Deterministic Identity Testing for Sum of Read-Once Oblivious Arithmetic Branching Programs*

Rohit Gurjar[†2], Arpita Korwar[‡1], Nitin Saxena[§1] and Thomas Thierauf[¶2]

[1]Department of Computer Science and Engineering, IIT Kanpur, India
[2]Aalen University, Germany

Abstract

A *read-once oblivious arithmetic branching program (ROABP)* is an arithmetic branching program (ABP) where each variable occurs in at most one layer. We give the first polynomial time whitebox identity test for a polynomial computed by a sum of constantly many ROABPs. We also give a corresponding blackbox algorithm with quasi-polynomial time complexity $n^{O(\log n)}$. In both the cases, our time complexity is double exponential in the number of ROABPs.

ROABPs are a generalization of set-multilinear depth-3 circuits. The prior results for the sum of constantly many set-multilinear depth-3 circuits were only slightly better than brute-force, i.e. exponential-time.

Our techniques are a new interplay of three concepts for ROABP: low evaluation dimension, basis isolating weight assignment and low-support rank concentration. We relate basis isolation to rank concentration and extend it to a sum of two ROABPs using evaluation dimension.

1 Introduction

Polynomial Identity Testing (PIT) is the problem of testing whether a given n-variate polynomial is identically zero or not. The input to the PIT problem may be in the form of arithmetic circuits or arithmetic branching programs (ABP). They are the arithmetic analogues of boolean circuits and boolean branching programs, respectively. It is well known that PIT can be solved in randomized polynomial time, see e.g. [Sch80, Zip79, DL78]. The randomized algorithm just evaluates the polynomial at random points; thus, it is a *blackbox* algorithm. In contrast, an algorithm is a *whitebox* algorithm if it looks inside the given circuit or branching program. In this work, we consider both whitebox and blackbox algorithms.

Since all problems with randomized polynomial-time solutions are conjectured to have deterministic polynomial-time algorithms, we expect that such an algorithm exists for PIT. It is also known that any sub-exponential time algorithm for PIT implies an arithmetic circuit lower bound [KI04, Agr05]. See also the surveys [Sax09, Sax14, SY10].

*A preliminary version of the paper was presented at CCC'15 [GKST15]

[†]rgurjar@cse.iitk.ac.in, supported by DFG grant TH 472/4 and TCS PhD research fellowship

[‡]arpk@cse.iitk.ac.in

[§]nitin@cse.iitk.ac.in, supported by DST-SERB

[¶]thomas.thierauf@htw-aalen.de, supported by DFG grant TH 472/4

1.1 PIT for arithmetic circuits

An arithmetic circuit is defined like a boolean circuit, except that its gates are addition or multiplication gates and inputs are variables or constants. It computes a polynomial in the natural way. Without loss of generality one can assume that the addition (Σ) and multiplication (Π) gates in an arithmetic circuit appear in alternate layers. An efficient deterministic solution for PIT is known only for very restricted input models, for example, sparse polynomials or depth-2 ($\Sigma\Pi$) circuits [BOT88, KS01], constant fan-in depth-3 ($\Sigma\Pi\Sigma$) circuits [DS07, KS07, KS09, KS11, SS11, SS12], set-multilinear circuits [RS05, FS12a, ASS13], read-once oblivious ABP (ROABP) [RS05, FS13, FSS14, AGKS15]. This lack of progress is not surprising: Gupta et al. [GKKS13] showed that a polynomial time test for depth-3 circuits over fields of characteristic zero would imply a sub-exponential time test for general circuits. For now, even a sub-exponential solution for depth-3 circuits seems elusive. However, an efficient test for depth-3 multilinear circuits looks within reach as a lower bound against this class of circuits is already known [RY09]. A circuit is called *multilinear* if all its gates compute a multilinear polynomial, i.e. polynomials such that the maximum degree of any variable is one.

A circuit is called *depth-3 set-multilinear* if it is a depth-3 multilinear ($\Sigma\Pi\Sigma$) circuit such that all the product gates in it induce the same partition on the set of variables. It is easy to see that a depth-3 multilinear circuit is a sum of at most s depth-3 set-multilinear circuits, where s is the fan-in of the top addition gate. Hence, a natural first step to attack depth-3 multilinear circuits is to find an efficient test for the sum of two depth-3 set-multilinear polynomials. Before this work, the only non-trivial test known for sum of two depth-3 set-multilinear circuits was a sub-exponential whitebox algorithm by Agrawal et al. [AGKS15]. Subsequently, a sub-exponential time blackbox test was also given for depth-3 multilinear circuits [OSV15]. In this work, we give the first polynomial-time whitebox algorithm, and the first quasi-polynomial-time blackbox algorithm, for the sum of two depth-3 set-multilinear circuits. Our results actually hold for a stronger model, the sum of two read-once oblivious arithmetic branching programs.

1.2 PIT for the sum of read-once oblivious arithmetic branching programs

An *arithmetic branching program* (ABP) is a directed layered graph with a source and a sink. Its edges are labeled by univariate polynomials. Each path of this graph computes the product of the labels on its edges and the ABP computes the sum of these products over all paths from the source to the sink.

In this paper, we deal with *read-once oblivious ABPs* (ROABP). These are arithmetic branching programs, where each variable occurs in at most one layer. ROABPs subsume depth-3 set-multilinear circuits; for each product gate of such a circuit, one can construct a width-2 ROABP in the same variable order (see for example [AGKS15, Lemma 14]).

There has been a long chain of work on identity testing for ROABP; see the thesis of Michael Forbes [For14] for an excellent overview. In 2005, Raz and Shpilka [RS05] gave a polynomial-time whitebox test for ROABP. Then, Forbes and Shpilka [FS13] gave an $s^{O(\log n)}$-time blackbox algorithm for ROABP with known variable order, where s is the size of the ROABP and n is number of variables. This was followed by a complete blackbox test [FSS14] that took $s^{O(d \log^2 s)}$ steps, where d is the syntactic degree bound of any variable. This was further improved by Agrawal et al. [AGKS15] to $s^{O(\log n)}$ time. They removed the exponen-

tial dependence on the degree d. Their test is based on the idea of *basis isolating weight assignment*. Given a polynomial over an algebra, it assigns weights to the variables, and naturally extends it to monomials, such that there is a unique minimum weight basis among the coefficients of the polynomial.

We consider the *sum of* ROABPs. Kayal, Nair and Saha [KNS15, Theorem 2] have shown that there is a polynomial $P(\boldsymbol{x})$ computed by a sum of two ROABPs such that any single ROABP that computes $P(\boldsymbol{x})$ has exponential size. Hence, the previous results on single ROABPs do not help here. In Section 3 we show our first main result, a whitebox PIT for the sum of ROABPs (Theorem 3.2):

PIT for the sum of constantly many ROABP*s is in polynomial time.*

The exact time bound we get for the PIT-algorithm is $(ndw^{2^c})^{O(c)}$, where n is the number of variables, d is the degree bound of the variables, c is the number of ROABPs and w is their width. Hence our time bound is double exponential in c, but polynomial in n, d, w.

In another work, Jansen et al. [JQS10] gave a blackbox test for a sum of constantly many arithmetic branching programs which they also call *ROABP*. However, they define a much weaker model where every variable may appear on at most one edge in the ABP. Hence our results are by far more general.

1.2.1 Techniques

Our algorithm uses the fact that the *evaluation dimension* of an ROABP is equal to the width of the ROABP [Nis91, FS12b]. The evaluation dimension of a polynomial is the dimension of its partial evaluations with respect to a subset of variables. Essentially, the linear dependencies among the partial evaluations of a polynomial define the ROABP computing it. Identity testing of the sum of two ROABPs is the same as testing the equivalence of two ROABPs. Our algorithm is inspired from a similar result in the boolean case. The boolean version of an ROABP is called an *oblivious binary decision diagram* (OBDD). Testing the equivalence of two OBDDs is in polynomial time [SW97]. OBDDs have a similar property of small evaluation dimension. However, when considering partial evaluations, the notion of *linear dependence* becomes *equality* in the boolean setting. Our equivalence test for two ROABPs A and B takes linear dependencies among partial evaluations of A and verifies them for the corresponding partial evaluations of B. As B is an ROABP, the verification of these dependencies reduces to identity testing for a single ROABP.

In Section 3.2, we generalize this test to the sum of c ROABPs. There we take A as one ROABP and B as the sum of the remaining $c-1$ ROABPs. In this case, the verification of the dependencies for B becomes the question of identity testing of a sum of $c-1$ ROABPs, which we solve recursively.

We can apply our technique to *boolean* branching programs. By arithmetization, one can transform a boolean branching program into an arithmetic one (see Section 2.4). Hence, we get a PIT for the xor of c OBDDs. Here, PIT means to test whether it is the boolean zero function. In fact, this generalizes to *parity-OBDDs* (Corollary 3.3). These are nondeterministic OBDDs with a parity acceptance mechanism.

1.3 Blackbox PIT for the sum of read-once oblivious arithmetic branching programs

In Section 4, we give an identity test for a sum of ROABPs in the blackbox setting. That is, we are given blackbox access to a sum of ROABPs and *not* to the individual ROABPs. Our main result here is as follows (Theorem 4.10):

> *There is a blackbox PIT for the sum of constantly many* ROABP*s that works in quasi-polynomial time.*

The exact time bound we get for the PIT-algorithm is $(ndw)^{O(c\,2^c \log(ndw))}$, where n is the number of variables, d is the degree bound of the variables, c is the number of ROABPs and w is their width. Hence our time bound is double exponential in c, and quasi-polynomial in n, d, w. In an independent work, Kayal et al. [KNS15] give a quasi-polynomial time blackbox PIT for the sum of c depth-3 set-multilinear circuits, where the dependence on c is only exponential. However, they impose the restriction on the circuit that it is a *superposition* of a small number of depth-3 set-multilinear circuits. That is, there is a partition of the variables into a small number of sets, such that with respect to each set, the circuit is set-multilinear.

1.3.1 Techniques

Like the whitebox test, we reduce the question to identity testing for a single ROABP, using the low evaluation dimension property. But, just a hitting-set for ROABP does not suffice here, we need an efficient shift of the variables which gives low-support concentration in any polynomial computed by an ROABP. An ℓ-concentration in a polynomial $P(\boldsymbol{x})$ means that all of its coefficients are in the linear span of its coefficients corresponding to monomials with support $< \ell$. Essentially we show that a shift, which achieves low-support concentration for an ROABP of width w^{2^c}, also works for a sum of c ROABPs (Lemma 4.9). This is surprising, because as mentioned above, a sum of c ROABPs is not captured by an ROABP with polynomially bounded width [KNS15].

A novel part of our proof is the idea that for a polynomial over a k-dimensional $\mathbb{F}$-algebra $\mathbb{A}_k$, a shift by a basis isolating weight assignment achieves low-support concentration. To elaborate, let $\mathrm{w}\colon \boldsymbol{x} \to \mathbb{N}$ be a basis isolating weight assignment for a polynomial $P(\boldsymbol{x}) \in \mathbb{A}_k[\boldsymbol{x}]$ and let t be a new variable. Then $P(\boldsymbol{x})$ shifted by t^{w}, namely $P(x_1 + t^{\mathrm{w}(x_1)}, \ldots, x_n + t^{\mathrm{w}(x_n)})$ has $O(\log k)$-concentration over $\mathbb{F}(t)$. As Agrawal et al. [AGKS15] gave a basis isolating weight assignment for ROABPs, we can use it to get low-support concentration. Forbes et al. [FSS14] had also achieved low-support concentration in ROABPs, but with a higher cost. Our concentration proof significantly differs from the older rank concentration proofs [ASS13, FSS14], which always assume *distinct* weights for all the monomials or coefficients. Here, we only require that the weight of a coefficient is greater than the weight of the basis coefficients that it depends on.

2 Preliminaries

2.1 Notation

Let $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ be a tuple of n variables. For any $\boldsymbol{a} = (a_1, a_2, \ldots, a_n) \in \mathbb{N}^n$, we denote by $\boldsymbol{x}^{\boldsymbol{a}}$ the monomial $\prod_{i=1}^{n} x_i^{a_i}$. The *support size* of a monomial $\boldsymbol{x}^{\boldsymbol{a}}$ is given by $\mathrm{supp}(\boldsymbol{a}) = |\{a_i \neq 0 \mid i \in [n]\}|$.

Let $\mathbb{F}$ be some field. Let $A(\boldsymbol{x})$ be a polynomial over $\mathbb{F}$ in n variables. A polynomial $A(\boldsymbol{x})$ is said to have *individual degree* d, if the degree of each variable is bounded by d for each monomial in $A(\boldsymbol{x})$. When $A(\boldsymbol{x})$ has individual degree d, then the exponent $\boldsymbol{a}$ of any monomial $\boldsymbol{x}^{\boldsymbol{a}}$ of $A(\boldsymbol{x})$ is in the set

$$M = \{0, 1, \ldots, d\}^n .$$

By $\mathrm{coeff}_A(\boldsymbol{x}^{\boldsymbol{a}}) \in \mathbb{F}$ we denote the coefficient of the monomial $\boldsymbol{x}^{\boldsymbol{a}}$ in $A(\boldsymbol{x})$. Hence, we can write

$$A(\boldsymbol{x}) = \sum_{\boldsymbol{a} \in M} \mathrm{coeff}_A(\boldsymbol{x}^{\boldsymbol{a}})\, \boldsymbol{x}^{\boldsymbol{a}} .$$

The *sparsity* of polynomial $A(\boldsymbol{x})$ is the number of nonzero coefficients $\mathrm{coeff}_A(\boldsymbol{x}^{\boldsymbol{a}})$.

We also consider *matrix polynomials* where the coefficients $\mathrm{coeff}_A(\boldsymbol{x}^{\boldsymbol{a}})$ are $w \times w$ matrices, for some w. In an abstract setting, these are polynomials over a w^2-dimensional $\mathbb{F}$-algebra $\mathbb{A}$. Recall that an $\mathbb{F}$-algebra is a vector space over $\mathbb{F}$ with a multiplication which is bilinear and associative, i.e. $\mathbb{A}$ is a ring. The *coefficient space* is then defined as the span of all coefficients of A, i.e., $\mathrm{span}_{\mathbb{F}}\{\mathrm{coeff}_A(\boldsymbol{x}^a) \mid \boldsymbol{a} \in M\}$.

Consider a partition of the variables $\boldsymbol{x}$ into two parts $\boldsymbol{y}$ and $\boldsymbol{z}$, with $|\boldsymbol{y}| = k$. A polynomial $A(\boldsymbol{x})$ can be viewed as a polynomial in variables $\boldsymbol{y}$, where the coefficients are polynomials in $\mathbb{F}[\boldsymbol{z}]$. For monomial $\boldsymbol{y}^{\boldsymbol{a}}$, let us denote the coefficient of $\boldsymbol{y}^{\boldsymbol{a}}$ in $A(\boldsymbol{x})$ by $A_{(\boldsymbol{y},\boldsymbol{a})} \in \mathbb{F}[\boldsymbol{z}]$. For example, in the polynomial $A(\boldsymbol{x}) = x_1 + x_1 x_2 + {x_1}^2$, we have $A_{(x_1,1)} = 1 + x_2$, whereas $\mathrm{coeff}_A(x_1) = 1$. Observe that $\mathrm{coeff}_A(\boldsymbol{y}^{\boldsymbol{a}})$ is the constant term in $A_{(\boldsymbol{y},\boldsymbol{a})}$.

Thus, $A(\boldsymbol{x})$ can be written as

$$A(\boldsymbol{x}) = \sum_{\boldsymbol{a} \in \{0,1,\ldots,d\}^k} A_{(\boldsymbol{y},\boldsymbol{a})}\, \boldsymbol{y}^{\boldsymbol{a}} . \tag{1}$$

The coefficient $A_{(\boldsymbol{y},\boldsymbol{a})}$ can also be expressed as a partial derivative $\frac{\partial A}{\partial \boldsymbol{y}^{\boldsymbol{a}}}$ evaluated at $\boldsymbol{y} = \boldsymbol{0}$ (and multiplied by an appropriate constant), see [FS12b, Section 6]. Therefore, we sometimes call the coefficients $A_{(\boldsymbol{y},\boldsymbol{a})}$ the *partial derivative polynomials of* A.

For a set of polynomials $\mathcal{P}$, we define their $\mathbb{F}$-span as

$$\mathrm{span}_{\mathbb{F}}\, \mathcal{P} = \left\{ \sum_{A \in \mathcal{P}} \alpha_A A \mid \alpha_A \in \mathbb{F} \text{ for all } A \in \mathcal{P} \right\} .$$

The set of polynomials $\mathcal{P}$ is said to be $\mathbb{F}$-*linearly independent* if $\sum_{A \in \mathcal{P}} \alpha_A A = 0$ holds only for $\alpha_A = 0$, for all $A \in \mathcal{P}$. The *dimension* $\dim_{\mathbb{F}} \mathcal{P}$ of $\mathcal{P}$ is the cardinality of the largest $\mathbb{F}$-linearly independent subset of $\mathcal{P}$.

For a matrix R, we denote by $R(i, \cdot)$ and $R(\cdot, i)$ the i-th row and the i-th column of R, respectively. For any $a \in \mathbb{F}^{k \times k'}, b \in \mathbb{F}^{\ell \times \ell'}$, the tensor product of a and b is denoted by $a \otimes b$. The inner product is denoted by $\langle a, b \rangle$. We abuse this notation slightly: for any $a, R \in \mathbb{F}^{w \times w}$, let $\langle a, R \rangle = \sum_{i=1}^{w} \sum_{j=1}^{w} a_{ij} R_{ij}$.

2.2 Arithmetic branching programs

An *arithmetic branching program* (ABP) is a directed graph with $\ell + 1$ layers of vertices $(V_0, V_1, \ldots, V_\ell)$. The layers V_0 and V_ℓ each contain only one vertex, the *start node*, v_0 and the *end node*, v_ℓ, respectively. The edges are allowed only between the consecutive layers of the

ABP. All the edges in the graph have weights from $\mathbb{F}[\boldsymbol{x}]$, for some field $\mathbb{F}$. The *length* of an ABP is the length of a longest path in the ABP, i.e. ℓ. An ABP has *width* w, if $|V_i| \leq w$ for all $0 \leq i \leq \ell$.

For an edge e, let us denote its weight by $W(e)$. For a path p, its weight $W(p)$ is defined to be the product of weights of all the edges in it,

$$W(p) = \prod_{e \in p} W(e).$$

The *polynomial* $A(\boldsymbol{x})$ *computed by the ABP* is the sum of the weights of all the paths from v_0 to v_ℓ,

$$A(\boldsymbol{x}) = \sum_{p \text{ path } v_0 \rightsquigarrow v_\ell} W(p).$$

Let the set of nodes in V_i be $\{v_{i,j} \mid j \in [w]\}$. The branching program can alternately be represented by a matrix product $\prod_{i=1}^{\ell} D_i$, where $D_1 \in \mathbb{F}[\boldsymbol{x}]^{1 \times w}$, $D_i \in \mathbb{F}[\boldsymbol{x}]^{w \times w}$ for $2 \leq i \leq \ell - 1$, and $D_\ell \in \mathbb{F}[\boldsymbol{x}]^{w \times 1}$ such that

$$\begin{aligned} D_1(j) &= W(v_0, v_{1,j}), \text{ for } 1 \leq j \leq w, \\ D_i(j,k) &= W(v_{i-1,j}, v_{i,k}), \text{ for } 1 \leq j,k \leq w \text{ and } 2 \leq i \leq n-1, \\ D_\ell(k) &= W(v_{\ell-1,k}, v_\ell), \text{ for } 1 \leq k \leq w. \end{aligned}$$

Here we use the convention that $W(u,v) = 0$ if (u,v) is not an edge in the ABP.

2.3 Read-once oblivious arithmetic branching programs

An ABP is called a *read-once oblivious ABP (ROABP)* if the edge weights in every layer are univariate polynomials in the same variable, and every variable occurs in at most one layer. Hence, the length of an ROABP is n, the number of variables. The entries in the matrix D_i defined above come from $\mathbb{F}[x_{\pi(i)}]$, for all $i \in [n]$, where π is a permutation on the set $[n]$. The order $(x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(n)})$ is said to be the *variable order* of the ROABP.

We will view D_i as a polynomial in the variable $x_{\pi(i)}$, whose coefficients are w-dimensional vectors or matrices. Namely, for an exponent $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$, the coefficient of

- $x_{\pi(1)}^{a_{\pi(1)}}$ in $D_1(x_{\pi(1)})$ is the row vector $\text{coeff}_{D_1}(x_{\pi(1)}^{a_{\pi(1)}}) \in \mathbb{F}^{1 \times w}$,
- $x_{\pi(i)}^{a_{\pi(i)}}$ in $D_i(x_{\pi(i)})$ is the matrix $\text{coeff}_{D_i}(x_{\pi(i)}^{a_{\pi(i)}}) \in \mathbb{F}^{w \times w}$, for $i = 2, 3, \ldots, n-1$, and
- $x_{\pi(n)}^{a_{\pi(n)}}$ in $D_n(x_{\pi(n)})$ is the vector $\text{coeff}_{D_n}(x_{\pi(n)}^{a_{\pi(n)}}) \in \mathbb{F}^{w \times 1}$.

The read once property gives us an easy way to express the coefficients of the polynomial $A(\boldsymbol{x})$ computed by an ROABP.

Lemma 2.1. *For a polynomial* $A(\boldsymbol{x}) = D_1(x_{\pi(1)}) D_2(x_{\pi(2)}) \cdots D_n(x_{\pi(n)})$ *computed by an* ROABP*, we have*

$$\text{coeff}_A(\boldsymbol{x}^{\boldsymbol{a}}) = \prod_{i=1}^{n} \text{coeff}_{D_i}(x_{\pi(i)}^{a_{\pi(i)}}) \quad \in \mathbb{F}\,. \tag{2}$$

We also consider matrix polynomials computed by an ROABP. A matrix polynomial $A(\boldsymbol{x}) \in F^{w\times w}[\boldsymbol{x}]$ is said to be computed by an ROABP if $A = D_1 D_2 \cdots D_n$, where $D_i \in F^{w\times w}[x_{\pi(i)}]$ for $i \in [n]$ and some permutation π on $[n]$. Similarly, a vector polynomial $A(\boldsymbol{x}) \in F^{1\times w}[\boldsymbol{x}]$ is said to be computed by an ROABP if $A = D_1 D_2 \cdots D_n$, where $D_1 \in F^{1\times w}[x_{\pi(1)}]$ and $D_i \in F^{w\times w}[x_{\pi(i)}]$ for $i \in \{2, \ldots, n\}$. Usually, we will assume that an ROABP computes a polynomial in $\mathbb{F}[\boldsymbol{x}]$, unless mentioned otherwise.

Let $A(\boldsymbol{x})$ be the polynomial computed by an ROABP and let $\boldsymbol{y}$ and $\boldsymbol{z}$ be a partition of the variables $\boldsymbol{x}$ such that $\boldsymbol{y}$ is a *prefix* of the variable order of the ROABP. Recall from equation (1) that $A_{(\boldsymbol{y},\boldsymbol{a})} \in \mathbb{F}[\boldsymbol{z}]$ is the coefficient of monomial $\boldsymbol{y}^{\boldsymbol{a}}$ in $A(\boldsymbol{x})$. Nisan [Nis91] showed that for every prefix $\boldsymbol{y}$, the dimension of the set of coefficient polynomials $A_{(\boldsymbol{y},\boldsymbol{a})}$ is bounded by the width of the ROABP[1]. This holds in spite of the fact that the number of these polynomials is large.

Lemma 2.2 ([Nis91], Prefix $\boldsymbol{y}$)**.** *Let $A(\boldsymbol{x})$ be a polynomial of individual degree d, computed by an ROABP of width w with variable order $(x_1, x_2, \ldots, x_n)$. Let $k \le n$ and $\boldsymbol{y} = (x_1, x_2, \ldots, x_k)$ be the prefix of length k of $\boldsymbol{x}$. Then $\dim_{\mathbb{F}}\{A_{(\boldsymbol{y},\boldsymbol{a})} \mid \boldsymbol{a} \in \{0, 1, \ldots, d\}^k\} \le w$.*

Proof. Let $A(\boldsymbol{x}) = D_1(x_1)\, D_2(x_2) \cdots D_n(x_n)$, where $D_1 \in \mathbb{F}^{1\times w}[x_1]$, $D_n \in \mathbb{F}^{w\times 1}[x_n]$ and $D_i \in \mathbb{F}^{w\times w}[x_i]$, for $2 \le i \le n-1$. Let $\boldsymbol{z} = (x_{k+1}, x_{k+2}, \ldots, x_n)$ be the remaining variables of $\boldsymbol{x}$. Define $P(\boldsymbol{y}) = D_1 D_2 \cdots D_k$ and $Q(\boldsymbol{z}) = D_{k+1} D_{k+2} \cdots D_n$. Then P and Q are vectors of length w,

$$P(\boldsymbol{y}) = [P_1(\boldsymbol{y})\ P_2(\boldsymbol{y})\ \cdots\ P_w(\boldsymbol{y})]$$
$$Q(\boldsymbol{z}) = [Q_1(\boldsymbol{z})\ Q_2(\boldsymbol{z})\ \cdots\ Q_w(\boldsymbol{z})]^T$$

where $P_i(\boldsymbol{y}) \in \mathbb{F}[\boldsymbol{y}]$ and $Q_i(\boldsymbol{z}) \in \mathbb{F}[\boldsymbol{z}]$, for $1 \le i \le w$, and we have $A(\boldsymbol{x}) = P(\boldsymbol{y})\, Q(\boldsymbol{z})$.

We get the following generalization of equation (2): for any $\boldsymbol{a} \in \{0, 1, \ldots, d\}^k$, the coefficient $A_{(\boldsymbol{y},\boldsymbol{a})} \in \mathbb{F}[\boldsymbol{z}]$ of monomial $\boldsymbol{y}^{\boldsymbol{a}}$ can be written as

$$A_{(\boldsymbol{y},\boldsymbol{a})} = \sum_{i=1}^{w} \operatorname{coeff}_{P_i}(\boldsymbol{y}^{\boldsymbol{a}})\, Q_i(\boldsymbol{z}). \tag{3}$$

That is, every $A_{(\boldsymbol{y},\boldsymbol{a})}$ is in the $\mathbb{F}$-span of the polynomials $Q_1, Q_2, \ldots, Q_w$. Hence, the claim follows. □

Observe that equation (3) tells us that the polynomials $A_{(\boldsymbol{y},\boldsymbol{a})}$ can also be computed by an ROABP of width w: by equation (2), we have $\operatorname{coeff}_{P_i}(\boldsymbol{y}^{\boldsymbol{a}}) = \prod_{x_i \in \boldsymbol{y}} \operatorname{coeff}_{D_i}(x_i^{a_i})$. Hence, in the ROABP for A, we simply have to replace the matrices D_i which belong to P by the coefficient matrices $\operatorname{coeff}_{D_i}(x_i^{a_i})$. Here, we have that $\boldsymbol{y}$ is a prefix of $\boldsymbol{x}$. But note that this is not necessary for the construction to work. The variables in $\boldsymbol{y}$ can be arbitrarily distributed in $\boldsymbol{x}$. We summarize the observation in the following lemma.

Lemma 2.3 (Arbitrary $\boldsymbol{y}$)**.** *Let $A(\boldsymbol{x})$ be a polynomial of individual degree d, computed by an ROABP of width w and $\boldsymbol{y} = (x_{i_1}, x_{i_2}, \ldots, x_{i_k})$ be any k variables of x. Then the polynomial $A_{(\boldsymbol{y},\boldsymbol{a})}$ can be computed by an ROABP of width w, for every $\boldsymbol{a} \in \{0, 1, \ldots, d\}^k$. Moreover, all these ROABPs have the same variable order, inherited from the order of the ROABP for A.*

[1] Nisan [Nis91] showed it for non-commutative ABP, but the same proof works for ROABP.

For a general polynomial, the dimension considered in Lemma 2.2 can be exponentially large in n. We will next show the converse of Lemma 2.2: if this dimension is small for a polynomial then there exists a small width ROABP for that polynomial. Hence, this property characterizes the class of polynomials computed by ROABPs. Forbes et al. [FS12b, Section 6] give a similar characterization in terms of evaluation dimension, for polynomials which can be computed by an ROABP, in any variable order. In contrast, we work with a fixed variable order.

As a preparation to prove this characterization we define a *characterizing set of dependencies* of a polynomial $A(\boldsymbol{x})$ of individual degree d, with respect to a variable order $(x_1, x_2, \ldots, x_n)$. This set of dependencies will essentially give us an ROABP for A in the variable order $(x_1, x_2, \ldots, x_n)$.

Definition 2.4. *Let $A(\boldsymbol{x})$ be polynomial of individual degree d, where $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$. For any $0 \leq k \leq n$ and $\boldsymbol{y}_k = (x_1, x_2, \ldots, x_k)$, let $w \geq 1$ such that*

$$\dim_{\mathbb{F}}\{A_{(\boldsymbol{y}_k, \boldsymbol{a})} \mid \boldsymbol{a} \in \{0, 1, \ldots, d\}^k\} \leq w.$$

For $0 \leq k \leq n$, we define the spanning sets $\mathrm{span}_k(A)$ *and the* dependency sets $\mathrm{depend}_k(A)$ *as subsets of $\{0, 1, \ldots, d\}^k$ as follows.*

For $k = 0$, let $\mathrm{depend}_0(A) = \emptyset$ and $\mathrm{span}_0(A) = \{\epsilon\}$, where $\epsilon = (\,)$ denotes the empty tuple. For $k > 0$, let

- $\mathrm{depend}_k(A) = \{(\boldsymbol{a}, j) \mid \boldsymbol{a} \in \mathrm{span}_{k-1}(A)$ *and* $0 \leq j \leq d\}$, *i.e.* $\mathrm{depend}_k(A)$ *contains all possible extensions of the tuples in* $\mathrm{span}_{k-1}(A)$.
- $\mathrm{span}_k(A) \subseteq \mathrm{depend}_k(A)$ *is any set of size $\leq w$, such that for any $\boldsymbol{b} \in \mathrm{depend}_k(A)$, the polynomial $A_{(\boldsymbol{y}_k, \boldsymbol{b})}$ is in the span of $\{A_{(\boldsymbol{y}_k, \boldsymbol{a})} \mid \boldsymbol{a} \in \mathrm{span}_k(A)\}$.*

The dependencies of the polynomials in $\{A_{(\boldsymbol{y}_k, \boldsymbol{a})} \mid \boldsymbol{a} \in \mathrm{depend}_k(A)\}$ over $\{A_{(\boldsymbol{y}_k, \boldsymbol{a})} \mid \boldsymbol{a} \in \mathrm{span}_k(A)\}$ are the characterizing set of dependencies.

The definition of $\mathrm{span}_k(A)$ is not unique. For our purpose, it does not matter which of the possibilities we take, we simply fix one of them.

Note that $|\mathrm{depend}_{k+1}(A)| \leq w(d+1)$ and for $k = n$, we have $\boldsymbol{y}_n = \boldsymbol{x}$ and therefore $A_{(\boldsymbol{y}_n, \boldsymbol{a})} = \mathrm{coeff}_A(\boldsymbol{x}^{\boldsymbol{a}})$ is a constant for every $\boldsymbol{a}$. Hence, the coefficient space has dimension one in this case, and thus $|\mathrm{span}_n(A)| = 1$.

Lemma 2.5 ([Nis91], Converse of Lemma 2.2)**.** *Let $A(\boldsymbol{x})$ be a polynomial of individual degree d with $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ and $w \geq 1$, such that for any $1 \leq k \leq n$ and $\boldsymbol{y}_k = (x_1, x_2, \ldots, x_k)$, we have*

$$\dim_{\mathbb{F}}\{\, A_{(\boldsymbol{y}_k, \boldsymbol{a})} \mid \boldsymbol{a} \in \{0, 1, \ldots, d\}^k \,\} \leq w\,.$$

Then there exists an ROABP *of width w for $A(\boldsymbol{x})$ in the variable order $(x_1, x_2, \ldots, x_n)$.*

Proof. To keep the notation simple, we assume that $|\mathrm{span}_k(A)| = w$ for each $1 \leq k \leq n-1$.[2] Let $\mathrm{span}_k(A) = \{\boldsymbol{a}_{k,1}, \boldsymbol{a}_{k,2}, \ldots, \boldsymbol{a}_{k,w}\}$ and $\mathrm{span}_n(A) = \{\boldsymbol{a}_{n,1}\}$.

To prove the claim, we construct matrices $D_1, D_2, \ldots, D_n$, where $D_1 \in \mathbb{F}[x_1]^{1 \times w}$, $D_n \in \mathbb{F}[x_n]^{w \times 1}$, and $D_i \in \mathbb{F}[x_i]^{w \times w}$, for $i = 2, \ldots, n-1$, such that $A(\boldsymbol{x}) = D_1\, D_2 \cdots D_n$. This representation shows that there is an ROABP of width w for $A(\boldsymbol{x})$.

[2] Otherwise we would have to use a separate value $w_k = |\mathrm{span}_k(A)| \leq w$ for every k.

The matrices are constructed inductively such that for any $k \in [n-1]$,

$$A(\boldsymbol{x}) = D_1 D_2 \cdots D_k \left[A_{(\boldsymbol{y}_k, \boldsymbol{a}_{k,1})} \; A_{(\boldsymbol{y}_k, \boldsymbol{a}_{k,2})} \; \cdots \; A_{(\boldsymbol{y}_k, \boldsymbol{a}_{k,w})} \right]^T. \tag{4}$$

To construct $D_1 \in \mathbb{F}[x_1]^{1 \times w}$, consider the equation

$$A(\boldsymbol{x}) = \sum_{j=0}^{d} A_{(\boldsymbol{y}_1, j)} \, x_1^j. \tag{5}$$

Recall that $\text{depend}_1(A) = \{0, 1, \ldots, d\}$. By the definition of $\text{span}_1(A)$, every $A_{(\boldsymbol{y}_1, j)}$ is in the span of the $A_{(\boldsymbol{y}_1, \boldsymbol{a})}$'s for $\boldsymbol{a} \in \text{span}_1(A)$. That is, there exist constants $\{\gamma_{j,i}\}_{i,j}$ such that for all $0 \le j \le d$, we have

$$A_{(\boldsymbol{y}_1, j)} = \sum_{i=1}^{w} \gamma_{j,i} \, A_{(\boldsymbol{y}_1, \boldsymbol{a}_{1,i})}. \tag{6}$$

From equations (5) and (6) we get, $A(\boldsymbol{x}) = \sum_{i=1}^{w} \left(\sum_{j=0}^{d} \gamma_{j,i} \, x_1^j \right) A_{(\boldsymbol{y}_1, \boldsymbol{a}_{1,i})}$. Hence, we define $D_1 = [D_{1,1} \; D_{1,2} \; \cdots \; D_{1,w}]$, where $D_{1,i} = \sum_{j=0}^{d} \gamma_{j,i} \, x_1^j$, for all $i \in [w]$. Then we have

$$A = D_1 \left[A_{(\boldsymbol{y}_1, \boldsymbol{a}_{1,1})} \; A_{(\boldsymbol{y}_1, \boldsymbol{a}_{1,2})} \; \cdots \; A_{(\boldsymbol{y}_1, \boldsymbol{a}_{1,w})} \right]^T. \tag{7}$$

To construct $D_k \in \mathbb{F}[x_k]^{w \times w}$ for $2 \le k \le n-1$, we consider the equation

$$\left[A_{(\boldsymbol{y}_{k-1}, \boldsymbol{a}_{k-1,1})} \cdots A_{(\boldsymbol{y}_{k-1}, \boldsymbol{a}_{k-1,w})} \right]^T = D_k \left[A_{(\boldsymbol{y}_k, \boldsymbol{a}_{k,1})} \cdots A_{(\boldsymbol{y}_k, \boldsymbol{a}_{k,w})} \right]^T. \tag{8}$$

We know that for each $1 \le i \le w$,

$$A_{(\boldsymbol{y}_{k-1}, \boldsymbol{a}_{k-1,i})} = \sum_{j=0}^{d} A_{(\boldsymbol{y}_k, (\boldsymbol{a}_{k-1,i}, j))} \, x_k^j. \tag{9}$$

Observe that $(\boldsymbol{a}_{k-1,i}, j)$ is just an extension of $\boldsymbol{a}_{k-1,i}$ and thus belongs to $\text{depend}_k(A)$. Hence, there exists a set of constants $\{\gamma_{i,j,h}\}_{i,j,h}$ such that for all $0 \le j \le d$ we have

$$A_{(\boldsymbol{y}_k, (\boldsymbol{a}_{k-1,i}, j))} = \sum_{h=1}^{w} \gamma_{i,j,h} \, A_{(\boldsymbol{y}_k, \boldsymbol{a}_{k,h})}. \tag{10}$$

From equations (9) and (10), for each $1 \le i \le w$ we get

$$A_{(\boldsymbol{y}_{k-1}, \boldsymbol{a}_{k-1,i})} = \sum_{h=1}^{w} \left(\sum_{j=0}^{d} \gamma_{i,j,h} \, x_k^j \right) A_{(\boldsymbol{y}_k, \boldsymbol{a}_{k,h})}.$$

Hence, we can define $D_k(i,h) = \sum_{j=0}^{d} \gamma_{i,j,h} \, x_k^j$, for all $i, h \in [w]$. Then D_k is the desired matrix in equation (8).

Finally, we obtain $D_n \in \mathbb{F}^{w \times 1}[x_n]$ in an analogous way. Instead of equation (8) we consider the equation

$$\left[A_{(\boldsymbol{y}_{n-1}, \boldsymbol{a}_{n-1,1})} \cdots A_{(\boldsymbol{y}_{n-1}, \boldsymbol{a}_{n-1,w})} \right]^T = D'_n \left[A_{(\boldsymbol{y}_n, \boldsymbol{a}_{n,1})} \right]. \tag{11}$$

Recall that $A_{(\boldsymbol{y}_n, \boldsymbol{a}_{n,1})} \in \mathbb{F}$ is a constant that can be absorbed into the last matrix D'_n, i.e. we define $D_n = D'_n \, A_{(\boldsymbol{y}_n, \boldsymbol{a}_{n,1})}$. Combining equations (7), (8), and (11), we get $A(\boldsymbol{x}) = D_1 \, D_2 \cdots D_n$. □

Consider the polynomial P_k defined as the product of the first k matrices $D_1, D_2, \ldots, D_k$ from the above proof, i.e. $P_k(\boldsymbol{y}_k) = D_1 D_2 \cdots D_k$. We can write P_k as

$$P_k(\boldsymbol{y}_k) = \sum_{\boldsymbol{a} \in \{0,1,\ldots,d\}^k} \operatorname{coeff}_{P_k}(\boldsymbol{y}_k^{\boldsymbol{a}})\, \boldsymbol{y}_k^{\boldsymbol{a}}\,,$$

where $\operatorname{coeff}_{P_k}(\boldsymbol{y}_k^{\boldsymbol{a}})$ is a vector in $\mathbb{F}^{1\times w}$. We will see next that it follows from the proof of Lemma 2.5 that the coefficient space of P_k, i.e., $\operatorname{span}_{\mathbb{F}}\{\operatorname{coeff}_{P_k}(\boldsymbol{y}_k^{\boldsymbol{a}}) \mid \boldsymbol{a} \in \{0,1,\ldots,d\}^k\}$ has full rank w.

Corollary 2.6 (Full Rank Coefficient Space)**.** *Let $D_1, D_2, \ldots, D_n$ be the matrices constructed in the proof of Lemma 2.5 with $A = D_1 D_2 \cdots D_n$. For $k \in [n]$, define the polynomial $P_k(\boldsymbol{y}_k) = D_1 D_2 \cdots D_k$ and let $\operatorname{span}_k(A) = \{\boldsymbol{a}_{k,1}, \boldsymbol{a}_{k,2}, \ldots, \boldsymbol{a}_{k,w}\}$.*

Then for any $\ell \in [w]$, we have $\operatorname{coeff}_{P_k}(\boldsymbol{y}_k^{\boldsymbol{a}_{k,\ell}}) = \boldsymbol{e}_\ell$, where $\boldsymbol{e}_\ell$ is the ℓ-th elementary unit vector*, $\boldsymbol{e}_\ell = (0,\ldots,0,1,0,\ldots,0)$ of length w, with a one at position ℓ, and zero at all other positions. Hence, the coefficient space of P_k has full rank w.*

Proof. In the construction of the matrices D_k in the proof of Lemma 2.5, consider the special case in equations (6) and (10) that the exponent $(\boldsymbol{a}_{k-1,i}, j)$ is in $\operatorname{span}_k(A)$, say $(\boldsymbol{a}_{k-1,i}, j) = \boldsymbol{a}_{k,\ell} \in \operatorname{span}_k(A)$. Then the γ-vector to express $A_{(\boldsymbol{y}_k,(\boldsymbol{a}_{k-1,i},j))}$ in equation (6) and (10) can be chosen to be $\boldsymbol{e}_\ell$, i.e. $(\gamma_{i,j,h})_h = \boldsymbol{e}_\ell$. By the definition of matrix D_k, vector $\boldsymbol{e}_\ell$ becomes the i-th row of D_k for the exponent j, i.e., $\operatorname{coeff}_{D_k(i,\cdot)}(x_k^j) = \boldsymbol{e}_\ell$.

This shows the claim for $k = 1$, because $\operatorname{coeff}_{P_1}(x_1^{\boldsymbol{a}_{1,\ell}}) = \operatorname{coeff}_{D_1}(x_1^{\ell-1}) = \boldsymbol{e}_\ell$.

For larger k, it follows by induction because for $(\boldsymbol{a}_{k-1,i}, j) = \boldsymbol{a}_{k,\ell}$, we have $\operatorname{coeff}_{P_k}(\boldsymbol{y}_k^{\boldsymbol{a}_{k,\ell}}) = \operatorname{coeff}_{P_{k-1}}(\boldsymbol{y}_{k-1}^{\boldsymbol{a}_{k-1,i}}) \operatorname{coeff}_{D_k}(x_k^j)$. By the induction hypothesis, we have $\operatorname{coeff}_{P_{k-1}}(\boldsymbol{y}_{k-1}^{\boldsymbol{a}_{k-1,i}}) = \boldsymbol{e}_i$. The product of $\boldsymbol{e}_i$ with the $w \times w$-matrix $\operatorname{coeff}_{D_k}(x_k^j)$ picks the i-th row of the matrix which is $\boldsymbol{e}_\ell$ as explained above. Hence, $\operatorname{coeff}_{P_k}(\boldsymbol{y}_k^{\boldsymbol{a}_{k,\ell}}) = \boldsymbol{e}_\ell$ as claimed. □

2.4 Boolean branching programs

The definition of a *boolean branching program* is similar to that of an ABP. It is a directed graph with layers of vertices $(V_0, V_1, \ldots, V_\ell)$, where layer V_0 contains the start node v_0 and layer V_ℓ contains an *accepting node* v_a and a *rejecting node* v_r. The nodes in $V_0, V_1, \ldots, V_{\ell-1}$ are called the *inner nodes*. The edges are allowed only between the consecutive layers of the boolean branching program.

Every inner node is labeled by a variable. In a *deterministic* boolean branching program, every inner node has outdegree two: there is an edge labeled 0 and an edge labeled 1. In a *nondeterministic* boolean branching program, the inner nodes can have arbitrarily many 0- and 1-edges.

On a given input $\boldsymbol{a} = (a_1, a_2, \ldots, a_n) \in \{0,1\}^n$, we start the evaluation at the start node. When we reach a node labeled x_i, we continue with an a_i-edge of this node. The input $\boldsymbol{a}$ is accepted, if we can reach the accepting node v_a. In the *parity model* introduced by [GM96], we have a nondeterministic branching program that accepts the input $\boldsymbol{a}$ if there is an odd number of accepting paths.

A branching program is called *oblivious*, if the nodes in every layer are labeled by the same variable, and every variable occurs in at most one layer. A common name for the deterministic model is *oblivious binary decision diagram* (OBDD). A *parity-OBDD* ($\oplus$-OBDD) is

a nondeterministic oblivious branching program with a parity acceptance mechanism. Note that $\oplus$-OBDD is a stronger model, as an OBDD is also a $\oplus$-OBDD.

There is an obvious way to *arithmetize* a boolean branching program P: if a node in P is labeled by a variable x, then, instead of labeling the node, we label the 1-edge with polynomial x, and the 0-edge with polynomial $1 - x$. Also, we delete the rejecting node from P. Then the polynomial $A(\boldsymbol{x})$ computed by the resulting arithmetic branching program agrees with $P(\boldsymbol{x})$ on $\{0,1\}^n$. A parity branching program can be arithmetized similarly. In this case, $A(\boldsymbol{x})$ has to be considered as a polynomial over GF[2].

Note that in case of an OBDD, its arithmetization will provide an ROABP that computes a multilinear polynomial. Similarly, the arithmetization of a $\oplus$-OBDD yields an ROABP of a multilinear polynomial over GF[2]. Hence, our results will apply to these models (see Corollary 3.3).

3 Whitebox Identity Testing

We will use the characterization of ROABPs provided by Lemmas 2.2 and 2.5 in Section 3.1 to design a polynomial-time algorithm to check if two given ROABPs are equivalent. This is the same problem as checking whether the sum of two ROABPs is zero. In Section 3.2, we extend the test to check whether the sum of constantly many ROABPs is zero.

3.1 Equivalence of two ROABPs

Let $A(\boldsymbol{x})$ and $B(\boldsymbol{x})$ be two polynomials of individual degree d, given by two ROABPs. If the two ROABPs have the same variable order then one can combine them into a single ROABP which computes their difference. Then one can apply the PIT for one ROABP [RS05]. So, the problem is non-trivial only when the two ROABPs have different variable order. W.l.o.g. we assume that A has order $(x_1, x_2, \ldots, x_n)$. Let w bound the width of both ROABPs. In this section we prove that we can find out in polynomial time whether $A(\boldsymbol{x}) = B(\boldsymbol{x})$.

Theorem 3.1. *The equivalence of two ROABPs can be tested in polynomial time in n, d, w, the number of variables, the individual degree, and the width, respectively.*

The idea is to determine the characterizing set of dependencies among the partial derivative polynomials of A, and verify that the same dependencies hold for the corresponding partial derivative polynomials of B. By Lemma 2.5, these dependencies essentially define an ROABP. Hence, our algorithm is to construct an ROABP for B in the variable order of A. Then it suffices to check whether we get the same ROABP, that is, whether all the matrices $D_1, D_2, \ldots, D_n$ constructed in the proof of Lemma 2.5 are the same for A and B. We give some more details.

Construction of $\mathrm{span}_k(A)$. Let $A(\boldsymbol{x}) = D_1(x_1)D_2(x_2)\cdots D_n(x_n)$ of width w. We give an iterative construction, starting from $\mathrm{span}_0(A) = \{\epsilon\}$. Let $1 \le k \le n$. By definition, $\mathrm{depend}_k(A)$ consists of all possible one-step extensions of $\mathrm{span}_{k-1}(A)$. Let $\boldsymbol{b} = (b_1, b_2, \ldots, b_k) \in \{0, 1, \ldots, d\}^k$. Define

$$C_{\boldsymbol{b}} = \prod_{i=1}^{k} \mathrm{coeff}_{D_i}(x_i^{b_i}) \,.$$

Recall that $\text{coeff}_{D_1}(x_1^{b_1}) \in \mathbb{F}^{1\times w}$ and $\text{coeff}_{D_i}(x_i^{b_i}) \in \mathbb{F}^{w\times w}$, for $2 \le i \le k$. Therefore $C_{\boldsymbol{b}} \in \mathbb{F}^{1\times w}$ for $k < n$. Since $D_n \in \mathbb{F}^{w\times 1}$, we have $C_{\boldsymbol{b}} \in \mathbb{F}$ for $k = n$. By equation (3), we have

$$A_{(\boldsymbol{y}_k,\boldsymbol{b})} = C_{\boldsymbol{b}}\, D_{k+1} \cdots D_n\,. \tag{12}$$

Consider the set of vectors $\mathcal{D}_k = \{C_{\boldsymbol{b}} \mid \boldsymbol{b} \in \text{depend}_k(A)\}$. This set has dimension bounded by w since the width of A is w. Hence, we can determine a set $\mathcal{S}_k \subseteq \mathcal{D}_k$ of size $\le w$ such that $\mathcal{S}_k$ spans $\mathcal{D}_k$. Thus we can take $\text{span}_k(A) = \{\boldsymbol{a} \mid C_{\boldsymbol{a}} \in \mathcal{S}_k\}$. Then, for any $\boldsymbol{b} \in \text{depend}_k(A)$, vector $C_{\boldsymbol{b}}$ is a linear combination

$$C_{\boldsymbol{b}} = \sum_{\boldsymbol{a}\in \text{span}_k(A)} \gamma_{\boldsymbol{b},\boldsymbol{a}}\, C_{\boldsymbol{a}}\,.$$

Recall that $|\text{depend}_k(A)| \le w(d+1)$, i.e. this is a small set. Therefore, we can efficiently compute the coefficients $\gamma_{\boldsymbol{b},\boldsymbol{a}}$ for every $\boldsymbol{b} \in \text{depend}_k(A)$. Note that by equation (12) we have the same dependencies for the polynomials $A_{(\boldsymbol{y}_k,\boldsymbol{b})}$. That is, with the same coefficients $\gamma_{\boldsymbol{b},\boldsymbol{a}}$, we can write

$$A_{(\boldsymbol{y}_k,\boldsymbol{b})} = \sum_{\boldsymbol{a}\in \text{span}_k(A)} \gamma_{\boldsymbol{b},\boldsymbol{a}}\, A_{(\boldsymbol{y}_k,\boldsymbol{a})}\,. \tag{13}$$

Verifying the dependencies for B. We want to verify that the dependencies in equation (13) computed for A hold for B as well, i.e. that for all $k \in [n]$ and $\boldsymbol{b} \in \text{depend}_k(A)$,

$$B_{(\boldsymbol{y}_k,\boldsymbol{b})} = \sum_{\boldsymbol{a}\in \text{span}_k(A)} \gamma_{\boldsymbol{b},\boldsymbol{a}}\, B_{(\boldsymbol{y}_k,\boldsymbol{a})}\,. \tag{14}$$

Recall that $\boldsymbol{y}_k = (x_1, x_2, \ldots, x_k)$ and the ROABP for B has a different variable order. By Lemma 2.3, every polynomial $B_{(\boldsymbol{y}_k,\boldsymbol{a})}$ has an ROABP of width w and the same order on the remaining variables as the one given for B. It follows that each of the $w+1$ polynomials that occur in equation (14) has an ROABP of width w and the same variable order. Hence, we can construct *one* ROABP for the polynomial

$$B_{(\boldsymbol{y}_k,\boldsymbol{b})} - \sum_{\boldsymbol{a}\in \text{span}_k(A)} \gamma_{\boldsymbol{b},\boldsymbol{a}} B_{(\boldsymbol{y}_k,\boldsymbol{a})}\,. \tag{15}$$

Simply identify all the start nodes and all the end nodes and put the appropriate constants $\gamma_{\boldsymbol{b},\boldsymbol{a}}$ to the weights. Then we get an ROABP of width $w(w+1)$. In order to verify equation (14), it suffices to do a zero-test for this ROABP. This can be done in polynomial time [RS05].

Constructing ROABP for B in the same sequence as A. Recall Lemma 2.5 and its proof. There, we constructed an ROABP just from the characterizing dependencies of the given polynomial. Hence, the construction applied to B will give an ROABP of width w for B with the same variable order $(x_1, x_2, \ldots, x_n)$ as for A. The matrices D_k will be the same as those for A because their definition uses only the dependencies provided by equation (14), and they are the same as those for A in equation (13).

The last matrix D_n can be written as $D'_n A_{(\boldsymbol{y}_n,\boldsymbol{a}_{n,1})}$, similar to equation (11). Since the dependencies of the coefficients in $\text{depend}_n(B)$ over coefficients in $\text{span}_n(B)$ are the same as those for A, we have $B(\boldsymbol{x}) = D_1 D_2 \cdots D'_n\, B_{(\boldsymbol{y}_n,\boldsymbol{a}_{n,1})}$.

Checking Equality. Clearly, if equation (14) fails to hold for some k and $\boldsymbol{b}$, then $A \neq B$. When equation (14) holds for all k and $\boldsymbol{b}$, we only need to check if $A_{(\boldsymbol{y}_n, \boldsymbol{a}_{n,1})} = B_{(\boldsymbol{y}_n, \boldsymbol{a}_{n,1})}$, which is a single evaluation of each ROABP.

The following pseudo-code summarizes the equivalence test.

EQUIVALENCE(A, B)

$*$ **input:** Two ROABPs computing polynomials $A(\boldsymbol{x})$ and $B(\boldsymbol{x})$, respectively.

$*$ The ROABP for A has variable order $(x_1, x_2, \ldots, x_n)$.

1 $\text{span}_0(A) \leftarrow \{\epsilon\}$

2 **for** $k \leftarrow 1$ **to** n **do**

3 $\quad \text{depend}_k(A) \leftarrow \text{span}_{k-1}(A) \times \{0, 1, \ldots, d\}$

4 $\quad$ **for each** $\boldsymbol{b} \in \text{depend}_k(A)$ **do** $C_{\boldsymbol{b}} \leftarrow \prod_{i=1}^{k} \text{coeff}_{D_i}(x_i^{b_i})$

5 $\quad$ compute a basis $\mathcal{S}_k \subseteq \mathcal{D}_k = \{C_{\boldsymbol{b}} \mid \boldsymbol{b} \in \text{depend}_k(A)\}$

6 $\quad \text{span}_k(A) \leftarrow \{\boldsymbol{a} \mid C_{\boldsymbol{a}} \in \mathcal{S}_k\}$

7 $\quad$ **for each** $\boldsymbol{b} \in \text{depend}_k(A)$ **do**

8 $\quad\quad$ determine coefficients $\{\gamma_{\boldsymbol{b},\boldsymbol{a}}\}_{\boldsymbol{a} \in \text{span}_k(A)}$ such that $C_{\boldsymbol{b}} = \sum_{\boldsymbol{a} \in \text{span}_k(A)} \gamma_{\boldsymbol{b},\boldsymbol{a}} C_{\boldsymbol{a}}$,

9 $\quad\quad$ **if** $B_{(\boldsymbol{y}_k, \boldsymbol{b})} \neq \sum_{\boldsymbol{a} \in \text{span}_k(A)} \gamma_{\boldsymbol{b},\boldsymbol{a}} B_{(\boldsymbol{y}_k, \boldsymbol{a})}$ **then output** '$A \neq B$'

10 **if** $B_{(\boldsymbol{y}_n, \boldsymbol{a}_{n,1})} = A_{(\boldsymbol{y}_n, \boldsymbol{a}_{n,1})}$ **then output** '$A = B$' **else output** '$A \neq B$'

The verification of the condition in line 9 involves the construction of ROABPs and a zero-test as discussed above in the verification paragraph.

Time Complexity. In the construction part, it takes $O(ndw^3)$ steps to compute all the coefficients $C_{\boldsymbol{b}}$ and $\gamma_{\boldsymbol{b},\boldsymbol{a}}$ in line 4 and lines 5 - 8. The running time is dominated by the zero-tests for the ROABPs in line 9. Note that one can easily derive a zero-test for an ROABP from our algorithm by choosing $B = 0$. The running time we get is that of the construction part. However, the ROABPs we get in line 9 have width $w(w+1)$. Therefore, the zero-test takes $O(ndw^6)$ steps. There are ndw such tests. Hence, we get a total running time of $O(n^2d^2w^7)$. This proves Theorem 3.1.

3.2 Sum of constantly many ROABPs

Let $A_1(\boldsymbol{x}), A_2(\boldsymbol{x}), \ldots, A_c(\boldsymbol{x})$ be polynomials of individual degree d, given by c ROABPs. Our goal is to test whether $A_1 + A_2 + \cdots + A_c = 0$. Here again, the question is interesting only when the ROABPs have different variable orders. We show how to reduce the problem to the case of the equivalence of two ROABPs from the previous section. For constant c this will lead to a polynomial-time test.

We start by rephrasing the problem as an equivalence test. Let $A = -A_1$ and $B = A_2 + A_3 + \cdots + A_c$. Then the problem has become to check whether $A = B$. Since A is computed by a single ROABP, we can use the same approach as in Section 3.1. Hence, we again get the dependencies from equation (13) for A. Next, we have to verify these dependencies for B, i.e. equation (14). Now, B is not given by a single ROABP, but is a sum of $c-1$ ROABPs. For every $k \in [n]$ and $\boldsymbol{b} \in \text{depend}_k(A)$, define the polynomial

$Q = B_{(\boldsymbol{y}_k,\boldsymbol{b})} - \sum_{\boldsymbol{a}\in \mathrm{span}_k(A)} \gamma_{\boldsymbol{b},\boldsymbol{a}} B_{(\boldsymbol{y}_k,\boldsymbol{a})}$. By the definition of B we have

$$Q = \sum_{i=2}^{c} \left(A_{i(\boldsymbol{y}_k,\boldsymbol{b})} - \sum_{\boldsymbol{a}\in \mathrm{span}_k(A)} \gamma_{\boldsymbol{b},\boldsymbol{a}} A_{i(\boldsymbol{y}_k,\boldsymbol{a})} \right). \tag{16}$$

As explained in the previous section for equation (15), for each summand in equation (16) we can construct an ROABP of width $w(w+1)$. Thus, Q can be written as a sum of $c-1$ ROABPs, each having width $w(w+1)$. To test whether $Q = 0$, we recursively use the same algorithm for the sum of $c-1$ ROABPs. The recursion ends when $c = 2$. Then we directly use the algorithm from Section 3.1.

To bound the running time of the algorithm, let us see how many dependencies we need to verify. There is one dependency for every $k \in [n]$ and every $\boldsymbol{b} \in \mathrm{depend}_k(A)$. Since $|\mathrm{depend}_k(A)| \leq w(d+1)$, the total number of dependencies verified is $\leq nw(d+1)$. Thus, we get the following recursive formula for $T(c,w)$, the time complexity for testing zeroness of the sum of $c \geq 2$ ROABPs, each having width w. For $c = 2$, we have $T(2,w) = \mathsf{poly}(n,d,w)$, and for $c > 2$,

$$T(c,w) = nw(d+1) \cdot T(c-1, w(w+1)) + \mathsf{poly}(n,d,w).$$

As solution, we get $T(c,w) = w^{O(2^c)}\mathsf{poly}(n^c, d^c)$, i.e. polynomial time for constant c.

Theorem 3.2. *Let $A(\boldsymbol{x})$ be an n-variate polynomial of individual degree d, computed by a sum of c* ROABPs *of width w. Then there is a PIT for $A(\boldsymbol{x})$ that works in time $w^{O(2^c)}(nd)^{O(c)}$.*

We apply our results to boolean branching programs. The equivalence of two OBDDs or, more generally, two $\oplus$-OBDDs can be checked in polynomial time [SW97, BW98]. This gives a zero-test for the xor of two $\oplus$-OBDDs. As explained in Section 2.4, the arithmetization of $\oplus$-OBDDs yields ROABPs over GF[2]. Hence, we conclude from Theorem 3.2 that we can test in polynomial time if an xor of constantly many $\oplus$-OBDDs is the boolean zero-function .

Corollary 3.3. *Let $A(\boldsymbol{x})$ be an n-variate boolean function computed by an xor of c $\oplus$-OBDDs of width w. Then one can test if $A(\boldsymbol{x})$ is the zero-function in time $w^{O(2^c)}n^{O(c)}$.*

4 Blackbox Identity Testing

In this section, we extend the blackbox PIT of Agrawal et. al [AGKS15] for one ROABP to the sum of constantly many ROABPs. In the blackbox model we are only allowed to evaluate a polynomial at various points. Hence, for PIT, our task is to construct a hitting-set.

Definition 4.1. *A set $H \subseteq \mathbb{F}^n$ is a* hitting-set *for a class $\mathcal{P}$ of n-variate polynomials, if for every nonzero polynomial $A(\boldsymbol{x}) \in \mathcal{P}$, there is a point $\boldsymbol{a} \in H$ such that $A(\boldsymbol{a}) \neq 0$.*

For a hitting-set to exist, we will need enough points in the underlying field $\mathbb{F}$. Henceforth, we will assume that the field $\mathbb{F}$ is large enough such that the constructions below go through (see [AL86] for constructing large $\mathbb{F}$). To construct a hitting-set for a sum of ROABPs we use the concept of *low support rank concentration* defined by Agrawal, Saha, and Saxena [ASS13]. A polynomial $A(\boldsymbol{x})$ has low support concentration if the coefficients of its monomials of low support span the coefficients of all the monomials.

Definition 4.2 ([ASS13])**.** *A polynomial $A(\boldsymbol{x})$ has* ℓ-support concentration *or, for short, is* ℓ-concentrated*, if for all monomials $\boldsymbol{x}^{\boldsymbol{a}}$ of $A(\boldsymbol{x})$, we have,*

$$\operatorname{coeff}_A(\boldsymbol{x}^{\boldsymbol{a}}) \in \operatorname{span}_{\mathbb{F}}\{\operatorname{coeff}_A(\boldsymbol{x}^{\boldsymbol{b}}) \mid \operatorname{supp}(\boldsymbol{b}) < \ell\}.$$

The above definition applies to polynomials over any $\mathbb{F}$-vector space, e.g. $\mathbb{F}[\boldsymbol{x}]$, $\mathbb{F}^w[\boldsymbol{x}]$ or $\mathbb{F}^{w\times w}[\boldsymbol{x}]$. For $A(\boldsymbol{x}) \in \mathbb{F}[\boldsymbol{x}]$, its coefficients form a one-dimensional vector space over $\mathbb{F}$. Thus, it has ℓ-support concentration if and only if there are nonzero coefficients of support $< \ell$. An ℓ-concentrated polynomial in $\mathbb{F}(\boldsymbol{x})$ has the following hitting set.

Lemma 4.3 ([ASS13])**.** *For n, d, ℓ, the set $H = \{\boldsymbol{h} \in \{0, \beta_1, \ldots, \beta_d\}^n \mid \operatorname{supp}(\boldsymbol{h}) < \ell\}$ of size $O(nd)^\ell$ is a hitting-set for all n-variate ℓ-concentrated polynomials $A(\boldsymbol{x}) \in \mathbb{F}[\boldsymbol{x}]$ of individual degree d, where $\{\beta_i\}_i$ are distinct nonzero elements in $\mathbb{F}$.*

Hence, when we have low support concentration, this solves blackbox PIT. Note that not every polynomial has low support concentration, for example $A(\boldsymbol{x}) = x_1 x_2 \cdots x_n$ is not n-concentrated. However, Agrawal, Saha, and Saxena [ASS13] showed that low support concentration can be achieved through an appropriate shift of the variables.

Definition 4.4. *Let $A(\boldsymbol{x})$ be an n-variate polynomial and $\boldsymbol{f} = (f_1, f_2, \ldots, f_n) \in \mathbb{F}^n$. The polynomial A* shifted by $\boldsymbol{f}$ *is $A(\boldsymbol{x} + \boldsymbol{f}) = A(x_1 + f_1, x_2 + f_2, \ldots, x_n + f_n)$.*

A shift preserves the coefficient space of a polynomial.

Lemma 4.5. *Let $A(\boldsymbol{x})$ be an n-variate polynomial and $\boldsymbol{f} = (f_1, f_2, \ldots, f_n) \in \mathbb{F}^n$. Then $A(\boldsymbol{x})$ and $A(\boldsymbol{x} + \boldsymbol{f})$ have the same coefficient space.*

Proof. We show that the coefficients of $A(\boldsymbol{x}+\boldsymbol{f})$ are in the span of the coefficients of $A(\boldsymbol{x}+\boldsymbol{f})$ Recall that $M = \{0, 1, \ldots, d\}^n$ denotes the set of all exponents of monomials in $\boldsymbol{x}$ of individual degree bounded by d.

For $A(\boldsymbol{x}) = \sum_{\boldsymbol{a}\in M} c_{\boldsymbol{a}} \boldsymbol{x}^{\boldsymbol{a}}$, we have $A'(\boldsymbol{x}) = A(\boldsymbol{x}+\boldsymbol{f}) = \sum_{\boldsymbol{a}\in M} c_{\boldsymbol{a}} (\boldsymbol{x}+\boldsymbol{f})^{\boldsymbol{a}}$. Expand all the expressions $(\boldsymbol{x}+\boldsymbol{f})^{\boldsymbol{a}}$ in the latter sum and resort according to $\boldsymbol{x}^{\boldsymbol{b}}$, for every $\boldsymbol{b} \in M$. Then we get

$$\operatorname{coeff}_{A'}(\boldsymbol{x}^{\boldsymbol{b}}) = \sum_{\boldsymbol{a}\in M} \binom{\boldsymbol{a}}{\boldsymbol{b}} \boldsymbol{f}^{(\boldsymbol{b}-\boldsymbol{a})} \cdot c_{\boldsymbol{a}}, \tag{17}$$

where $\binom{\boldsymbol{b}}{\boldsymbol{a}} = \prod_{i=1}^n \binom{b_i}{a_i}$ for any $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{N}^n$. Hence, the coefficient of $\boldsymbol{x}^{\boldsymbol{b}}$ in $A(\boldsymbol{x}+\boldsymbol{f})$ is a linear combination of the coefficients $c_{\boldsymbol{a}}$.

Since a shift is an invertible process, we conclude that also the coefficients $c_{\boldsymbol{a}}$ are in the span of the coefficients of $A(\boldsymbol{x}+\boldsymbol{f})$. □

In the above example, we shift every variable by 1. That is, we consider $A(\boldsymbol{x}+\boldsymbol{1}) = (x_1+1)(x_2+1)\cdots(x_n+1)$. Observe that $A(\boldsymbol{x}+\boldsymbol{1})$ has 1-support concentration. A polynomial $A(\boldsymbol{x})$ can also be shifted by polynomials. Then, $\boldsymbol{f}$ would be a tuple of n polynomials. Agrawal, Saha, and Saxena [ASS13] provide an efficient shift that achieves low support concentration for polynomials computed by *set-multilinear depth-3 circuits*. Here, a shift is efficient if $\boldsymbol{f}$ itself can be computed in quasi-polynomial time and $A(\boldsymbol{x}+\boldsymbol{f})$ has a hitting set that is computable in quasi-polynomial time. Forbes, Saptharishi and Shpilka [FSS14] extended their result to polynomials computed by ROABPs. However their cost is exponential in the individual degree of the polynomial.

For our purposes, any efficient shift that achieves low support concentration for ROABPs will suffice. In Section 5, we will give a new shift for ROABPs with quasi-polynomial cost. Namely, in Theorem 5.6 below (on page 25), we present a shift polynomial $\boldsymbol{f}(t) \in \mathbb{F}[t]^n$ in one variable t of degree $(ndw)^{O(\log n)}$ that can be computed in time $(ndw)^{O(\log n)}$. It has the property that for every n-variate polynomial $A(\boldsymbol{x}) \in \mathbb{F}^{w\times w}[\boldsymbol{x}]$ of individual degree d that can be computed by an ROABP of width w, the shifted polynomial $A(\boldsymbol{x} + \boldsymbol{f}(t))$ has $O(\log w)$-concentration, where the coefficient space of $A(\boldsymbol{x} + \boldsymbol{f}(t))$ is considered over $\mathbb{F}(t)$. Finally to get a shift in $\mathbb{F}$, we can plug in as many values for $t \in \mathbb{F}$ as the degree of $\boldsymbol{f}(t)$, i.e. $(ndw)^{O(\log n)}$ many. For at least one value of t, the shift $\boldsymbol{f}(t)$ will $O(\log w)$-concentrate $A(\boldsymbol{x}+\boldsymbol{f}(t))$. That is, we consider $\boldsymbol{f}(t)$ as a family of shifts. The same shift also works when the ROABP computes a polynomial in $\mathbb{F}[\boldsymbol{x}]$ or $\mathbb{F}^{1\times w}[\boldsymbol{x}]$.

The rest of the paper is organized as follows. The construction of a shift to obtain low support concentration for single ROABPs is postponed to Section 5. We start in Section 4.1 to show how the shift for a single ROABP can be applied to obtain a shift for the sum of constantly many ROABPs.

4.1 Sum of ROABPs

We will first give a hitting set for the sum of two ROABPs, $A + B$. We will then extend this result for the sum of c ROABPs. Let $A \in \mathbb{F}[\boldsymbol{x}]$ be a polynomial of individual degree d that has an ROABP of width w, with variable order $(x_1, x_2, \ldots, x_n)$. Let $B \in \mathbb{F}[\boldsymbol{x}]$ be another polynomial. We start by reconsidering the whitebox test from the previous section. The dependency equations (13) and (14) were used to construct an ROABP for $B \in \mathbb{F}[\boldsymbol{x}]$ in the same variable order as for A, and the same width. If this succeeds, then the polynomial $A + B$ has one ROABP of width $2w$. Since there is already a blackbox PIT for one ROABP [AGKS15], we are done in this case.

Hence, the interesting case that remains is when the dependency equations (13) for A do not carry over to B as in equation (14). Let $k \in [n]$ be the first such index. In the following Lemma 4.6 we decompose A and B into a common ROABP R up to layer k, and the remaining different parts P and Q. That is, for $\boldsymbol{y}_k = (x_1, x_2, \ldots, x_k)$ and $\boldsymbol{z}_k = (x_{k+1}, \ldots, x_n)$, we obtain $A = RP$ and $B = RQ$, where $R \in \mathbb{F}[\boldsymbol{y}_k]^{1\times w'}$ and $P, Q \in \mathbb{F}[\boldsymbol{z}_k]^{w'\times 1}$, for some $w' \leq w(d + 1)$. The construction we give is such that that the coefficient space of R has full rank w'. Since the dependency equations (13) for A do not fulfill equation (14) for B, we get a constant vector $\Gamma \in \mathbb{F}^{1\times w'}$ such that $\Gamma P = 0$ but $\Gamma Q \neq 0$. From these properties, we will see in Lemma 4.7 below that we get low support concentration for $A + B$ when we use the shift constructed in Section 5 for one ROABP.

Lemma 4.6 (Common ROABP R)**.** *Let $A(\boldsymbol{x})$ be a polynomial of individual degree d, computed by an ROABP of width w in variable order $(x_1, x_2, \ldots, x_n)$. Let $B(\boldsymbol{x})$ be another polynomial for which there does* not *exist an ROABP of width w in the same variable order.*

Then there exists $k \in [n]$ such that for some $w' \leq w(d + 1)$, there are polynomials $R \in \mathbb{F}[\boldsymbol{y}_k]^{1\times w'}$ and $P, Q \in \mathbb{F}[\boldsymbol{z}_k]^{w'\times 1}$, such that

1. *$A = RP$ and $B = RQ$,*
2. *there exists a vector $\Gamma \in \mathbb{F}^{1\times w'}$ with $\operatorname{supp}(\Gamma) \leq w + 1$ such that $\Gamma P = 0$ and $\Gamma Q \neq 0$,*
3. *the coefficient space of R has full rank w'.*

Proof. Let $D_1, D_2, \ldots, D_n$ be the matrices constructed in Lemma 2.5 for A. Assume again w.l.o.g. that $\mathrm{span}_k(A) = \{\boldsymbol{a}_{k,1}, \boldsymbol{a}_{k,2}, \ldots, \boldsymbol{a}_{k,w}\}$ has size w for each $1 \leq k \leq n-1$, and $\mathrm{span}_n(A) = \{\boldsymbol{a}_{n,1}\}$. Then we have $D_1 \in \mathbb{F}^{1\times w}[x_1]$, $D_n \in \mathbb{F}^{w\times 1}[x_n]$ and $D_i \in \mathbb{F}^{w\times w}[x_i]$, for $2 \leq i \leq n-1$.

In the proof of Lemma 2.5 we consider the dependency equations for A and carry them over to B. By the assumption of the lemma, there is no ROABP of width w for B now. Therefore there is a smallest $k \in [n]$ where a dependency for A is not followed by B. That is, the coefficients $\gamma_{\boldsymbol{a}}$ computed for equation (13) do not fulfill equation (14) for B. Since the dependencies carry over up to this point, the construction of the matrices $D_1, D_2, \ldots, D_{k-1}$ work out fine for B. Hence, by equation (4), we can write

$$A(\boldsymbol{x}) = D_1\, D_2 \cdots D_{k-1}\, [A_{(\boldsymbol{y}_{k-1},\boldsymbol{a}_{k-1,1})}\, A_{(\boldsymbol{y}_{k-1},\boldsymbol{a}_{k-1,2})} \cdots A_{(\boldsymbol{y}_{k-1},\boldsymbol{a}_{k-1,w})}]^T \tag{18}$$

$$B(\boldsymbol{x}) = D_1\, D_2 \cdots D_{k-1}\, [B_{(\boldsymbol{y}_{k-1},\boldsymbol{a}_{k-1,1})}\, B_{(\boldsymbol{y}_{k-1},\boldsymbol{a}_{k-1,2})} \cdots B_{(\boldsymbol{y}_{k-1},\boldsymbol{a}_{k-1,w})}]^T \tag{19}$$

Since the difference between A and B occurs at x_k, we consider all possible extensions from x_{k-1}. That is, by equation (9), for every $i \in [w]$ we have

$$A_{(\boldsymbol{y}_{k-1},\boldsymbol{a}_{k-1,i})} = \sum_{j=0}^{d} A_{(\boldsymbol{y}_k,(\boldsymbol{a}_{k-1,i},j))} x_k^j\,. \tag{20}$$

Recall that our goal is to decompose polynomial A into $A = RP$. We first define polynomial P as the vector of coefficient polynomials of all the one-step extensions of $\mathrm{span}_{k-1}(A)$, i.e., $P = \left(A_{(\boldsymbol{y}_k,(\boldsymbol{a}_{k-1,i},j))}\right)_{1\leq i\leq w,\ 0\leq j\leq d}$ is of length $w' = w(d+1)$. Written explicitly, this is

$$P = [A_{(\boldsymbol{y}_k,(\boldsymbol{a}_{k-1,1},0))} \cdots A_{(\boldsymbol{y}_k,(\boldsymbol{a}_{k-1,1},d))} \ \cdots\ A_{(\boldsymbol{y}_k,(\boldsymbol{a}_{k-1,w},0))} \cdots A_{(\boldsymbol{y}_k,(\boldsymbol{a}_{k-1,w},d))}]^T\,.$$

To define $R \in \mathbb{F}[\boldsymbol{y}_k]^{1\times w'}$, let I_w be the $w \times w$ identity matrix. Define matrix $E_k \in \mathbb{F}[x_k]^{w\times w'}$ as the tensor product

$$E_k = I_w \otimes \begin{bmatrix} x_k^0 & x_k^1 & \cdots & x_k^d \end{bmatrix}\,.$$

From equation (20) we get that

$$[A_{(\boldsymbol{y}_{k-1},\boldsymbol{a}_{k-1,1})} \cdots A_{(\boldsymbol{y}_{k-1},\boldsymbol{a}_{k-1,w})}]^T = E_k\, P.$$

Thus, equation (18) can be written as $A(\boldsymbol{x}) = D_1\, D_2 \cdots D_{k-1} E_k P$. Hence, when we define

$$R(\boldsymbol{y}_k) = D_1\, D_2 \cdots D_{k-1} E_k$$

then we have $A = RP$ as desired. By an analogous argument we get $B = RQ$ for $Q = \left(B_{(\boldsymbol{y}_k,(\boldsymbol{a}_{k-1,i},j))}\right)_{1\leq i\leq w,\ 0\leq j\leq d}$.

For the second claim of the lemma let $\boldsymbol{b} \in \mathrm{depend}_k(A)$ such that the dependency equation (13) for A is fulfilled, but not equation (14) for B. Define $\Gamma \in \mathbb{F}^{1\times w'}$ to be the vector that has the values $\gamma_{\boldsymbol{a}}$ used in equation (13) at the position where P has entry $A_{(\boldsymbol{y}_k,\boldsymbol{a})}$, and zero at all other positions. Then $\mathrm{supp}(\Gamma) \leq w+1$ and we have $\Gamma P = 0$ and $\Gamma Q \neq 0$.

It remains to show that the coefficient space of R has full rank. By Corollary 2.6, the coefficient space of $D_1\, D_2 \cdots D_{k-1}$ has full rank w. Namely, for any $\ell \in [w]$, the coefficient

of the monomial $\boldsymbol{y}_{k-1}^{\boldsymbol{a}_{k-1,\ell}}$ is $\boldsymbol{e}_\ell$, the ℓ-th standard unit vector. Therefore the coefficient of $R(\boldsymbol{y}_k) = D_1\, D_2 \cdots D_{k-1} E_k$ at monomial $\boldsymbol{y}_k^{(\boldsymbol{a}_{k-1,\ell,j})}$ is

$$\operatorname{coeff}_R(\boldsymbol{y}_k^{\boldsymbol{a}_{k-1,\ell,j}}) = \boldsymbol{e}_\ell \operatorname{coeff}_{E_k}(x_k^j),$$

for $1 \le \ell \le w$ and $0 \le j \le d$. By the definition of E_k, we get $\operatorname{coeff}_R(\boldsymbol{y}_k^{\boldsymbol{a}_{k-1,\ell,j}}) = \boldsymbol{e}_{(\ell-1)(d+1)+j+1}$. Thus, the coefficient space of R has full rank w'. □

Lemma 4.6 provides the technical tool to obtain low support concentration for the sum of several ROABPs by the shift developed for a single ROABP. We start with the case of the sum of two ROABPs.

Lemma 4.7. *Let $A(\boldsymbol{x})$ and $B(\boldsymbol{x})$ be two n-variate polynomials of individual degree d, each computed by an* ROABP *of width w. Define $W_{w,2} = (d+1)(2w)^2$ and $\ell_{w,2} = \log(W_{w,2}^2 + 1)$. Let $\boldsymbol{f}_{w,2}(t) \in \mathbb{F}[t]^n$ be a shift that $\ell_{w,2}$-concentrates any polynomial (or matrix polynomial) that is computed by an* ROABP *of width $\le W_{w,2}$.*

Then $(A+B)' = (A+B)(\boldsymbol{x} + \boldsymbol{f}_{w,2})$ is $2\,\ell_{w,2}$-concentrated.

Proof. If B can be computed by an ROABP of width w in the same variable order as the one for A, then there is an ROABP of width $2w$ that computes $A+B$. In this case, the lemma follows because $2w \le W_{w,2}$. So let us assume that there is no such ROABP for B. Thus the assumption from Lemma 4.6 is fulfilled. Hence, we have a decomposition of A and B at the k-th layer into $A(\boldsymbol{x}) = R(\boldsymbol{y}_k)P(\boldsymbol{z}_k)$ and $B(\boldsymbol{x}) = R(\boldsymbol{y}_k)Q(\boldsymbol{z}_k)$, and there is a vector $\Gamma \in \mathbb{F}^{1\times w'}$ such that $\Gamma P = 0$ and $\Gamma Q \neq 0$, where $w' = (d+1)w$ and $\operatorname{supp}(\Gamma) \le w+1$.

Define R', P', Q' as the polynomials R, P, Q shifted by $\boldsymbol{f}_{w,2}$, respectively. Since $\Gamma P = 0$, we also have $\Gamma P' = 0$.

By the definition of R, there is an ROABP of width w' that computes R. Since $w' \le W_{w,2}$, polynomial R' is $\ell_{w,2}$-concentrated by the assumption of the lemma.

We argue that also $\Gamma Q'$ is $\ell_{w,2}$-concentrated: let $Q = [Q_1\, Q_2 \cdots Q_{w'}]^T \in \mathbb{F}[\boldsymbol{z}_k]^{w'\times 1}$. By Lemma 2.3, from the ROABP for B we get an ROABP for each Q_i of the same width w and the same variable order. Therefore we can combine them into one ROABP that computes $\Gamma Q = \sum_{i=1}^{w'} \gamma_i Q_i$. Its width is $w(w+1)$ because $\operatorname{supp}(\Gamma) \le w+1$. Since $w(w+1) \le W_{w,2}$, the polynomial $\Gamma Q'$ is $\ell_{w,2}$-concentrated.

Since $\Gamma Q \neq 0$ and $\Gamma Q'$ is $\ell_{w,2}$-concentrated, there exists at least one $\boldsymbol{b} \in \{0,1,\ldots,d\}^{n-k}$ with $\operatorname{supp}(\boldsymbol{b}) < \ell_{w,2}$ such that $\Gamma \operatorname{coeff}_{Q'}(\boldsymbol{z}_k^{\boldsymbol{b}}) \neq 0$. Because $\Gamma P = 0$, we have $\Gamma \operatorname{coeff}_{P'}(\boldsymbol{z}_k^{\boldsymbol{b}}) = 0$, and therefore

$$\Gamma \operatorname{coeff}_{P'+Q'}(\boldsymbol{z}_k^{\boldsymbol{b}}) \neq 0. \tag{21}$$

Recall that the coefficient space of R has full rank w'. Since a shift preserves the coefficient space, R' also has a full rank coefficient space. Because R' is $\ell_{w,2}$-concentrated, already the coefficients of the $< \ell_{w,2}$-support monomials of R' have full rank w'. That is, for $M_{\ell_{w,2}} = \{\boldsymbol{a} \in \{0,1,\ldots,d\}^k \mid \operatorname{supp}(\boldsymbol{a}) < \ell_{w,2}\}$, we have $\operatorname{rank}_{\mathbb{F}(t)}\{\operatorname{coeff}_{R'}(\boldsymbol{y}_k^{\boldsymbol{a}}) \mid \boldsymbol{a} \in M_{\ell_{w,2}}\} = w'$. Therefore, we can express Γ as a linear combination of these coefficients,

$$\Gamma = \sum_{\boldsymbol{a}\in M_{\ell_{w,2}}} \alpha_{\boldsymbol{a}} \operatorname{coeff}_{R'}(\boldsymbol{y}_k^{\boldsymbol{a}}),$$

where $\alpha_{\boldsymbol{a}}$ is a rational function in $\mathbb{F}(t)$, for $\boldsymbol{a} \in M_{\ell_{w,2}}$. Hence, from equation (21) we get

$$\begin{aligned}
\Gamma \operatorname{coeff}_{P'+Q'}(\boldsymbol{z}_k^{\boldsymbol{b}}) &= \left(\sum_{\boldsymbol{a} \in M_{\ell_{w,2}}} \alpha_{\boldsymbol{a}} \operatorname{coeff}_{R'}(\boldsymbol{y}_k^{\boldsymbol{a}}) \right) \operatorname{coeff}_{P'+Q'}(\boldsymbol{z}_k^{\boldsymbol{b}}) \\
&= \sum_{\boldsymbol{a} \in M_{\ell_{w,2}}} \alpha_{\boldsymbol{a}} \operatorname{coeff}_{R'(P'+Q')}(\boldsymbol{y}_k^{\boldsymbol{a}}\, \boldsymbol{z}_k^{\boldsymbol{b}}) \\
&= \sum_{\boldsymbol{a} \in M_{\ell_{w,2}}} \alpha_{\boldsymbol{a}} \operatorname{coeff}_{(A+B)'}(\boldsymbol{x}^{(\boldsymbol{a},\boldsymbol{b})}) \\
&\neq 0\,.
\end{aligned}$$

Since $\operatorname{supp}(\boldsymbol{a},\boldsymbol{b}) = \operatorname{supp}(\boldsymbol{a}) + \operatorname{supp}(\boldsymbol{b}) < 2\ell_{w,2}$, it follows that there is a monomial in $(A+B)'$ of support $< 2\ell_{w,2}$ with a nonzero coefficient. In other words, $(A+B)'$ is $2\ell_{w,2}$-concentrated. □

In Section 5, Theorem 5.6, we will show that the shift polynomial $\boldsymbol{f}_{w,2}(t) \in \mathbb{F}[t]^n$ used in Lemma 4.7 can be computed in time $(ndw)^{O(\log n)}$. The degree of $\boldsymbol{f}_{w,2}(t)$ is also $(ndw)^{O(\log n)}$. Recall that when we say that we shift by $\boldsymbol{f}_{w,2}(t)$, we actually mean that we plug in values for t up to the degree of $\boldsymbol{f}_{w,2}(t)$. That is, we have a family of $(ndw)^{O(\log n)}$ shifts, and at least one of them will give low support concentration. By Lemma 4.3, we get for each t, a potential hitting-set H_t of size $(nd)^{O(\ell_{w,2})} = (nd)^{O(\log dw)}$,

$$H_t = \{\boldsymbol{h} + \boldsymbol{f}(t) \mid \boldsymbol{h} \in \{0, \beta_1, \ldots, \beta_d\}^n \text{ and } \operatorname{supp}(\boldsymbol{h}) < 2\ell_{w,2}\}\,.$$

The final hitting-set is the union of all these sets, i.e. $H = \bigcup_t H_t$, where t takes $(ndw)^{O(\log n)}$ distinct values. Hence, save for a proof of Theorem 5.6, we have the following main result.

Theorem 4.8. *Given n, d, w, in time $(ndw)^{O(\log ndw)}$ one can construct a hitting-set for all n-variate polynomials of individual degree d, that can be computed by a sum of two ROABPs of width w.*

We extend Lemma 4.7 to the sum of c ROABPs.

Lemma 4.9. *Let $A = A_1 + A_2 + \cdots + A_c$, where the A_i's are n-variate polynomials of individual degree d, each computed by an* ROABP *of width w. Define $W_{w,c} = (d+1)(2w)^{2^{c-1}}$ and $\ell_{w,c} = \log(W_{w,c}^2 + 1)$. Let $\boldsymbol{f}_{w,c}(t) \in \mathbb{F}[t]^n$ be a shift that $\ell_{w,c}$-concentrates any polynomial (or matrix polynomial) that is computed by an* ROABP *of width $W_{w,c}$.*

Then $A' = A(\boldsymbol{x} + \boldsymbol{f}_{w,c})$ is $c\,\ell_{w,c}$-concentrated.

Proof. The proof is by induction on c. Lemma 4.7 provides the base case $c = 2$. For the induction step let $c \geq 3$. We follow the proof of Lemma 4.7 with $A = A_1$ and $B = \sum_{j=2}^{c} A_j$. Consider again the decomposition of A and B at the k-th layer into $A = RP$ and $B = RQ$, and let $\Gamma \in \mathbb{F}^{1 \times w'}$ such that $\Gamma P = 0$ and $\Gamma Q \neq 0$, where $w' = (d+1)w$ and $\operatorname{supp}(\Gamma) \leq w+1$.

The only difference to the proof of Lemma 4.7 is $Q = [Q_1\, Q_2 \cdots Q_{w'}]^T$. Recall from Lemma 4.6 that $Q_i = B_{(\boldsymbol{y}_k, \boldsymbol{a}_i)} = \sum_{j=2}^{c} A_{j(\boldsymbol{y}_k, \boldsymbol{a}_i)}$, for $\boldsymbol{a}_i \in \operatorname{depend}_k(A)$. Hence,

$$\Gamma Q = \sum_{i=1}^{w'} \gamma_i \left(\sum_{j=2}^{c} A_{j(\boldsymbol{y}_k, \boldsymbol{a}_i)} \right) = \sum_{j=2}^{c} \sum_{i=1}^{w'} \gamma_i A_{j(\boldsymbol{y}_k, \boldsymbol{a}_i)}\,.$$

By Lemma 2.3, ΓQ can be computed by a sum of $c-1$ ROABPs, each of width $w(w+1) \leq 2w^2 = w''$, because $\mathrm{supp}(\Gamma) \leq w+1$. Our definition of $W_{w,c}$ was chosen such that

$$W_{w'',c-1} = (d+1)(2w'')^{2^{c-2}} = (d+1)(2 \cdot 2w^2)^{2^{c-2}} = (d+1)(2w)^{2^{c-1}} = W_{w,c}\,.$$

Hence, $\boldsymbol{f}_{w,c}(t)$ is a shift that $\ell_{w'',c-1}$-concentrates any polynomial that is computed by an ROABP of width $W_{w'',c-1}$. By the induction hypothesis, we get that $\Gamma Q' = \Gamma Q(\boldsymbol{x} + \boldsymbol{f}_{w,c}(t))$ is $(c-1)\,\ell_{w'',c-1}$-concentrated, which is same as $(c-1)\,\ell_{w,c}$-concentrated.

Now we can proceed as in the proof of Lemma 4.7 and get that $(A+B)' = \sum_{j=1}^{c} A'_j$ has a monomial of support $< \ell_{w,c} + (c-1)\,\ell_{w,c} = c\,\ell_{w,c}$. □

We combine the lemmas similarly as for Theorem 4.8 and obtain our main result for the sum of constantly many ROABPs.

Theorem 4.10. *Given n, w, d, in time $(ndw)^{O(c\,2^c \log ndw)}$ one can construct a hitting-set for all n-variate polynomials of individual degree d, that can be computed by the sum of c ROABPs of width w.*

4.2 Concentration in matrix polynomials

As a by-product, we show that low support concentration can be achieved even when we have a sum of matrix polynomials, each computed by an ROABP. For a matrix polynomial $A(\boldsymbol{x}) \in \mathbb{F}^{w\times w}[\boldsymbol{x}]$, an ROABP is defined similar to the standard case. We have layers of nodes $V_0, V_1, \ldots, V_n$ connected by directed edges from V_{i-1} to V_i. Here, $V_0 = \{v_{0,1}, v_{0,2}, \ldots, v_{0,w}\}$ and $V_n = \{v_{n,1}, v_{n,2}, \ldots, v_{n,w}\}$ also consist of w nodes. The polynomial $A_{i,j}(\boldsymbol{x})$ at position (i,j) in $A(\boldsymbol{x})$ is the polynomial computed by the standard ROABP with start node $v_{0,i}$ and end node $v_{n,j}$.

Note that Definition 4.2 for ℓ-support concentration can be applied to polynomials over any $\mathbb{F}$-algebra.

We need the following lemma which is also of independent interest.

Lemma 4.11. *Let $A \in \mathbb{F}^{w\times w}[\boldsymbol{x}]$ be an n-variate polynomial and $\boldsymbol{f}(t)$ be a shift. Then $A(\boldsymbol{x}+\boldsymbol{f}(t))$ is ℓ-concentrated iff $\forall \alpha \in \mathbb{F}^{w\times w}$, $\langle \alpha, A\rangle\,(\boldsymbol{x}+\boldsymbol{f}(t))$ is ℓ-concentrated.*

Proof. Assume that $A'(\boldsymbol{x}) = A(\boldsymbol{x}+\boldsymbol{f})$ is not ℓ-concentrated. Then there exists a monomial $\boldsymbol{x}^{\boldsymbol{b}}$ such that $\mathrm{coeff}_{A'}(\boldsymbol{x}^{\boldsymbol{b}}) \notin \mathrm{span}_{\mathbb{F}(t)}\{\mathrm{coeff}_{A'}(\boldsymbol{x}^{\boldsymbol{a}}) \mid \mathrm{supp}(\boldsymbol{a}) < \ell\}$. Hence, there exists an $\alpha \in \mathbb{F}^{w\times w}$ such that $\langle \alpha, \mathrm{coeff}_{A'}(\boldsymbol{x}^{\boldsymbol{a}})\rangle = 0$, for all $\boldsymbol{a}$ with $\mathrm{supp}(\boldsymbol{a}) < \ell$, but $\langle \alpha, A'\rangle \neq 0$. We thus found an $\alpha \in \mathbb{F}^{w\times w}$ such that $\langle \alpha, A\rangle\,(\boldsymbol{x}+\boldsymbol{f}(t))$ is not ℓ-concentrated.

For the other direction, let $A(\boldsymbol{x}+\boldsymbol{f})$ be ℓ-concentrated. Hence, any coefficient $\mathrm{coeff}_{A'}(\boldsymbol{x}^{\boldsymbol{a}})$ can be written as a linear combination of the small support coefficients,

$$\mathrm{coeff}_{A'}(\boldsymbol{x}^{\boldsymbol{a}}) = \sum_{\substack{\boldsymbol{b} \\ \mathrm{supp}(\boldsymbol{b})<\ell}} \gamma_{\boldsymbol{b}}\ \mathrm{coeff}_{A'}(\boldsymbol{x}^{\boldsymbol{b}}),$$

for some $\gamma_{\boldsymbol{b}} \in \mathbb{F}$. Hence, for any $\alpha \in \mathbb{F}^{w\times w}$, we also have

$$\langle \alpha, \mathrm{coeff}_{A'}(\boldsymbol{x}^{\boldsymbol{a}})\rangle = \left\langle \alpha,\ \sum_{\substack{\boldsymbol{b} \\ \mathrm{supp}(\boldsymbol{b})<\ell}} \gamma_{\boldsymbol{b}}\ \mathrm{coeff}_{A'}(\boldsymbol{x}^{\boldsymbol{b}}) \right\rangle.$$

That is, $\langle \alpha, A\rangle\,(\boldsymbol{x}+\boldsymbol{f}(t))$ is ℓ-concentrated. □

Now we can show low support concentration for a sum of matrix polynomials, each computed by an ROABP, analogous to Lemma 4.9.

Corollary 4.12. *Let $A = A_1 + A_2 + \cdots + A_c$, where each $A_i \in \mathbb{F}^{w\times w}[\boldsymbol{x}]$ is an n-variate matrix polynomials of individual degree d, each computed by an* ROABP *of width w. Let $\boldsymbol{f}_{w,c}$ and $\ell_{w,c}$ be defined as in Lemma 4.9.*

Then $A(\boldsymbol{x} + \boldsymbol{f}_{w^2,c})$ is $c\ell_{w^2,c}$-concentrated.

Proof. Let $\alpha \in \mathbb{F}^{w\times w}$ and consider the dot-product $\langle \alpha, A_i\rangle \in \mathbb{F}[\boldsymbol{x}]$. This polynomial can be computed by an ROABP of width w^2: we take the ROABP of width w for A_i and make w copies of it, and two new nodes s and t. We add the following edges.

- Connect the new start node s to the h-th former start node of the h-th copy of the ROABP by edges of weight one, for all $1 \le h \le w$.
- Connect the j-th former end node of the h-th copy of the ROABP to the new end node t by an edge of weight $\alpha_{h,j}$, for all $1 \le h, j \le w$.

The resulting ROABP has width w^2 and computes $\langle \alpha, A_i\rangle$.

Now consider the polynomial $\langle \alpha, A\rangle = \langle \alpha, A_1\rangle + \langle \alpha, A_2\rangle + \cdots + \langle \alpha, A_c\rangle$. It can be computed by a sum of c ROABPs, each of width w^2, for every $\alpha \in \mathbb{F}^{w\times w}$. Hence, by Lemma 4.9, the polynomial $\langle \alpha, A\rangle\,(\boldsymbol{x} + \boldsymbol{f}_{w^2,c})$ is $c\ell_{w^2,c}$-concentrated, for every $\alpha \in \mathbb{F}^{w\times w}$. By Lemma 4.11, it follows that $A(\boldsymbol{x} + \boldsymbol{f}_{w^2,c})$ is $c\ell_{w^2,c}$-concentrated. □

5 Low Support Concentration in ROABPs

Recall that a polynomial $A(\boldsymbol{x})$ over an $\mathbb{F}$-algebra $\mathbb{A}$ is called low-support concentrated if its low-support coefficients span all its coefficients. We show an efficient shift which achieves concentration in matrix polynomials computed by ROABPs. We use the quasi-polynomial size hitting-set for ROABPs given by Agrawal et al. [AGKS15]. Their hitting-set is based on a *basis isolating weight assignment* which we define next.

Recall that $M = \{0, 1, \ldots, d\}^n$ denotes the set of all exponents of monomials in $\boldsymbol{x}$ of individual degree bounded by d. For $\ell \ge 1$, we define the set $M_\ell \subseteq M$ as the exponents of low support,

$$M_\ell = \{\boldsymbol{a} \in M \mid \mathrm{supp}(\boldsymbol{a}) < \ell\}.$$

For a weight function $\mathrm{w}\colon [n] \to \mathbb{N}$ and $\boldsymbol{a} = (a_1, a_2, \ldots, a_n) \in M$, let the weight of $\boldsymbol{a}$ be $\mathrm{w}(\boldsymbol{a}) = \sum_{i=1}^n \mathrm{w}(i)a_i$. Let $\mathbb{A}_k$ be a k-dimensional algebra over the field $\mathbb{F}$. For any two sets M_1, M_2, an $M_1 \times M_2$ matrix will be a matrix whose rows and columns are indexed by the elements in the sets M_1 and M_2, respectively.

Definition 5.1. *A weight function* $\mathrm{w}\colon [n] \to \mathbb{N}$ *is called a* basis isolating weight assignment for a polynomial $A(\boldsymbol{x}) \in \mathbb{A}_k[\boldsymbol{x}]$*, if there exists $S \subseteq M$ with $|S| \le k$ such that*

- $\forall\, \boldsymbol{a} \ne \boldsymbol{b} \in S, \quad \mathrm{w}(\boldsymbol{a}) \ne \mathrm{w}(\boldsymbol{b})$ *and*
- $\forall\, \boldsymbol{a} \in \overline{S} := M - S, \quad \mathrm{coeff}_A(\boldsymbol{x}^{\boldsymbol{a}}) \in \mathrm{span}_{\mathbb{F}}\{\mathrm{coeff}_A(\boldsymbol{x}^{\boldsymbol{b}}) \mid \boldsymbol{b} \in S$ *and* $\mathrm{w}(\boldsymbol{b}) < \mathrm{w}(\boldsymbol{a})\}$.

Agrawal et al. [AGKS15, Lemma 8] presented a quasi-polynomial time construction of such a weight function for any polynomial $A(\boldsymbol{x}) \in \mathbb{F}^{w\times w}[\boldsymbol{x}]$ computed by an ROABP. The hitting-set is then defined by points $(t^{\mathrm{w}(1)}, t^{\mathrm{w}(2)}, \ldots, t^{\mathrm{w}(n)})$ for $\mathsf{poly}(n,d,w)^{\log n}$ many t's. Our approach now is to use this weight function for a shift of $A(\boldsymbol{x})$ by $\left(t^{\mathrm{w}(i)}\right)_{i=1}^{n}$. Let $A'(\boldsymbol{x})$ denote the shifted polynomial,

$$A'(\boldsymbol{x}) = A(\boldsymbol{x} + t^{\mathrm{w}}) = A(x_1 + t^{\mathrm{w}(1)}, x_2 + t^{\mathrm{w}(2)}, \ldots, x_n + t^{\mathrm{w}(n)})\,.$$

We will prove that A' has low support concentration.

Recall from Lemma 4.5 that the coefficients of A' are linear combinations of the coefficients of A. Adapting equation (17), we have

$$\operatorname{coeff}_{A'}(\boldsymbol{x}^{\boldsymbol{a}}) = \sum_{\boldsymbol{b}\in M} \binom{\boldsymbol{b}}{\boldsymbol{a}} t^{\mathrm{w}(\boldsymbol{b}-\boldsymbol{a})} \cdot \operatorname{coeff}_{A}(\boldsymbol{x}^{\boldsymbol{b}}), \tag{22}$$

where $\binom{\boldsymbol{b}}{\boldsymbol{a}} = \prod_{i=1}^{n} \binom{b_i}{a_i}$ for any $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{N}^n$.

Equation (22) can be expressed in terms of matrices. Let C be the coefficient matrix of A, i.e. the $M \times [k]$ matrix with the coefficients $\operatorname{coeff}_A(\boldsymbol{x}^{\boldsymbol{a}})$ as rows,

$$C(\boldsymbol{a}, \cdot) = \operatorname{coeff}_A(\boldsymbol{x}^{\boldsymbol{a}})^T\,.$$

Similarly, let C' be the $M \times [k]$ with the coefficients $\operatorname{coeff}_{A'}(\boldsymbol{x}^{\boldsymbol{a}})$ as rows. Let furthermore T be the $M \times M$ *transfer matrix* given by

$$T(\boldsymbol{a}, \boldsymbol{b}) = \binom{\boldsymbol{b}}{\boldsymbol{a}}\,,$$

and D be the $M \times M$ diagonal matrix given by

$$D(\boldsymbol{a}, \boldsymbol{a}) = t^{\mathrm{w}(\boldsymbol{a})}\,.$$

The inverse of D is the diagonal matrix given by $D^{-1}(\boldsymbol{a}, \boldsymbol{a}) = t^{-\mathrm{w}(\boldsymbol{a})}$. Now equation (22) becomes

$$C' = D^{-1}TDC\,. \tag{23}$$

As shifting is an invertible operation, the matrix T is also invertible and $\operatorname{rank}(C') = \operatorname{rank}(C)$.

Lemma 5.2 (Isolation to concentration)**.** *Let $A(\boldsymbol{x})$ be a polynomial over a k-dimensional algebra $\mathbb{A}_k$. Let* w *be a basis isolating weight assignment for $A(\boldsymbol{x})$. Then $A(\boldsymbol{x} + t^{\mathrm{w}})$ is ℓ-concentrated, where $\ell = \lceil \log(k+1) \rceil$.*

Proof. Let $A'(\boldsymbol{x}) = A(\boldsymbol{x} + t^{\mathrm{w}})$. We reconsider equation (23) with respect to the low support monomials. Recall that $M_\ell = \{\boldsymbol{a} \in M \mid \operatorname{supp}(\boldsymbol{a}) < \ell\}$. We define matrices

- C'_ℓ : the $M_\ell \times [k]$ submatrix of C' that contains the coefficients of A' of support $< \ell$,
- T_ℓ : the $M_\ell \times M$ submatrix of T restricted to the rows $\boldsymbol{a} \in M_\ell$,
- D_ℓ : the $M_\ell \times M_\ell$ submatrix of D restricted to the rows and columns from M_ℓ.

To show that A' is ℓ-concentrated, we need to prove that $\mathrm{rank}(C'_\ell) = \mathrm{rank}(C)$. By equation (23), matrix C'_ℓ can be written as $C'_\ell = D_\ell^{-1} T_\ell D C$. Since D_ℓ and D_ℓ^{-1} are diagonal matrices, they have full rank. Hence, it suffices to show that $\mathrm{rank}(T_\ell D C) = \mathrm{rank}(C)$.

W.l.o.g. we assume that the order of the rows and columns in all the above matrices that are indexed by M or M_ℓ is according to increasing weight $\mathrm{w}(\boldsymbol{a})$ of the indices $\boldsymbol{a}$. The rows with the same weight can be arranged in an arbitrary order.

Now, recall that w is a basis isolating weight assignment. Hence, there exists a set $S \subseteq M$ such that the coefficients $\mathrm{coeff}_A(\boldsymbol{b})$, for $\boldsymbol{b} \in S$, span all coefficients $\mathrm{coeff}_A(\boldsymbol{a})$, for $\boldsymbol{a} \in M$. In terms of the coefficient matrix C, for any $\boldsymbol{a} \in M$ we can write

$$C(\boldsymbol{a}, \cdot) \in \mathrm{span}\{C(\boldsymbol{b}, \cdot) \mid \boldsymbol{b} \in S \text{ and } \mathrm{w}(\boldsymbol{b}) < \mathrm{w}(\boldsymbol{a})\}. \tag{24}$$

Let $S = \{\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_{k'}\}$ for some $k' \leq k$. Let C_0 be the $k' \times k$ submatrix of C whose i-th row is $C(\boldsymbol{s}_i, \cdot)$, i.e. $C_0(i, \cdot) = C(\boldsymbol{s}_i, \cdot)$. By (24), for every $\boldsymbol{a} \in M$, there is a vector $\boldsymbol{\gamma_a} = (\gamma_{\boldsymbol{a},1}, \gamma_{\boldsymbol{a},2}, \ldots, \gamma_{\boldsymbol{a},k'}) \in \mathbb{F}^{k'}$ such that $C(\boldsymbol{a}, \cdot) = \sum_{j=1}^{k'} \gamma_{\boldsymbol{a},j}\, C_0(j, \cdot)$. Let $\Gamma = (\gamma_{\boldsymbol{a},j})_{\boldsymbol{a},j}$ be the $M \times [k']$ matrix with these vectors as rows. Then we get

$$C = \Gamma C_0 \,.$$

Observe that the $\boldsymbol{s}_i$-th row of Γ is simply $\boldsymbol{e}_i$, the i-th standard unit vector. By (24), the coefficient $C(\boldsymbol{s}_i, \cdot)$ is used to express $C(\boldsymbol{a}, \cdot)$ only when $\mathrm{w}(\boldsymbol{a}) > \mathrm{w}(\boldsymbol{s}_i)$. Recall that the rows of the matrices indexed by M, like Γ, are in order the of increasing weight of the index. Therefore, when we consider the i-th column of Γ from the top, the entries are all zero down to row s_i, where we hit on the one from $\boldsymbol{e}_i$,

$$\Gamma(\boldsymbol{s}_i, i) = 1 \;\text{ and }\; \forall\, \boldsymbol{a} \neq \boldsymbol{s}_i,\ \mathrm{w}(\boldsymbol{a}) \leq \mathrm{w}(\boldsymbol{s}_i) \implies \Gamma(\boldsymbol{a}, i) = 0 \,. \tag{25}$$

Recall that our goal is to show $\mathrm{rank}(T_\ell D C) = \mathrm{rank}(C)$. For this, it suffices to show that the $M_\ell \times k'$ matrix $R = T_\ell D \Gamma$ has full column rank k', because then we have $\mathrm{rank}(T_\ell D C) = \mathrm{rank}(T_\ell D \Gamma C_0) = \mathrm{rank}(R C_0) = \mathrm{rank}(C_0) = \mathrm{rank}(C)$.

To show that R has full column rank k', observe that the j-th column of R can be written as

$$R(\cdot, j) = \sum_{\boldsymbol{a} \in M} T_\ell(\cdot, \boldsymbol{a})\, \Gamma(\boldsymbol{a}, j)\, t^{\mathrm{w}(\boldsymbol{a})} \,. \tag{26}$$

By (25), the term with the lowest degree in equation (26) is $t^{\mathrm{w}(\boldsymbol{s}_j)}$. By $\mathrm{lc}(R(\cdot, j))$ we denote the coefficient of the lowest degree term in the polynomial $R(\cdot, j)$. Because $\Gamma(\boldsymbol{s}_j, j) = 1$, we have

$$\mathrm{lc}(R(\cdot, j)) = T_\ell(\cdot, \boldsymbol{s}_j) \,.$$

We define the $M_\ell \times [k']$ matrix R_0 whose j-th column is $\mathrm{lc}(R(\cdot, j))$, i.e. $R_0(\cdot, j)) = T_\ell(\cdot, \boldsymbol{s}_j)$. We will show in Lemma 5.3 below that the columns of matrix T_ℓ indexed by the set S are linearly independent. Therefore the k' columns of R_0 are linearly independent.

Hence, there are k' rows in R_0 such that its restriction to these rows, say R'_0, is a square matrix with nonzero determinant. Let R' denote the restriction of R to the same set of rows. Now observe that the lowest degree term in $\det(R')$ has coefficient precisely $\det(R'_0)$, i.e., $\mathrm{lc}(\det(R')) = \det(R'_0)$. This is because the lowest degree term in $\det(R')$ has degree $\sum_{j=1}^{k'} \mathrm{w}(\boldsymbol{s}_j)$, and this degree can only be obtained when the degree $\mathrm{w}(\boldsymbol{s}_j)$ term is taken from the j-th column, for all j. We conclude that $\det(R') \neq 0$ and hence R has full column rank. □

It remains to show that the $k' \leq k$ columns of matrix T_ℓ indexed by the set S are linearly independent. In fact, we will show that any $k = 2^\ell - 1$ columns of T_ℓ are independent.

Lemma 5.3. *Let T_ℓ be the $M_\ell \times M$ matrix with $T_\ell(\boldsymbol{a}, \boldsymbol{b}) = \binom{\boldsymbol{b}}{\boldsymbol{a}}$. Any $2^\ell - 1$ columns of matrix T_ℓ are linearly independent.*

Proof. Let $S \subseteq M$ now be any set of size $k = 2^\ell - 1$. Let $T_{\ell,k}$ be the $M_\ell \times S$ submatrix of T_ℓ that consists of the columns indexed by S. To prove the lemma we will show that for any $0 \neq \boldsymbol{v} \in \mathbb{F}^k$ we have $T_{\ell,k}\boldsymbol{v} \neq 0$.

Let $\boldsymbol{v} = (v_{\boldsymbol{a}})_{\boldsymbol{a} \in S}$. Define the polynomial $V(\boldsymbol{x}) = \sum_{\boldsymbol{a} \in S} v_{\boldsymbol{a}} \boldsymbol{x}^{\boldsymbol{a}} \in \mathbb{F}[\boldsymbol{x}]$. Let $V'(\boldsymbol{x})$ be the polynomial where every variable in $V(\boldsymbol{x})$ is shifted by one: $V'(\boldsymbol{x}) = V(\boldsymbol{x} + \mathbf{1})$. From equation (22) we get that for any $\boldsymbol{a} \in M_\ell$,

$$\operatorname{coeff}_{V'}(\boldsymbol{x}^{\boldsymbol{a}}) = \sum_{\boldsymbol{b} \in S} \binom{\boldsymbol{b}}{\boldsymbol{a}} v_{\boldsymbol{b}} = T_{\ell,k}(\boldsymbol{a}, \cdot)\, \boldsymbol{v}\,.$$

Hence, $T_{\ell,k}\boldsymbol{v}$ gives all the coefficients of $V'(\boldsymbol{x})$ of support $< \ell$. Now it remains to show that at least one of these coefficients is nonzero. We show this in our next claim about *concentration in sparse polynomials*. This claim was independently proven by Forbes [For15, Corollary 5.14] using a different technique.

Claim 5.4. *Let $V(\boldsymbol{x}) \in \mathbb{F}[\boldsymbol{x}]$ be a non-zero n-variate polynomial with sparsity bounded by $2^\ell - 1$. Then $V'(\boldsymbol{x}) = V(\boldsymbol{x} + \mathbf{1})$ has a nonzero coefficient of support $< \ell$.*

We prove the claim by induction on the number of variables n. For $n = 1$, polynomial $V(\boldsymbol{x})$ is univariate, i.e. all monomials in $V(\boldsymbol{x})$ have support 1. Hence, for $\ell > 1$ it suffices to show that $V'(\boldsymbol{x}) \neq 0$. But this is equivalent to $V(\boldsymbol{x}) \neq 0$, which holds by assumption. If $\ell = 1$, then $V(\boldsymbol{x})$ is a univariate polynomial with exactly one monomial, and therefore $V(\boldsymbol{x}+\mathbf{1})$ has a nonzero constant part.

Now assume that the claim is true for $n - 1$ and let $V(\boldsymbol{x})$ have n variables. Let $\boldsymbol{x}_{n-1}$ denote the set of first $n - 1$ variables. Let us write $V(\boldsymbol{x}) = \sum_{i=0}^{d} U_i\, x_n^i$, where $U_i \in \mathbb{F}[\boldsymbol{x}_{n-1}]$, for every $0 \leq i \leq d$. Let $U_i'(\boldsymbol{x}_{n-1}) = U_i(\boldsymbol{x}_{n-1} + \mathbf{1})$ be the shifted polynomial, for every $0 \leq i \leq d$. We consider two cases:

Case 1: There is exactly one index $i \in [0, d]$ for which $U_i \neq 0$. Then U_i has sparsity $\leq 2^\ell - 1$. Because U_i is an $(n-1)$-variate polynomial, U_i' has a nonzero coefficient of support $< \ell$ by inductive hypothesis.

Thus, $V'(\boldsymbol{x}) = (x_n + 1)^i\, U_i'$ also has a nonzero coefficient of support $< \ell$.

Case 2: There are at least two U_i's which are nonzero. Then there is at least one index in $i \in [0, d]$ such that U_i has sparsity $2^{\ell-1} - 1$. And hence, by the inductive hypothesis, U_i' has a nonzero coefficient of support $< \ell - 1$. Consider the largest index j such that U_j' has a nonzero coefficient of support $< \ell - 1$. Let the corresponding monomial be $\boldsymbol{x}_{n-1}^{\boldsymbol{a}}$. Now, as $V'(\boldsymbol{x}) = \sum_{i=0}^{d} U_i'\,(x_n + 1)^i$, we have that

$$\operatorname{coeff}_{V'}(\boldsymbol{x}_{n-1}^{\boldsymbol{a}} x_n^j) = \sum_{r=j}^{d} \binom{r}{j} \operatorname{coeff}_{U_r'}(\boldsymbol{x}_{n-1}^{\boldsymbol{a}}).$$

By our choice of j we have $\operatorname{coeff}_{U_j'}(\boldsymbol{x}_{n-1}^{\boldsymbol{a}}) \neq 0$ and $\operatorname{coeff}_{U_r'}(\boldsymbol{x}_{n-1}^{\boldsymbol{a}}) = 0$, for $r > j$. Hence, $\operatorname{coeff}_{V'}(\boldsymbol{x}_{n-1}^{\boldsymbol{a}} x_n^j) \neq 0$. The monomial $\boldsymbol{x}_{n-1}^{\boldsymbol{a}} x_n^j$ has support $< \ell$, which proves our claim and the lemma. □

We can use Lemma 5.2 to get concentration in a polynomial computed by an ROABP. Let $A(\boldsymbol{x}) \in \mathbb{F}^{w\times w}[\boldsymbol{x}]$ be a polynomial in n variables of individual degree d that can be computed by an ROABP of width w. Agrawal et al. [AGKS15, Lemma 8] constructed a family $\mathcal{B} = \{\mathrm{w}_1, \mathrm{w}_2, \ldots, \mathrm{w}_N\}$ of weight assignments such that at least one of the weights is a basis isolating weight assignment for $A(\boldsymbol{x})$, where $N = (ndw)^{O(\log n)}$. Hence, by Lemma 5.2, at least one of the n-tuples in the family $\mathcal{F} = \{t^{\mathrm{w}_1}, t^{\mathrm{w}_2}, \ldots, t^{\mathrm{w}_N}\}$ gives $\log(w^2+1)$-concentration in $A(\boldsymbol{x})$ upon shifting by it.

Let $\boldsymbol{f}_i = \{f_{i,1}, f_{i,2}, \ldots, f_{i,n}\}$ denote the n-tuple t^{w_i} for each $1 \le i \le N$. The degrees of these tuples are bounded by $D = \max\{\deg(f_{i,j}) \mid i \in [N] \text{ and } j \in [n]\} = (ndw)^{O(\log n)}$. The family $\mathcal{F}$ can be generated in time $(ndw)^{O(\log n)}$.

By Lemma 5.2, we now have an alternative PIT for *one* ROABP because we could simply try all $\boldsymbol{f}_i \in \mathcal{F}$ as a shift, and we know that at least one will provide low-support concentration. However, in Lemmas 4.7 and 4.9 we apply the shift to several ROABPs simultaneously, and we have no guarantee that one of the shifts works for all of them. We solve this problem by combining the n-tuples in $\mathcal{F}$ into one single shift that works for every ROABP.

Let $\boldsymbol{L}(y,t) \in \mathbb{F}[y,t]^n$ be the Lagrange interpolation of $\mathcal{F}$. That is, for all $j \in [n]$,

$$L_j = \sum_{i\in[N]} f_{i,j}(t) \prod_{\substack{i'\in[N]\\ i'\neq i}} \frac{y-\alpha_{i'}}{\alpha_i - \alpha_{i'}}, \tag{27}$$

where α_i is an arbitrary unique field element associated with i, for all $i \in [N]$. (Recall that we assume that the field $\mathbb{F}$ is large enough that these elements exist.) Note that $L_j|_{y=\alpha_i} = f_{i,j}$. Thus, $\boldsymbol{L}|_{y=\alpha_i} = \boldsymbol{f}_i$. Also, $\deg_y(L_j) = N-1$ and $\deg_t(L_j) \le D$.

Lemma 5.5. *Let $A(\boldsymbol{x})$ be a n-variate polynomial over a k-dimensional $\mathbb{F}$-algebra $\mathbb{A}_k$ and $\mathcal{F}$ be a family of n-tuples of polynomials in variable t, such that there exists an $\boldsymbol{f}(t) \in \mathcal{F}$ such that $A'(\boldsymbol{x},t) = A(\boldsymbol{x}+\boldsymbol{f}(t)) \in \mathbb{A}_k(t)[\boldsymbol{x}]$ is ℓ-concentrated. Let $\boldsymbol{L}(y,t)$ be the Lagrange interpolation of $\mathcal{F}$. Then $A''(\boldsymbol{x},y,t) = A(\boldsymbol{x}+\boldsymbol{L}(y,t)) \in \mathbb{A}_k(y,t)[\boldsymbol{x}]$ is ℓ-concentrated.*

Proof. Let $\mathrm{rank}_{\mathbb{F}}\{\mathrm{coeff}_A(\boldsymbol{x}^{\boldsymbol{a}}) \mid \boldsymbol{a} \in M\} = k'$, for some $k' \le k$, and $M_\ell = \{\boldsymbol{a} \in M \mid \mathrm{supp}(\boldsymbol{a}) < \ell\}$. We need to show that $\mathrm{rank}_{\mathbb{F}(y,t)}\{\mathrm{coeff}_{A''}(\boldsymbol{x}^{\boldsymbol{a}}) \mid \boldsymbol{a} \in M_\ell\} = k'$.

Let $\boldsymbol{f}(t) \in \mathcal{F}$ be a function as in the assumption of the theorem, and let $\alpha \in \mathbb{F}$ be such that $\boldsymbol{L}|_{y=\alpha} = \boldsymbol{f}$.

Since $A'(\boldsymbol{x})$ is ℓ-concentrated, we have that $\mathrm{rank}_{\mathbb{F}(t)}\{\mathrm{coeff}_{A'}(\boldsymbol{x}^{\boldsymbol{a}}) \mid \boldsymbol{a} \in M_\ell\} = k'$. Recall that $A'(\boldsymbol{x})$ is an evaluation of A'' at $y=\alpha$, i.e. $A'(\boldsymbol{x},t) = A''(\boldsymbol{x},\alpha,t)$. Thus, for all $\boldsymbol{a} \in M$ we have $\mathrm{coeff}_{A'}(\boldsymbol{x}^{\boldsymbol{a}}) = \mathrm{coeff}_{A''}(\boldsymbol{x}^{\boldsymbol{a}})|_{y=\alpha}$.

Let $C \in \mathbb{F}[t]^{k\times|M_\ell|}$ be the matrix whose columns are $\mathrm{coeff}_{A'}(\boldsymbol{x}^{\boldsymbol{a}})$, for $\boldsymbol{a} \in M_\ell$. Let similarly $C' \in \mathbb{F}[y,t]^{k\times|M_\ell|}$ be the matrix whose columns are $\mathrm{coeff}_{A''}(\boldsymbol{x}^{\boldsymbol{a}})$, for $\boldsymbol{a} \in M_\ell$. Then we have $C = C'|_{y=\alpha}$.

As $\mathrm{rank}_{\mathbb{F}(t)}(C) = k'$, there are k' rows in C, say indexed by R, such that $\det(C(R,\cdot)) \neq 0$. Because $\det(C(R,\cdot)) = \det(C'(R,\cdot))|_{y=\alpha_i}$, it follows that $\det(C'(R,\cdot)) \neq 0$. Hence, we have $\mathrm{rank}_{\mathbb{F}(y,t)}(C') = k'$. □

Using the Lagrange interpolation, we can construct a single shift, which works for all ROABPs of width $\le w$.

Theorem 5.6. *Given n, d, w, in time $(ndw)^{O(\log n)}$ one can compute a polynomial $\boldsymbol{f}(t) \in \mathbb{F}[t]^n$ of degree $(ndw)^{O(\log n)}$ such that for any n-variate polynomial $A(\boldsymbol{x}) \in \mathbb{F}^{w\times w}[\boldsymbol{x}]$ (or*

$\mathbb{F}^{1\times w}[\boldsymbol{x}]$, *or* $\mathbb{F}[\boldsymbol{x}]$*) of individual degree* d *that can be computed by an ROABP of width* w*, the polynomial* $A(\boldsymbol{x}+\boldsymbol{f}(t))$ *is* $\log(w^2+1)$*-concentrated, where the coefficient space of* A *is considered over* $\mathbb{F}(t)$*.*

Proof. Recall that for any polynomial $A(\boldsymbol{x}) \in \mathbb{F}^{w\times w}[\boldsymbol{x}]$ computed by an ROABP, at least one tuple in the family $\{\boldsymbol{f}_1, \boldsymbol{f}_2, \ldots, \boldsymbol{f}_N\}$ obtained from [AGKS15, Lemma 8], gives $\log(w^2+1)$-concentration. By Lemma 5.5, the Lagrange interpolation $\boldsymbol{L}(y,t)$ of $\{\boldsymbol{f}_1, \boldsymbol{f}_2, \ldots, \boldsymbol{f}_N\}$ has y- and t-degrees $(ndw)^{O(\log n)}$. After shifting an n-variate polynomial of individual degree d by $\boldsymbol{L}(y,t)$, its coefficients will be polynomials in y and t, with degree $d' = dn(ndw)^{O(\log n)}$. Consider the determinant polynomial $\det(C'(R,\cdot))$ from Lemma 5.5. As the set of coefficients of polynomial $A(\boldsymbol{x})$ have rank bounded by w^2, $\det(C'(R,\cdot))$ has degree bounded by $d'' = w^2 d'$.

Note that when we replace y by $t^{d''+1}$, this will not affect the non-zeroness of the determinant, and hence, the concentration is preserved. Thus, $\boldsymbol{f} = \boldsymbol{L}(t^{d''+1}, t)$ is an n-tuple of univariate polynomials in t that fulfills the claim of the theorem.

Now, consider the case when the ROABP computes a polynomial $A(\boldsymbol{x}) \in \mathbb{F}^{1\times w}[x]$. It is easy to see that there exist $S \in \mathbb{F}^{1\times w}$ and $B \in \mathbb{F}^{w\times w}[\boldsymbol{x}]$ computed by a width-w ROABP such that $A = SB$. We know that $B(\boldsymbol{x}+\boldsymbol{f}(t))$ has $\log(w^2+1)$-concentration. As multiplying by S is a linear operation, one can argue as in the proof of Lemma 4.11 that any linear dependence among coefficients of $B(\boldsymbol{x}+\boldsymbol{f}(t))$ also holds among coefficients of $A(\boldsymbol{x}+\boldsymbol{f}(t))$. Hence, $A(\boldsymbol{x}+\boldsymbol{f}(t))$ has $\log(w^2+1)$-concentration. A similar argument would work when $A(\boldsymbol{x}) \in \mathbb{F}[\boldsymbol{x}]$, by writing $A = SBT$, for some $S \in \mathbb{F}^{1\times w}$ and $T \in \mathbb{F}^{w\times 1}$. □

6 Discussion

The first question is whether one can make the time complexity for PIT for the sum of c ROABPs proportional to $w^{O(c)}$ instead of $w^{O(2^c)}$. This blow up happens because, when we want to combine $w+1$ partial derivative polynomials given by ROABPs of width w, we get an ROABP of width $O(w^2)$. There are examples where this bound seems tight. So, a new property of sum of ROABPs needs to be discovered.

It also needs to be investigated if these ideas can be generalized to work for sum of more than constantly many ROABPs, or depth-3 multilinear circuits.

As mentioned in the introduction, the idea for equivalence of two ROABPs was inspired from the equivalence of two read once boolean branching programs (OBDD). It would be interesting to know if there are concrete connections between arithmetic and boolean branching programs. In particular, can ideas from identity testing of an ROABP be applied to construct pseudo-randomness for OBDD.

7 Acknowledgements

We thank Manindra Agrawal, Chandan Saha and Vineet Nair for very useful discussions and constant encouragement. The work was initiated when TT was visiting CSE, IIT Kanpur. Part of the work was done during Dagstuhl Seminar 14391 on Algebra in Computational Complexity 2014. We thank anonymous referees for the useful suggestions.

References

[AGKS15] Manindra Agrawal, Rohit Gurjar, Arpita Korwar, and Nitin Saxena. Hitting-sets for ROABP and sum of set-multilinear circuits. *SIAM Journal on Computing*, 44(3):669–697, 2015.

[Agr05] Manindra Agrawal. Proving lower bounds via pseudo-random generators. In *Proceedings of Foundations of 25th Software Technology and Theoretical Computer Science (FSTTCS)*, volume 3821 of *Lecture Notes in Computer Science*, pages 92–105, 2005.

[AL86] Leonard M. Adleman and Hendrik W. Lenstra. Finding irreducible polynomials over finite fields. In *Proceedings of the 18th ACM Symposium on Theory of Computing (STOC)*, pages 350–355, New York, NY, USA, 1986. ACM.

[ASS13] Manindra Agrawal, Chandan Saha, and Nitin Saxena. Quasi-polynomial hitting-set for set-depth-D formulas. In *Proceedings of the 45th ACM Symposium on Theory of Computing (STOC)*, pages 321–330, 2013.

[BOT88] Michael Ben-Or and Prasoon Tiwari. A deterministic algorithm for sparse multivariate polynomial interpolation. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, Proceedings of the 20th ACM Symposium on Theory of Computing (STOC), pages 301–309, New York, NY, USA, 1988. ACM.

[BW98] Jan Behrens and Stephan Waack. Equivalence test and ordering transformation for parity-OBDDs of different variable ordering. In *15th Symposium on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science 1373, pages 227–237. Springer-Verlag, 1998.

[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 Journal on Computing*, 36(5):1404–1434, 2007.

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

[For15] Michael A. Forbes. Deterministic divisibility testing via shifted partial derivatives. In *Proceedings of the 56th IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 451–465, 2015.

[FS12a] Michael A. Forbes and Amir Shpilka. On identity testing of tensors, low-rank recovery and compressed sensing. In *Proceedings of the 44th ACM Symposium on Theory of Computing (STOC)*, pages 163–172, 2012.

[FS12b] Michael A. Forbes and Amir Shpilka. Quasipolynomial-time identity testing of non-commutative and read-once oblivious algebraic branching programs. Technical report, CoRR, 2012.

[FS13] 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 IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 243–252, 2013.

[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 ACM Symposium on Theory of Computing (STOC)*, 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 IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 578–587, 2013.

[GKST15] Rohit Gurjar, Arpita Korwar, Nitin Saxena, and Thomas Thierauf. Deterministic identity testing for sum of read-once oblivious arithmetic branching programs. In *30th Conference on Computational Complexity (CCC)*, pages 323–346, 2015.

[GM96] Jordan Gergov and Christoph Meinel. Mod-2-OBDDs - a data structure that generalizes EXOR-sum-of-products and ordered binary decision diagrams. *Formal Methods in System Design*, 8:273–282, 1996.

[JQS10] Maurice J. Jansen, Youming Qiao, and Jayalal Sarma. Deterministic identity testing of read-once algebraic branching programs. Technical report, Electronic Colloquium on Computational Complexity (ECCC), 2010.

[KI04] Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004.

[KNS15] Neeraj Kayal, Vineet Nair, and Chandan Saha. Separation between read-once oblivious algebraic branching programs (ROABPs) and multilinear depth three circuits. Technical report, Electronic Colloquium on Computational Complexity (ECCC), 2015.

[KS01] Adam Klivans and Daniel A. Spielman. Randomness efficient identity testing of multivariate polynomials. In *Proceedings of the 33rd ACM Symposium on Theory of Computing (STOC)*, pages 216–223, 2001.

[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 IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 198–207, 2009.

[KS11] Zohar Shay Karnin and Amir Shpilka. Black box polynomial identity testing of generalized depth-3 arithmetic circuits with bounded top fan-in. *Combinatorica*, 31(3):333–364, 2011.

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

[OSV15] Rafael Oliveira, Amir Shpilka, and Ben Lee Volk. Subexponential size hitting sets for bounded depth multilinear formulas. In *30th Conference on Computational Complexity (CCC)*, pages 304–322, 2015.

[RS05] Ran Raz and Amir Shpilka. Deterministic polynomial identity testing in non-commutative models. *Computational Complexity*, 14(1):1–19, 2005.

[RY09] Ran Raz and Amir Yehudayoff. Lower bounds and separations for constant depth multilinear circuits. *Computational Complexity*, 18(2):171–207, 2009.

[Sax09] Nitin Saxena. Progress on polynomial identity testing. In *Bulletin of the EATCS*, volume 99, pages 49–79, 2009.

[Sax14] Nitin Saxena. Progress on polynomial identity testing-II. In *Perspectives in Computational Complexity*, pages 131–146. Birkhäuser Basel, 2014.

[Sch80] Jacob T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*, 27(4):701–717, October 1980.

[SS11] Nitin Saxena and Comandur Seshadhri. An almost optimal rank bound for depth-3 identities. *SIAM Journal on Computing*, 40(1):200–224, 2011.

[SS12] Nitin Saxena and Comandur Seshadhri. Blackbox identity testing for bounded top-fanin depth-3 circuits: The field doesn't matter. *SIAM Journal on Computing*, 41(5):1285–1298, 2012.

[SW97] Petr Savický and Ingo Wegener. Efficient algorithms for the transformation between different types of binary decision diagrams. *Acta Informatica*, 34(4):245–256, 1997.

[SY10] Amir Shpilka and Amir Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5(3-4):207–388, 2010.

[Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation (EUROSAM)*, pages 216–226. Springer-Verlag, 1979.