}\|_1$ by K. Therefore, we have:

$$\left\| A_1 \cdots A_n - \sum_{i=1}^{K} \sigma_i B^x_{0,i_1} B^x_{i_1,i_2} \dots B^x_{i_{k-1},i_k} \right\|_1 \le 1 + K \cdot W^k$$

Combining the two cases, we have the following inequality:

$$\begin{aligned}
&\left| \mathbb{E} f - \frac{1}{K \cdot 2^{kp+r}} \sum_{x \in \{0,1\}^r} \sum_{y_1,\dots,y_k \in \{0,1\}^p} w(x, y_1, \dots, y_k, i) f(G(x, y_1, \dots, y_k)) \right| \\
\le &\left\| A_1 \cdots A_n - \frac{1}{2^r} \sum_{x \in \{0,1\}^r} \sum_{i=1}^{K} \sigma_i B^x_{0,i_1} B^x_{i_1,i_2} \dots B^x_{i_{k-1},i_k} \right\|_1 \\
\le &\frac{1}{2^r} \sum_{\substack{x \in \{0,1\}^r \\ x \text{ is bad}}} \left\| A_1 \cdots A_n - \sum_{i=1}^{K} \sigma_i B^x_{0,i_1} B^x_{i_1,i_2} \dots B^x_{i_{k-1},i_k} \right\|_1 \\
&+ \frac{1}{2^r} \sum_{\substack{x \in \{0,1\}^r \\ x \text{ is good}}} \left\| A_1 \cdots A_n - \sum_{i=1}^{K} \sigma_i B^x_{0,i_1} B^x_{i_1,i_2} \dots B^x_{i_{k-1},i_k} \right\|_1 \\
\le &\varepsilon/2 + (1 + K \cdot W^k) \cdot \gamma \\
\le &\varepsilon.
\end{aligned}$$

The resulting generator uses $r + kp + ks$ random bits in total: r bits for the sampler seed, p bits for each of the k sampler queries, and s bits for each of the k one-step transitions.

The lemma follows. □

D Proof sketch for Lemma 4.6.

Though not explicitly mentioned, the analysis of the INW generator in appendix B of [CHL+23] works for large alphabet ROBPs. Here we slightly modifies their proof to make it explicitly works.

We start with the definition of the INW generator. Some notions are defined in the preliminaries of Section 5

Definition D.1 (INW generator for large alphabets)**.** *Let n, c, d be powers of two. For each $t \in [\log n]$, let H_t be a c-regular bigraph $H_t = ([d \cdot c^{t-1}], [d \cdot c^t], E_{H_t})$, with a one-way label. Relative to the family $(H_t)_{t \in [\log n]}$, we recursively define the INW generator as follows:*

Define $\mathsf{INW}_0 : \{0,1\}^{\log d} \to \{0,1\}^{\log d}$ as the trivial PRG. For each $t \in [\log n]$, having defined $\mathsf{INW}_{t-1} : \{0,1\}^{d+(t-1)\log c} \to \{0,1\}^{\log d \cdot 2^{t-1}}$, we define $\mathsf{INW}_t : \{0,1\}^{d+t\log c} \to \{0,1\}^{\log d \cdot 2^t}$ as $\mathsf{INW}_t(x, y) = \mathsf{INW}_{t-1}(x), \mathsf{INW}_{t-1}(H_t[x, y])$. Here the comma denotes concatenation.

The INW generator relates to the derandomized product of permutation ROBPs, since they have consistent one-way labelings.

Definition D.2 (consistent consistent one-way labelings)**.** *Let $G = (U, V, E)$ be a d-regular bigraph. A consistent one-way labeling of G is a one-way labeling of G such that for every $v \in V$, the labels of the incoming edges of v are distinct, i.e., the $G[v, i] = G[u, i]$ implies $u = v$. In this case, we can extend the labeling to a two-way labelings:*

$$\mathsf{Rot}_G(u, i) = (G[u, i], i),$$

i.e., the incoming label and the outgoing label of an edge (u, v) are the same.

Lemma D.3 (Derandomized Product with consistent consistent one-way labelings,[RV05])**.** *Let $G_1 = (U, V, E_1)$ and $G_2 = (V, W, E_2)$ be two d-regular bigraphs with consistent one-way labelings. Let H be a x-regular bigraph with one-way labeling. Then $G_1 ⓟ_H G_2$ has a consistent one-way labeling.*

Let f be a permutation ROBP in $\mathcal{P}_{(n,s)}$. Then f is also a regular ROBP with a consistent labeling. It can be sv-approximated with the method of Lemma 5.12. Furthermore, the method corresponds to fooling f with the INW generator.

Lemma D.4 (equivariance between INW generator and Lemma 5.12)**.** *Let f be a permutation ROBP in $\mathsf{P}_{(n,s)}$. Let $t \in [\log n]$, $(j, j + 2^t) \in E_{SC_n}$ and $\widetilde{G}_{j,j+2^t}$ be the $(d \cdot c^t)$-regular bigraph with two-way labeling as defined in Lemma 5.12. Then for any $u \in V^j$ and $x \in \{0,1\}^{d+t \log c}$, we have*

$$\exists x', \widetilde{G}_{j,j+2^t}[u, x] = [v, x'] \Leftrightarrow \left[f^{[j,j+2^t]}(\mathsf{INW}(x))\right]_{u,x} = v,$$

Proof. The proof is by induction on t. For the base case $t = 0$, both sides mean u has an outgoing edge labeled x going to v. Assume the statement holds for $t - 1$. Recall that

$$\widetilde{G}_{j\to j+2^t} = \widetilde{G}_{j\to j+2^{t-1}} ⓟ_{H_t} \widetilde{G}_{j+2^{t-1}\to j+2^t}, \forall t \geq 1,$$

Then for t, the left side can be computed as follows (where $x = (z, y)$):

$$\begin{aligned}
(w, z) &= \mathsf{Rot}_{\widetilde{G}_{j,j+2^{t-1}}}(u, z),\\
(z', y') &= \mathsf{Rot}_{H_t}(z, y),\\
(v, w) &= \mathsf{Rot}_{\widetilde{G}_{j+2^{t-1},j+2^t}}(z', y').
\end{aligned}$$

Where the right side can be computed as follows:

$$\begin{aligned}
&f^{[j,j+2^{t-1}]}(\mathsf{INW}_{t-1}(z)) = w,\\
&H_t[z, y] = z',\\
&f^{[j+2^{t-1},j+2^t]}(\mathsf{INW}_{t-1}(z')) = v.
\end{aligned}$$

The induction hypothesis implies that the two sides are equivalent. □

Finally, we can prove the lemma.

Lemma D.5 (Lemma 4.6 restated)**.** *For all s, n, ε there exists a PRG (actually the INW generator) that fools $\mathcal{P}_{(n,s)}$ with ε-sv-error. The seed length is*

$$s + O(\log n(\log\log n + \log(1/\varepsilon))).$$

Proof. Let $\lambda = \min\{\varepsilon/(11 \log n), 1/6(\log^2 n)\}$. Let $d = 2^s$ and for each $t \in [\log n]$, let H_t be a c-regular expanders with $\lambda(H_t) \leq \lambda$ from Lemma 5.14. Let $\{\mathsf{INW}_t\}_{t\in[\log n]}$ be the INW generator from the definition. Let $\mathsf{INW} = \mathsf{INW}_{\log n}$. Then INW is a PRG with seed lengths

$$\log d + t \cdot \log c = s + O(\log n(\log\log n + \log(1/\varepsilon))).$$

For any $f \in \mathcal{P}_{(n,s)}$, the derandomized product of f coincides with the the output of $f \circ INW$, i.e. $\widetilde{\mathbf{W}}_{n\leftarrow 0} = E_X f^{[0,n]}(\mathsf{INW}(X))$. Since $\widetilde{\mathbf{W}}_{n\leftarrow 0} \overset{sv}{\approx}_{11\lambda \log n} \widetilde{\mathbf{W}}_{n\leftarrow n-1} \cdots \widetilde{\mathbf{W}}_{1\leftarrow 0}$ by Lemma 5.12, INW fools f with ε-sv-error. □

E Reduction for regular ROBPs over large alphabets.

In this section, we show that the WPRG construction in Section 3 of [CL24] naturally extends to the case of regular ROBPs with large alphabets. This will immediately give us Theorem 3.21. We start by introducing an equivalent definition of the weight of ROBPs,

Definition E.1 (An Equivalent Definition of Weight of ROBPs)**.** *Let $f \in \mathcal{B}_{n,s,w}$ be an ROBP. Let $V = \bigcup_{i=0}^{n} V_i$ be its vertices and $E = \bigcup_{i=1}^{n} E_{i-1,i}$ be its edges, where $E_{i-1,i} \subseteq V_{i-1} \times V_i$ For every $0 \le l \le r \le n$, Let $\mathbf{M}_{l\dots r}$ be the stochastic matrix of f from layer l to layer r. Let e_i denote the unitary vector at the i-th coordinate. For every $y \in \mathbb{R}^w$, $i \in [n]$, define the layer i weight of f on y as:*

$$\mathsf{W}(f,i,y) = \sum_{(u,v)\in E_{i-1,i}} \left|e_u^T \mathbf{M}_{i-1\dots i} y - e_v^T y\right|$$

For every $0 \le l < r \le n$, the total weight between layer l and layer r of f on y is defined as:

$$\mathsf{W}(f,l,r,y) = \sum_{i=l+1}^{r} \mathsf{W}(f,i,\mathbf{M}_{i\dots r} y).$$

We immediately have the following facts:

Lemma E.2. $0 \le l < m < r \le n$, $\mathsf{W}(f,l,r,y) = \mathsf{W}(f,l,m,\mathbf{M}_{m\dots r} y) + \mathsf{W}(f,m,r,y)$

Lemma E.3. $\mathsf{W}(f) = \mathsf{W}(f,0,n,y_{acc})$, *where y_{acc} is the accept vertices vector of f.*

Proof.

$$\begin{aligned}\mathsf{W}(f,0,n,y_{acc}) &= \sum_{i=1}^{n} \mathsf{W}(f,i,\mathbf{M}_{i\dots n} y_{acc}) = \sum_{i\in[n]} \sum_{(u,v)\in E_{i-1,i}} \left|e_u^T \mathbf{M}_{i-1\dots n} y_{acc} - e_v^T \mathbf{M}_{i\dots n} y_{acc}\right| \\ &= \sum_{(u,v)\in E} |\mathbb{E}[f^{u\to} - f^{v\to}]| = \mathsf{W}(f)\end{aligned}$$

□

We need a main result of [BRRY14].

Lemma E.4 ([BRRY14])**.** *For every $n, s, w \in \mathbb{N}$ and $\varepsilon > 0$, there exists a PRG $G : \{0,1\}^d \to (\{0,1\}^s)^w$ such that for every $f \in \mathcal{B}_{n,s,w}$ and every $y \in \mathbb{R}^w$:*

$$\left\| \left(\mathbb{E}_{x\in\{0,1\}^d} f^{[l,r]}(G(x)) \right) y - \mathbf{M}_{l\dots r} y \right\|_\infty \le \varepsilon \mathsf{W}(f,l,r,y).$$

Furthermore, $d = s + O(\log n(\log\log n + \log(w/\varepsilon))$

We use the error reduction polynomials in [CL24]. Without loss of generality, assume that n is a power of 2 and define the set BS_n as

$$\mathsf{BS}_n = \{(l,r) \in [n]^2 \mid \exists i,k \in \mathbf{N} \cup \{0\}, l = i\cdot 2^k, r = l + 2^k, 0 \le l < r \le n\}.$$

Given a set of substochastic matrices $\{\mathbf{M}_1, \dots, \mathbf{M}_n\}$, suppose for every $(l,r) \in \mathsf{BS}_n$, we have a matrix $\mathbf{M}^{(0)}_{l\dots r}$ that coarsely approximates the product $\mathbf{M}_{l+1} \cdots \mathbf{M}_r$. They define the following matrices $\mathbf{M}^{(k)}_{l\dots r}$ as

$$\mathbf{M}^{(k)}_{l\dots r} = \begin{cases} \mathbf{M}_r & \text{if } r = l+1, \\ \sum_{i+j=k} \mathbf{M}^{(i)}_{l\dots m} \mathbf{M}^{(j)}_{m\dots r} - \sum_{i+j=k-1} \mathbf{M}^{(i)}_{l\dots m} \mathbf{M}^{(j)}_{m\dots r} & \text{otherwise, where } m = (l+r)/2. \end{cases} \tag{2}$$

The following result of [CL24] shows that $\mathbf{M}^{(k)}_{l\dots r}$ is a good approximation as long as k is large enough.

Theorem E.5 (Lemma 3.6 of [CL24]). *For any $0 < \varepsilon < \frac{1}{30\log n}$ Assume that for every $(l,r) \in \mathsf{BS}_n$*

$$\forall y \in \mathbb{R}^w, \left\| \left(\mathbf{M}^{(0)}_{l\dots r} - \mathbf{M}_{l\dots r} \right) y \right\|_\infty \le \varepsilon \frac{\mathsf{W}(f,l,r,y)}{\mathsf{W}(f)}$$

Then for every $(l,r) \in \mathsf{BS}_n$ and $k \in \mathbb{N}$

$$\forall y \in \mathbb{R}^w, \left\| \left(\mathbf{M}^{(k)}_{l\dots r} - \mathbf{M}_{l\dots r} \right) y \right\|_\infty \le (30\varepsilon \cdot \log n)^k \frac{\mathsf{W}(f,l,r,y)}{\mathsf{W}(f)}$$

Now we prove Theorem 3.21

Proof of Theorem 3.21. Set $k = O(\sqrt{\log(1/\varepsilon)})$. Set $\tau = \frac{\varepsilon^{1/k}}{30\log n \cdot \mathsf{W}(f))}$. Let $\mathsf{PRG} : \{0,1\}^d \to (\{0,1\}^s)^n$ be the τ-PRG given by Lemma E.4 and let $(\mathsf{PRG}(\cdot))_m$ denote its first m symbols over $\{0,1\}^s$. we define $\mathbf{M}^{(0)}_{l\dots r} := \mathbb{E}_{x\in\{0,1\}^d} f^{[l,r]}((\mathsf{PRG}(x))_{r-l})$ and expand $\mathbf{M}^{(k)}_{0\dots n}$ as a weighted sum of products of $\mathbf{M}^{(0)}_{l\dots r}$, we get the following sum:

$$\mathbf{M}^{(k)}_{0\dots n} = \frac{1}{K} \sum_{i\in K} \gamma_i \cdot \mathbf{M}^{(0)}_{m_{i,0}\dots m_{i,1}} \mathbf{M}^{(0)}_{m_{i,1}\dots m_{i,2}} \cdots \mathbf{M}^{(0)}_{m_{i,t-1}\dots m_{i,t}},$$

where $\gamma_i \in \{-K, K\}$ are weights and $0 = m_{i,0} \le m_{i,1} \le \cdots \le m_{i,t} = n$ are partitions of $[n]$. We set the $(\log K, K, \varepsilon)$-reduction (R, σ) from $\mathcal{R}_{n,s,w}$ to $\mathcal{B}_{t,d,w}$ as

$$\begin{aligned} \mathsf{R}(x_1, x_2, \cdots, x_t, i) &= ((\mathsf{PRG}(x_1))_{m_{i,1}-m_{i,0}}, (\mathsf{PRG}(x_2))_{m_{i,2}-m_{i,1}}, \cdots, (\mathsf{PRG}(x_t))_{m_{i,t}-m_{i,t-1}}), \\ \sigma(i) &= \gamma_i. \end{aligned}$$

By Lemma E.4, $\forall y \in \mathbb{R}^w, \left\| \left(\mathbf{M}^{(0)}_{l\dots r} - \mathbf{M}_{l\dots r} \right) y \right\|_\infty \le \frac{\varepsilon^{1/k}}{30\log n} \cdot \frac{\mathsf{W}(f,l,r,y)}{\mathsf{W}(f)}$. Then by Theorem E.5, $\left\| \left(\mathbf{M}^{(k)}_{0\dots n} - \mathbf{M}_{0\dots n} \right) y_{acc} \right\|_\infty \le \varepsilon \cdot \frac{\mathsf{W}(f,0,n,y_{acc})}{\mathsf{W}(f)} = \varepsilon$, which indicates that our reduction achieves the target precision.

Given the construction of the error reduction polynomial, one can find $K = n^{O(k)} = 2^{O(\log n\sqrt{\log(1/\varepsilon)})}$ and $t = O(k\log n) = O(\log n\sqrt{\log(1/\varepsilon)})$. Also, Lemma E.4 indicates $d = s + O(\log n(\log\log n + \log w + \log(\mathsf{W}(f)) + \sqrt{\log(1/\varepsilon)})$. That completes the proof. □

F A revised WPRG construction for standard ROBPs

We show the following revised WPRG for standard ROBPs, whose seed length attains optimal dependence on the alphabet bit-length, matching that of Chen, Cohen, Doron, Khaskelberg, and Ta-Shma [CCD+26].

Theorem F.1. *For all integer n, s, w, there exists an explicit construction of a ε-WPRG for $\mathcal{B}_{(n,s,w)}$ with seed length*

$$s + O\left(\frac{\log n \log(nw)}{\max\{1, \log\log w - \log\log n\}} + \log w \left(\log\log \min\{n, \log w\} - \log\log \max\left\{ \frac{\log w}{\log \frac{n}{\varepsilon}}, 2 \right\} \right) + \log(1/\varepsilon) \right),$$

and weight $\mathrm{poly}(nw/\varepsilon)$.

The revised construction does not use the sampler-based techniques of [Hoz21]. It follows a three-stage weighted pseudorandom reductions. In the first stage we apply a single reduction to reduce the length from n to $n_0 = \min\{n, \log(n/\varepsilon)/\log n\}$. Then we consider two cases. Case 1 is that $w > n/\varepsilon$, and for this case we mainly follow our previous construction but replacing the alphabet reduction by using the PRG of [CTS25b]. Case 2 is that $w \le n/\varepsilon$ and this is the part we mainly adjust. We can further reduce the length to

$$n' = O\left(\frac{\log^2(n/\varepsilon)}{\log^2 w}\right),$$

by using a stage of iterative reductions. If this is smaller than a constant then we will finish by using the PRG of [CTS25b]. So for the remaining (the 3rd stage) we only consider this length to be a super constant. Notice that at the beginning of the 3rd stage, we have a collection of ROBPs of length n', each of which needs to be fooled with error $\varepsilon/\operatorname{poly}(nw)$. Notice that $\log(n/\varepsilon) \ge \Theta(\sqrt{n'}\log(w)) \ge \Theta(\log^2 n' \log(n'w))$, by the definition of n'. We then set

$$\varepsilon_0 = 2^{-\frac{\log(n/\varepsilon)}{\log^2 n'}}.$$

Therefore, an application of the PRG of [CTS25b] with error ε_0 requires only

$$O(\log(n'w/\varepsilon_0)\log n') = O\left(\frac{\log(n/\varepsilon)}{\log n'}\right)$$

bits of extra randomness. By inserting this PRG into the error reduction framework, we can shorten the length to $O(\log^2 n')$ at the cost of increasing the alphabet bit length by

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

This procedure can be repeated until the length is reduced to $O(1)$ and then we use the PRG of [CTS25b]. The total amount of extra randomness used accumulates to at most

$$O(\log(n/\varepsilon)).$$

F.1 Three weighted pseudorandom reductions

We use the Chen-Ta-Shma generator [CTS25b] which achieves the optimal seed length dependency on the alphabet size.

Theorem F.2 (Chen-Ta-Shma generator [CTS25b])**.** *For every length n, width w, alphabet Σ, and error $0 < \varepsilon < 1/2$, there is an explicit PRG that ε-fools every length-n, width-w, alphabet-Σ ordered branching program, with seed length*

$$r = \log|\Sigma| + O\left(\frac{\log(wn/\varepsilon)\cdot\log n}{\log\left(2+\frac{\log w}{\log(n/\varepsilon)}\right)}\right).$$

Instantiating our length reduction framework Lemma 3.7 with this generator, we have the following reduction lemmas, which now optimally depend on the alphabet. The first two lemmas are variants of Lemma 3.10 and Lemma 3.11, adapted to the setting of arbitrarily small error ε.

Lemma F.3 (Length reduction 1)**.** *For all integers n, s, w and $\varepsilon > 0$, there exists an explicit $(\log K, K, \varepsilon)$-weighted pseudorandom reduction from $\mathcal{B}_{(n,s,w)}$ to $\mathcal{B}_{\left(k, s+O\left(\frac{\log n \log(nw)}{\log\log w - \log\log n}\right), w\right)}$, where $K = (8n)^{k+1}$, $k = \frac{\log(n/\varepsilon)}{\log n}$.*

Proof. We use Theorem F.2 for $\mathcal{B}_{(n,s,w)}$ with error $\varepsilon_0 = \frac{2^{-4\log n}}{(n+1)^2}$. By Theorem F.2, the seed length is $s + O\left(\frac{\log n \log(nw)}{\log\log w - \log\log n}\right)$. Then we can set $k = \frac{\log(n/\varepsilon)}{\log n} \geq \frac{2\log((n+1)/\varepsilon)}{\log(1/((n+1)^2\varepsilon_0))}$ in Lemma 3.7 and invoke it. This gives a reduction of error $(\varepsilon_0(n+1)^2)^{\frac{k+1}{2}} \cdot (n+1) \leq \varepsilon$. □

Lemma F.4 (Length reduction 2)**.** *For any constants $c \in \mathbb{N}$, for all integers n, w and $\varepsilon > 0$, if $n \leq \log^c w$, then there exists an explicit $(\log K, K, \varepsilon)$-weighted pseudorandom reduction from $\mathcal{B}_{(n,s,w)}$ to $\mathcal{B}_{(k, s+C'\log w, w)}$, where $K \leq (8n)^{k+1}, k = n^{1/c} \cdot \frac{\log(n/\varepsilon)}{\log w}$, C' is a constant depending on c.*

Proof. Fix c to be a constant. We use Theorem F.2 for $\mathcal{B}_{(n,s,w)}$ with error $\varepsilon_0 = \frac{2^{-\frac{4\log w}{n^{1/c}}}}{(n+1)^2}$. By Theorem F.2, the seed length is $s + O\left(\log(nw/\varepsilon_0) \cdot \frac{\log n}{\log\log w - \log\log(n/\varepsilon_0)}\right) = s + O\left(\log w \cdot \frac{\log n}{\log n^{1/c}}\right) = s + C'\log w$, where C' depends on c. Then we can set $k = n^{1/c} \cdot \frac{\log(n/\varepsilon)}{\log w} \geq \frac{\log(1/\varepsilon)+\log(n+1)}{1/2\cdot\log(1/((n+1)^2\varepsilon_0))}$ in Lemma 3.7 and invoke it. This gives a reduction of error $(\varepsilon_0(n+1)^2)^{\frac{k+1}{2}} \cdot (n+1) \leq \varepsilon$. □

Our third reduction, which resembles Lemma 4.10, but instead instantiated with the Chen-Ta-Shma generator.

Lemma F.5 (length reduction 3)**.** *For all integers n, s, w and $\varepsilon > 0$ satisfying $\varepsilon \leq 2^{-2\log^2 n \log(nw)}$, there exists an explicit $(\log K, K, \varepsilon)$-weighted pseudorandom reduction from $\mathcal{B}_{(n,s,w)}$ to $\mathcal{B}_{\left(k, s+O\left(\frac{\log(1/\varepsilon)}{\log n}\right), w\right)}$, where $K = (8n)^{k+1}$, $k = 10\log^2 n$.*

Proof. We use Theorem F.2 for $\mathcal{B}_{(n,s,w)}$ with error $\varepsilon_0 = \frac{2^{-\frac{\log((n+1)/\varepsilon)}{\log^2 n}}}{(n+1)^2}$. The condition $\varepsilon \leq 2^{-2\log^2 n \log(nw)}$ guarantees that $\log(nw/\varepsilon_0) \leq 2\log(1/\varepsilon_0)$.

By Theorem F.2, the seed length is $s + O((\log(nw/\varepsilon_0) \cdot \log n)) = s + O\left(\frac{\log(1/\varepsilon)}{\log n}\right)$. Then we can set $k = 10\log^2 n \geq \frac{\log((n+1)/\varepsilon)}{1/2\cdot\log(1/(n+1)^2\varepsilon_0)}$ in Lemma 3.7 and invoke it. This gives a reduction of error $(\varepsilon_0(n+1)^2)^{\frac{k+1}{2}} \cdot (n+1) \leq \varepsilon$. □

F.2 Proof of Theorem F.1

Proof. We do three stages of reductions: In the first stage, we reduce the length from n to $n_0 = \min\{n, \log(n/\varepsilon)/\log n\}$. Then we split the algorithm into two cases: if $w > n/\varepsilon$, we reduce the length to $\frac{\log w}{\log n/\varepsilon}$, then fool it with PRG; if $w \leq n/\varepsilon$, we reduce the length to $\left(\frac{10\log(n/\varepsilon)}{\log w}\right)^2$, then we further reduce the length to $O(1)$ using a different parameter scheme, and fool it with PRG.

For the first stage, if $n \leq \frac{\log(n/\varepsilon)}{\log n}$, then the goal is already achieved, and we set $n_0 = n$. Otherwise, we use Lemma F.3 to reduce the length from n to $\frac{\log(n/\varepsilon)}{\log n}$ via a $(\log K, K, \varepsilon/2)$-weighted pseudorandom reduction, where $K \leq n^2/\varepsilon^2$. Recall that the number of reduced ROBPs is K. The reduced ROBPs have length $n_0 = \frac{\log(n/\varepsilon)}{\log n}$, width w, and alphabet bit-length

$$s_0 = s + O\left(\frac{\log n \log(nw)}{\log\log w - \log\log n}\right).$$

We set $\varepsilon' = \varepsilon/2K^2$ which will serve as the error parameter for the remaining stages.

1. **Case** $w > n/\varepsilon$**:** We iteratively use Lemma F.4 for l times (l will be defined later): In iteration $0 \le i \le l$, we reduce the length from n_i to $n_{i+1} = n_i^{1/4}\frac{\log(n_i w/\varepsilon')}{\log w} \le 4n_i^{1/4}$, decomposing one ROBP into K_i shorter ones and increasing the alphabet bit-length from s_i to s_{i+1}. Parameter calculation shows

$$s_{i+1} = s_i + C\log w, \qquad K_i \le (8n_i)^{\frac{\log(n_i/\varepsilon')}{\log w}n_i^{1/4}+1} \le (n/\varepsilon)^{\frac{2n_i^{1/4}\log n_i}{\log w}} \le (n/\varepsilon)^{1/\log^{1/4} w},$$

where the first inequality is directly due to Lemma F.4 and the second inequality follows from $n_i \le n_0 = \log(1/\varepsilon)/\log n \le \log^2 w$.

The iteration goes until $n_l \le \frac{\log w}{\log n/\varepsilon'}$, which continues for at most $l = \log\log n_l - \log\log n' \le \log\log\min\{n, \log w\} - \log\log\frac{\log w}{\log n/\varepsilon}$ rounds. When the iteration stops, the reduced ROBPs of length n_l and alphabet bit-length s_l could be ε'-fooled by Chen-Ta-Shma PRG with seed-length $s_l + O\left(\log(nw/\varepsilon')\cdot\frac{\log n_l}{\log\log w - \log\log(n_l/\varepsilon')}\right) = s_l + O(\log(nw/\varepsilon)$.

To account for the total error introduced, we first bound the number of reductions we use in the second stage. We have l rounds of reductions, and in each round we have at most $K\cdot\prod_{i=0}^{l} K_i$ parallel reductions. Therefore, the total amount of reductions we use is at most $l\cdot K\cdot\prod_{i=0}^{l} K_i \le K\cdot(n/\varepsilon)^{\frac{2\log\log w}{\log^{1/4} w}} \le K^2$. The total error introduced in the second stages is at most $K^2\cdot\varepsilon' = \varepsilon/2$. Together with the error $\varepsilon/2$ introduced in the first stage, we have that the total error introduced is at most ε.

Then we bound the randomness we use. The randomness used in selecting the index of reductions is at most $\log(K\cdot\prod_{i=0}^{l} K_i) \le O(\log(nw/\varepsilon))$. The randomness used in fooling the final reduced ROBPs equals $s_l + O(\log nw/\varepsilon')$. By the constructions, we introduce $s_0 - s = O\left(\frac{\log n\log(nw)}{\log\log w - \log\log n}\right)$ bits in the first stage, $s_l - s_0 = O(l\cdot\log w)$ bits, which is $O\left(\log w\cdot\left(\log\log\min\{n,\log w\} - \log\log\frac{\log w}{\log n/\varepsilon}\right)\right)$ bits in the second stage. Therefore, the total randomness we use is at most

$$s + O\left(\frac{\log n\log(nw)}{\log\log w - \log\log n} + \log w(\log\log\min\{n,\log w\} - \log\log\frac{\log w}{\log n/\varepsilon}) + \log(1/\varepsilon)\right)$$

as we claimed.

2. **Case** $w \le n/\varepsilon$**:** We will reduce the length to $n' = \left(\frac{10\log(n/\varepsilon)}{\log w}\right)^2$. If $n_0 \le n'$, the goal is already achieved and we directly go to the 3rd stage.

Now consider $n' < n_0$. One can deduce that $\log(1/\varepsilon) \le \log^2 w\log n$, since $n_0 \le \log(1/\varepsilon)/\log n$. We iteratively use Lemma F.4 for l times (l will be defined later): In iteration $0 \le i \le l$, we reduce the length from n_i to $n_{i+1} = n_i^{1/4}\frac{\log(n_i w/\varepsilon')}{\log w}$, decomposing one ROBP into K_i shorter ones and increasing the alphabet bit-length from s_i to s_{i+1}. Notice that $n_{i+1} < n_i^{3/4}$ when $n_i \ge \left(\frac{\log(n_i w/\varepsilon')}{\log w}\right)^2$, which is always true when $n_i \ge n'$. Parameter calculation shows

$$s_{i+1} = s_i + C\log w, \qquad K_i \le (8n_i)^{\frac{\log(n_i w/\varepsilon')}{\log w}n_i^{1/4}+1} \le (nw/\varepsilon)^{\frac{2n_i^{1/4}\log n_i}{\log w}} \le (nw/\varepsilon)^{1/\log^{1/4} w},$$

where the first inequality is directly due to Lemma F.4 and the second inequality follows from $n_i \le n_0 = \log(1/\varepsilon)/\log n \le \log^2 w$. The iteration goes until $n_l \le n'$, which continues for at most $l = \log\log n_l - \log\log n' \le \log\log\min\{n,\log w\}$ rounds.

Now we enter the third stage and we have that $n' \leq \left(\frac{\log(1/\varepsilon')}{\log w}\right)^2$. Thus, $\log(1/\varepsilon') \geq \log w \cdot \sqrt{n'} \geq \log(n'w) \cdot \log^2 n'$, enabling us to apply Lemma F.5. We use an iterative approach: We set $n'_0 = n_l < n', s'_0 = s_l$, and in iteration $0 \leq i \leq l'$, we reduce the length from n'_i to $n'_{i+1} = 10\log^2 n'_i$, decomposing one ROBP into K'_i shorter ones and increasing the alphabet bit-length from s'_i to s'_{i+1}. Parameter calculation shows

$$\begin{aligned}
s'_{i+1} &= s'_i + O\left(\frac{\log(1/\varepsilon')}{\log n'_i}\right), \\
K'_i &\leq (8n'_i)^{10\log^2 n'_i+1} \\
&\leq (8n')^{10\log^2 n'+1} \\
&\leq 2^{100\log^3 n'} \\
&\leq 2^{100\log^3 \log(1/\varepsilon')} \\
&\leq 2^{100\log^3 \log(n/\varepsilon)} \\
&\leq 2^{\frac{\log(n/\varepsilon)}{\log^{1/2} w}} \\
&\leq (n/\varepsilon)^{1/\log^{1/2} w}.
\end{aligned}$$

We stress that the sequence $\log n_i$ decreases exponentially, making the sum $\sum_{i=0}^{l'} O\left(\frac{\log(1/\varepsilon')}{\log n'_i}\right)$ dominated by a geometric series (i.e. $\sum_{i=0}^{l'} O\left(\frac{\log(1/\varepsilon')}{2^{l'-i}\log n'_{l'}}\right)$), and therefore the total increase of alphabet bit-length is at most $O(\log(1/\varepsilon'))$. The iteration continues for at most $l' = \log\log n'_l \leq \log\log\log(n/\varepsilon) \leq \log\log w$ rounds, until $n'_{l'} \leq O(1)$ and we fool the final reduced ROBPs with Chen-Ta-Shma PRG with seed-length $s'_{l'} + O(\log nw/\varepsilon')$.

Overall, for this case, the total amount of reductions we use is at most $(l+l') \cdot K \cdot \prod_{i=0}^{l} K_i \cdot \prod_{i=0}^{l'} K'_i \leq K \cdot (n/\varepsilon)^{\frac{2\log\log w}{\log^{1/4} w}} \leq K^2$. The total error introduced in the second and third stages is at most $K^2 \cdot \varepsilon' = \varepsilon/2$. Together with the error $\varepsilon/2$ introduced in the first stage, we have that the total error introduced is at most ε.

Then we bound the randomness we use. The randomness used in selecting the index of reductions is at most $\log(K \cdot \prod_{i=0}^{l} K_i \cdot \prod_{i=0}^{l'} K'_i) \leq O(\log(nw/\varepsilon))$. The randomness used in fooling the final reduced ROBPs equals $s'_{l'} + O(\log nw/\varepsilon')$. By the constructions, we introduce $s_0 - s = O\left(\frac{\log n\log(nw)}{\log\log w - \log\log n}\right)$ bits in the first stage, and $s_l - s_0 = O(l \cdot \log w)$ bits in the 2nd stage, which is $O\left(\log w \cdot \log\log \min\{n, \log w\}\right)$. In the third stage, we introduce $s'_{l'} - s_l = O(\log(1/\varepsilon')) = O(\log(n/\varepsilon))$ bits. Therefore, the total randomness we use is at most

$$s + O\left(\frac{\log n\log(nw)}{\log\log w - \log\log n} + \log w \cdot \log\log \min\{n, \log w\} + \log(n/\varepsilon)\right)$$

as we claimed.

□

ECCC ISSN 1433-8092
https://eccc.weizmann.ac.il