New PRGs for Unbounded-Width/Adaptive-Order Read-Once Branching Programs

Lijie Chen
Miller Institute for Basic Research in Science at University of California at Berkeley, CA, USA

Xin Lyu
University of California at Berkeley, CA, USA

Avishay Tal
University of California at Berkeley, CA, USA

Hongxun Wu
University of California at Berkeley, CA, USA

Abstract

We give the first pseudorandom generators with sub-linear seed length for the following variants of read-once branching programs (roBPs):

1. First, we show there is an explicit PRG of seed length $O(\log^2(n/\varepsilon)\log(n))$ fooling *unbounded-width unordered* permutation branching programs with a single accept state, where n is the length of the program. Previously, [Lee-Pyne-Vadhan RANDOM 2022] gave a PRG with seed length $\Omega(n)$ for this class. For the ordered case, [Hoza-Pyne-Vadhan ITCS 2021] gave a PRG with seed length $\widetilde{O}(\log n \cdot \log 1/\varepsilon)$.
2. Second, we show there is an explicit PRG fooling *unbounded-width unordered* regular branching programs with a single accept state with seed length $\widetilde{O}(\sqrt{n \cdot \log(1/\varepsilon)} + \log(1/\varepsilon))$. Previously, no non-trivial PRG (with seed length less than n) was known for this class (even in the ordered setting). For the ordered case, [Bogdanov-Hoza-Prakriya-Pyne CCC 2022] gave an HSG with seed length $\widetilde{O}(\log n \cdot \log 1/\varepsilon)$.
3. Third, we show there is an explicit PRG fooling width w *adaptive* branching programs with seed length $O(\log n \cdot \log^2(nw/\varepsilon))$. Here, the branching program can choose an input bit to read depending on its current state, while it is guaranteed that on any input $x \in \{0,1\}^n$, the branching program reads each input bit exactly once. Previously, no PRG with a non-trivial seed length is known for this class.

   We remark that there are some functions computable by constant-width adaptive branching programs but not by sub-exponential-width unordered branching programs.

In terms of techniques, we indeed show that the Forbes-Kelly PRG (with the right parameters) from [Forbes-Kelly FOCS 2018] already fools all variants of roBPs above. Our proof adds several new ideas to the original analysis of Forbes-Kelly, and we believe it further demonstrates the versatility of the Forbes-Kelly PRG.

2012 ACM Subject Classification Theory of computation → Pseudorandomness and derandomization

Keywords and phrases pseudorandom generators, derandomization, read-once branching programs

Digital Object Identifier 10.4230/LIPIcs.ICALP.2023.39

Category Track A: Algorithms, Complexity and Games

Funding *Lijie Chen*: Supported by a Miller Research Fellowship.
Avishay Tal: Supported by a Sloan Research Fellowship and NSF CAREER Award CCF-2145474.

1 Introduction

One central question in complexity theory is whether randomness is necessary for efficient computation. In the time setting, the question is essentially asking whether $\mathsf{P} = \mathsf{BPP}$. While it is commonly believed that $\mathsf{P} = \mathsf{BPP}$ [23, 16], it is known that establishing this would imply breakthrough lower bounds in complexity theory [13, 17], which seems to be out of reach for current techniques. Therefore, most previous works are devoted to derandomizing sub-classes of BPP. In particular, the class of randomized log-space algorithms (BPL) has attracted a lot of attention, since not only it contains many interesting problems, but also it is indeed possible to give unconditional derandomizations of BPL [22, 27].

A leading approach to derandomize BPL is to construct explicit PRGs for ordered read-once branching programs (see below for a formal definition) with short seed length.

▶ **Definition 1.** *An ordered read-once branching program (roBP) B of length n and width w computes a function $B\colon \{0,1\}^n \to \{0,1\}$. The program has $(n+1)$ layers of states $V_0 \cup V_1 \cup \cdots \cup V_n$ where V_i contains all states in the i-th layer. Being width-w means that $|V_i| \le w$ for every $i \in [n]$. On an input $x \in \{0,1\}^n$, the branching program computes as follows. It starts at a fixed start state $s \in V_0$. Then for every $i = 1, 2, \ldots, n$, it reads the next input bit x_i and updates its state according to a transition function $B_i\colon V_{i-1} \times \{0,1\} \to V_i$ by taking $v_i = B_t(v_{i-1}, x_i)$. Note that the transition function B_i can differ at each time step.*

When we use the program to compute a decision problem, we specify a set $V_{\mathsf{acc}} \subseteq [w]$ of accepting states in the final layer. Let v_n be the final state reached by the branching program on input x. If $v_n \in V_{\mathsf{acc}}$, the branching program accepts, denoted by $B(x) = 1$. Otherwise, the program rejects, denoted by $B(x) = 0$.

Next, we recall the definition of a pseudorandom generator (PRG).

▶ **Definition 2.** *Let $\mathcal{F}$ be a class of functions $f\colon \{0,1\}^n \to \{0,1\}$. An ε-PRG for $\mathcal{F}$ is a function $G\colon \{0,1\}^s \to \{0,1\}^n$ such that for every $f \in \mathcal{F}$,*

$$\left| \Pr_{x \in \{0,1\}^n}[f(x) = 1] - \Pr_{x \in \{0,1\}^s}[f(G(x)) = 1] \right| \le \varepsilon.$$

We say that G ε-fools $\mathcal{F}$ if it is an ε-PRG for $\mathcal{F}$. The input length s is the seed length of the PRG G. We say a generator is explicit*, if given as input a seed $x \in \{0,1\}^s$, the output is computable in space $O(s)$.*

In a seminal work, Nisan constructed an explicit PRG that ε-fools length-n width-w ordered roBPs with seed length $O(\log n \cdot \log(nw/\varepsilon))$. Since then, many PRGs with improved seed lengths were constructed for sub-classes of ordered roBPs (see [5, 10, 20] and the references therein), but Nisan's PRG remains the state-of-the-art even for width-4 general roBPs.

Nisan's PRG (and [15, 9]) crucially relies on the following "communication" argument: The first half of the roBP can only communicate $\log w$ bits (describing the state reached at the end of the first half) to the second half. Due to this, it is possible to reuse all but $\log w$ bits from the seed that is used to generate the first half of the pseudorandom input, when generating the second half of the pseudorandom input. Recursively applying the idea gives the $\log n \log(nw/\varepsilon)$ seed length of Nisan's PRG.

However, some researchers have the feeling that this type of argument is inherently limited to having seed length at least $\log^2 n$ [6, 26, 28][1], and different approaches are required to

[1] For example, in [26], "This paradigm seems unlikely to yield pseudorandom generators for general logspace computations that have a seed length of $O(\log^{1.99} n)$."

overcome this $\log^2 n$ barrier. The search for a different paradigm for designing PRGs has motivated the study of models stronger than normal roBPs, with the hope that studying them would inspire us to find new techniques. In particular, two interesting models, unordered roBPs and unbounded-width roBPs, were introduced recently. It turns out that designing PRGs for both models requires inherently new techniques or analysis compared to Nisan's original PRG (or the INW PRG [15]).

Unordered roBPs. Let $\mathcal{B}$ be a class of ordered roBPs. We say a function $g\colon \{0,1\}^n \to \{0,1\}$ is computable by an unordered $\mathcal{B}$ roBP, if there is a function $f\colon \{0,1\}^n \to \{0,1\}$ and a permutation π on $[n]$ such that f is computable by a roBP in $\mathcal{B}$ and $g(x_1,\ldots,x_n) = f(x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(n)})$.

It is known that Nisan's PRG fails to fool unordered roBPs [29]. After a long line of previous works [4, 14, 26, 28, 11, 19, 7], Forbes and Kelly [8] constructed $O(\log^2 n \log(nw/\varepsilon))$-seed-length PRGs fooling length-n width-w unordered roBPs with error ε.

Unbounded-width roBPs. Another recent line of works studied roBPs with unbounded width [12, 25, 24, 3, 18]. Of course, a general roBP with unbounded width can compute any function (even with a single accept state), so we must restrict our attention to sub-classes of such roBPs. The following two sub-classes of roBPs are the most studied ones in the literature.

▶ **Definition 3.** *Let B be an ordered roBP with length n and width w. We say that B is a regular roBP, if for every $t \in [n]$ and every $v \in [w]$, there are exactly 2 pairs $(u,b) \in [w]\times\{0,1\}$ such that $B_t(u,b) = v$. We say that B is a permutation roBP, if for every $t \in [n]$ and every $b \in \{0,1\}$, $B_t(\cdot,b)$ is a permutation on $[w]$.*

In [12], an $\widetilde{O}(\log n \cdot \log 1/\varepsilon)$-seed length PRG with error ε is constructed for ordered unbounded-width permutation roBP with length-n and a single accept state. A later work [25] (building on a prior work [1]) constructed an $\widetilde{O}(\log n \cdot \sqrt{\log(n/\varepsilon)} + \log(1/\varepsilon))$-seed length weighted PRG for the same class.[2]

1.1 Our Results

In this work, we consider two even stronger models of roBPs: (1) roBPs that are *both* unordered and have unbounded width and (2) roBPs that can read input in an adaptive order (that is, the next bit to read can depend on the current state).

1.1.1 Unordered and Unbounded-width roBPs

Given the recent developments on unordered roBPs and on unbounded-width roBPs, a natural question is whether one can construct non-trivial PRGs for unordered *and* unbounded-width (permutation or regular) roBPs. A prior, it is even unclear whether such a class admits *non-explicit* PRGs with short seed length, since the usual probabilistic argument for the existence of PRGs with short seed length does not apply here [12].

Our first result is a $\mathrm{polylog}(n/\varepsilon)$-seed-length PRG for unordered unbounded-width permutation roBPs with a single accept state, significantly improving the previous $\Omega(n)$-seed length PRGs from [18].

[2] A weighted PRG for a class of functions $\mathcal{F}$ is a pair of functions $G\colon \{0,1\}^s \to \{0,1\}^n$ and $\rho\colon \{0,1\}^s \to \mathbb{R}$ such that $\mathbf{E}_{x\in\{0,1\}^s}[\rho(x)f(G(x))]$ is ε-close to $\mathbf{E}_{x\in\{0,1\}^n}[f(x)]$.

▶ **Theorem 4** (Unbounded width permutation BP). *For all integers n and $\varepsilon > 0$, there is an explicit ε-PRG $G\colon \{0,1\}^s \to \{0,1\}^n$ with seed length*

$$s = O(\log n \cdot \log^2(n/\varepsilon))$$

that fools unordered unbounded-width permutation branching programs with a single accept state.

Our second result is a $\widetilde{O}(\sqrt{n}\log(1/\varepsilon))$-seed-length PRG for unordered unbounded-width regular roBPs with a single accept state. No (even non-explicit) non-trivial PRG is known for this class even in the ordered setting; Bogdanov, Hoza, Prakriya, and Pyne [3] has constructed $\widetilde{O}(\log n \cdot \log(1/\varepsilon))$-seed-length HSG for the ordered case.[3]

▶ **Theorem 5** (Unbounded width regular BP). *For all integers n and $\varepsilon > 0$, there is an explicit ε-PRG $G\colon \{0,1\}^s \to \{0,1\}^n$ with seed length*

$$s = O\left(\sqrt{n \log\left(\frac{n}{\varepsilon}\right)} \cdot \log n\right)$$

that fools unordered unbounded-width regular branching programs with a single accept state.

In terms of techniques, we indeed prove that the Forbes-Kelly PRG suffices for the two theorems above. Our analysis carefully modifies the original analysis from [8]. In fact, we prove that a single round of Forbes-Kelly pseudorandom restriction fools unordered unbounded-width regular roBPs (see Theorem 10). Iterating this restriction $O(\log(n/\varepsilon))$ times proves Theorem 4. Unfortunately, it is unclear whether the same iterative construction fools unordered unbounded-width regular roBPs since they are not closed under restrictions. Still, doing the pseudorandom restriction exactly once with the right parameters proves Theorem 5.

1.1.2 Adaptive roBPs

While an unordered roBP can read its input in any order, it cannot change the ordering based on the input it has read so far (*i.e.*, the order is input oblivious). We also consider an even stronger variant of roBPs, called adaptive roBPs, which are programs that can decide the next bit to read given its current state. We formally define them as follows.

▶ **Definition 6.** *An adaptive read-once branching program B of length n and width w computes a function $B\colon \{0,1\}^n \to \{0,1\}$. The program has states $V_0 \cup V_1 \cup \cdots \cup V_n$ where V_i consists of the w states in the i-th layer. On an input $x \in \{0,1\}^n$, the branching program B computes as follows. It starts at a fixed start state $v_0 \in [w]$. Then for every $t = 1, 2, \ldots, n$, it reads the bit $x_{\mathsf{pos}(t-1,v_{t-1})}$ and updates its state according to a transition function $B_t\colon V_{t-1} \times \{0,1\} \to V_t$ by taking $v_t = B_t(v_{t-1}, x_{\mathsf{pos}(t-1,v_{t-1})})$. Here, $\mathsf{pos}\colon V_0 \cup \cdots \cup V_{n-1} \to [n]$ is a function specifying the index of the next bit to read given the current state v_{t-1}. We require that on every input $x \in \{0,1\}^n$, B reads each bit in x exactly once.*

We remark that adaptive roBPs are strictly stronger than unordered roBPs as shown by an example function $f : \{0,1\} \times \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ as

$$f(b,x,y) = \mathbf{1}[b=0] \cdot \mathbf{1}[x=y] + \mathbf{1}[b=1] \cdot \mathbf{1}[x=y^R].$$

[3] A hitting set generator (HSG) $H\colon \{0,1\}^s \to \{0,1\}$ for a class of functions $\mathcal{F}$ satisfies the following: for every $f \in \mathcal{F}$ such that $\Pr_{x\in\{0,1\}^n}[f(x)=1] > \varepsilon$, there exists $z \in \{0,1\}^s$ such that $f(H(z)) = 1$. Note that a PRG is automatically an HSG, while the converse may not hold.

Here, y^R denotes the reversed string of y. Observe that there is a constant-width adaptive roBP for f. The program first reads b. If $b = 0$, the program reads and compares x and y bit by bit. Otherwise, the program compares x and y^R bit by bit. Moreover, it is easy to see (via a communication complexity argument) that every unordered roBP for f requires exponential width.

Our third result gives $O(\text{polylog}(nw/\varepsilon))$-seed-length PRG for adaptive roBPs. To the best of our knowledge, no explicit PRGs with seed length less than n was known prior to our work.

▶ **Theorem 7.** *For every $n, w \geq 1$ and $\varepsilon > 0$, there is an explicit ε-PRG $G : \{0,1\}^s \to \{0,1\}^n$ fooling width-w adaptive roBPs with seed length $s = O(\log n \cdot \log^2(nw/\varepsilon))$.*

We prove Theorem 7 by adapting the argument in [8]. The key observation allowing us to do so is the following. Suppose B satisfies the read-once promise. Then, for every vertex $v \in V_i$, if we denote by Pre_v (resp. Post_v) the set of possible variables read in any path from the starting state to v (resp. v to the accepting state). It must be the case that Pre_v and Post_v are disjoint for every v. By a delicate argument (Claim 15), we show that this disjointness property is sufficient for applying the key technique of Forbes-Kelley proof: decomposing the branching program by high/low-degree Fourier terms.

Moreover, when the width w of the adaptive roBP is small, we can show that the branching program has bounded Fourier growth (following [7]). In particular, we show that the L-th level Fourier mass of a width-w adaptive roBP is bounded by $O(\log(nw))^{2Lw}$. As shown in [8], for programs with bounded Fourier growth, we can further improve the seed length by a $\log(n)$ factor. Formally, we show

▶ **Theorem 8.** *For every $n, w \geq 1$ and $\varepsilon > 0$, there is an explicit ε-PRG $G : \{0,1\}^s \to \{0,1\}^n$ fooling width-w adaptive roBPs with seed length $s = \widetilde{O}(w \log^2(n/\varepsilon))$.*

Theorem 8 is a direct corollary of the new Fourier growth bound. We briefly comment on how we get the Fourier growth of $O(\log n)^{2Lw}$ for adaptive roBP. Roughly speaking, given a width-w, length-n adaptive roBP B, we construct a related witdh-$2w$, length-n^2 *oblivious* roBP B', such that the Fourier spectrum of B is "dominated" by that of B'. The idea is simple: we duplicate each input of B for n times and get n^2 bits. Now, it is easy to construct a width-$2w$ oblivious roBP running on the n^2 bits to simulate B. (Essentially, the n^2 bits allow us to make n passes over the input, we can use each pass to implement one step of transition of B.)

Although B' has n^2 input bits, we can exploit the promise that B is read-once, and prove the following nice property: For any input $z \in \{0,1\}^{n^2}$, $B'(z)$ depends only on n bits of z (that is to say, there is a subset of n bits from z, such that flipping all other bits of z cannot change the output). This allows us to connect the Fourier weights of B' and B. The details can be found in Appendix A.

2 Preliminaries

For a Boolean predicate P, we use $\mathbf{1}_{\{P\}}$ to denote the indicator of P, which takes value 1 if P holds, value 0 otherwise. We often use U_n to denote the uniform distribution over $\{0,1\}^n$ (when n is clear from the context, we will just write U for simplicity), and $U(X)$ to denote the uniform distribution over a set X. For two strings $\alpha, \beta \in \{0,1\}^n$, we use $\alpha \wedge \beta$ and $\alpha + \beta$ to denote their bit-wise AND and bit-wise XOR, respectively. Similarly, for two distributions D_1, D_2, we use $D_1 \wedge D_2$ (resp. $D_1 + D_2$) to denote the distributions obtained by drawing $\alpha \sim D_1$ and $\beta \sim D_2$ and outputting $\alpha \wedge \beta$ (resp. $\alpha + \beta$).

We always work with the $\{-1,1\}^n$ basis for Boolean function analysis. For a function $f\colon \{-1,1\}^n \to \mathbb{R}$, recall that its Fourier characters indexed by $\alpha \subseteq [n]$, is defined by

$$\widehat{f}(\alpha) = \mathop{\mathbb{E}}_{x\in\{-1,1\}^n}\left[f(x)\cdot\prod_{i\in\alpha} x_i\right].$$

We often use greek letters (such as α, β, γ) to index Fourier characters.

We will need k-wise independent and γ-almost k-wise independent distributions throughout the paper, which look locally uniform and thus fool functions that only depend on a few bits.

▶ **Definition 9.** *Let D be a distribution over $\{0,1\}^n$. We say D is k-wise independent if, for every $f\colon \{0,1\}^n \to [-1,1]$ that depends on at most k bits, we have*

$$\mathop{\mathbb{E}}_{D} f(D) = \mathop{\mathbb{E}}_{U} f(U).$$

If D merely satisfies

$$\left|\mathop{\mathbb{E}}_{D} f(D) - \mathop{\mathbb{E}}_{U} f(U)\right| \le \gamma$$

for every such f, we say that D is γ-almost k-wise independent.

It is possible to sample from a k-wise independent distribution using $O(k\cdot\log n)$ random bits ([30]) and from a γ-almost k-wise independent distribution using $O(k+\log\log n+\log 1/\gamma)$ random bits ([21, 2]).

3 PRGs for Unbounded-width Branching Programs

In this section, we will prove the following theorem, which shows that one round of pseudorandom restriction fools regular branching programs with unbounded width and a single accept state.

▶ **Theorem 10.** *Let B be an unbounded-width regular branching program of length n with starting state $s \in V_0$ and a single accept state $t \in V_n$. Let D, U denote a $2k$-wise independent distribution and a uniform distribution over $\{0,1\}^n$, respectively. Let $T^{(a)}$ denote a $2k$-wise independent distribution over $[a]^n$, and let distribution T be defined as $T_i = \mathbf{1}_{\{T_i^{(a)}=1\}}$ for all $i \in [n]$. Then*

$$|\mathbf{E}[B(U)] - \mathbf{E}[B(D + T \wedge U)]| \le n\cdot(1-1/a)^{k/2}.$$

Since the class of permutation BPs is closed under restrictions, we can iteratively apply Theorem 10 to it with $k = \log(n)$ and $a = 2$. The immediate consequence is that we get a PRG for unordered unbounded-width permutation branching programs.

▶ **Corollary 11** (Restating Theorem 4)**.** *For all integers n and $\varepsilon > 0$, there is an explicit ε-PRG $G\colon \{0,1\}^s \to \{0,1\}^n$ with seed length*

$$s = O(\log n\cdot\log^2(n/\varepsilon))$$

that fools unordered unbounded-width permutation branching programs with a single accept state.

However, when it comes to regular branching programs, this class is no longer closed under restrictions. Hence we can only apply Theorem 10 once and set $k = \widetilde{O}(\sqrt{n})$ and $a = \widetilde{O}(\sqrt{n})$. It remains an interesting open problem to apply iterative restriction for unbounded-width regular branching programs.

▶ **Corollary 12** (Restating Theorem 5). *For all integers n and $\varepsilon > 0$, there is an explicit ε-PRG $G\colon \{0,1\}^s \to \{0,1\}^n$ with seed length*

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

that fools unordered unbounded-width regular branching programs with a single accept state.

We will prove Theorem 10 in Subsection 3.1 and Subsection 3.2. In Subsection 3.3, we prove Corollary 11 and Corollary 12.

3.1 Fourier Decomposition of Regular BPs

Recall that V_i is the set of nodes in the i-th level of our branching program. $s \in V_0$ is the starting point and $t \in V_n$ is the unique accepting state. $x \in \{0,1\}^n$ is the input to our branching program B. In order to work with $\{-1,1\}$ basis, we let $y_i = (-1)^{x_i}$ for all $i \in [n]$.

For any two nodes $a \in V_i$ and $b \in V_j$. We define the indicator $P_{a,b} : \{-1,+1\}^n \to \{0,1\}$,

$$P_{a,b}(y) = \begin{cases} 1 & \text{Starting from } a, \text{ we reach at node } b \text{ on inputs } x_{i+1}, \ldots, x_j; \\ 0 & \text{Otherwise.} \end{cases}$$

Its Fourier expansion is as follows:

$$P_{a,b}(y) = \sum_{\alpha \subseteq \{i+1, i+2, \ldots, j\}} \widehat{P}_{a,b}(\alpha) \cdot \chi_\alpha(y)$$

where the Fourier characters are defined as

$$\chi_\alpha(y) = \prod_{i \in \alpha} y_i.$$

This naturally extends $P_{a,b}$ to $\mathbb{R}^n \to \mathbb{R}$.

Furthermore, we define

$$\bar{P}^{[k]}_{a,b}(y) = \sum_{\substack{\alpha \subseteq \{i+1, i+2, \ldots, j\} \\ |\alpha| = k,\ j \in \alpha}} \widehat{P}_{a,b}(\alpha) \cdot \chi_\alpha(y)$$

which is the sum all the degree k terms that contain y_j.

We also define

$$P^{(k)}_{a,b}(y) = \sum_{\substack{\alpha \subseteq \{i+1, i+2, \ldots, j\} \\ |\alpha| = k,\ i+1 \in \alpha}} \widehat{P}_{a,b}(\alpha) \cdot \chi_\alpha(y)$$

which is the sum all the degree k terms that contain y_{i+1}.

▶ **Fact 13.** *Let D, T, U be the distributions defined in Theorem 10, and let G be a distribution defined as*

$$G_i = \begin{cases} (-1)^{D_i} & T_i = 0, \\ 0 & T_i = 1. \end{cases} \quad \forall i \in [n].$$

Then, we have $\mathbf{E}[B(U)] = \widehat{P}_{s,t}(\varnothing)$ and $\mathbf{E}[B(D + T \wedge U)] = \mathbf{E}_{y \sim G}[P_{s,t}(y)]$.

Proof. Notice that when $y_i = (-1)^{x_i}$ for all i, $B(x) = P_{s,t}(y)$. The first fact holds because for all $\alpha \neq \varnothing$, we have $\mathbf{E}_{y\sim U(\{\pm 1\}^n)}[\chi_\alpha(y)] = 0$. Hence $\mathbf{E}[B(U)] = \mathbf{E}_{y\sim U(\{\pm 1\}^n)}[P_{s,t}(y)] = \widehat{P}_{s,t}(\varnothing)$.

For the second fact, conditioned on an instantiation of T, we define an intermediate distribution G' as

$$G'_i = \begin{cases} (-1)^{D_i} & T_i = 0, \\ (-1)^{U_i} & T_i = 1. \end{cases}.$$

we know that $\mathbf{E}[B(D + T \wedge U)] = \mathbf{E}_{y\sim G'}[P_{s,t}(y)]$.

- When $T_i = 1$, we have $\mathbf{E}_{y\sim G'}[y_i \mid T_i = 1] = \mathbf{E}_{y\sim G}[y_i \mid T_i = 1] = 0$ since y_i is sampled uniformly and independently from $\{\pm 1\}$.
- When $T_i = 0$, we always have $G_i = G'_i = (-1)^{D_i}$.

Hence for all α, we know that

$$\mathbf{E}_{y\sim G'}[\chi_\alpha(y)] = \mathbf{E}_{y\sim G'}\left[\prod_{i\in\alpha} y_i\right] = \mathbf{E}_T\left[\prod_{\substack{i\in\alpha \\ T_i=1}} \mathbf{E}_{y\sim G'}\left[y_i \mid T_i = 1\right] \cdot \mathbf{E}_{y\sim G'}\left[\prod_{\substack{i\in\alpha \\ T_i=0}} y_i \,\middle|\, T\right]\right]$$

$$= \mathbf{E}_T\left[\prod_{\substack{i\in\alpha \\ T_i=1}} \mathbf{E}_{y\sim G}\left[y_i \mid T_i = 1\right] \cdot \mathbf{E}_{y\sim G}\left[\prod_{\substack{i\in\alpha \\ T_i=0}} y_i \,\middle|\, T\right]\right] = \mathbf{E}_{y\sim G}[\chi_\alpha(y)].$$

As a result, $\mathbf{E}_{y\sim G'}[P_{s,t}(y)] = \mathbf{E}_{y\sim G}[P_{s,t}(y)]$. This finishes the proof. ◀

3.2 Bounding the Error

In the error analysis, we follow the approach of Forbes and Kelley [8]. By Fact 13, the result we wish to prove is equivalent to

$$\left|\mathbf{E}_{y\sim G}[P_{s,t}(y)] - \widehat{P}_{s,t}(\varnothing)\right| \leq n \cdot (1 - 1/a)^{k/2}.$$

In the analysis of [8], they considered the decomposition,

$$L_k(y) = \sum_{\substack{\alpha\subseteq\{1,2,\ldots,n\} \\ 0<|\alpha|<k}} \widehat{P}_{s,t}(\alpha) \cdot \chi_\alpha(y),$$

$$P_{s,t}(y) - \widehat{P}_{s,t}(\varnothing) = L_k(y) + \sum_{i=1}^{n} \sum_{m\in V_i} \bar{P}^{[k]}_{s,m}(y) \cdot P_{m,t}(y).$$

Here $L_k(y)$ are the low-degree terms, and $\bar{P}^{[k]}_{s,m}(y) \cdot P_{m,t}(y)$ are the terms that reaches degree k exactly at node $m \in V_i$. The intuition is that the $2k$-wise independent distribution D fools $L_k(y)$ while the high-degree terms are fooled by $T \wedge U$.

However, in order to work for unbounded-width regular branching programs, we have to consider a different decomposition. Let $L_k(y)$ be the same as before. We have

$$P_{s,t}(y) - \widehat{P}_{s,t}(\varnothing) = L_k(y) + \sum_{i=1}^{n} \sum_{m\in V_i} P_{s,m}(y) \cdot P^{(k)}_{m,t}(y).$$

Observe that we are using $P^{(k)}_{m,t}(y)$ instead of $\bar{P}^{[k]}_{m,t}(y)$. The benefit of this decomposition is that now for all y, we have

$$\sum_{m\in V_i} P_{s,m}(y)^2 \leq 1,$$

since from s only one state m can be reached under input y. In contrast, in the original decomposition, $\sum_{m\in V_i} P_{m,t}(y)^2$ could be very large. This difference will be essential in our analysis.

Now we are ready to prove Theorem 10.

Proof of Theorem 10. By our decomposition, we know

$$\left|\mathbf{E}_{y\sim G}[P_{s,t}(y)] - \widehat{P}_{s,t}(\varnothing)\right| \le |\mathbf{E}_{y\sim G}[L_k(y)]| + \sum_{i=1}^{n}\sum_{m\in V_i}\left|\mathbf{E}_{y\sim G}\left[P_{s,m}(y)\cdot P^{(k)}_{m,t}(y)\right]\right|. \tag{1}$$

Since G is k-wise independent, we know $\mathbf{E}_{y\sim G}\left[L_k(y)\right] = 0$. Now we bound the second term. We will need the following fact: For any two (not necessarily independent) sequences of random variables $\{f_m\}_{m\in V_i}, \{g_m\}_{m\in V_i}$, we have

$$\mathbf{E}\left[\sum_{m\in V_i} f_m g_m\right] \le \mathbf{E}\left[\sum_{m\in V_i} f_m^2\right]^{1/2} \mathbf{E}\left[\sum_{m\in V_i} g_m^2\right]^{1/2}.$$

This is the Cauchy-Schwarz Inequality for random variables.

Let $f_m = |P_{s,m}(y)|$ and $g_m = |P^{(k)}_{m,t}(y)|$. We have

$$\sum_{m\in V_i}\left|\mathbf{E}_{y\sim G}\left[P_{s,m}(y)P^{(k)}_{m,t}(y)\right]\right| \le \mathbf{E}_{y\sim G}\left[\sum_{m\in V_i}\left(P_{s,m}(y)\right)^2\right]^{1/2} \mathbf{E}_{y\sim G}\left[\sum_{m\in V_i} P^{(k)}_{m,t}(y)^2\right]^{1/2}$$

We bound these two separately.

We first bound $\mathbf{E}_{y\sim G}\left[\sum_{m\in V_i} P^{(k)}_{m,t}(y)^2\right]$. Suppose

$$P^{(k)}_{m,t}(y) = \sum_{\alpha} c_\alpha \chi_\alpha(y).$$

By $2k$-wise independence of G, we know for all $\alpha \neq \beta$ and $|\alpha| + |\beta| \le 2k$, the cross term

$$\mathbf{E}_{y\sim G}[\chi_\alpha(y)\chi_\beta(y)] = 0.$$

For the square terms, notice that for all $T_i = 1$, we have $y_i = 0$. When $T_i = 0$, $y_i = (-1)^{D_i}$. $T_i = 1$ happens with probability $1/a$. D, T are $2k$-wise independent. Hence when $|\alpha| = k$, we have[4]

$$\begin{aligned}\mathbf{E}_{y\sim G}[\chi_\alpha(y)^2] &= \mathbf{E}_{y\sim D}[\chi_\alpha(y)^2\cdot \mathbf{1}_{\{\forall i\in\alpha, T_i=0\}}]\\ &= \left(1-\frac{1}{a}\right)^k \cdot \mathbf{E}_{y\sim U}[\chi_\alpha(y)^2]\\ &= \left(1-\frac{1}{a}\right)^k.\end{aligned}$$

Hence,

$$\begin{aligned}\mathbf{E}_{y\sim G}\left[\left(P^{(k)}_{m,t}(y)\right)^2\right] &= \mathbf{E}_{y\sim G}\left[\sum_{|\alpha|=k} c_\alpha^2\chi_\alpha(y)^2\right]\\ &= \left(1-\frac{1}{a}\right)^k \sum_{|\alpha|=k} c_\alpha^2 \le \left(1-\frac{1}{a}\right)^k \mathbf{E}_{y\sim U}\left[\left(P_{m,t}(y)\right)^2\right].\end{aligned}$$

[4] For brevity, we use $y\sim U$ to mean that $y\sim U(\{-1,1\}^n)$.

The last step follows from Parseval identity. Summing over all $m \in V_i$ for a fixed i, we get

$$\sum_{m\in V_i} \mathbf{E}_{y\sim G}\left[\left(P^{(k)}_{m,t}(y)\right)^2\right] \le \left(1-\frac{1}{a}\right)^k \sum_{m\in V_i} \mathbf{E}_{y\sim U}\left[\left(P_{m,t}(y)\right)^2\right]$$
$$= \left(1-\frac{1}{a}\right)^k \sum_{m\in V_i} \mathbf{E}_{y\sim U}\left[P_{m,t}(y)\right] = \left(1-\frac{1}{a}\right)^k.$$

The last step is because now $y \sim U$, and the branching program is regular so that $\sum_{m\in V_i} \mathbf{E}_{y\sim U}[P_{m,t}(y)] = 1$.[5]

- On the other hand, as we mentioned, for any y, s can reach a single vertex in V_i, hence

$$\sum_{m\in V_i} \mathbf{E}_{y\sim G}\left[P_{s,m}(y)^2\right] \le 1.$$

Putting these two together, we get

$$|\mathbf{E}_{y\sim G}[P_{s,t}(y)] - \widehat{P}_{s,t}(\varnothing)| \le \sum_{i=1}^{n} \mathbf{E}_{y\sim G}\left[\sum_{m\in V_i}\left(P_{s,m}(y)\right)^2\right]^{1/2} \cdot \mathbf{E}_{y\sim G}\left[\sum_{m\in V_i} P^{(k)}_{m,t}(y)^2\right]^{1/2}$$
$$\le \left(1-\frac{1}{a}\right)^{k/2} n. \qquad \blacktriangleleft$$

3.3 Applications

Finally, we prove Corollary 11 and Corollary 12 in the rest of this section.

Proof of Corollary 11. Let $\{D^{(i)}\}_{i\in[\ell]}, \{T^{(i)}\}_{i\in[\ell]}$ be ℓ independent copies of $2k$-wise independent dsitributions defined in Theorem 10 with $k = \log\left(\frac{n}{\varepsilon}\right) + \log\log(\frac{n}{\varepsilon}) + 1$ and $a = 2$.

We construct pseudorandom distributions $G^{(0)}, G^{(1)}, \ldots, G^{(\ell)}$ with $\ell = \Theta(\log(n/\varepsilon))$. We let G_0 be the set of all one strings in $\{0,1\}^n$ and set

$$G^{(i+1)} = D^{(i)} + T^{(i)} \wedge G^{(i)}.$$

Let branching program $B^{(i)}$ be defined as $B^{(\ell)} = B$ and

$$B^{(i)}(x) = B^{(i+1)}(D^{(i)} + T^{(i)} \wedge x).$$

Since any restriction of a permutation branching program is still a permutation branching program. For any realization of $D^{(i)}$ and $T^{(i)}$, Theorem 10 says that,

$$\left|\mathbf{E}[B^{(i+1)}(U)] - \mathbf{E}[B^{(i)}(U)]\right| = \left|\mathbf{E}_{x\sim D^{(i)}+T^{(i)}\wedge U}[B^{(i)}(x)] - \mathbf{E}[B^{(i)}(U)]\right|$$
$$\le \left(1-\frac{1}{2}\right)^{\log\left(\frac{n}{\varepsilon}\right)+\log\log\left(\frac{n}{\varepsilon}\right)+1} n \le \frac{\varepsilon/2}{\log(\frac{n}{\varepsilon})}.$$

From a standard Chernoff bound, with probability at least $1-\varepsilon/2$, $T^{(1)} \wedge T^{(2)} \wedge \cdots \wedge T^{(\ell)} = 0000\ldots0$. This implies that $\left|\mathbf{E}[B^{(0)}(U)] - \mathbf{E}_{x\sim G^{(0)}}[B^{(0)}(x)]\right| \le \varepsilon/2$ since $B^{(0)}$ does not depend on its input when $T^{(1)} \wedge T^{(2)} \wedge \cdots \wedge T^{(\ell)} = 0000\ldots0$.

[5] This is the only place we use the regularity of the program.

On the other hand, by definition, we know $\mathbf{E}_{B^{(0)},x\sim G^{(0)}}[B^{(0)}(x)] = \mathbf{E}_{x\sim G^{(\ell)}}[B(x)]$. Hence a hybrid argument proves that

$$\begin{aligned}|\mathbf{E}_{x\sim G^{(\ell)}}[B(x)] - \mathbf{E}[B(U)]| &= \left|\mathbf{E}_{x\sim G^{(0)}}[B^{(0)}(x)] - \mathbf{E}[B^{(0)}(U)]\right| + \left|\mathbf{E}[B^{(0)}(U)] - \mathbf{E}[B^{(\ell)}(U)]\right| \\ &\le \varepsilon/2 + \sum_{i=1}^{\ell}\left|\mathbf{E}[B^{(i-1)}(U)] - \mathbf{E}[B^{(i)}(U)]\right| \\ &\le \varepsilon.\end{aligned}$$

◀

Proof of Corollary 12. For regular branching programs, let D and T be $2k$-wise independent distributions defined in Theorem 10 with $k = 2\sqrt{n\log\left(\frac{n}{\varepsilon}\right)} + \log\left(\frac{1}{\varepsilon}\right) + 2$ and $a = \sqrt{\frac{n}{\log\left(\frac{n}{\varepsilon}\right)}}$. We let D' be another independent copy of D.

We construct pseudorandom distribution $G = D + T \wedge D'$. From Theorem 10, we know that

$$|\mathbf{E}_{x\sim D+T\wedge U}[B(x)] - \mathbf{E}[B(U)]| \le n\cdot\left(1-\frac{1}{a}\right)^k \le \varepsilon/2.$$

Let $N = |\{i \mid T_i = 1\}|$. Since T is $2k$-wise independent,

$$\begin{aligned}\mathbf{E}[N^k] &\le \sum_{i_1,i_2,\ldots,i_k\in[n]} \Pr[T_{i_1} = T_{i_2} = \cdots = T_{i_k} = 1] \\ &= n^k \cdot \Pr_{i_1,i_2,\ldots,i_k\in[n]}[T_{i_1} = T_{i_2} = \cdots = T_{i_k} = 1] \\ &= n^k \cdot \prod_{j=1}^{k} \Pr[T_{i_j} = 1 \mid T_{i_1} = T_{i_2} = \cdots = T_{i_{j-1}} = 1] \\ &\le n^k \cdot \prod_{j=1}^{k} (\Pr[i_j \in \{i_1, i_2, \ldots, i_{j-1}\}] + \Pr[T_{i_j} = 1 \mid i_j \notin \{i_1, i_2, \ldots, i_{j-1}\}]) \\ &\le n^k \cdot \left(\frac{k}{n} + \frac{1}{a}\right)^k\end{aligned}$$

From Markov inequality, we get that $\Pr[N \ge 2k] \le 2\cdot(1/2)^k + 2\cdot(n/(2ak))^k$. By the $2k$ wise independence of D',

$$|\mathbf{E}_{x\sim G}[B(x)] - \mathbf{E}_{x\sim D+T\wedge U}[B(x)]| \le 2\cdot(1/2)^k + 2\cdot(n/(2ak))^k \le \varepsilon/2.$$

The seed length is $3k(\log n + \log a) = O\left(\left(\sqrt{n\log\left(\frac{n}{\varepsilon}\right)} + \log\left(\frac{1}{\varepsilon}\right)\right)\cdot\log n\right)$. ◀

▶ Remark 14. We believe the seed length in Corollary 11 can be improved to $O(\log^2 n\cdot\log(n/\varepsilon))$ following the sharper analysis in Section 7.1 of [8]. However, for the simplicity of presentation, we choose to only present it for the seed length of $O(\log n\cdot\log^2(n/\varepsilon))$.

4 PRGs for Adaptive Branching Programs

In this section, we prove our results for adaptive roBPs.

4.1 Decomposition of roBPs

As before, We use $B : \{0,1\}^n \to \{0,1\}$ to denote the adaptive branching program we are analyzing and use $P : \{\pm1\}^n \to \{0,1\}$ to denote the function computed by BP over $\{\pm1\}$ basis. For every input $x \in \{0,1\}^n$, define $y \in \{\pm1\}^n$ as $y_i = (-1)^{x_i}$ for every $i \in [n]$, and

define $P(y) = B(x)$. For any state v in the program, we denote by pos_v the index of the variable queried on state v. We have two outgoing edges from v, one marked with $x_{\mathsf{pos}_v} = 0$ and another with $x_{\mathsf{pos}_v} = 1$.

For any state v in the program, we denote by Pre_v the set of variables read in any path from the starting state to v, and by Post_v the set of variables read in any path from v to the accepting state. We observe that Pre_v and Post_v are disjoint as otherwise there exists a path from the starting state to the accepting state (and passes through v) and reads the same variable twice.

Formally, suppose there exists a vertex v and an index $i \in \mathsf{Pre}_v \cap \mathsf{Post}_v$. We choose a computation path π from starting vertex v_0 to v that queries the set $S \subseteq [n]$ of variables, and a path π' from v to the final layer that queries the set $T \subseteq [n]$, where $i \in S$.

We can construct an input $x \in \{0,1\}^n$ that guides the program to follow the computational path of $\pi \circ \pi'$. Each time the program reads a variable x_j, if x_j has been queried before, this clearly violates the read-once requirement. Otherwise, we can set x_j to make the program follow the path of $\pi \circ \pi'$. However, since know x_i is queried at least twice along the path, there must be some point, where the "read-once" requirement is violated.

Define $P : \{-1,1\}^n \to \{0,1\}$ as the function computed by the program. For a state v in the branching program, we denote by $P_{\to v}$ the event that the path from the starting state passes through v. Note that $P_{\to v}$ can be described as a branching program on the variables Pre_v. We denote by $P_{v\to}$ the sub-program of P starting at v. Note that $P_{v\to}$ can be described as a branching program on the variables Post_v.

4.1.1 Fourier Decomposition for Adaptive BP

Recall that the Fourier representation of any function $f : \{\pm 1\}^n \to \mathbb{R}$ is $\sum_{\alpha\subseteq[n]} \widehat{f}(\alpha)\chi_\alpha(y)$ where $\chi_\alpha(y) = \prod_{i\in\alpha} y_i$ and $\widehat{f}(\alpha) = \mathbf{E}_{y\sim\{\pm1\}^n}[f(y)\cdot\chi_\alpha(y)]$. Furthermore, we have that $\mathbf{E}_{y\sim\{\pm1\}^n}[f(y)^2] = \sum_\alpha \widehat{f}(\alpha)^2$ and $\mathbf{E}_{y\sim\{\pm1\}^n}[f(y)] = \widehat{f}(\emptyset)$.

Let $k \in \mathbb{N}$. Let α be a set of size $\ell > k$. We express $\widehat{P}(\alpha)$ as a sum of products of Fourier coefficients, where the Fourier coefficients come from sub-programs of B. In particular, we have the following claim.

▷ Claim 15. We have

$$\widehat{P}(\alpha) = \sum_{v:|\mathsf{Pre}_v\cap\alpha|=k,\mathsf{pos}_v\in\alpha} \widehat{P_{\to v}}(\alpha\cap\mathsf{Pre}_v)\cdot\widehat{P_{v\to}}(\alpha\cap\mathsf{Post}_v).$$

Proof. By definition

$$\widehat{P}(\alpha) = \mathbf{E}_{y\sim\{\pm1\}^n}[P(y)\cdot\chi_\alpha(y)] = \mathbf{E}_{y\sim\{\pm1\}^n}[P(y)\cdot\mathbf{1}_{\{B \text{ on } y \text{ reads all the variables in } \alpha\}}\cdot\chi_\alpha(y)]$$

where the second equality holds due to the following reason. For any $\beta \subseteq \alpha$ let X_β be the set of strings on which the program reads β and doesn't read $\alpha \setminus \beta$. We note that if $x \in X_\beta$ for some set β which is a strict subset of α, then also $x' := x \oplus e_i$ for $i \in \alpha \setminus \beta$ is in X_β, since the path for both x and x' will be the same (as the path doesn't query x_i). We see that the inputs in X_β can be partitioned to pairs, and each pair contributed 0 to $\mathbf{E}_{y\sim\{\pm1\}^n}[\chi_\alpha(y)]$.

For any x for which $B(x)$ reads all the variables in α, there is a unique state v along the path such that B reads exactly k variables in α before v, and B reads the $k+1$ variable from α immediately on the edge that goes out from v.

Thus, we can partition these paths according to the state v. We observe that v is the state immediately before reading the $k+1$ variable in α if $|\mathsf{Pre}_v \cap \alpha| = k$ and if $\mathsf{pos}_v \in \alpha$. This gives

$$\begin{aligned}\widehat{P}(\alpha) &= \mathbf{E}_{y\sim\{\pm1\}^n}\left[\sum_{v:|\mathsf{Pre}_v\cap\alpha|=k,\mathsf{pos}_v\in\alpha} P_{\to v}(y)P_{v\to}(y)\cdot\chi_\alpha(y)\cdot\mathbf{1}_{\{B \text{ reads all the variables in } \alpha\}}\right]\\ &= \sum_{v:|\mathsf{Pre}_v\cap\alpha|=k,\mathsf{pos}_v\in\alpha} \mathbf{E}_{y\sim\{\pm1\}^n}\left[P_{\to v}(y)P_{v\to}(y)\cdot\chi_\alpha(y)\right]\cdot\mathbf{1}_{\{B \text{ reads all the variables in } \alpha\}}\big]\\ &= \sum_{v:|\mathsf{Pre}_v\cap\alpha|=k,\mathsf{pos}_v\in\alpha} \mathbf{E}_{y\sim\{\pm1\}^n}[P_{\to v}(y)P_{v\to}(y)\cdot\chi_\alpha(y)]\end{aligned}$$

where the last equality follows from the same argument as before by observing that $P_{\to v}(y)P_{v\to}(y)$ is equivalent to the indicator of a program B' that checks that we passed through v and reached the accept state of B.

Now, for every state $v : |\mathsf{Pre}_v \cap \alpha| = k, \mathsf{pos}_v \in \alpha$, we get

$$\begin{aligned}&\mathbf{E}_x[P_{\to v}(y)P_{v\to}(y)\cdot\chi_\alpha(y)]\\ &= \mathbf{E}_{y\in\{0,1\}^{\mathsf{Pre}_v}}[P_{\to v}(y)\chi_{\alpha\cap\mathsf{Pre}_v}(y)]\cdot\mathbf{E}_{y\in\{0,1\}^{\mathsf{Post}_v}}[P_{v\to}(y)\chi_{\alpha\cap\mathsf{Post}_v}(y)]\\ &= \widehat{P_{\to v}}(\alpha\cap\mathsf{Pre}_v)\cdot\widehat{P_{v\to}}(\alpha\cap\mathsf{Post}_v),\end{aligned}$$

which completes the proof. ◁

By summing over all sets of size larger than k we get

$$\begin{aligned}&\sum_{\alpha:|\alpha|>k}\widehat{P}(\alpha)\chi_\alpha(y)\\ &= \sum_{\alpha,v:|\mathsf{Pre}_v\cap\alpha|=k,\mathsf{pos}_v\in\alpha}\widehat{P_{\to v}}(\alpha\cap\mathsf{Pre}_v)\chi_{\alpha\cap\mathsf{Pre}_v}(y)\cdot\widehat{P_{v\to}}(\alpha\cap\mathsf{Post}_v)\chi_{\alpha\cap\mathsf{Post}_v}(y)\\ &= \sum_v\left(\sum_{\alpha':\alpha'\subseteq\mathsf{Pre}_v,|\alpha'|=k}\widehat{P_{\to v}}(\alpha')\chi_{\alpha'}(y)\right)\cdot\left(\sum_{\alpha'':\alpha''\subseteq\mathsf{Post}_v,\mathsf{pos}_v\in\alpha''}\widehat{P_{\to v}}(\alpha'')\chi_{\alpha''}(y)\right).\end{aligned}$$

For each v we denote

$$H_v(y) := \sum_{\alpha':\alpha'\subseteq\mathsf{Pre}_v,|\alpha'|=k}\widehat{P_{\to v}}(\alpha')\chi_{\alpha'}(y)$$

and

$$G_v(y) := \sum_{\alpha'':\alpha''\subseteq\mathsf{Post}_v,\mathsf{pos}_v\in\alpha''}\widehat{P_{\to v}}(\alpha'')\chi_{\alpha''}(y).$$

We observe that $G_v(y)$ is the pos_v-Laplacian[6] of $P_{v\to}$. As such, $G_v(y)$ is a bounded function, i.e., $|G_v(y)| \le 1$ for all $y \in \{\pm1\}^n$.

▶ **Lemma 16.** *For any read-once adaptive branching program B, let P be the function computed by B. We have*

$$P(y) = \mathbf{E}[P(U)] + L(y) + \sum_{v\in V} H_v(y)\cdot G_v(y)$$

where $L(y) = \sum_{1\le|\alpha|\le k}\widehat{P}(\alpha)\chi_\alpha(y)$.

[6] Given a function $f : \{\pm1\}^n \to \mathbb{R}$ and index $i \in [n]$. The i-Laplacian of f is defined as a new function $\mathsf{L}_i f(y) := \frac{f(y)-f(y+e_i)}{2}$, where e_i denotes the i-th unit vector. Observe that $\mathsf{L}_i f(y) = \sum_{S:i\in S}\widehat{f}(S)\chi_S(y)$.

Proof. Any function can be written in the Fourier representation, i.e.

$$P(y) = \sum_{\alpha \subseteq [n]} \widehat{P}(\alpha) \cdot \chi_\alpha(y).$$

Now, we can partition this sum to the sum of sets of size at least k and the sum of sets of size smaller than k,

$$P(y) = \mathbf{E}[P] + L(y) + H(y)$$

where

$$\mathbf{E}[P] = \widehat{P}(\emptyset), \qquad L(y) = \sum_{1 \le |\alpha| \le k} \widehat{P}(\alpha)\chi_\alpha(y) \qquad \text{and} \qquad H(y) = \sum_{|\alpha| > k} \widehat{P}(\alpha)\chi_\alpha(y).$$

We decompose $H(y)$ by the above decomposition. ◀

4.2 Forbes-Kelley PRG fools Adaptive roBP

In this section, we prove that the Forbes-Kelley PRG fools adaptive roBP. First, the following lemma is the analog of [8, Lemma 6.3] for adaptive roBP.

▶ **Lemma 17.** *Let B be a read-once adaptive branching program of size s. Suppose D, T, and U are independently drawn from a $2(k+1)$-wise independent distribution, a $(k+1)$-wise independent distribution, and the uniform distribution over $\{\pm 1\}^n$, respectively. Then,*

$$|\mathbf{E}[P(U)] - \mathbf{E}[P(D + T \wedge U)]| \le s \cdot 2^{-k/2}.$$

Since we are working over ± 1 basis, $T \wedge U$ is a coordinate-wise operation defined as $(T \wedge U)_i = -1$ if and only if $T_i = U_i = -1$, and $D + (T \wedge U)$ is defined as $(D + (T \wedge U))_i = D_i \times (T \wedge U)_i$.

Proof. We use the Decomposition Lemma (Lemma 16):

$$|\mathbf{E}[P] - \mathbf{E}[P(D + T \wedge U)]| \le |\mathbf{E}[L(D + T \wedge U)]| + \sum_{v \in V} |\mathbf{E}[(H_v \cdot G_v)(D + T \wedge U)]|. \tag{2}$$

The first summand in the RHS of Eq. (2) equals zero since $D + T \wedge U$ fools any χ_α for $|\alpha| \le k$. Namely,

$$\mathbf{E}[L(D + T \wedge U)] = \sum_{0 < |\alpha| \le k} \widehat{P}(\alpha) \cdot \mathbf{E}[\chi_\alpha(D + T \wedge U)] = 0.$$

We bound the second summand in the RHS of Eq. (2) term by term. For each $v \in V$:

$$\begin{aligned}
&|\mathbf{E}_{D,T,U}[(H_v \cdot B_{v\to})(D + T \wedge U)]| \\
&\le \mathbf{E}_{D,T}\left[|\mathbf{E}_U[(H_v \cdot B_{v\to})(D + T \wedge U)]|\right] \\
&= \mathbf{E}_{D,T}\left[|\mathbf{E}_U[H_v(D + T \wedge U)]| \cdot |\mathbf{E}_U[G_v(D + T \wedge U)]|\right] && (3) \\
&\le \mathbf{E}_{D,T}\left[|\mathbf{E}_U[H_v(D + T \wedge U)]|\right] && (4) \\
&\le 2^{-k/2}. && \text{(Claim 18)}
\end{aligned}$$

Here, (3) follows by observing that, for every fixed T and D, $H_v(D + T \wedge U)$ and $G_v(D + T \wedge U)$ are independent. (4) is due to that $B_{v\to}$ is bounded. Finally, the last line utilizes a claim that is to be introduced and proved next.

Overall, we get

$$|\mathbf{E}[B(U)] - \mathbf{E}[B(D + T \wedge U)]| \le \sum_{i, v \in V_i} 2^{-k/2} = s \cdot 2^{-k/2},$$

as desired. ◀

The following claim has been used in the proof of Lemma 17. We show its proof now.

▷ Claim 18. Let $H_v : \{\pm 1\}^m \to \mathbb{R}$ be a function whose Fourier spectrum is k-homogeneous, i.e.,

$$H_v(y) = \sum_{\alpha \subseteq [m]:|\alpha|=k} \widehat{H_v}(\alpha) \cdot \chi_\alpha(y).$$

Let D, T, and U denote a $2k$-wise independent distribution, a k-wise independent distribution, and uniform distribution over $\{0,1\}^n$. Then,

$$\mathbf{E}_{D,T}\left[|\mathbf{E}_U[H_v(D + T \wedge U)]|\right] \leq 2^{-k/2} \cdot \sqrt{\sum_\alpha \widehat{H_v}(\alpha)^2}$$

Proof. We verify the claim by direction calculation.

$$\begin{aligned}
&\left(\mathbf{E}_{D,T}\left[|\mathbf{E}_U[H_v(D + T \wedge U)]|\right]\right)^2 \\
&\leq \mathbf{E}_{D,T}\left[\mathbf{E}_U[H_v(D + T \wedge U)]^2\right] \\
&= \mathbf{E}_{D,T,U,U'}\left[H_v(D + T \wedge U) \cdot H_v(D + T \wedge U')\right] \\
&= \sum_{\alpha,\alpha'} \widehat{H_v}(\alpha) \cdot \widehat{H_v}(\alpha') \cdot \mathbf{E}_{D,T,U,U'}[\chi_\alpha(D + T \wedge U) \cdot \chi_{\alpha'}(D + T \wedge U')] \\
&= \sum_{\alpha,\alpha'} \widehat{H_v}(\alpha) \cdot \widehat{H_v}(\alpha') \cdot \mathbf{E}_D[\chi_\alpha(D)\chi_{\alpha'}(D)] \cdot \mathbf{E}_{T,U,U'}[\chi_\alpha(T \wedge U) \cdot \chi_{\alpha'}(T \wedge U')] \\
&= \sum_\alpha \widehat{H_v}(\alpha)^2 \cdot \mathbf{E}_{T,U,U'}[\chi_\alpha(T \wedge U) \cdot \chi_\alpha(T \wedge U')] \quad + \sum_{\alpha \neq \alpha'} |\widehat{H_v}(\alpha)| \cdot |\widehat{H_v}(\alpha')| \cdot 0 \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (D \text{ is } 2k\text{-wise}) \\
&= \sum_\alpha \widehat{H_v}(\alpha)^2 \cdot \mathbf{E}_T[\mathbf{1}_{\{\alpha \cap T = \emptyset\}}] \\
&= 2^{-k} \cdot \sum_\alpha \widehat{H_v}(\alpha)^2. \qquad\qquad\qquad\qquad\qquad\qquad (T \text{ is } k\text{-wise independent}, |\alpha| = k)
\end{aligned}$$

◁

Finally, let us remark that we can also use δ-almost k-wise independent distributions T, D to construct $D + T \wedge U$. Doing an analysis similar as we have done for Claim 18, one can show that

$$|\mathbf{E}[P(U)] - \mathbf{E}[P(D + T \wedge U)]| \leq s \cdot \left(\sqrt{\gamma} + 2^{-k/2} + \sqrt{\gamma}\left(\sum_\alpha |\widehat{H}(\alpha)|\right)\right).$$

The argument is also similar to the one done in [8, Lemma 7.2]. We omit the detail here. Note that, if we can prove a good upper bound of $\sum_\alpha |\widehat{H}(\alpha)|$, we can hope to construct D, T using γ-almost k-wise independent distributions with larger γ. Recall the seed length to sample a γ-almost distribution is $O(\log(1/\gamma) + k + \log\log(n))$, which is smaller than the seed length to sample a perfect k-wise independent distribution by a $\log(n)$ factor for large γ (e.g., when $\gamma \approx 2^{-k}$).

4.2.1 PRG for Adaptive roBP

Given Lemma 17, we prove Theorem 7 now.

Proof of Theorem 7. The proof is nearly identical to that of Corollary 11.

Let $\{D^{(i)}\}_{i\in[\ell]}, \{T^{(i)}\}_{i\in[\ell]}$ be ℓ independent copies of $2k$-wise independent dsitributions defined in Lemma 17 with $k = \log\left(\frac{n}{\varepsilon}\right) + \log\log(\frac{n}{\varepsilon}) + 1$.

We construct pseudorandom distributions $G^{(0)}, G^{(1)}, \ldots, G^{(\ell)}$ with $\ell = \Theta(\log(n/\varepsilon))$. We let G_0 be the trivial PRG that outputs $1^n \in \{0,1\}^n$. Then we set

$$G^{(i+1)} = D^{(i)} + T^{(i)} \wedge G^{(i)}.$$

Define a branching program $B^{(i)}$ as $B^{(\ell)} = B$ and

$$B^{(i)}(x) = B^{(i+1)}(D^{(i)} + T^{(i)} \wedge x).$$

Note that $B^{(i)}$ is a random variable depending on $D^{(i)}$ and $T^{(i)}$. Since the restriction of an adaptive roBP is still a roBP. For any realization of $D^{(i)}$ and $T^{(i)}$, Lemma 17 says that,

$$\left|\mathbf{E}[B^{(i+1)}(U)] - \mathbf{E}[B^{(i)}(U)]\right| \leq \frac{\varepsilon/2}{\log(\frac{n}{\varepsilon})}.$$

From a standard Chernoff bound, with probability at least $1 - \varepsilon/2$, $T^{(1)} \wedge T^{(2)} \wedge \cdots \wedge T^{(\ell)} = 0000\ldots0$. Conditioning on this event, $B^{(0)}$ does not depend on its input, implying that $\left|\mathbf{E}[B^{(0)}(U)] - \mathbf{E}_{x\sim G^{(0)}}[B^{(0)}]\right| \leq \varepsilon/2$ since $B^{(0)}$.

On the other hand, by definition, we know $\mathbf{E}_{B^{(0)},x\sim G^{(0)}}[B^{(0)}(x)] = \mathbf{E}_{x\sim G^{(\ell)}}[B(x)]$. Hence a hybrid argument proves that

$$\begin{aligned}
|\mathbf{E}_{x\sim G^{(\ell)}}[B(x)] - \mathbf{E}[B(U)]| &= \left|\mathbf{E}_{x\sim G^{(0)}}[B^{(0)}(x)] - \mathbf{E}[B^{(0)}(U)]\right| + \left|\mathbf{E}[B^{(0)}(U)] - \mathbf{E}[B^{(\ell)}(U)]\right| \\
&\leq \varepsilon/2 + \sum_{i=1}^{\ell} \left|\mathbf{E}[B^{(i-1)}(U)] - \mathbf{E}[B^{(i)}(U)]\right| \\
&\leq \varepsilon,
\end{aligned}$$

completing the proof. ◀

References

1. AmirMahdi Ahmadinejad, Jonathan A. Kelner, Jack Murtagh, John Peebles, Aaron Sidford, and Salil P. Vadhan. High-precision estimation of random walks in small space. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 1295–1306. IEEE, 2020.
2. Noga Alon, Oded Goldreich, Johan Håstad, and René Peralta. Simple constructions of almost k-wise independent random variables. *Random Structures & Algorithms*, 3(3):289–304, 1992.
3. Andrej Bogdanov, William M. Hoza, Gautam Prakriya, and Edward Pyne. Hitting sets for regular branching programs. In Shachar Lovett, editor, *37th Computational Complexity Conference, CCC 2022, July 20-23, 2022, Philadelphia, PA, USA*, volume 234 of *LIPIcs*, pages 3:1–3:22. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.
4. Andrej Bogdanov, Periklis A. Papakonstantinou, and Andrew Wan. Pseudorandomness for read-once formulas. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 240–246. IEEE Computer Society, 2011.
5. Mark Braverman, Anup Rao, Ran Raz, and Amir Yehudayoff. Pseudorandom generators for regular branching programs. *SIAM J. Comput.*, 43(3):973–986, 2014.
6. Joshua Brody and Elad Verbin. The coin problem and pseudorandomness for branching programs. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 30–39. IEEE Computer Society, 2010.

7 Eshan Chattopadhyay, Pooya Hatami, Omer Reingold, and Avishay Tal. Improved pseudorandomness for unordered branching programs through local monotonicity. In *STOC*, pages 363–375. ACM, 2018.

8 Michael A Forbes and Zander Kelley. Pseudorandom generators for read-once branching programs, in any order. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 946–955. IEEE, 2018.

9 Anat Ganor and Ran Raz. Space pseudorandom generators by communication complexity lower bounds. In Klaus Jansen, José D. P. Rolim, Nikhil R. Devanur, and Cristopher Moore, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2014, September 4-6, 2014, Barcelona, Spain*, volume 28 of *LIPIcs*, pages 692–703. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2014.

10 Parikshit Gopalan, Raghu Meka, Omer Reingold, Luca Trevisan, and Salil Vadhan. Better pseudorandom generators from milder pseudorandom restrictions. In *Proc. 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 120–129. IEEE, 2012.

11 Elad Haramaty, Chin Ho Lee, and Emanuele Viola. Bounded independence plus noise fools products. *SIAM J. Comput.*, 47(2):493–523, 2018.

12 William M. Hoza, Edward Pyne, and Salil P. Vadhan. Pseudorandom generators for unbounded-width permutation branching programs. In James R. Lee, editor, *12th Innovations in Theoretical Computer Science Conference, ITCS 2021, January 6-8, 2021, Virtual Conference*, volume 185 of *LIPIcs*, pages 7:1–7:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.

13 Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: exponential time vs. probabilistic polynomial time. *Journal of Computer and System Sciences*, 65(4):672–694, 2002.

14 Russell Impagliazzo, Raghu Meka, and David Zuckerman. Pseudorandomness from shrinkage. In *Proc. 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 111–119. IEEE, 2012.

15 Russell Impagliazzo, Noam Nisan, and Avi Wigderson. Pseudorandomness for network algorithms. In *Proc. 26th Annual ACM Symposium on Theory of Computing (STOC)*, pages 356–364, 1994.

16 Russell Impagliazzo and Avi Wigderson. P = BPP if E requires exponential circuits: derandomizing the XOR lemma. In *Proc. 29th Annual ACM Symposium on Theory of Computing (STOC)*, pages 220–229, 1997.

17 Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004.

18 Chin Ho Lee, Edward Pyne, and Salil P. Vadhan. Fourier growth of regular branching programs. In Amit Chakrabarti and Chaitanya Swamy, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2022, September 19-21, 2022, University of Illinois, Urbana-Champaign, USA (Virtual Conference)*, volume 245 of *LIPIcs*, pages 2:1–2:21. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.

19 Chin Ho Lee and Emanuele Viola. More on bounded independence plus noise: Pseudorandom generators for read-once polynomials. *Theory of Computing*, 16:1–50, 2020.

20 Raghu Meka, Omer Reingold, and Avishay Tal. Pseudorandom generators for width-3 branching programs. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 626–637. ACM, 2019.

21 Joseph Naor and Moni Naor. Small-bias probability spaces: efficient constructions and applications. *SIAM Journal of Computing*, 22(4):838–856, 1993.

22 Noam Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992.

23 Noam Nisan and Avi Wigderson. Hardness vs. randomness. *Journal of Computer and System Sciences*, 49(2):149–167, 1994.

24 Edward Pyne and Salil P. Vadhan. Limitations of the impagliazzo-nisan-wigderson pseudorandom generator against permutation branching programs. In Chi-Yeh Chen, Wing-Kai Hon, Ling-Ju Hung, and Chia-Wei Lee, editors, *Computing and Combinatorics – 27th International Conference, COCOON 2021, Tainan, Taiwan, October 24-26, 2021, Proceedings*, volume 13025 of *Lecture Notes in Computer Science*, pages 3–12. Springer, 2021.

25 Edward Pyne and Salil P. Vadhan. Pseudodistributions that beat all pseudorandom generators (extended abstract). In Valentine Kabanets, editor, *36th Computational Complexity Conference, CCC 2021, July 20-23, 2021, Toronto, Ontario, Canada (Virtual Conference)*, volume 200 of *LIPIcs*, pages 33:1–33:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.

26 Omer Reingold, Thomas Steinke, and Salil P. Vadhan. Pseudorandomness for regular branching programs via fourier analysis. In Prasad Raghavendra, Sofya Raskhodnikova, Klaus Jansen, and José D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques – 16th International Workshop, APPROX 2013, and 17th International Workshop, RANDOM 2013, Berkeley, CA, USA, August 21-23, 2013. Proceedings*, volume 8096 of *Lecture Notes in Computer Science*, pages 655–670. Springer, 2013.

27 Michael E. Saks and Shiyu Zhou. BP $_{\mathrm{h}}$space(s) subseteq dspace($s^{3/2}$). *Journal of Computer and System Sciences*, 58(2):376–403, 1999.

28 Thomas Steinke, Salil P. Vadhan, and Andrew Wan. Pseudorandomness and fourier-growth bounds for width-3 branching programs. *Theory Comput.*, 13(1):1–50, 2017.

29 Yoav Tzur. Notions of weak pseudorandomness and gf (2n)-polynomials. *Master's thesis, Weizmann Institute of Science*, 2009.

30 Salil P. Vadhan. *Pseudorandomness*. Foundations and Trends in Theoretical Computer Science. Now Publishers, 2012.

A Fourier Growth of Constant-Width Adaptive roBP

In this appendix, we show that the Fourier growth of width-w adaptive roBP is upper bounded by that of width-$2w$ oblivious roBP. As a corollary, we can use almost k-wise independent primitives in the construction of Forbes-Kelley PRG, which saves the seed length from $O(\log^3(n/\varepsilon))$ to $\widetilde{O}(w\log^2(n/\varepsilon))$ when w is small.

A.1 Reducing Adaptive roBP to Oblivious roBP

We start by proving the following lemma.

▶ **Lemma 19.** *Suppose $B : \{0,1\}^n \to \{0,1\}$ is computed by a width-w adaptive roBP. Then there is a width-$2w$ oblivious roBP $B' : \{0,1\}^{n^2} \to \{0,1\}$ such that the following inequality holds for every $L \geq 1$,*

$$\sum_{\alpha\subseteq[n]:|\alpha|=L} |\widehat{B}(\alpha)| \leq \sum_{\alpha\subseteq[n^2]:|\alpha|=L} |\widehat{B'}(\alpha)|.$$

Proof. For an input $x \in \{0,1\}^{n^2}$ to B', we partition the bits into chunks of length n. Namely,

$$x = ((x_1^1,\ldots,x_1^n),(x_2^1,\ldots,x_2^n),\ldots,(x_n^1,\ldots,x_n^n)).$$

For each $i \in [n]$, we will think of $(x_j^i)_{j\in[n]}$ as n duplicate bits that equal to the i-th input bit to the original program B. Namely, consider a mapping $\sigma\colon \{0,1\}^n \to \{0,1\}^{n^2}$ as

$$\sigma(z) = ((z_1,z_2,\ldots,z_n),\ldots,(z_1,z_2,\ldots,z_n)).$$

Constructing the oblivious program. We construct a width-$2w$ oblivious roBP B' such that $B(x) = B'(\sigma(x))$. To illustrate, in the following, we use $x = (x_1, \ldots, x_n)$ to denote the input of B, and $z = (z_1^1, \ldots, z_n^n)$ to denote the input of B'. Note that if $z = \sigma(x)$, then $z_i^j = x_j$ for every i, j,

We describe the construction now. Note that B' involves $n^2 + 1$ layers and n^2 transitions. For each $i \in [n]$, We use the $((i-1)n+1)$-th to the (in)-th transitions of B' to implement the i-th transition of B.

Recall that the $(i-1)$-th (resp. i-th) layer of B contains states V_{i-1} (resp. V_i). Write $V_{i-1} = \{v_1, \ldots, v_w\}$ and $V_i = \{u_1, \ldots, u_w\}$. We construct $\overline{V}_{i-1} = \{v'_1, \ldots, v'_w, u'_1, \ldots, u'_w\}$. Identify $v'_1, \ldots, v'_w$ with $v_1, \ldots, v_w$, and $u'_1, \ldots, u'_w$ with $u_1, \ldots, u_w$. Recall that each vertex v_j reads one input bit $x_{\mathsf{pos}_{v_j}}$ from x.

We make $n+1$ copies of $\overline{V}_{i-1}$, denoted by $\overline{V}^0_{i-1}, \ldots, \overline{V}^n_{i-1}$. Next, we build a sub-program from $\overline{V}^0_{i-1}$ to $\overline{V}^n_{i-1}$, using inputs $z_i^1, \ldots, z_i^n$. For each $t \in [n]$, we add edges from $\overline{V}^{t-1}_{i-1}$ to $\overline{V}^t_{i-1}$. We let all states of $\overline{V}^{t-1}_{i-1}$ read the variable z_i^t (which is supposed to be x_t if $\sigma(x) = z$). For every v_j such that $\mathsf{pos}_{v_j} = t$, suppose v_j has two out edges to u_{j_0}, u_{j_1} with label 0 and 1. We add two edges from v'_j (in $\overline{V}^{t-1}_{i-1}$) to u'_{j_0} and $u'_{j,1}$ (in $\overline{V}^t_{i-1}$) with label 0 and 1. For every v_j where $\mathsf{pos}_{v_j} \neq t$ and every u_j, the state reads the input and simply ignores it. (operationally, this means we add two edges from the current state to the corresponding state in the next layer.)

Now we have n sub-programs: for each $i \in [n]$, we have a subprogram from $\overline{V}^0_{i-1}$ to $\overline{V}^n_{i-1}$. For each $i \in [n]$, observe that both the "u"-states of $\overline{V}^n_{i-1}$ and the "v"-states of $\overline{V}^0_i$ are identified with states in V_i. We naturally glue each pair of corresponding states together. We also glue "v"-states of $\overline{V}^n_{i-1}$ and "u"-states of $\overline{V}^0_i$ arbitrarily. This way, we construct a larger branching program of length n^2 (from $\overline{V}^0_1$ to $\overline{V}^n_n$) and width $2w$. It is straightforward to verify that $B(x) = B'(\sigma(x))$.

Calculating Fourier weights. Now we verify that B' satisfies the lemma statement. Consider the Fourier spectrum of B':

$$B'(z) = \sum_{\alpha \subseteq [n^2]} \widehat{B'}(\alpha)\chi_\alpha(z).$$

We claim that, for every $\alpha \subseteq [n^2]$ such that there exists $\{k_1 n + i, k_2 n + i\} \subseteq \alpha$ for some $k_1 \neq k_2$ and i, it must be the case that $|\widehat{B'}(\alpha)| = 0$. Indeed, we have

$$\widehat{B'}(\alpha) = \mathbf{E}_{z \sim U_{n^2}}[\chi_\alpha(z) \cdot B'(z)]. \tag{5}$$

We observe that

$$B'(z) = \sum_{\pi : \text{accepting computation path}} \mathbf{1}[B' \text{ on input } z \text{ follows } \pi].$$

Let $z^i_{k_1}, z^i_{k_2}$ be the two variables associated with indices $\{k_1 n + i, k_2 n + i\}$. By the promise that B is read-once, in any computation path π of B', it cannot be the case that both $z^i_{k_1}$ and $z^i_{k_2}$ are used (i.e., at least one of them is ignored in the path). It follows that each path contributes zero to (5). Consequently, $\widehat{B'}(\alpha) = 0$.

Next, we have

$$B(x) = B'(\sigma(x)) = \sum_{\alpha \subseteq [n^2]} \widehat{B'}(\alpha)\chi_\alpha(\sigma(x)).$$

As we have shown, $\widehat{B'}(\alpha)$ is non-zero only when α does not contain two variables $z^i_{k_1}, z^i_{k_2}$ in the same group k. For every such α, $\chi_\alpha(\sigma(x)) = \chi_{\Pi(\alpha)}(x)$ where Π denotes the projection of α onto $[n]$. Namely, $\Pi(\alpha)_i = 1$ if and only if $\alpha_{kn+i} = 1$ for some $k \in [n]$. It follows that $|\alpha| = |\Pi(\alpha)|$. Finally, applying the triangle inequality gives

$$\sum_{\beta \subseteq [n]:|\beta|=L} |\widehat{B}(\beta)| \leq \sum_{\alpha \subseteq [n^2]:|\alpha|=L} |\widehat{B'}(\alpha)|,$$

as desired. ◀

A.2 Fourier Growth and Pseudorandomness

Chattopadhyay, Hatami, Reingold and Tal [7] proved the following Fourier growth bound for width-w oblivious roBP.

▶ **Theorem 20** ([7]). *Suppose $B : \{0,1\}^n \to \{0,1\}$ is computed by a width-w oblivious roBP. Then, for every $k \geq 1$, it holds that*

$$\sum_{\alpha:|\alpha|=k} |\widehat{B}(\alpha)| \leq O(\log(nw))^{wk}.$$

As a direct corollary from Lemma 19 and Theorem 20, we obtain the following Fourier growth bound for width-w adaptive roBP.

▶ **Corollary 21.** *Suppose $B : \{0,1\}^n \to \{0,1\}$ is computed by a width-w adaptive roBP. Then, for every $k \geq 1$, it holds that*

$$\sum_{\alpha:|\alpha|=k} |\widehat{B}(\alpha)| \leq O(\log(nw))^{2wk}.$$

Similarly as done by Forbes and Kelley [8], one can use the Fourier growth bound to improve the seed length for small-width adaptive roBP, and obtain the following corollary.

▶ **Corollary 22** (Restating Theorem 8). *For every $n, w \geq 1$ and $\varepsilon > 0$, there is an explicit ε-PRG $G : \{0,1\}^s \to \{0,1\}^n$ fooling width-w adaptive roBPs with seed length $s = \widetilde{O}(w \log^2(n/\varepsilon))$.*