Identity Testing and Lower Bounds for Read-k Oblivious Algebraic Branching Programs[*]

Matthew Anderson[†] Michael A. Forbes[‡] Ramprasad Saptharishi[§]
Amir Shpilka[‡] Ben Lee Volk[‡]

Abstract

Read-k oblivious algebraic branching programs are a natural generalization of the well-studied model of read-once oblivious algebraic branching program (ROABPs). In this work, we give an exponential lower bound of $\exp(n/k^{O(k)})$ on the width of any read-k oblivious ABP computing some explicit multilinear polynomial f that is computed by a polynomial-size depth-3 circuit. We also study the polynomial identity testing (PIT) problem for this model and obtain a white-box subexponential-time PIT algorithm. The algorithm runs in time $2^{\tilde{O}(n^{1-1/2^{k-1}})}$ and needs white box access only to know the order in which the variables appear in the ABP.

[*]A preliminary version of this paper appeared in [AFS+16].

[†]Department of Computer Science, Union College, Schenectady, New York, USA, E-mail: `andersm2@union.edu`

[‡]Department of Computer Science, Princeton University, USA, E-mail: `miforbes@csail.mit.edu`. Supported by the Princeton Center for Theoretical Computer Science.

[§]Department of Computer Science, Tel Aviv University, Tel Aviv, Israel, E-mails: `ramprasad@cmi.ac.in`, `shpilka@post.tau.ac.il`, `benleevolk@gmail.com`. The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement number 257575.

1 Introduction

Algebraic complexity studies the complexity of syntactically computing polynomials using arithmetic operations. The most natural model for computing polynomials is an *algebraic circuit*, which is a directed acyclic graph whose leaves are labeled by either variables from $\{x_1, \ldots, x_n\}$ or elements from a field $\mathbb{F}$, and whose internal nodes are labeled by the arithmetic operations $+$ and $\times$. Each node computes a polynomial in the natural way. The associated complexity measures are the *size* (the number of wires) and the *depth* (the length of a longest path from an input node to the output node) of the circuit. A circuit whose underlying graph is a tree is called a *formula*.

Another model of computation, whose power lies between that of circuits and formulas, is that of an *algebraic branching program* (ABP). An ABP is a layered directed acyclic graph with a source node and a sink node, whose edges are labeled by polynomials. An ABP computes a polynomial in the following way. Every directed source-sink path computes the polynomial that is obtained from taking the product of all edge labels along the path. The polynomial computed by the ABP is the sum over all paths of those polynomials.[1] In addition to the size and depth of an ABP, another relevant complexity measure is the *width* of the program, which is the maximal number of vertices in a layer. See Section 1.1 for the formal definitions of the models that are considered in this work.

Two of the most important problems in algebraic complexity are (i) proving exponential lower bounds for arithmetic circuits (i.e., proving that any circuit computing some explicit polynomial f must be of exponential size), and (ii) giving an efficient *deterministic* algorithm for the *polynomial identity testing* (PIT) problem. The latter is the problem of deciding, given an arithmetic circuit, formula or ABP computing a polynomial f, whether f is the identically zero polynomial. PIT has a simple randomized algorithm that follows from the Schwartz-Zippel-DeMillo-Lipton lemma [Sch80, Zip79, DL78] that says that over a large enough field, a non-zero polynomial evaluates to a non-zero value on *most* points. Hence, in order to decide whether f is zero it suffices to evaluate the circuit/formula/ABP on a random point (which can be done efficiently).

We further note that the randomized algorithm described above only needs the ability to evaluate f at a given point. Such algorithms are called *black-box* PIT algorithms. It is readily seen that black-box algorithms are equivalent to producing a small *hitting set*, that is, a set $\mathcal{H}$ of points such that for every non-zero f there is a point in $\mathcal{H}$ that f evaluates to non-zero on. Algorithms that are given the computation graph of the circuit / formula / ABP as input are called *white-box* algorithms. Naturally, white-box access is much less restrictive and one expects it is easier to obtain better algorithms in this case.

Apart from being a very natural problem about arithmetic computation, PIT is one of the most general problems for which an efficient randomized algorithm is known, but an efficient deterministic one is not. Indeed, many other randomized algorithms, e.g. parallel algorithms for finding matching in graphs [KUW86, MVV87] or algorithms for polynomial factorization [SV10, KSS15], reduce to PIT in the sense that derandomization of PIT implies derandomization of these algorithms as well.

For more background on arithmetic circuits we refer the reader to the survey [SY10].

At first glance, the two problems described above seem rather different, as one is concerned with proving lower bounds and the other with providing efficient algorithms. However, a series of works uncovered an intricate web of connections between the two, both in the white-box

[1] This is analogous to boolean branching programs. There each path computes the AND of edge labels and the output is the OR of all path-functions.

[KI04, DSY09] and in the black-box [HS80, Agr05] models. That is, derandomizing PIT implies lower bounds for circuits, which gives a convincing explanation for why this derandomization problem is hard. Conversely, an explicit hard polynomial gives a recipe to "fool" small arithmetic circuits with respect to non-zeroness, in a very similar manner to the hardness-versus-randomness paradigm in boolean complexity (c.f., e.g., [NW94]).

In light of the hardness of proving lower bounds for general algebraic circuits, research has focused on trying to understand the effect that structural restrictions, like constant depth and multilinearity, have on the expressive power of the model.

One research direction that has attracted a lot of attention considers small-depth arithmetic circuits. Following Valiant et al. [VSBR83], Agrawal and Vinay gave a reduction from general circuits (of polynomial degree) to depth-4 circuits, that maps subexponential size to subexponential size [AV08]. This reduction was later improved and extended in [Koi12, Tav15, GKKS13]. In a breakthrough work Gupta et al. [GKKS14] proved exponential lower bounds for computing the $n \times n$ determinant by depth-4 homogeneous formulas with bottom fan-in $O(\sqrt{n})$, which is the kind of circuits one gets from the above depth reduction. In the work that followed, tighter lower bounds for homogeneous depth-4 circuits were proved both for "hard" polynomials such as the permanent but also for easier polynomials such as the determinant and the iterated matrix multiplication polynomial [KSS14, FLMS14, KLSS14, KS14b, KS14a].

In parallel, a lot of research effort was also focused on PIT for small-depth circuits with various restrictions such as bounded top fan-in or multilinearity [DS07, KS07, KS09, SS12, KMSV13, SV11, OSV15]. Similar to the situation with lower bounds, derandomization of PIT for depth-4 circuits (or, depth-3 in certain cases) implies derandomization of the general case [AV08, GKKS13]. As depth-3 multilinear formulas that have small top fan-in are a special case of sum of read-once arithmetic formulas (here, a read-once formula is an arithmetic formula in which each variable labels at most one node), Shpilka and Volkovich gave polynomial identity tests for this model [SV15]. Later, Anderson, van Melkebeek and Volkovich gave a PIT for multilinear read-k formulas, which extend both models [AvMV15].

Another line of work focused on read-once oblivious ABPs (ROABPs, and we again refer to Section 1.1 for the exact definition). ROABPs were defined by Nisan [Nis91] in the context of proving lower bounds for non-commutative formulas. While this model seems a bit restrictive, it was shown that derandomizing PIT for ROABPs implies derandomization of Noether's normalization lemma for certain important varieties [Mul12, FS13a]. It is also not hard to show that ROABPs are strictly stronger than read-once arithmetic formulas. Another motivation to study this model is that it is the algebraic analog of a read-once boolean branching program, which arises in the context of pseudorandomness for small-space computation [Nis92]. Thus, one could hope for cross-fertilization of ideas between the models that could facilitate progress on both fronts.

Exponential lower bounds for ROABPs were known since their inception [Nis91], and a white-box polynomial-time PIT algorithm was given by Raz and Shpilka [RS05]. In the black-box setting, hitting sets of quasipolynomial size were obtained in [FS13b, FSS14, AGKS15], where the last two papers being applicable even if the order in which the variable are read is unknown. This marks a striking difference between the algebraic model and the boolean model. Indeed, in the boolean domain, pseudorandom generators for read-once branching programs in unknown order are much weaker, in terms of the seed length, than Nisan's generator [Nis92] which works only if the order is known. Recently, Gurjar et al. obtained PIT algorithms for sum of ROABPs [GKST16].

In this work, we consider the natural next step, which are read-k oblivious algebraic branching programs. This model generalizes and extends both the models of read-k arithmetic formulas and sums of ROABPs. We are able to prove exponential lower bounds and to give subexponential-time PIT algorithms for this model. Prior to our work there were no results known for this model. A summary of our results appears in Section 1.2.

1.1 Computational Models

In this section we define the computational models we consider in this work. We begin with the definition of *Algebraic Branching Programs* (ABPs).

Definition 1.1 (Algebraic Branching Program, [Nis91]). *An* Algebraic Branching Program (ABP) *is a directed acyclic graph with one vertex s of in-degree zero (the* source*) and one vertex t of out-degree zero (the* sink*). The vertices of the graph are partitioned into layers labeled $0, 1, \dots, L$. Edges in the graph can only go from layer $\ell - 1$ to layer ℓ, for $1 \leq \ell \leq L$. The source is the only vertex at layer 0 and the sink is the only vertex at layer L. Each edge is labeled with a polynomial over a field $\mathbb{F}$ in the input variables $\{x_1, x_2, \dots, x_n\}$. Each path from s to t computes the polynomial that is the product of the labels of the path edges, and the ABP computes the sum, over all s to t paths, of such polynomials.*

The width *of an ABP is the maximum number of nodes in any layer, and the* size *of an ABP is the number of vertices in the ABP. The* degree *of an ABP is defined to be the maximum degree of the polynomial edge labels.* $\Diamond$

The expressive power of ABPs lies between arithmetic formulas and arithmetic circuits. Every formula of size s can be simulated by an ABP of size s. Similarly, an ABP of width w, degree d and L layers can be simulated by an arithmetic circuit of size $O(wdL^2)$.

In this work we consider a restricted model of ABPs that we call read-k oblivious ABPs. In an oblivious ABP the labels in each layer must be univariate polynomials in the same variable. We also restrict each variable to be read in at most k layers while still allowing them to label any number of the edges in those layers.

Definition 1.2 (Read-k Oblivious ABPs, [FS13b]). *An algebraic branching program is said to be* oblivious *if for every layer ℓ, all the edge labels in that layer are univariate polynomials in a variable x_{i_ℓ}.*

Such a branching program is said to be a read-once *oblivious ABP (ROABP) if the x_{i_ℓ}'s are distinct variables, that is, each x_i appears in the edge labels of at most one layer.*

An oblivious ABP is said to be a read-k *if each variable x_i appears in the edge labels of at most k layers.* $\Diamond$

Remark 1.3. *For the remainder of the article, it will be convenient to assume that in a read-k oblivious ABP, every variable x appears in* exactly *k layers. This assumption can be made without loss of generality, since if x appears in $k' < k$ layers, we can add $k - k'$ "identity" layers to the program that vacuously read x. This transformation does not increase the width of the program and increases the length by no more than kn.* $\Diamond$

A special case of read-k oblivious ABPs is one where the ABPs make "multiple passes" over the input.

Definition 1.4 (k-pass ABPs). *An oblivious ABP is said to be a k-*pass ABP *if there exists a permutation π on $[n] = \{1, 2, \dots, n\}$ such that the ABP reads variables in the order*

$$\underbrace{x_{\pi(1)}, \dots, x_{\pi(n)}, x_{\pi(1)}, \dots, x_{\pi(n)}, \dots, x_{\pi(1)}, \dots, x_{\pi(n)}}_{k \text{ times}}.$$

An oblivious ABP is said to be a k-pass varying-order ABP *if there are permutations* $\pi_1, \cdots, \pi_k$ *over* $[n]$ *such that the ABP reads variables in the order*

$$x_{\pi_1(1)}, \ldots, x_{\pi_1(n)}, x_{\pi_2(1)}, \ldots, x_{\pi_2(n)}, \ldots, x_{\pi_k(1)}, \ldots, x_{\pi_k(n)}. \qquad \Diamond$$

1.2 Our Results

We give various results about the class of read-k oblivious ABPs, including lower bounds, PIT algorithms and separations.

Lower Bounds: We show an explicit polynomial f such that for bounded k any read-k oblivious ABP computing f must be of exponential width.

Theorem 1.5 (proved in Section 4)**.** *There exists an explicit polynomial f, which is computed by a depth-3 polynomial-size multilinear circuit, such that any read-k oblivious ABP computing f must have width* $\exp(n/k^{O(k)})$.

Prior to this work, there were no lower bounds for this model.

Identity Testing: For the class of k-pass ABPs with bounded k, we provide a black-box PIT algorithm that runs in quasipolynomial time.

Theorem 1.6 (proved in Section 5.1)**.** *There exists a black-box PIT algorithm for the class of n-variate, degree-d, and width-w k-pass ABPs that runs in time* $(nw^{2k}d)^{O(\log n)}$.

For the more general class of read-k oblivious ABPs, we provide a white-box PIT algorithm that runs in subexponential time.

Theorem 1.7 (proved in Section 5)**.** *There exists a white-box PIT algorithm for the class of n-variate, degree-d, and width-w read-k oblivious ABPs that runs in time* $(nwd)^{\tilde{O}(n^{1-1/2^{k-1}})\cdot\exp(k^2)}$. *Furthermore, white-box access is only needed to know the order in which the variables are read. That is, given this order, we construct an explicit hitting set of the above size for the class of read-k oblivious ABPs that read their variables in that order.*

Separations: Recently, Kayal, Nair and Saha [KNS16] constructed a polynomial f that can be computed by a sum of *two* ROABPs in different orders, each of constant width, such that any ROABP computing f must be of width $2^{\Omega(n)}$. Note that sum of two ROABPs is a special case of a 2-pass varying-order ABP.

In order to exemplify the strength of the multiple-reads model, we show a polynomial that can be computed by a small 2-pass varying-order ABP, but cannot be computed by a small sum of ROABPs of small width.

Theorem 1.8 (proved in Section 3)**.** *There exists an explicit polynomial f on n^2 variables that is computed by a 2-pass varying-order ABP of constant width, but any sum of k ROABPs computing f must be of width* $\exp(\Omega(\sqrt{n}/2^k))$.

1.3 Related Work

Algebraic Models As mentioned before, Nisan [Nis91] proved exponential lower bounds for ROABPs, and Raz and Shpilka [RS05] gave a white-box polynomial-time PIT algorithm for this model.

Forbes and Shpilka [FS13b] were the first to consider the black-box version of this problem, and obtained a hitting set of size $(nwd)^{O(\log n)}$, for n-variate, degree-d and width-w ROABPs, if the order in which the variables are read is known in advance. Forbes, Shpilka and Saptharishi [FSS14] obtained a hitting set of size $(nwd)^{O(d \log(w) \log n)}$ for *un*known order ROABPs. This was improved later by Agrawal et al. [AGKS15] who obtained a hitting set of size $(nwd)^{O(\log n)}$ which matches the parameters of the known-order case.

For models that read variables more than once, much less was known. Gurjar et al. [GKST16] considered the model of a *sum* of k ROABPs, and obtained a white-box algorithm that runs in time $(ndw^{2^k})^{O(k)}$, and a black-box algorithm that runs in time $(ndw)^{O(k2^k \log(ndw))}$, so that the running time is polynomial in the former case and quasipolynomial in the latter, when k is constant. A sum of k ROABPs can be simulated by read-k oblivious ABPs, and we show (in Section 3) that read-k oblivious ABPs are in fact strictly stronger.

Lower bounds against the model of sums of ROABPs were obtained in a recent work of Arvind and Raja [AR16], who showed that for every constant $\varepsilon > 0$, if the permanent is computed by a sum of $n^{1-\varepsilon}$ ROABPs, then at least one of the ROABPs must be of width $2^{\Omega(n^{\varepsilon})}$.

We also mention an earlier work of Jansen et al. [JQS10b], who also gave white-box and black-box tests for the weaker model of sum of constantly many read-once ABPs, where in their definition every variable is allowed to label only a single *edge* in the ABP. In a follow-up work [JQS10a], the same authors extended the results for read-k ABPs in which all paths read the variables in the same order, and again, k is a bound on the *total* number of labels on which a variable is allowed to appear.

Another model which is subsumed by oblivious read-k ABPs is that of bounded-read formulas. Shpilka and Volkovich [SV15] constructed quasipolynomial-size hitting set for read-once formulas. This was later improved by Minahan and Volkovich [MV17] who obtained a polynomial-size hitting set. Anderson, van Melkebeek and Volkovich [AvMV15] extended this line of results to multilinear read-k formulas and obtained a polynomial-time white-box algorithm and quasipolynomial-time black-box algorithm. The natural simulation of read-k formulas by ABPs produces an ABP in which every variable labels at most k edges, and it can be seen that such programs can be converted to read-k oblivious ABPs with only a polynomial overhead.

To conclude, earlier results apply only to restricted submodels of read-k oblivious ABPs.

Boolean Models Let us now make a small detour and consider the boolean analogs for our models. A (boolean) branching program is a directed acyclic graph with a source node s and two sink nodes, t_0 and t_1. Each internal *node* is labeled by a variable x_i with two outgoing edges, labeled 0 and 1. The program computes a boolean function on an input $(x_1, \ldots, x_n) \in \{0,1\}^n$ by following the corresponding path along the program.

A read-k-times boolean branching program is allowed to query every variable at most k times along every path from the source to the sink. Note that this is more general than our definition of read-k oblivious algebraic branching programs because each variable is only restricted to appear at most k times on a each path, rather than in at most k layers. Further distinction is made in the boolean case between *semantic* read-k boolean branching programs, in which this restriction is enforced only on paths that are consistent with some input, and *syntactic* read-k boolean branching programs, in which this restriction applies for all paths (further note that in the read-once case, there is no distinction between the syntactic and the semantic model because every path is consistent with some input). Our model of read-k oblivious ABPs does not permit an analogous distinction.

Exponential lower bounds for read-once boolean branching programs for explicit functions have been known since the 1980's [Zák84, BHST87, Weg88], even for functions that are computed by a polynomial-size read-twice boolean branching program.

Okolnishnikova [Oko91], and Borodin, Razborov and Smolensky [BRS93] extended these results and obtained exponential lower bounds for syntactic read-k-times boolean branching programs, by giving an explicit boolean function f such that every syntactic read-k-times branching program for f has size $\exp(n/2^{O(k)})$ (in fact, the lower bound in the second work also holds for the stronger class of non-deterministic branching programs).

A strong separation result was obtain by Thathachar [Tha98], who showed a *hierarchy theorem* for syntactic read-k-times boolean branching program, by giving, for every k, a boolean function f which is computed by a linear-size syntactic read-$(k+1)$-times branching program such that every syntactic read-k-times branching program computing f must have size $\exp(\Omega(n^{1/k}/2^{O(k)}))$.

The semantic model seemed more difficult, but nevertheless Ajtai [Ajt05] was able to prove an exponential lower bound for semantic read-k-times programs (when k is constant), which was extended by Beame at al. [BSSV03] to randomized branching programs.

Derandomizing PIT is the algebraic analog of constructing pseudorandom generators (PRGs) for boolean models. A PRG for a class $\mathcal{C}$ of boolean circuits is an easily computable function $G : \{0,1\}^{\ell} \to \{0,1\}^{n}$, such that for any circuit $C \in \mathcal{C}$, the probability distributions $C(U_n)$ and $C(G(U_\ell))$ are ε-close, where U_m is the uniform distribution over $\{0,1\}^{m}$.

Nisan [Nis92] constructed a PRG for polynomial-size read-once oblivious branching programs with seed length $O(\log^2 n)$. This was followed by a different construction with the same seed length by Impagliazzo, Nisan and Wigderson [INW94]. However, for the constructions to work it is crucial that the order in which the variables are read is known in advance.

Beyond that, and despite a large body of work devoted to this topic [BDVY13, BPW11, BRRY14, De11, GMR⁺12, IMZ12, KNP11, RSV13, Ste12, SVW14], all the results for the unknown order case or for read-k oblivious branching programs have much larger seed length, unless further structural restrictions are put on the program (such as very small width, regularity, or being a permutation branching program). Specifically, we highlight that even for read-2 oblivious branching programs, the best result is by Impagliazzo, Meka and Zuckerman [IMZ12] who gave a PRG with seed length $s^{1/2+o(1)}$ for size s branching program (note that the the dependence here is on s rather than on n). In particular, no non-trivial results are known for general polynomial-size read-2 oblivious boolean branching program.

1.4 Proof Technique

Before delving into the details of our proof technique it is perhaps instructive to think again about read-once branching programs. Because read-once branching programs can only read each variable once we can split their computation into two subcomputations over disjoint sets of variables. The interface between the two subcomputations is a "window" of vertices limited by the width

w of the branching program. If w is small relative to the number of variables, then the branching program must "forget" most of the values of the variables it read during the first subcomputation before it proceeds with the second subcomputation. This read-once limitation has been successfully exploited by divide-and-conquer strategies to derandomize read-once branching programs in both the boolean [Nis92] and algebraic [JQS10b, FS13b, FSS14, AGKS15] models.

1.4.1 Evaluation dimension and ROABPs

Unfortunately, the above intuition breaks down when we allow a variable to be read multiple times, and this model requires a different strategy. Our main starting point is the observation that, perhaps surprisingly, multiple "passes" over the input variables, *in the same order*, do not provide the program with much additional power. That is, a k-pass ABP can be simulated by a ROABP, with a blow-up which is exponential in k (hence, only a polynomial blow-up, if k is constant).

This fact can be directly seen through analysis of the *evaluation dimension* measure. For a polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ and a subset of variables $\mathbf{y} \subseteq \{x_1, \ldots, x_n\}$, we denote by $\mathrm{eval}_{\mathbf{y}}(f)$ the $\mathbb{F}$-linear subspace of $\mathbb{F}[x_1, \ldots, x_n]$ that is spanned by the polynomials obtained from f by fixing the variables in $\mathbf{y}$ to arbitrary elements in $\mathbb{F}$. The evaluation dimension of f with respect to a partition $\mathbf{y}, \overline{\mathbf{y}}$ of the variables, which is denoted $\mathrm{evalDim}_{\mathbf{y},\overline{\mathbf{y}}}(f)$, is the dimension of $\mathrm{eval}_{\mathbf{y}}(f)$. Over large enough fields, this dimension equals the rank of the *partial derivative matrix* associated with this partition as defined by Nisan [Nis91]. In many contexts, however, it is easier to work with the evaluation dimension. We refer to Chapter 4 of [For14] for a detailed and formal discussion on this equivalence.

The importance of the evaluation dimension measure stems from the fact that f can be computed by a width-w ROABP in the order $x_1, x_2, \ldots, x_n$ if and only if $\mathrm{evalDim}_{\{x_1,\ldots,x_i\},\{x_{i+1},\ldots,x_n\}}(f) \le w$ for every $1 \le i \le n$ (see Theorem 2.3). Thus, this measure provides a precise characterization for the amount of resources needed to compute a polynomial in this model.

1.4.2 Evaluation dimension and k-pass ABPs

We are able to adapt the proof of the "only if" part of the above fact in order to show that if f is computed by a k-pass ABP (recall, here the ABP reads the n variables k times in the same order) then $\mathrm{evalDim}_{\{x_1,\ldots,x_i\},\{x_{i+1},\ldots,x_n\}}(f) \le w^{2k}$ for every $i \in [n]$. That is, k passes over the input in the same order cannot create many independent evaluations. Then, using the "if" part of the equivalence, it follows that f can also be computed using a ROABP of width w^{2k} (see Lemma 2.5).

This immediately implies a hitting set of size $(ndw^{2k})^{O(\log n)}$ for the class of k-pass ABPs (Theorem 1.6). It also implies exponential lower bounds for this model, simply by applying the corresponding results for ROABPs. It is still not clear, however, how to extend this to the general case, since even read-2 oblivious ABPs are exponentially stronger than ROABPs. Indeed, recall that [KNS16] give an exponential separation between a sum of two ROABPs and ROABPs. We also give an exponential separation between 2-pass varying-order ABPs from sums of ROABPs in Section 3.

1.4.3 PIT for read-k oblivious ABPs

Before discussing our identity test for read-k oblivious ABPs let us focus, for the time being, on the simplest instance of the more general problem, by considering a 2-pass varying-order ABP computing a non-zero polynomial f. That is, consider an ABP A of width w that, without loss of generality, reads the variables in the order $x_1, x_2, \ldots, x_n, x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(n)}$, for some permutation π. As mentioned above, we cannot possibly hope to simulate any such branching program by a small ROABP. We can, however, find a large subset of the variables $\mathbf{y}$ such that if we fix all the other variables arbitrarily the resulting polynomial has a small ROABP. Equivalently, we can instead think of f as a polynomial in the variables of $\mathbf{y}$ over the field of rational functions $\mathbb{F}(\overline{\mathbf{y}})$, that is, $f \in \mathbb{F}(\overline{\mathbf{y}})[\mathbf{y}]$.

By the well-known Erdős–Szekeres Theorem [ES35], any sequence of n distinct integers contains a subsequence of length $\sqrt{n}$ that is either monotonically increasing or monotonically decreasing. Applied to the sequence $x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(n)}$, with respect to the natural order $x_1 < x_2 < \cdots < x_n$, we get a monotone subsequence of variables. For the sake of exposition, assume that this subsequence is monotonically increasing (the case of a decreasing sequence is, somewhat counter-intuitively, even simpler). Let $\mathbf{y} = \left\{y_1, \ldots, y_{\sqrt{n}}\right\}$ be the set of $\sqrt{n}$ variables that appear in this monotone subsequence. If we fix all the variables in $\overline{\mathbf{y}}$ to elements of $\mathbb{F}$, we are left, by the monotonicity property, with a branching program that reads the variables in the order $y_1, y_2, \ldots, y_{\sqrt{n}}, y_1, y_2, \ldots, y_{\sqrt{n}}$. Observe that this is exactly a 2-pass branching program! Moreover, the previous discussion on k-pass ABPs applies here, and if f is non-zero, we can efficiently find an assignment to the variables in $\mathbf{y}$ from $\mathbb{F}$ that keeps the polynomial non-zero. Having reached this point, we can "resurrect" the variables in $\overline{\mathbf{y}}$. Note that we are left with a non-zero 2-pass varying-order ABP on only $n - \sqrt{n}$ variables. From here we apply the same argument repeatedly to this 2-pass varying-order ABP and after $O(\sqrt{n})$ iterations we are guaranteed to find an assignment (to all n variables) on which f evaluates to a non-zero output.

In each iteration we construct a hitting set of size $(nwd)^{O(\log n)}$ for width-poly(w) ROABPs. The final hitting set is the Cartesian product of the hitting sets produced by each of the $O(\sqrt{n})$ iterations, and hence the total size of the hitting set is $(nwd)^{\tilde{O}(\sqrt{n})}$, as promised by Theorem 1.7.

Generalizing the argument above for k-pass varying-order ABPs is fairly straightforward, and is done using repeated applications of the Erdős–Szekeres Theorem to each of the k pass sequences in order. This produces a subsequence on a subset $\mathbf{y}$ of the variables which is monotone in every pass. Hence, as before, when the branching program A is restricted to $\mathbf{y}$ it becomes a k-pass ABP. The repeated application of the Erdős–Szekeres theorem weakens the lower bound on the size of $\mathbf{y}$ from $\sqrt{n}$ to $n^{1/2^{k-1}}$, and accounts for most of the loss in the parameters in our theorem.[2]

In order to handle general read-k oblivious ABPs, we need additional ideas. We observe that after repeatedly applying the Erdős–Szekeres Theorem to the subsequence of every "read", we do not get a k-pass ABP as before, but rather k monotone sequences that are intertwined together. We next show that by discarding more variables, but not too many, we get a structure that we call a "k-regularly interleaving sequence". This is a technical notion which is presented in full detail in Section 5, but the main point is that this definition allows us to argue that the obtained read-k oblivious ABP has a (small) evaluation dimension and therefore it can be simulated by a not-too-large ROABP. Obtaining this k-regularly interleaving property is the main technical difficulty of the proof.

[2]This lower bound on the length of a subsequence which is monotone in every pass is the best possible. This fact is attributed to de Bruijn (unpublished, see [Kru53]), and the actual construction which shows that the lower bound is tight appears in [AFK85].

1.4.4 Lower bounds for read-k oblivious ABPs

The arguments above that give PIT algorithms already give lower bounds for read-k oblivious ABPs. We have shown that if f is computed by a 2-pass varying-order ABP of width w, then there exist a subset of $\sqrt{n}$ variables $\mathbf{y}$ such that f is computed by an ROABP of width w^4 over $\mathbb{F}(\overline{\mathbf{y}})$. This implies that if we pick f so that every restriction to $\sqrt{n}$ variables has an exponential (in $\sqrt{n}$) lower bound for ROABPs, we would receive a subexponential lower bound for computing f in a 2-pass varying-order ABP. These arguments, again, generalize to read-k oblivious ABPs.

In order to get an exponential lower bound (Theorem 1.5), we observe that we do not need to bound the evaluation dimension for *every* prefix (namely, to show that a subset of the variables is computed by a small ROABP), but only to show that the evaluation dimension is small for *some* prefix. This is much easier to achieve since we do not need the order of the reads to be "nicely-behaved" with respect to *every* prefix, but just with respect to *a* prefix.

In particular, we invoke a simple averaging argument to show that if f is computed by a width-w read-k oblivious ABP, then there exist sets of variables $\mathbf{y}$ (of size at least $n/k^{O(k)}$) and $\mathbf{z}$ (of size at most $n/10$), so that whenever we fix the variables in $\mathbf{z}$ we get that $\mathrm{evalDim}_{\mathbf{y},\overline{\mathbf{y}}}(g) \leq w^{2k}$, where g is any restriction of f obtained by fixing the variables in $\mathbf{z}$. We then construct an explicit polynomial whose evaluation dimension with respect to every set remains large, even after arbitrarily fixing a small set of the variables (see Theorem 4.3).

1.4.5 Separating 2-pass ABPs from sums of ROABPs

In order to prove the separation between 2-pass varying-order ABPs and sums of k ROABPs (Theorem 1.8), we use a structural result proved by Gurjar et al. [GKST16] that gives a way to argue by induction on ROABPs. Given a polynomial f which is computed by a sum $h_1 + h_2 + \cdots h_k$ of ROABPs of width w, we would like to find a related polynomial f' that is computed by a sum of $k-1$ ROABPs of perhaps slightly larger width. Here, the evaluation dimension plays a role as well. The way to do this is to pick a non-trivial linear combination of $w+1$ partial evaluations of f that make h_1 zero, which is possible since h_1 has a small evaluation dimension with respect to prefixes of variables corresponding to the order in which the variables are read in h_1. One can then show that, having eliminated h_1, each of the other summands can still be computed by a ROABP of width $w(w+1)$.

We provide a simple polynomial computed by a 2-pass varying-order ROABP whose partial evaluations are complex enough in the sense that they contain many linear independent evaluations and also a "scaled-down" version of the original polynomial as a projection. It then follows by induction, using the above arguments, that this polynomial cannot be computed by a small sum of small ROABPs (see Lemma 3.6).

1.5 Organization

We start with some preliminaries and useful facts about the evaluation dimension in Section 2 that almost all the results in this article rely on. In Section 3, we present the separation between the class of 2-pass varying-order ABPs and sums of ROABPs. Following that, in Section 4, we present an exponential lower bound for the class of read-k oblivious ABPs. Then in Section 5 we present the white-box PIT for read-k oblivious ABPs. Finally, we conclude with some open problems in Section 6.

2 Preliminaries

2.1 Notation

For $n \in \mathbb{N}$, we denote by $[n]$ the set $\{1, 2, \ldots, n\}$. For disjoint subsets S, T we denote by $S \sqcup T$ their disjoint union.

We commonly denote by $\mathbf{x}$ a set of n indeterminates $\{x_1, \ldots, x_n\}$, where the number of indeterminates n is understood from the context. Although $\mathbf{x}$ is a set we often use the natural ordering of its elements, $x_1 < x_2 < \ldots < x_n$, to treat it and its subsets as sequences. For $i \in [n]$, we use $\mathbf{x}_{[i]}$ to denote the set $\{x_1, \ldots, x_i\}$ which is a prefix of $\mathbf{x}$. For a subset $\mathbf{y} \subseteq \mathbf{x}$ of variables, we denote its complement by $\overline{\mathbf{y}}$.

For a polynomial $f \in \mathbb{F}[\mathbf{x}]$, a set of variable $\mathbf{y} \subseteq \mathbf{x}$ and vector $\mathbf{a} = (a_1, \ldots, a_{|\mathbf{y}|}) \in \mathbb{F}^{|\mathbf{y}|}$, we denote by $f|_{\mathbf{y}=\mathbf{a}}$ the restriction of f obtained by fixing the j-th variable in $\mathbf{y}$ to a_j.

In our PIT algorithm, we need to combine hitting sets for smaller sets of variables. Hence, for a partition of $\mathbf{x}$, $\mathbf{y}_1 \sqcup \mathbf{y}_2 \sqcup \cdots \sqcup \mathbf{y}_m = [n]$, and sets $\mathcal{H}_i \subseteq \mathbb{F}^{|\mathbf{y}_i|}$, we denote by $\mathcal{H}_1^{\mathbf{y}_1} \times \cdots \times \mathcal{H}_m^{\mathbf{y}_m}$ the set of all vectors in $\mathbb{F}^n$ whose restriction to the coordinates indexed by $\mathbf{y}_i$ is an element of $\mathcal{H}_i$ (note that here we naturally associate a subset $\mathbf{y} \subseteq \mathbf{x}$ with a subset of indices in $[n]$). That is

$$\mathcal{H}_1^{\mathbf{y}_1} \times \cdots \times \mathcal{H}_m^{\mathbf{y}_m} = \left\{ v \in \mathbb{F}^n \ : \ \forall i \in [m],\ v|_{\mathbf{y}_i} \in \mathcal{H}_i \right\}.$$

2.2 ABPs and iterated matrix products

The computation of an ABP corresponds to iterated multiplication of matrices of polynomials. In the case of oblivious branching programs, the ABP computes an iterated matrix product of univariate matrices. We record this fact as a lemma, and refer to [For14] for a proof and a detailed discussion on this subject.

Lemma 2.1. *Suppose f is a polynomial computed by an oblivious ABP A of width w and length ℓ, that reads the variables in some order $x_{i_1}, x_{i_2}, \ldots, x_{i_\ell}$. Then f is the $(1,1)$ entry of a matrix of the form*

$$A_1(x_{i_1}) \cdot A_2(x_{i_2}) \cdots A_\ell(x_{i_\ell})$$

where for every $j \in [\ell]$, $A_j \in \mathbb{F}[x_{i_j}]^{w \times w}$ is a $w \times w$ matrix in which each entry is a univariate polynomial in x_{i_j}. □

2.3 Evaluation dimension and ROABPs

We now define a complexity measure for polynomials that is useful for analyzing read-k oblivious ABPs.

Definition 2.2 (Evaluation dimension)**.** *Let $f \in \mathbb{F}[\mathbf{x}]$ be a polynomial and $\mathbf{y} \subseteq \mathbf{x}$ be subset of variables. We define* $\mathrm{eval}_{\mathbf{y}}(f)$ *to be*

$$\mathrm{eval}_{\mathbf{y}}(f) = \mathrm{span}\left\{ f|_{\mathbf{y}=\mathbf{a}} \ : \ \mathbf{a} \in \mathbb{F}^{|\mathbf{y}|} \right\} \subseteq \mathbb{F}[\overline{\mathbf{y}}],$$

which is the space of polynomials spanned by all partial evaluations of the $\mathbf{s}$ *variables in f.*

If $\mathbf{x} = \mathbf{y} \sqcup \mathbf{z} \sqcup \mathbf{w}$ *we define the* evaluation dimension of f with respect to $\mathbf{y} \sqcup \mathbf{z}$ over $\mathbb{F}(\mathbf{w})$, *which shall be denoted by* $\mathrm{evalDim}_{\mathbf{y},\mathbf{z};\mathbf{w}}(f)$, *as the dimension of the space* $\mathrm{eval}_{\mathbf{y}}(f)$ *when taken over the field of* *rational functions $\mathbb{F}(\mathbf{w})$. That is, we first "move" the variables $\mathbf{w}$ into the field and treat them as constants, and then consider the dimension of* $\mathrm{eval}_{\mathbf{y}}(f)$ *over $\mathbb{F}(\mathbf{w})$.*

In the special case where $\mathbf{w} = \emptyset$, we shall just use the notation $\mathrm{evalDim}_{\mathbf{y},\mathbf{z}}(f)$. $\Diamond$

If $|\mathbb{F}| > \deg(f)$, then $\mathrm{evalDim}_{\mathbf{y},\mathbf{z}}(f)$ is the rank of the *partial derivative matrix with respect to* $\mathbf{y}$,$\mathbf{z}$, as defined by Nisan [Nis91]. The rows of the partial derivative matrix are indexed by monomials $m_{\mathbf{y}}$ over the variables $\mathbf{y}$ and its columns are indexed by monomials $m_{\mathbf{z}}$ over the variables $\mathbf{z}$. The $(m_{\mathbf{y}}, m_{\mathbf{z}})$ entry is the coefficient of $m_{\mathbf{y}}m_{\mathbf{z}}$ in the polynomial f. Although these two perspectives are equivalent, the formulation via evaluation dimension is sometimes easier to work with. The evaluation dimension measure is useful when arguing about ROABPs since it characterizes the width needed to compute a polynomial f using a ROABP.

Theorem 2.3 ([Nis91], and see also [For14])**.** *Let f be a polynomial on $\mathbf{x} = \{x_1, \ldots, x_n\}$ with individual degree at most d, and suppose for every $i \in [n]$ we have $\mathrm{evalDim}_{\mathbf{x}_{[i]},\overline{\mathbf{x}_{[i]}}}(f) \leq w$. Then, there is a ROABP of width w in the order $x_1, \ldots, x_n$ with individual degree at most d that computes f.*

Conversely, if $\mathrm{evalDim}_{\mathbf{x}_{[i]},\overline{\mathbf{x}_{[i]}}}(f) = w$, then in any ROABP that computes f in the order $x_1, x_2, \ldots, x_n$, the width of the i-th layer must be at least w.

Below is an example of a polynomial that has large evaluation dimension with respect to a specific subset of the variables. This example is helpful not only because it is simple and illustrates evaluation dimension, but also because all of our constructions of hard polynomials are based on a reduction to this polynomial.

Lemma 2.4. *Let $f(\mathbf{u}, \mathbf{v}, \mathbf{w})$ be a polynomial of the form*

$$f = \left(\prod_{i=1}^{t}(\ell_i(\mathbf{v}) + \ell'_i(\mathbf{u}))\right) \cdot g(\mathbf{u}, \mathbf{w}),$$

where:

1. *For every $\mathbf{a} \in \mathbb{F}^{|\mathbf{u}|}$, it holds that $g|_{\mathbf{u}=\mathbf{a}} = g(\mathbf{a}, \mathbf{w}) \not\equiv 0$.*
2. *$\{\ell_i\}_{i=1}^t$ is a set of linearly-independent linear polynomials, and so is $\{\ell'_i\}_{i=1}^t$.*

Then $\mathrm{evalDim}_{\mathbf{u},\mathbf{v}\sqcup\mathbf{w}}(f) \geq 2^t$.

Proof. Since the ℓ_i's and ℓ'_i's are linearly independent we can apply linear transformations to the variables, which cannot increase the dimension, so that for all $i \in [t]$, $\ell_i(v)$ maps to v_i and $\ell'_i(u)$ maps to u_i. Hence, we may assume without loss of generality that

$$f = \left(\prod_{i=1}^{t}(v_i + u_i)\right) \cdot g(\mathbf{u}, \mathbf{w}).$$

Additionally, for any $\mathbf{a} = (a_1, \ldots, a_t) \in \{0,1\}^t$ we can fix $\mathbf{u}$ to $\mathbf{a}$ in f so that

$$f|_{\mathbf{u}=\mathbf{a}} = \left(\prod_{i=1}^{t}(v_i + a_i)\right) \cdot g|_{\mathbf{u}=\mathbf{a}}(\mathbf{w}) \in \mathrm{eval}_{\mathbf{u}}(f).$$

The 2^t polynomials $\left\{\prod_{i=1}^t (v_i + a_i) \;:\; \mathbf{a} \in \{0,1\}^t\right\}$ are linearly independent. Further, by the assumption on g, we also have that $g(\mathbf{a}, \mathbf{w})$ is non-zero. Hence these polynomials (in $\mathbf{v}$) remain linearly independent even when multiplied by the variable-disjoint polynomial $g(\mathbf{a}, \mathbf{w})$ and so $\mathrm{evalDim}_{\mathbf{u},\mathbf{v}\sqcup\mathbf{w}}(f) \geq 2^t$. $\square$

The following simple lemma is an illustration of using the evaluation dimension of a polynomial to obtain a small ROABP for that polynomial.

Lemma 2.5. *Let $f \in \mathbb{F}[x_1, \dots, x_n]$ be a polynomial computed by a k-pass ABP of width w and degree d, with the variables read in order $x_1, x_2, \dots, x_n$. Then f can be computed by a width-w^{2k} read-once ABP with degree dk and the same read order.*

Proof. Let A be the k-pass ABP of width w and degree d that computes f. Recall that for any $i \in [n]$, we denote $\mathbf{x}_{[i]} = \{x_1, \dots, x_i\}$. The assumption on A implies that the individual degree of each variable is at most dk, and so by Theorem 2.3, it is enough to show that for any $i \in [n]$,

$$\text{evalDim}_{\mathbf{x}_{[i]}, \overline{\mathbf{x}_{[i]}}}(f) \leq w^{2k}.$$

By the assumption on A and by Lemma 2.1, for every $i \in [n]$ and $j \in [k]$ there exists a matrix $M^{i,j} \in \mathbb{F}[x_i]^{w \times w}$ such that the entries of $M^{i,j}$ are univariate polynomials in x_i of degree d and

$$f = \left(M^{1,1}(x_1) M^{2,1}(x_2) \cdots M^{n,1}(x_n) M^{1,2}(x_1) M^{2,2}(x_2) \cdots M^{n,k}(x_n) \right)_{1,1}.$$

Fix $i \in [n]$, and consider any assignment of the form $\mathbf{x}_{[i]} = \mathbf{a}$ for $\mathbf{a} = (a_1, \dots, a_i) \in \mathbb{F}^i$. Having fixed $\mathbf{x}_{[i]}$, we get that for some k matrices $N_1(\mathbf{a}), \dots, N_k(\mathbf{a})$, that depend on $\mathbf{a}$,

$$\begin{aligned} f|_{\mathbf{x}_{[i]}=\mathbf{a}} = \Big(& N_1(\mathbf{a}) \cdot M^{i+1,1}(x_{i+1}) \cdots M^{n,1}(x_n) \cdot N_2(\mathbf{a}) \cdot M^{i+1,2}(x_{i+1}) \cdots M^{n,2}(x_n) \\ & \cdots N_k(\mathbf{a}) \cdot M^{i+1,k}(x_{i+1}) \cdots M^{n,k}(x_n) \Big)_{1,1}. \end{aligned} \tag{2.6}$$

It follows that any polynomial $g(x_{i+1}, \dots, x_n) \in \text{eval}_{\mathbf{x}_{[i]}}(f)$ is completely determined by $N_1, \dots, N_k$ which have w^2 entries each. More precisely, let $\{B_1, \dots, B_{w^2}\}$ be a basis for $\mathbb{F}^{w \times w}$. For each $j \in [k]$, we can write $N_j(\mathbf{a}) \in \mathbb{F}^{w \times w}$ in (2.6) as a linear combination of $\{B_1, \dots, B_{w^2}\}$. Then, by expanding the matrix product in (2.6), we see that every polynomial of the form $f|_{\mathbf{x}_{[i]}=\mathbf{a}}$ (and as a consequence, every polynomial in $\text{eval}_{\mathbf{x}_{[i]}}(f)$) is spanned by the w^{2k} polynomials of the form

$$\left(B_{\sigma_1} \cdot M^{i+1,1}(x_{i+1}) \cdots M^{n,1}(x_n) \cdot B_{\sigma_2} \cdot M^{i+1,2}(x_{i+1}) \cdots M^{n,2}(x_n) \cdots B_{\sigma_k} \cdot M^{i+1,k}(x_{i+1}) \cdots M^{n,k}(x_n) \right)_{1,1}$$

for $\sigma_1, \dots, \sigma_k \in [w^2]$, which implies that $\text{evalDim}_{\mathbf{x}_{[i]}, \overline{\mathbf{x}_{[i]}}}(f) \leq w^{2k}$. By Theorem 2.3, the claim follows. □

In fact, the proof of Lemma 2.5 permits a slight generalization, by weakening the assumptions on the ABP, which is captured by the following definition.

Definition 2.7. *Let A be an ABP that computes a polynomial $f \in \mathbb{F}[x_1, \dots, x_n]$. We say that A has the* k-gap property *with respect to $\{x_1, \dots, x_i\}$, if there exist k matrices $M_1, \dots, M_k \in \mathbb{F}^{w \times w}[x_{i+1}, \dots, x_n]$ such that for every $\mathbf{a} \in \mathbb{F}^i$, there exists k matrices $N_1(\mathbf{a}), \dots, N_k(\mathbf{a}) \in \mathbb{F}^{w \times w}$ such that*

$$\begin{aligned} f|_{\mathbf{x}_i=\mathbf{a}} = \Big(& N_1(\mathbf{a}) \cdot M_1(x_{i+1}, \dots, x_n) \cdot N_2(\mathbf{a}) \cdot M_2(x_{i+1}, \dots, x_n) \\ & \cdots N_k(\mathbf{a}) \cdot M_k(x_{i+1}, \dots, x_n) \Big)_{1,1}. \end{aligned} \tag{2.8}$$

A is simply said to have the k-gap property if it has this property with respect to $\mathbf{x}_{[i]}$, for every $i \in [n]$. ◊

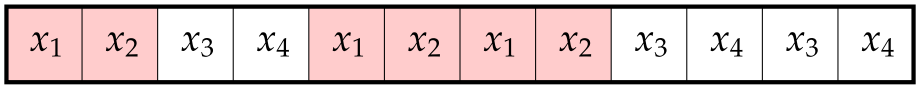
x_1 x_2 x_3 x_4 x_1 x_2 x_1 x_2 x_3 x_4 x_3 x_4

Figure 1: An ABP that reads the variables in this (left-to-right) order is a read-3 ABP that has the 2-gap property with respect to $\{x_1, x_2\}$. However, an ABP that reads the variables in this order does not have the 2-gap property with respect to the prefixes $\{x_1\}$ and $\{x_1, x_2, x_3\}$ (though it does have the 3-gap property for these prefixes).

Note that the k-gap property is implicitly defined with respect to the natural order of $\mathbf{x}$, $x_1 < x_2 < \cdots < x_n$. Figure 1 provides a pictorial explanation for the choice of this terminology. Using the same arguments as in the proof of Lemma 2.5, we obtain the following lemma.

Lemma 2.9. *Let $f \in \mathbb{F}[x_1, \ldots, x_n]$ be a polynomial computed by an ABP of width w of individual degree d that has the k-gap property. Then f can be computed by a width-w^{2k} read-once ABP, with individual degree dk, such that the variables are read in order $x_1, x_2, \ldots, x_n$.* □

3 Separating 2-pass ABPs from sums of ROABPs

In order to motivate our study of read-k oblivious ABP, we begin by demonstrating a polynomial that can be computed by a constant-width, 2-pass varying-order ABP, and yet cannot be computed by a small sum of polynomial-width ROABPs. Recall that every *sum* of k ROABPs can be realized by an oblivious read-k ABP. Thus, even a weak, but non-trivial, form of read-k oblivious ABPs, for $k = 2$, is already stronger than sums of ROABPs.

Suppose $\mathbf{x} = \{x_{1,1}, \ldots, x_{n,n}\}$ is a set of n^2 variables. It is useful to think of $\mathbf{x}$ as an $n \times n$ matrix X such that $x_{i,j}$ appears in the (i, j)-th entry. For every $m \in [n]$, define

$$\mathrm{rowSum}_m = \sum_j x_{m,j} \quad \text{and} \quad \mathrm{colSum}_m = \sum_i x_{i,m}.$$

Let

$$P_n(\mathbf{x}) = \left(\prod_{i=1}^{n} \mathrm{rowSum}_i\right) \cdot \left(\prod_{j=1}^{n} \mathrm{colSum}_j\right). \tag{3.1}$$

Observe that for all i, j, rowSum_i and colSum_j can be computed by width-2 ROABPs, and therefore $\prod_{i=1}^{n} \mathrm{rowSum}_i$ and $\prod_{j=1}^{n} \mathrm{colSum}_j$ can also be computed by a ROABP of the same width. It follows that their product P_n can be computed by a 2-pass varying-order ABP.

Theorem 3.2 (Restatement of Theorem 1.8)**.** *Let $P_n(x_{1,1}, \ldots, x_{n,n})$ be the n^2-variate polynomial defined in (3.1). For every $k > 0$, any sum of k ROABPs that computes it must have width $\exp(\sqrt{n}/2^k)$.*

The proof of this theorem exploits the structure of sums of few ROABPs that Gurjar, Korwar, Saxena and Thierauf [GKST16] used to construct hitting sets for this class of ABPs. Our argument requires the following lemma that is implicit in their result; for completeness, we provide a proof.

Lemma 3.3 (Implicit in [GKST16])**.** *Let $f = h_1 + \cdots + h_k \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ where each h_i is computed by a width-w ROABP in possibly different orders. Then, for every $0 < t < n$, there exists a subset $\mathbf{y} \subseteq \mathbf{x}$ of t variables such that for every set of $w + 1$ partial assignments $\mathbf{a}_1, \ldots, \mathbf{a}_{w+1} \in \mathbb{F}^t$, there is some non-trivial* *linear combination of $\{f|_{\mathbf{y}=\mathbf{a}_i}\}_{i=1}^{t}$ that is computable by a sum of $c-1$ ROABPs of width $w(w+1)$ in possibly different orders. That is, there exists $\alpha_1, \ldots, \alpha_{w+1} \in \mathbb{F}$, not all zero, such that*

$$\sum_{i=1}^{w+1} \alpha_i \cdot f|_{\mathbf{y}=\mathbf{a}_i} = h'_1 + \cdots + h'_{k-1}$$

where each h'_i is a ROABP of width at most $w(w+1)$.

Proof. Let $\mathbf{y}$ be the first t variables that are read in the ROABP that computes h_k. Since h_k is computed by a width-w ROABP, $\text{evalDim}_{\mathbf{y},\overline{\mathbf{y}}}(h_1) \leq w$. Hence, every $w+1$ partial evaluations $\{h_k|_{\mathbf{y}=\mathbf{a}_i}\}_{i=1}^{w+1}$ are linearly dependent, that is, there exist $\alpha_1, \ldots, \alpha_{w+1} \in \mathbb{F}$, not all zero, such that

$$\sum_{i=1}^{w+1} \alpha_i h_k|_{\mathbf{y}=\mathbf{a}_i} = 0. \tag{3.4}$$

Consider the polynomial $\sum_{i=1}^{w+1} \alpha_i \cdot f|_{\mathbf{y}=\mathbf{a}_i}$. By the assumption on f,

$$\begin{aligned}\sum_{i=1}^{w+1} \alpha_i \cdot f|_{\mathbf{y}=\mathbf{a}_i} &= \sum_{i=1}^{w+1} \alpha_i \sum_{j=1}^{k} h_j|_{\mathbf{y}=\mathbf{a}_i} \\ &= \sum_{i=1}^{w+1} \alpha_i \cdot h_k|_{\mathbf{y}=\mathbf{a}_i} + \sum_{j=1}^{k-1}\sum_{i=1}^{w+1} \alpha_i h_j|_{\mathbf{y}=\mathbf{a}_i} \\ &= \sum_{j=1}^{k-1}\sum_{i=1}^{w+1} \alpha_i h_j|_{\mathbf{y}=\mathbf{a}_i}, \end{aligned} \tag{3.5}$$

where the last equality follows from (3.4).

Hence, to prove the statement of the lemma it remains to be shown that for every $j \in [k-1]$, $\sum_{i=1}^{w+1} \alpha_i h_j|_{\mathbf{y}=\mathbf{a}_i}$ is computed by a ROABP of width $w(w+1)$. Fix such j. Observe that since h_j is computed by a ROABP of width w, for every $i \in [w+1]$ we have that $h_j|_{\mathbf{y}=\mathbf{a}_i}$ is computed by a ROABP of width w (by replacing the variables with the appropriate constants in the ABP that computes h_j), and furthermore all the ROABPs of the form $h_j|_{\mathbf{y}=\mathbf{a}_i}$ for $i \in [w+1]$ are in the same order (inherited from the order of the ROABP computing h_j).

Therefore, we can connect those $(w+1)$ ROABPs in parallel to obtain a single ROABP, of width $w(w+1)$, computing $\sum_{i=1}^{w+1} \alpha_i h_j|_{\mathbf{y}=\mathbf{a}_i}$. $\square$

The following lemma shows that the polynomial P_n defined in (3.1) has many linearly-independent partial evaluations.

Lemma 3.6. *Let $\mathbf{s}$ be a subset of $\mathbf{x} = \{x_{1,1}, \ldots, x_{n,n}\}$ of size $t < n$. Then there exists $r \geq 2^{\sqrt{t}}$ partial evaluations $\mathbf{a}_1, \cdots, \mathbf{a}_r \in \{0,1\}^t \subseteq \mathbb{F}^t$ such that the polynomials $\{P_n|_{\mathbf{s}=\mathbf{a}_1}, \ldots, P_n|_{\mathbf{s}=\mathbf{a}_r}\}$ are linearly independent.*

Furthermore, for any $g \in \text{span}\{P_n|_{\mathbf{s}=\mathbf{a}_i} : i \in [r]\}$, there is a set $\mathbf{y} \subseteq \mathbf{x} \setminus \mathbf{s}$ of $(n-t-1)^2$ variables, such that $P_{n-t-1}(\mathbf{y})$ can be obtained as a projection of g: namely, for $\mathbf{z} = \mathbf{x} \setminus (\mathbf{y} \cup \mathbf{s})$ we can find $\mathbf{a} \in \mathbb{F}^{n-|\mathbf{s}|-|\mathbf{y}|}$ such that $g|_{\mathbf{z}=\mathbf{a}} = \gamma \cdot P_{n-t-1}(\mathbf{y})$, for some non-zero $\gamma \in \mathbb{F}$.

Proof. Recall that we think of the n^2 variables as an $n \times n$ matrix X. By rearranging the rows and columns of X and relabeling, we can assume that all variables in $\mathbf{s}$ are present in the first a rows and first b columns. Observe that $a, b \leq t$. We can assume that $a \leq b$, as otherwise we can work with the transpose of X. Since $ab \geq t$, we also have $b \geq \sqrt{t}$. Additionally, any element of $\mathrm{eval}_{\mathbf{s}}(P_n)$ is divisible by $Q = \prod_{i>a} \mathrm{rowSum}_i \cdot \prod_{j>b} \mathrm{colSum}_j$ and hence we shall just work with

$$P' = \prod_{i=1}^{a} \mathrm{rowSum}_i \cdot \prod_{j=1}^{b} \mathrm{colSum}_j .$$

In the $[a] \times [b]$ sub-matrix, set all variables not in $\mathbf{s}$ to zero, and let P'' be the resulting polynomial. Clearly, it suffices to establish linear independence of partial evaluations of P''. We label the remaining variables in the first b columns by $\mathbf{u}$ if they belong to $\mathbf{s}$ and by $\mathbf{v}$ otherwise. The variables which are not in the first b columns are labeled by $\mathbf{w}$ (see Figure 2).

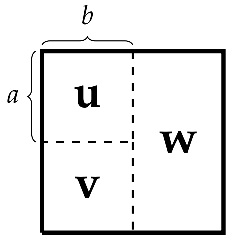

Figure 2: Labeling of variables in the matrix. Variables in the top $[a] \times [b]$ submatrix which are not in $\mathbf{s}$ are set to 0.

Then, we can write

$$P'' = \prod_{i=1}^{b} (\ell_i(\mathbf{v}) + \ell_i'(\mathbf{u})) \cdot g(\mathbf{u}, \mathbf{w}),$$

so that we have the properties:

1. For $i \neq j$, ℓ_i and ℓ_j are supported on disjoint sets of variables, because they correspond to different column sums, and similarly for ℓ_i' and ℓ_j'. In particular, each of the sets $\{\ell_i\}_{i=1}^{b}$ and $\{\ell_i'\}_{i=1}^{b}$ is linearly independent.
2. g is the product the first a row sums. This means that g is a product of variable-disjoint linear functions in $\mathbf{u}$ and $\mathbf{w}$ where each depends on at least one variable in $\mathbf{w}$. This implies that for any $\mathbf{a} \in \mathbb{F}^{|\mathbf{u}|}$, $g(\mathbf{a}, \mathbf{w}) \not\equiv 0$.

By Lemma 2.4, $\mathrm{evalDim}_{\mathbf{u}}(P'') \geq 2^b \geq 2^{\sqrt{t}}$. This implies that

$$\mathrm{evalDim}_{\mathbf{s}}(P') \geq \mathrm{evalDim}_{\mathbf{s}}(P'') \geq \mathrm{evalDim}_{\mathbf{u}}(P'') \geq 2^{\sqrt{t}}.$$

Hence, there exists $r \geq 2^{\sqrt{t}}$ evaluations $\mathbf{a}_1, \ldots, \mathbf{a}_r$ for the variables in $\mathbf{s}$ for which the set of polynomials $\{P_n'|_{\mathbf{s}=\mathbf{a}_1}, \ldots, P_n'|_{\mathbf{s}=\mathbf{a}_r}\}$ are linearly independent. It also follows that the set of polynomials $\{P_n|_{\mathbf{s}=\mathbf{a}_1}, \ldots, P_n|_{\mathbf{s}=\mathbf{a}_r}\}$ are linearly independent as well. This completes the first claim of the lemma.

Now observe that since Q divides each $P_n|_{\mathbf{s}=\mathbf{a}_i}$ it follows that any non-trivial linear combination of $\{P_n|_{\mathbf{s}=\mathbf{a}_1}, \ldots, P_n|_{\mathbf{s}=\mathbf{a}_r}\}$ is a non-zero multiple of Q. Let us fix one such linear combination $g =$

$h \cdot Q$ for a non-zero polynomial h. Note that h depends on just the variables in the first a rows and first b columns. By the Schwartz-Zippel-DeMillo-Lipton lemma [Sch80, Zip79, DL78] there exists an assignment to the variables in the first t rows and the first t columns that keeps $h \cdot Q$ non-zero. If $\mathbf{y}' = \{y_{ij} : i \in [n-t], j \in [n-t]\}$ is a relabeling of the variables in the last $(n-t)$ rows and columns, such an evaluation to the first t rows and columns would result in a polynomial of the form

$$Q' = \prod_{i=1}^{n-t}(\mathrm{rowSum}_i(\mathbf{y}') + \alpha_i) \cdot \prod_{j=1}^{n-t}(\mathrm{colSum}_j(\mathbf{y}') + \beta_j)$$

for some field elements $\{\alpha_i, \beta_j : i, j \in [n-t]\}$. By further setting $y_{i,n-t} = (-\alpha_i)$ and $y_{n-t,j} = (-\beta_j)$ for $i, j \in [n-t-1]$, and fixing $y_{n-t,n-t}$ to a value that preserves non-zeroness, we obtain the projection up to a non-zero constant factor $\gamma \in \mathbb{F}$.

$$Q'' = \gamma \cdot \prod_{i=1}^{n-t-1} \mathrm{rowSum}_i(\mathbf{y}) \cdot \prod_{j=1}^{n-t-1} \mathrm{colSum}_j(\mathbf{y})$$

where $\mathbf{y} = \mathbf{y}' \setminus \{y_{ij} : i = n-t \text{ or } j = n-t\}$, which equals $P_{n-t-1}(\mathbf{y})$. □

With the above two lemmas, Theorem 3.2 is straightforward.

Proof of Theorem 3.2. The proof is a simple induction on k. The base case, $k = 1$, follows immediately from Lemma 3.6 and the evaluation dimension characterization of ROABPs, Theorem 2.3.

For $k > 1$, we show that if P_n is computable by a sum of k ROABPs of width at most w, then

$$n \quad \leq \quad \log^2(w+1) + \log^2\left((w+1)^2\right) + \cdots \log^2\left((w+1)^{2^{k-1}}\right) + k.$$

Let us assume the hypothesis is true for $k-1$ and we now prove it for k. Suppose P_n is computable by a sum of k ROABPs of width w. Assume that $t = \log^2(w+1) < n$, for otherwise the lower bound follows immediately. By Lemma 3.3, there is some set $\mathbf{s}$ of t variables such that for any $r = (w+1)$ partial evaluations $\mathbf{a}_1, \ldots, \mathbf{a}_r$ on $\mathbf{s}$, some linear combination is computable by a sum of $k-1$ ROABPs of width $w(w+1)$.

On the other hand, Lemma 3.6 states that we can find $r \geq 2^{\sqrt{t}} = w+1$ partial evaluations on $\mathbf{s}$ that are linearly independent and any linear combination of them can be written has P_{n-t-1} as a projection.

Therefore, if P_n is computable by a sum of k ROABPs of width w, then P_{n-t-1} is computable by a sum of $(k-1)$ ROABPs of width at most $w(w+1) < (w+1)^2$, by taking the sum of $(k-1)$ ROABPs computing $\sum_{i=1}^{w+1} \alpha_i \cdot P_n|_{\mathbf{s}=\mathbf{a}_i}$ and projecting further to obtain P_{n-t-1}. But the inductive hypothesis then forces

$$n - \log^2(w+1) - 1 \leq \left(\log^2(w+1)^2\right) + \log^2\left(\left((w+1)^2\right)^2\right) + \cdots \log^2\left(\left((w+1)^{2^{k-2}}\right)^2\right) + (k-1)$$
$$\implies n \leq \log^2(w+1) + \log^2(w+1)^2 + \log^2(w+1)^{2^2} + \cdots \log^2(w+1)^{2^{k-1}} + k$$
$$\leq 4^k \cdot \log^2(w+1) + k.$$

Thus, $w \geq \exp(\Omega(\sqrt{n}/2^k))$ as claimed. □

4 Lower bounds for read-k oblivious ABPs

In this section we show an explicit polynomial that has a polynomial-size depth-3 multilinear circuit and yet cannot be computed efficiently by a read-k oblivious ABP. To accomplish this we first demonstrate a polynomial that as large evaluation dimension even when a small fraction of the variables are fixed. We then argue that every polynomial computed by a read-k oblivious ABP has a small fraction of variables that when fixed result in polynomial with small evaluation dimension. Combined these two fact provided an exponential lower bound for read-k oblivious ABPs, when k is bounded.

4.1 An explicit polynomial with large evaluation dimension

Raz and Yehudayoff [RY09] constructed an explicit multilinear polynomial $f(\mathbf{x})$ with evaluation dimension as high as possible with respect to any partition $\mathbf{y}, \overline{\mathbf{y}}$ of the variables $\mathbf{x}$. Our requirements are slightly different, as we need a "robustness" property, namely, we argue that the evaluation dimension of the polynomial remains high even when we fix a small constant fraction, say, $1/10$, of the variables. Our construction is inspired by a recent similar construction of Kayal, Nair and Saha [KNS16].

Consider the complete bipartite graph $K_{n,n}$ with n vertices on each side. We shall label the left vertices as $x_1, \ldots, x_n$ and the right vertices as $y_1, \cdots, y_n$. We can write $K_{n,n}$ as a union of n edge-disjoint perfect matchings $M_1 \cup \cdots \cup M_n$, where for every $i \in [n]$, M_i contains all edges of the form $(x_j, y_{(j+i \bmod n)+1})$ for $j \in [n]$. Define the polynomial Q_n as

$$Q_n(x_1, \ldots, x_n, y_1, \ldots, y_n, z_1, \ldots, z_n) = \sum_{i=1}^{n} z_i \prod_{(j,k) \in M_i} (x_j + y_k) = \sum_{i=1}^{n} z_i \prod_{j=1}^{n} (x_j + y_{(j+i \bmod n)+1}). \quad (4.1)$$

By its definition, it is clear that Q_n is computed by a depth-3 polynomial-size circuit. We now show that even if we fix a small fraction of the variables in $\mathbf{x} \cup \mathbf{y}$, Q_n retains a large evaluation dimension with respect to any partition of the variables we have not fixed.

Lemma 4.2. *Let* $\mathbf{s}$, $\mathbf{t}$ *be two disjoint subsets of* $\mathbf{x} \cup \mathbf{y}$ *such that* $|\mathbf{s} \sqcup \mathbf{t}| \geq 0.9 \cdot 2n$. *Let* $\mathbf{r} = \mathbf{x} \cup \mathbf{y} \setminus (\mathbf{s} \cup \mathbf{t})$. *Then,*

$$\mathrm{evalDim}_{\mathbf{s},\mathbf{t};\mathbf{r}}(Q_n) \quad \geq \quad \exp(\Omega(\min(|\mathbf{s}|, |\mathbf{t}|))).$$

Proof. Assume without loss of generality that $|\mathbf{s}| \leq |\mathbf{t}|$, and that $\mathbf{s}_L := \mathbf{s} \cap \mathbf{x}$ satisfies $|\mathbf{s}_L| \geq |\mathbf{s}|/2$. Since $(\mathbf{s} \cup \mathbf{t}) \cap \mathbf{y} \geq 0.8n$, $|\mathbf{t}_R| = |\mathbf{t} \cap \mathbf{y}| \geq (0.8n - |\mathbf{s}|/2) \geq 0.3n$. Thus, there are at least $|\mathbf{s}_L| \cdot |\mathbf{t}_R| = \Omega(n \cdot |\mathbf{s}|)$ edges between $\mathbf{s}$ and $\mathbf{t}$ in $K_{n,n}$. By averaging, some matching M_i must include at least $t = \Omega(|\mathbf{s}|)$ of these edges. Consider the polynomial $f_i = \prod_{(j,k) \in M_i} (x_j + y_k)$. For every edge $(j,k) \in M_i$ that does not go between $\mathbf{s}$ and $\mathbf{t}$, if either of the variables belong to $\mathbf{s}$ we can further restrict them to 0/1 values as to preserve the non-zeroness of the polynomial $(x_j + y_k)$. Let the resulting polynomial be h_i, and note that it cannot have greater evaluation dimension than f_i. Hence, we can write

$$h_i = \prod_{m=1}^{\ell} (s_m + t_m) \cdot g(\mathbf{w}),$$

where for every $m \in [\ell]$ we have that $s_m \in \mathbf{s}$, $t_m \in \mathbf{t}$, and $g(\mathbf{w})$ is a polynomial that does not contain any variables of $\mathbf{s}$. Here we have "pushed" into g all the factors of f_i that correspond to edges in the matching M_i that do not go between $\mathbf{s}$ and $\mathbf{t}$. This means that $\mathbf{w}$ is the union of $\mathbf{r}$ and the variables in $\mathbf{t}$ that are not matched with variables in $\mathbf{s}$ by M_i.

By Lemma 2.4, $\mathrm{evalDim}_{\mathbf{s},\mathbf{t};\mathbf{r}}(f_i) \geq \mathrm{evalDim}_{\mathbf{u},\mathbf{v}\sqcup\mathbf{w}}(h_i) \geq 2^{\ell} = 2^{\Omega(|\mathbf{s}|)}$. Since f_i is a projection of Q_n, achieved by setting $z_i = 1$ and $z_j = 0$ for all $j \neq i$, it follows that $\mathrm{evalDim}_{\mathbf{s},\mathbf{t};\mathbf{r}}(Q_n) \geq \mathrm{evalDim}_{\mathbf{s},\mathbf{t};\mathbf{r}}(f_i) \geq \exp(\Omega(|\mathbf{s}|))$. As we assumed $|\mathbf{s}| \leq |\mathbf{t}|$, the lemma follows.

□

We note that the difference between the above polynomial Q_n and the one constructed by Kayal, Nair and Saha ([KNS16]) is that they use a 3-regular bipartite expander instead of $K_{n,n}$. This is important in their setting because the degree of the graph corresponds to the number of matchings, and this is the top fan-in of the depth-3 circuit computing the polynomial. In fact, in order to show hardness for read-k oblivious ABPs it is also possible to use a good enough bipartite expander (with a constant degree d that depends only on k) whose expansion property guarantees that a proof strategy along the lines of Lemma 4.2 and Theorem 4.4 would work. This would have allowed us to present a hard polynomial which is computed by a depth-3 circuit with bounded top fan-in, however, this very small gain would have come at the cost of increasing the complexity of the construction and making it depend on k.

4.2 Upper bound on evaluation dimension for read-k oblivious ABPs

In this section we show that if f is computed by a read-k oblivious ABP of width w, then we can fix a "small" subset of variables such that the remaining variables can be partitioned into two carefully chosen "large" subsets, under which the evaluation dimension is at most w^{2k}. We then apply this result to the polynomial Q_n (from (4.1)) and use Lemma 4.2 to show that if Q_n is computed by a width-w read-k oblivious ABP, then $w \geq \exp(n/k^{O(k)})$.

Theorem 4.3. *Let $f \in \mathbb{F}[\mathbf{x}]$ be a polynomial computed by a width-w read-k oblivious ABP. Then, there exist a partition of $\mathbf{x} = \mathbf{u} \sqcup \mathbf{v} \sqcup \mathbf{z}$, such that*

1. $|\mathbf{u}|, |\mathbf{v}| \geq |\mathbf{x}|/k^{O(k)}$,
2. $|\mathbf{z}| \leq |\mathbf{x}|/10$, *and*
3. $\mathrm{evalDim}_{\mathbf{u},\mathbf{v};\mathbf{z}}(f) \leq w^{2k}$.

Proof. Consider an ABP A that computes f. Divide the $k|\mathbf{x}|$ layers into r equal-sized contiguous blocks of $k|\mathbf{x}|/r$ layers (where r shall be set shortly). For each variable, consider the (at most) k blocks that its k reads fall in (if the number of such blocks is strictly smaller than k, we can fill up to k blocks arbitrarily). By a simple averaging, there must exist k blocks $B_1, \ldots, B_k$ that contain all k reads of a set $\mathbf{u}$ of at least $|\mathbf{x}|/\binom{r}{k}$ variables. Let $\mathbf{z}$ be the set of variables in $B_1 \cup B_2 \cup \cdots \cup B_k$ that are not in $\mathbf{u}$, and $\mathbf{v}$ be the set of all remaining variables. As each block is of size $k|\mathbf{x}|/r$, we have that $|\mathbf{z}| \leq k^2|\mathbf{x}|/r$, which is at most $|\mathbf{x}|/10$ if we set $r = 10k^2$. Observe that $|\mathbf{v}| \geq |\mathbf{x}| - k^2|\mathbf{x}|/r \geq 9n/10$ and $|\mathbf{u}| \geq n/\binom{10k^2}{k} \geq |\mathbf{x}|/k^{O(k)}$. Let us ignore the variables in $\mathbf{z}$ by considering the ABP over the field $\mathbb{F}(\mathbf{z})$.

We now claim that $\mathrm{evalDim}_{\mathbf{u},\mathbf{v};\mathbf{z}}(f) \leq w^{2k}$. Having moved the variables in $\mathbf{z}$ to the field, each of the r blocks is either entirely contained in $\mathbf{u}$ or entirely contained in $\mathbf{v}$. Therefore, since the reads comprise of at most k alternating blocks of variables in $\mathbf{u}$ and $\mathbf{v}$, the resulting branching program has the k-gap property with respect to $\mathbf{u}$. It follows immediately from Lemma 2.9 that $\mathrm{evalDim}_{\mathbf{u},\mathbf{v};\mathbf{z}}(f)$ is at most w^{2k}. □

We now show that Q_n (defined in (4.1)) is hard for read-k oblivious ABPs to compute.

Theorem 4.4 (Restatement of Theorem 1.5)**.** *Let A be a width-w, read-k oblivious ABP computing the polynomial Q_n (defined in* (4.1)*). Then $w \geq \exp(n/k^{O(k)})$.*

Proof. First observe that we can eliminate the $\mathbf{z}$ variables by considering the ABP over the field $\mathbb{F}(\mathbf{z})$ so that it now computes a polynomial in the variables $\mathbf{x} \cup \mathbf{y}$. By Theorem 4.3, there exists a partition $\mathbf{u} \sqcup \mathbf{v} \sqcup \mathbf{z}$ of $\mathbf{x} \cup \mathbf{y}$ with the prescribed sizes as in the statement of the theorem, such that $\mathrm{evalDim}_{\mathbf{u},\mathbf{v};\mathbf{z}}(Q_n) \leq w^{2k}$. Since $|\mathbf{z}| \leq 2n/10$, Lemma 4.2 implies that $\mathrm{evalDim}_{\mathbf{u},\mathbf{v};\mathbf{z}}(f) = \exp(\Omega(\min(|\mathbf{u}|, |\mathbf{v}|)))$. Using the fact that $\min(|\mathbf{u}|, |\mathbf{v}|) \geq n/k^{O(k)}$, we get that $w^{2k} \geq \exp(n/k^{O(k)})$, which implies $w \geq \exp(n/k^{O(k)})$ as well. $\square$

5 Identity tests for read-k oblivious ABPs

In this section we give PIT algorithms for the class of read-k oblivious ABPs. Our algorithms are based on the following theorem that efficiently constructs small hitting sets for read-once ABPs.

Theorem 5.1 (Hitting Set for ROABPs, [AGKS15])**.** *There exists a hitting set $\mathcal{H}$ for the class of n-variate polynomials computed by width-w individual-degree-d ROABPs of size $(nwd)^{O(\log n)}$, in any variable order. $\mathcal{H}$ can be constructed in time $\mathrm{poly}(|\mathcal{H}|)$.*

5.1 Identity tests for k-pass ABPs

First, observe that Lemma 2.5 immediately implies a black-box algorithm for the subclass of k-pass ABPs, as they can be simulated efficiently by a ROABP and then tested using Theorem 5.1.

Corollary 5.2. *There is a hitting set of size $(ndw)^{O(k \log n)}$ for the class of n-variate k-pass ABPs of width w and degree d.*

Proof. Follows directly from Lemma 2.5 and the $(ndw')^{O(\log n)}$-sized hitting set for width w' read-once ABPs from Theorem 5.1. $\square$

We now turn to general read-k oblivious ABPs.

5.2 From read-k to per-read-monotone and regularly-interleaving sequences

In this section we show that given any read-k oblivious ABP over $\mathbf{x} = \{x_1, \ldots, x_n\}$ computing a polynomial f, we can find a "large" subset of variables $\mathbf{y} \subseteq \mathbf{x}$ such that f has a "small" ROABP when we think of f as a polynomial in the $\mathbf{y}$ variables over the field $\mathbb{F}(\overline{\mathbf{y}})$. This process, in fact, involves only finding the correct subset $\mathbf{y}$ (without rewiring any part of the ABP). Therefore, in order to avoid technical overhead it is useful to think in terms of sequences over abstract sets of elements, which correspond to the order in which the ABP reads the variables, and not in terms of variables in branching programs.

Let X be a set, and let $n = |X|$. Let $S \in X^m$ be a sequence of m elements from X. We say S is *read-k* if each element $x \in X$ occurs k times in S (in this case we also have $m = nk$). As mentioned in Remark 1.3, we restrict ourselves to considering sequences that are read-k for some k. For $i \in [k]$, we denote by $S^{(i)}$ the subsequence of S which consists of the i-th occurrences of elements in X. That is, $S^{(i)}$ is a permutation of the elements of X, according to the order in which they appear in S

for the i-th time. Similarly, for $i \neq j \in [k]$, we use the notation $S^{(i,j)}$ for the subsequence of S which consists of the i-th and j-th occurrences of elements in X. We use $\mathrm{rev}(S)$ to denote the sequence S in reverse order.

For $x \in X$ and $i \in [k]$ let $\mathrm{Occur}^i_S(x)$ denote the index of the i-th occurrence of x in S. For an index $\ell \in [kn]$ let $\mathrm{Var}_S(\ell)$ denote the pair (x_i, c) such that the c-th occurrence of x_i appears at index ℓ in S. For a subset $X' \subseteq X$, let $S|_{X'}$ denote the restriction of S to the set X' that is the result of dropping all elements of $X \setminus X'$ from S. Thus, $S|_{X'} \in (X')^{m'}$ for $m' = |X'|k$.

Next we define a special subclass of read-k sequences where the elements are read in either increasing or decreasing order relative to the subsequence of first occurrences.

Definition 5.3. *Let $S \in X^{nk}$ be a read-k sequence. We say S is* per-read-monotone *if for every $i \in [k]$, $S^{(i)}$ is either monotonically increasing, i.e., $S^{(i)} = S^{(1)}$, or monotonically decreasing, i.e., $S^{(i)} = \mathrm{rev}(S^{(1)})$.* $\diamondsuit$

Observe that read-once sequences are naturally per-read-monotone and monotonically increasing. It is often convenient to assume that $S^{(1)} = (x_1, \dots, x_n)$, that is, that the elements of X are labeled according to the order of their first occurrences. This can be ensured without loss of generality by renaming elements.

The following well-known theorem asserts that any long enough sequence contains a large monotone subsequence:

Theorem 5.4 (Erdős–Szekeres Theorem, [ES35, AZ04])**.** *Let m be a positive integer. Let S be a sequence of distinct integers of length at least $m^2 + 1$. Then, there exists a monotonically increasing subsequence of S of length $m + 1$,* or *a monotonically decreasing subsequence of S of length $m + 1$.*

As an immediate corollary of Theorem 5.4, we get the following lemma:

Lemma 5.5. *Let S be a read-2 sequence over $X = \{x_1, \dots, x_n\}$. Then, there exists a subset $X' \subseteq X$ with $|X'| \geq \sqrt{n}$ such that the subsequence $S' = S|_{X'}$ is per-read-monotone. The subsequence S' can be constructed in time* $\mathrm{poly}(n, k)$.

Proof. We want to show that there is a long subsequence in $S^{(2)}$ that is monotone with respect to the ordering of $S^{(1)}$. By Theorem 5.4, there exists such a monotonic subsequence of $S^{(2)}$ of length $m + 1$ where $m = \lfloor \sqrt{n - 1} \rfloor$. It follows that this subsequence is of length at least $\sqrt{n}$. Let $X' \subseteq X$ be the set of elements that appear in this monotonic subsequence, and let $S' = S|_{X'}$. Then by the choice of X', it follows that S' is per-read-monotone. The subsequence S' can be constructed in polynomial time using standard dynamic programming techniques. $\square$

We can generalize Lemma 5.5 to read-k sequences, at the cost of settling for a weaker lower bound of $n^{1/2^{k-1}}$ on the length of the subsequence:

Lemma 5.6. *Let S be a read-k sequence over $X = \{x_1, \dots, x_n\}$. Then, there exists a subset $X' \subseteq X$ with $|X'| \geq n^{1/2^{k-1}}$ such that the subsequence $S' = S|_{X'}$ is per-read-monotone. The subsequence S' can be constructed in time* $\mathrm{poly}(n, k)$.

Proof. As in the proof of Lemma 5.5, the set X' can be constructed by repeatedly pruning X using $k - 1$ repeated applications of Theorem 5.4.

That is, we first apply Theorem 5.4 on the subsequence $S^{(2)}$ of second occurrences and obtain a monotonic subsequence of length $n_2 := \lceil \sqrt{n} \rceil$. We discard all elements of X which do not appear in this monotonic subsequence. We move on to the subsequence $S^{(3)}$ of third occurrences, and find a monotonic subsequence of length $n_3 := \lceil \sqrt{n_2} \rceil$, again discarding all elements that do not appear in this subsequence. After finding the k-th monotonic subsequence, we are left with a subset X' of size at least $n^{1/2^{k-1}}$ that satisfies the conditions of the lemma. $\square$

We now show how to prune per-read-monotone read-2 sequences even further, trading a constant fraction of their size for stronger structural properties. We begin by stating the property we look for.

Definition 5.7. *Let S be a read-2 sequence over a set of elements X. We say S is* 2-regularly-interleaving *if there exists a partition of X to* blocks $\{X_i\}_{i\in[t]}$ *such that for every $i \in [t]$:*

- *For every $c \in \{1, 2\}$, all the c-th occurrences of the block X_i appear consecutively in S.*
- *The interval containing the second occurrences of the block X_i immediately follows the interval containing the first occurrences of X_i.*

A read-k sequence S is said to be k-regularly-interleaving *if for any $i \neq j \in [k]$, the subsequence $S^{(i,j)}$ is 2-regularly-interleaving. That is, S is k-regularly-interleaving if restricted to any two reads it is 2-regularly-interleaving.* $\Diamond$

To get an intuitive sense of this definition, the reader may consult Figure 3.

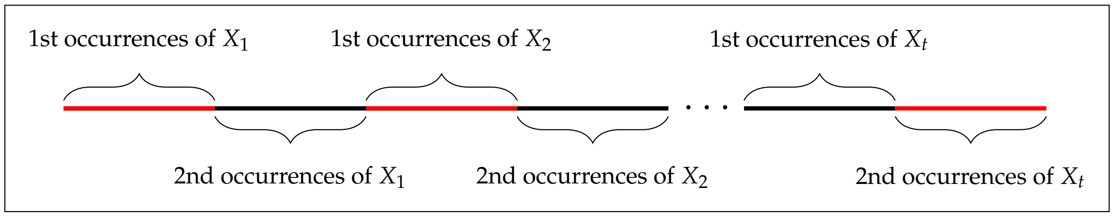

Figure 3: A 2-regularly-interleaving sequence.

The following lemma is used to simplify some of the later arguments. It shows that in a read-k per-read-monotone sequence, the monotonically increasing subsequences cannot intersect with monotonically decreasing subsequences.

Lemma 5.8. *Let S be a read-k, per-read-monotone sequence over $X = \{x_1, \ldots, x_n\}$. We can write S as a concatenation $S = (T_1, T_2, \ldots, T_t)$, such that:*

1. *for every $j \in [t]$, T_j is a read-k_j sequence for $k_j \leq k$.*
2. *for every $i \in [k]$ there exists $j \in [t]$ so that $S^{(i)}$ is contained in T_j.*
3. *for every odd $j \in [t]$, all the subsequences $S^{(i)}$ that appear in T_j are monotonically increasing, and for any even j, all are monotonically decreasing.*
4. *for every $j \in [t-1]$, the last element that appears in T_j equals the first element appearing in T_{j+1}, and this element can be either the last element of $S^{(1)}$, if T_j contains monotonically increasing subsequences and T_{j+1} contains monotonically decreasing subsequences, or the first element of $S^{(1)}$, in the opposite case.*

In other words, we can partition S into t disjoint contiguous subsequences, such that every $S^{(i)}$ is completely contained in exactly one subsequence, and in every subsequence, either all reads are increasing or all reads are decreasing, with the pattern alternating.

Proof. The proof is by induction on k. For $k = 1$, $S = S^{(1)}$ and this is a trivial statement. Assume without loss of generality that $S^{(1)} = (x_1, x_2, \dots, x_n)$.

For larger values of k, we first show that no decreasing sequence can intersect an increasing one. Consider any $i, j \in [k]$ with $i < j$. Suppose $S^{(i)}$ is monotonically increasing and $S^{(j)}$ is monotonically decreasing. Since S is per-read-monotone, $S^{(i)} = S^{(1)}$ and $S^{(j)} = \mathrm{rev}(S^{(1)})$. This implies that

$$\mathrm{Occur}^i_S(x_1) < \cdots < \mathrm{Occur}^i_S(x_n) < \mathrm{Occur}^j_S(x_n) < \cdots < \mathrm{Occur}^j_S(x_1),$$

because $i < j$ and so the i^{th} occurrence of x_n must come before its j^{th} occurrence. We conclude that $S^{(i)}$ and $S^{(j)}$ cannot intersect. The other case, where $S^{(i)}$ is decreasing and $S^{(j)}$ is increasing, is handled analogously.

Let ℓ denote the first index in which a decreasing subsequence $S^{(i)}$ begins (if no such ℓ exists, the lemma is clearly satisfied by picking $T_1 = S$). By the above argument, all the elements before the ℓ-th index must belong to increasing subsequences which are read entirely. We can define T_1 to be the subsequence of S from index 1 up to index $\ell - 1$, and continue inductively on the subsequence S' of S from index ℓ to the end, which has $k' < k$ reads, to construct the remainder of the T_j's in accordance with the statement of the lemma.

As for item 4, it follows from the fact that a sequence of monotonically increasing subsequences must end in x_n (as to maintain monotonicity), and a sequence of monotonically decreasing subsequences must begin with x_n for the same reason. The opposite case is handled analogously. □

The following lemma shows that given a 2-read per-read-monotone sequence, we can find a large subsequence which is also 2-regularly interleaving.

Lemma 5.9. *Let S be a read-2 per-read-monotone sequence over X. Then there is a subset $X' \subseteq X$ with $|X'| \geq n/3$ such that the sequence $S' = S|_{X'}$ is per-read-monotone and 2-regularly-interleaving.*

Proof. We show how to erase the occurrences of (not too many) elements from S, such that the remaining sequence is 2-regularly-interleaving and maintains its per-read-monotonicity property.

First observe that if the subsequence $S^{(2)}$ of second occurrences is monotonically *de*creasing, then, by Lemma 5.8, S is already also 2-regularly-interleaving. In this case we have $S = (S^{(1)}, S^{(2)}) = (S^{(1)}, \mathrm{rev}(S^{(1)}))$ and we can pick just one block, X, and satisfy the definition.

From now on we assume then that $S^{(2)}$ is monotonically *in*creasing. For every $z \in X$, denote by $d_z = \mathrm{Occur}^2_S(z) - \mathrm{Occur}^1_S(z)$ the distance between the first and the second occurrence of z in S. Pick $x \in X$ such that d_x is maximal and let $r := d_x$. By averaging, among the r occurrences between $\mathrm{Occur}^1_S(x)$ and $\mathrm{Occur}^2_S(x)$, there exist either $r/2$ first occurrences or $r/2$ second occurrences. Assume there are at least $r/2$ first occurrences in this interval (the other case is handled in an analogous way), and let A be the set of variables (including x) whose first occurrence appears between the $\mathrm{Occur}^1_S(x)$ and $\mathrm{Occur}^2_S(x)$, so that $|A| \geq r/2$.

Since $S^{(2)}$ is monotonically increasing, for every $z \in A$ it holds that $\mathrm{Occur}^2_S(z) > \mathrm{Occur}^2_S(x)$. Let $y \in A$ be the element such that $\mathrm{Occur}^2_S(y)$ is maximal. Observe that

$$\mathrm{Occur}^2_S(y) - \mathrm{Occur}^2_S(x) \leq \mathrm{Occur}^2_S(y) - \mathrm{Occur}^1_S(y) \leq r,$$

where the first inequality follows from the fact that S is per-read-monotone and the second follows from the choice of x. Hence, it follows that

$$\text{Occur}_S^2(y) - \text{Occur}_S^1(x) \le 2r. \tag{5.10}$$

We now erase from S all the elements that appear in the interval $[\text{Occur}_S^1(x), \text{Occur}_S^2(y)]$ but do not appear in A. Having done that, the subsequence in this interval satisfies the requirements of the lemma (one can relabel the elements if necessary in order to ensure contiguous indexing in this subsequence). See Figure 4 for a diagram of this. Furthermore, we have kept at least $|A| \ge r/2$ elements alive. By (5.10), we have erased at most r elements.

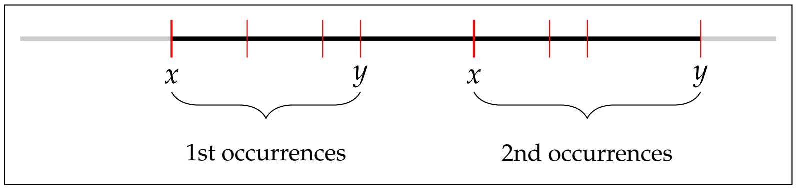

Figure 4: The construction of A. Elements of A are marked by a vertical hash. All other elements in the black interval are discarded. The process then continues inductively on the gray subsequences.

We continue this process recursively on the subsequences in both of the intervals $[1, \text{Occur}_S^1(x) - 1]$ and $[\text{Occur}_S^2(y) + 1, 2|X|]$. Observe that these intervals cannot share any element, as that would mean that the two occurrences of this element are of distance more than r apart, which contradicts the choice of x. Hence, we may continue independently on both subintervals. Let $X' \subseteq X$ be the set of elements that stay alive throughout this process (this is exactly the union of the sets A). By induction, $|X'| \ge |X|/3$, because for each element that is kept alive in A we erase at most two elements from the sequence. The statement of the lemma follows. $\square$

Viewed as an algorithm, the proof of Lemma 5.9 is a procedure that, given a per-read-monotone sequence S over X, decides which elements of X should be erased in order to be left with a 2-regular-interleaving sequence $S' = S|_{X'}$. It can also be noted that both properties of being per-read monotone and being 2-regularly interleaving are downward-closed, in the sense that for any subset $X'' \subseteq X'$, the subsequence $S'' = S'|_{X''}$ maintains both properties. Hence, if we are given a read-k per-read-monotone sequence S, by repeatedly applying the algorithmic of Lemma 5.9 separately on each subsequence $S^{(i,j)}$ for $i \ne j \in [k]$ (maintaining a constant fraction of the elements on each application), we get the following corollary:

Corollary 5.11. *Let S be a read-k per-read-monotone sequence over $X = \{x_1, \dots, x_n\}$. Then there is a subset $X' \subseteq X$ with $|X'| \ge n/3^{k^2}$ such that the sequence $S' = S|_{X'}$ is per-read-monotone and k-regularly-interleaving.* $\square$

5.3 ROABPs for regularly interleaving sequences

In this section we show that if a polynomial f is computed by a small-width read-k oblivious ABP A such that the sequence S of the reads in A is per-read-monotone and k-regularly-interleaving, then f can in fact also be computed by a small-width ROABP A' (in the same order as $S^{(1)}$). We show this by proving that A has the k-gap property with respect to that order, and then applying Lemma 2.9.

Lemma 5.12. *Let $f \in \mathbb{F}[x_1, \ldots, x_n]$ be computed by a read-k oblivious ABP A of width w, and let S be the sequence of variables read by A. Suppose further that S is per-read-monotone with respect to the order $x_1 < x_2 < \cdots < x_n$ and k-regularly-interleaving. Then A has the k-gap property.*

In the proof of Lemma 5.12 we use the following lemma in order to bound the number of "gaps" one obtains for any prefix.

Lemma 5.13. *Let S be a read-k, per-read-monotone, k-regularly-interleaving sequence over $X = \{x_1, \ldots, x_n\}$. Let $\ell \in [kn]$ be an integer and suppose that $\mathrm{Var}_S(\ell) = (x_i, c)$ and $\mathrm{Var}_S(\ell + 1) = (x_j, d)$.*[3]

- *Suppose that for every $m \in [k]$, $S^{(m)} = (x_1, x_2, \ldots, x_n)$ and $j > i$, then $j = i + 1$.*
- *Suppose that for every $m \in [k]$, $S^{(m)} = (x_n, x_{n-1}, \ldots, x_1)$ and $i > j$, then $i = j + 1$.*

Proof. We argue only the first conclusion, as the second follows by a symmetric argument.

Observe that if $c = d$ the claim is true by monotonicity of $S^{(c)}$. If $c \neq d$, consider the subsequence $S^{(c,d)}$. Since S is k-regularly interleaving, $S^{(c,d)}$ is 2-regularly interleaving, and in this sequence it also holds that x_j immediately follows x_i. Furthermore, we have that $d < c$, as otherwise $\mathrm{Occur}_S^c(x_j) < \mathrm{Occur}_S^d(x_j) = \mathrm{Occur}_S^c(x_i) + 1$ implies that $\mathrm{Occur}_S^c(x_j) < \mathrm{Occur}_S^c(x_i)$, which contradicts the monotonicity of $S^{(c)}$, as we assumed that $j > i$. Hence, in $S^{(c,d)}$ d plays the role of the *first* read, and c plays the role of the *second* read.

Since $S^{(c,d)}$ is a 2-regularly-interleaving sequence and per-read monotonically increasing, the blocks in Definition 5.7 at not just sets but are contiguous sequences of variables

$$X_1 = (x_1, x_2, \ldots, x_{i_1}), X_2 = (x_{i_1+1}, x_{i_1+2}, \ldots, x_{i_2}), \ldots, X_t = (x_{i_{t-1}+1}, \ldots, x_n).$$

such that

$$S^{(c,d)} = (X_1, X_1, X_2, X_2, \ldots, X_t, X_t).$$

Suppose that x_i belongs to X_{b_i} and x_j belongs to X_{b_j}. Recall that the lemma assumes that the first read of x_j immediately follows the second read of x_i in $S^{(c,d)}$. Hence, x_i is at the end of the second read of the block X_{b_i} and x_j is at the beginning of the first read of the block X_{b_j}. This implies that $b_j = b_i + 1$ and hence that $j = i + 1$. □

We are now ready to prove Lemma 5.12.

Proof of Lemma 5.12. Let S be the sequence of reads in A. Let $i \in [n]$ and $\mathbf{a} \in \mathbb{F}^i$. By applying Lemma 2.1 and fixing $\mathbf{x}_{[i]} = \mathbf{a}$, we can write

$$f|_{\mathbf{x}_{[i]}=\mathbf{a}} = \Big(N_1(\mathbf{a}) \cdot M_1(x_{i+1}, \ldots, x_n) \cdot N_2(\mathbf{a}) \cdot M_2(x_{i+1}, \ldots, x_n) \\ \cdots N_t(\mathbf{a}) \cdot M_t(x_{i+1}, \ldots, x_n)\Big)_{1,1}. \tag{5.14}$$

for some integer q, where for each $\sigma \in [q]$, N_σ is a product of univariate matrices of layers that read $\{x_1, \ldots, x_i\}$, and M_σ is a product of univariate matrices of layers that read $\{x_{i+1}, \ldots, x_n\}$. We wish to show that $q \leq k$ which implies A has the k-gap property.

For each M_σ we define its *interface* to be the pair (ℓ_σ, r_σ), which are, respectively, the indexes of the first and last layers of A that define M_σ. Thus $\ell_\sigma - 1$ is the last layer that this part of N_σ

$$f|_{x_1=a_1,x_2=a_2} = \big(N_1(a_1,a_2)M_1(x_3,x_4)N_2(a_1,a_2)M_2(x_3,x_4)\big)_{(1,1)}$$

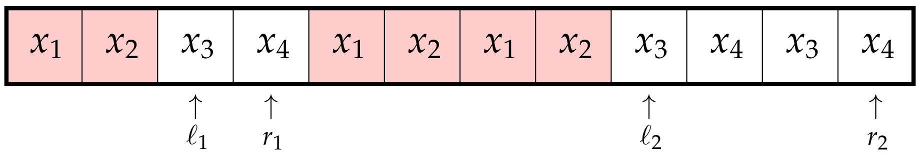
x_1 x_2 x_3 x_4 x_1 x_2 x_1 x_2 x_3 x_4 x_3 x_4
ℓ_1
r_1
ℓ_2
r_2

Figure 5: The ABP reads the variables in the order that appears in the box. For $i = 2$, the locations of both interfaces are marked.

and $r_\sigma + 1$ is the first layer that is part of $N_{\sigma+1}$ (when it exists). See Figure 5 for a diagram of this notation.

We apply Lemma 5.8 to rewrite S as $(T_1, T_2, \ldots, T_t)$. Consider any T_j for $j \in [t]$, and two case based on the third property of Lemma 5.8.

Consider the case when j is odd, then all occurrence subsequences in T_j are monotonically increasing. Consider the representation (5.14) and the left side of interfaces (ℓ_σ, r_σ) that lie in T_j. Let us denote $\text{Var}_S(\ell_\sigma - 1) = (x_{\sigma_1}, c)$ and $\text{Var}_S(\ell_\sigma) = (x_{\sigma_2}, d)$. Note that since $i \geq 1$ and T_j begins with x_1, by the fourth property of Lemma 5.8, it is the case that both ℓ_σ and $\ell_\sigma - 1$ lie in T_j. Furthermore, since we are considering the left side of interfaces, it must be the case that $\sigma_1 \leq i < \sigma_2$. By the first property of Lemma 5.13, we must have $\sigma_2 = \sigma_1 + 1$, and thus $\sigma_1 = i$ and $\sigma_2 = i + 1$. Hence, we can map each left side of an interface in T_j to a unique occurrence of x_i in T_j.

Now consider the case when j is even, then all occurrence subsequences in T_j are monotonically decreasing. By an argument symmetric to the one above, and using the second property of Lemma 5.13 instead of the first, we can map each right side of an interface in T_j to a unique occurrence of x_i in T_j.

Observe that by the third and fourth properties of Lemma 5.8 and above discussion of left and right side of interfaces we have: For every $\sigma \in [q]$, M_σ has either the layer preceding it or the layer succeeding it labeled by x_i. Since S is a read-k sequence and x_i occurs k times, $q \leq k$. We conclude that A has the k-gap property. $\square$

It now immediately follows that any read-k oblivious ABP the reads the variables in a per-read-monotone and k-regularly-interleaving fashion can be simulated by a small ROABP. We record this fact in the following corollary.

Corollary 5.15. *Let $f \in \mathbb{F}[x_1, \ldots, x_n]$ be computed by a read-k oblivious ABP A of width w, and let S be the sequence of variables read by A. Suppose further that S is per-read-monotone with respect to the order $x_1 < x_2 < \cdots < x_n$ and k-regularly-interleaving. Then for any $i \in [n]$, $\text{evalDim}_{\mathbf{x}_{[i]}, \overline{\mathbf{x}_{[i]}}}(f) \leq w^{2k}$. In particular, f is computed by a ROABP of width at most w^{2k} in the variable order $x_1, x_2, \ldots, x_n$.*

Proof. Immediate from Lemma 5.12 and Lemma 2.9. $\square$

5.4 Identity testing for read-k oblivious ABPs

In this section we give our white-box identity testing algorithm for read-k oblivious ABPs. Before giving the proof, let us first give an overview of the algorithm for the slightly simpler read-2 case.

[3]Recall that $\text{Var}_S(\ell) = (x_i, c)$ means that the ℓ-th element in S is x_i, and this is its c-th occurrence.

Given a read-2 oblivious ABP A with read sequence S which computes a polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$, Lemma 5.9 shows how to find a read-2 subsequence on a set $\mathbf{y} = \{y_1, \ldots, y_{\sqrt{n}}\}$ of roughly $\sqrt{n}$ variables. When we think of f as a polynomial in the $\mathbf{y}$ variables over the field $\mathbb{F}(\overline{\mathbf{y}})$, Corollary 5.15 allows us compute f with a small ROABP. We can then use a hitting set for ROABPs in order to find an assignment (from $\mathbb{F}$) to the $\mathbf{y}$ variables that keeps the polynomial f non-zero. Having done that, we are left with a non-zero polynomial over a smaller set of $n - \sqrt{n}$ variables, which is again computed by a read-2 oblivious ABP, so we may repeat this process. After at most $O(\sqrt{n})$ iterations we find an assignment for all the variables that keeps the polynomial non-zero. We note that a very similar "hybrid argument" that uses a hitting set for ROABPs appears both in [AGKS15] and [OSV15]. The argument for read-k is identical, apart from the loss in the parameters incurred by Corollary 5.11.

Our PIT algorithm is presented in Algorithm 1.

Algorithm 1 : PIT for read-k oblivious ABPs

Input: a read-k sequence S on n variables, degree d and width w as integers, and a field $\mathbb{F}$.
Output: a hitting set $\mathcal{H}$ for read-k ABPs of degree d and width w over $\mathbb{F}$ that read n variables in the order S.

```
1: $\mathbf{x} = \{x_1, \ldots, x_n\}$, $i = 1$
2: while $\mathbf{x} \neq \emptyset$ do
3:     Pick a subset $\mathbf{y}_i \subseteq \mathbf{x}$ of size at least $|\mathbf{x}|^{1/2^{k-1}}/3^{k^2}$, such that the subsequence that reads only
       the $\mathbf{y}_i$ variables is per-read-monotone and $k$-regularly-interleaving, using Lemma 5.6 (and
       Corollary 5.11).
4:     Construct a set $\mathcal{H}_i \subseteq \mathbb{F}^{|\mathbf{y}_i|}$ of size $(nw^{2k}d)^{O(\log n)}$ that hits ROABPs of width $w^{2k}$ in the $\mathbf{y}_i$
       variables, using Theorem 5.1.
5:     $\mathbf{x} \leftarrow \mathbf{x} \setminus \mathbf{y}_i$, $i \leftarrow i + 1$
6: end while
7: $t = i - 1$
8: return the set $\mathcal{H} = \mathcal{H}_1^{\mathbf{y}_1} \times \cdots \times \mathcal{H}_t^{\mathbf{y}_t}$.
```

We now argue that the set $\mathcal{H}$ produced by Algorithm 1 is a hitting set, and bound its size and construction time.

Theorem 5.16. *There is a white-box polynomial identity test for read-k oblivious ABPs of width w and degree d on n variables that runs in time* $\mathrm{poly}(n, w, d)^{n^{1-1/2^{k-1}} \exp(k^2) \operatorname{polylog}(n)}$. *Furthermore, given only the order in which the variables are read, we can construct a hitting set for such ABPs that read their variables in this order, of size* $\mathrm{poly}(n, w, d)^{n^{1-1/2^{k-1}} \exp(k^2) \operatorname{polylog}(n)}$.

Proof. Suppose $k = 1$, in this case we can immediately apply Theorem 5.1 to get an appropriate hitting set. Now suppose $k > 1$, and consider Algorithm 1. We first show that the set $\mathcal{H}$ it returns hits A, and then we bound the size of $\mathcal{H}$.

By Corollary 5.15, the polynomial f in the $\mathbf{y}_1$ variables is computed by a width-w^{2k} ROABP over the field $\mathbb{F}(\overline{\mathbf{y}_1})$. Hence, by Theorem 5.1, there exists $\mathbf{a}_1 \in \mathcal{H}_1$ such that $f(\mathbf{a}_1)$ is non-zero over $\mathbb{F}(\overline{\mathbf{y}_1})$. Similarly, we can now find $\mathbf{a}_2 \in \mathcal{H}_2$ and assign it to the $\mathbf{y}_2$ variables and keep the polynomial non-zero, etc. all the way up to $\mathbf{a}_t$. It follows, by induction, that $(\mathbf{a}_1, \ldots, \mathbf{a}_t)$ is an assignment from $\mathbb{F}$ to all the variables such that $f(\mathbf{a}_1, \ldots, \mathbf{a}_t)$ is non-zero, as required.

Furthermore, for all $i \in [t]$, $|\mathcal{H}_i| = (nw^{2k}d)^{O(\log n)}$, so, $|\mathcal{H}| = ((nw^{2k}d)^{O(\log n)})^t$. We now bound the number of iterations t. Let $T(n)$ denote the number of iterations needed for n variables. We show, by induction on n, that

$$T(n) \leq cn^{1-1/2^{k-1}},$$

for some $c = c(k)$ which will be set in a moment. This will imply the desired bound on $\mathcal{H}$.

Set $p = 1/2^{k-1}$. After the first iteration, the number of variables we are left with is $n' := n - n^p/3^{k^2}$ variables. By the induction hypothesis, we may assume that $T(n') \leq c \cdot (n')^{1-p}$. Hence

$$T(n) \leq 1 + T(n') \leq 1 + c(n')^{1-p} = 1 + c \cdot \left(n - \frac{n^p}{3^{k^2}}\right)^{1-p}$$

and we wish to show that

$$1 + c\left(n - \frac{n^p}{3^{k^2}}\right)^{1-p} \leq cn^{1-p}.$$

This is equivalent to

$$\frac{1}{c} \leq n^{1-p} - \left(n - \frac{n^p}{3^{k^2}}\right)^{1-p},$$

This is satisfied as long as

$$\frac{1-p}{3^{k^2}} \geq \frac{1}{c}$$

(see Lemma A.1 in the appendix for a proof of this fact), which we can ensure by picking $c = 2 \cdot 3^{k^2}$, as $k > 1$ implies that $p \leq 1/2$.

Finally, since finding the set $\mathbf{y}_i$ on each iteration can be done in polynomial time, the running time of the algorithm is dominated by the time required to construct $\mathcal{H}$, which is $\text{poly}(|\mathcal{H}|)$. □

6 Conclusions and Open Problems

In this work, we have obtained the first non-trivial lower bounds and identity testing algorithms for read-k oblivious ABPs. We briefly mention some directions that we find worth pursuing in future research.

The most natural open problem we pose is designing an identity testing algorithm for read-k oblivious ABPs with better running time than the algorithm we presented in this paper. Since for ROABPs (the $k = 1$ case) there exist a white-box polynomial time and black-box quasipolynomial-time algorithms, it seems reasonable to hope that the deterioration in the parameters would not be as sharp when $k > 1$ (the flip side of this argument, however, is the relative lack of progress in the analogous question in the boolean domain).

Another open problem is obtaining a completely black-box test for read-k oblivious ABPs in any variable order, that is, without knowing the order in which the variable appear. As we mentioned, for ROABPs there exist a black-box hitting set that works for any variable order [AGKS15], whose size is essentially the same as that of the hitting set that was obtained earlier for the known-order case [FS13b]. In our construction, we need to know the order so that we can pick the per-read-monotone and k-regularly-interleaving sequences to which we assign the hitting sets for ROABPs, and simply "guessing" those sets would require exponential time. Still, given the progress in obtaining hitting sets in any order for ROABPs, it might be the case that such a construction could follow from our strategy, even using known techniques.

Finally, we turn back to boolean complexity, and ask whether our ideas and techniques can be adapted to attack the problem of constructing pseudorandom generators for read-k oblivious *boolean* branching program with sublinear seed length.

References

[AFK85] Noga Alon, Zoltán Füredi, and Meir Katchalski. Separating pairs of points by standard boxes. *European Journal of Combinatorics*, 6(3):205–210, 1985.

[AFS$^+$16] Matthew Anderson, Michael A. Forbes, Ramprasad Saptharishi, Amir Shpilka, and Ben Lee Volk. Identity Testing and Lower Bounds for Read-k Oblivious Algebraic Branching Programs. In *Proceedings of the 31st Annual Computational Complexity Conference (CCC 2016)*, volume 50, pages 30:1–30:25, 2016.

[AGKS15] Manindra Agrawal, Rohit Gurjar, Arpita Korwar, and Nitin Saxena. Hitting-Sets for ROABP and Sum of Set-Multilinear Circuits. *SIAM Journal of Computing*, 44(3):669–697, 2015. Pre-print available at `arXiv:1406.7535`.

[Agr05] Manindra Agrawal. Proving Lower Bounds Via Pseudo-random Generators. In *Proceedings of the 25th International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2005)*, pages 92–105, 2005.

[Ajt05] Miklós Ajtai. A Non-linear Time Lower Bound for Boolean Branching Programs. *Theory of Computing*, 1(1):149–176, 2005. Preliminary version in the *40th Annual IEEE Symposium on Foundations of Computer Science (FOCS 1999)*.

[AR16] Vikraman Arvind and S. Raja. Some Lower Bound Results for Set-Multilinear Arithmetic Computations. *Chicago Journal of Theoretical Computer Science*, 2016.

[AV08] Manindra Agrawal and V. Vinay. Arithmetic Circuits: A Chasm at Depth Four. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2008)*, pages 67–75, 2008. Pre-print available at `eccc:TR08-062`.

[AvMV15] Matthew Anderson, Dieter van Melkebeek, and Ilya Volkovich. Deterministic Polynomial Identity Tests for Multilinear Bounded-Read Formulae. *Computational Complexity*, 24(4):695–776, 2015.

[AZ04] Martin Aigner and Günter M. Ziegler. *Proofs from THE BOOK*. Springer, 2004.

[BDVY13] Andrej Bogdanov, Zeev Dvir, Elad Verbin, and Amir Yehudayoff. Pseudorandomness for Width-2 Branching Programs. *Theory of Computing*, 9:283–293, 2013.

[BHST87] László Babai, Péter Hajnal, Endre Szemerédi, and György Turán. A Lower Bound for Read-Once-Only Branching Programs. *J. Comput. Syst. Sci.*, 35(2):153–162, 1987.

[BPW11] Andrej Bogdanov, Periklis A. Papakonstantinou, 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, 2011.

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

[BRS93] Allan Borodin, Alexander A. Razborov, and Roman Smolensky. On Lower Bounds for Read-k-Times Branching Programs. *Computational Complexity*, 3:1–18, 1993.

[BSSV03] Paul Beame, Michael E. Saks, Xiaodong Sun, and Erik Vee. Time-space trade-off lower bounds for randomized computation of decision problems. *J. ACM*, 50(2):154–195, 2003.

[De11] Anindya De. Pseudorandomness for Permutation and Regular Branching Programs. In *Proceedings of the 26th Annual IEEE Conference on Computational Complexity (CCC 2011)*, pages 221–231, 2011.

[DL78] Richard A. DeMillo and Richard J. Lipton. A Probabilistic Remark on Algebraic Program Testing. *Information Processing Letters*, 7(4):193–195, 1978.

[DS07] Zeev Dvir and Amir Shpilka. Locally Decodable Codes with Two Queries and Polynomial Identity Testing for Depth 3 Circuits. *SIAM J. Comput.*, 36(5):1404–1434, 2007. Preliminary version in the *37th Annual ACM Symposium on Theory of Computing (STOC 2005)*.

[DSY09] Zeev Dvir, Amir Shpilka, and Amir Yehudayoff. Hardness-Randomness Tradeoffs for Bounded Depth Arithmetic Circuits. *SIAM J. Comput.*, 39(4):1279–1293, 2009.

[ES35] Paul Erdős and George Szekeres. A combinatorial problem in geometry. *Compositio Math.*, 2:463–470, 1935.

[FLMS14] Hervé Fournier, Nutan Limaye, Guillaume Malod, and Srikanth Srinivasan. Lower bounds for depth 4 formulas computing iterated matrix multiplication. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC 2014)*, pages 128–135, 2014. Pre-print available at `eccc:TR13-100`.

[For14] Michael Forbes. *Polynomial Identity Testing of Read-Once Oblivious Algebraic Branching Programs*. PhD thesis, Massachusetts Institute of Technology, 2014.

[FS13a] Michael A. Forbes and Amir Shpilka. Explicit Noether Normalization for Simultaneous Conjugation via Polynomial Identity Testing. In *Proceedings of the 17th International Workshop on Randomization and Computation (RANDOM 2013)*, volume 8096 of *Lecture Notes in Computer Science*, pages 527–542. Springer, 2013.

[FS13b] Michael A. Forbes and Amir Shpilka. Quasipolynomial-Time Identity Testing of Non-commutative and Read-Once Oblivious Algebraic Branching Programs. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2013)*, pages 243–252, 2013. Full version at `arXiv:1209.2408`.

[FSS14] Michael A. Forbes, Ramprasad Saptharishi, and Amir Shpilka. Hitting sets for multilinear read-once algebraic branching programs, in any order. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC 2014)*, pages 867–875, 2014.

[GKKS13] Ankit Gupta, Pritish Kamath, Neeraj Kayal, and Ramprasad Saptharishi. Arithmetic Circuits: A Chasm at Depth Three. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2013)*, pages 578–587, 2013. Pre-print available at `eccc:TR13-026`.

[GKKS14] Ankit Gupta, Pritish Kamath, Neeraj Kayal, and Ramprasad Saptharishi. Approaching the Chasm at Depth Four. *Journal of the ACM*, 61(6):33:1–33:16, 2014. Preliminary version in the *28th Annual IEEE Conference on Computational Complexity (CCC 2013)*. Pre-print available at `eccc:TR12-098`.

[GKST16] Rohit Gurjar, Arpita Korwar, Nitin Saxena, and Thomas Thierauf. Deterministic Identity Testing for Sum of Read-once Oblivious Arithmetic Branching Programs. *Computational Complexity*, pages 1–46, 2016. Preliminary version in the *30th Annual Computational Complexity Conference (CCC 2015)*. Pre-print available at `arXiv:1411.7341`.

[GMR$^+$12] Parikshit Gopalan, Raghu Meka, Omer Reingold, Luca Trevisan, and Salil P. 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 Computer Society, 2012.

[HS80] Joos Heintz and Claus-Peter Schnorr. Testing Polynomials which Are Easy to Compute (Extended Abstract). In *Proceedings of the 12th Annual ACM Symposium on Theory of Computing (STOC 1980)*, pages 262–272, 1980.

[IMZ12] Russell Impagliazzo, Raghu Meka, and David Zuckerman. Pseudorandomness from Shrinkage. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2012)*, pages 111–119, 2012.

[INW94] 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, 1994.

[JQS10a] Maurice J. Jansen, Youming Qiao, and Jayalal Sarma. Deterministic Black-Box Identity Testing pi-Ordered Algebraic Branching Programs. In *Proceedings of the 30th International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2010)*, volume 8, pages 296–307, 2010.

[JQS10b] Maurice J. Jansen, Youming Qiao, and Jayalal Sarma. Deterministic Identity Testing of Read-Once Algebraic Branching Programs. *Electronic Colloquium on Computational Complexity (ECCC)*, 17:84, 2010.

[KI04] Valentine Kabanets and Russell Impagliazzo. Derandomizing Polynomial Identity Tests Means Proving Circuit Lower Bounds. *Computational Complexity*, 13(1-2):1–46, 2004. Preliminary version in the *35th Annual ACM Symposium on Theory of Computing (STOC 2003)*.

[KLSS14] Neeraj Kayal, Nutan Limaye, Chandan Saha, and Srikanth Srinivasan. An Exponential Lower Bound for Homogeneous Depth Four Arithmetic Circuits. In *Proceedings of the 55th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2014)*, 2014. Pre-print available at `eccc:TR14-005`.

[KMSV13] Zohar Shay Karnin, Partha Mukhopadhyay, Amir Shpilka, and Ilya Volkovich. Deterministic Identity Testing of Depth-4 Multilinear Circuits with Bounded Top Fan-in. *SIAM Journal of Computing*, 42(6):2114–2131, 2013. Preliminary version in the *42nd Annual ACM Symposium on Theory of Computing (STOC 2010)*. Pre-print available at `eccc:TR09-116`.

[KNP11] Michal Koucký, Prajakta Nimbhorkar, and Pavel Pudlák. Pseudorandom generators for group products: extended abstract. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing (STOC 2011)*, pages 263–272, 2011.

[KNS16] Neeraj Kayal, Vineet Nair, and Chandan Saha. Separation Between Read-once Oblivious Algebraic Branching Programs (ROABPs) and Multilinear Depth Three Circuits. In *Proceedings of the 33rd Symposium on Theoretical Aspects of Computer Science (STACS 2016)*, volume 47 of *LIPIcs*, pages 46:1–46:15, 2016. Pre-print available at `eccc:TR15-154`.

[Koi12] Pascal Koiran. Arithmetic Circuits: The Chasm at Depth Four Gets Wider. *Theoretical Computer Science*, 448:56–65, 2012. Pre-print available at `arXiv:1006.4700`.

[Kru53] Joseph B Kruskal. Monotonic subsequences. *Proceedings of the American Mathematical Society*, 4(2):264–274, 1953.

[KS07] Neeraj Kayal and Nitin Saxena. Polynomial Identity Testing for Depth 3 Circuits. *Computational Complexity*, 16(2):115–138, 2007.

[KS09] Neeraj Kayal and Shubhangi Saraf. Blackbox polynomial identity testing for depth-3 circuits. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2009)*, 2009.

[KS14a] Mrinal Kumar and Shubhangi Saraf. The limits of depth reduction for arithmetic formulas: it's all about the top fan-in. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC 2014)*, pages 136–145, 2014. Pre-print available at `eccc:TR13-068`.

[KS14b] Mrinal Kumar and Shubhangi Saraf. On the power of homogeneous depth 4 arithmetic circuits. In *Proceedings of the 55th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2014)*, 2014. Pre-print available at `eccc:TR14-045`.

[KSS14] Neeraj Kayal, Chandan Saha, and Ramprasad Saptharishi. A super-polynomial lower bound for regular arithmetic formulas. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC 2014)*, pages 146–153, 2014. Pre-print available at `eccc:TR13-091`.

[KSS15] Swastik Kopparty, Shubhangi Saraf, and Amir Shpilka. Equivalence of Polynomial Identity Testing and Polynomial Factorization. *Computational Complexity*, 24(2):295–331, 2015. Preliminary version in the *29th Annual IEEE Conference on Computational Complexity (CCC 2014)*.

[KUW86] Richard M. Karp, Eli Upfal, and Avi Wigderson. Constructing a perfect matching is in random NC. *Combinatorica*, 6(1):35–48, 1986. Preliminary version in the *17th Annual ACM Symposium on Theory of Computing (STOC 1985)*.

[Mul12] Ketan Mulmuley. Geometric Complexity Theory V: Equivalence between Blackbox Derandomization of Polynomial Identity Testing and Derandomization of Noether's Normalization Lemma. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2012)*, pages 629–638, 2012.

[MV17] Daniel Minahan and Ilya Volkovich. Complete Derandomization of Identity Testing and Reconstruction of Read-Once Formulas. In *Proceedings of the 32nd Annual Computational Complexity Conference (CCC 2017)*, 2017.

[MVV87] Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987. Preliminary version in the *19th Annual ACM Symposium on Theory of Computing (STOC 1987)*.

[Nis91] Noam Nisan. Lower bounds for non-commutative computation. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing (STOC 1991)*, pages 410–418, 1991. Available on `citeseer:10.1.1.17.5067`.

[Nis92] Noam Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992.

[NW94] Noam Nisan and Avi Wigderson. Hardness vs Randomness. *Journal of Computer and System Sciences*, 49(2):149–167, 1994. Available on `citeseer:10.1.1.83.8416`.

[Oko91] E.A. Okolnishnikova. Lower bounds on the complexity of realization of characteristic functions of binary codes by branching programs. *Metody Diskretnogo Analiza*, 51:61–83, 1991.

[OSV15] Rafael Oliveira, Amir Shpilka, and Ben Lee Volk. Subexponential Size Hitting Sets for Bounded Depth Multilinear Formulas. In *Proceedings of the 30th Annual Computational Complexity Conference (CCC 2015)*, pages 304–322, 2015. Pre-print available at `arXiv:1411.7492`.

[RS05] Ran Raz and Amir Shpilka. Deterministic polynomial identity testing in non-commutative models. *Computational Complexity*, 14(1):1–19, 2005. Preliminary version in the *19th Annual IEEE Conference on Computational Complexity (CCC 2004)*.

[RSV13] Omer Reingold, Thomas Steinke, and Salil P. Vadhan. Pseudorandomness for Regular Branching Programs via Fourier Analysis. In *Proceedings of the 17th International Workshop on Randomization and Computation (RANDOM 2013)*, pages 655–670, 2013.

[RY09] Ran Raz and Amir Yehudayoff. Lower Bounds and Separations for Constant Depth Multilinear Circuits. *Computational Complexity*, 18(2):171–207, 2009. Preliminary version in the *23rd Annual IEEE Conference on Computational Complexity (CCC 2008)*. Pre-print available at `eccc:TR08-006`.

[Sch80] Jacob T. Schwartz. Fast Probabilistic Algorithms for Verification of Polynomial Identities. *Journal of the ACM*, 27(4):701–717, 1980.

[SS12] Nitin Saxena and C. Seshadhri. Blackbox Identity Testing for Bounded Top-Fanin Depth-3 Circuits: The Field Doesn't Matter. *SIAM J. Comput.*, 41(5):1285–1298, 2012. Preliminary version in the *43rd Annual ACM Symposium on Theory of Computing (STOC 2011)*.

[Ste12] Thomas Steinke. Pseudorandomness for Permutation Branching Programs Without the Group Theory. *Electronic Colloquium on Computational Complexity (ECCC)*, 19:83, 2012.

[SV10] Amir Shpilka and Ilya Volkovich. On the Relation between Polynomial Identity Testing and Finding Variable Disjoint Factors. In *Proceedings of the 37th International Colloquium on Automata, Languages and Programming (ICALP 2010)*, pages 408–419, 2010. Pre-print available at `eccc:TR10-036`.

[SV11] Shubhangi Saraf and Ilya Volkovich. Black-box identity testing of depth-4 multilinear circuits. In *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing (STOC 2011)*, pages 421–430, 2011.

[SV15] Amir Shpilka and Ilya Volkovich. Read-once polynomial identity testing. *Computational Complexity*, 24(3):477–532, 2015. Preliminary version in the *40th Annual ACM Symposium on Theory of Computing (STOC 2008)*.

[SVW14] Thomas Steinke, Salil P. Vadhan, and Andrew Wan. Pseudorandomness and Fourier Growth Bounds for Width-3 Branching Programs. In *Proceedings of the 18th International Workshop on Randomization and Computation (RANDOM 2014)*, pages 885–899, 2014.

[SY10] Amir Shpilka and Amir Yehudayoff. Arithmetic Circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5:207–388, March 2010.

[Tav15] Sébastien Tavenas. Improved bounds for reduction to depth 4 and depth 3. *Inf. Comput.*, 240:2–11, 2015. Preliminary version in the *38th Internationl Symposium on the Mathematical Foundations of Computer Science (MFCS 2013)*.

[Tha98] Jayram S. Thathachar. On Separating the Read-k-Times Branching Program Hierarchy. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC 1998)*, pages 653–662, 1998.

[VSBR83] Leslie G. Valiant, Sven Skyum, S. Berkowitz, and Charles Rackoff. Fast Parallel Computation of Polynomials Using Few Processors. *SIAM Journal of Computing*, 12(4):641–644, 1983. Preliminary version in the *6th Internationl Symposium on the Mathematical Foundations of Computer Science (MFCS 1981)*.

[Weg88] Ingo Wegener. On the complexity of branching programs and decision trees for clique functions. *J. ACM*, 35(2):461–471, 1988.

[Zák84] Stanislav Zák. An Exponential Lower Bound for One-Time-Only Branching Programs. In *Proceedings of the 9th Internationl Symposium on the Mathematical Foundations of Computer Science (MFCS 1984)*, volume 176 of *Lecture Notes in Computer Science*, pages 562–566. Springer, 1984.

[Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. In *Symbolic and Algebraic Computation, EUROSAM '79, An International Symposiumon Symbolic and Algebraic Computation*, volume 72 of *Lecture Notes in Computer Science*, pages 216–226. Springer, 1979.

A Lemma A.1

Lemma A.1. *Let p be a real number such that $0 < p < 1$ and r be a positive integer. For any $n \in \mathbb{N}$,*

$$n^{1-p} - (n - n^p/r)^{1-p} \geq (1-p)/r.$$

Proof. Define $f : \mathbb{R}^+ \to \mathbb{R}^+$ by $f(x) = x^{1-p} - (x - x^p/r)^{1-p}$. We show that this real function is non-increasing for non-negative x, and that its limit as x tends to infinity is $(1-p)/r$, which implies the statement of the lemma.

To show that f is non-increasing, we show that its derivative is non-positive. Note that

$$f'(x) = (1-p)x^{-p} - (1-p)\left(1 - \frac{px^{p-1}}{r}\right)\left(x - \frac{x^p}{r}\right)^{-p}.$$

To show that $f'(x) \leq 0$ for all x, it thus suffices, after some rearrangements, to prove the inequality

$$\left(x - \frac{x^p}{r}\right)^p \leq x^p\left(1 - \frac{px^{p-1}}{r}\right). \tag{A.2}$$

We have that

$$\left(x - \frac{x^p}{r}\right)^p = x^p\left(1 - \frac{x^{p-1}}{r}\right)^p,$$

and thus after dividing by x^p, (A.2) follows as a corollary of the well-known inequality $(1-y)^s \leq 1 - sy$ for $y > 0$ and $0 < s < 1$, that can be proved using the Taylor expansion of $(1-y)^s$ around 0.

In order to calculate the limit, observe that,

$$f(x) = x^{1-p} \cdot \left(1 - \left(1 - \frac{x^{p-1}}{r}\right)^{1-p}\right) = \frac{1 - \left(1 - \frac{x^{p-1}}{r}\right)^{1-p}}{x^{p-1}},$$

so by L'Hôpital's Rule we get that

$$\lim_{x\to\infty} \frac{1 - \left(1 - \frac{x^{p-1}}{r}\right)^{1-p}}{x^{p-1}} = \lim_{x\to\infty} \frac{\frac{-(p-1)^2}{r} \cdot x^{p-2} \cdot \left(1 - \frac{x^{p-1}}{r}\right)^{-p}}{(p-1)x^{p-2}} = \frac{1-p}{r}.$$ □