Pseudorandom Generators for Read-Once Monotone Branching Programs

Dean Doron
Department of Computer Science, Stanford University, CA, USA

Raghu Meka
Department of Computer Science, University of California at Los Angeles, CA, USA

Omer Reingold
Department of Computer Science, Stanford University, CA, USA

Avishay Tal
Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, CA, USA

Salil Vadhan
John A. Paulson School of Engineering & Applied Sciences, Harvard University, Cambridge, MA, USA

Abstract

Motivated by the derandomization of space-bounded computation, there has been a long line of work on constructing pseudorandom generators (PRGs) against various forms of read-once branching programs (ROBPs), with a goal of improving the $O(\log^2 n)$ seed length of Nisan's classic construction [33] to the optimal $O(\log n)$.

In this work, we construct an explicit PRG with seed length $\widetilde{O}(\log n)$ for constant-width ROBPs that are *monotone*, meaning that the states at each time step can be ordered so that edges with the same labels never cross each other. Equivalently, for each fixed input, the transition functions are a monotone function of the state. This result is complementary to a line of work that gave PRGs with seed length $O(\log n)$ for (ordered) *permutation* ROBPs of constant width [7, 26, 12, 37], since the monotonicity constraint can be seen as the "opposite" of the permutation constraint.

Our PRG also works for monotone ROBPs that can read the input bits in any order, which are strictly more powerful than read-once AC^0. Our PRG achieves better parameters (in terms of the dependence on the depth of the circuit) than the best previous pseudorandom generator for read-once AC^0, due to Doron, Hatami, and Hoza [13].

Our pseudorandom generator construction follows Ajtai and Wigderson's approach of iterated pseudorandom restrictions [1, 18]. We give a randomness-efficient width-reduction process which proves that the branching program simplifies to an $O(\log n)$-junta after only $O(\log \log n)$ independent applications of the Forbes–Kelley pseudorandom restrictions [16].

2012 ACM Subject Classification Theory of computation → Pseudorandomness and derandomization; Theory of computation → Circuit complexity

Keywords and phrases Branching programs, pseudorandom generators, constant depth circuits

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2021.58

Category RANDOM

Funding *Dean Doron*: Supported by NSF award CCF-1763311 and Simons Foundation investigators award 689988.
Raghu Meka: Supported by NSF Career award CCF-1553605 and NSF award CCF-2007682.
Omer Reingold: Supported by Supported by NSF award CCF-1763311 and Simons Foundation investigators award 689988.
Salil Vadhan: Supported by NSF grant CCF-1763299 and a Simons Investigator Award.

Acknowledgements We are grateful to Kristoffer Hansen for pointing us to [3] and explaining how their results imply Theorem 3.

1 Introduction

Branching programs are a fundamental model in computational complexity, capturing both space-bounded computation and circuit classes. In this paper, we study a restricted class of branching programs we call *monotone*, giving a new pseudorandom generator for their read-once version.

1.1 Monotone Branching Programs

First we recall the standard definition of a layered branching program:

▶ **Definition 1.** *For $w, n, s \in \mathbb{N}$, a* (layered) branching program *(BP) B on n variables, with length s and width w, or an $[n, s, w]$ BP, is specified by a* start state $v_0 \in [w]$*, a set of* accept states $V_{\text{acc}} \subseteq [w]$*, a sequence of variable indices $i_1, \dots, i_s \in [n]$, and sequence of transition functions $E_j \colon \{0,1\} \times [w] \to [w]$ for $j = 1, \dots, s$.*

A branching program B as above naturally defines a function $B \colon \{0,1\}^n \to \{0,1\}$: Start at the starting state v_0, and then for $j = 1, \dots, s$, read the input bit x_{i_j} and then transition to state $v_j = E_j(x_{i_j}, v_{j-1})$, The branching program accepts *$(B(x) = 1)$ if $v_n \in V_{\text{acc}}$ and* rejects *$(B(x) = 0)$ otherwise.*

B is a read-once *branching program, or an $[n, w]$ ROBP, if $s = n$ and $i_1, \dots, i_s$ is a permutation of $[n]$. If this is the identity permutation (i.e. the variables are read in order), then we say B is an* ordered *branching program.*

A layered branching program B has an associated directed graph. The vertex set has $s+1$ layers of w vertices each. For each $j = 1, \dots, s$, layer j is labelled with an input variable, namely x_{i_j}, and there are two edges, labelled 0 and 1, going from each vertex v in layer j to vertices layer $j+1$, namely $E_j(0, v)$ and $E_j(1, v)$.

We now introduce the model of *monotone* programs that we consider.

▶ **Definition 2** (monotone branching program (MBP))**.** *We say a BP B is* monotone *if for every $j \in [s]$ and $\sigma \in \{0,1\}$, the j-th transition function with input bit restricted to σ, denoted $E_j^\sigma \triangleq E_j(\sigma, \cdot) \colon [w] \to [w]$, is a monotone function according to the standard ordering of $[w]$, i.e. if $v \geq v'$, then $E_j^\sigma(v) \geq E_j^\sigma(v')$.*

That is, put differently, if we draw the layered graph as an $w \times (s+1)$ grid, then whenever we consider the edges associated with a fixed input x, there are no edges crossing. We will refer to BPs that are both monotone and read once as read-once MBPs.

It is important to note that this definition only requires monotonicity with respect to the *state* of the branching program; MBPs can easily compute functions that are non-monotone as a function of their *input* (as we will see below). We remark that the definition of read-once MBPs as defined here is different from the notion of *locally monotone* studied in [10]. Importantly, the latter property is not preserved under restrictions and hence is less nice structurally. The read-once definition also coincides with the notion of *monotone ROBPs* as defined in [31], if we require all reject states to precede the accepting ones in the last layer.[1] However, the formulation above is more convenient for us.

[1] In fact, for the sake of constructing PRGs, we can remove this requirement by replacing ε with ε/w.

1.2 Monotone Branching Programs and AC^0

Recall Barrington's celebrated theorem that constant-width branching programs are equivalent in power to NC^1 circuits [2]. However, when we restrict to *monotone* branching programs, then they become equivalent in power to the much weaker AC^0 circuits. Our model of monotone branching programs is closely related to the models of planar branching programs studied in [3, 4] and the following can be deduced from their results:[2]

▶ **Theorem 3** (corollary of [4]). *A sequence of functions $f_n : \{0,1\}^n \to \{0,1\}$ is in AC^0 if and only if it is computable by a constant-width MBP of polynomial length.*

In this paper, our focus is on read-once MBPs. We prove that these are *strictly stronger* than read-once AC^0:

▶ **Proposition 4.**

1. *If a sequence of functions $f_n\colon \{0,1\}^n \to \{0,1\}$ is in read-once AC^0, then it can be computed by constant-width read-once MBP. Moreover, if f_n can be computed in depth w read-once AC^0, then it can be computed by width $w+1$ read-once MBPs.*
2. *For every $n \geq 3$, there exists a function $f : \{0,1\}^n \to \{0,1\}$ computable by a width 3 read-once MBP, but not computable by any read-once De Morgan formula (regardless of depth).*

Item 1 is proven in the same way as the easier direction of Theorem 3, noting that if we start with a read-once AC^0 circuit, we end up with a read-once MBP. Item 2 is proven by showing that simple functions, like checking whether the input contains at least two ones cannot be computed by a read-once De Morgan formula, but can be computed be width three MBPs. We give the proof for Item 2 in Section 4.

Thus, constant-width read-once MBPs form an intermediate class between read-once AC^0 and AC^0.[3]

1.3 Pseudorandom Generators for Read-Once Branching Programs

A longstanding quest in complexity theory is to understand the power of randomness in relation to space complexity. A central challenge in this direction is to construct pseudorandom generators for read-once branching programs.

In this work we study the question of designing explicit PRGs for small-width ROBPs.

▶ **Definition 5.** *Given a class of functions $\mathcal{F}\colon \{0,1\}^n \to \{0,1\}$, a function $G\colon \{0,1\}^r \to \{0,1\}^n$ is a* PRG *for $\mathcal{F}$ with* error *ε if for any $f \in \mathcal{F}$, we have*

$$\left| \Pr_{y \in_u \{0,1\}^r}[f(G(y)) = 1] - \Pr_{x \in_u \{0,1\}^n}[f(x) = 1] \right| \leq \varepsilon.$$

We call r the seed length *of the generator and the generator is* explicit *if its output can be computed in polynomial time (in n). We often say G ε-fools $\mathcal{F}$.*

[2] In an earlier version of our paper [15], we claimed Theorem 3 as a new contribution. Kristoffer Hansen then explained to us how the result follows from [3]. More precisely, there are simple gadget reductions to show that a constant-width monotone branching program according to our definition can be simulated by a planar branching program according to the definition of [3]. Thus the result of [3] establishing the equivalence of the latter with AC^0 implies the "if" direction of Theorem 3. The "only if" direction is much easier, and amounts to observing that the standard simulation of AC^0 by constant-width branching programs yields a monotone program. For completeness, we provide a direct and self-contained proof of both directions of Theorem 3 in Appendix A.

[3] Both inclusions are strict, since there are functions in AC^0 that cannot be computed by circuits of size smaller than n^4, and the proof of Theorem 3 shows that any constant-width read-once MBP can be simulated by an AC^0 circuit of size $O(n^3)$.

Designing *pseudorandom generators* against ordered ROBPs has received a lot of attention and is intimately connected to the question of understanding the power of randomness vs. space. It has also found a number of applications beyond derandomizing space. ([29, 35, 21, 24, 25, 19] are just few examples.)

The best known PRGs for ordered ROBPs to date are those of Nisan [33] and Impagliazzo–Nisan–Wigderson [23] which give seed length $O(\log^2 n)$ when $w = n^{O(1)}$ and $\varepsilon = 1/n^{O(1)}$. However, even for width four and constant error their construction requiring seed length $O(\log^2 n)$ is still the best. Improving on this seed length, even for constant width, has been a longstanding barrier. We do have better PRGs for various special classes of ROBPs that are independently interesting:

- Braverman, Rao, Raz, and Yehudayoff [7] construct PRGs with $\widetilde{O}(\log n)$ seed length for constant-width ordered *regular* branching programs and $\varepsilon = 1/\operatorname{poly}(\log n)$. Regular branching programs are a special class of ROBPs where we require the structural condition that each vertex has the same in-degree in the underlying layered graph.
- Starting with the work of Koucký, Nimbhorkar, and Pudlák [26], several works [12, 37, 22] have achieved a seed length of $O(\log n)$ (with no $\log\log n$ factors) for the further restricted model of ordered *permutation branching programs*. In these, we require that at each layer j, and for each symbol σ, the transition function E_i^σ is a permutation of $[w]$.
- Meka, Reingold, and Tal [30] construct PRGs with $\widetilde{O}(\log n)$ seed length for width three ordered ROBPs and $\varepsilon = 1/\operatorname{poly}(\log n)$, as well as for unordered ones with $\varepsilon = 1/\operatorname{poly}(\log\log n)$.
- [16] gave a PRG that is significantly different from that of [33, 23] and achieves seed length $O(\log^3 n)$ for polynomial width and $\widetilde{O}(\log^2 n)$ for constant-width ROBPs (again, even unordered).

1.4 Our Main Result

We give an explicit PRG with seed length $\widetilde{O}(\log(n/\varepsilon))$ for read-once MBPs.

▶ **Theorem 6** (see Section 3.2). *For any positive integers n, $w \le n$, and $\varepsilon \in (0, 1/2)$, there is an explicit PRG that ε-fools $[n, w]$ read-once MBPs (even unordered ones) with seed length $O\left(w^2 \log(n/\varepsilon) \cdot (\log\log(n/\varepsilon))^2\right)$.*

We believe that fooling read-once MBPs is an important (and clearly necessary) step toward breaking the $O(\log^2 n)$-barrier for constant-width ROBPs. The class of (ordered) branching programs that we understand best from the perspective of pseudorandomness is that of *permutation branching programs*, thanks to the aforementioned works of [7, 26, 12, 37, 22], all of which obtain their results by showing that the Impagliazzo–Nisan–Wigderson [23] pseudorandom generator can be analyzed better for such programs. Monotone BPs can be seen as the extreme opposite of permutation BPs: the only monotone function $E\colon [w] \to [w]$ that is also a permutation is the identity. Thus the only layers a monotone BP can share with a permutation BP are redundant (can be eliminated from the branching program without changing its functionality). Furthermore, in stark contrast to the case of ordered permutation branching programs, it is known that instantiations of the classical constructions of [33, 23] with $\widetilde{O}(\log n)$ seed length provably do not work against ordered MBPs of width 3 [8].

Technically, our arguments build on the paradigm of using random restrictions for fooling ROBPs as studied in the works of [18, 34, 38, 11, 20, 28, 10, 16, 30, 27, 13, 14]. This gives more evidence that this approach can perhaps lead to $\widetilde{O}(\log n)$ seed length for constant-width ROBPs. Our analysis introduces the idea of exploiting *width reduction* combined with *alphabet reduction* that could be useful for the general problem.

By Proposition 4, the PRG of Theorem 6 is also a PRG for read-once De Morgan formulas. For read-once AC^0, corresponding to width $w = O(1)$, it achieves a better dependence on the width w than the previous generator for read-once AC^0, which has a seed of length $\log(n/\varepsilon) \cdot O(w \log\log(n/\varepsilon))^{2w+2}$ for depth-w formulas [13] (our dependence on w is w^2). For read-once formulas of arbitrary depth, the best PRG prior to our work was the PRG of Forbes and Kelley [16], which has seed length $O(\log(n/\varepsilon)\log^2 n)$. Thus, the PRG of Theorem 6 attains better seed length for read-once formulas that have depth up to $w = o\left(\frac{\log n}{\log\log(n/\varepsilon)}\right)$.

We note that constructing PRGs that are not sensitive to the ordering of the bits in which the input is read is a natural question. First, fooling read-once AC^0, for example, is inherently an unordered task. But also, PRGs for ROBPs that follows the "classical" and successful seed recycling approach due to Nisan (e.g., [33, 23, 26, 7, 6]) heavily depends on the ordering of the bits. In fact, Tzur [39] proved that Nisan's PRG can in fact be distinguished from uniform by an unordered constant width branching program (see also [5]). Thus, the hope is that PRGs that are not sensitive to the ordering will help make progress on the problem of fooling ordered ROBPs with seed length $o(\log^2 n)$.

1.5 Techniques

We proceed by giving an overview of the construction of our PRG and of the techniques we use.

1.5.1 The Iterated Restrictions Approach

We construct our PRG using the *iterated pseudorandom restrictions* approach, pioneered by Ajtai and Wigderson [1] and further developed by Gopalan et al. [18]. That is, we pseudorandomly assign values to a pseudorandomly chosen subset of the variables, and then repeat the process until we assigned values to all variables. Intuitively, designing a pseudorandom restriction for some function f is easier than fooling f outright, because designing a pseudorandom restriction amounts to fooling a "smoothed out" version of f [18], or equivalently, designing a PRG that would fool f after some noise was added [20]. Previous works that used this approach include PRGs for unordered ROBPs [34, 10, 16], PRGs for width-3 ROBPs [18, 38, 30], PRGs for bounded-depth read-once formulas [18, 11, 13, 14], and PRGs for arbitrary-order product tests [20, 28, 27].

Following the iterated restrictions approach, we need our pseudorandom distribution X over restrictions to satisfy two key properties. The first property is that the restriction should approximately *preserve the expectation* of the function. i.e., in expectation over X, the restricted function $f|_X$ should have approximately the same bias as f itself, i.e. $\mathbb{E}_X[\mathbb{E}_U[f|_X(U)]] \approx \mathbb{E}_U[f(U)]$, where U denotes the uniform distribution on the appropriate number of bits. This feature ensures that after sampling the restriction X, our remaining task is simply to fool $f|_X$. The second property is the *simplification* property. That is, we want that the restricted function, for a typical restriction, should be in a sense simpler than f itself. Clearly, simplifying f would make it easier to fool.

To achieve the first property of preserving the expectation, we follow Forbes and Kelley [16], who constructed a simple pseudorandom distribution over restrictions that approximately preserves the expectation of any constant-width ROBP. In the Forbes–Kelley distribution, we determine which coordinates stay alive in an almost k-wise independent manner, and sample the fixed coordinates using a small-bias space. This distribution, per a single restriction, can be sampled using $\widetilde{O}(\log(n/\varepsilon))$ uniform bits. Next, we proceed to discuss how to achieve the simplification property.

1.5.2 Iterative Width Reduction

In our setting, we design our restrictions in a way that fits nicely with the [16] distribution. Thus, the remaining challenge is indeed to ensure that such restrictions *simplify* constant width monotone ROBPs. In [16], the measure of complexity was simply the number of remaining unset variables. That is, Forbes and Kelley argued that after applying $O(\log n)$ independent pseudorandom restrictions, with high probability, all variables are set, and hence there is nothing left to fool. Such an analysis gives seed length of $\widetilde{O}(\log(n/\varepsilon) \cdot \log n)$, and recent works used more sophisticated measures of complexity to show that for more restricted classes of bounded width ROBPs, one can reach a function which is simple enough after only $O(\log \log n)$ independent pseudorandom restrictions [18, 30, 13, 14]. In this work, we continue this line of research, and show that after $O(\log \log n)$ iterations, roughly speaking, the width of the ROBP decreases by 1.

Since the construction and analysis of PRG will not depend on the ordering of the input bits, for simplicity we will describe it here assuming that the monotone ROBP B is ordered (to avoid the indexing i_j of input variables). Before getting to the construction, we highlight the key concept of *colliding layers* in a branching program, which was also paramount in [8, 36, 38, 10, 30]. We say that a BP layer i is a collision one, if there exist two edges with the same label σ that are mapped to the same vertex, i.e. E_i^σ is not a permutation. We say a collision is *realized* if a restriction fixes x_i to σ, and thus effectively introduces a layer with smaller width. The property of monotone BPs we use is that every non-identity layer is a collision one, and crucially, that this property is preserved under restrictions.

Another technical, yet powerful, component of our analysis is treating a branching program with edges labeled 0 and 1, i.e., over the alphabet $\{0,1\}$, as a branching program over a much larger alphabet. Expressing the branching program over a larger alphabet preserves monotonicity and allows us to reduce the width of the BP in some cases. In fact, we will treat both the width and the alphabet size as progress measures.

Towards describing our iterative simplifying process, express our ROBP B, over $\{0,1\}$, as a branching program over $\Sigma = \{0,1\}^\ell$ in the straightforward way, where ℓ will start out as $O(\log(n/\varepsilon))$ and eventually will be reduced to $O(\log\log(n/\varepsilon))$. Each "layer" is now a function from $\Sigma \times [w]$ to $[w]$. Initially, moving to a larger alphabet only makes our task more difficult, but the generality will be useful as we induct on the width below (i.e. even if we start out with a width w program with alphabet $\{0,1\}$, the argument below will force us to handle width $w-1$ programs having alphabet $\{0,1\}^{O(\log(n/\varepsilon))}$.

We iteratively apply the following two observations.

1. **Realizing a collision.** After a suitable pseudorandom restriction X^1, in every sequence of $\exp(O(\ell)) \cdot \log(n/\varepsilon) = \exp(O(\ell))$ collision layers, we will have a collision in one of these layers. As each layer in a read-once MBP is either an identity layer or a collision layer, and this remains true also after transitioning to a larger alphabet, we can deduce that after X^1 every C^ℓ consecutive nontrivial layers contains a layer of width at most $w-1$, for a sufficiently large constant C.
2. **Alphabet reduction.** After a suitable pseudorandom restriction X^2, up to a few "unruly" layers, we can shrink the alphabet size of B so that all layers are effectively over $\{0,1\}^{\ell/2}$. Specifically, we can assume that in every sequence of C^ℓ consecutive layers of alphabet size B, all but $O(\log(n/\varepsilon))$ of them will have their alphabet size reduced to $\{0,1\}^{\ell/2}$.

Both X^1 and X^2 will consist of almost k-wise independent distributions on $\{0,1,\star\}$ where $\star$ represents the bits not assigned by the restriction and we take X^1 to have $\star$-probability $1/2$ and X^2 to have a smaller, yet still constant, $\star$-probability.

Equipped with the above two observations, aiming at reducing the width of B, we apply, independently, the above X^1 and X^2 for $t = O(\log\log(n/\varepsilon))$ iterations. After the first application of X^1, we can write B as $B = B_1 \circ \ldots \circ B_r$, each B_i is of length at most C^ℓ over an ℓ-bit alphabet, starting and ending in a layer of width $w-1$. Then the first application of X^2 will reduce the alphabet of each of the B_i-s to consist of $\ell/2$ bits, except for $O(\log(n/\varepsilon))$ unruly layers within each B_i. The second application of X^1 will now create collisions every $C^{\ell/2}$ non-unruly layers, refining the program further into $B = B'_1 \circ \cdots B'_{r'}$, where each B'_i is of length at most $C^{\ell/2}$ over an $\ell/2$-bit alphabet (except for $O(\log(n/\varepsilon))$ unruly layers), starting and ending with a layer of width $w-1$. The second of application of X^2 will then reduce the alphabet size of each B'_i to at most $\ell/4$ except for $O(\log(n/\varepsilon))$ additional unruly layers within each B'_i. In general, each iteration *reduces the distance* between consecutive layers of width $w-1$ and *reduces the alphabet size*, except for increasing the number of unruly layers by $O(\log(n/\varepsilon))$ within each interval. Finally after $t = O(\log\log(n/\varepsilon))$ iterations, we will have an alphabet where each symbol consists of $\ell^* = O(\log\log(n/\varepsilon))$ bits, so the distance between width $w-1$ layers is at most $C^{\ell^*} = \mathrm{poly}(\log(n/\varepsilon))$. Even including the unruly layers, we can now view our as a width $w-1$ read-once MBP over $\Sigma' = \{0,1\}^{\mathrm{poly}(\log(n/\varepsilon))}$.

Before we can repeat the above process and reduce the width from $w-1$ to $w-2$, etc., we need to reduce Σ' back to $\Sigma = \{0,1\}^{O(\log(n/\varepsilon))}$. We can achieve this by an additional alphabet reduction using an almost k-wise independent distribution with $\star$-probability $1/\operatorname{poly}\log(n/\varepsilon)$ suffices.

Recall that due to [16], we can set the above restrictions to preserve the expectation of our original B, up to a small error. Hence, with seed of length $\widetilde{O}(\log(n/\varepsilon))$ we can both preserve the expectation and reduce the width by 1. Applying this $w-1$ times, with high probability our program will be very simple – a function depending on only $O(\log(n/\varepsilon))$ bits (i.e. a junta), which is fooled by an almost $O(\log(n/\varepsilon))$-wise independent distribution.

All in all, our construction consists of commonly used primitives for PRGs: pseudorandom restrictions in which both the choice of live variables and the the choice of fixed coordinates are sampled from an almost k-wise independent distributions, with varying parameters. The analysis of iterative width reduction via resorting to larger alphabets is new, and we believe can be of use for designing PRGs for other models of computation. Naturally, there are some additional subtleties in the analysis and the choice of parameters, which we leave to the complete analysis in Section 3.

2 Preliminaries

We denote by U_n the uniform distribution over $\{0,1\}^n$. Suppose $\mathcal{C}$ is a class of functions in $\{0,1\}^n \to \mathbb{R}$ and G is a distribution over $\{0,1\}^n$. We say that G ε-fools $\mathcal{C}$ if for every $f \in \mathcal{C}$ it holds that $|\mathbb{E}[f(G)] - \mathbb{E}[f(U_n)]| \le \varepsilon$. Recall that a PRG ε-fooling $\mathcal{C}$ is a function $G\colon \{0,1\}^s \to \{0,1\}^n$ such that $G(U_s)$ ε-fools $\mathcal{C}$. As a shorthand, we often write $\mathbb{E}[f]$ to denote $\mathbb{E}[f(U_n)]$, and omit the subscript n when the number of input bits is clear from context.

2.1 Branching Programs

We extend the definition of branching programs from Definition 1 to large alphabets. We do so by grouping together at most ℓ consecutive bits in a single edge-layer of the program. The main advantage in such a transformation is that we can potentially express a width w program over $\{0,1\}$ as a width $w' < w$ program over $\{0,1\}^\ell$. This will be crucial in our analysis.

We say that a *read-once branching program* (ROBP) B is a $[n, w']_\ell$ ROBP if B can be written as a directed layered graph with $m+1$ layers (for some $m \le n$) denoted $V_1, \ldots, V_{m+1}$. Each V_i consists of at most w' many vertices. Furthermore, there exists a partition of $[n]$ to disjoint subsets $S_1, \ldots, S_m \subseteq [n]$ of size at most ℓ each, and between every consecutive layers of vertices V_i and V_{i+1} there exists a set of directed edges such that any vertex in V_i has $2^{|S_i|}$ edges going towards V_{i+1}. We can treat the the i-th layer of edges as a transition function $E_i \colon \{0,1\}^{S_i} \times [w'] \to [w']$ between V_i and V_{i+1}. Namely, for each $\sigma \in \{0,1\}^{S_i}$ we have the function $E_i^\sigma \triangleq E_i(\sigma, \cdot) \colon [w] \to [w]$ that is defined in the natural way by following the edges labeled σ from V_i to V_{i+1}. Such a program naturally describes a read-once computation on $x \in \{0,1\}^n$, where in the i-th step we follow the edge marked with $x_{S_i} \in \{0,1\}^{S_i}$ from a vertex in V_i to a vertex in V_{i+1}. We often denote ℓ as the *alphabet length* of B and 2^ℓ as the *alphabet size* of B.

We say that an $[n, w']_\ell$ ROBP is monotone if for every $i \in [m]$, its i-th layer E_i satisfies the following. For any $\sigma \in \{0,1\}^{S_i}$ and distinct $x_1, x_2 \in [w']$, $x_1 \ge x_2$ implies $E_i^\sigma(x_1) \ge E_i^\sigma(x_2)$. We say E_i is an *identity layer* if for any $\sigma \in \{0,1\}^{S_i}$ it holds that E_i^σ is the identity function. We say that E_i is a *collision layer* if there exists $\sigma \in \{0,1\}^{S_i}$ such that E_i^σ contains a collision, i.e., there exist distinct $x_1, x_2 \in [w']$ such that $E_i^\sigma(x_1) = E_i^\sigma(x_2)$. We will make use of the following key observation.

▷ Claim 7. In a read-once MBP, every layer is either an identity layer or a collision layer.

As noted above, our techniques will also hold for the unordered setting, so we may assume that the bits of x are permuted by some permutation $\pi \in S_n$, i.e., the i-th layer of the program follow the edge marked by $x_{\pi(i)}$. Since we are in the unordered setting we can assume without loss of generality that there are no identity layers in the program, by skipping these layers.

Observe that if an (unordered) $[n, w]$ ROBP B over $\{0,1\}$ has $m+1$ of its $n+1$ vertex-layers with width at most w' and of distance at most ℓ apart, then we can write B as a $[n, w']_\ell$ ROBP B'. Furthermore, if B is monotone so is B'.

2.2 k-Wise and δ-Biased Distributions

We say that a random variable $Y \sim \{0,1\}^n$ is *δ-biased* if it δ-fools all parity functions. Namely, if for any nonempty $I \subseteq [n]$ it holds that

$$\left| \Pr\left[\bigoplus_{i \in I} Y_i = 1 \right] - \frac{1}{2} \right| \le \delta.$$

There are explicit constructions of δ-biased distributions over $\{0,1\}^n$ that can be sampled efficiently with $O(\log n + \log \frac{1}{\delta})$ truly random bits [32].

▶ **Lemma 8** (Vazirani's XOR Lemma, See e.g., [17, Section 1]). *Let $Y \sim \{0,1\}^n$ be a δ-biased distribution, and let $S \subseteq [n]$. Then, $\left|Y_S - U_{|S|}\right| \le 2^{|S|/2} \cdot \delta$.*

For $p \in [0,1]$, we denote by $\text{Bernoulli}(p)^{\otimes n}$ the distribution over $\{0,1\}^n$ where the bits are i.i.d. and each bit has expectation p. We say that $Z \sim \{0,1\}^n$ is *γ-almost k-wise independent with marginals p* if for every set $I \subseteq [n]$ satisfying $|I| \le k$ it holds that $\left|Z_I - \text{Bernoulli}(p)^{\otimes |I|}\right| \le \gamma$. We can sample such distributions efficiently.

▷ Claim 9 (see, e.g., in [14]). For any positive integers n, k, C, and any $\gamma > 0$, there is an explicit γ-almost k-wise independent distribution with marginals $p = 2^{-C}$ that can be sampled efficiently with $O(Ck + \log \frac{1}{\gamma} + \log\log n)$ truly random bits.

Moreover, we have good tail bounds for almost k-wise distribution.

▶ **Lemma 10** (following [9, 38]). *Let $X_1, \ldots, X_n$ be γ-almost k-wise independent random variables over $\{0,1\}$ with marginals q, and let $\alpha > 0$. Then, for an even $k \leq qn$,*

$$\Pr\left[\left|\sum_{i\in[n]} X_i - qn\right| \geq \alpha qn\right] \leq \left(\frac{16k}{\alpha^2 qn}\right)^{k/2} + 2k\gamma\left(\frac{1}{\alpha q}\right)^k.$$

▶ **Corollary 11.** *Let $X'_1, \ldots, X'_n$ be γ-almost k-wise independent random variables over $\{0,1\}$ with marginals $\geq q$. Then, for an even $k \leq qn$,*

$$\Pr\left[\sum_{i\in[n]} X'_i = 0\right] \leq \left(\frac{16k}{qn}\right)^{k/2} + 2k\gamma\left(\frac{1}{q}\right)^k.$$

Proof. Take $X_i = X'_i \wedge Y_i$ where Y_i is a coin toss with $\Pr[Y_i = 1] = q/\mathbb{E}[X'_i]$. We have that $\mathbb{E}[X_i] = q$, and that $X_1, \ldots, X_n$ are γ-almost k-wise independent with marginals q. Applying Lemma 10 with $\alpha = 1$ implies that

$$\Pr\left[\sum_{i\in[n]} X_i = 0\right] \leq \left(\frac{16k}{qn}\right)^{k/2} + 2k\gamma\left(\frac{1}{q}\right)^k.$$

The proof is complete since $\Pr\left[\sum_{i\in[n]} X'_i = 0\right] \leq \Pr\left[\sum_{i\in[n]} X_i = 0\right]$. ◀

2.3 Restrictions and Pseudorandom Restrictions

A *restriction* is a string $x \in \{0,1,\star\}^n$. Intuitively, $x_i = \star$ means the i-th coordinate has not been set by the restriction. A restriction x can be specified by two strings $y, z \in \{0,1\}^n$ where z determines the $\star$ locations and y determines the assigned values in the non-$\star$ locations. Namely, we define $\mathrm{Res}\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1,\star\}$ by

$$\mathrm{Res}(y,z)_i = \begin{cases} \star & z_i = 1, \\ y_i & z_i = 0. \end{cases}$$

We define a *composition* operation on restrictions, by

$$(x \circ x')_i = \begin{cases} x_i & x_i \neq \star, \\ x'_i & \text{otherwise.} \end{cases}$$

For a function f on $\{0,1\}^n$, the restricted function $f|_x$ on $\{0,1\}^n$ is defined by $f|_x(x') = f(x \circ x')$.

We will repeatedly use the following fact.

▷ Claim 12. Let B be a read-once MBP of length n, and let $x \in \{0,1,\star\}^n$ be any restriction. Then, $B|_x$ is a read-once MBP.

Given a function $f\colon \{0,1\}^n \to \mathbb{R}$ and a distribution $X \sim \{0,1,\star\}^n$, we say that X *preserves the expectation* of f with error ε if $|\mathbb{E}[f|_X(U)] - \mathbb{E}[f]| \leq \varepsilon$.

Forbes and Kelley showed that pseudorandom restrictions preserve the expectation of constant-width ROBPs. We give a "with high probability" version of their result, proved in [14].

▶ **Lemma 13** ([16], restated). *There exists a constant $c \geq 1$ such that the following holds for any positive integers n, w, and $\eta > 0$. Let Z be a γ-almost k-wise independent distribution over $\{0,1\}^n$, where $k \geq c \log \frac{nw}{\eta}$ and $\gamma \leq 2^{-k}$. Let Y be a δ-biased distribution over $\{0,1\}^n$, where $\log \frac{1}{\delta} \geq cwk \log \log n$. Then, for any $[n, w, \{0,1\}]$ BP B it holds that with probability at least $1 - \eta$ over $z \sim Z$,*

$$\left| \mathop{\mathbb{E}}_{Y,U} \left[B|_{\mathrm{Res}(Y,z)}(U) \right] - \mathbb{E}[B] \right| \leq \eta.$$

For $X \sim \{0,1,\star\}^n$ and a positive integer t, we denote by $X^{\circ t}$ the distribution over $\{0,1,\star\}^n$ obtained by drawing independent samples $x^{(1)}, \ldots, x^{(t)} \sim X$ and composing them, namely $x = x^{(1)} \circ \ldots \circ x^{(t)}$. We record two easy claims.

▷ Claim 14. Let $\mathcal{F} \subseteq \{0,1\}^n \to \mathbb{R}$ be some function class which is closed under restrictions. Then, if X preserves the expectation of every $f \in \mathcal{F}$ with error ε, then $X^{\circ t}$ preserves the expectation of every $f \in \mathcal{F}$ with error $t \cdot \varepsilon$.

▷ Claim 15. Let $X = \mathrm{Res}(Y, Z)$ where $Y \sim \{0,1\}^n$ and Z is γ-almost k-wise independent with marginals p. Then, for any positive integer t, the distribution of the $\star$ positions in $X^{\circ t}$ is $(t\gamma)$-almost k-wise independent with marginals p^t.

Finally, we turn to define the notion of realizing a collision, in which a restriction "hits" a symbol in a collision layer that indeed causes a collision.

▶ **Definition 16** (realizing a collision). *Let B be an $[n, w]_\ell$ ROBP and let $E_i : \{0,1\}^{S_i} \times [w] \to [w]$ be a collision layer in B for some $i \in [n]$. We say a string $(y, z) \in \{0,1\}^n \times \{0,1\}^n$* realizes a collision in *E_i if for any symbol $\sigma \in \{0,1\}^{S_i}$ consistent with the restriction $\mathrm{Res}(y, z)$ (i.e., $\sigma_j = y_j$ for all $j \in S_i$ with $z_j = 0$) we have that E_i^σ contains a collision (i.e. $E_i^\sigma(v) = E_i^\sigma(v')$ for two distinct states v, v'). We say (y, z)* realizes a collision in *B if it realizes a collision in some layer E_i.*

We will always use the special case where a collision is realized by $z_{S_i} = 0^{|S_i|}$ and $E_i^{y_{S_i}}$ having a collision.

3 PRGs for Constant-Width Read-Once MBPs

We set forth two auxiliary lemmas that will serve as the building blocks for our iterative argument.

The first claim states that in a read-once MBP with enough colliding layers from $[w]$ to $[w]$, each depending on at most ℓ bits, it is likely that one of the layers will realize a collision under a pseudorandom restriction. The second claim will help us implement *alphabet reduction* as outlined in the introduction.

▶ **Lemma 17** (realizing a collision). *Let $\ell \in \mathbb{N}$ and $m \geq 16^\ell$. For $i = 1, \ldots, m$, let $E_i \colon \{0,1\}^{S_i} \times [w] \to [w]$ where $S_1, \ldots, S_m \subseteq [n]$ are disjoint sets of size at most ℓ. Suppose that each E_i is a collision layer. Let $Y, Z \sim \{0,1\}^n$ be γ-almost k-wise independent distributions, for $\ell \leq k \leq 2^\ell/16$. Then,*

$$\Pr_{Z,Y} \left[\exists i : (Y, Z) \text{ realizes a collision in } E_i \right] \geq 1 - 2^{-k/2} - \gamma \cdot 8^k.$$

Proof. For $j \in [m]$ let $\mathcal{E}_j$ be the event that $z_{S_i} = 0^{|S_i|}$ and $y_{S_i} = \sigma_i$, where σ_i is an arbitrary choice of a string for which $E_i^{\sigma_i}$ collides. Observe that when $\mathcal{E}_j$ occurs, (Y, Z) realizes a collision in E_j. Thus, it suffices to lower bound the probability that some of the $\mathcal{E}_j$ occurs.

The key observation is that the events $\mathcal{E}_1, \ldots, \mathcal{E}_m$ are 2γ-almost k/ℓ-wise independent with marginals $\geq 4^{-\ell}$. Indeed, for any test that depends on k/ℓ of the events $\mathcal{E}_1, \ldots, \mathcal{E}_m$, the test can be written as a function of k bits from Y and k bits from Z, and since any k bits from Y are γ-almost uniform and any k bits of Z are γ-almost uniform, we get that the test is fooled by the distribution with error at most 2γ. Since on the uniform distribution $z_{S_i} = 0^{|S_i|}$ and $y_{S_i} = \sigma_i$ has probability $4^{-|S_i|} \geq 4^{-\ell}$, we get that $\mathcal{E}_1, \ldots, \mathcal{E}_m$ are 2γ-almost k/ℓ-wise independent with marginals $\geq 4^{-\ell}$.

By Corollary 11,

$$\Pr\left[\sum_{j=1}^{m} 1_{\mathcal{E}_j} = 0\right] \leq \left(\frac{16k/\ell}{4^{-\ell}16^{\ell}}\right)^{k/2\ell} + 4(k/\ell)\gamma\left(\frac{1}{4^{-\ell}}\right)^{k/\ell}$$
$$\leq (2^{\ell}/4^{\ell})^{k/2\ell} + \gamma \cdot 2^{\ell} \cdot (4^{\ell})^{k/\ell} \leq 2^{-k/2} + \gamma \cdot 8^{k}.$$

Thus, we get

$$\Pr_{Z,Y}[\exists i : (Y,Z) \text{ realizes a collision in } E_i] \geq \Pr\left[\sum_{j=1}^{m} 1_{\mathcal{E}_j} > 0\right] \geq 1 - 2^{-k/2} + \gamma \cdot 8^{k}. \quad \blacktriangleleft$$

▶ **Lemma 18** (alphabet reduction). *For every constant $C > 1$ there exists a constant $p \in (0,1)$ such that the following holds. Let $\ell \in \mathbb{N}$ and $m \leq C^{\ell}$. For $i = 1, \ldots, m$, let $E_i \colon \{0,1\}^{S_i} \times [w] \to [w]$ where $S_1, \ldots, S_m \subseteq [n]$ are disjoint sets of size at most ℓ. Let $Z \sim \{0,1\}^n$ be a γ-almost k-wise independent distribution with marginals p, for $k \geq \ell$. For $j = 1, \ldots, m$ let B_j be the indicator that Z_{S_j} has more than $\ell/2$ ones. Then,*

$$\Pr\left[\sum_{j=1}^{m} B_j \geq \tfrac{k}{\ell}\right] \leq C^{k} \cdot \gamma + 2^{-k}$$

Proof. Fix a set $T \subseteq [m]$ of size $t = \frac{k}{\ell}$. For $j \in T$, let $B_j(z)$ be the indicator random variable that is 1 if and only if z_{S_j} has more than $\frac{\ell}{2}$ ones. We bound $\Pr_Z[\forall j \in T : B_j(Z) = 1]$. Note that this event depends only on k bits of Z and thus

$$\Pr_Z[\forall j \in T, B_j(Z) = 1] \leq \Pr_U[\forall j \in T : B_j(U) = 1] + \gamma.$$

To bound the probability of $\forall j \in T, B_j(U)$ we note that each B_j happens with probability at most $\binom{\ell}{\ell/2} p^{\ell/2} \leq 2^{\ell} p^{\ell/2}$ and that k/ℓ of these events happen simultaneously with probability at most $(2^{\ell} p^{\ell/2})^{k/\ell} = 2^{k} p^{k/2}$.

Taking the union-bound over all subsets, we get the probability there exists $T \subseteq [m]$ of size t for which $B_j = 1$ for every $j \in T$ is at most

$$\binom{C^{\ell}}{t}\left(\gamma + 2^{k} p^{k/2}\right) \leq C^{\ell t} \cdot \left(\gamma + 2^{k} p^{k/2}\right) = C^{k} \cdot \gamma + 2^{-k},$$

for $p = \frac{1}{16C^2}$. $\blacktriangleleft$

3.1 Width Reduction

▶ **Lemma 19.** *Let B be an $[n,w]_{\ell}$ read-once MBP, and let $\varepsilon > 0$. Let $k = \max(\ell, 4\log(2n/\varepsilon))$ and $\gamma = 32^{-k}$. Set $t = \log(\ell/\log(16k))$. Also, for every $j \in [t]$,*

- *Let $Y_1^j \sim \{0,1\}^n$ and $Z_1^j \sim \{0,1\}^{\ell n}$ be γ-almost k-wise independent distribution;*

- *Let* $Y_2^j \sim \{0,1\}^n$ *be any distribution; and,*
- *Let* $Z_2^j \sim \{0,1\}^n$ *be a* γ*-almost* k*-wise independent distribution with marginal probability* p *as obtained from Lemma 18 for the constant* $C = 16$*.*

For every $j \in [t]$ *we denote the* j*-th restriction as*

$$X^j = X^{j,1} \circ X^{j,2} = \mathrm{Res}(Y_1^j, Z_1^j) \circ \mathrm{Res}(Y_2^j, Z_2^j),$$

and we set the pseudorandom restriction $\overline{X} = X^1 \circ \ldots \circ X^t$*.*

Then, with probability at least $1 - \varepsilon$ *over* $\overline{x} \sim \overline{X}$*,* $B|_{\overline{x}}$ *can be written as an* $[n, w-1]_{\ell'}$ *read-once MBP for* $\ell' = O(k^9)$*.*

Proof. Consider $\ell_0 = \ell, \ell_1 = \ell/2, \ldots, \ell_t = \ell/2^t$. Note that for all i we have $\ell_i \leq k \leq 2^{\ell_i}/16$. Denote $\Sigma^{(0)} = \Sigma$ and $\ell_0 = \ell$. Consider the pseudorandom restriction $X^{1,1}$, denoting $A^1 = B|_{X^{1,1}}$. By Lemma 17, followed by a union bound, we get that with probability at least

$$1 - n \cdot \left(2^{-k/2} + \gamma \cdot 8^k\right) \geq 1 - \varepsilon/2n,$$

every 16^{ℓ_0} consecutive layers of A^1 contains a layer of vertices of width $w-1$. [4] In the following, we condition on the event mentioned in the previous sentence. After the restriction we identify all layers of width $w-1$ and decompose the program to a concatenation of subprograms starting and ending with width at most $w-1$. That is, we can write A^1 as $A_1^1 \circ \ldots \circ A_r^1$, where A_i^1 has initial and final width at most $w-1$, and length at most 16^{ℓ_0} over alphabet $\Sigma^{(0)} = \{0,1\}^{\ell_0}$.

Next, consider the application of $X^{1,2}$ on $A^1 = A_1^1 \circ \ldots \circ A_r^1$. By Lemma 18 and a union bound, with probability at least

$$1 - n(16^k\gamma + 2^{-k}) \geq 1 - \varepsilon/2n,$$

we can reduce the alphabet in each $A_i^1|_{X^{1,2}}$ to $\Sigma^{(1)} = \{0,1\}^{\ell_0/2}$, except for $k/\ell_0 \leq k$ "unruly" wide layers whose alphabet is a subset of $\{0,1\}^{\ell_0}$. To sum up, after the first restriction, with probability at least $1 - \varepsilon/n$, $B^1 = B|_{X^1}$ can be written as a read-once MBP $\tilde{B}_1^1 \circ \ldots \tilde{B}_r^1$, such that for every subprogram $\tilde{B}_i^1$: (i) starts and ends with width $w-1$ (ii) has at most 16^{ℓ_0} good layers with alphabet length $\leq \ell_1$, and (iii) has up to k unruly layers with alphabet length $\leq \ell_0$.

We show by induction on j that, with probability at least $1 - \varepsilon j/n$, after the j-th restriction $B^j = B|_{X^1 \circ \cdots \circ X^j}$ can be written as $\tilde{B}_1^1 \circ \ldots \tilde{B}_{r_j}^1$, such that for every subprogram $\tilde{B}_i^1$:

- Starts and ends with width $w-1$,
- Has at most $16^{\ell_{j-1}}$ good layers with alphabet length $\leq \ell_j$, and,
- Has up to jk unruly layers with alphabet length $\leq \ell_0$.

Assume this to be the case for some $j < t$, we show how to prove it to be the case for $j+1$. We denote by $A^{j+1} = B^j|_{X^{j+1,1}}$. By Lemma 17, with probability at least $1 - n \cdot (2^{-k} + \gamma \cdot 8^k) \geq 1 - \varepsilon/2n$, every 16^{ℓ_j} consecutive good layers of B^{j+1} realizes a collision in A^{j+1}. We write A^{j+1} as

$$A_1^{j+1} \circ \ldots A_{r_{j+1}}^{j+1},$$

[4] Observe that if $16^{\ell_0} > n$ we may not apply Lemma 17. However, then the statement that "every 16^{ℓ_0} consecutive layers of A^1 contains a layer of vertices of width $w-1$" is always true.

where each subprogram A_i^{j+1}: (i) starts and ends with width $w-1$, (ii) has at most 16^{ℓ_j} good layers with alphabet length $\leq \ell_j$, and (iii) has up to jk unruly layers with alphabet length $\leq \ell_0$. To see Item (iii) note that the partition to subprograms is a refinement of the previous partition and thus cannot increase the maximal number of "unruly" layers in a subprogram.

Applying $X^{j+1,2}$, by Lemma 18, with probability at least $1 - n(16^k\gamma + 2^{-k}) \geq 1 - \varepsilon/2n$, in each subprogram A_i^{j+1} we can reduce the alphabet to $\Sigma_{j+1} = \{0,1\}^{\ell_{j+1}}$ except for at most the previous jk unruly layers and potentially k new unruly layers.

Overall, with probability at least $1 - t\varepsilon/n \geq 1 - \varepsilon$, the branching program B^t can be written as $B^t = B_1^t \circ \ldots B_{r_t}^t$, where B_i^t starts and ends with width $w-1$, has at most $16^{\ell_{t-1}}$ good layers and at most kt unruly layers. Thus, each B_i^t is a function of at most

$$16^{\ell_{t-1}} \cdot \ell_t + kt \cdot \ell_0 \leq k \cdot 16^{2(4+\log(k))} + k^3 = O(k^9)$$

bits. We can merge all bits participating in B_i^t to a single symbol in $\Sigma' = \{0,1\}^{\ell'}$ where $\ell' = O(k^9)$. We can thus write B^t as an $[n, w-1]_{\ell'}$ read-once MBP. ◀

As a second step, we reduce the alphabet size from $\text{poly}(\log(n/\varepsilon))$ down to $O(\log(n/\varepsilon))$.

▶ **Lemma 20.** *Let $\varepsilon > 0$, $k = 4\log(n/\varepsilon)$, $\gamma = 1/(16\ell)^k$. Let B be an $[n,w]_\ell$ read-once MBP. Let Z be a γ-almost k-wise independent distribution over $\{0,1\}^n$ with marginals $p_2 = 1/2\ell$; Let Y be any distribution over $\{0,1\}^n$. Let $X = \text{Res}(Y,Z)$.*

Then, with probability at least $1-\varepsilon$ over $\overline{x} \sim \overline{X}$, $B|_{\overline{x}}$ can be written as an $[n,w]_k$ read-once MBP.

Proof. Let $z \sim Z$. As in Lemma 18, for each layer j let B_j be the indicator random variable that is 1 if and only if z_j has more than k ones. By the union bound,

$$\Pr_Z[B_j = 1] \leq \binom{\ell}{k} \cdot (p_2^k + \gamma) \leq \ell^k p_2^k + \ell^k \cdot \gamma \leq 2 \cdot 2^{-k}.$$

Union bounding over all layers, the probability that we failed to reduce the alphabet size to 2^k in any of the layers is at most $1 - 2n2^{-k} \geq 1 - \varepsilon$. ◀

3.2 Putting It Together

Our process will apply a sequence of $w-1$ restrictions sampled using Lemma 13, reducing the program width one at a time, with high probability, while preserving the acceptance probability.

Let c be a large enough constant. Set $k = c\log(nw/\varepsilon)$ and $t = \log(k)$. Set $\gamma = 1/(ck^9)^k$ and $\delta = \min\{\gamma/2^k, 2^{-cwk\log\log n}\}$. Set $C = 16$, $p_1 = \frac{1}{16C^2}$ and $p_2 = \frac{1}{ck^9}$.

For $i \in [w-2]$ and for $j \in [t]$:

- Let $X^{i,j,1} = \text{Res}(Y^{i,j,1}, Z^{i,j,1})$ be a restriction from Lemma 13 with parameters k, γ and δ as above. We have that $Y^{i,j,1}$ is a δ-biased distribution, which is also a γ-almost k-wise independent distribution (due to Lemma 8). We have that $Z^{i,j,1}$ is γ-almost k-wise independent (with marginals $1/2$).
- Let $X^{i,j,2} = \text{Res}(Y^{i,j,2}, Z^{i,j,2})$ be a composition of $\log(1/p_1) = O(1)$ restrictions from Lemma 13 with parameters k, γ and δ as above. We have that $Y^{i,j,1}$ is a δ-biased distribution, which is also a γ-almost k-wise independent distribution. By Claim 15 we have that $Z^{i,j,2}$ is $\log(1/p_1)\gamma$-almost k-wise independent with marginals p_1.

- Let $\tilde{X}^i = \mathrm{Res}(\tilde{Y}^i, \tilde{Z}^i)$ be a composition of $\log(1/p_2) = O(\log\log(nw/\varepsilon))$ restrictions from Lemma 13 with parameters k, γ and δ as above. By Claim 15 we have that $\tilde{Z}^i$ is $\log(1/p_2)\gamma$-almost k-wise independent with marginals p_2.

We define $X^{i,j} = (X^{i,j,1} \circ X^{i,j,2})$ and $X^i = (X^{i,1} \circ X^{i,2} \circ \cdots \circ X^{i,t}) \circ \tilde{X}^i$. And finally, $X = X^1 \circ X^2 \circ \cdots \circ X^{w-1}$. Let $S \sim \{0,1\}^n$ be a ε-almost k-wise independent distribution. Our PRG G is given by

$$G = X \circ S.$$

Let $s = s(n, w, \varepsilon)$ be the seed length required to sample from G. Following the seed lengths of the above primitives in Section 2, we can give the following bound.

▷ Claim 21. It holds that $s = O\left(w^2 \log(n/\varepsilon) \cdot (\log\log(n/\varepsilon))^2\right)$.

▷ Claim 22. G fools width-w read-once MBPs of length n with error at most $4\varepsilon n$.

Proof. Let B be an $[n, w]_1$ read-once MBP, which can also be written as an $[n, w]_k$ read-once MBP by grouping every k-consecutive layers. Note that this transformation preserves monotonicity. Since our restriction is picked as a $m = O(t \cdot w + w \log k) \leq n$ compositions of restrictions that each maintain the acceptance probability of the ROBP up to error ε (Lemma 13), we see that

$$\left| \mathop{\mathbb{E}}_{X,U}[B|_X(U)] - \mathop{\mathbb{E}}_{U}[B(U)] \right| \leq \varepsilon \cdot n.$$

It remains to show that $\mathbb{E}_{X,U}[B|_X(U)] \approx \mathbb{E}_{X,S}[B|_X(S)]$. For that we show that with high probability $B|_X$ can be expressed as a $[n, 1]_k$ read-once MBP. Let $E = E\left(\overline{X}\right)$ be the union of the following bad events:

- There exists an $i \in [w-1]$ such that $(X^{i,1} \circ X^{i,2} \circ \cdots \circ X^{i,t})$ fails to reduce the width, in the sense of Lemma 19.
- There exists an $i \in [w-1]$ such that $\tilde{X}^i$ fails to reduce the alphabet size from $O(k^9)$ to k, in the sense of Lemma 20.

By Lemmas 19 and 20, $\Pr[E] \leq 2w\varepsilon$. Note that in the case that E does not occur, we have that $B|_X$ is a $[n, 1]_k$ ROBP or in other words that it is a junta that depends on at most k bits. In such a case, $B|_X$ will be ε-fooled by S. Overall we have

$$\begin{aligned}\left| \mathop{\mathbb{E}}_{X,U}[B|_X(U)] - \mathop{\mathbb{E}}_{X,S}[B|_X(S)] \right| &\leq \Pr[E] + \Pr[\bar{E}] \cdot \left| \mathop{\mathbb{E}}_{X}\left[\mathop{\mathbb{E}}_{U}[B|_X(U)] - \mathop{\mathbb{E}}_{S}[B|_X(S)] \mid \bar{E} \right] \right| \\ &\leq 2w\varepsilon + \varepsilon.\end{aligned}$$

Combining both estimates we see that

$$\left| \mathop{\mathbb{E}}_{G}[B(G)] - \mathop{\mathbb{E}}_{U}[B(U)] \right| \leq \varepsilon \cdot (n + 2w + 1) \leq 4\varepsilon n. \qquad \triangleleft$$

▶ **Theorem 23.** *Let $n \in \mathbb{N}$, $\varepsilon' > 0$ and $w \leq n$. There exists a generator G that fools width-w read-once MBPs of length n, with error at most ε' and seed-length $O(w^2 \cdot \log(n/\varepsilon') \cdot (\log\log(n/\varepsilon'))^2)$.*

Proof. Apply Claim 22 and Claim 21 with $\varepsilon = \varepsilon'/4n$. ◀

4 Relation to Read-Once AC^0

In this section we study the relation between constant-width read-once MBPs and read-once AC^0:

▶ **Proposition 4.**

1. *If a sequence of functions $f_n\colon \{0,1\}^n \to \{0,1\}$ is in read-once AC^0, then it can be computed by constant-width read-once MBP. Moreover, if f_n can be computed in depth w read-once AC^0, then it can be computed by width $w+1$ read-once MBPs.*
2. *For every $n \geq 3$, there exists a function $f : \{0,1\}^n \to \{0,1\}$ computable by a width 3 read-once MBP, but not computable by any read-once De Morgan formula (regardless of depth).*

First, we establish Item 1 of Proposition 4 by observing that the known implication, that read-once AC^0 formulas can be computed by constant-width ROBPs, yields a monotone ROBP.

▶ **Lemma 24.** *Let $f\colon \{0,1\}^n \to \{0,1\}$ be a function computable by a read-once, depth-w AC^0 formula. Then, f can also be computed by an $[n, w+1]$ read-once MBP.*

Proof. We prove the claim by induction on the depth w, and further prove that the 'accept' states in our read-once MBP are above the 'reject' states (that is, if s is an accept state and s' is a reject state than $s \geq s'$). For $w = 1$, f computes either the disjunction or the conjunction of at most s literals. This can clearly be done by an $[n, s, 2]$ read-once MBP, if we set state 2 to be an accept state (and so state 1 is a reject one).

Next, fix some f computable by a formula $F\colon \{0,1\}^n \to \{0,1\}$ of depth $w > 1$ and size s, and assume that its top gate is an AND gate (the other case is similar). We denote the subformulas feeding into the top gate as $F_1, \ldots, F_m$, and these are on disjoint variables because the formula is read-once. By the induction's hypothesis, each subformula F_i is computable by a width w read-once MBP over its variables with the accept states being on top.

To construct B that computes F, we can concatenate the B_i-s and add another "sudden reject" level at level $s = 1$.[5] The starting vertex of B is the starting vertex of B_1. Whenever a computation of some B_i, for some $i < m$, reaches its final layer, we rewire the edges in that layer to either the sudden reject level, if B_i did not reach an accepting vertex, or to the starting vertex of B_{i+1}. The accept vertices of B are the accept vertices of B_m. Note that this transformation preserves the ordering between accepts and reject states, since B_m does.

The fact that B computes f readily follows, and B is read-once because the B_i-s are on disjoint variables. To argue that monotonicity is preserved, simply observe that the rewiring preserves the order: In the AND case, accept vertices are rewired to the next starting vertex, which is indeed above the sudden reject level, to which all reject vertices are rewired. The OR case is similar. ◀

We now prove Item 2 of Proposition 4, giving a family of functions computable by read-once MBPs but not by read-once formulas. Our proof also gives a new characterization of read-once formulas.

[5] In a sudden reject level, each vertex transitions to the same level with both its edges, and the last vertex in that level is a reject vertex. When the top gate is an OR gate, we would replace the sudden reject level with a sudden accept level at at $s = w + 1$, and make the last vertex of the sudden accept level an accepting vertex.

▶ **Lemma 25.** *For every $n \geq 3$, there exists a function $f : \{0,1\}^n \to \{0,1\}$ computable by a width 3 read-once MBP, but not computable by any read-once De Morgan formula (regardless of depth).*

Proof. We first give a property of functions computable by read-once formulas. Given $g\colon \{0,1\}^m \to \{0,1\}$ and $b \in \{0,1\}$, let $W_b(g) \in \{0,\ldots,m\}$ denote the size of the smallest set of coordinates $I \subseteq [m]$ for which there exists a $z \in \{0,1\}^{|I|}$ such that for every $x \in \{0,1\}^m$ it holds that $x_I = z$ implies $g(x) = b$.

▶ **Lemma 26.** *Let $g\colon \{0,1\}^n \to \{0,1\}$ be a function computable by a read-once De Morgan formula. Then, $W_0(g) \cdot W_1(g) \leq n$.*

Roughly speaking, this lemma says that for a function computable by a read-once formula, we can either find a short witness for it being 0, or a short witness for it being 1. In particular, it cannot be highly resilient.

Proof. We prove the lemma by induction on the formula's depth d. For $d = 1$, g is either an AND of literals or an OR of literals. For the AND function, $W_0(\text{AND}) = 1$ and $W_1(\text{AND}) = n$. For the OR function, $W_0(\text{OR}) = n$ and $W_1(\text{OR}) = 1$. Thus, indeed, $W_0(g) \cdot W_1(g) \leq n$.

Assume our lemma holds for formulas of depth $d \geq 1$, and let g be some formula of depth $d+1$, say with an AND top gate, so $g = \text{AND}(f_1,\ldots,f_k)$, each $f_i\colon \{0,1\}^{n_i} \to \{0,1\}$ is a depth-d formula. In this case, $W_0(g) = \min_{j\in[k]} W_0(f_j)$ and $W_1(g) = \sum_{i\in[k]} W_1(f_i)$. By our induction's hypothesis, we get that

$$W_0(g) \cdot W_1(g) = \left(\min_{j\in[k]} W_0(f_j)\right) \cdot \sum_{i\in[k]} W_1(f_i) \leq \sum_{i\in[k]} W_0(f_i) \cdot W_1(f_i) \leq \sum_{i\in[k]} n_i = n.$$

The case of an OR top gate is analogous. ◀

Now, our function $f\colon \{0,1\}^n \to \{0,1\}$ will simply be the Thr_2^n function, that returns 1 if and only if the Hamming weight of the input string $x \in \{0,1\}^n$ is at least 2. There, $W_1(f) = 2$ and $W_0(f) = n-1$, so it is not computable by read-once formulas, however f is computable by a simple width-3 read-once MBP. ◀

We note that we can also construct balanced functions f separating read-once MBPs from read-once De Morgan formulas. In particular, $f = \text{AND}_m \circ \text{Thr}_2^w$ for $m = O(2^w/w)$ (which resembles the Tribes function) has this property. More generally, one can consider, say, Thr_2, as a "gadget" to construct richer families of read-once MBPs not computable by read-once formulas.

References

1 Miklos Ajtai and Avi Wigderson. Deterministic simulation of probabilistic constant depth circuits. *Advances in Computing Research*, 5(199-222):1, 1989.

2 David A. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. *Journal of Computer and System Sciences*, 38(1):150–164, 1989.

3 David A. Barrington, Chi-Jen Lu, Peter Bro Miltersen, and Sven Skyum. Searching constant width mazes captures the AC^0 hierarchy. In *Proceedings of the 15th Annual Symposium on Theoretical Aspects of Computer Science (STACS 1998)*, pages 73–83. Springer, 1998.

4 David A. Barrington, Chi-Jen Lu, Peter Bro Miltersen, and Sven Skyum. On monotone planar circuits. In *Proceedings of the 14th Annual IEEE Conference on Computational Complexity (CCC 1999)*, pages 24–31. IEEE, 1999.

5 Andrej Bogdanov, Periklis A Papakonstaninou, and Andrew Wan. Pseudorandomness for read-once formulas. In *Proceedings of the 52nd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2011)*, pages 240–246. IEEE, 2011.

6 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–STOC18–299, 2020. doi:10.1137/18M1197734.

7 Mark Braverman, Anup Rao, Ran Raz, and Amir Yehudayoff. Pseudorandom generators for regular branching programs. *SIAM Journal on Computing*, 43(3):973–986, 2014.

8 J. Brody and E. Verbin. The coin problem and pseudorandomness for branching programs. In *Proceedings of the 51st IEEE Annual Symposium on Foundations of Computer Science (FOCS 2012)*, pages 30–39, October 2010. doi:10.1109/FOCS.2010.10.

9 L. Elisa Celis, Omer Reingold, Gil Segev, and Udi Wieder. Balls and bins: Smaller hash families and faster evaluation. *SIAM Journal on Computing*, 42(3):1030–1050, 2013.

10 Eshan Chattopadhyay, Pooya Hatami, Omer Reingold, and Avishay Tal. Improved pseudorandomness for unordered branching programs through local monotonicity. In *Proceedings of the 50th Annual ACM Symposium on Theory of Computing (STOC 2018)*, pages 363–375. ACM, 2018.

11 Sitan Chen, Thomas Steinke, and Salil Vadhan. Pseudorandomness for read-once, constant-depth circuits. *arXiv preprint arXiv:1504.04675*, 2015.

12 Anindya De. Pseudorandomness for permutation and regular branching programs. In *Proceedings of the 26th Annual IEEE 26th Annual Conference on Computational Complexity (CCC 2011)*, pages 221–231. IEEE, 2011.

13 Dean Doron, Pooya Hatami, and William M. Hoza. Near-optimal pseudorandom generators for constant-depth read-once formulas. In *Proceedings of the 34th Computational Complexity Conference (CCC 2019)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2019.

14 Dean Doron, Pooya Hatami, and William M. Hoza. Log-seed pseudorandom generators via iterated restrictions. In *Proceedings of the 35th Computational Complexity Conference (CCC 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

15 Dean Doron, Raghu Meka, Omer Reingold, Avishay Tal, and Salil Vadhan. Monotone branching programs: Pseudorandomness and circuit complexity. In *Electronic Colloquium on Computational Complexity (ECCC)*, number TR21-018, February 2021. Version 1.

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

17 Oded Goldreich. Three XOR-lemmas – an exposition. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, pages 248–272. Springer, 2011.

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

19 Iftach Haitner, Danny Harnik, and Omer Reingold. On the power of the randomized iterate. *SIAM Journal on Computing*, 40(6):1486–1528, 2011.

20 Elad Haramaty, Chin Ho Lee, and Emanuele Viola. Bounded independence plus noise fools products. *SIAM Journal on Computing*, 47(2):493–523, 2018. doi:10.1137/17M1129088.

21 Alexander Healy, Salil Vadhan, and Emanuele Viola. Using nondeterminism to amplify hardness. *SIAM Journal on Computing*, 35(4):903–931, 2006.

22 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. doi:10.4230/LIPIcs.ITCS.2021.7.

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

24 Piotr Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *Journal of the ACM*, 53(3):307–323, 2006.

25 Adam R. Klivans, Homin Lee, and Andrew Wan. Mansour's conjecture is true for random DNF formulas. In *Proceedings of the 23rd Annual Conference on Learning Theory (COLT 2010)*, 2010.

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

27 Chin Ho Lee. Fourier bounds and pseudorandom generators for product tests. In *Proceedings of the 34th Computational Complexity Conference (CCC 2019)*, pages 7:1–7:25. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019.

28 Chin Ho Lee and Emanuele Viola. More on bounded independence plus noise: Pseudorandom generators for read-once polynomials. In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 24, page 167, 2017.

29 Michael Luby, Boban Velickovic, and Avi Wigderson. Deterministic approximate counting of depth-2 circuits. In *Proceedings of the 2nd Annual Israel Symposium on Theory and Computing Systems (ISTCS 1993)*, pages 18–24. IEEE, 1993.

30 Raghu Meka, Omer Reingold, and Avishay Tal. Pseudorandom generators for width-3 branching programs. In *Proceedings of the 51st Annual ACM Symposium on Theory of Computing (STOC 2019)*, pages 626–637. ACM, New York, 2019.

31 Raghu Meka and David Zuckerman. Pseudorandom generators for polynomial threshold functions. *SIAM Journal on Computing*, 42(3):1275–1301, 2013.

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

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

34 Omer Reingold, Thomas Steinke, and Salil Vadhan. Pseudorandomness for regular branching programs via Fourier analysis. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 655–670. Springer, 2013.

35 D. Sivakumar. Algorithmic derandomization via complexity theory. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC 2002)*, pages 619–626. ACM, 2002.

36 John Steinberger. The distinguishability of product distributions by read-once branching programs. In *Proceedings of the 28th IEEE Conference on Computational Complexity (CCC 2013)*, pages 248–254. IEEE, 2013.

37 Thomas Steinke. Pseudorandomness for permutation branching programs without the group theory. Technical Report TR12-083, Electronic Colloquium on Computational Complexity (ECCC), July 2012. URL: http://eccc.hpi-web.de/report/2012/083/.

38 Thomas Steinke, Salil Vadhan, and Andrew Wan. Pseudorandomness and Fourier-growth bounds for width-3 branching programs. *Theory of Computing*, 13(1):1–50, 2017.

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

A Monotone Branching Programs and AC^0 Circuits

In this section we give a self-contained proof of the equivalence between constant width MBPs and AC^0 circuits, proving Theorem 3:

▶ **Theorem 3** (corollary of [4]). *A sequence of functions $f_n : \{0,1\}^n \to \{0,1\}$ is in AC^0 if and only if it is computable by a constant-width MBP of polynomial length.*

First we note that the "if" direction follows from Lemma 24.

▶ **Lemma 27.** *Let $f\colon \{0,1\}^n \to \{0,1\}$ be a function computable by a (read-many) AC^0 formula of depth w and size s. Then, f can also be computed by an $[n, s, w+1]$ MBP.*

(Here take the *size* of an AC^0 formula to be the number of leaves.)

Proof. Let $F(x_1, \ldots, x_n)$ be depth-w AC^0 formula of size s. Then by putting distinct variables on the leaves of F we obtain a read-once AC^0 formula $G(y_1, \ldots, y_s)$ on s variables such that $F(x_1, \ldots, x_n) = G(\sigma_1 x_{i_1}, \ldots, \sigma_s x_{i_s})$ where $i_1, \ldots, i_s \in [n]$ and $\sigma_1, \ldots, \sigma_s \in \{\pm 1\}$ (referring to whether or not the variable is negated at each leaf). By Lemma 24, there is a read-once MBP $B(y_1, \ldots, y_s)$ of width $w+1$ computing G. Then, $B(\sigma_1 x_{i_1}, \ldots, \sigma_s x_{i_s})$ is a (read-many) MBP of width $w+1$ and length s computing F. ◀

The result naturally extends to AC^0 circuits, due to the standard transformation expressing a size s depth w AC^0 circuit as a size s^w depth w AC^0 formula.

▶ **Corollary 28.** *Let $f\colon \{0,1\}^n \to \{0,1\}$ be a function computable by a depth-w AC^0 circuit of size s. Then, f can also be computed by an $[n, s^w, w+1]$ MBP.*

Next, we give the other direction of Theorem 3. Similarly to the other direction, we start by showing that *read-once* MBPs can the simulated by (read-many) AC^0:

▶ **Lemma 29.** *Let $f\colon \{0,1\}^n \to \{0,1\}$ be a function computable by a read-once MBP of width w. Then, f can also be computed by a circuit of depth $O(w)$ and size $O(w^4 n^3)$.*

Proof. We prove this lemma by induction on the width. For $w = 1$ the claim is trivial. Fix some $w > 1$ and let B be an $[n, w]$ read-once MBP. We define two BPs, B^u and B^ℓ, each of width $w-1$, as follows.

- For B^u, we remove the first level of vertices (that is, removing state number 1 in each layer) and reroute edges that go into state 1 to state 2. Formally, each transition $U_i^b\colon \{2, \ldots, w\} \to \{2, \ldots, w\}$ of B^u is defined by $U_i^b(x) = \max\left\{E_i^b(x), 2\right\}$, for $E_i^b\colon [w] \to [w]$ being the corresponding transition of B.
- Similarly, for B^ℓ, we remove the last level of vertices: Each transition $L_i^b\colon [w-1] \to [w-1]$ of B^ℓ is defined by $L_i^b(x) = \min\left\{E_i^b(x), w-1\right\}$.

Notice that these transformations preserve monotonicity. Roughly speaking, our goal is to first argue that at each transition, B acts the same as either B^u or B^ℓ, depending on whether B last reached the state 1 or the state w. Then, we show that we can efficiently detect, given any layer j and an input x, if indeed $B(x)$ passed through the state 1 or through the state w before reaching the layer j.

Let s_0 be the starting vertex of B, and denote $u_0 = \max\{s_0, 2\}$ and $\ell_0 = \min\{s_0, w-1\}$. Given some input $x \in \{0,1\}^n$, we consider the computation path of all three BPs on x. Towards this end, denote by $s_1, \ldots, s_n \in [w]$ the states that x traverses in B, $u_1, \ldots, u_n \in \{2, \ldots, w\}$ the states that x traverses in B^u and $\ell_1, \ldots, \ell_n \in [w-1]$ the states that x traverses in B^ℓ. First, observe that:

▷ Claim 30. For every $i \in [n]$, $u_i \geq s_i \geq \ell_i$.

The above claim readily follows by induction on i, using the monotonicity property. Next, we argue:

▷ Claim 31. For every $i \in [n]$, let $j \leq i$ be the largest integer such that $s_j \in \{1, w\}$, if it exists. Thus, if $s_j = w$ then $u_i = s_i$ and if $s_j = 1$ then $\ell_i = s_i$.

Proof. Fix some $i \in [n]$ and assume that $j \leq i$ is the largest integer such that $s_j \in \{1, w\}$, say $s_j = 1$. By Claim 30, we must also have $\ell_j = 1$. Then by induction, we also have $\ell_{j'} = s_{j'}$ for all $j' = j, j+1, \ldots, i$, because the only way in which the transition in B^ℓ and B can differ is if $s_{j'} = w$, which by assumption does not occur in this interval. ◁

Hence, for each layer i, we know that either $s_i = u_i$ or $s_i = \ell_i$, and we know which is the case by looking at the last place the original path reached either 1 or w.

By our induction's hypothesis, for every $i \in [n]$ and $s \in \{2, \ldots, w\}$ there exists a circuit $C^u_{i,s} \colon \{0,1\}^n \to \{0,1\}$ such that $C^u_{i,s}(x) = 1$ if and only if B^u reached the state s after reading $x_1, \ldots, x_i$. Similarly, there exists a circuit $C^\ell_{i,s}$ that detects whether B^ℓ reached $s \in [w-1]$ in the i-th layer upon traversing with x. Using these circuits, for each $s \in [w]$, we will construct a circuit $C_s(x)$ that determined whether $s_n = s$.

The construction goes as follows. The circuit will determine the last j where there was a "switch" between the two cases of Claim 31, i.e., the smallest $j \in [n]$ such that $s_j \in \{1, w\}$ and for every $k \geq j$ it holds that $s_k \in \{2, \ldots, w-1\} \cup \{s_j\}$. Observe that if $s_j = 1$ then $s_{j-1} = u_{j-1}$, so $E^{x_j}_j(u_{j-1}) = 1$. Afterward, we keep following B^ℓ, i.e., $s_k = \ell_k$ and so $E^{x_{k+1}}_{k+1}(\ell_k) \neq w$ for all $k \geq j$. The converse also holds. Namely, $E^{x_j}_j(u_{j-1}) = 1$ implies that $s_j = \ell_j = 1$ (since $\ell_{j-1} \leq u_{j-1}$ and the program is monotone) and $E^{x_{k+1}}_{k+1}(\ell_k) \neq w$ for all $k \geq j$ implies that indeed $s_{k+1} = \ell_{k+1}$. Thus, the predicate

$$P_L(x) = \left(\bigvee_{j \in [n]} \left(\left(E^{x_j}_j(u_{j-1}) = 1 \right) \wedge \bigwedge_{k \geq j} \left(E^{x_{k+1}}_{k+1}(\ell_k) \neq w \right) \right) \right) \vee \left(u_1 \neq w \wedge \bigwedge_{k \geq 1} E^{x_k}_k(\ell_k) \neq w \right)$$

evaluates to 1 if and only if the largest integer $j \leq n$ such that $s_j \in \{1, w\}$ has $s_j = 1$, or s_j never equals w (and hence $s_n = \ell_n$). Following the same reasoning,

$$P_U(x) = \left(\bigvee_{j \in [n]} \left(\left(E^{x_j}_j(\ell_{j-1}) = w \right) \wedge \bigwedge_{k \geq j} \left(E^{x_{k+1}}_{k+1}(u_k) \neq 1 \right) \right) \right) \vee \left(\ell_1 \neq 1 \bigwedge_{k \geq 1} E^{x_k}_k(u_k) \neq 1 \right)$$

evaluates to 1 if and only if the largest integer $j \leq n$ such that $s_j \in \{1, w\}$ has $s_j = w$, or s_j never equals 1 (and hence $s_n = u_n$).

We now wish to compute $P_L \colon \{0,1\}^n \to \{0,1\}$ by a shallow circuit. Determining u_{j-1} can be done by querying $C^u_{j-1,s}(x)$ for each $s \in \{2, \ldots, w\}$. Similarly, determining ℓ_k can be done by querying $C^\ell_{k,s}(x)$ for each $s \in [w-1]$. The functions E^b_j and E^b_{k+1}, for each $b \in \{0,1\}$, are determined solely by B and can be hardwired. Letting $\mathsf{size}(w-1)$ and $\mathsf{depth}(w-1)$ be the size and depth upper bound for the circuits guaranteed to us by the hypothesis, we can bound $\mathsf{size}(P_L)$ by $2nw \cdot \mathsf{size}(w-1) + O(wn^2)$ and $\mathsf{depth}(P_L)$ by $\mathsf{depth}(w-1) + O(1)$. The same bounds for $P_U \colon \{0,1\}^n \to \{0,1\}$ also hold.

Equipped with circuits C_L and C_U computing P_L and P_U respectively, we are ready to compute B. Indeed, all that is left is to determine whether $s_n = \ell_n$ or $s_n = u_n$ and invoke the relevant circuit from the previous level. This incurs additional constant depth and $O(wn)$ size. Overall, the size and depth of C satisfies the recurrence relations $\mathsf{size}(w) = O(nw) \cdot \mathsf{size}(w-1) + O(wn^2)$ and $\mathsf{depth}(w) = \mathsf{depth}(w-1) + O(1)$. As $\mathsf{size}(1) = O(n)$ and $\mathsf{depth}(1) = O(1)$, this gives us depth $O(w)$ and size $w^{O(w)} \cdot n^w$.

We can improve the size of the circuit by a dynamic programming approach. For $1 \leq a \leq b \leq w$, let $B^{[a,b]}$ be the ROBP in which we keep only the levels $a, \ldots, b$ and rewire edges accordingly. Namely, we replace each $E^\sigma_i(x)$ with $\max\{a, \min\{b, E^\sigma_i(x)\}\}$. Observe that for when $a < b$, $(B^{[a,b]})^\ell = B^{[a,b-1]}$ and $(B^{[a,b]})^u = B^{[a+1,b]}$.

For every $1 \leq a \leq b \leq w$ and $s \in \{a, \ldots, b\}$, let $X_i^{a,b,s}$ be the indicator which is 1 if and only if upon reading the first i bits of x, the program $B^{[a,b]}$ reached the state s. Note that there are at most $w^3 \cdot n$ such indicators overall.

Fix some integer $\Delta \in \{0, \ldots, w-1\}$. We can compute the values

$$\mathcal{I}_\Delta = \left\{ X_i^{a,a+\Delta,s} : a \in [w-\Delta], s \in [a, a+\Delta], i \in [n] \right\}$$

in the following manner. For $\Delta = 0$, all indicators are true. For $\Delta \geq 1$, assume we already computed the values

$$\mathcal{I}_{\Delta-1} = \left\{ X_i^{a,a+\Delta-1,s} : a \in [w-(\Delta-1)], s \in [a, a+\Delta-1], i \in [n] \right\}.$$

Thus, to compute a single indicator from $\mathcal{I}_\Delta$ given $\mathcal{I}_{\Delta-1}$, we can use the above recurrence relations, as each ℓ_i and u_i correspond to some indicator from $\mathcal{I}_{\Delta-1}$. This takes $O(wn^2)$ size and $O(1)$ depth. Computing the entire $\mathcal{I}_\Delta$ thus takes $O(w^3n^3)$ size and $O(1)$ depth. Overall, computing ◀

Similarly to the proof of Lemma 27, we can handle the read-many case by noting that a read-many MBP of length s can be obtained from a read-once MBP on s variables by a variable substitution. This gives us the "only if" direction of Theorem 3:

▶ **Corollary 32.** *Let $f\colon \{0,1\}^n \to \{0,1\}$ be a function computable by an $[n, s, w]$ MBP for $s \geq n$. Then, f can also be computed by a circuit of depth $O(w)$ and size $O(w^4s^3)$.*