Polynomial Identity Testing and Lower Bounds for Sum of Special Arithmetic Branching Programs

A Thesis Submitted

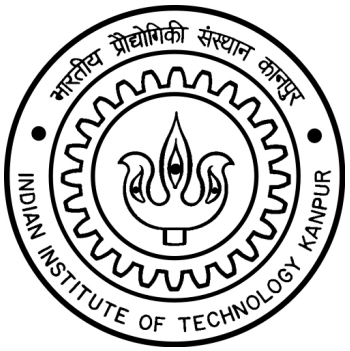
in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

by

Arpita Korwar



to the

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

July, 2016

CERTIFICATE

It is certified that the work contained in the thesis entitled "Polynomial Identity Testing and Lower Bounds for Sum of Special Arithmetic Branching Programs", by "Arpita Korwar", has been carried out under our supervision and that this work has not been submitted elsewhere for a degree.

Manindra Agrawal
Department of Computer Science and Engineering
Indian Institute of Technology Kanpur

Nitin Saxena
Department of Computer Science and Engineering
Indian Institute of Technology Kanpur

July, 2016

SYNOPSIS

Polynomials are fundamental objects studied in mathematics. Though univariate polynomials are fairly well-understood, multivariate polynomials are not. Arithmetic circuits are the primary tool used to study polynomials in computer science. They allow for the classification of polynomials according to their complexity.

Polynomial identity testing (PIT) asks if a polynomial, input in the form of an arithmetic circuit, is identically zero. Though the PIT problem is interesting in itself, it is also important in computational complexity. E.g. it is related to lower bounds.

One way to check whether the polynomial computed by the circuit is zero is to set its variables to constants. If the circuit evaluates to a nonzero value, we definitely know that the polynomial is nonzero. But, what if the circuit evaluates to zero at this point? A blackbox PIT is a set of points, such that if the polynomial is nonzero, then it evaluates to a nonzero constant at at least one of these points. A whitebox PIT algorithm is allowed to make use of the internal structure of the circuit.

We don't yet have a deterministic polynomial time algorithm for PIT. But we do have an exponential time algorithm, obtained by unrolling the legendary Schwartz-Zippel lemma. Can we get a deterministic PIT algorithm which takes less than exponential time?

An arithmetic branching program (ABP) is an iterated product of polynomial matrices. It is a model for computing polynomials and is a special arithmetic circuit. It is known to be equivalent to the determinant. The important complexity-theoretic parameters related to an ABP are its number of variables, n, the width of the ABP, w, the degree of the variables d and the size of the ABP. The width of the ABP is the maximum dimension of any matrix in the iterated matrix product.

One special kind of ABPs are *read-once oblivious* ABPs (ROABPs), which can be written as a product of univariate polynomial matrices over distinct variables. Besides the properties inherited from the ABP, the ROABPs are also parameterized by π, the sequence of the variables.

Sum of ROABPs: We give the first polynomial time ($n^{O(1)}$) whitebox PIT and the first quasi-polynomial time ($n^{\log^{O(1)} n}$) blackbox PIT for the sum of constantly many ROABPs. Though a polynomial time whitebox algorithm and a quasi-polynomial time blackbox algorithm was known for an ROABP, only a trivial, exponential time PIT was known for the sum of two ROABPs before this work. The sum of two ROABPs is provably a stronger model than a single ROABP. We match the complexity of identity testing of the sum of constantly many ROABPs with that of a single ROABP. Both of our algorithms are doubly exponential in the number of ROABPs.

The basic idea we use is that the dependencies amongst the 'partial evaluations of the polynomial' (polynomials obtained by substituting various constants for a subset of the variables) essentially define the ROABP computing the polynomial. The proof of the blackbox PIT shows that 'low-support rank concentration' in a polynomial can be achieved by shifting it by a 'basis isolating weight assignment'.

Sparse, Invertible-factor ROABPs: We next study a natural restriction that can be applied to the ROABPs: that the matrix layers of the ROABP are *invertible*. Now, a layer can have more than one variable, but the ABP is still read-once. We give a $(n\delta s)^{O(w^2 \log w)}$ time PIT for this model, where δ is the degree of each polynomial layer (which is a matrix) and s is its sparsity.

It is known that the PIT of depth-3 circuits, which are general enough, reduces to the PIT of width-2 ABP with invertible layers. Also, polynomial-sized formulas reduce to width-3 invertible-factor ABPs. So, invertible-factor ABPs are strong enough. But, the 'read-once' restriction weakens the model.

When the layers are univariate, we get a $(n\delta)^{O(w^2)}$ time blackbox PIT. Further, we could do away with the invertibility restriction for width-2 ROABPs. We obtain a poly-

nomial time PIT for a width-2 ROABP.

Towards impossibility results for the sum of two width-2 invertible ABPs: It was known that the polynomial $f = x_1x_2 + x_3x_4 + \cdots + x_{15}x_{16}$ cannot be computed by a width-2 ABP with linear polynomials as its entries. We address the question of whether the polynomial f can be computed as a sum of two width-2 ABPs. We make partial progress towards this question by addressing the question:

> *Can f be computed as a sum of two width-2 ABPs, A and B, where the determinant of each layer in the ABP is a nonzero field constant?*

With a width-2 ABP, A, where the determinant of each layer in the ABP is a nonzero field constant, we can associate a set, L_A, of linear forms (linear forms are homogeneous linear polynomials. I.e. they have no constant term). We show that if $A + B = f$, then, the sets of linear forms, L_A and L_B, have to be equal. For a subset S of L_A, we naturally define the polynomial $A|_S$, the polynomial computed by the ABP A, restricted to the subset of linear forms S. It could be viewed as a 'derivative of the ABP' with respect to the complement of S. We show that if $A + B = f$, then, for *almost all* subsets S of $L_A = L_B$, $A|_S + B|_S = 0$.

This is a partial progress towards proving that f cannot be computed as a sum of two width-2 ABPs.

Acknowledgements

This is my thesis. But its existence would be very difficult if not for the constant support of the people around me.

Thank you, Somenath Biswas Sir, for meeting with me and listening to my ideas, even though you were never my guide officially. You have always been my go-to person.

Thank you, Manindra Sir, for guiding me for all these years. You have time and again extended yourself for my sake. I was very grateful when Coco died and to help me, you invoked God. It is tough for an academic to talk about God in the presence of a student. You attended all my presentations. When I had called you for some work, you told me what to do and then said, "If need be, I will come tomorrow". After processing that for some time, I realized that the next day was the main day of Diwali! You have suffered my faults with me and have only encouraged me.

Thank you, Nitin, for all the meetings and discussions and for trying to teach me algebra. Thank you for spending so much time with me. Thank you for reading my thesis.

Thank you, Rohit, for being a good listener. You have influenced my work so much... even while writing the thesis, I thought about you every day (almost :-)). "We had discussed this in that canteen", "Rohit had been so enthusiastic about this", "I had called Rohit on the phone for this", and so on. Thank you for being around.

Thank you, Prof. Thierauf, for inviting me to the Ulm University twice and for being so generous with your time when I was there and when you were here at IITK. When I said, "I don't want to work on matching, I will work on PIT", you said OK, and then we worked on PIT. It meant a lot to me.

Thank you, Jochen, for your infectious enthusiasm and thoroughness. You have caught

so many defective proofs. :)

Thank you, Simon Straub, for the discussions.

Thank you, Amit Sinhababu, for all the doubts. I am very grateful for all the *gyan* I could give you. Thank you for reading a chapter in this thesis.

Thank you, Kartik Kale for proof-reading a chapter in this thesis and the feedback.

Thank you, Rishabh Vaid, for being you. When I was your TA, you corrected me! You did your own thing... like the wall-hanging problem. Your answers were different, but insightful.

Thank you, Chandan, for hosting me at IISc. I learnt a lot during that time. Your sincerity is inspiring.

Thank you, Ramprasad, for being you! You are so enthusiastic about maths. *Whatever* doubts I had, nothing could stump you. Not even for 5 seconds! In fact, you are that way about everything, not just maths. You are one-of-a-kind and I am lucky that you were my senior at IITK.

Thank you to Aurko Roy and Mrinalkanti Ghosh for awesome discussions and talks.

Thank you to the CSE department at IITK for the financial support. This department has been home to me for all these years. Thank you for the unconditional support throughout.

Thank you, Ganguly Sir, SSax Sir, TVP Sir, Prof. S.K. Mehta, Raghunath and Satyadev for always being kind to me and for constructive feedback.

Thank you, Sharmistha Ma'am, the counsellor at IITK. Over the years, I have come to you so many times. You have always listened to me patiently, helping me clear my thoughts and even shared your own experiences. The time you spent with me meant a lot to me. My PhD (and M.Tech) would have been very different if you had not been around.

Thank you to my lab-mates, especially Sudhanshu, Purushottam and Umair for the *bulla* sessions.

Thank you, Mradula, Monica and Abhishek of CCD for the awesome coffee and the conducive environment.

Thank you to the Paws family - Bhagat, Kajree, Custard, Dora, Oreo, Coco, Butter

and Muddy for your unconditional love. And to their human caregivers - Pallavi, Gagan, Amrita, Sanjukta and Esha. When I said that I don't want to continue contributing to the good (but hard) work we are doing, you said OK. In fact, you encouraged me. I am grateful that you let me go.

Thank you to the Taekwondo club and the Yoga class at IITK.

Thank you to the Jorapur family for the support and advice over the years.

Thank you, Amma and Amrita. You are my life.

And finally, thank you Raghavendra Swami. :)

Contents

Chapter 1

Introduction

1.1 Polynomial identity testing (PIT)

Definition and applications: *Polynomial identity testing* asks if a given n-variate polynomial is identically zero. I.e. whether all the coefficients of the given polynomial are zero. We know that $(x+y)^2 - x^2 - y^2 - 2xy$ is zero because we have seen the expansion of $(x+y)^2$ in high school. It may take us a little more time to check if the polynomial $(ux+vy)^2 + (uy-vx)^2 - (u^2+v^2)(x^2+y^2)$ is zero or not.

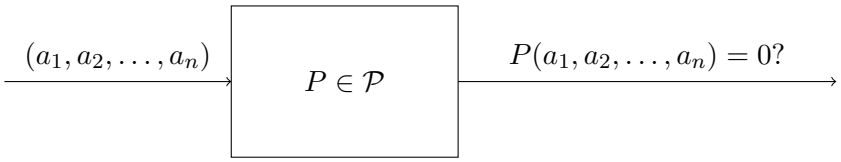
Though the PIT problem is natural and interesting in itself, it is also important in computational complexity. E.g. it is related to lower bounds [KI04, Agr05, HS80, DSY09]. Also, many important problems in computer science reduce to solving PIT for special classes of polynomials. E.g. a *perfect matching* exists in a graph G if and only if the determinant of a related matrix M_G (which has polynomial entries) is not identically zero [Lov79, Tut47, MVV87]. The non-trivial direction of $\mathsf{IP} = \mathsf{PSPACE}$ is to show that $\mathsf{PSPACE} \subseteq \mathsf{IP}$. Since TQBF is PSPACE-complete, it is enough to show that the truth value of a given quantified Boolean formula can be determined by an IP system. This is done by reducing the given quantified Boolean formula to a series of polynomial identity tests [LFKN92, Sha92]. Checking if a number n is prime reduces to checking if a related polynomial f_n is identically zero in an appropriate ring [AKS04]. Checking if a read-once Boolean branching program A with input of length n accepts any string reduces to checking

if a related multilinear polynomial f_A is identically zero (see, for example, [Thi98]).

Form of the input to the PIT algorithm: *How* the polynomial is *input* plays a big role in the design of the PIT algorithm. If the polynomial was input as a list of coefficient-monomial pairs, then PIT is in linear time and trivial. It is not always efficient to expand the given polynomial and reduce it to this form. E.g. the polynomial $f(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n}(1 + x_i)$ is a simple polynomial. It is obtained by adding 1 to each variable in the polynomial $g(x_1, x_2, \ldots, x_n) = x_1 x_2 \cdots x_n$. But, it has 2^n terms. So, we cannot expand the polynomial efficiently.

However, it is easy to compute an evaluation of the polynomial by substituting the variables with constants. If the polynomial evaluates to a nonzero value, we definitely know that the polynomial is nonzero. But, if it evaluates to zero, we cannot say that it is the zero polynomial. E.g. the polynomial $x - y$ evaluates to zero at infinite number of points: $\{(1, 1), (2, 2), \ldots\}$. The points where a polynomial evaluates to zero are called as its *roots*. DeMillo-Lipton [DL78], Schwartz [Sch80] and Zippel [Zip79] proved that the roots of a nonzero polynomial, when picked randomly from suitable sets, occur with small probability. This gives us a randomized algorithm for PIT: pick a point from a suitable set randomly and evaluate our polynomial at that point. It puts the problem of PIT in the complexity class co-RP. This was the first randomized algorithm for PIT. Subsequently, there were other randomized PIT algorithms which traded the number of random bits with the time required or the error parameter [CK00, LV98, AB03, KS01]. Since all problems with randomized polynomial-time solutions are conjectured to have deterministic polynomial-time algorithms (see [AB09, Chapter 16]), we expect that such an algorithm exists for PIT.

For a family of polynomials $\mathcal{P}$, a *blackbox PIT* or equivalently, a *hitting set* is a set of evaluation points $\{\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_s\}$, such that for every nonzero polynomial $P \in \mathcal{P}$, given as a blackbox, there is at least one evaluation point $\boldsymbol{a}$ such that $P(\boldsymbol{a}) \neq 0$. The PIT algorithm is not allowed to access the structure of the circuit.



For example, let $\mathcal{P}$ be the set of all univariate polynomials of degree bounded by d. Then, any polynomial in $\mathcal{P}$ can have $\leq d$ roots. So, the set $\{a_1, a_2, \ldots, a_{d+1}\}$ of $d+1$ distinct points acts as a hitting set for $\mathcal{P}$.

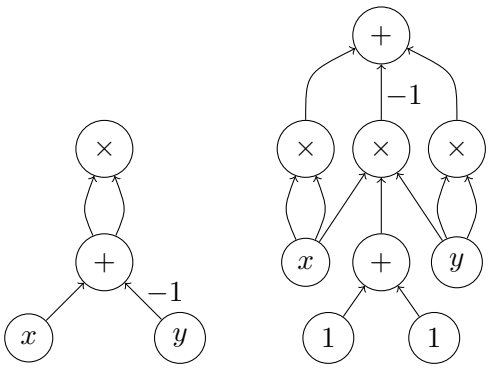
In contrast, a *whitebox PIT* algorithm is allowed to make use of the internal structure of the polynomial. This brings us back to the question: how is the polynomial input to the algorithm. The internal structure of the polynomial available to us highly depends on how the polynomial is input to the algorithm.

Arithmetic circuits: *Arithmetic circuits* are the most natural model for polynomial computation. They are the primary tool used to study polynomials in computer science. They allow for the classification of polynomials according to their complexity.

Arithmetic circuits are defined over a field $\mathbb{F}$. They are directed acyclic graphs, where every node is a '+' or '×' gate and each input gate is a constant from the field $\mathbb{F}$ or a variable from $\boldsymbol{x} = \{x_1, x_2, \ldots, x_n\}$. The *syntactic degree* of an input gate is 1. Every edge has a weight from the underlying field $\mathbb{F}$. The computation is done in the natural way, starting with the input nodes and proceeding towards the output node. At each step, the weight of the edge (u, v) is multiplied with the output of the previous gate u and then input to the next gate, v. So, every gate of the circuit computes a polynomial in $\mathbb{F}[\boldsymbol{x}]$. The syntactic degree at a '+' node is the max of the syntactic degrees of the polynomials input to it. The syntactic degree at a '×' node is the sum of the syntactic degrees of the polynomials input to it. An *output node* has no outgoing edges. The polynomial computed at this node is said to be the polynomial computed by the circuit.

The *size* of the circuit is the number of edges in it. The *degree* of the circuit is the syntactic degree of the output gate. The *depth* of the circuit is the number of nodes in a longest path from an input node to the output node. The size, the degree and the depth are used to calculate the complexity of the circuit.

E.g. Both of the following arithmetic circuits compute $(x-y)^2$.



Size = 4	Size = 12
Depth = 3	Depth = 4
Degree = 2	Degree = 3

Without loss of generality one can assume that the addition (+) and multiplication (×) gates in an arithmetic circuit appear in alternate layers.

Lower bounds and the complexity classes VP and VNP: Arithmetic circuits are a very important part of computational complexity, because of the VP Vs VNP question. VP is the algebraic analogue of the complexity class P and VNP is the algebraic analogue of the counting class #P. VP is the class of polynomials with polynomial degree that are computed by polynomial size arithmetic circuits. VNP is the class of polynomials $\{p_n\}_{n>0}$ such that there exists a polynomial family $\{q_n\}_{n>0} \in \mathsf{VP}$ such that for all n:

$$p_n(x_1, x_2, \ldots, x_n) = \sum_{e \in \{0,1\}^m} q_{m+n}(x_1, x_2, \ldots, x_n, e_1, e_2, \ldots e_m),$$

where m is polynomial in n.

Under the assumption of generalized Riemann hypothesis, if VP = VNP over any field, then P/poly = NP/poly [Bür00]. But the implication in the other direction is not known to hold. We would like to prove that VP $\neq$ VNP; it is an easier version of the P Vs NP question. So, we want to know if there are *lower bounds* for polynomials in VNP. I.e. if there are polynomials in VNP that don't have polynomial sized circuits. Over the past few years, there has been tremendous interest and progress in this question. We now know that lower bounds for arithmetic circuits give polynomial identity tests and

vice versa. [KI04] says that lower bounds for the permanent polynomial imply that sub-exponential time PIT exists for arithmetic formulas. They also prove that if we have a polynomial-time whitebox PIT for polynomials in VP, then we are guaranteed that at least one lower bound exists - an arithmetic lower bound or a Boolean lower bound; the permanent polynomial is not in VP or $\mathsf{NEXP} \not\subseteq \mathsf{P/poly}$. We expect both these lower bounds to be true. [Agr05, HS80] show that under stronger assumptions - a polynomial-time blackbox PIT, a stronger lower bound can be proved. [DSY09] shows connections between PIT and lower bounds for bounded depth circuits.

We also know that to prove lower bounds for arithmetic circuits over fields of characteristic 0, it is enough to prove strong enough lower bounds for depth-3 circuits [GKKS16].

History of PIT: [GKKS16] actually showed that a polynomial computed by a polynomial sized circuit over a field of characteristic zero can also be computed by a depth-3 circuit of sub-exponential size. This shows that PIT for depth-3 circuits is a very important question. Polynomial-time PIT for depth-3 circuits would imply sub-exponential time PIT for a general circuit.

For PIT of arithmetic circuits computing polynomial $f(\boldsymbol{x})$, we can always assume that the top gate is '+'. If it was a '$\times$' gate, then the polynomial f is a product of polynomials $f = f_1 f_2 \cdots f_k$. Then, PIT for these factor polynomials would give a PIT for f, the product. If we want a blackbox PIT for $f = f_1 f_2 \dots f_k$, then, using the blackbox PIT for the f_is and with the idea of 'interpolation', a blackbox PIT for the product polynomial f can be obtained.

We don't yet have a deterministic polynomial-time algorithm for PIT. But we do have an exponential-time algorithm, obtained by de-randomizing the Schwartz-Zippel lemma. So, a valid open question is whether we can give a deterministic polynomial-time algorithm for PIT. Or even, any deterministic PIT algorithm which takes sub-exponential time. An efficient deterministic solution for PIT is known only for very restricted input models.

One of the first non-trivial PIT algorithms was for depth-2 circuits. They are also known as $\Sigma\Pi$ circuits; Σ signifies the top '+' gate and Π signifies the layer of '$\times$' gates

which feed to it. If the fan-in of the top '+' gate is k (denoted by $\Sigma^{[k]}\Pi$), then these circuits compute k-sparse polynomials - polynomials with $\leq k$ terms. They have a polynomial-time hitting set [BOT88, KS01, AB03].

The class of polynomials for which PIT would be most interesting is the class of depth-3 circuits because general circuits reduce to it. They are $\Sigma\Pi\Sigma$ circuits and compute sum of product of linear polynomials. They have a $(nd)^{k^{O(1)}}$ size hitting set, where k is the top fan-in and d is the degree of the circuit [DS07, KS07, KS09, KS11, SS11, SS12, SS13]. So, when the top fan-in k is constant, it has a polynomial-time algorithm.

Read-k multilinear formulas are formulas where a variable is read only k times. They have a $n^{k^{O(k)}}$ whitebox PIT algorithm and a $n^{\log n + k^{O(k)}}$ time blackbox test [AvMV15]. Multilinear depth-4 circuits ($\Sigma\Pi\Sigma\Pi$ circuits or sum of products of sparse polynomials) have a $n^{k^{O(1)}}$ time hitting set [KMSV13, SV11].

There are other restricted arithmetic circuits: *set-multilinear circuits*, *ROABP*s, etc. for which non-trivial hitting sets and white box PIT are known. We will study these in detail throughout this thesis. See also the surveys [Sax09, Sax14, SY10].

1.2 Arithmetic branching programs (ABP)

Arithmetic branching programs (ABP) are special arithmetic circuits. An arithmetic branching program is a directed layered graph with a source and a sink. The edges starting at layer i go only to layer $i+1$ and are labelled by 'simple' 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. The *size* of the ABP is the number of nodes in it. The *width* of the ABP is the maximum number of nodes in any layer.

ABPs are equivalent to iterated matrix multiplication and symbolic determinant. I.e. any size s ABP can be written as a size s iterated matrix multiplication and vice versa. A size s ABP can be written as a size s symbolic determinant. The symbolic determinant of a $s \times s$ matrix can be written as a size s^3 ABP [MV97].

Also, any circuit in VP (polynomial size circuit with polynomially large syntactic degree) can be reduced to a quasi-polynomial size ABP[1]. Thus, ABP is VP-complete under quasi-polynomial reductions.

The power of the ABPs lies between that of arithmetic circuits and arithmetic formulas. I.e. any polynomial computed by an arithmetic formula of size s can be computed by an ABP of size s and any polynomial computed by an ABP of s_1 nodes and s_2 edges can be computed by an arithmetic circuit of size $s_1 + s_2$. Out of all the children of the multiplication gates in these arithmetic circuits obtained from ABPs, only one can be a non-input node. [LMS15] have studied it in the non-commutative model. See [Raz05]'s lecture notes for a good overview. We do not know if these inclusions are strict.

Recall that the width of an ABP is the maximum of the number of nodes that can occur at any layer of the ABP. It is known that the class of width-2 ABPs is an incomplete model [AW16]. I.e. there are polynomials that cannot be computed by width-2 ABPs, irrespective of the size of the ABP. All polynomials can be computed by width-3 ABPs - each term of the polynomial is computed one at a time and added to the polynomial. One line carries the sum of all the terms computed, the second line is used as a workspace to compute a term and the third line is used to initialize the second line to 1 at the start of computing each monomial. Also, constant-width ABPs are equivalent to logarithmic-depth formulas, or equivalently, polynomial-sized formulas. I.e. a log-depth formula is obtained from a constant-width ABP by a divide-and-conquer method and a log-depth formula can be converted to a constant-width ABP [BOC92]. Thus, the width of an ABP is an important parameter to consider while studying the power of ABPs.

Let us now study a restricted subclass of general-width ABPs, which are known to be weaker than ABPs.

[1]A function $f(n) = 2^{\log^c n}$ is *quasi-polynomial* in n if $c > 1$ is a constant. If $c = 1$, $f(n)$ is polynomial in n.

1.3 Read-once Oblivious ABP (ROABP)

A read-once oblivious ABP is an ABP such that each variable occurs in only one layer[2]. With a small overhead, a given ROABP can be converted so that only one variable occurs in a layer. Thus, it can be written as a product of univariate polynomial matrices over distinct variables. A permutation on the variables $\boldsymbol{x}$ can be associated with any ROABP. Hence, the defining properties of an ROABP are n, its number of variables, π, the sequence of these variables, d, the degree of these variables and w, the width, which is the maximum dimension of any matrix.

Besides the whitebox and the blackbox settings, ROABP allows for a hitting set when the sequence of the variables is known, but we cannot see the actual edge weights. This, we shall call as the *grey-box* setting.

There has been a long chain of work on identity testing for ROABP. [RS05] gave a polynomial-time whitebox algorithm for the identity testing of ROABPs. In [FS13b], Forbes and Shpilka gave a $n^{O(\log n \log w)}$ i.e. quasi-polynomial-time grey-box hitting-set. In [FSS14], Forbes, Saptharishi and Shpilka gave a $(ndw)^{O(d \log^2 n)}$ i.e. quasi-polynomial-time hitting-set for the blackbox setting. So, there is an exponential dependence on the individual degree, d. Their basic idea is that a shift of the variables condenses the rank of the coefficients of all monomials into the coefficients of small monomials. This was further improved by Agrawal et al. [AGKS15] to $(ndw)^{O(\log n)}$ time. They removed the exponential 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.

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.

[2]In a *read-once* ABP, each variable occurs at most once on every path from the source to the sink. A read-once *oblivious* ABP has the property that every variable occurs in only one layer.

Applications of ROABP: ROABP is one of the few models for which a polynomial-time identity test is known in the whitebox regime, but not in the blackbox regime. It also has a lot of structure to it, as we will see in this thesis. Hence, a blackbox PIT algorithm for ROABP is expected in the next few years.

Arithmetic branching programs have been defined on the lines of (Boolean) branching programs, also known as Binary Decision Diagrams. A Boolean branching program is also a directed acyclic graph with a 'start' vertex. But the out degree of all vertices is 2. Each vertex is labelled with a variable and one outgoing edge from that vertex has label 0 and the other outgoing edge from that vertex has label 1. Given an input Boolean string $b_1 b_2 \cdots b_n \in \{0,1\}^n$, at a vertex with label x_i, we take the edge which has the value b_i. There are two vertices - *terminal vertices* - which have no outgoing edges. They are the 'accept' and 'reject' vertices. We accept the string $b_1 b_2 \cdots b_n$ if the 'accept' vertex can be reached from the start node by reading this input string $b_1 b_2 \cdots b_n$.

In a *reduced ordered binary decision diagram*, an order is associated with the Boolean variables and the variables occur in the binary decision diagram in this order. Moreover, all the isomorphic sub-graphs of the binary decision diagram are merged. So, for every input string, there is a single path from the start node to one of the terminal nodes. Checking if a reduced ordered binary decision diagram accepts any string reduces to PIT of a multilinear ROABP. Similarly, the equality of two read-once Boolean branching programs reduces to the equality of two ROABPs.

The branching programs with 'read-once oblivious' property are known as Oblivious Boolean Decision Diagrams (OBDDs). A $\log n$ seed length pseudo-random generator for OBDDs would de-randomize RL. It would bring it into the complexity class L. Presently, only a $O(\log^2 n)$ seed length pseudo-random generator is known for OBDDs [RR99]. A polynomial-time hitting set for ROABPs may give us new directions towards improving this bound.

Recall that a polynomial computed by any circuit in VP can be computed by a depth-3 circuit of sub-exponential size. Hence, finding a PIT algorithm for depth-3 circuits almost solves the general case. But 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. Hence, each product gate in a depth-3 multilinear circuits induces a partition on the variables.

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 k depth-3 set-multilinear circuits, where k is the fan-in of the top addition gate. [RS05] gave the first polynomial-time whitebox PIT for set-multilinear circuits. [ASS13] gave the first quasi-polynomial-time hitting set for set-multilinear circuits using the concept of low-support rank concentration, which they introduced in this paper.

[AGKS15] gave a sub-exponential whitebox algorithm for multilinear circuits when the top fan-in is constant. Then, [dOSV16] gave an general sub-exponential time blackbox test for depth-3 multilinear circuits.

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. Adding such width-2 ROABPs, set-multilinear circuits with top fan-in k reduce to ROABPs of width $2k$.

Diagonal circuits ($\sum \bigwedge \sum$) compute sum of powers of linear polynomials, i.e. polynomials of the form $\sum_{i=1}^{k} (\ell_i)^{d_i}$. After the polynomial-time hitting set for depth-3 circuits with constant top fan-in, these were the 'simplest' depth-3 circuits open to attack. There are exponential lower bounds known for this circuit family [NW96, Sax08]. The proof uses the partial derivative method and has one of the most elementary lower bound proofs. So, it is used to introduce lower bounding techniques. [Sax08] gave the first polynomial-time whitebox PIT for this model. [FSS14] a blackbox PIT which takes $n^{O(\log \log n)}$ time. Before that, a $n^{O(\log n)}$ time blackbox PIT algorithm was known [ASS13, FS13b, FS13a] . Diagonal circuits reduce to ROABPs through Saxena's trick [Sax08].

Though interesting models like set-multilinear circuits and diagonal circuits reduce

to ROABPs, the ROABPs are not a very strong model. Kayal et al. [KNS16] showed a polynomial f that can be computed by a small multilinear circuit, but any ROABP that computes f would take exponential size.

1.4 Contribution of this thesis

In this thesis, we work with ROABPs and width-2 ABPs.

1.4.1 Sum of ROABPs (Chapter 3)

[RS05] gave a polynomial-time white box PIT for set-multilinear circuits. The same algorithm works for a generalization of set-multilinear circuits: the ROABPs. [ASS13] gave a quasi-polynomial-time hitting set for set-multilinear circuits.

In the whitebox arena, there was a $2^{O(\sqrt{n})}$-time PIT for sum of two set-multilinear circuits [AGKS15]. But, this time bound degraded rapidly with PIT for sum of more set-multilinear circuits. Then, de Oliveira, Shpilka and Volk [dOSV16] gave a blackbox $2^{\tilde{O}(n^{2/3})}$-time PIT for multilinear circuits, And, in effect for sum of two multilinear circuits.

In this thesis, 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 ROABPs. In fact, it holds for the sum of constantly many ROABPs.

As we already saw, there is a polynomial-time whitebox PIT [RS05] and a $(ndw)^{O(\log n)}$ time blackbox hitting set for an ROABP [AGKS15]. [KNS16] showed that the sum of two ROABPs is strictly stronger than a single ROABP. Hence, the results of [RS05] or [AGKS15] cannot be used for the sum of two ROABPs. However, our algorithm for the sum of constantly many ROABPs matches their respective time complexities:

> *There is a* $\mathsf{poly}(n, d, w^{2^c})$ *time whitebox PIT algorithm and a* $(ndw)^{O(c2^c \log(ndw))}$ *time hitting set for any* n*-variate polynomial computed as the sum of* c *ROABPs, each of width* w *and individual variable degree* d*.*

Consequently, [AFS$^+$16] gave a sub-exponential time whitebox PIT for a generalization

of this model: the read-c ABP. Their algorithm takes $2^{\tilde{O}\left(n^{1-1/2^{c-1}}\right)}$ time. So, it takes worse time. On the other hand, they solve a more general case. The algorithm presented in this thesis is the best known for sum of ROABP.

We want to test if an n-variate polynomial, given as sum of c ROABPs, $A_1 + A_2 + \cdots + A_c$, each of width w and with potentially different permutations on the variables, is identically zero. A PIT for this model is equivalent to testing if $A_1 \stackrel{?}{=} A_2 + A_3 + \cdots A_c$. Let us first consider the question $A \stackrel{?}{=} B$.

Whitebox test

Our algorithm uses the fact that the *evaluation dimension* of an ROABP is equal to the width of the ROABP [Nis91, FS13a]. 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. We also use the fact that the 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 3.3.2, 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.

Blackbox test

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 $< \ell$ (Support of a monomial is the set of variables that occur non-trivially in that monomial. E.g. Support of ${x_1}^{500}$ is $\{x_1\}$. Support of ${x_1}^5 x_2$ is $\{x_1, x_2\}$). 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. This is surprising because, as mentioned above, a sum of c ROABPs is not captured by an ROABP with polynomially bounded width [KNS16].

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.

1.4.2 Sparse-Invertible-Factor ROABP (Chapter 4)

The next model we study is an ROABP with a natural restriction: invertibility. The input is of the kind: $A = M_0 M_1 M_2 \cdots M_n M_{n+1}$ is an ROABP, where M_0 and M_{n+1} are constant vectors and $M_i \in \mathbb{F}[x_{\pi(i)}]^{w \times w}$ is invertible for all i. Here, π is a permutation on $[n]$.

Ben-Or and Cleve [BOC92] reduce formulas to width-3 ABP with *invertible* factors.

Saha, Saptharishi and Saxena [SSS09] reduce PIT for depth-3 circuits to PIT for width-2 ABP with invertible factors. So, efficient PIT for constant-width ABPs with invertible layers would give efficient PIT for formulas and depth-3 circuits. This would give sub-exponential time PIT for all polynomials in VP [GKKS16]. It seems that the 'read-once, oblivious' restriction weakens the model, because of which, we can find a hitting set. Though the model is weakened, it includes the interesting class of diagonal circuits. Saxena's trick [Sax08] reduces diagonal circuits (sum of power of linear polynomials) to ROABPs with invertible factors of polynomial width.

The result we prove is:

> *There is a $(n\delta)^{O(w^2)}$-time hitting-set for the class of n-variate invertible ROABPs of width w and individual degree δ.*

We basically do this by achieving w^2-support concentration.

Any monomial of k support, when viewed as a word with the variable ordering $x_{\pi(1)} \prec x_{\pi(2)} \prec \cdots \prec x_{\pi(n)}$, has $k+1$ prefixes and 2^k substrings. Let $\text{coeff}(m_1)$ be a prefix of $\text{coeff}(m_2)$. We show that if $\text{coeff}(m_1)$ is linearly dependent on substrings of $\text{coeff}(m_1)$, then using this dependency and the invertibility condition, $\text{coeff}(m_2)$ is linearly dependent on substrings of $\text{coeff}(m_2)$. Using this, we can prove (w^2+1)-concentration among the coefficients.

We actually give a hitting set when each layer of the invertible-factor ROABP may have more than one variable. We work with the model $A = M_0 M_1(\boldsymbol{x}_1) M_2(\boldsymbol{x}_2) \cdots M_d(\boldsymbol{x}_d) M_{d+1}$, where $M_0 \in \mathbb{F}^{1\times w}, M_{d+1} \in \mathbb{F}^{w\times 1}$ as before. M_is are invertible for all i, as before. But now, M_is are possibly over more than one variable. But the ROABP is still read-once, oblivious. I.e. there is a *partition* of the variables $\boldsymbol{x} = \boldsymbol{x}_1 \sqcup \boldsymbol{x}_2 \sqcup \cdots \sqcup \boldsymbol{x}_d$ such that M_i is a matrix polynomial over variables in $\boldsymbol{x}_i$. We call this the class of sparse invertible ROABPs. We saw that with a small increase in the width, the ABP can be written as a product of univariates. But, our algorithm gives a better time complexity than when we convert it to univariates and apply the hitting set from above.

> *There is a $(n\delta s)^{w^2 \log w}$-time hitting-set for the class of n-variate s-sparse*

invertible ROABPs of width w and individual degree δ.

In this method, we have to concentrate the coefficients within a matrix M_i to coefficients of low-support monomials.

1.4.3 Sum of two width-2 ABPs (Chapter 5)

Our third contribution concerns the sum of width-2 ABPs. We study the question:

Is there a polynomial which cannot be computed as a sum of two width-2 ABPs?

Width-2 ABP is interesting because an efficient PIT for width-2 ABP would imply an efficient PIT for depth-3 circuits [SSS09], and hence, effectively, for all polynomials in VP [GKKS16].

Allender and Wang [AW16] showed that $x_1x_2+x_3x_4+\cdots+x_{15}x_{16}$ cannot be computed by a single width-2 ABP; it is an *incomplete* model. They show that if a general width-2 ABP computes $x_1x_2+x_3x_4+\cdots+x_{15}x_{16}$, then, a width-2 ABP where the determinant of each layer of the ABP is a nonzero field constant[3] computes $x_1x_2+x_3x_4$. They then show that the highest degree homogeneous part[4] of any polynomial computed by an invertible width-2 ABP can always be written as a product of homogeneous linear polynomials. I.e. with any invertible width-2 ABP A, we can associate a multiset L_A of homogeneous linear polynomials, such that the highest degree homogeneous part of the polynomial computed by the ABP A is $\prod_{l\in L_A} l$. Thus, $x_1x_2+x_3x_4$ cannot be computed by any invertible ABP.

In contrast, the family of width-3 ABPs can compute all polynomials. It is a *complete* model. This is analogous to the fact that $\prod\sum$ circuits cannot compute $x_1x_2+x_3x_4$, but the family of $\sum\prod$ circuits can compute all polynomials.

[3]We call this as an *invertible ABP*. If the determinant of any layer in the ABP is a non-constant polynomial, then, it is not invertible. Hence, *this definition for an invertible layer is not the same as that in the previous problem.*

[4]Suppose a given polynomial f has degree d. Let S be the set of all n-variate monomials of degree d. Then, the highest degree homogeneous part of f is

$$\sum_{m\in S} \mathrm{coeff}_f(m)\cdot m.$$

We are interested in this question for the sum of *two* width-2 ABPs. It is easy to see that $x_1x_2 + x_3x_4$ can be computed by a single width-2 ABP. So, $x_1x_2 + \cdots + x_7x_8$ can be written as a sum of two width-2 ABPs.

> *Can the polynomial* $x_1x_2 + x_3x_4 + \cdots + x_9x_{10}$ *be computed as a sum of two width-2 ABPs?*

We will be studying a restriction of this model: the sum of two width-2 *invertible* ABPs. It is easy to see that a width-2 invertible ABP can compute x_1x_2. Hence, the polynomial $x_1x_2 + x_3x_4$ can be computed as a sum of two invertible width-2 ABPs.

> *Can the polynomial* $x_1x_2 + x_3x_4 + x_5x_6$ *be computed as a sum of two invertible width-2 ABPs?*

We first show that if $A + B = f$, then, the sets of linear forms, L_A and L_B, have to be equal.

If the multiset L_A of the homogeneous linear polynomials associated with one of the ABPs A is 3-independent, then, we show that $x_1x_2 + x_3x_4 + x_5x_6$ cannot be computed as a sum of two width-2 ABPs. We actually prove a generalization of this.

For a subset S of L_A, we naturally define the polynomial $A|_S$, the polynomial computed by the ABP A, restricted to the subset of linear forms S. It could be viewed as a 'derivative of the ABP' with respect to the complement of S. We show that,

> *If* $A + B = f$*, then, for* almost all *subsets* S *of* $L_A = L_B$*,* $A|_S + B|_S = 0$*.*

This is a partial progress towards proving that f cannot be computed as a sum of two width-2 ABPs. Using the above result, we hope to prove that if $A + B = f$, then, for $S = L_A = L_B$, $A|_S + B|_S = 0$. I.e. $A + B = 0$, a contradiction.

Chapter 2

Related models and Techniques

In this chapter, we will study some related models of computation and the known identity testing algorithms for them. We will also study a few well-known techniques used in this area. Many of the open problems in the field of 'Polynomial Identity Testing' are to improve the time complexity of the PIT for these models. Let us get familiar with the notations used first.

2.1 Notation

- $\boldsymbol{x}$ will denote the set of variables $\{x_1, x_2, \ldots, x_n\}$.
- Sometimes, there may be a sequence on the variables and we use $\boldsymbol{x}$ for that also.
- By $[n]$, we represent the set $\{1, 2, \ldots, n\}$.
- We will sometimes denote a monomial as $\boldsymbol{a} = (a_1, a_2, \ldots, a_n) \in \mathbb{N}^n$. Then, $\boldsymbol{x}^{\boldsymbol{a}}$ denotes the monomial $\prod_{i=1}^{n} x_i^{a_i}$.
- We will denote by $\text{coeff}_D(\boldsymbol{x}^{\boldsymbol{a}})$, the coefficient of the monomial $\boldsymbol{x}^{\boldsymbol{a}}$ in the polynomial $D(\boldsymbol{x})$.
- coeffs(p) is the set of the coefficients of the polynomial $p(\boldsymbol{x})$. The coefficients of the polynomial $p(\boldsymbol{x})$ could be field elements or vectors or matrices.

- We overload the notation for inner product and say that the inner product $\langle A, B \rangle$ of two matrices $A_{m \times n}$ and $B_{m \times n}$ is defined as $\sum_{(i,j) \in [m] \times [n]} A_{i,j} B_{i,j}$.

- *Polynomials as vectors-* Polynomials can be viewed as vectors; each coordinate corresponds to a monomial. Now, the notions related to linear algebra like vector spaces, span, basis, independent set, etc. carry over to polynomials and can be defined for any set of polynomials.

  In this thesis, the set of n-variate polynomials we will study will have bounded degree. The number of n-variate, bounded degree monomials is finite. Hence, the corresponding vector spaces are finite dimensional.

- We use the terms 'easily computable', 'efficient' and 'small set' to mean polynomial size/polynomial time.

- For a polynomial $f(x_1, x_2, \ldots, x_n), \deg_{x_i}(f)$ is the degree of the variable x_i in the polynomial f.

- A polynomial tuple $\boldsymbol{g} \in \mathbb{F}[y]^n$ is a tuple $(g_1, g_2, \ldots, g_n)$, where $g_i \in \mathbb{F}[y]$.

- $\mathbb{A}_k$ is used to denote a k-dimensional algebra over some field $\mathbb{F}$. Recall that an algebra is a vector space over the field $\mathbb{F}$, endowed with bilinear multiplication.

  We work with the following algebras (vector spaces with multiplication defined on them) in this thesis:

  - The algebra of $w \times w$ matrices over the field $\mathbb{F}$.
  - The k-dimensional Hadamard algebra $\mathbb{H}_k(\mathbb{F})$ over the field $\mathbb{F}$. It is the vector space $\mathbb{F}^k$ endowed with point-wise multiplication. I.e. given two vectors $[a_1, a_2, \ldots, a_k]$ and $[b_1, b_2, \ldots, b_k]$, their product is given by $[a_1 b_1, a_2 b_2, \ldots, a_k b_k]$. It is clear that this multiplication is associative, distributive and commutative. See [Mil07] for a more detailed introduction.

- $\mathrm{Part}(S)$ denotes the set of all possible partitions of the set S. Elements in a partition are called *colors*.

2.2 A randomized algorithm for PIT

Given that this thesis is about PIT, it is fitting that the first lemma we study is the following classical randomized algorithm that puts PIT in the complexity class RP.

Lemma 2.2.1 (Schwartz-Zippel lemma[1])**.** *Let $f(\boldsymbol{x}) \neq 0 \in \mathbb{F}[\boldsymbol{x}]$ be a nonzero degree d polynomial over a field $\mathbb{F}$. Let S be a set of field values of size $> d$. Then,*

$$\Pr_{\boldsymbol{a} \in S^n} [f(\boldsymbol{a}) = 0] \leq \frac{d}{|S|}.$$

Proof. The proof is by induction on n, the number of variables. When $n = 1$, the polynomial is univariate. Hence, it can have $\leq d$ roots.

Induction hypothesis: We assume that for any $n-1$-variate polynomial g of degree d,

$$\Pr_{\boldsymbol{a} \in S^{n-1}} [g(\boldsymbol{a}) = 0] \leq \frac{d}{|S|}.$$

Induction step: We can assume without loss of generality that $\deg_{x_1}(f) > 0$. Let $d' = \deg_{x_1}(f)$. Write $f(x_1, x_2, \ldots, x_n) = \sum_{i=0}^{d'} {x_1}^i \cdot f_i(x_2, x_3, \ldots, x_n)$. Note that $\deg(f_i) \leq d - i$. Then,

$$\begin{aligned}
\Pr_{\boldsymbol{a} \in S^n} [f(\boldsymbol{a}) = 0] &= \Pr_{\boldsymbol{a} \in S^n} [f(\boldsymbol{a}) = 0 \mid \forall i : f_i(\boldsymbol{a}) = 0] \cdot \Pr_{\boldsymbol{a} \in S^n} [\forall i : f_i(\boldsymbol{a}) = 0] \\
&\quad + \Pr_{\boldsymbol{a} \in S^n} [f(\boldsymbol{a}) = 0 \mid \exists i : f_i(\boldsymbol{a}) \neq 0] \cdot \Pr_{\boldsymbol{a} \in S^n} [\exists i : f_i(\boldsymbol{a}) \neq 0] \\
&\leq 1 \cdot \Pr_{\boldsymbol{a} \in S^n} [\forall i : f_i(\boldsymbol{a}) = 0] \\
&\quad + \Pr_{\boldsymbol{a} \in S^n} [f(\boldsymbol{a}) = 0 \mid \exists i : f_i(\boldsymbol{a}) \neq 0] \cdot 1 \\
&\leq \Pr_{\boldsymbol{a} \in S^n} [f_{d'}(\boldsymbol{a}) = 0] + \Pr_{\boldsymbol{a} \in S^n} [f(\boldsymbol{a}) = 0 \mid \exists i : f_i(\boldsymbol{a}) \neq 0]
\end{aligned}$$

By the induction hypothesis, we know that

$$Pr_{\boldsymbol{a} \in S^n} [f_{d'}(\boldsymbol{a}) = 0] \leq \frac{d - d'}{|S|}.$$

Now, suppose there exists $i \in [d']$ such that $f_i(\boldsymbol{a}) \neq 0$. Let $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$. Then,

[1] Though the lemma is commonly called Schwartz-Zippel lemma, DeMillo-Lipton had also given the same bounds. See [DL78, Sch80, Zip79].

the univariate polynomial $f'(x) = f(x, a_2, a_3, \ldots, a_n) \neq 0$. Now, $\deg(f') \leq d'$. Hence,

$$\Pr_{\boldsymbol{a} \in S^n} [f(\boldsymbol{a}) = 0 \mid \exists i : f_i(\boldsymbol{a}) \neq 0] \leq \frac{d'}{|S|}.$$

Thus,

$$\Pr_{\boldsymbol{a} \in S^n} [f(\boldsymbol{a}) = 0] \leq \frac{d - d'}{|S|} + \frac{d'}{|S|} = \frac{d}{|S|}.$$

□

Thus, we need that that the field size is $> d$.

Also, this bound is tight. That is because we can find univariate polynomials which have d unique roots.

We can de-randomize the above lemma by evaluating the polynomial at all the points in S^n. And we can take S to be any set of $d + 1$ distinct field elements. Observe that we only need blackbox access to the polynomial for this test to work.

We also use a special case of Alon's Combinatorial Nullstellensatz.

Lemma 2.2.2 ([Alo99])**.** *Let $f(\boldsymbol{x}) \neq 0 \in \mathbb{F}[\boldsymbol{x}]$ be a nonzero* individual *degree d polynomial over a field $\mathbb{F}$. Let S be a set of field values of size $> d$. Then, there exists a point $\boldsymbol{a} \in S^n$ such that $f(\boldsymbol{a}) \neq 0$.*

2.3 Basics and some common techniques

2.3.1 Hitting sets

Definition 2.3.1. *A set $\mathcal{H} \subseteq \mathbb{F}^n$ 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 $\boldsymbol{a} \in \mathcal{H}$ such that $A(\boldsymbol{a}) \neq 0$.*

Is is well known that a hitting set is equivalent to blackbox testing. For a hitting-set to exist, we will need enough points in the underlying field $\mathbb{F}$. E.g. the polynomial $x^2 + x$ over the field $\mathbb{F}_2$ is a nonzero polynomial. But it evaluates to 0 at every point in the field. Hence, to find enough points for a hitting set, we may have to go to a large enough field extension. Henceforth, we will assume that the field $\mathbb{F}$ is large enough such that the constructions below go through (see [AL86] for constructing large $\mathbb{F}$).

2.3.2 Kronecker substitution

The following lemma is a classic tool.

Lemma 2.3.2 (Kronecker map [Kro82]). *Given a set of k monomial pairs $\{(m_i, m_i')\}_{i=1}^{k}$ where each monomial is over n variables $\{x_1, x_2, \ldots, x_n\}$, and has individual variable degree $< d$, there exists a prime $p \leq N \log N$ such that $\psi_p(m_i) \neq \psi_p(m_i')$ for all $i \leq k$, under the map $\psi_p : x_i \mapsto (y^{d^{i-1}} \mod (y^p - 1))$. Here, $N = nk \log d$.*

This gives a polynomial time mapping that separates all the given monomials pairs and thus, a polynomial size hitting set for a $\sqrt{k}$-sparse polynomial with bounded degree.

Proof. Let M be the set of n-variate monomials with individual variable degree d. Consider the monomial map $\psi : M \longrightarrow \left\{y^0, y^1, \ldots, y^{d^n-1}\right\}$ defined as $\psi : x_i \longmapsto y^{d^{i-1}}$. It maps a monomial $\prod_{i=1}^{n} {x_i}^{a_i}$ to $y^{\sum_{i=1}^{n} a_i \cdot d^{i-1}}$. Since $a_i < d$ for all i, $(d_1, d_2, \ldots, d_n)$ is the base-d representation of the natural number $\sum_{i=1}^{n} a_i \cdot d^{i-1}$. Thus, the map ψ is bijective. This separates all the given monomial pairs. Let the degrees of the monomials $\psi(m_i)$ and $\psi(m_i')$ be d_i and d_i' respectively. But, the degree is now $d^n - 1$, which is exponential. We bring this degree down by taking the degrees modulo a prime p; $\psi_p : x_i \mapsto (y^{d^{i-1}} \mod (y^p - 1))$. This no longer guarantees that all the monomial pairs are separated. There may exist an i such that $d_i \equiv d_i' \mod p$, i.e. $p|(d_i - d_i')$. So, we try many primes. We want a small set of primes $\mathcal{P}$, such that $\exists p \in \mathcal{P}, \forall 1 \leq i \leq k : p \nmid (d_i - d_i')$. Thus, we want $\prod_{p \in \mathcal{P}} \nmid \prod (d_i - d_i')$. I.e. $\prod_{p \in \mathcal{P}} \nmid \prod |d_i - d_i'|$. We achieve that by ensuring $\prod_{p \in \mathcal{P}} > \prod_{i \leq k} |d_i - d_i'|$. Let us find a satisfactory set of primes, $\mathcal{P}$. Let $|\mathcal{P}| = N$. Since every prime is ≥ 2, $\prod_{p \in \mathcal{P}} > 2^N$. Now, for every i, $|d_i - d_i'| < d^n$. Thus, we want $2^N \geq d^{nk}$. I.e. $N = nk \log d$ will suffice. □

Corollary 2.3.3. *We can efficiently compute a monomial map where every monomial over c variables and individual variable degree $< d$, maps to a monomial in y with degree d^c and assigns distinct degrees to every monomial. Note that if c is a constant, then every monomial is mapped to a monomial in y with polynomial degree.*

Thus, a lower bound for a model with the restriction of multi-linearity also gives a lower bound for a model computing an exponential degree polynomial.

2.3.3 Generator

A concept equivalent to a hitting set is that of a generator (see [SY10, Section 4.1]). Instead of a set of points (hitting set) where we check the value of the polynomial $A(x_1, x_2, \ldots, x_n)$, we replace the variables $\{x_1, x_2, \ldots, x_n\}$ with polynomials. These polynomials are over a constant number of variables. So, the substituted polynomial A is 'simpler'.

Definition 2.3.4 (Generator)**.** *Let $\mathcal{P}$ be a class of n-variate polynomials. We use c new variables $\{t_1, t_2, \ldots, t_c\}$. We call a map $\phi : \{x_1, x_2, \ldots, x_n\} \longrightarrow \mathbb{F}[t_1, t_2, \ldots, t_c]$ with $\phi : x_i \longmapsto f_i(t_1, t_2, \ldots, t_c)$ a* generator *for the class $\mathcal{P}$ if for every nonzero polynomial $A(x_1, x_2, \ldots, x_n) \in \mathcal{P}$, $A(f_1, f_2, \ldots, f_n) \neq 0$.*

The degree of this generator ϕ is $\max\{\deg(f_i)\}_{i=1}^{n}$.

Finding a generator is equivalent to finding a hitting set. That is because, by Lemma 2.2.1, given a c-variate, degree d generator for a family of polynomials of degree δ, we can find a hitting set of size $(d\delta + 1)^c$. And, for the opposite direction, given a hitting set, by Corollary 2.3.6 below, we can find a generator.

2.3.4 Lagrange interpolation

Suppose we are given a set of distinct points $\beta_1, \beta_2, \ldots, \beta_k$ and a set of values $\{a_1, a_2, \ldots, a_k\}$ and we want to construct a univariate polynomial $f(x)$ such that f evaluates to a_i at the point β_i, i.e. $f|_{x=\beta_i} = a_i$ for all $i \leq k$. The following polynomial is the smallest degree polynomial for which this holds.

$$f(x) = \sum_{i=1}^{k} \frac{\prod_{j \neq i}(x - \beta_j)}{\prod_{j \neq i}(\beta_i - \beta_j)} a_i.$$

Each summand corresponds to an index i. When $x = \beta_i$, the ith summand evaluates to a_i and the other summands evaluate to 0. The degree of this polynomial is at most $k - 1$.

In this thesis, we use the Lagrange interpolation to find a small degree polynomial that takes a set of values $\{a_1, a_2, \ldots, a_k\}$ at *some* points (we don't care which points). We do that by choosing the set of distinct points $\{\beta_1, \beta_2, \ldots, \beta_k\}$ arbitrarily.

Lagrange interpolation can be applied to various structures, not just field elements.

As a warmup to the upcoming lemma, we take a set of *tuples*, $\{\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_k\}$, where $\boldsymbol{a}_i = [a_{i,1}, a_{i,2}, \ldots, a_{i,n}]$ with $a_{i,j} \in \mathbb{F}$. We fix $\{\beta_i\}_{i=1}^k$ to an arbitrary set of distinct constants. Lagrange interpolation, applied on the set of tuples $\{\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_k\}$, gives a tuple $\boldsymbol{f}(x) = [f_1(x), f_2(x), \ldots, f_n(x)]$ of univariate polynomials, where

$$f_r(x) = \sum_{i=1}^{k} \frac{\prod_{j \neq i}(x - \beta_j)}{\prod_{j \neq i}(\beta_i - \beta_j)} a_{i,r}, \qquad \text{for } 1 \leq r \leq n.$$

Observe that $\boldsymbol{f}(\beta_i) = \boldsymbol{a}_i$. The above equation can be expressed compactly as:

$$\boldsymbol{f}(x) = \sum_{i=1}^{k} \frac{\prod_{j \neq i}(x - \beta_j)}{\prod_{j \neq i}(\beta_i - \beta_j)} \boldsymbol{a}_i.$$

We next apply Lagrange interpolation on a set of tuples of polynomials.

Lemma 2.3.5. *Let $\mathcal{C}$ be a class of polynomials over the variables $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$. Let $\mathcal{G} = \{\mathbf{g}_1(t), \mathbf{g}_2(t), \ldots, \mathbf{g}_h(t)\}$ be a set of* polynomial tuples *where, for all i, $\mathbf{g}_i(t) \in \mathbb{F}[t]^n$. I.e. each $\boldsymbol{g}_i(t)$ is an n-sized tuple of univariate polynomials. Also suppose that for any polynomial $C(x_1, x_2, \ldots, x_n) \in \mathcal{C}$, there exists a polynomial tuple $\mathbf{g}_i$ in $\mathcal{G}$ that acts as a generator for C, i.e. $C(\mathbf{g}_i) \neq 0$. Let $d = \max\{\deg(\boldsymbol{g}_i)\}_{\boldsymbol{g}_i \in \mathcal{G}}$. Then, there exists a single polynomial tuple $\boldsymbol{f}(t, y) \in \mathbb{F}[t, y]^n$ such that for every nonzero polynomial $C(\boldsymbol{x}) \in \mathcal{C}$, $C(\boldsymbol{f}(y, t)) \neq 0$.*

Moreover, $\deg_t(\boldsymbol{f}) = d$ and $\deg_y(\boldsymbol{f}) = |\mathcal{G}| - 1$.

Proof. Let $\{\beta_i\}_{i=1}^h$ be an arbitrary set of distinct constants. The Lagrange interpolation $\boldsymbol{f}(y, t)$ of the polynomial tuples in $\mathcal{G}$ is

$$\boldsymbol{f}(y, t) = \sum_{i=1}^{h} \frac{\prod_{j \neq i}(y - \beta_j)}{\prod_{j \neq i}(\beta_i - \beta_j)} \mathbf{g}_i(t).$$

The key property of the interpolation is that when we put $y = \beta_i$, $\boldsymbol{f}(\beta_i, t) = \mathbf{g}_i(t)$ for all $i \in [h]$.

Pick any $C(\boldsymbol{x}) \in \mathcal{C}$. We know that there exists $i \in [h]$ such that $C(\mathbf{g}_i(t)) \neq 0$. I.e. $C(\boldsymbol{f}(\beta_i, t)) \neq 0$. So, $C(\boldsymbol{f}(y, t))$, whose evaluation is $C(\boldsymbol{f}(\beta_i, t))$, is nonzero.

We can see that $\deg_t(\boldsymbol{f}) = \max\{\deg(\boldsymbol{g}_i)\}_{g_i \in \mathcal{G}} = d$ and $\deg_y(\boldsymbol{f}) = h - 1$.

□

Thus, the tuple of polynomial $\boldsymbol{f}(y,t)$ from the above lemma is a generator for the class of polynomials C.

A corollary of Lemma 2.3.5 is given below.

Corollary 2.3.6. *Let $\mathcal{C}'$ be the class of degree δ' polynomials that can be written as a product of polynomials from the class of polynomials $\mathcal{C}$. Suppose $\mathcal{H}$ is a hitting-set for a class of polynomials $\mathcal{C}$. Then, there is a hitting-set of size $\delta'|\mathcal{H}|$ for the class of polynomials $\mathcal{C}'$.*

Proof. Let $h = |\mathcal{H}|$. The idea is to use the Lagrange interpolation of the hitting set $\mathcal{H}$ as a generator for the class of product polynomials, $\mathcal{C}'$.

By Lemma 2.3.5, we can find a polynomial tuple $\boldsymbol{f}$ over the variable y (the $\boldsymbol{g}_i$s are points in the hitting set, and hence, constants. So, there is no variable t), such that for any $C \in \mathcal{C}$, $C(\boldsymbol{f}(y)) \neq 0$. Also, $\deg(\boldsymbol{f}) = \deg_y(\boldsymbol{f}) = h - 1$.

Let $C'(\mathbf{x}) = C_1(\mathbf{x})C_2(\mathbf{x})\cdots C_m(\mathbf{x})$, be a degree δ' nonzero polynomial over $\mathbb{F}$, where $C_i \in \mathcal{C}$ for all $i \in [m]$. For any $a \in [m]$, the polynomial C_a, on substituting the polynomial tuple $\boldsymbol{f}$, is nonzero, i.e. $C_a(\boldsymbol{f}(y)) \neq 0$. So, $C'(\boldsymbol{f}(y))$ is a nonzero univariate polynomial.

Degree of $\boldsymbol{f}(y)$ is $h-1$. So, degree of $C'(\boldsymbol{f}(y))$ is bounded by $\delta'(h-1)$. We can substitute $\delta' h$ distinct values of y, so that on at least one of the substitutions, the polynomial $C'(\boldsymbol{f}(y))$ remains nonzero.

Thus, there exists a hitting-set of size $\delta' h$ for the class of polynomials $\mathcal{C}'$. □

2.4 Arithmetic branching programs (ABPs)

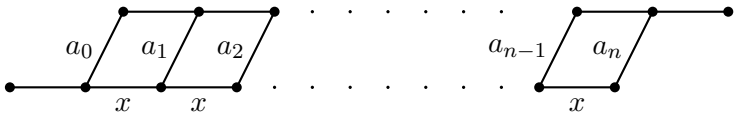
An *arithmetic branching program* (ABP) is a directed acyclic graph with two special vertices: the *start node*, s and the *end node*, t. All the edges in the graph have weights from the polynomial ring $\mathbb{F}[\boldsymbol{x}]$, for some field $\mathbb{F}$. For an edge e, let us denote its weight 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 s to t,

$$A(\boldsymbol{x}) = \sum_{p \text{ path } s \rightsquigarrow t} W(p).$$

These edge weights are the building blocks of the ABP. We study the ABP by putting various restrictions on its edge weights. For example, the following ABP, which computes the generic univariate polynomial $\sum_{i=0}^{d} a_i x^i$ has edge weights from the set $\mathbb{F} \cup \{x\}$ [2] .



In chapter 3 on the sum of ROABPs, we assume that the weights in each layer are univariate polynomials over distinct variables. In chapter 4 on invertible ROABPs, we assume that the edge weights are sparse and the variables in a layer are not used in any other layer. Whereas in chapter 5 on width-2 ABPs, the edge weights are linear polynomials.

We will assume that the ABP is *layered*, i.e. the vertices are partitioned into layers and the edges only join successive layers. I.e. an edge from the ith layer can only go to the $(i+1)$th layer. The *length* of an ABP is the length of a longest path from s to t. The *width* of an ABP is the maximum number of vertices in a layer.

Observe that the inner product of two w-dimensional vectors $\boldsymbol{a}, \boldsymbol{b}$ can *equivalently* be represented by a length 2 ABP of width w. Extending this idea, we can represent the arithmetic branching program as a matrix product.

By adding dummy vertices, we can assume that every layer has exactly w vertices, where w is the width of the ABP. Let the set of nodes in the ith layer be $\{v_{i,j} \mid j \in [w]\}$ for $1 \leq i \leq d-1$. Let the length of the ABP be d. Then, the arithmetic branching program can alternately be represented by a matrix product $\prod_{i=1}^{d} D_i$, where $D_1 \in \mathbb{F}[\boldsymbol{x}]^{1 \times w}$,

[2] **Drawings of ABPs in this thesis:** the edges are directed from left to right. We do not draw the edges with 0 weight. When an edge is present, but without any weight on it, it is assumed to have weight 1. The leftmost vertex is the start node and the rightmost vertex is the end node.

$D_i \in \mathbb{F}[\boldsymbol{x}]^{w\times w}$ for $2 \le i \le d-1$, and $D_d \in \mathbb{F}[\boldsymbol{x}]^{w\times 1}$ such that

$$\begin{aligned} D_1(j) &= W(s, v_{1,j}), \text{ for } 1 \le j \le w, \\ D_i(j,k) &= W(v_{i-1,j}, v_{i,k}), \text{ for } 1 \le j,k \le w \text{ and } 2 \le i \le n-1, \\ D_d(k) &= W(v_{d-1,k}, t), \text{ for } 1 \le k \le w. \end{aligned}$$

From this formulation, it is easy to see that ABPs are equivalent to iterated matrix multiplication.

2.5 Read-once oblivious ABPs (ROABPs)

Recall that an ABP can be written as a product of matrices $D(\boldsymbol{x}) = D_1(\boldsymbol{x})D_2(\boldsymbol{x})\cdots D_d(\boldsymbol{x})$, where each layer $D_i(\boldsymbol{x})$ is a matrix of polynomials (for $2 \le i < d$) and $D_1(\boldsymbol{x})$ and $D_d(\boldsymbol{x})$ are vectors of polynomials.

Equivalently, we can also write the ABP as $D_0D_1(\boldsymbol{x}_1)D_2(\boldsymbol{x}_2)\cdots D_d(\boldsymbol{x}_d)D_{d+1}$, where $D_0 \in \mathbb{F}^{1\times w}$ and $D_{d+1} \in \mathbb{F}^{w\times 1}$.

Definition 2.5.1. *An ABP is a read-once oblivious ABP (ROABP) if every variable x_i is present in only one of the layers of the ABP.*

Sequence of variable partitions- This naturally associates a partition $\boldsymbol{x} = \boldsymbol{x}_1 \sqcup \boldsymbol{x}_2 \sqcup \cdots \sqcup \boldsymbol{x}_d$ of the variables to the ROABP according to the variables present in the various layers of the ROABP. Moreover, we can also associate a sequence $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_d)$ on these variable partitions, according to the sequence of the layers that are present in the ROABP. Let $|\boldsymbol{x}_i| = n_i$. Thus, $\sum_{i=1}^{d} n_i = n$.

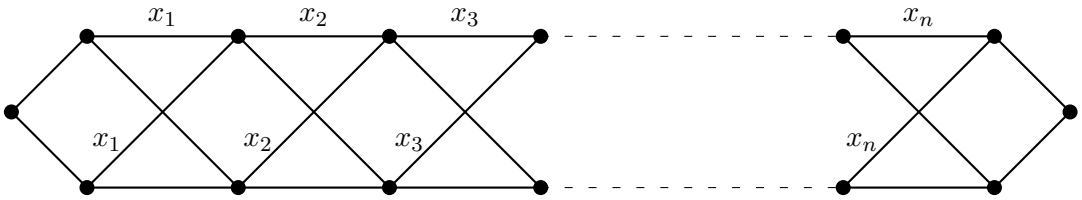
Polynomial over matrices- Observe that a matrix of polynomials is also a polynomial over matrices. I.e. any layer $D_i(\boldsymbol{x}_i) \in \mathbb{F}[\boldsymbol{x}_i]^{w\times w}$ can also be written as $D_i(\boldsymbol{x}_i) = \sum_{\boldsymbol{a}\in\mathbb{N}^{n_i}} D_{i,\boldsymbol{a}}\boldsymbol{x}_i^{\boldsymbol{a}}$, where $\boldsymbol{x}_i^{\boldsymbol{a}}$ are monomials over the variables in $\boldsymbol{x}_i$ and $D_{i,\boldsymbol{a}} \in \mathbb{F}^{w\times w}$ is a matrix of field constants. E.g. $\begin{bmatrix} x+y & x^2-x+1 \\ -y & 5y-3 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x^2 + \begin{bmatrix} 1 & -1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 1 & 0 \\ -1 & 5 \end{bmatrix} y +$

$$\begin{bmatrix} 0 & 1 \\ 0 & -3 \end{bmatrix} 1.$$

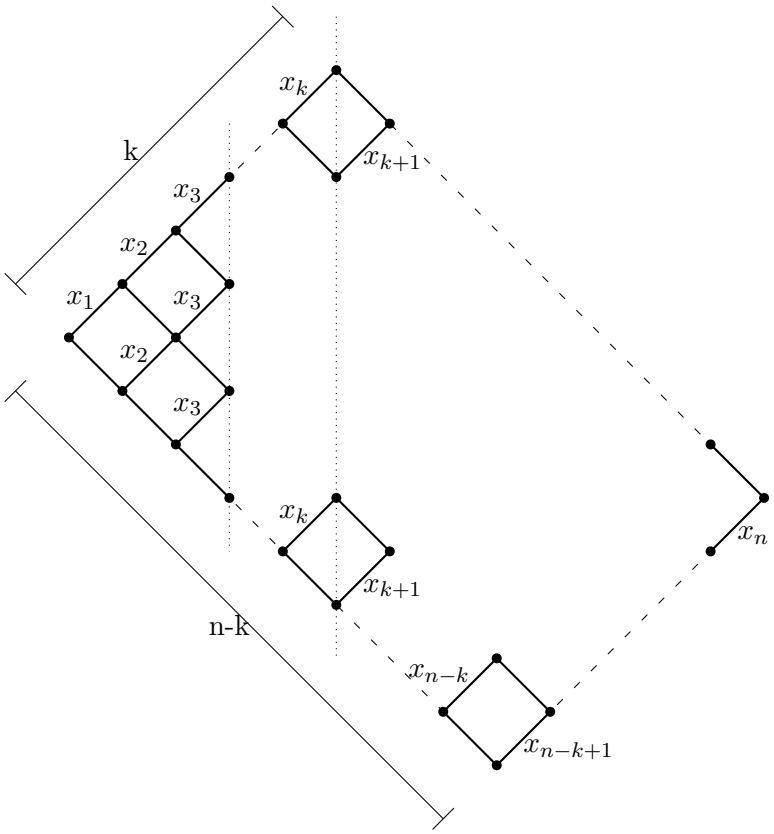
Coefficients of a monomial in an ROABP are easy to compute- An exponent $\boldsymbol{a} \in \mathbb{N}^n$ can be viewed as $(\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_d)$, where $\boldsymbol{a}_i$ is the subsequence of the tuple $\boldsymbol{a}$, corresponding to the variables in $\boldsymbol{x}_i$. Now, since the variables in the layers are not repeated,

$$\operatorname{coeff}_D(\boldsymbol{x}^{\boldsymbol{a}}) = \prod_{i=1}^{d} \operatorname{coeff}_{D_i}\left(\boldsymbol{x}_i{}^{\boldsymbol{a}_i}\right). \tag{2.1}$$

Some examples- The following is an example of an ROABP computing the polynomial $\prod_{i\in[n]}(1+x_i) = \sum_{S\subseteq[n]} \prod_{i\in S} x_i$.



And the following is an example of an ROABP computing the symmetric polynomial of degree k over n variables, $\sum_{S\subseteq[n],|S|=k} \prod_{i\in S} x_i$.



Each path from s to t takes k-many northward steps and $n-k$-many southward steps. On every northward step, we associate a unique variable.

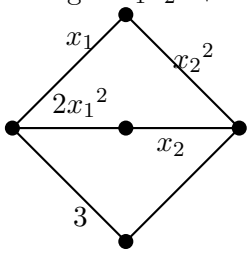
Though the ROABP can compute some interesting polynomials, the 'read-once' restriction severely limits the power of the arithmetic branching program. E.g. Kayal, Nair and Saha showed a polynomial that cannot be computed by any ROABP of sub-exponential size [KNS16]. Their lower bounds hold for any variable order. If a variable order is specified, then simpler lower bounds are known [Nis91].

Sparse layers: We already know that the variables are partitioned into $\boldsymbol{x}_1 \sqcup \boldsymbol{x}_2 \sqcup \cdots \boldsymbol{x}_d$, where the ith layer D_i is a matrix polynomial over the variables $\boldsymbol{x}_i$.

Lemma 2.5.2. *If the sparsity of each of the entries of each layer is s, then, there exists a width sw^2 ROABP with univariate layers that computes the same polynomial.*

Proof. There are $\leq w^2$ edges between any two layers. We replace each of these lines with $\leq s$ parallel lines, one for each monomial which has a nonzero coefficient in that edge

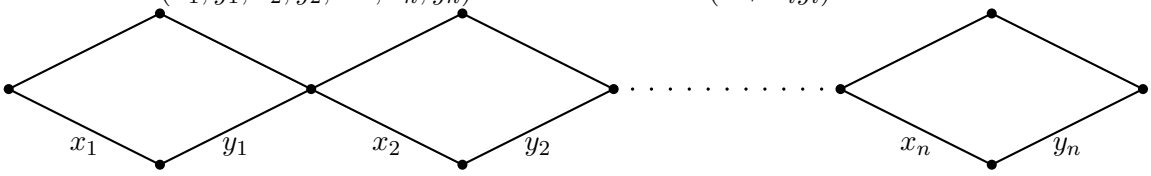
weight polynomial. Now, each of these lines is responsible for one monomial. We can order the variables in $\boldsymbol{x}_i$ arbitrarily and replace each line with n_i edges. E.g. An edge with weight $x_1{x_2}^2 + 2{x_1}^2x_2 + 3$ becomes



□

Hence, if s and w are both polynomially large, then we can assume without loss of generality that the ROABP has univariate layers.

Permutation associated with an ROABP: A *permutation* π of the variables $(x_i)_{i=1}^n$ can be associated with an ROABP. If the variables $(x_1, x_2, \ldots, x_n)$ occur in the ROABP in the sequence $\big(x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(n)}\big)$, then the permutation π is associated with it. It is an important property of the ROABP. An ROABP of a small width for a polynomial may exist in one permutation π. But, it may not exist in some other permutation. E.g. The polynomial $f(\boldsymbol{x}, \boldsymbol{y}) = \prod_{i=1}^n (1 + x_i y_i)$ has a width 2 ROABP when the permutation is $(x_1, y_1, x_2, y_2, \ldots, x_n, y_n)$. That is because $(1 + x_i y_i)$ has a width-2 ROABP.



But, it is known that when the permutation is $(y_1, y_2, \ldots, y_n, x_1, x_2, \ldots, x_n)$, any ROABP which computes $f(\boldsymbol{x}, \boldsymbol{y})$ has width 2^n. See Section 2.5.1 below for a proof sketch.

2.5.1 Evaluation Dimension

Definition 2.5.3. *Let $\boldsymbol{y} = (y_1, y_2, \ldots, y_m)$ be a subset of the variables $\boldsymbol{x}$ and let $\boldsymbol{a} = (a_1, a_2, \ldots, a_m) \in \mathbb{F}^m$. Given a polynomial $f(\boldsymbol{x})$, let $f|_{\boldsymbol{y}=\boldsymbol{a}}$ be the partial evaluation of the polynomial f obtained by substituting y_i with a_i for $i \in [m]$. Thus, $f|_{\boldsymbol{y}=\boldsymbol{a}} \in \mathbb{F}[\boldsymbol{x} \setminus \boldsymbol{y}]$. Then the* evaluation dimension, $\text{EvalDim}_{\boldsymbol{y}}(f)$ *of the polynomial $f(\boldsymbol{x})$ with respect to the set of variables $\boldsymbol{y}$ is defined as* $\text{span}\{f|_{\boldsymbol{y}=\boldsymbol{a}} \mid \boldsymbol{a} \in \mathbb{F}^m\}$.

Suppose an ROABP of width w and permutation of variables $\boldsymbol{x}$ computes a polynomial $f(\boldsymbol{x})$. Now, if $\boldsymbol{y}$ is a *prefix* of $\boldsymbol{x}$, then, the evaluation dimension of f with respect to $\boldsymbol{y}$, $\text{EvalDim}_{\boldsymbol{y}}(f) \leq w$. (A modification of the proof of Lemma 3.2.3 proves this statement.)

For example, consider the polynomial $f(\boldsymbol{x}, \boldsymbol{y}) = \prod_{i=1}^{n}(1 + x_i y_i) = \sum_{S \subseteq [n]} \prod_{i \in S} x_i y_i$. Consider its partial evaluation polynomial $f|_{\boldsymbol{y}=\boldsymbol{b}}$, where $\boldsymbol{b} \in \{0,1\}^n$. Let $\boldsymbol{b}$ be the indicator vector for the set $S_{\boldsymbol{b}}$. Then the monomial $x_{S_{\boldsymbol{b}}}$ occurs with coefficient 1 in the partial evaluation polynomial $f|_{\boldsymbol{y}=\boldsymbol{b}}$ and occurs with coefficient 0 in $f|_{\boldsymbol{y}=\boldsymbol{b}'}$ when $S_{\boldsymbol{b}} \nsubseteq S_{\boldsymbol{b}'}$. Thus the polynomials $\{f|_{\boldsymbol{y}=\boldsymbol{b}}\}_{\boldsymbol{b} \in \{0,1\}^n}$ are linearly independent. There are 2^n such partial evaluations. Hence, $w \geq \text{EvalDim}_{\boldsymbol{y}}(f) = 2^n$.

This method is used to show lower bounds on ROABPs. See [KNS16] for examples.

We introduce the *partial coefficient dimension* in Chapter 3. (See Equation 3.1.) The partial coefficient space and the partial evaluation space of any polynomial f with respect to any subset of variables $\boldsymbol{y}$ are equal. See Section 3.2.2 for a proof.

Coefficient space of an ROABP- Let the polynomial computed be $D_0 D(\boldsymbol{x}) D_{d+1} = \left\langle \left({D_0}^\top D_{d+1}^\top \right), D(\boldsymbol{x}) \right\rangle$, the inner product, where $D_0 \in \mathbb{F}^{1 \times w}$ and $D_{d+1} \in \mathbb{F}^{w \times 1}$. Thus, $D_0 D(\boldsymbol{x}) D_{d+1}$ computes the zero polynomial iff $\left({D_0}^\top D_{d+1}^\top \right)$ is 'orthogonal' to every coefficient of $D(\boldsymbol{x})$. I.e. $\left({D_0}^\top D_{d+1}^\top \right)$ is orthogonal to the space spanned by the coefficients of $D(\boldsymbol{x})$, the *coefficient space of* $D(\boldsymbol{x})$. Hence, we should find the basis of the coefficient space of $D(\boldsymbol{x})$.

The dimension of the coefficients of $D(\boldsymbol{x})$ is small; $\dim(\text{coeffs}(D)) \leq w^2$.

2.5.2 Whitebox PIT [RS05]

Using the distributivity of algebras and the read-once property of ROABPs, we can give a polynomial time PIT for ROABPs. We use the property that the coefficient of a monomial is the product of the corresponding coefficients in various layers (See Equation 2.1). The black box PIT for set-multilinear circuits by [ASS13], the blackbox PIT for ROABPs by [FS12, FSS14, AGKS15] are all based on this idea.

There are two variants of this method: the primal method and the dual method.

Primal method

We will assume that the ROABP is $D_0D_1(\boldsymbol{x}_1)D_2(\boldsymbol{x}_2)\cdots D_d(\boldsymbol{x}_d)D_{d+1}$, where $D_0 \in \mathbb{F}^{1\times w}$ and $D_{d+1} \in \mathbb{F}^{w\times 1}$ and the sparsity of each D_i is s for $1 \leq i \leq d$. We wish to find the coefficient space of $D_1(\boldsymbol{x}_1)D_2(\boldsymbol{x}_2)\cdots D_d(\boldsymbol{x}_d)$.

Starting with $j = 1$, we go iteratively over all the layers j for $1 \leq j \leq d$ and compute the basis of the vector space of the coefficients in $D_1D_2\ldots D_j$. Let the basis of the coefficients of $D_1D_2\ldots D_{j-1}$ be $\mathcal{B}_{j-1}$. Then, because of Equation 2.1, $\text{span}\{\text{coeffs}(D_1D_2\cdots D_j)\} = \text{span}\{\text{coeffs}(D_1D_2\cdots D_{j-1}) \times \text{coeffs}(D_j)\} = \text{span}\{\mathcal{B}_{j-1} \times \text{coeffs}(D_j)\}$.

Since the coefficients are w^2 dimensional, $|\mathcal{B}_{j-1}| \leq w^2$. Hence, $|\mathcal{B}_{j-1} \times \text{coeffs}(D_j)| \leq w^2s$ and we can find the basis $\mathcal{B}_j$ of the coefficients of $D_1D_2\ldots D_j$ in time polynomial in w, s. We do this for n rounds and get the coefficient space of $D_1(\boldsymbol{x})D_2(\boldsymbol{x})\cdots D_n(\boldsymbol{x})$.

Dual method

In the dual method, in the jth round, we conserve the nullspace $\mathcal{N}_j$ of $D_1D_2\cdots D_j$. Thus, in effect, we conserve the coefficient space of $D_1D_2\cdots D_j$.

Recall that the w^2 entries in D_1 form a linear space with the monomials acting as the indices. Let us call this set of w^2 entries $\{p_1, p_2, \ldots, p_{w^2}\}$.

Let us recall the method of finding the basis of the nullspace of a vector space and apply it to the set $\{p_1, p_2, \ldots, p_{w^2}\}$: Initially, the basis $\mathcal{B}_1 = \emptyset$ [3]. For $1 \leq i \leq w^2$, if the polynomial p_i is in the space spanned by the basis $\mathcal{B}_1$ constructed till now, i.e. if $p_i = \sum_{p_r\in\mathcal{B}_1} \alpha_r p_r$, then, we represent it as that linear combination. This linear combination $(\bar{\alpha}, -1, 0, 0, \ldots, 0)$ represents a basis element of the nullspace and is added to $\mathcal{N}_1$. If p_i is not in this space, we add p_i to the basis $\mathcal{B}_1$.

Now, for the ROABP, we follow a similar process in each round. We start with D_1. The polynomials in the basis $\mathcal{B}_1$ are replaced with new variables $y_1, y_2, \ldots, y_{|\mathcal{B}_1|}$. The polynomials not in the basis are replaced with their linear dependence on the polynomials in the basis. Now, all the entries are linear polynomials in y_is. There are $\leq w^2$ such new

[3] This basis comprises of the entries of the matrices, which are in $\mathbb{F}[\boldsymbol{x}]$. Thus, it is s dimensional, where s is the sparsity of the layers.

variables. Moreover, the since the nullspace is the same, the coefficient span of the new matrix is the same as that of D_1.

In the jth round ($j \geq 2$), we assume that the preceding matrix product $D_1 D_2 \cdots D_{j-1}$ is replaced by a single matrix D'_{j-1} with linear entries in $y_1, y_2, \ldots, y_{|\mathcal{B}_{j-1}|}$. The coefficient space of D'_{j-1} is the same as that of $D_1 D_2 \cdots D_{j-1}$. In the next round, we consider $D_1 D_2 \cdots D_j$ whose coefficient span is span $\Big\{\text{coeffs}(D'_{j-1}) \times \text{coeffs}(D_j)\Big\}$. There are $\leq w^2 s$ monomials. Thus, the above procedure of replacing polynomial entries with polynomials linear in the new variables $y_1, y_2, \ldots, y_{|\mathcal{B}_j|}$ can be done efficiently.

2.5.3 Basis Isolating Weight Assignment

[AGKS15] gave a possible framework, 'Basis Isolating Weight Assignment' for identity testing of polynomials of the kind $\langle c, P \rangle$, where c is a vector of field constants and P is a vector of polynomials. If we could give a polynomial substitution ψ for the variables, such that the coefficient space of P is maintained, then the identity testing of $\langle c, P \rangle$ is equivalent to the identity testing of $\langle c, \psi(P) \rangle$.

Note that ROABP is such a model. If the ROABP is $D_0 D(\boldsymbol{x}) D_{d+1}$, where $D(\boldsymbol{x})$ is an ROABP computing a $w \times w$ matrix, $D_0 \in \mathbb{F}^{1 \times w}$ and $D_{d+1} \in \mathbb{F}^{w \times 1}$, then $D_0 D D_{d+1}$ can be written as follows: let $c = \text{flattened}(D_0^\top D_{d+1}^\top)$, where the 'flattened' operator takes a $w \times w$ matrix and writes it row-by-row, as a w^2-dimensional vector. Let $P = \text{flattened}(D(\boldsymbol{x}))$, a w^2-dimensional vector of polynomials. Then, $D_0 D D_{d+1} = \langle c, P \rangle$.

[AGKS15] gave a quasi-polynomial degree univariate substitution which preserves the non-zeroness of the polynomial computed by an ROABP. This substitution $x_i \longmapsto t^{w(i)}$ for all i, with $w : [n] \longrightarrow \mathbb{N}$ is called as a *basis isolating weight assignment* (Definition 3.5.1). When this map is naturally extended to monomials, it ensures that there is a unique basis of the set of coefficients of $D(\boldsymbol{x})$ which has the minimum weight. [AGKS15] could find this map efficiently for ROABPs.

Lemma 2.5.4 ([AGKS15])**.** *Let $N = (nwd)^{\log n}$. We can find in time $\mathsf{poly}(N)$ a family of weight assignments $\left\{\mathrm{w}_1, \mathrm{w}_2, \ldots, \mathrm{w}_{\mathsf{poly}(N)}\right\}$ such that at least one weight w is a basis isolating weight assignment for the given ROABP of width w, over n variables, where the*

individual degree of the variables is d.

2.5.4 Shifting and concentration

One way to find a hitting-set is by showing a *low-support concentration* in the polynomial. It was first defined by Agrawal, Saha, and Saxena in [ASS13]. Low-support concentration in the polynomial $D(\boldsymbol{x})$ means that the coefficients of the low-support monomials in $D(\boldsymbol{x})$ span the whole coefficient space of $D(\boldsymbol{x})$. So, to check whether the polynomial is 0 or not, we just have to check the coefficients of the low-support monomials.

Now, we define *ℓ-concentration* for a polynomial over an algebra.

Definition 2.5.5 (ℓ-concentration)**.** *The polynomial $D(\boldsymbol{x}) \in \mathbb{A}_k[\boldsymbol{x}]$ is ℓ-concentrated if* $\operatorname{rank}_{\mathbb{F}}\{\operatorname{coeff}_D(\boldsymbol{x}^{\boldsymbol{a}}) \mid \boldsymbol{a} \in \mathbb{N}^n,\ \operatorname{supp}(\boldsymbol{a}) < \ell\} = \operatorname{rank}_{\mathbb{F}}\{\operatorname{coeff}_D(\boldsymbol{x}^{\boldsymbol{a}}) \mid \boldsymbol{a} \in \mathbb{N}^n\}$.

For a polynomial $C(\boldsymbol{x}) \in \mathbb{F}[\boldsymbol{x}]$, its coefficients form a one-dimensional vector space over $\mathbb{F}$. Thus, it has ℓ-support concentration if and only if it has at least one nonzero coefficient of support $< \ell$.

Low-support concentration in polynomial $D(\boldsymbol{x})$ implies low-support concentration in the polynomial $C(\boldsymbol{x}) = \langle c, D(\boldsymbol{x})\rangle$ for any $c \in \mathbb{F}^k$. I.e. $C(\boldsymbol{x})$ will have a nonzero coefficient for at least one of the low-support monomials.

Lemma 2.5.6. *Let $D(\boldsymbol{x}) \in \mathbb{A}_k[\boldsymbol{x}]$ be ℓ-concentrated. Let $C(\boldsymbol{x}) = \langle c, D(\boldsymbol{x})\rangle = \sum_i c_i D_i(\boldsymbol{x})$ for some constant vector $c \in \mathbb{F}^k$. Then, $C(\boldsymbol{x}) \in \mathbb{F}[\boldsymbol{x}]$ is ℓ-concentrated.*

Proof. Suppose $C(\boldsymbol{x}) \neq 0$, but all the ($< \ell$) support monomials of $C(\boldsymbol{x})$ have zero coefficient. So, there exists a ($\geq \ell$) support monomial $\boldsymbol{x}^{\boldsymbol{b}}$ of $C(\boldsymbol{x})$ with a nonzero coefficient.

Since D is ℓ-concentrated, it is in the span of the coefficients of low-support monomials. Let $M = \{\boldsymbol{a} \in \mathbb{N}^n \mid \operatorname{supp} \boldsymbol{x}^{\boldsymbol{a}} < \ell\}$ be the set of low-support monomials. Then,

$$\operatorname{coeff}_D(\boldsymbol{x}^{\boldsymbol{b}}) = \sum_{\boldsymbol{a} \in M} \gamma_{\boldsymbol{a}} \operatorname{coeff}_D(\boldsymbol{x}^{\boldsymbol{a}}),$$

where $\gamma_{\boldsymbol{a}}$ is a field constant for all monomials $\boldsymbol{a}$. Taking inner product of the above equation with c,

$$\left\langle c, \operatorname{coeff}_D(\boldsymbol{x}^{\boldsymbol{b}})\right\rangle = \sum_{\boldsymbol{a} \in M} \gamma_{\boldsymbol{a}} \left\langle c, \operatorname{coeff}_D(\boldsymbol{x}^{\boldsymbol{a}})\right\rangle.$$

So,

$$\operatorname{coeff}_C(\boldsymbol{x}^{\boldsymbol{b}}) = \sum_{\boldsymbol{a}\in M} \gamma_{\boldsymbol{a}} \operatorname{coeff}_C(\boldsymbol{x}^{\boldsymbol{a}}).$$

But, $\operatorname{coeff}_C(\boldsymbol{x}^{\boldsymbol{a}}) = 0$ for all low-support monomials $\boldsymbol{x}^{\boldsymbol{a}}$. So, $\operatorname{coeff}_C(\boldsymbol{x}^{\boldsymbol{b}}) = 0$, which is a contradiction. □

In other words, when D is low-support concentrated, $C(\boldsymbol{x})$ will have a nonzero coefficient for at least one of the low-support monomials. Thus, we get a hitting set by testing these low-support coefficients. We use the following lemma from [ASS13].

Lemma 2.5.7 ([ASS13])**.** *Given 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 $C(\boldsymbol{x}) \in \mathbb{F}[\boldsymbol{x}]$ of individual degree d, where $\{\beta_i\}_i$ are distinct nonzero elements in $\mathbb{F}$.*

Proof. It is easy to see that $|H| = O(nd)^\ell$. ℓ-Concentration for $C(\boldsymbol{x})$ simply means that it has at least one ($< \ell$)-support monomial with nonzero coefficient. Our set H will essentially test all these ($< \ell$)-support coefficients. We go over all subsets S of $\boldsymbol{x}$ with size $\ell - 1$ and do the following: Substitute 0 for all the variables outside the set S. There will be at least one choice of S, for which the polynomial $C(\boldsymbol{x})$ remains nonzero after the substitution. Now, it is an $(\ell - 1)$-variate nonzero polynomial of individual degree d.

We take the hitting set obtained from Lemma 2.2.2: $\{0, \beta_1, \ldots, \beta_d\}^{\ell-1}$ for this polynomial. □

Hence, once we have low-support concentration, we solve blackbox PIT.

Note that not every polynomial has low-support concentration, for example $C(\boldsymbol{x}) = x_1 x_2 \cdots x_n$ is not n-concentrated. However, Agrawal, Saha, and Saxena [ASS13] showed that for depth-3 set-multilinear circuits , low-support concentration can be achieved through an appropriate *shift* of the variables.

Let $\mathbf{f} = (f_i(t))_{i=1}^n$ be a tuple of polynomials, where $f_i \in \mathbb{F}[t]$ for each i. By 'shifting by a tuple $\mathbf{f}$', we mean replacement of x_i with $x_i + f_i$ for all $i \in [n]$. Note that $C(\boldsymbol{x} + \mathbf{f}) \neq 0$ if and only if $C(\boldsymbol{x}) \neq 0$. Hence, finding a blackbox PIT for $C(\boldsymbol{x} + \mathbf{f})$ is equivalent to finding a blackbox PIT for $C(\boldsymbol{x})$.

In the example of the polynomial $C(\boldsymbol{x}) = x_1 x_2 \cdots x_n$, we shift every variable by 1. That is, we consider $C(\boldsymbol{x} + \mathbf{1}) = (x_1 + 1)(x_2 + 1) \cdots (x_n + 1)$. Observe that $C(\boldsymbol{x} + \mathbf{1})$ has 1-concentration. To get some understanding about shifts, we will show that a shift preserves the coefficient space of a polynomial.

Lemma 2.5.8. *Let $A(\boldsymbol{x})$ be an n-variate polynomial and $\boldsymbol{f} = (f_1, f_2, \ldots, f_n) \in \mathbb{F}[t]^n$. Then $A(\boldsymbol{x})$ and $A(\boldsymbol{x} + \boldsymbol{f})$ have the same coefficient space as vectors over the extension field $\mathbb{F}(t)$.*

Note that the coefficients of the polynomial $A(\boldsymbol{x} + \boldsymbol{f})$ are vectors over the extension field $\mathbb{F}(t)$. In the proof of Lemma 2.5.8, we will be working only in this field, $\mathbb{F}(t)$. Also note that the rank of the coefficient space of $A(\boldsymbol{x})$ over $\mathbb{F}(t)$ remains the same as its rank over $\mathbb{F}$.

Proof of Lemma 2.5.8. We show that the coefficients of $A(\boldsymbol{x} + \boldsymbol{f})$ are in the span of the coefficients of $A(\boldsymbol{x})$. Recall that $M = \{0, 1, \ldots, d\}^n$ denotes the set of the exponents of all the 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 $A'(\boldsymbol{x}) = A(\boldsymbol{x} + \boldsymbol{f}) = \sum_{\boldsymbol{a} \in M} c_{\boldsymbol{a}} (\boldsymbol{x} + \boldsymbol{f})^{\boldsymbol{a}}$. Consider the coefficient of the monomial $\boldsymbol{x}^{\boldsymbol{b}}$, for every $\boldsymbol{b} \in M$.

$$\operatorname{coeff}_{A'}(\boldsymbol{x}^{\boldsymbol{b}}) = \sum_{\boldsymbol{a} \in M} \binom{\boldsymbol{a}}{\boldsymbol{b}} \boldsymbol{f}^{(\boldsymbol{a}-\boldsymbol{b})} \cdot c_{\boldsymbol{a}}, \tag{2.2}$$

where $\binom{\boldsymbol{a}}{\boldsymbol{b}} = \prod_{i=1}^{n} \binom{a_i}{b_i}$ for any $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{N}^n$. Recall that if $a_i < b_i$ for any $i \in [n]$, then $\binom{a_i}{b_i} = 0$ and hence, $\binom{\boldsymbol{a}}{\boldsymbol{b}} = 0$.

In other words, the coefficient of $\boldsymbol{x}^{\boldsymbol{b}}$ in $A'(\boldsymbol{x} + \boldsymbol{f})$ is a linear combination (over the extension field $\mathbb{F}(t)$) of the coefficients $c_{\boldsymbol{a}}$.

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

We will next see how to unify multiple shifts into one.

Lagrange Interpolation of a set of maps for ℓ-concentration

Lemma 2.5.9 ([GKST15])**.** *Let $\mathcal{C} \subseteq \mathbb{A}_k(\boldsymbol{x})$ be a class of n-variate, individual degree d polynomials. Let $\mathcal{G} = \{\boldsymbol{g}_i(t)\}_{i=1}^N$ be a set of univariate maps, with $\boldsymbol{g}_i(t) \in \mathbb{F}[t]^n$ for all i, such that for every polynomial $A(\boldsymbol{x}) \in \mathcal{C} \subseteq \mathbb{A}_k(\boldsymbol{x})$, there exists $\boldsymbol{g}_i \in \mathcal{G}$ such that $A' = A(\boldsymbol{x} + \boldsymbol{g}_i)$ is ℓ-concentrated. Let $N = |\mathcal{G}|$ and $d_t = \max\{\deg(\boldsymbol{g}_i)\}_{i\in[N]}$. Then, there exists a univariate map $\boldsymbol{f}(t)$ that ℓ-concentrates every polynomial in $\mathcal{C}$.*

Moreover, $\deg(\boldsymbol{f}(t)) = O(k^2d^2n^2Nd_t)$.

Proof. First, we find a single *bivariate* map $\boldsymbol{g}'(y,t) \in \mathbb{F}[y,t]^n$ of bounded degree that ℓ-concentrates every polynomial $A(\boldsymbol{x}) \in \mathcal{C}$.

We claim that when $\boldsymbol{g}'$ is the Lagrange interpolation (Section 2.3.4) of $\{\boldsymbol{g}_i(t)\}_{i=1}^N$, this property holds. I.e., we choose an arbitrary set of distinct constants $\{\beta_i\}_{i=1}^N$ and set

$$\boldsymbol{g}'(y,t) = \sum_{i=1}^{N} \frac{\prod_{j\neq i}(y-\beta_j)}{\prod_{j\neq i}(\beta_i-\beta_j)}\mathbf{g}_i(t).$$

The key property of the interpolation is that when we put $y = \beta_i$, $\boldsymbol{g}'(\beta_i, t) = \mathbf{g}_i(t)$ for all $i \in [N]$.

We will now prove that all polynomials in the family of polynomials $\mathcal{C}$ become ℓ-concentrated when shifted by $\boldsymbol{g}'(y,t)$. Suppose there exists $A(\boldsymbol{x}) \in \mathcal{C}$ such that shifting by the interpolation $\boldsymbol{g}'(y,t)$ does not ℓ-concentrate A. We know that there exists a polynomial $\boldsymbol{g}_i(t) \in \mathcal{G}$ such that $A' = A(\boldsymbol{x} + \boldsymbol{g}_i(t))$ is ℓ-concentrated. I.e. $\mathrm{rk}_{\mathbb{F}}\{\mathrm{coeffs}(A)\} = \mathrm{rk}_{\mathbb{F}(t)}\{\mathrm{coeff}_{A'}(\boldsymbol{x}^{\boldsymbol{a}}) \mid \mathrm{supp}(\boldsymbol{x}^{\boldsymbol{a}}) < \ell\} = k_A$, where $k_A \leq k$ is the dimension of the coefficient space of A[4]. But $\mathrm{rk}_{\mathbb{F}(y,t)}\{\mathrm{coeff}_{A''}(\boldsymbol{x}^{\boldsymbol{a}}) \mid \mathrm{supp}(\boldsymbol{x}^{\boldsymbol{a}}) < \ell\} < k_A$, where $A'' = A(\boldsymbol{x}+\boldsymbol{g}'(y,t))$ is the polynomial $A(\boldsymbol{x})$ shifted by the interpolation polynomial $\boldsymbol{g}'$. Hence, there exists a set $S = \{\boldsymbol{x}^{\boldsymbol{a}_1}, \boldsymbol{x}^{\boldsymbol{a}_2}, \ldots, \boldsymbol{x}^{a_{k_A}}\}$ of $(<\ell)$-support monomials such that the coefficients of these monomials are linearly independent in A'. But the coefficients of these same monomials are dependent in A''. So, there exists a linear dependency $(\gamma_{a_1}, \gamma_{a_2}, \ldots, \gamma_{a_{k_A}}) \neq \mathbf{0}$ such that

$$\sum_{\boldsymbol{x}^{\boldsymbol{a}}\in S} \gamma_{\boldsymbol{a}}\,\mathrm{coeff}_{A''}(\boldsymbol{x}^{\boldsymbol{a}}) = 0. \tag{2.3}$$

[4]Observe that $\mathrm{rk}_{\mathbb{F}}\{\mathrm{coeffs}(A)\} = \mathrm{rk}_{\mathbb{F}(y,t)}\{\mathrm{coeffs}(A)\}$ and $\mathrm{rk}_{\mathbb{F}(t)}\{\mathrm{coeff}_{A'}(\boldsymbol{x}^{\boldsymbol{a}}) \mid \mathrm{supp}(\boldsymbol{x}^{\boldsymbol{a}}) < \ell\} = \mathrm{rk}_{\mathbb{F}(y,t)}\{\mathrm{coeff}_{A'}(\boldsymbol{x}^{\boldsymbol{a}}) \mid \mathrm{supp}(\boldsymbol{x}^{\boldsymbol{a}}) < \ell\}$.

But,

$$\sum_{\boldsymbol{x}^{\boldsymbol{a}} \in S} \gamma_{\boldsymbol{a}} \operatorname{coeff}_{A'}(\boldsymbol{x}^{\boldsymbol{a}}) \neq 0. \tag{2.4}$$

Note that γ_{a_i}s are elements of the extension field $\mathbb{F}(y,t)$, i.e. $\gamma_{a_i} \in \mathbb{F}(y,t)$ for all i. Without loss of generality, by multiplying throughout by the least common multiple of the denominators of γ_{a_i}s, we can assume γ_{a_i}s are polynomials over the variables y,t. I.e. $\gamma_{a_i} \in F[y,t]$ for all i.

By the property of Lagrange interpolation, for any monomial $\boldsymbol{x}^{\boldsymbol{a}}$, $\operatorname{coeff}_{A'}(\boldsymbol{x}^{\boldsymbol{a}}) \in \mathbb{F}[t]^k$ is obtained by setting $y = \beta_i$ in $\operatorname{coeff}_{A''}(\boldsymbol{x}^{\boldsymbol{a}}) \in \mathbb{F}[y,t]^k$. Thus, Equations 2.3 and 2.4 cannot hold simultaneously. This is a contradiction. So, when the polynomials in C are shifted by $\boldsymbol{g}'(y,t)$, they become ℓ-concentrated. Moreover, the degree of the map will be bounded as: $\deg_t(\boldsymbol{g}') = d_t$ and $\deg_y(\boldsymbol{g}') = N - 1$.

We will now see how to make the bivariate map univariate. For any polynomial $A(\boldsymbol{x}) \in \mathcal{C}$ with $\operatorname{rk}_{\mathbb{F}}(\operatorname{coeffs}(A)) = k_A \leq k$, there is a set $S = \{\boldsymbol{x}^{\boldsymbol{a}_1}, \boldsymbol{x}^{\boldsymbol{a}_2}, \ldots, \boldsymbol{x}^{a_{k_A}}\}$ of $(< \ell)$-support monomials such that the coefficients of these monomials are linearly independent in $A''(\boldsymbol{x}) = A(\boldsymbol{x} + \boldsymbol{g}'(y,t))$. Let $d_y = \deg_y(\boldsymbol{g}')$.

Consider the $k \times k_A$ matrix M obtained by taking the coefficients $\{\operatorname{coeff}_{A''}(\boldsymbol{x}^{\boldsymbol{a}_i})\}_{i=1}^{k_A}$ of the monomials in the set S in A''. The matrix M has a submatrix M' of rank k_A. The determinant $\det(M')$ of this submatrix is a polynomial in y,t. And $\deg_y(\det(M')) = k_A \delta d_y$, $\deg_t(\det(M')) = k_A \delta d_t$, where $\delta = dn$ is the degree of the $\boldsymbol{x}$ variables in the polynomial A. Using Corollary 2.3.3, we get a univariate map $\boldsymbol{f}$ of degree $O(k^2 d^2 n^2 d_t d_y)$ that keeps $\det(M')$ nonzero, hence, the ℓ-concentration of A'' after the univariate substitution still holds. Hence, there exists a map $\boldsymbol{f}$ that ℓ-concentrates every polynomial $A \in \mathcal{C}$ after we shift by it.

□

2.6 Depth-3 circuits

A depth-3 circuit is usually defined as a $\Sigma\Pi\Sigma$ circuit: the circuit gates are in three layers, the top layer has an output gate which is $+$, second layer has all $\times$ gates and the last layer

has all + gates. In other words, the polynomial computed by a $\Sigma\Pi\Sigma$ circuit is of the form $C(\boldsymbol{x}) = \sum_{i=1}^{k} a_i \prod_{j=1}^{n_i} \ell_{ij}$, where a_is are field constants, n_i is the number of input lines to the i-th product gate and ℓ_{ij} is a linear polynomial of the form $b_0 + \sum_{r=1}^{n} b_r x_r$. An efficient solution for depth-3 PIT is still not known. The depth-3 model has recently gained much importance, as it has become a stepping-stone to understanding general arithmetic circuits: it was shown by Gupta et al. [GKKS16], that depth-3 circuits are almost as powerful as general circuits. A polynomial time hitting-set for a depth-3 circuit implies a quasi-poly-time hitting-set for general circuits. Till now, for depth-3 circuits, efficient PIT is known when the top fan-in is assumed to be constant [DS07, KS07, KS09, KS11, SS11, SS12, SS13] and for certain other restrictions [Sax08, SSS13, ASSS12].

2.7 Multilinear depth-3 circuits

A polynomial is said to be *multilinear* if the degree of every variable in every term is at most 1. The circuit $C(\boldsymbol{x})$ is a multilinear circuit if the polynomial computed at every gate is multilinear.

There are exponential lower bounds for depth-3 *multilinear* circuits [RY09]. Since there is a connection between lower bounds and PIT [Agr05], we can hope that solving PIT for depth-3 multilinear circuits should also be feasible. This should also lead to new tools for general depth-3.

A polynomial time algorithm is known only for a sub-class of multilinear depth-3 circuits, called *depth-3 set-multilinear circuits*. This algorithm is due to Raz and Shpilka [RS05] and is whitebox (See Section 2.5.2). In a depth-3 multilinear circuit, since every product gate computes a multilinear polynomial, a variable occurs in at most one of the n_i linear polynomials input to it. Thus, each product gate naturally induces a *partition* of the variables, where each *color* (i.e. part) of the partition contains the variables present in a linear polynomial ℓ_{ij}. Further, if the partitions induced by all the k product gates are the same then the circuit is called a depth-3 set-multilinear circuit.

Any depth-3 multilinear circuit $C(\boldsymbol{x})$ with top fan-in k can also be written as $C(\boldsymbol{x}) =$

$\langle c, D(\boldsymbol{x})\rangle$, where $D(\boldsymbol{x})$ is a small $\Pi\Sigma$ circuit over $\mathbb{H}_k(\mathbb{F})$ and $c \in \mathbb{F}^{1\times k}$.

2.7.1 Set-multilinear depth-3 circuits

Agrawal et al. [ASS13] gave a quasi-polynomial time blackbox algorithm for the class of depth-3 set-multilinear circuits. Their approach is to view the vector of k products, $D(\boldsymbol{x}) = (\prod_{j=1}^{n_i} \ell_{ij})_{i=1}^{k}$ as a polynomial over the Hadamard algebra, $\mathbb{H}_k(\mathbb{F})$, and to achieve a *low-support concentration* in it. Recall that low-support concentration means that all the coefficient vectors in $D(\boldsymbol{x})$ are linearly dependent on low-support coefficient vectors.

2.7.2 Low-distance multilinear depth-3 circuits

In our attempt to give a hitting set for general multilinear circuits, we studied low-distance multilinear depth-3 circuits. We defined a notion of *distance* for multilinear depth-3 circuits (say, in n variables and k product gates) that measures how far are the partitions from a mere *refinement*. The 1-distance strictly subsumes the set-multilinear model, while n-distance captures general multilinear depth-3.

Each product gate in a depth-3 multilinear circuit induces a partition on the variables. Let these partitions be $\mathbb{P}_1, \mathbb{P}_2, \ldots, \mathbb{P}_k$. Recall that $\mathrm{Part}(S)$ denotes the set of all possible partitions of the set S. Elements in a partition are called *colors*.

Definition 2.7.1 (Distance for a partition sequence, $\mathrm{d}(\mathbb{P}_1, \ldots, \mathbb{P}_k)$)**.** *Let* $\mathbb{P}_1, \mathbb{P}_2, \ldots, \mathbb{P}_k \in \mathrm{Part}([n])$ *be the* k *partitions of the variables* $\{x_1, x_2, \ldots, x_n\}$*. Then* $\mathrm{d}(\mathbb{P}_1, \mathbb{P}_2, \ldots, \mathbb{P}_k) = \delta$ *if* $\forall i \in \{2, 3, \ldots, k\}$,
$\forall$*colors* $Y_1 \in \mathbb{P}_i$, $\exists Y_2, Y_3, \ldots, Y_{\delta'} \in \mathbb{P}_i$ $(\delta' \leq \delta)$ *such that* $Y_1 \cup Y_2 \cup \cdots \cup Y_{\delta'}$ *equals a union of some colors in* $\mathbb{P}_j, \forall j \in [i-1]$.

In other words, in every partition $\mathbb{P}_i$, each color Y_1 has a set of colors called 'friendly neighborhood', $\{Y_1, Y_2, \ldots, Y_{\delta'}\}$, consisting of at most δ colors, which is exactly partitioned in the 'upper partitions'. We call $\mathbb{P}_i$ an *upper* partition relative to $\mathbb{P}_j$ (and $\mathbb{P}_j$ a *lower* partition relative to $\mathbb{P}_i$), if $i < j$. For a color X_a of a partition $\mathbb{P}_j$, let $\mathrm{nbd}_j(X_a)$ denote its friendly neighborhood. The friendly neighborhood $\mathrm{nbd}_j(x_i)$ of a variable x_i in a partition

$\mathbb{P}_j$ is defined as $\text{nbd}_j(\text{color}_j(x_i))$, where $\text{color}_j(x_i)$ is the color in the partition $\mathbb{P}_j$ that contains the variable x_i. The friendly neighborhood $\text{nbd}_j(\{x_i\}_{i\in\mathcal{I}})$ of a set of variables $\{x_i\}_{i\in\mathcal{I}}$ in a partition $\mathbb{P}_j$ is given by $\bigcup_{i\in\mathcal{I}} \text{nbd}_j(x_i)$.

Definition 2.7.2 (δ-distance circuits)**.** *A multilinear depth-3 circuit C has δ-distance if its product gates can be ordered to correspond to a partition sequence $(\mathbb{P}_1, \ldots, \mathbb{P}_k)$ with $\text{d}(\mathbb{P}_1, \mathbb{P}_2, \ldots, \mathbb{P}_k) \leq \delta$.*

The corresponding $\Pi\Sigma$ circuit $D(\boldsymbol{x})$ over $\mathbb{H}_k(\mathbb{F})$ is also said to have δ-distance.

Every depth-3 multilinear circuit is thus an n-distance circuit. A circuit with a partition sequence, where the partition $\mathbb{P}_i$ is a refinement of the partition $\mathbb{P}_{i+1}, \forall i \in [k-1]$, exactly characterizes a 1-distance circuit. All depth-3 multilinear circuits have distance between 1 and n. Also observe that the circuits with 1-distance subsume set-multilinear circuits.

Friendly neighborhoods - To get a better picture, we ask: Given a color X_a of a partition $\mathbb{P}_j$ in a circuit $D(\boldsymbol{x})$, how do we find its friendly neighborhood $\text{nbd}_j(X_a)$? Consider a graph G_j which has the colors of the partitions $\{\mathbb{P}_1, \mathbb{P}_2, \ldots, \mathbb{P}_j\}$, as its vertices. For all $i \in [j-1]$, there is an edge between the colors $X \in \mathbb{P}_i$ and $Y \in \mathbb{P}_j$ if they share at least one variable. Observe that if any two colors X_a and X_b of partition $\mathbb{P}_j$ are reachable from each other in G_j, then, they should be in the same neighborhood. As reachability is an equivalence relation, *the neighborhoods are equivalence classes of colors.*

Moreover, observe that for any two variables x_a and x_b, if their respective colors in partition $\mathbb{P}_j$, $\text{color}_j(x_a)$ and $\text{color}_j(x_b)$ are reachable from each other in G_j then their respective colors in partition $\mathbb{P}_{j+1}$, $\text{color}_{j+1}(x_a)$ and $\text{color}_{j+1}(x_b)$ are also reachable from each other in G_{j+1}. Hence,

Observation 2.7.3. *If at some partition, the variables x_a and x_b are in the same neighborhood, then, they will be in the same neighborhood in all of the lower partitions. I.e.* $\text{nbd}_j(x_a) = \text{nbd}_j(x_b) \implies \text{nbd}_i(x_a) = \text{nbd}_i(x_b), \forall i \geq j$.

In other words, at the level of the variables, the neighborhoods in the upper partitions are *refinements* of the neighborhoods in the lower partitions.

We will now show that a depth-3 multilinear circuit with δ-distance reduces to a polynomial size ROABP. As a toy case, we will show that a depth-3 multilinear circuit with 1-distance reduces to a polynomial size ROABP.

Lemma 2.7.4. *A depth-3, top fan-in k multilinear circuit where the partition $\mathbb{P}_i$ is a refinement of the partition $\mathbb{P}_{i+1}, \forall i \in [k-1]$ reduces to a width-$2k$ ROABP.*

Proof. Firstly, observe that a linear polynomial can be computed as a width-2 commutative ROABP. Thus, for every product gate $\mathbb{P}_i$, we get a product of width-2 ROABPs. We put these ROABPs in parallel to get a width-$2k$ ABP.

We will use the fact that the circuit has distance 1 to convert this into an ROABP in k iterations. Recall that if a polynomial $A(\boldsymbol{x}) \in \mathbb{F}(\boldsymbol{x})$ is a product of polynomials $A = A_1 A_2 \cdots A_d$, where each factor $A_i(\boldsymbol{x}) \in \mathbb{F}(\boldsymbol{x})$ is computed by an ROABP, then the factors can also commute. In the ith iteration, we consider a color $\mathcal{X}$ in the i-th partition, $\mathbb{P}_i$. We shift all the ROABPs corresponding to the colors in $\mathrm{nbd}_j(\mathcal{X})$, so that they are in the same ROABP layer as that of color $\mathcal{X}$ for all the upper partitions $j < i$. We do this for all the colors $\mathcal{X}$ in the ith layer. □

Lemma 2.7.5. *A depth-3, δ-distance multilinear circuit with top fan-in k reduces to a width-$2k$ ROABP.*

Proof. We multiply out the linear polynomials corresponding to each of the neighborhoods in each of the partitions. Then, we get a $\Sigma^{[k]}\Pi\Sigma^{[n^\delta]}\Pi$ distance-1 circuit[5]. Since the distance of the original circuit is $\leq \delta$ and since there are $\leq n$ variables in each linear polynomial, each partition is now a product of polynomials with sparsity $\leq n^\delta$. Each of the factors in a partition has distinct variables.

$$C(\boldsymbol{x}) = \sum_{i=1}^{k} \prod_{j=1}^{n_i} Q_{ij},$$

where, Q_{ij}s are n^δ-sparse and for every i, $\{Q_{ij}\}_j$ are over distinct variables. Moreover, for every partition $\mathbb{P}'_i$ in this new circuit, for every $Q_{ij}, \exists \left\{Q_{i-1,j_1}, Q_{i-1,j_2}, \ldots, Q_{i-1,j_{m_j}}\right\}$,

[5] $\Sigma^{[m]}$ means that all the summation gates at that particular level have fan-in m.

a set of factors in P_{i-1} such that the union of their variables equals the variables in Q_{ij}. The partition $\mathbb{P}'_i$ is a refinement of the partition $\mathbb{P}'_{i+1}$.

We build a commutative ROABP of width n^δ for each of the factors in each partition. Once we have these building blocks, we use the same trick as Lemma 2.7.4 to build one single ROABP of width kn^δ. □

Thus, from Lemma 2.5.4, we have a $(kn^\delta)^{O(\log n)}$ time hitting set for a δ-distance depth-3 multilinear circuit.

Chapter 3

Deterministic PIT for Sum of ROABPs

Abstract

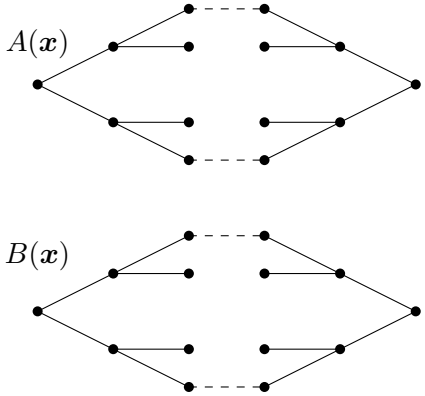
An arithmetic branching program (ABP) where each variable occurs in at most one layer is known as a *read-once oblivious arithmetic branching program (ROABP)*. In this chapter, we give the first polynomial time whitebox identity test for a polynomial computed by a sum of constantly many ROABPs. We also 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.

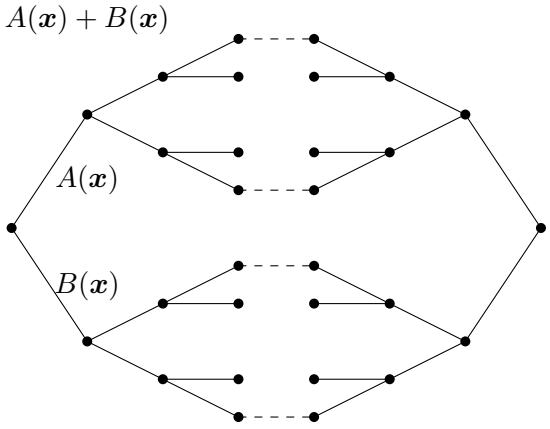
Our techniques are a new interplay of three concepts for ROABP: low evaluation dimension, basis isolating weight assignment and low-support rank concentration. We relate basis isolation to rank concentration and extend it to a sum of two ROABPs using evaluation dimension.

3.1 Introduction

We consider the *sum of* ROABPs. In the following figure, we have two ROABPs, A and B. The variable sequence of these ROABPs need not be the same.



The following ABP is obtained by introducing a new source node and connecting it with the old source nodes using edges of weight 1. A similar operation is done for the sink nodes. The ABP in the figure below computes $A + B$, the sum of two ROABPs.



Kayal, Nair and Saha [KNS16, 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.

Whitebox PIT: In Section 3.3 we show our first main result, a whitebox PIT for the sum of ROABPs (Theorem 3.3.2):

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.

Techniques

In the *whitebox* case, we try to build an ROABP for B in the same variable sequence as that of A. If $A = B$, obviously, such an ROABP exists. We build the ROABP for B in iterations, layer-by-layer. It is known that the partial coefficient dimension of an ROABP is equal to its width [Nis91]. The linear dependencies amongst the partial coefficients at any layer describe the ROABP exactly. So, if we knew these partial dependencies for B, we could build the ROABP for B. But, given a polynomial, there is no technique known which can find these linear dependencies amongst the partial coefficients. However, in this case, help is available. A small set of candidate linear dependencies which describe the ROABP B completely are provided by the ROABP A. If $A = B$, all the dependencies of A must hold for B. If $A \neq B$, there exist dependencies of A which do not hold for B, unless $A = \beta B$, for some constant β (which can be checked easily). Checking if a candidate linear dependency holds for B is equivalent to PIT for a single ROABP of $O(w^2)$ width. Thus, we reduce PIT of sum of ROABPs to many PITs of single ROABPs (one for each linear dependency). We use [RS05] for the PIT of a single ROABP.

Blackbox PIT: In Section 3.4, 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 3.4.5):

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

Our exact time bound 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. [KNS16] 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.

Before this work, it was not known that the shifting by a basis isolating weight assignment would concentrate the ROABP. The shifts which were used for concentration before this work were the following: [ASS13] gave a shift which concentrated depth-3 set-multilinear circuits and [FSS14] gave a shift which concentrated ROABPs. These shifts concentrated a *part* of the circuit at a time. Then, they extended these concentrations to the whole circuit. In contrast, the shift presented in this chapter is the first one that directly works on the complete ROABP.

Techniques

Like the whitebox test, we check if the linear dependencies of the partial coefficients of A are satisfied by B. But, we cannot do that directly when we don't have whitebox access to the ROABPs. So, we shift the variables and achieve *low-support concentration*, which ensures that the coefficients of an ROABP are spanned by the coefficients of its low support monomials. This ensures that if a dependency of partial coefficients in A is not followed by B, then, a nonzero monomial of low-support survives.

This idea is then extended to a sum of constantly many ROABPs. We show that a shift which concentrates a width-w^{2^c} ROABP also concentrates a polynomial computed as the sum of c ROABPs, each of width-w (Lemma 3.4.4).

In the proof, we show that shifting the input polynomial by a basis isolating weight assignment achieves low-support concentration.

3.2 Preliminaries

3.2.1 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 work with *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 $A(\boldsymbol{x}) = x_1 + x_1x_2 + {x_1}^2$, 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

$$A(\boldsymbol{x}) = \sum_{\boldsymbol{a}\in\{0,1,\ldots,d\}^k} A_{(\boldsymbol{y},\boldsymbol{a})}\, \boldsymbol{y}^{\boldsymbol{a}}\,. \tag{3.1}$$

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 [FS13a, 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

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

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 $\mathcal{P}$.

3.2.2 Equivalence of evaluation dimension and partial coefficient dimension

Partial evaluation was introduced in Section 2.5.1. We will now show that the partial evaluation space and the partial coefficient space are equivalent.

Lemma 3.2.1. *The partial coefficient space and the partial evaluation space of any polynomial $A(\boldsymbol{x})$ with respect to any subset of variables $\boldsymbol{y}$ are equal.*

Proof. Equation (3.1) says

$$[\boldsymbol{y}^{\boldsymbol{a}_1}\ \boldsymbol{y}^{\boldsymbol{a}_2}\ \cdots\ \boldsymbol{y}^{\boldsymbol{a}_N}] \begin{bmatrix} A_{(\boldsymbol{y},\boldsymbol{a}_1)} \\ A_{(\boldsymbol{y},\boldsymbol{a}_2)} \\ \vdots \\ A_{(\boldsymbol{y},\boldsymbol{a}_N)} \end{bmatrix} = [A]. \tag{3.2}$$

Recall that $A|_{\boldsymbol{y}=\boldsymbol{b}}$ is the polynomial obtained by substituting $\boldsymbol{y}$ with $\boldsymbol{b}$ in the polynomial

$A(\boldsymbol{x})$. So,

$$[\boldsymbol{b}^{\boldsymbol{a}_1}\ \boldsymbol{b}^{\boldsymbol{a}_2}\ \cdots\ \boldsymbol{b}^{\boldsymbol{a}_N}] \begin{bmatrix} A_{(\boldsymbol{y},\boldsymbol{a}_1)} \\ A_{(\boldsymbol{y},\boldsymbol{a}_2)} \\ \vdots \\ A_{(\boldsymbol{y},\boldsymbol{a}_N)} \end{bmatrix} = [A|_{\boldsymbol{y}=\boldsymbol{b}}].$$

So, every partial evaluation is a linear combination of the partial coefficients. Now, by substituting the appropriate set of values for $\boldsymbol{y}$, we can ensure that the first matrix in the left hand side of the below equation is invertible.

$$\begin{bmatrix} \boldsymbol{b}_1^{\boldsymbol{a}_1} & \boldsymbol{b}_1^{\boldsymbol{a}_2} & \cdots & \boldsymbol{b}_1^{\boldsymbol{a}_N} \\ \boldsymbol{b}_2^{\boldsymbol{a}_1} & \boldsymbol{b}_2^{\boldsymbol{a}_2} & \cdots & \boldsymbol{b}_2^{\boldsymbol{a}_N} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{b}_N^{\boldsymbol{a}_1} & \boldsymbol{b}_N^{\boldsymbol{a}_2} & \cdots & \boldsymbol{b}_N^{\boldsymbol{a}_N} \end{bmatrix} \begin{bmatrix} A_{(\boldsymbol{y},\boldsymbol{a}_1)} \\ A_{(\boldsymbol{y},\boldsymbol{a}_2)} \\ \vdots \\ A_{(\boldsymbol{y},\boldsymbol{a}_N)} \end{bmatrix} = \begin{bmatrix} A|_{\boldsymbol{y}=\boldsymbol{b}_1} \\ A|_{\boldsymbol{y}=\boldsymbol{b}_2} \\ \vdots \\ A|_{\boldsymbol{y}=\boldsymbol{b}_N} \end{bmatrix}.$$

Thus, the partial coefficient space is also a linear combination of the partial evaluation space. □

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 $a \otimes b$. For any $a, R \in \mathbb{F}^{w\times w}$, let $\langle a, R\rangle = \sum_{i=1}^{w}\sum_{j=1}^{w} a_{ij}R_{ij}$ be the inner product of a and R, when both are viewed as vectors.

3.2.3 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 from any layer V_i are only allowed to go to its successive layer V_{i+1}. 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 $|V_i| \le w$ for all $0 \le i \le \ell$.

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

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

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

3.2.4 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. Without loss of generality, we can assume that exactly one variable occurs per layer (see Lemma 2.5.2). Hence, without loss of generality, 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 3.2.2. *For a polynomial $A(\boldsymbol{x}) = \prod_{i=1}^{n} D_i(x_{\pi(i)})$ computed by an* ROABP*, we have*

$$\text{coeff}_A(\boldsymbol{x}^{\boldsymbol{a}}) = \prod_{i=1}^{n} \text{coeff}_{D_i}\left(x_{\pi(i)}^{a_{\pi(i)}}\right) \quad \in \mathbb{F}\,. \tag{3.3}$$

We also consider matrix polynomials computed by an ROABP. A matrix polynomial $A(\boldsymbol{x}) \in \mathbb{F}^{w\times w}[\boldsymbol{x}]$ is said to be computed by an ROABP if $A = D_1 D_2 \cdots D_n$, where $D_i \in \mathbb{F}^{w\times w}[x_{\pi(i)}]$ for $i \in [n]$ and some permutation π on $[n]$. Similarly, a vector polynomial $A(\boldsymbol{x}) \in \mathbb{F}^{1\times w}[\boldsymbol{x}]$ is said to be computed by an ROABP if $A = D_1 D_2 \cdots D_n$, where $D_1 \in \mathbb{F}^{1\times w}[x_{\pi(1)}]$ and $D_i \in \mathbb{F}^{w\times w}[x_{\pi(i)}]$ 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 (3.1) 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 3.2.3 ([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 $k \leq n$ and $\boldsymbol{y} = (x_1, x_2, \ldots, x_k)$ be the prefix of length k of $\boldsymbol{x}$. Then $\dim_{\mathbb{F}}\{A_{(\boldsymbol{y},\boldsymbol{a})} \mid \boldsymbol{a} \in \{0, 1, \ldots, d\}^k\} \leq w$.*

Proof. Let the polynomial $A(\boldsymbol{x}) = D_1(x_1)\, D_2(x_2) \cdots D_n(x_n)$, 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 $2 \leq i \leq n-1$. 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 $Q(\boldsymbol{z}) = D_{k+1} D_{k+2} \cdots D_n$.

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

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 $1 \leq i \leq w$, and we have $A(\boldsymbol{x}) = P(\boldsymbol{y})\, Q(\boldsymbol{z})$.

We get the following generalization of Equation (3.3): 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

$$A_{(\boldsymbol{y},\boldsymbol{a})} = \sum_{i=1}^{w} \operatorname{coeff}_{P_i}(\boldsymbol{y}^{\boldsymbol{a}})\, Q_i(\boldsymbol{z}). \tag{3.4}$$

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. □

The above lemma is used to prove lower bounds for ROABPs (e.g. [KNS16]). Basically, if we prove that the partial coefficient space of a polynomial A is large with respect to any subset of cardinality m of variables, then the width of an ROABP computing A should be large.

Observe that Equation (3.4) tells us that the polynomials $A_{(\boldsymbol{y},\boldsymbol{a})}$ can also be computed by an ROABP of width w: by Equation (3.3), 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 this observation in the following lemma.

Lemma 3.2.4 (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.*

For a general polynomial, the dimension considered in Lemma 3.2.3 can be exponentially large in n. We will next show the converse of Lemma 3.2.3: 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. [FS13a, 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 3.2.5. *Let $A(\boldsymbol{x})$ be a polynomial of individual degree d. For any $0 \leq k \leq n$ and $\boldsymbol{y}_k = (x_1, x_2, \ldots, x_k)$, suppose*

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

For $0 \leq k \leq n$, we recursively 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 3.2.6 ([Nis91], Converse of Lemma 3.2.3)**.** *Let $A(\boldsymbol{x})$ be a polynomial of individual degree d with $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$, such that for some w and 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 $A(\boldsymbol{x})$.

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

$$A(\boldsymbol{x}) = D_1 D_2 \cdots D_k\, [A_{(\boldsymbol{y}_k, \boldsymbol{a}_{k,1})}\ A_{(\boldsymbol{y}_k, \boldsymbol{a}_{k,2})}\ \cdots\ A_{(\boldsymbol{y}_k, \boldsymbol{a}_{k,w})}]^T\,. \tag{3.5}$$

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

$$A(\boldsymbol{x}) = \sum_{j=0}^{d} A_{(\boldsymbol{y}_1, j)}\, x_1^j. \tag{3.6}$$

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

$$A_{(\boldsymbol{y}_1, j)} = \sum_{i=1}^{w} \gamma_{j,i}\, A_{(\boldsymbol{y}_1, \boldsymbol{a}_{1,i})}. \tag{3.7}$$

From Equations (3.6) and (3.7) 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

$$A = D_1\, [A_{(\boldsymbol{y}_1, \boldsymbol{a}_{1,1})}\ A_{(\boldsymbol{y}_1, \boldsymbol{a}_{1,2})}\ \cdots\ A_{(\boldsymbol{y}_1, \boldsymbol{a}_{1,w})}]^T. \tag{3.8}$$

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

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

$$[A_{(\boldsymbol{y}_{k-1},\boldsymbol{a}_{k-1,1})} \cdots A_{(\boldsymbol{y}_{k-1},\boldsymbol{a}_{k-1,w})}]^T = D_k\,[A_{(\boldsymbol{y}_k,\boldsymbol{a}_{k,1})} \cdots A_{(\boldsymbol{y}_k,\boldsymbol{a}_{k,w})}]^T\,. \tag{3.9}$$

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

$$A_{(\boldsymbol{y}_{k-1},\boldsymbol{a}_{k-1,i})} = \sum_{j=0}^{d} A_{(\boldsymbol{y}_k,(\boldsymbol{a}_{k-1,i},j))}\, x_k^j. \tag{3.10}$$

Observe that $(\boldsymbol{a}_{k-1,i}, j)$ is just an extension of $\boldsymbol{a}_{k-1,i}$ and thus belongs to the set $\mathrm{depend}_k(A)$. Recall that $\mathrm{span}_k(A) = \{\boldsymbol{a}_{k,h}\}_{h=1}^{w}$. 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

$$A_{(\boldsymbol{y}_k,(\boldsymbol{a}_{k-1,i},j))} = \sum_{h=1}^{w} \gamma_{i,j,h}\, A_{(\boldsymbol{y}_k,\boldsymbol{a}_{k,h})}. \tag{3.11}$$

From Equations (3.10) and (3.11), 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 (3.9).

Finally, we obtain $D_n \in \mathbb{F}^{w\times 1}[x_n]$ in an analogous way. Instead of Equation (3.9) we consider the equation

$$[A_{(\boldsymbol{y}_{n-1},\boldsymbol{a}_{n-1,1})} \cdots A_{(\boldsymbol{y}_{n-1},\boldsymbol{a}_{n-1,w})}]^T = D'_n\,[A_{(\boldsymbol{y}_n,\boldsymbol{a}_{n,1})}]\,. \tag{3.12}$$

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 (3.8), (3.9), and (3.12), 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

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

where $\mathrm{coeff}_{P_k}(\boldsymbol{y}_k^{\boldsymbol{a}})$ is a vector in $\mathbb{F}^{1\times w}$. We will see next that it follows from the proof of

Lemma 3.2.6 that the coefficient space of P_k, i.e., $\text{span}_{\mathbb{F}}\{\text{coeff}_{P_k}(\boldsymbol{y}_k^{\boldsymbol{a}}) \mid \boldsymbol{a} \in \{0,1,\ldots,d\}^k\}$ has full rank w.

Corollary 3.2.7 (Full Rank Coefficient Space)**.** *Let $D_1, D_2, \ldots, D_n$ be the matrices constructed in the proof of Lemma 3.2.6 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 $\text{span}_k(A) = \{\boldsymbol{a}_{k,1}, \boldsymbol{a}_{k,2}, \ldots, \boldsymbol{a}_{k,w}\}$.*

Then for any $\ell \in [w]$, we have $\text{coeff}_{P_k}(\boldsymbol{y}_k^{\boldsymbol{a}_{k,\ell}}) = \boldsymbol{e}_\ell$, 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 3.2.6, consider the special case in Equations (3.7) and (3.11) that the exponent $(\boldsymbol{a}_{k-1,i}, j)$ is in $\text{span}_k(A)$, say $(\boldsymbol{a}_{k-1,i}, j) = \boldsymbol{a}_{k,\ell} \in \text{span}_k(A)$. Then the γ-vector to express $A_{(\boldsymbol{y}_k,(\boldsymbol{a}_{k-1,i},j))}$ in Equation (3.7) and (3.11) 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., $\text{coeff}_{D_k(i,\cdot)}(x_k^j) = \boldsymbol{e}_\ell$.

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

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

3.3 Whitebox Identity Testing

We will use the characterization of ROABPs provided by Lemmas 3.2.3 and 3.2.6 in Section 3.3.1 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 3.3.2, we extend the test to check whether the sum of constantly many ROABPs is zero.

3.3.1 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 3.3.1. *The equivalence of two ROABPs can be tested in time polynomial 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 3.2.6, 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 3.2.6 are the same for A and B. We give some more details with emphasis on the computability.

Construction of $\mathrm{span}_k(A)$ and the characterizing set of dependencies of 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 $\mathrm{coeff}_{D_1}(x_1^{b_1}) \in \mathbb{F}^{1\times w}$ and $\mathrm{coeff}_{D_i}(x_i^{b_i}) \in \mathbb{F}^{w\times w}$, 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 (3.4), we have

$$A_{(\boldsymbol{y}_k, \boldsymbol{b})} = C_{\boldsymbol{b}}\, D_{k+1} \cdots D_n\,. \tag{3.13}$$

Consider the set of vectors $\mathcal{D}_k = \{C_{\boldsymbol{b}} \mid \boldsymbol{b} \in \mathrm{depend}_k(A)\}$. 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 $\leq w$ such that $\mathcal{S}_k$ spans $\mathcal{D}_k$. Thus we can take $\mathrm{span}_k(A) = \{\boldsymbol{a} \mid C_{\boldsymbol{a}} \in \mathcal{S}_k\}$. Then, for any $\boldsymbol{b} \in \mathrm{depend}_k(A)$, vector $C_{\boldsymbol{b}}$ is a linear combination

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

Recall that $|\mathrm{depend}_k(A)| \leq w(d+1)$, i.e. this is a small set. Therefore, we can efficiently compute the coefficients $\gamma_{\boldsymbol{b},\boldsymbol{a}}$ for every $\boldsymbol{b} \in \mathrm{depend}_k(A)$. Note that by Equation (3.13) 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

$$A_{(\boldsymbol{y}_k,\boldsymbol{b})} = \sum_{\boldsymbol{a} \in \mathrm{span}_k(A)} \gamma_{\boldsymbol{b},\boldsymbol{a}}\, A_{(\boldsymbol{y}_k,\boldsymbol{a})}\,. \tag{3.14}$$

Verifying the dependencies for B. We want to verify that the dependencies in Equation (3.14) computed for A hold for B as well, i.e. that for all $k \in [n]$ and $\boldsymbol{b} \in \mathrm{depend}_k(A)$,

$$B_{(\boldsymbol{y}_k,\boldsymbol{b})} = \sum_{\boldsymbol{a} \in \mathrm{span}_k(A)} \gamma_{\boldsymbol{b},\boldsymbol{a}}\, B_{(\boldsymbol{y}_k,\boldsymbol{a})}\,. \tag{3.15}$$

Recall that $\boldsymbol{y}_k = (x_1, x_2, \ldots, x_k)$ and that the ROABP for B has a different variable order. By Lemma 3.2.4, 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 inherited from B. It follows that each of the $w+1$ polynomials that occur in Equation (3.15) has an ROABP of width w and the same variable order. Hence, we can construct *one* ROABP for the polynomial

$$B_{(\boldsymbol{y}_k,\boldsymbol{b})} - \sum_{\boldsymbol{a} \in \mathrm{span}_k(A)} \gamma_{\boldsymbol{b},\boldsymbol{a}} B_{(\boldsymbol{y}_k,\boldsymbol{a})}\,. \tag{3.16}$$

Simply identify all the start nodes and all the end nodes and use the appropriate constants $\gamma_{\boldsymbol{b},\boldsymbol{a}}$ as the weights. Then we get an ROABP of width $w(w+1)$. In order to verify Equation (3.15), it suffices to do a zero-test for this ROABP. This can be done in polynomial time [RS05].

As we will soon see (Section 3.3.2), this step blows up the complexity of the algorithm

as the number of ROABPs increases. Hence, a more efficient algorithm to verify the dependencies for B would reduce the time complexity for the sum of ROABPs drastically.

Constructing ROABP for B in the same sequence as A. Recall Lemma 3.2.6 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 (3.15), and they are the same as those for A in Equation (3.14).

The last matrix D_n can be written as $D'_n A_{(\boldsymbol{y}_n, \boldsymbol{a}_{n,1})}$, similar to Equation (3.12). 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 (3.15) fails to hold for some k and $\boldsymbol{b}$, then $A \neq B$. When Equation (3.15) 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 $A = D_1(x_1)D_2(x_2)\cdots D_n(x_n)$.

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 3.3.1.

3.3.2 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 reduces to checking whether $A = B$. Since A is computed by a single ROABP, we can use the same approach as in Section 3.3.1. Hence, we again get the dependencies from Equation (3.14) for A. Next, we have to verify these dependencies for B, i.e. Equation (3.15). Now, B is not given by a single ROABP, but is a sum of $c-1$ ROABPs. For every $k \in [n]$ and $\boldsymbol{b} \in \mathrm{depend}_k(A)$, 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

$$Q = \sum_{i=2}^{c} \left(A_{i(\boldsymbol{y}_k,\boldsymbol{b})} - \sum_{\boldsymbol{a}\in\mathrm{span}_k(A)} \gamma_{\boldsymbol{b},\boldsymbol{a}} A_{i(\boldsymbol{y}_k,\boldsymbol{a})} \right). \tag{3.17}$$

As explained in the previous section for Equation (3.16), for each summand in Equation (3.17) 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 3.3.1.

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 3.3.2. *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*

$w^{O(2^c)}(nd)^{O(c)}$.

3.4 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.

We achieve this by shifting the input polynomial by an appropriate polynomial. That is, we give a polynomial tuple $\boldsymbol{f}(t) = (f_1(t)\ f_2(t)\ \cdots\ f_n(t))$, such that given any nonzero polynomial $A(\boldsymbol{x})$ computed as the sum of c ROABPs, each of width w, the shifted polynomial $A(\boldsymbol{x}+\boldsymbol{f}) = A(x_1+f_1(t), x_2+f_2(t), \ldots, x_n+f_n(t))$ has a low-support monomial with a nonzero coefficient. Once, low support concentration is achieved, then a hitting set can be found efficiently (Lemma 2.5.7). See Section 2.5.4 for an introduction to ℓ-concentration and shifting.

For our purposes, any efficient shift that achieves low support concentration for ROABPs will suffice. In Section 3.5, we will give a new shift for ROABPs with quasi-polynomial cost. Namely, Theorem 3.5.6 states that

> *We can compute a shift polynomial $\boldsymbol{f}(t) \in \mathbb{F}[t]^n$ of degree $(ndw)^{O(\log n)}$ in time $(ndw)^{O(\log n)}$ such that for every $A(\boldsymbol{x})$ of individual degree d, computed by an ROABP of width w, the shifted polynomial $A(\boldsymbol{x} + \boldsymbol{f}(t))$ has $O(\log w)$-concentration.*

The shift also works for polynomials computed as vectors or matrices.

In this section, we show that the shift for a single ROABP can be applied to obtain a shift for the sum of constantly many ROABPs. The construction of a shift to obtain low support concentration for single ROABPs is postponed to Section 3.5.

3.4.1 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 (3.14) and (3.15) 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 (3.14) for A do not carry over to B as in Equation (3.15). Let $k \in [n]$ be the first such index. In the following Lemma 3.4.1 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 the coefficient space of R has full rank w'. Since the dependency Equations (3.14) for A do not fulfill Equation (3.15) for B, we get a constant vector $\Gamma \in \mathbb{F}^{1\times w'}$ such that $\Gamma P = 0$ but $\Gamma Q \neq 0$. Since R is full rank, Γ lies in the coefficient space of R. Hence, low support concentration of R ensures that Γ lies in the space of the small support coefficients of R. The simultaneous low support concentration of R, P and Q ensures that $R(P+Q)$ has low support concentration. We thus get low support concentration for $A + B$ (Lemma 3.4.2).

In the next lemma, we will decompose ROABPs A and B as $A = RP$ and $B = RQ$.

Lemma 3.4.1 (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* $\mathrm{supp}(\Gamma) \leq w+1$ *such that* $\Gamma P = 0$ *and* $\Gamma Q \neq 0$[3]*,*

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 3.2.6 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 3.2.6, 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 (3.14) do not fulfill Equation (3.15) 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 (3.5), we can write

$$A(\boldsymbol{x}) \;=\; D_1\, D_2 \cdots D_{k-1} \begin{bmatrix} A_{(\boldsymbol{y}_{k-1}, \boldsymbol{a}_{k-1,1})} \\ A_{(\boldsymbol{y}_{k-1}, \boldsymbol{a}_{k-1,2})} \\ \vdots \\ A_{(\boldsymbol{y}_{k-1}, \boldsymbol{a}_{k-1,w})} \end{bmatrix} \tag{3.18}$$

$$B(\boldsymbol{x}) \;=\; D_1\, D_2 \cdots D_{k-1} \begin{bmatrix} B_{(\boldsymbol{y}_{k-1}, \boldsymbol{a}_{k-1,1})} \\ B_{(\boldsymbol{y}_{k-1}, \boldsymbol{a}_{k-1,2})} \\ \vdots \\ B_{(\boldsymbol{y}_{k-1}, \boldsymbol{a}_{k-1,w})} \end{bmatrix} \tag{3.19}$$

Since the difference between A and B occurs at x_k, we consider all possible extensions from x_{k-1}. That is, by Equation (3.10), for every $i \in [w]$ we have

$$A_{(\boldsymbol{y}_{k-1}, \boldsymbol{a}_{k-1,i})} \;=\; \sum_{j=0}^{d} A_{(\boldsymbol{y}_k, (\boldsymbol{a}_{k-1,i}, j))} x_k^j \,. \tag{3.20}$$

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

[3] $\mathrm{supp}(\Gamma)$ is the number of nonzero entries in the vector Γ.

explicitly, this is

$$P = \begin{bmatrix} A_{(\boldsymbol{y}_k,(\boldsymbol{a}_{k-1,1},0))} \\ \vdots \\ A_{(\boldsymbol{y}_k,(\boldsymbol{a}_{k-1,1},d))} \\ \vdots \\ A_{(\boldsymbol{y}_k,(\boldsymbol{a}_{k-1,w},0))} \\ \vdots \\ A_{(\boldsymbol{y}_k,(\boldsymbol{a}_{k-1,w},d))} \end{bmatrix}.$$

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 (3.20), we get that

$$\begin{bmatrix} A_{(\boldsymbol{y}_{k-1},\boldsymbol{a}_{k-1,1})} \\ \vdots \\ A_{(\boldsymbol{y}_{k-1},\boldsymbol{a}_{k-1,w})} \end{bmatrix} = E_k\, P.$$

Thus, Equation (3.18) 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 $Q = \left(B_{(\boldsymbol{y}_k,(\boldsymbol{a}_{k-1,i},j))}\right)_{1\le i\le w,\ 0\le j\le d}$.

For the second claim of the lemma, let $\boldsymbol{b} \in \mathrm{depend}_k(A)$ such that the dependency Equation (3.14) for A is fulfilled, but not Equation (3.15) for B. There may be more that one such dependencies. We choose any one. Define $\Gamma \in \mathbb{F}^{1\times w'}$ to be the vector that has -1 at the position where P has entry $A_{(\boldsymbol{y}_k,\boldsymbol{b})}$, the values $\gamma_{\boldsymbol{b},\boldsymbol{a}}$ used in Equation (3.14) at the position where P has entry $A_{(\boldsymbol{y}_k,\boldsymbol{a})}$ for all $\boldsymbol{a} \in \mathrm{span}_k(A)$, and zero at all other positions. Then $\mathrm{supp}(\Gamma) \le w+1$ 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 3.2.7, 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 $\operatorname{coeff}_R(\boldsymbol{y}_k^{\boldsymbol{a}_{k-1,\ell},j}) = \boldsymbol{e}_{(\ell-1)(d+1)+j+1}$. As we vary ℓ and j, we cover $\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_{w(d+1)}\}$. Thus, the coefficient space of R has full rank w'. □

Lemma 3.4.1 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. Like we stated earlier, the shift that ℓ-concentrates every single ROABP of width w is given in Theorem 3.5.6. For now, we proceed with the assumption that it can efficiently computed.

Lemma 3.4.2. *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. Then, the assumption from Lemma 3.4.1 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 $\Gamma Q'$ is also $\ell_{w,2}$-concentrated: let $Q = [Q_1\, Q_2 \cdots Q_{w'}]^T \in \mathbb{F}[\boldsymbol{z}_k]^{w' \times 1}$. By Lemma 3.2.4, 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 $\mathrm{supp}(\Gamma) \leq w+1$. Since $w(w+1) \leq W_{w,2}$, 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 monomial $\boldsymbol{b} \in \{0,1,\ldots,d\}^{n-k}$ with $\mathrm{supp}(\boldsymbol{b}) < \ell_{w,2}$ 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

$$\Gamma \operatorname{coeff}_{P'+Q'}(\boldsymbol{z}_k^{\boldsymbol{b}}) \neq 0. \tag{3.21}$$

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 $M_{\ell_{w,2}} = \{\boldsymbol{a} \in \{0,1,\ldots,d\}^k \mid \mathrm{supp}(\boldsymbol{a}) < \ell_{w,2}\}$, we have $\mathrm{rank}_{\mathbb{F}(t)}\{\operatorname{coeff}_{R'}(\boldsymbol{y}_k^{\boldsymbol{a}}) \mid \boldsymbol{a} \in M_{\ell_{w,2}}\} = w'$. 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 all $\boldsymbol{a} \in M_{\ell_{w,2}}$. Hence, from Equation (3.21) 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 $\mathrm{supp}(\boldsymbol{a},\boldsymbol{b}) = \mathrm{supp}(\boldsymbol{a}) + \mathrm{supp}(\boldsymbol{b}) < 2\ell_{w,2}$, 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. □

Time Complexity. In Section 3.5, Theorem 3.5.6, we will show that the shift polynomial $\boldsymbol{f}_{w,2}(t) \in \mathbb{F}[t]^n$ used in Lemma 3.4.2 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 $(nd \deg(\boldsymbol{f}_{w,2}(t)) + 1)$-many values for 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 2.5.7, 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 3.5.6, we have the following main result.

Theorem 3.4.3. *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 now extend Lemma 3.4.2 to the sum of c ROABPs.

Lemma 3.4.4. *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 3.4.2 provides the base case $c = 2$. For the induction step let $c \geq 3$. We follow the proof of Lemma 3.4.2 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 the number of nonzero entries in Γ, $\operatorname{supp}(\Gamma) \leq w + 1$.

The only difference with the proof of Lemma 3.4.2 is $Q = [Q_1\, Q_2 \cdots Q_{w'}]^T$. Recall

from Lemma 3.4.1 that $Q_i = B_{(\boldsymbol{y}_k,\boldsymbol{a}_i)} = \sum_{j=2}^{c} A_{j\,(\boldsymbol{y}_k,\boldsymbol{a}_i)}$, for $\boldsymbol{a}_i \in \mathrm{depend}_k(A)$. 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 3.2.4, 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 the same as $(c-1)\,\ell_{w,c}$-concentrated.

Now we can proceed as in the proof of Lemma 3.4.2 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 3.4.3 and obtain our main result for the sum of constantly many ROABPs.

Theorem 3.4.5. *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.*

3.4.2 Concentration in matrix polynomials

As a by-product, we show that a special kind of low support concentration can be achieved in 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 2.5.5 for ℓ-support concentration can be applied to polynomials over any $\mathbb{F}$-algebra.

Consider a polynomial $A(\boldsymbol{x}) \in \mathbb{F}^k[\boldsymbol{x}]$, whose coefficients are k-dimensional vectors over the field $\mathbb{F}$. Consider a tuple of polynomials $\boldsymbol{f}(t) = (f_1(t)\ f_2(t)\ \cdots\ f_n(t)) \in \mathbb{F}[t]^n$. Then, the shifted polynomial $A'(\boldsymbol{x} + \boldsymbol{f}(t)) \in \mathbb{F}^k[t][\boldsymbol{x}]$ can be viewed as a polynomial in the polynomial ring $\mathbb{F}^k[\boldsymbol{x}, t]$. In this section, we will use the notation $A''(\boldsymbol{x}, t)$ to denote the shifted polynomial $A'(\boldsymbol{x} + \boldsymbol{f}(t)) \in \mathbb{F}^k[\boldsymbol{x}, t]$.

When the matrix polynomial A is computed by a sum of ROABPs, we will show $(\ell+1)$-concentration (for some ℓ) of the shifted polynomial $A''(\boldsymbol{x}, t) = A(\boldsymbol{x}+\boldsymbol{f}(t))$ when viewed as an $(n+1)$-variate polynomial, with coefficients from $\mathbb{F}^k$. In fact, the proof can be modified so that the coefficients of monomials in the set $\left\{\boldsymbol{x}^{\boldsymbol{a}}t^i \mid \text{supp}(\boldsymbol{a}) < \ell\right\}$ will span the space of all of its coefficients (Corollary 3.4.7). Note that this does not imply ℓ-concentration of $A'(\boldsymbol{x} + \boldsymbol{f}(t))$, a polynomial in $\mathbb{F}[t]^k[\boldsymbol{x}]$.

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

Lemma 3.4.6. *Let $A \in \mathbb{F}^{w\times w}[\boldsymbol{x}]$ be an n-variate polynomial and $\boldsymbol{f}(t)$ be a shift. Then $A(\boldsymbol{x} + \boldsymbol{f}(t)) \in \mathbb{F}^{w\times w}[\boldsymbol{x}, t]$ is ℓ-concentrated over the field $\mathbb{F}$ iff $\forall \alpha \in \mathbb{F}^{w\times w}$, $\langle \alpha, A\rangle (\boldsymbol{x} + \boldsymbol{f}(t)) \in \mathbb{F}[\boldsymbol{x}, t]$ is ℓ-concentrated over the field $\mathbb{F}$.*

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

For the other direction, let $A''(\boldsymbol{x}, t) = A(\boldsymbol{x} + \boldsymbol{f})$ be ℓ-concentrated over $\mathbb{F}$. So, any coefficient $\text{coeff}_{A''}(\boldsymbol{x}^{\boldsymbol{a}}t^i)$ can be written as a linear combination of the small support coefficients,

$$\text{coeff}_{A''}(\boldsymbol{x}^{\boldsymbol{a}}t^i) = \sum_{\substack{(\boldsymbol{b},j)\\ \text{supp}(\boldsymbol{b},j)<\ell}} \gamma_{\boldsymbol{b},j}\ \text{coeff}_{A''}(\boldsymbol{x}^{\boldsymbol{b}}t^j),$$

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

$$\langle \alpha, \text{coeff}_{A''}(\boldsymbol{x}^{\boldsymbol{a}})\rangle = \left\langle \alpha, \sum_{\substack{(\boldsymbol{b},j)\\ \text{supp}(\boldsymbol{b},j)<\ell}} \gamma_{\boldsymbol{b},j}\ \text{coeff}_{A''}(\boldsymbol{x}^{\boldsymbol{b}}t^j) \right\rangle = \sum_{\substack{(\boldsymbol{b},j)\\ \text{supp}(\boldsymbol{b},j)<\ell}} \gamma_{\boldsymbol{b},j} \left\langle \alpha, \text{coeff}_{A''}(\boldsymbol{x}^{\boldsymbol{b}}t^j)\right\rangle.$$

That is, $\langle \alpha, A'' \rangle (\boldsymbol{x}, t) = \langle \alpha, A \rangle (\boldsymbol{x} + \boldsymbol{f}(t))$ is ℓ-concentrated over $\mathbb{F}$. □

Now we can show low support concentration for a sum of matrix polynomials, each computed by an ROABP, analogous to Lemma 3.4.4.

Corollary 3.4.7 (Of Lemma 3.4.4)**.** *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 3.4.4.*

Then $A(\boldsymbol{x} + \boldsymbol{f}_{w^2,c}) \in \mathbb{F}^{w \times w}[\boldsymbol{x}, t]$ is $(c\ell_{w^2,c} + 1)$-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 for A_i which is of width w and make w copies of it, and add two new nodes s and t. We add the following edges.

- Connect the new start node s to the h-th former start node $v_{0,h}$ of the h-th copy of the ROABP A_i by edges of weight one, for all $1 \leq h \leq w$.
- 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 $1 \leq h, j \leq w$.

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 3.4.4, the polynomial $\langle \alpha, A' \rangle (\boldsymbol{x}) = \langle \alpha, A \rangle (\boldsymbol{x} + \boldsymbol{f}_{w^2,c}) \in \mathbb{F}(t)[\boldsymbol{x}]$ is $c\ell_{w^2,c}$-concentrated over $\mathbb{F}(t)$, for every $\alpha \in \mathbb{F}^{w \times w}$.

I.e. for any α such that $\langle \alpha, A' \rangle \in \mathbb{F}[t][\boldsymbol{x}] \neq 0$, there exists a monomial $\boldsymbol{x}^{\boldsymbol{a}}$ of support $< c\ell_{w^2,c}$, such that $\text{coeff}_{\langle \alpha, A' \rangle}(\boldsymbol{x}^{\boldsymbol{a}}) \in \mathbb{F}[t]$ is nonzero. So, there exists $i \in \mathbb{N}$ such that $\text{coeff}_{\langle \alpha, A'' \rangle}(\boldsymbol{x}^{\boldsymbol{a}} t^i) \in \mathbb{F}$ is nonzero. I.e. $\langle \alpha, A'' \rangle$ is $(c\ell_{w^2,c}+1)$-concentrated. By Lemma 3.4.6, it follows that $A(\boldsymbol{x} + \boldsymbol{f}_{w^2,c})$ is $(c\ell_{w^2,c} + 1)$-concentrated. □

3.5 Low Support Concentration in ROABPs

To complete the proof of Theorem 3.4.5, we need to prove that there exists a shift of quasi-polynomial cost which concentrates any width-w ROABP over n variables.

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 will show an efficient shift which achieves concentration in matrix polynomials computed by ROABPs via 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 $\ell \geq 1$, 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 $\mathrm{w}(\boldsymbol{a}) = \sum_{i=1}^{n} \mathrm{w}(i)a_i$. The definition of the weight function is extended to a subset of monomials $M' \subseteq M$ by adding the weights of the monomials in the set: $\mathrm{w}(M') = \sum_{\boldsymbol{a} \in M'} \mathrm{w}(\boldsymbol{x}^{\boldsymbol{a}})$.

Let $\mathbb{A}_k$ be a k-dimensional algebra over the field $\mathbb{F}$. For any two sets of monomials, 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 3.5.1. *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* $|S| \leq k$ *such that*

- $\forall\, \boldsymbol{a} \neq \boldsymbol{b} \in S, \quad \mathrm{w}(\boldsymbol{a}) \neq \mathrm{w}(\boldsymbol{b})$ *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* $\mathrm{w}(\boldsymbol{b}) < \mathrm{w}(\boldsymbol{a})\}$.

When a basis of the coefficients is a subset of the coefficients of the monomials, then, we can naturally define the weight of this basis as the sum of the weights of the monomials of the coefficients present in that basis. Let $\mathcal{B}_A = \{M_1, M_2, \ldots, M_r\}$ be the family of the subsets of monomials whose coefficients form a basis of the coefficient space of the polynomial $A(\boldsymbol{x}) \in \mathbb{A}_k[\boldsymbol{x}]$. The definition of the basis isolating weight assignment is such that if it exists, then, amongst all the bases of the coefficient space in $\mathcal{B}_A$, a *unique* basis has the minimum weight and the monomials in this basis have distinct weights.

Proof idea- Agrawal et al. [AGKS15, Lemma 8] presented a quasi-polynomial time construction of such a weight function $(\mathrm{w}(1), \mathrm{w}(2), \ldots, \mathrm{w}(n))$ for any polynomial $A(\boldsymbol{x}) \in \mathbb{F}^{w\times w}[\boldsymbol{x}]$ computed by an ROABP.

Lemma 3.5.2 ([AGKS15])**.** *Given n, w, d, a set of N weight functions $\{\mathrm{w}_1, \mathrm{w}_2, \ldots, \mathrm{w}_N\}$ can be constructed in time $(nwd)^{O(\log n)}$ such that for any n-variate, individual degree d polynomial $A(\boldsymbol{x}) \in \mathbb{F}^{w\times w}[\boldsymbol{x}]$ computed by a width-w ROABP, there exists $i \in [N]$ such that w_i is a basis isolating weight assignment for $A(\boldsymbol{x})$. Here, $N = (nwd)^{O(\log n)}$.*

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,

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

We will prove that A' has low support concentration. There is a standard method of proving this. We follow the ideas from [ASS13, AGKS13, FSS14] to show low support concentration. We write the coefficients of A' as linear combinations of the coefficients of A. Since they are linear combinations, there exists a transfer matrix $D^{-1}TD$ (Equation 3.23). To study the coefficients of the $(<\ell)$-support monomials of the shifted polynomial, we truncate the matrices in Equation 3.23 appropriately (Equation 3.24). We then prove that the truncated transfer matrix $D_\ell^{-1}T_\ell D$ is a rank extractor. See the section on rank extractors in the Introduction chapter.

Formulation of the shifting via matrices- Recall from Lemma 2.5.8 that the coefficients of A' are linear combinations of the coefficients of A. Adapting Equation (2.2), we have

$$\mathrm{coeff}_{A'}(\boldsymbol{x}^{\boldsymbol{a}}) = \sum_{\boldsymbol{b}\in M} \binom{\boldsymbol{b}}{\boldsymbol{a}} t^{\mathrm{w}(\boldsymbol{b}-\boldsymbol{a})} \cdot \mathrm{coeff}_A(\boldsymbol{x}^{\boldsymbol{b}}), \tag{3.22}$$

where $\binom{\boldsymbol{b}}{\boldsymbol{a}} = \prod_{i=1}^{n}\binom{b_i}{a_i}$ for any $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{N}^n$. Recall that $\binom{b_i}{a_i} = 0$ if $b_i < a_i$. Hence, $\binom{\boldsymbol{b}}{\boldsymbol{a}} = 0$ if $b_i < a_i$ for any i.

Equation (3.22) 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,

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

Similarly, let C' be the $M \times [k]$ matrix 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 (3.22) becomes

$$C' = D^{-1} T D C . \tag{3.23}$$

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

Lemma 3.5.3 (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 (3.23) with respect to the low support monomials. Recall that $M_\ell = \{\boldsymbol{a} \in M \mid \operatorname{supp}(\boldsymbol{a}) < \ell\}$. We define the matrices

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

To show that A' is ℓ-concentrated, we need to prove that $\operatorname{rank}(C'_\ell) = \operatorname{rank}(C)$. By Equation (3.23), matrix C'_ℓ can be written as

$$C'_\ell = D_\ell^{-1} T_\ell D C. \tag{3.24}$$

Since D_ℓ^{-1} is a diagonal matrix with nonzero entries, it has full rank. Hence, it suffices to show that $\mathrm{rank}(T_\ell DC) = \mathrm{rank}(C)$.

Now, recall that w is a basis isolating weight assignment. Hence, there exists a set $S \subseteq M$ such that the set of coefficients $\{\mathrm{coeff}_A(\boldsymbol{b})\}_{\boldsymbol{b}\in S}$ spans all coefficients $\mathrm{coeff}_A(\boldsymbol{a})$, for $\boldsymbol{a} \in M$. Without loss of generality, we assume that the rows and columns in all the above matrices that are indexed by M or M_ℓ are ordered according to the increasing weight $\mathrm{w}(\boldsymbol{a})$ of the indices $\boldsymbol{a}$. If there is an index $\boldsymbol{a} \notin S$ and index $\boldsymbol{b} \in S$ such that $\mathrm{w}(\boldsymbol{a}) = \mathrm{w}(\boldsymbol{b})$, then the index $\boldsymbol{a}$ is put before the index $\boldsymbol{b}$. Amongst all the indices with the same weight, only one index can belong to the set S. The other rows with the same weight can be arranged in an arbitrary order.

The set $\{\mathrm{coeff}_A(\boldsymbol{b})\}_{\boldsymbol{b}\in S}$ is the spanning set of all the rows of C. In terms of the coefficient matrix C, for any $\boldsymbol{a} \in M$ we can write

$$C(\boldsymbol{a}, \cdot) \in \mathrm{span}\{C(\boldsymbol{b}, \cdot) \mid \boldsymbol{b} \in S \text{ and } \mathrm{w}(\boldsymbol{b}) < \mathrm{w}(\boldsymbol{a})\}. \tag{3.25}$$

This property can be used to factorize the coefficients matrix C as follows. Let $S = \{\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_{k'}\}$ for some $k' \leq k$. Let C_0 be the $k' \times k$ sub-matrix of C whose i-th row is $C(\boldsymbol{s}_i, \cdot)$, i.e. $C_0(i, \cdot) = C(\boldsymbol{s}_i, \cdot)$. By (3.25), 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 (3.25), 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 $\boldsymbol{e}_i$,

$$\Gamma(\boldsymbol{s}_i, i) = 1 \;\text{ and }\; \forall\, \boldsymbol{a} \neq \boldsymbol{s}_i,\; \mathrm{w}(\boldsymbol{a}) \leq \mathrm{w}(\boldsymbol{s}_i) \implies \Gamma(\boldsymbol{a}, i) = 0\,. \tag{3.26}$$

Recall that our goal is to show $\mathrm{rank}(T_\ell DC) = \mathrm{rank}(C)$. 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 $\mathrm{rank}(T_\ell DC) = \mathrm{rank}(T_\ell D\Gamma C_0) = \mathrm{rank}(RC_0) = \mathrm{rank}(C_0) = \mathrm{rank}(C)$.

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

$$R(\cdot, j) = \sum_{\boldsymbol{a} \in M} T_\ell(\cdot, \boldsymbol{a})\, \Gamma(\boldsymbol{a}, j)\, t^{\mathrm{w}(\boldsymbol{a})}\,. \tag{3.27}$$

By (3.26), the term with the lowest degree in Equation (3.27) 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 3.5.4 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.

For any polynomial $V(\boldsymbol{x}) = \sum_{\boldsymbol{a}} v_{\boldsymbol{a}} \boldsymbol{x}^{\boldsymbol{a}} \in \mathbb{F}[\boldsymbol{x}]$, when it is shifted by 1, the new coefficients are given by the equation

$$\mathrm{coeff}_{V'}(\boldsymbol{x}^{\boldsymbol{a}}) = \sum_{\boldsymbol{b}} \binom{\boldsymbol{b}}{\boldsymbol{a}} v_{\boldsymbol{b}} = T(\boldsymbol{a}, \cdot)\, \boldsymbol{v}\,,$$

where the matrix T comes from Equation 3.23. The matrix T_ℓ was obtained by truncating the matrix T to its rows indexed by the $(< \ell)$-support monomials. To show that any k columns of T_ℓ are linearly independent, we need to show that when the any k-sparse

polynomial $\boldsymbol{v}(\boldsymbol{x}) \in \mathbb{F}[\boldsymbol{x}]$ is shifted by $\mathbf{1}$, it becomes ℓ-concentrated.

Lemma 3.5.4. *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 $M_\ell \times S$ sub-matrix 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 1: $V'(\boldsymbol{x}) = V(\boldsymbol{x} + \mathbf{1})$. From Equation (3.22) 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*.

Claim 3.5.5. *Let $V(\boldsymbol{x}) \in \mathbb{F}[\boldsymbol{x}]$ be a nonzero 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 n, the number of variables. 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. □

Unifying many maps into one using Lagrange interpolation - We can use Lemma 3.5.3 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 3.5.3, 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. $\mathcal{F}$ can be generated in time $(ndw)^{O(\log n)}$.

Thus, by Lemma 3.5.3, 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 3.4.2 and 3.4.4 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.

Theorem 3.5.6. *There exists an n-tuple $\boldsymbol{f}(t) \in \mathbb{F}[t]^n$, of degree bounded by $(ndw)^{O(\log n)}$ and computable in time $(ndw)^{O(\log n)}$, such that given any polynomial $A(\boldsymbol{x}) \in \mathbb{F}^{w\times w}[\boldsymbol{x}]$ (or $\mathbb{F}^{1\times w}[\boldsymbol{x}]$, or $\mathbb{F}[\boldsymbol{x}]$) computed by an ROABP over n variables, of individual degree d and of*

width w, $A(\boldsymbol{x} + \boldsymbol{f}(t))$ *is* $\log(w^2+1)$*-concentrated.*

Proof. We take the family $\mathcal{F}$ of maps for the polynomials computed by n-variate, individual degree d ROABPs of width w. The maps in $\mathcal{F}$ are of degree $(ndw)^{O(\log n)}$ and there are $(ndw)^{O(\log n)}$ of them. For any polynomial $A(\boldsymbol{x}) \in \mathbb{F}^{w\times w}[\boldsymbol{x}]$ computed by an ROABP of the above description, there exists a map $f \in \mathcal{F}$ such that $A(\boldsymbol{x}+f)$ is $\log(w^2+1)$-concentrated.

By Lemma 2.5.9, there exists a single univariate map $\boldsymbol{f}$ of degree $(ndw)^{O(\log n)}$ that $\log(w^2+1)$-concentrates the ROABPs.

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 3.4.6 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 $T \in \mathbb{F}^{w\times 1}$. □

3.6 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.

In this thesis chapter, the hitting set for a sum of ROABPs is obtained by shifting the ROABPs, so that the polynomial computed by the ROABPs is ℓ-concentrated. A hitting set which does not go through shifting and concentrating the polynomial would be interesting. The earlier hitting sets for depth-3 set multilinear circuits and ROABPs made use of shifting and concentrating techniques [ASS13, FSS14]. But, [AGKS15] had mapped

all the variables to polynomials directly. This may be possible for sum of ROABPs also.

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.

Chapter 4

Sparse, Invertible Constant-Width ROABP

Abstract

In this chapter, we explore the model of read-once arithmetic branching programs (ROABP) where the factor-matrices are *invertible* (called invertible-factor ROABP). We design a hitting-set in time $\mathsf{poly}(n^{w^2})$ for width-w invertible-factor ROABP. Further, we could do *without* the invertibility restriction when $w = 2$. Before this, the best result for width-2 ROABP was quasi-polynomial time (Forbes-Saptharishi-Shpilka, STOC 2014).

4.1 Introduction

Our result is for ROABP with the restriction that all the matrices in the matrix product, except the left-most and the right-most matrices, are invertible. We give a blackbox test for this class of ROABP. Our test works in *polynomial time* if the dimension of the matrices is constant.

Note that the class of ABP, where the factor matrices are invertible, is quite powerful,

as Ben-Or and Cleve [BOC92] actually reduce formulas to width-3 ABP with *invertible* factors. Saha, Saptharishi and Saxena [SSS09] reduce PIT for depth-3 circuits to PIT for width-2 ABP with invertible factors. The class of invertible ROABPs subsume the class of diagonal circuits through Saxena's trick ([Sax08]). But the read-once constraint seems to restrict the computing power of ABP, because of which we can find a hitting set for invertible ROABPs of constant width. Interestingly, an analogous class of read-once Boolean branching programs called *permutation branching programs* has been studied recently [KNP11, De11, Ste12]. These works give pseudo-random generators for this class (for constant width) with seed-length $O(\log n)$. In other words, they give polynomial size sample set which can fool these programs. Our polynomial size hitting sets for the arithmetic setting is analogous to this result.

Our algorithm works even when the factor matrices have their entries as general sparse polynomials (still over disjoint sets of variables) instead of univariate polynomials. Gurjar et al. recently found a $n^{O(\log w)}$ time PIT for ROABPs over fields of zero characteristic (or with large enough characteristic) [GKS17]. Hence, it has a much better dependence on the width w of the ROABP. But, their PIT [GKS17] is grey-box. I.e. the variable sequence of the ROABP is input to the PIT algorithm. The $\mathsf{poly}(n^{w^2})$ time hitting set presented in this chapter is still the best known for constant width invertible-factor ROABP over fields of small characteristic.

If the matrices are 2×2, we do not need the assumption of invertibility (see Theorem 4.6.3). Here again, there is a comparable result in the Boolean setting. Pseudo-random generators with $O(\log n)$ seed-length (polynomial size sample set) are known for width-2 Boolean branching programs [BDVY13].

We will show a hitting-set for a sparse-factor ROABP $D_0 \left(\prod_{i=1}^{d} D_i\right) D_{d+1}$, where the D_is are *invertible* matrices, for all $i \in [d]$. Hence, we name this model *sparse-invertible-factor ROABP*. A polynomial $C(\boldsymbol{x})$ computed by a s-sparse-factor width-w ROABP can be written as $D_0 \left(\prod_{i=1}^{d} D_i\right) D_{d+1}$, where $D_0 \in \mathbb{F}^{1\times w}$, $D_{d+1} \in \mathbb{F}^{w\times 1}$, $D_i \in \mathbb{F}^{w\times w}[\boldsymbol{x}_i]$ is an s-sparse polynomial for all $i \in [d]$, and $\boldsymbol{x}_1, \boldsymbol{x}_2 \ldots, \boldsymbol{x}_d$ are disjoint sets of variables.

For a polynomial D, let its sparsity $\mathsf{s}(D)$ be the number of monomials in D with

nonzero coefficients.

Theorem 4.1.1. *Let $\mathbf{x} = \mathbf{x}_1 \sqcup \cdots \sqcup \mathbf{x}_d$, with $|\mathbf{x}| = n$. Let $C(\mathbf{x}) = D_0^\top D D_{d+1} \in \mathbb{F}[\mathbf{x}]$ be a polynomial with $D_0, D_{d+1} \in \mathbb{F}^w$, $D(\mathbf{x}) = \prod_{i=1}^d D_i(\mathbf{x}_i)$, and for all $i \in [d]$, $D_i \in \mathbb{F}^{w\times w}[\mathbf{x}_i]$ is an invertible matrix. For all $i \in [d]$, D_i has degree bounded by δ and sparsity $\mathsf{s}(D_i) \leq s$. Then there is a hitting-set of size $\mathsf{poly}((n\delta s)^{w^2 \log w})$ for $C(\mathbf{x})$.*

Remark [1]. *If the width w is constant, then it is clear that we get a polynomial sized hitting-set.*

Remark [2]. *If the D_is are univariate, then we get a $(n\delta)^{O(w^2)}$ sized hitting-set. The proof is presented along with the proof of Theorem 4.1.1.*

4.2 Preliminaries

4.2.1 Notations and definitions

Let $\boldsymbol{x}$ be the set of variables $\{x_1, x_2 \ldots, x_n\}$. For an exponent $\boldsymbol{a} = (a_1, a_2, \ldots, a_m) \in \mathbb{N}^m$, and for a set of variables $\mathbf{y} = \{y_1, y_2, \ldots, y_m\}$, $\mathbf{y}^{\boldsymbol{a}}$ will denote ${y_1}^{a_1} {y_2}^{a_2} \ldots {y_m}^{a_m}$. For any $\boldsymbol{a} \in \mathbb{N}^n$, support of the monomial $\mathbf{x}^{\boldsymbol{a}}$ is defined as $\mathrm{S}(\boldsymbol{a}) = \{i \in [n] \mid a_i \neq 0\}$ and support size is defined as $\mathrm{s}(\boldsymbol{a}) = |\mathrm{S}(\boldsymbol{a})|$.

We will shift $C(\boldsymbol{x})$ by univariate polynomials, say, given by the map $\phi\colon \mathbf{t} \to \{t^a\}_{a\geq 0}$, where $\mathbf{t} = \{t_1, t_2, \ldots, t_n\}$. The ϕ is said to be an efficient map if $\phi(t_i)$ is efficiently computable, for each $i \in [n]$.

Let the matrix product $D(\mathbf{x}) = \prod_{i=1}^d D_i$ correspond to an ROABP such that $D_i \in \mathbb{F}^{w\times w}[\mathbf{x}_i]$ for all $i \in [d]$. Let n_i be the cardinality of $\mathbf{x}_i$ and let $n = \sum_{i=1}^d n_i$. Viewing D_i as belonging to $\mathbb{F}^{w\times w}[\mathbf{x}_i]$, one can write $D_i = \sum_{\boldsymbol{a}\in\mathbb{N}^{n_i}} D_{i,\boldsymbol{a}} \mathbf{x}_i^{\boldsymbol{a}}$, where $D_{i,\boldsymbol{a}} \in \mathbb{F}^{w\times w}$, for all $\boldsymbol{a} \in \mathbb{N}^{n_i}$. In particular $D_{i,\mathbf{0}}$ refers to the constant part of the polynomial D_i.

Any monomial $\mathbf{x}^{\boldsymbol{a}}$ for $\boldsymbol{a} \in \mathbb{N}^n$, can be seen as a product $\prod_{i=1}^d {\mathbf{x}_i}^{\boldsymbol{a}_i}$, where $\boldsymbol{a}_i \in \mathbb{N}^{n_i}$ for all $i \in [d]$, such that $\boldsymbol{a} = (\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_d)$.

Definition 4.2.1 (Block-support and block-support size of a monomial)**.** *We define the* block-support *of the monomial $\boldsymbol{a}$,* $\mathrm{bS}(\boldsymbol{a})$ *as* $\{i \in [d] \mid \boldsymbol{a}_i \neq \mathbf{0}\}$ *and* block-support size *of $\boldsymbol{a}$,* $\mathrm{bs}(\boldsymbol{a}) = |\mathrm{bS}(\boldsymbol{a})|$.

Thus, the block-support of a monomial is the set of blocks which contribute non-trivially to the monomial. The coefficient of the monomial $\boldsymbol{x}^{\boldsymbol{a}}$ is $D_{\boldsymbol{a}} = \prod_{i=1}^{d} D_{i,\boldsymbol{a}_i}$. Observe that when $i \notin \mathrm{bS}(\boldsymbol{a})$, the i-th block contributes its constant part, $D_{i,\mathbf{0}}$ to the coefficient.

We will now import some terminology from the 'string' data-type to the coefficient of a monomial. Analogous to the definition of a subsequence of a string, we define a substring of a coefficient.

Definition 4.2.2 (Substring)**.** *The coefficient $D_{\boldsymbol{b}}$ is called a* substring *of the coefficient $D_{\boldsymbol{a}}$ if* $\mathrm{bs}(\boldsymbol{b}) < \mathrm{bs}(\boldsymbol{a})$ *and $\boldsymbol{b}_i = \boldsymbol{a}_i$ whenever $\boldsymbol{b}_i \neq \mathbf{0}$. We will use the operator* $\mathrm{substrings}(\boldsymbol{a})$ *to denote the set of monomials $\{\boldsymbol{b} \mid D_{\boldsymbol{b}}$ is a substring of $D_{\boldsymbol{a}}\}$.*

Consider the toy example $D = (A_1 + B_{11}x_1 + B_{12}{x_1}^2)(A_2 + B_2x_2)(A_3 + B_3x_3 + B_4x_3x_4)$. Here, $D_{i,\mathbf{0}} = A_i$, for all i and the block-support of the monomial $x_1x_3x_4$ is $\{1, 3\}$. $A_1B_2B_3$ is a substring of $B_{12}B_2B_3$, whereas $B_{11}B_2B_3$ is not.

Analogous to the definition of a prefix of a string, we define a prefix of a coefficient.

Definition 4.2.3 (Prefix)**.** *A substring $D_{\boldsymbol{b}}$ of the coefficient $D_{\boldsymbol{a}}$ is called its* prefix *if* $\forall i\ (\boldsymbol{b}_i \neq \boldsymbol{a}_i \implies (\forall j \geq i, \boldsymbol{b}_j = \mathbf{0}))$.

In the above example, it means that only the trailing Bs can be replaced with the corresponding As. Thus, $B_{12}A_2A_3$ is a prefix of $B_{12}B_2B_4$, whereas $B_{12}A_2B_4$ is not, though both are its substrings.

Definition 4.2.4 ($D_{\boldsymbol{b}^{-1}\boldsymbol{a}}$, when the $D_{i,\mathbf{0}}$s are invertible and $D_{\boldsymbol{b}}$ is a prefix of $D_{\boldsymbol{a}}$)**.** *Observe that when the $D_{i,\mathbf{0}}$s are invertible, if $D_{\boldsymbol{b}}$ is a prefix of $D_{\boldsymbol{a}}$, then we can write $D_{\boldsymbol{a}} = D_{\boldsymbol{b}}A^{-1}B$, where $A = \prod_{i=r+1}^{d} D_{i,\mathbf{0}}$ and $B = \prod_{i=r+1}^{d} D_{i,\boldsymbol{a}_i}$ with $r = \max\{\mathrm{bS}(\boldsymbol{b})\}$. For a prefix $D_{\boldsymbol{b}}$ of the coefficient $D_{\boldsymbol{a}}$, we will denote the matrix product $A^{-1}B$ as $D_{\boldsymbol{b}^{-1}\boldsymbol{a}}$.*

4.2.2 Proof Idea

We find a hitting-set by showing a *low-support concentration*. Low-support concentration in the polynomial $D(\boldsymbol{x}) = \prod_{i=1}^{d} D_i$ means that the coefficients of the low-support monomials in $D(\boldsymbol{x})$ span the whole coefficient space of $D(\boldsymbol{x})$.

Throughout this chapter, we will let $\ell = w^2 + 1$ and $\ell' = \lceil \log w^2 \rceil + 1$. There are

four steps that we take to prove low-support concentration in the sparse, invertible-factor, width-w ROABP D.

1. Given an invertible matrix $D_i \in \mathbb{F}^{w\times w}[\boldsymbol{x}_i]$, we use a shift, so that the constant part of the shifted matrix is invertible.
2. Assuming the constant part of the shifted matrix D is invertible, we show that any coefficient with block support $(= \ell)$ is linearly dependent on its substrings with $(< \ell)$ block support (Lemma 4.3.3).
3. We use this lemma to show that any coefficient in D is linearly dependent on its substrings with $< \ell$ block support (Lemma 4.3.4).
4. We show that the shifted polynomial has ℓ'-concentration within each of the blocks.

The novel part of this chapter is in the second step.

We will give the proof idea for the second step through a toy example. Consider $D = (A_1 + B_1x_1)(A_2 + B_2x_2)\cdots(A_n + B_nx_n)$, where each of the A_is are invertible and $A_i, B_i \in \mathbb{F}^{w\times w}, \forall i$. Take the ℓ-block-support monomial $x_{[\ell]} = x_1x_2\cdots x_\ell$. We will show that its coefficient is linearly dependent on its substrings with $(< \ell)$ block-support. Let $M_j = \prod_{i=1}^{j} B_i \prod_{i=j+1}^{\ell} A_i$, for $0 \le j \le \ell$. Consider the set of matrices $\{M_j\}_{j=0}^{\ell}$. These $\ell + 1$ matrices lie in $\mathbb{F}^{w\times w}$. Hence, there exists a $r \in [\ell]$ such that $M_r = \sum_{j=0}^{r-1} \gamma_j M_j$, where $\gamma_j \in \mathbb{F}$. All the M_js on the right hand side have $< r$ many Bs. Since the A_is are invertible, we can post-multiply throughout by $(\prod_{i=r+1}^{\ell} A_i)^{-1} \prod_{i=r+1}^{\ell} B_i \prod_{i=\ell+1}^{n} A_i$ to obtain that the coefficient of $x_{[\ell]}$ is linearly dependent on strictly smaller block-support coefficients.

The third step is a simple extension of the above idea. For the first and fourth steps, we use an appropriate shift (Section 4.4). The sparsity of D_i is used crucially here.

For $D_{i,\mathbf{0}}$ to be invertible, we have to assume that $D_i(\mathbf{x}_i)$ is an invertible matrix for all $i \in [d]$. For the shifted polynomial $D'_i(\mathbf{x}_i) = D_i(\mathbf{x}_i + \phi(\mathbf{t}_i))$, its constant term $D'_{i,\mathbf{0}}$ is just an evaluation of $D_i(\mathbf{x})$, i.e. $D_i|_{\mathbf{x}_i=\phi(\mathbf{t}_i)}$. Hence, if $\det(D_i(\mathbf{x}_i)) = 0$ (viewing $D_i(\mathbf{x}_i)$ as an element in $(\mathbb{F}[\mathbf{x}_i])^{w\times w}$), then $\det(D'_{i,\mathbf{0}}) = 0$. This means that if $\det(D_i(\mathbf{x}_i)) = 0$, then even after shifting, $D'_{i,\mathbf{0}}$ cannot become invertible.

4.3 ℓ-block-concentration when $D_{i,\mathbf{0}}s$ are invertible

In this section, we will prove that when $D_{i,\mathbf{0}}$ are invertible for all $1 \leq i \leq d$, then D is ℓ-block-concentrated.

Definition 4.3.1 (ℓ-Block-concentration)**.** *$D(\boldsymbol{x})$ is ℓ-block-concentrated if any coefficient in $D(\boldsymbol{x})$ is dependent on coefficients in D with block-support $\leq \ell - 1$.*

First, let us prove that a particular kind of dependency can be lifted.

Lemma 4.3.2. *Let D be an ROABP, such that $D_{i,\mathbf{0}}$ are invertible for all $1 \leq i \leq d$. Let $D_{\boldsymbol{a}}$ be a prefix of $D_{\boldsymbol{a}^*}$. Then, if $D_{\boldsymbol{a}}$ is linearly dependent on its substrings, then $D_{\boldsymbol{a}^*}$ is linearly dependent on its substrings.*

Proof. Since $D_{\boldsymbol{a}}$ is a prefix of $D_{\boldsymbol{a}^*}$,

$$D_{\boldsymbol{a}^*} = D_{\boldsymbol{a}} D_{\boldsymbol{a}^{-1}\boldsymbol{a}^*}. \tag{4.1}$$

Let the dependence of $D_{\boldsymbol{a}}$ on its substrings be the following:

$$D_{\boldsymbol{a}} = \sum_{\boldsymbol{b} \in \text{substrings}(\boldsymbol{a})} \gamma_{\boldsymbol{b}} D_{\boldsymbol{b}}.$$

Using Equation (4.1) we can write,

$$D_{\boldsymbol{a}^*} = \sum_{\boldsymbol{b} \in \text{substrings}(\boldsymbol{a})} \gamma_{\boldsymbol{b}} D_{\boldsymbol{b}} D_{\boldsymbol{a}^{-1}\boldsymbol{a}^*}.$$

Now, we just need to show that for any substring $D_{\boldsymbol{b}}$ of $D_{\boldsymbol{a}}$, $D_{\boldsymbol{b}} D_{\boldsymbol{a}^{-1}\boldsymbol{a}^*}$ is a valid coefficient of some monomial in $D(\boldsymbol{x})$ and also that it is a substring of $D_{\boldsymbol{a}^*}$.

Let $r = \max\{\text{bS}(\boldsymbol{a})\}$. Recall that $D_{\boldsymbol{a}^{-1}\boldsymbol{a}^*} = A^{-1}B$, where $A = \prod_{i=r+1}^{d} D_{i,\mathbf{0}}$ and $B = \prod_{i=r+1}^{d} D_{i,\boldsymbol{a}_i^*}$. Thus, $D_{\boldsymbol{b}} D_{\boldsymbol{a}^{-1}\boldsymbol{a}^*}$ is the coefficient of $\mathbf{x}^{\boldsymbol{b}^*} = \prod_{i=1}^{r} \mathbf{x}_i^{\boldsymbol{b}_i} \prod_{i=r+1}^{d} \mathbf{x}_i^{\boldsymbol{a}_i^*}$. Since $\boldsymbol{b} \in \text{substrings}(\boldsymbol{a})$ and $\boldsymbol{a} \in \text{substrings}(\boldsymbol{a}^*)$, $\boldsymbol{b} \in \text{substrings}(\boldsymbol{a}^*)$. From these two facts, it is easy to see that $D_{\boldsymbol{b}} D_{\boldsymbol{a}^{-1}\boldsymbol{a}^*}$ is a substring of $D_{\boldsymbol{a}^*}$.

□

We will now prove the existence of a dependency for any ℓ-block-support coefficient.

Lemma 4.3.3. *Let $D_{\boldsymbol{a}}$ be a coefficient in D with $\mathrm{bs}(\boldsymbol{a}) = \ell$. Then $D_{\boldsymbol{a}}$ $\mathbb{F}$-linearly depends on its substrings.*

Proof. Consider the set of coefficients $\{M_0, M_1, \ldots, M_\ell\}$, where M_j is the prefix of $D_{\boldsymbol{a}}$ with block support size j, for $0 \leq j < \ell$. And $M_\ell = D_{\boldsymbol{a}}$. These $\ell+1$ vectors lie in $\mathbb{F}^{w\times w} \cong \mathbb{F}^{\ell-1}$. Hence, there exists an $r \in [\ell]$ such that M_r is linearly dependent on $\{M_j\}_{j=0}^{r-1}$. (Note that $M_r = \mathbf{0}$ is also a dependency.) The coefficients in the set $\{M_j\}_{j=0}^{r-1}$ are prefixes of M_r, and thus, substrings of M_r.

Now, by applying Lemma 4.3.2, we conclude that $D_{\boldsymbol{a}}$ is dependent on its substrings. □

Lemma 4.3.3 implies that coefficients with block-support ℓ depend on coefficients with block-support $\leq \ell - 1$. We will next show that this is true for all coefficients of $D(\boldsymbol{x})$.

We will now prove that when the $D_{i,\mathbf{0}}$s are invertible, D is ℓ-block-concentrated.

Lemma 4.3.4 (ℓ-Block-concentration)**.** *Let $D(\boldsymbol{x}) = \prod_{i=1}^{d} D_i(\mathbf{x}_i) \in \mathbb{F}^{w\times w}[\mathbf{x}]$ be a polynomial with $D_{i,\mathbf{0}}$ being invertible for each $i \in [d]$. Then $D(\boldsymbol{x})$ has ℓ-block-concentration.*

Proof. We will actually prove that for any coefficient $D_{\boldsymbol{a}}$ with $\mathrm{bs}(\boldsymbol{a}) \geq \ell$ (the case when $\mathrm{bs}(\boldsymbol{a}) < \ell$ is trivial),

$$D_{\boldsymbol{a}} \in \mathrm{span}\{D_{\boldsymbol{b}} \mid \boldsymbol{b} \in \mathbb{N}^n,\ \boldsymbol{b} \in \mathrm{substrings}(\boldsymbol{a}) \text{ and } \mathrm{bs}(\boldsymbol{b}) \leq \ell - 1\}.$$

We will prove this by induction on the block-support of $D_{\boldsymbol{a}}$, $\mathrm{bs}(\boldsymbol{a})$.

Base case: When $\mathrm{bs}(\boldsymbol{a}) = \ell$, it has been already shown in Lemma 4.3.3.

Induction Hypothesis: For any coefficient $D_{\boldsymbol{a}}$ with $\mathrm{bs}(\boldsymbol{a}) = i - 1$ for $i - 1 \geq \ell$,

$$D_{\boldsymbol{a}} \in \mathrm{span}\{D_{\boldsymbol{b}} \mid \boldsymbol{b} \in \mathbb{N}^n,\ \boldsymbol{b} \in \mathrm{substrings}(\boldsymbol{a}) \text{ and } \mathrm{bs}(\boldsymbol{b}) \leq \ell - 1\}.$$

Induction step: Let us take a coefficient $D_{\boldsymbol{a}}$ with $\mathrm{bs}(\boldsymbol{a}) = i$. Consider the unique prefix $D_{\boldsymbol{a}'}$ of $D_{\boldsymbol{a}}$ such that $\mathrm{bs}(\boldsymbol{a}') = i - 1$.

As $\mathrm{bs}(\boldsymbol{a}') = i - 1$, by our induction hypothesis, $D_{\boldsymbol{a}'}$ is linearly dependent on its substrings. So, from Lemma 4.3.2, $D_{\boldsymbol{a}}$ is linearly dependent on its substrings. In other words,

$$D_{\boldsymbol{a}} \in \mathrm{span}\{D_{\boldsymbol{b}} \mid \boldsymbol{b} \in \mathrm{substrings}(\boldsymbol{a}) \text{ and } \mathrm{bs}(\boldsymbol{b}) \leq i - 1\}. \tag{4.2}$$

Again, by our induction hypothesis, for any coefficient $D_{\boldsymbol{b}}$, with $\mathrm{bs}(\boldsymbol{b}) \leq i-1$,

$$D_{\boldsymbol{b}} \in \mathrm{span}\{D_{\boldsymbol{c}} \mid \boldsymbol{c} \in \mathrm{substrings}(\boldsymbol{b}) \text{ and } \mathrm{bs}(\boldsymbol{c}) \leq \ell - 1\}. \tag{4.3}$$

Combining Equations (4.2) and (4.3), we get,

$$D_{\boldsymbol{a}} \in \mathrm{span}\{D_{\boldsymbol{c}} \mid \boldsymbol{c} \in \mathrm{substrings}(\boldsymbol{a}) \text{ and } \mathrm{bs}(\boldsymbol{c}) \leq \ell - 1\}.$$

□

In Lemma 4.3.4, we had assumed that the constant term $D_{i,\mathbf{0}}$ is invertible for every block D_i. In the next subsection, we will show how to achieve this invertibility and low-support concentration within each block D_i.

4.4 Achieving invertibility and low-support concentration through shifting

Let the shifted polynomial $D' = D(\boldsymbol{x}+\phi(\mathbf{t}))$. Then, $D' = \prod_{i=1}^{d} D'_i$ and $D'_{i,\mathbf{0}}$ is the constant part of D'_i. Shifting will serve two purposes.

- Recall that for Lemmas 4.3.2 and 4.3.4, we need invertibility of the constant term $D'_{i,\mathbf{0}}$ in D'_i, for all $i \in [d]$.
- D'_i should have low-support concentration after shifting.

Now, we want a shift for D_i which would ensure that $\det(D'_{i,\mathbf{0}}) \neq 0$ and that D'_i has low-support concentration. For both the goals we use the sparsity of the polynomial.

Definition 4.4.1. *For a polynomial p, let its sparsity set $\mathsf{S}(p)$ be the set of monomials in p with nonzero coefficients and $\mathsf{s}(p)$ be its sparsity, i.e. $\mathsf{s}(p) = |\mathsf{S}(p)|$. Let $\mathsf{S}^w(p) = \{m_1 m_2 \cdots m_w \mid m_i \in \mathsf{S}(p),\ \forall i \in [w]\}$.*

A map ϕ over $\mathbf{t}$ *separates* all the monomials in a set S if for any two monomials $\mathbf{t}^{\boldsymbol{a}_1}, \mathbf{t}^{\boldsymbol{a}_2} \in S$, $\phi(\mathbf{t}^{\boldsymbol{a}_1}) \neq \phi(\mathbf{t}^{\boldsymbol{a}_2})$.

Let us now characterize the shift which makes the determinant of $D'_{i,\mathbf{0}}$ nonzero.

Lemma 4.4.2. *Suppose D_i is invertible. Let $\phi\colon \mathbf{t} \to \{t^i\}_{i=0}^{\infty}$ be a monomial map which separates all the monomials in $\mathsf{S}^w(D_i)$. Then, the constant term $D'_{i,\mathbf{0}}$ of the shifted polynomial $D'_i = D_i(\boldsymbol{x} + \phi(\mathbf{t}))$ is invertible.*

Proof. Observe that $\mathsf{S}(\det(D_i)) \subseteq \mathsf{S}^w(D_i)$.

Since ϕ separates all the monomials in $\det(D_i(\mathbf{t}))$ and since $\det(D_i) \neq 0$, we get $\det(D_i|_{\mathbf{x}=\phi(\mathbf{t})}) \neq 0$. Hence, $\det(D'_{i,\mathbf{0}}) = \det(D_i|_{\mathbf{x}=\phi(\mathbf{t})}) \neq 0$. □

In Lemma 3.5.3, we prove that shifting by a basis isolating weight assignment gives concentration. The proof of Lemma 4.4.3 uses this. Let w be basis isolating weight assignment for D (Definition 3.5.1). Then, $D(\boldsymbol{x} + t^{\mathrm{w}})$ has $(\lceil \log w^2 \rceil + 1)$-concentration. Recall that $\ell' = \lceil \log w^2 \rceil + 1$.

Lemma 4.4.3. *Let $\phi\colon \mathbf{t} \to \{t^i\}_{i=0}^{\infty}$ be a monomial map which separates all the monomials in $\mathsf{S}(D_i)$. Then, $D'_i = D_i(\boldsymbol{x} + \phi(\mathbf{t}))$ is ℓ'-concentrated.*

Proof. D_i is a polynomial over a w^2-dimensional algebra, $\mathbb{F}^{w\times w}$. A map ϕ which separates all the monomials in $\mathsf{S}(D_i)$ is trivially a basis isolating weight assignment for D_i. Thus, by Lemma 3.5.3, $D_i(\boldsymbol{x} + \phi(\mathbf{t}))$ is $(\lceil \log w^2 \rceil + 1)$-concentrated. □

We will now show how to find such a map ϕ.

Lemma 4.4.4. *Let $D(\mathbf{x}) = \prod_{i=1}^{d} D_i(\mathbf{x}_i)$ be a polynomial in $\mathbb{F}^{w\times w}[\mathbf{x}]$ such that for all $i \in [d]$, $\det(D_i) \neq 0$, D_i has degree bounded by δ and the sparsity $\mathsf{s}(D_i) \leq s$. Let $M = n^2 s^{2w} \log(w\delta)$. There is a set of M monomial maps with degree bounded by $2M \log M$ such that for at least one of the maps ϕ, all D'_is are ℓ'-concentrated and all $D'_{i,\mathbf{0}}$s are invertible, where $D' = D(\boldsymbol{x} + \phi(\mathbf{t}))$.*

Proof. We will provide a map ϕ that satisfies the pre-conditions of Lemmas 4.4.2 and 4.4.3. This will ensure that all D'_is are ℓ'-concentrated and all $D'_{i,\mathbf{0}}$s are invertible.

Observe that a map which separates all the monomials in $\mathsf{S}^w(D_i)$ also separates all the monomials in $\mathsf{S}(D_i)$. This can be proved by considering the two monomials $M_1 = m_1 m_2 \dots m_w$ and $M'_1 = m'_1 m_2 \dots m_w$, where $m_j \in \mathsf{S}(D_i), \forall j \in [w]$ and $m'_1 \in \mathsf{S}(D_i)$. $M_1, M'_1 \in \mathsf{S}^w$. Thus, if ϕ separates M_1 and M'_1, then ϕ should also separate m_1 and m'_1.

Hence, it is enough if ϕ separates all the monomial pairs in $\mathsf{S}^w(D_i)$, for $i \in [d]$ simultaneously. There are n variables, the number of monomial pairs is $\leq d \cdot s^{2w} \leq n \cdot s^{2w}$ and degree of the monomials (in $\mathsf{S}^w(D_i)$) is bounded by $w \cdot \delta$. Hence, by Lemma 2.3.2, $M = n^2 s^{2w} \log(w\delta)$ suffices. □

4.5 Concentration in $D(\boldsymbol{x})$

Now, we want to show that if $D(\mathbf{x}) = \prod_{i=1}^{d} D_i$ has low-block-concentration, and moreover if each D_i has low-support concentration, then $D(\boldsymbol{x})$ has an appropriate low-support concentration.

Lemma 4.5.1 (Composition)**.** *If $D(\boldsymbol{x})$ has ℓ-block-concentration and $D_i(\mathbf{x}_i)$ has ℓ'-support concentration for all $i \in [d]$ then $D(\boldsymbol{x})$ has $((\ell-1)(\ell'-1)+1)$-support concentration.*

Proof. We have to prove that the any monomial $\boldsymbol{x}^{\boldsymbol{a}}$,

$$D_{\boldsymbol{a}} = \operatorname{coeff}_D(\boldsymbol{x}^{\boldsymbol{a}}) \in \operatorname{span}\left\{\operatorname{coeff}_D(\boldsymbol{x}^{\boldsymbol{c}}) \mid \mathrm{s}(\boldsymbol{c}) \leq (\ell-1)(\ell'-1)\right\}.$$

Since $D(\boldsymbol{x})$ has ℓ-block-concentration,

$$D_{\boldsymbol{a}} \in \operatorname{span}\left\{D_{\boldsymbol{b}} \mid \mathrm{bs}(\boldsymbol{b}) < \ell\right\}. \tag{4.4}$$

Recall that as D_i's are polynomials over disjoint sets of variables, any coefficient $D_{\boldsymbol{b}}$ in $D(\boldsymbol{x})$ can be written as

$$D_{\boldsymbol{b}} = \prod_{i=1}^{d} D_{i,\boldsymbol{b}_i}, \tag{4.5}$$

where $\boldsymbol{b} = (\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_d)$ and $D_{i,\boldsymbol{b}_i}$ is the coefficient corresponding to the monomial $\mathbf{x}_i^{\boldsymbol{b}_i}$ in D_i for all $i \in [d]$. Also, $|\{i : \boldsymbol{b}_i \neq \mathbf{0}\}| < \ell$.

From ℓ'-support concentration of $D_i(\mathbf{x}_i)$, we know that for any coefficient $D_{i,\boldsymbol{b}_i}$,

$$D_{i,\boldsymbol{b}_i} \in \operatorname{span}\{D_{i,\boldsymbol{c}_i} \mid \boldsymbol{c}_i \in \mathbb{N}^{n_i},\ \mathrm{s}(\boldsymbol{c}_i) \leq \ell' - 1\}. \tag{4.6}$$

Using Equations (4.5) and (4.6), we can write the following for any coefficient $D_{\boldsymbol{b}}$:

$$D_{\boldsymbol{b}} \in \text{span}\left\{ \prod_{i=1}^{d} D_{i,\boldsymbol{c}_i} \mid \boldsymbol{c}_i \in \mathbb{N}^{n_i},\ \text{s}(\boldsymbol{c}_i) \leq \ell' - 1,\ \forall i \in [d] \right.$$
$$\left. \text{and } \boldsymbol{c}_i = \boldsymbol{0},\ \forall i \notin \text{bS}(\boldsymbol{b}) \right\}.$$

Note that the product $\prod_{i=1}^{d} D_{i,\boldsymbol{c}_i}$ will be the coefficient of a monomial $\mathbf{x}^{\boldsymbol{c}}$ such that $\text{bS}(\boldsymbol{c}) \subseteq \text{bS}(\boldsymbol{b})$ because $\boldsymbol{c}_i = \boldsymbol{0},\ \forall i \notin \text{bS}(\boldsymbol{b})$. Clearly, if $\text{s}(\boldsymbol{c}_i) \leq \ell' - 1,\ \forall i \in \text{bS}(\boldsymbol{b})$ then $\text{s}(\boldsymbol{c}) \leq (\ell' - 1)\,\text{bs}(\boldsymbol{b})$. So, one can write

$$D_{\boldsymbol{b}} \in \text{span}\{D_{\boldsymbol{c}} \mid \boldsymbol{c} \in \mathbb{N}^n,\ \text{s}(\boldsymbol{c}) \leq (\ell' - 1)\,\text{bs}(\boldsymbol{b})\}. \tag{4.7}$$

Using Equations (4.4) and (4.7), we can write for any coefficient $D_{\boldsymbol{a}}$,

$$D_{\boldsymbol{a}} \in \text{span}\{D_{\boldsymbol{c}} \mid \boldsymbol{c} \in \mathbb{N}^n,\ \text{s}(\boldsymbol{c}) \leq (\ell' - 1)(\ell - 1)\}.$$

□

We are now ready to go to the final step of the proof - getting the actual hitting set.

4.5.1 From Concentration to Hitting Set

Let $C(\mathbf{x}) = D_0 D D_{d+1} \in \mathbb{F}[\mathbf{x}]$. Since any coefficient $\text{coeff}_C(\boldsymbol{x}^{\boldsymbol{b}})$ in C can be written as $D_0\,\text{coeff}_D(\boldsymbol{x}^{\boldsymbol{b}})D_{d+1}$, we can use Lemma 2.5.6 to get the following.

Lemma 4.5.2 (Concentration in C)**.** *Let $C(\mathbf{x}) = D_0 D D_{d+1} \in \mathbb{F}[\mathbf{x}]$ be a polynomial with $D_0 \in \mathbb{F}^{1\times w}$ and $D_{d+1} \in \mathbb{F}^{w\times 1}$. If $D(\boldsymbol{x})$ has $((\ell-1)(\ell'-1)+1)$-concentration then $C(\mathbf{x})$ has $((\ell-1)(\ell'-1)+1)$-concentration.*

Now, we come back to the proof of Theorem 4.1.1. From Lemmas 4.4.4, 4.3.4, 4.5.1 and 4.5.2, we get $\mathsf{poly}(ns^w log(w\delta))$ maps, such that for at-least one map $\phi : \mathbf{t} \to \{t_i\}_{i=0}^{\infty}$, $C'(\boldsymbol{x}) = C(\boldsymbol{x} + \phi(\mathbf{t}))$ is $(w^2\lceil \log w^2 \rceil + 1)$-concentrated. The degree of t is bounded by $\mathsf{poly}(ns^w log(w\delta))$.

By Lemma 2.5.7 we get a hitting set for $C'(\boldsymbol{x}) = C(\boldsymbol{x}+\phi(\mathbf{t}))$ of size $(n\delta)^{O(w^2 \log w)}$. Each of these evaluations of C will be a polynomial in t with degree bounded by $\mathsf{poly}(ns^w log(w\delta))$.

Hence, total time complexity becomes $\mathsf{poly}(s^w(n\delta)^{w^2 \log w})$.

For the proof of Remark 2, observe that when the D_is are univariate, D_i's are 1-concentrated and sparsity $s \leq \delta$. Thus, when the D_is are univariate, we get a $(n\delta)^{O(w^2)}$ sized hitting-set.

4.6 Width-2 Read Once ABP

In the previous section, the crucial part in finding a hitting-set for an ROABP is the assumption that the matrix product $D(\boldsymbol{x})$ is invertible. Now, we will show that for width-2 ROABP, this assumption is not required. Via a factorization property of 2×2 matrices, we will show that PIT for width-2 sparse-factor ROABP reduces to PIT for width-2 sparse-invertible-factor ROABP. This factorization of width-2 ABPs has also been studied by Allender and Wang [AW16]. But their reduction cannot maintain the sparsity of the matrix entries.

Lemma 4.6.1 (2×2 invertibility)**.** *Let $C(\mathbf{x}) = D_0 \left(\prod_{i=1}^{d} D_i\right) D_{d+1}$ be a polynomial computed by a width-2 sparse-factor ROABP. Then for some nonzero $\gamma \in \mathbb{F}[\mathbf{x}]$ and some $m \leq d$, we can write $\gamma(\mathbf{x})C(\mathbf{x}) = C_1(\mathbf{x})C_2(\mathbf{x}) \cdots C_{m+1}(\mathbf{x})$, where each of the C_is are of the form $P_iQ_iR_i$. $Q_i \in \mathbb{F}^{2\times 2}[\boldsymbol{x}]$ is a polynomial computed by a width-2 sparse-invertible-factor ROABP, $P_i \in \mathbb{F}^{1\times 2}[\boldsymbol{x}], R_i \in \mathbb{F}^{2\times 1}[\boldsymbol{x}]$, and P_i, Q_i and R_i are over disjoint sets of variables for all $i \in [m+1]$.*

Proof. Let us say, for some $i \in [d]$, $D_i(\mathbf{x}_i)$ is not invertible. Let $D_i = \left[\begin{smallmatrix} a_i & b_i \\ c_i & d_i \end{smallmatrix}\right]$ with $a_i, b_i, c_i, d_i \in \mathbb{F}[\mathbf{x}_i]$ and $a_id_i = b_ic_i$. Without loss of generality, at least one of $\{a_i, b_i, c_i, d_i\}$ is nonzero. Let us say $a_i \neq 0$ (other cases are similar). Then we can write,

$$\begin{bmatrix} a_i & b_i \\ c_i & d_i \end{bmatrix} = \frac{1}{a_i} \begin{bmatrix} a_i \\ c_i \end{bmatrix} \begin{bmatrix} a_i & b_i \end{bmatrix}.$$

In other words, we can write $\gamma_i D_i = A_iB_i$, where $A_i \in \mathbb{F}^{2\times 1}[\boldsymbol{x}_i], B_i \in \mathbb{F}^{1\times 2}[\mathbf{x}_i]$ and $0 \neq \gamma_i \in \{a_i, b_i, c_i, d_i\}$. Note that $\mathsf{s}(\gamma_i), \mathsf{s}(A_i), \mathsf{s}(B_i) \leq \mathsf{s}(D_i)$. Let us say that the set of

non-invertible D_is is $\{D_{i_1}, D_{i_2}, \ldots, D_{i_m}\}$. Writing all of them in the above form we get,

$$C(\mathbf{x}) \prod_{j=1}^{m} \gamma_{i_j} = \prod_{j=1}^{m+1} C_j,$$

where

$$C_j = \begin{cases} D_0 \left(\prod_{i=1}^{i_1-1} D_i\right) A_{i_1} & \text{if } j = 1, \\ B_{i_{j-1}} \left(\prod_{i=i_{j-1}+1}^{i_j-1} D_i\right) A_{i_j} & \text{if } 2 \leq j \leq m, \\ B_{i_m} \left(\prod_{i=i_m+1}^{d} D_i\right) D_{d+1} & \text{if } j = m+1. \end{cases}$$

Clearly, for all $j \in [m+1]$, $\left(\prod_{i=i_{j-1}+1}^{i_j-1} D_i\right)$ can be computed by a sparse-invertible-factor ROABP. The required P_j is $B_{i_{j-1}}$ and the required Q_j is A_{i_j}. Moreover, P_j, Q_j and R_j are over disjoint variables for all j. □

Now, from the above lemma it is easy to construct a hitting-set.

Consider the polynomial $C(\boldsymbol{x}) = P(\boldsymbol{x})Q(\boldsymbol{x})R(\boldsymbol{x})$, where $Q \in \mathbb{F}^{w \times w}[\boldsymbol{x}]$ is a polynomial computed by a width-w s-sparse-invertible-factor ROABP, $P \in \mathbb{F}^{1 \times w}[\boldsymbol{x}], R \in \mathbb{F}^{w \times 1}[\boldsymbol{x}]$, and P, Q and R are over disjoint sets of variables for all $i \in [m+1]$. It is similar to the polynomial described in Theorem 4.1.1, except that P and R are now polynomials over $\mathbb{F}^w$. By adapting the proof of Theorem 4.1.1, we can show a hitting set of size $\mathsf{poly}(n\delta s)^{w^2 \log w}$ in such a model. Lemma 4.4.3 can also be applied to P and R to make them $(\lceil \log w^2 \rceil + 1)$-concentrated. Since Q is (w^2+1)-block-concentrated by Lemma 4.3.4, C would be (w^2+3)-block-concentrated. The rest of the proof goes through similarly. We thus get the following lemma.

Lemma 4.6.2. *Let $C(\boldsymbol{x}) = P(\boldsymbol{x})Q(\boldsymbol{x})R(\boldsymbol{x})$, where $Q \in \mathbb{F}^{w \times w}[\boldsymbol{x}]$ is a polynomial computed by a width-w s-sparse-invertible-factor ROABP, $P \in \mathbb{F}^{1 \times w}[\boldsymbol{x}], R \in \mathbb{F}^{w \times 1}[\boldsymbol{x}]$, and P, Q and R are over disjoint sets of variables. Let the degree of each layer in Q be bounded by δ and the sparsity of each layer in Q be bounded by s. Then there is a hitting-set of size $\mathsf{poly}((n\delta s)^{w^2 \log w})$ for $C(\mathbf{x})$.*

Note that a hitting-set for $\gamma(\mathbf{x})C(\mathbf{x})$ is also a hitting-set for $C(\mathbf{x})$ if γ is a nonzero polynomial. We get a hitting-set for each of the factors by Lemma 4.6.2. Lemma 4.6.1 tells us how to factorize a width-2 ROABP into a product of width-2 invertible ROABPs.

Combining these results with Corollary 2.3.6 we get the following.

Theorem 4.6.3. *The family of polynomials of the form* $D_0(\mathbf{x}_0)\left(\prod_{i=1}^{d} D_i(\mathbf{x}_i)\right) D_{d+1}(\mathbf{x}_{d+1})$ *in* $\mathbb{F}[\mathbf{x}]$ *that are computed by a width-2 ROABP, such that for all* $1 \leq i \leq d$, D_i *has degree bounded by* δ *and sparsity* $\mathsf{s}(D_i)$ *bounded by* s, *has a hitting-set of size* $\mathsf{poly}(n\delta s)$.

4.7 Discussion

Subsequent to the development of this method, a new hitting set in time $n^{O(\log w)}$ was developed for constant-width ROABPs over fields of large characteristic ([GKS17]). It is a grey-box test. So, if the ROABP is over a field of small characteristic or if we do not know the variable sequence, we can use the hitting set presented in this chapter. Otherwise, we should use the more efficient hitting set from [GKS17].

However, it does remain a unique method for proving ℓ-concentration in an invertible ROABP. Hence, it would be worth our while to bear this result in mind in our strife towards finding a polynomial time hitting set of ROABPs.

Chapter 5

Towards Impossibility Results for the Sum of Two Width-2 Invertible ABPs

Abstract

In the other chapters, we worked with ROABPs. In this chapter, we will specialize the ABP by bounding its width to 2. We wish to give a polynomial which cannot be computed as the sum two width-2 invertible ABPs. We have not been successful in this endeavor yet. We will give some partial results.

5.1 Introduction

The basic model we will study in this chapter is the width-2 ABP. Now, the weights of the edges of the ABP are linear polynomials in $\mathbb{F}[\boldsymbol{x}]$ and the variables can be reused across the layers.

A width-2 ABP can be seen as a product of 2×2 matrices, with linear polynomials as its entries. These matrices are also called as the *layers* of the ABP. The entries of these

matrices are linear polynomials in $\mathbb{F}[\boldsymbol{x}]$. The polynomial $P(\boldsymbol{x})$ computed by the ABP is

$$P(\boldsymbol{x}) = M_1 M_2 \cdots M_d,$$

where $M_1 \in \mathbb{F}[\boldsymbol{x}]^{1\times 2}$, $M_d \in \mathbb{F}[\boldsymbol{x}]^{2\times 1}$ and $M_2, M_3, \ldots, M_{d-1} \in \mathbb{F}[\boldsymbol{x}]^{2\times 2}$.

The width-2 arithmetic branching programs and their sums are interesting because of the following result by Ben-Or and Cleve [BOC92]: Any polynomial computed by an arithmetic formula of depth-d and fan-in 2 can be computed by a width-3 ABP of size 4^d.

Consider a degree d, n-variate polynomial computed by a size s arithmetic circuit in $\mathsf{VP}_{\mathbb{F}}$[1]. We can reduce the depth of this arithmetic circuit to $O(\log d \log s)$ [VSBR83, AJMV98]. The size of this new circuit is still $\mathsf{poly}(s, n, d)$. By repeating the nodes with fan-out ≥ 2 recursively, we get a *formula* of size $s^{O(\log d \log s)}$ and depth $O(\log d \log s)$. The fan-in of each node in this formula is then reduced to 2. This increases the size and the depth of the circuit by an additional log factor. I.e. any polynomial in VP can be computed by a formula of fan-in 2 and size $\tilde{O}(s^{(\log d \log s)})$ and depth $(\log d \log s)^{O(1)}$. Now, by using the result of [BOC92], we get a size $s^{(\log d \log s)^{O(1)}}$, width-3 ABP for any polynomial in $\mathsf{VP}_{\mathbb{F}}$. In fact, given *any* polynomial, there exists a large enough width-3 ABP that can compute it.

Saha, Saptharishi and Saxena [SSS09] had proven that for any polynomial $p(\boldsymbol{x})$ computed by a depth-3 arithmetic circuit, there exists a polynomial $q(\boldsymbol{x})$ such that $p(\boldsymbol{x}) \cdot q(\boldsymbol{x})$ can be computed by a small width-2 ABP. Hence, the PIT question for depth-3 arithmetic circuits reduces to the PIT question for width-2 ABP. Gupta, Kamath, Kayal and Saptharishi proved that any polynomial in the complexity class VP can be computed by a sub-exponential size depth-3 circuit [GKKS16]. Thus, polynomial time PIT for width-2 ABP will give sub-exponential time PIT for the complexity class VP.

Impossibility results for a family of circuits $\mathcal{C}$ are polynomials which can never be computed by $\mathcal{C}$, irrespective of the size of the circuit. E.g. polynomials which cannot be factorized into linear polynomials cannot be computed by a $\prod\sum$ circuit. If there is a polynomial which cannot be computed by a particular model, then that model is said to

[1] $\mathsf{VP}_{\mathbb{F}}$ is the set of polynomial degree polynomial families over the field $\mathbb{F}$, which are computed by polynomial size arithmetic circuits.

be *incomplete*. [AW16] proved that the family of width-2 ABPs is incomplete. In contrast, the family of width-3 ABPs is complete.

The width-2 ABP model is provably incomplete. So, it has known lower bounds. PIT for width-2 ABP is provably hard. This make the width-2 ABP model quite peculiar. In a way, lower bounds and PIT are considered equivalent for general circuits [HS80, KI04, Agr05, DSY09]. The discovery of lower bounds for a model raises the hope of finding efficient PIT for it and vice versa. In many cases, these predictions come true; there are models where both lower bounds and PIT are known. E.g. diagonal circuits, ROABPs, set-multilinear circuits. But, the width-2 ABP defies these expectations.

5.1.1 Result and Proof Outline

Allender and Wang [AW16] had proven that $x_1x_2+x_3x_4+\cdots+x_{15}x_{16}$ cannot be computed by a width-2 ABP. We will see an overview of their proof in Section 5.3. We simplify Allender and Wang's invertible ABP to a *triangular ABP*.

We introduce the triangular ABP which encapsulates the power of the width-2 ABP; a lower bound for triangular ABP implies a lower bound for width-2 ABP.

A triangular ABP A has a multi-set L_A of linear forms (homogeneous linear polynomials) associated with it. The highest degree homogeneous part of the polynomial computed by such an ABP is the product of all the linear forms in the multi-set, $\prod_{l\in L_A} l$. Thus, the highest degree homogeneous parts of all polynomials computable by triangular ABPs factorize completely into linear forms.

We study the sum of two triangular ABPs, $A+B$, and ask if such a model can compute $x_1x_2 + x_3x_4 + x_5x_6$. Since a single triangular ABP cannot compute this polynomial, it means that $\deg(A) = \deg(B) = d$, say. Moreover, if $d \geq 3$, then the highest degree homogeneous parts of A and B cancel, $A_{[d]} + B_{[d]} = 0$. We also show that if the sum of two triangular ABPs computes $x_1x_2+x_3x_4+x_5x_6$, then, the dimension of the linear forms in the ABPs A and B should be 6,

$$\dim(L_A) = \dim(L_B) = 6.$$

Using these, we conclude that $d \geq 6$.

We then define a restriction $A|_S$, which is another width-2 ABP, on the ABP A with respect to a subset S of the linear forms L_A. This restriction could be seen, roughly, as the partial derivative of the circuit with respect to the complement of S. We show in Theorem 5.4.5 that:

> *Given two triangular ABPs A and B such that $A+B = x_1x_2+x_3x_4+x_5x_6$, then, for* almost all $S \subseteq L_A$, $A|_S + B|_S = 0$.

Using this result, we show that when the linear forms in the triangular ABPs A and B are 3-*wise linearly independent*, then $x_1x_2+x_3x_4+x_5x_6$ cannot be computed as $A+B$.

5.1.2 Overview of the Chapter

In Section 5.3, we take an overview of the impossibility result by Allender and Wang, which says that $x_1x_2 + x_3x_4 + \cdots + x_{15}x_{16}$ cannot be computed by a width-2 ABP. An impossibility result for triangular ABP (Definition 5.3.7) is the most important stepping stone for proving the impossibility result for general width-2 ABP. In Section 5.4, we try to prove an impossibility result for a sum of two triangular ABPs. We are able to prove such an impossibility result when the linear forms appearing in the two ABPs are 3-wise linearly independent (Definition 5.4.6). For that, we use a lower bound on the dimension of the span of linear forms appearing in the two ABPs (Lemma 5.4.1).

5.2 Preliminaries

Notations: $\mathcal{L} = \{\ell \in \mathbb{F}[\boldsymbol{x}] \mid \deg(\ell) \leq 1\}$ is the set of all linear polynomials over the field $\mathbb{F}$.

A linear form is a *homogeneous*, degree 1 polynomial. Thus, it is of the form $\sum_{i\in[n]} a_ix_i$.

In this chapter, a linear polynomial is denoted by ℓ and a linear form is denoted by l (with subscripts and superscripts, as needed).

In the drawings of ABPs, the edge with weight 0 is not drawn. The edge with weight 1 is drawn, but no weight is written on it.

We define $A_{[\delta]}$ to be the degree δ homogeneous part of the multivariate polynomial $A(\boldsymbol{x})$.

In many places in this chapter, we take an ABP A and apply operations on it. The operations are such that the polynomial computed by the ABP remains the same. We now consider this new, modified ABP and, with abuse of notation, we reuse the ABP name A.

Partial coefficient with respect to a subset of variables: Let the set of variables $\boldsymbol{y}$ be a disjoint union of its subsets $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$, $\boldsymbol{y} = \boldsymbol{z}_1 \sqcup \boldsymbol{z}_2$. Then, any polynomial $f \in \mathbb{F}[\boldsymbol{y}]$ can be viewed as a polynomial in $\mathbb{F}[\boldsymbol{z}_1][\boldsymbol{z}_2]$. I.e. f is a polynomial over the variables $\boldsymbol{z}_2$, with coefficients coming from $\mathbb{F}[\boldsymbol{z}_1]$. Let $f_{(\boldsymbol{z}_2,m)} \in \mathbb{F}[\boldsymbol{z}_1]$ denote the partial coefficient of the monomial m in the polynomial f with respect to the subset of variables $\boldsymbol{z}_2$. Thus, $f = \sum_{m \in \mathcal{M}} f_{(\boldsymbol{z}_2,m)} m$, where $\mathcal{M}$ is the set of all monomials over $\boldsymbol{z}_2$.

Observation 5.2.1. *If $f(\boldsymbol{y}) = g(\boldsymbol{y})$, then for all $\boldsymbol{z}_2 \subseteq \boldsymbol{y}$ and for all monomials m over the variables $\boldsymbol{z}_2$, $f_{(\boldsymbol{z}_2,m)} = g_{(\boldsymbol{z}_2,m)}$.*

5.2.1 Lower bounds using partial derivatives

The partial derivative method is a powerful technique for proving circuit lower bounds. Nisan [Nis91] first used it to prove that the permanent polynomial cannot be computed by a small non-commutative ABP. Nisan and Wigderson [NW96] had used the partial derivatives method to show that homogeneous depth-3 circuits cannot compute the permanent. See [Sap16] for more examples.

There are many variants of the partial derivative method. We will be using only first order partial derivatives with respect to all the variables. We define

$$\partial_i(p(\boldsymbol{x})) = \frac{\partial}{\partial x_i} p(\boldsymbol{x})$$

as the partial derivative of the polynomial $p(\boldsymbol{x})$ with respect to the variable x_i and

$$\partial_{(=1)}(p(\boldsymbol{x})) = \left\{ \frac{\partial}{\partial x_i} p(\boldsymbol{x}) \right\}_{i=1}^{n}$$

as the set of all first order partial derivatives of the polynomial $p(\boldsymbol{x})$.

In Lemma 5.4.1, we use it to show that invertible ABPs need at least 6 layers to compute the polynomial $x_1x_2 + x_3x_4 + x_5x_6$.

5.2.2 Annihilating polynomials

Definition 5.2.2 (Annihilating polynomial)**.** *Given a set $\{p_1(\boldsymbol{x}), p_2(\boldsymbol{x}), \ldots, p_k(\boldsymbol{x})\}$ of polynomials over the field $\mathbb{F}$, a polynomial $\mathcal{A}(y_1, y_2, \ldots, y_k) \neq 0 \in \mathbb{F}[y_1, y_2, \ldots, y_k]$ is called the* annihilating polynomial *of $(p_1, p_2, \ldots, p_k)$ if $\mathcal{A}(p_1, p_2, \ldots, p_k) = 0$. I.e. in the polynomial $\mathcal{A}$, when y_i is substituted with p_i for all i, the polynomial becomes 0.*

E.g. When $p_1(x) = x^2$ and $p_2(x) = x$, then, $\mathcal{A}(y_1, y_2) = y_1 - {y_2}^2$ is an annihilating polynomial for (p_1, p_2). When $p_1(x_1, x_2) = x_1$ and $p_2(x_1, x_2) = x_2$, then, no annihilating polynomial exists.

Polynomials $p_1, p_2, \ldots, p_k$ are *algebraically dependent* if an annihilating polynomial of $(p_1, p_2, \ldots, p_k)$ exists. Otherwise, they are *algebraically independent*. See Kayal's paper [Kay09] and Mittmann's PhD thesis [Mit13] for a good introduction to this topic.

The following lemma is well-known and can be proved through the Jacobian Criterion. The proof holds for fields of all characteristics. Mittmann's thesis [Mit13] alludes to this in Section 4.2.1.

Lemma 5.2.3 (For linear forms, linear and algebraic independence are equivalent)**.** *Let $\{l_1, l_2, \ldots, l_k\}$ be a set of k linear forms. Then $\{l_1, l_2, \ldots, l_k\}$ are algebraically dependent if and only if they are linearly dependent.*

5.2.3 Isomorphism between $\{f(l_1, \ldots, l_k)\}_{f \in \mathbb{F}[z_1, z_2, \ldots, z_k]}$ and $\mathbb{F}[y_1, y_2, \ldots, y_k]$

Let $P = \{p_i\}_{i=1}^{d}$ be a set of polynomials. Then $\{f(p_1, p_2, \ldots, p_k)\}_{f \in \mathbb{F}[z_1, z_2, \ldots, z_k]}$ is the set of all polynomials over $\{p_i\}_{i=1}^{d}$. Let $\psi : \{f(p_1, p_2, \ldots, p_k)\}_{f \in \mathbb{F}[z_1, z_2, \ldots, z_k]} \longrightarrow \mathbb{F}[y_1, y_2, \ldots, y_d]$, be a homomorphism map defined by $\psi : p_i \longmapsto y_i$ for all $1 \leq i \leq d$ and $a \longmapsto a$, where $a \in \mathbb{F}$. Polynomials over $\{p_i\}_{i=1}^{d}$ are mapped accordingly. I.e. $\psi(q_1q_2) = \psi(q_1)\psi(q_2)$ and $\psi(q_1 + q_2) = \psi(q_1) + \psi(q_2)$. But, first we have to check if this map well-defined. I.e. for every $p \in \{f(p_1, p_2, \ldots, p_k)\}_{f \in \mathbb{F}[z_1, z_2, \ldots, z_k]}$, is $\psi(p)$ is unique? This is not true necessarily. But when the set $\{p_i\}_{i=1}^{d}$ is special, the map is well-defined. When $\{p_i\}_{i=1}^{d}$ is a set of

linearly independent linear forms, then not only is the map ψ well-defined, but it is also an isomorphism.

The following lemma was stated in [KS07] without proof. For the sake of completeness, we provide the proof here.

Lemma 5.2.4 (Isomorphism between $\{f(l_1, l_2, \ldots, l_k)\}_{f \in \mathbb{F}[z_1, z_2, \ldots, z_k]}$ and $\mathbb{F}[y_1, y_2, \ldots, y_k]$)**.** *Let $\{l_i\}_{i=1}^{k}$ be a set of linearly independent linear forms in $\mathbb{F}[x_1, x_2, \ldots, x_n]$. Consider the homomorphism map $\psi : \{f(p_1, p_2, \ldots, p_k)\}_{f \in \mathbb{F}[z_1, z_2, \ldots, z_k]} \longrightarrow \mathbb{F}[y_1, y_2, \ldots, y_k]$, defined by $\psi : l_i \longmapsto y_i$ for all $1 \leq i \leq k$ and $a \longmapsto a$, where $a \in \mathbb{F}$. Then the map ψ is a ring isomorphism.*

Proof. We use the homomorphism $\phi : \mathbb{F}[y_1, y_2, \ldots, y_k] \longrightarrow \{f(l_1, l_2, \ldots, l_k)\}_{f \in \mathbb{F}[z_1, z_2, \ldots, z_k]}$, defined by $\phi : y_i \longmapsto l_i$ for all $1 \leq i \leq k$ and $a \longmapsto a$, where $a \in \mathbb{F}$.

We will first prove that the map ψ is well-defined. It means that we need to prove that every polynomial in $\{f(l_1, l_2, \ldots, l_k)\}_{f \in \mathbb{F}[z_1, z_2, \ldots, z_k]}$ maps to only one polynomial under ψ. Let $p \in \{f(l_1, l_2, \ldots, l_k)\}_{f \in \mathbb{F}[z_1, z_2, \ldots, z_k]}$ be a polynomial that maps to two polynomials $f, g \in \mathbb{F}[y_1, y_2, \ldots, y_k]$ under ψ. Thus, $p = \phi(f) = \phi(g)$, but $f \neq g$. Thus, $h := f - g \neq 0$, but $\phi(h) = 0$. Thus, $h(y_1, y_2, \ldots, y_k)$ is a nonzero annihilating polynomial for the set of linearly independent linear forms $\{l_1, l_2, \ldots, l_k\}$. But by Lemma 5.2.3, we know that algebraically dependent linear forms are also linearly dependent, a contradiction. Thus, the map is well-defined.

To see that the map ψ is surjective, we need to show that for every polynomial $f(y_1, y_2, \ldots, y_k)$, there exists a polynomial in $\{f(l_1, l_2, \ldots, l_k)\}_{f \in \mathbb{F}[z_1, z_2, \ldots, z_k]}$ that maps to f. Given $f(y_1, y_2, \ldots, y_k)$, $\psi(\phi(f)) = f$.

We next prove that ψ is injective. Let $f(y_1, y_2, \ldots, y_k) = 0$. Then, $\phi(f)$, which is obtained by substituting l_i for every y_i is 0. □

5.2.4 Operations on an ABP

One polynomial can be computed by many ABPs. E.g. both the ABPs $\left[\begin{smallmatrix}1 & 2\end{smallmatrix}\right]\left[\begin{smallmatrix}x_1 & x_2\\ x_3 & x_4\end{smallmatrix}\right]\left[\begin{smallmatrix}1\\ 2\end{smallmatrix}\right]$ and $\left[\begin{smallmatrix}2 & 1\end{smallmatrix}\right]\left[\begin{smallmatrix}x_3 & x_4\\ x_1 & x_2\end{smallmatrix}\right]\left[\begin{smallmatrix}1\\ 2\end{smallmatrix}\right]$ compute the polynomial $x_1 + 2x_2 + 2x_3 + 4x_4$. We will introduce a few operations on an ABP, such that the resulting ABP computes the same polynomial.

Our operations are obtained by the introduction of NN^{-1} between two layers, where N is a 'valid' invertible 2×2 matrix.

Let us see an example below, which we will use frequently.

Lemma 5.2.5. *Let $A = M_1 M_2 \cdots M_d$ be a width-2 ABP. Let $1 \leq i \leq d-1$. Let N be an invertible layer, i.e. $\det(N) \in \mathbb{F} \setminus \{0\}$. Then, $A' = M_1 M_2 \cdots M_i N N^{-1} M_{i+1} \cdots M_d$ computes the same polynomial as A.*

Also, $\det(M_i N) = 0$ if and only if $\det(M_i) = 0$ and $\det(N^{-1} M_{i+1}) = 0$ if and only if $\det(M_{i+1}) = 0$.

Definition 5.2.6 ($\mathrm{Mul}(A, i, N)$)**.** *The operation $\mathrm{Mul}(A, i, N)$ returns the ABP A' from the above lemma.*

5.3 Width-2 ABP

We will now study the impossibility result given by Allender and Wang [AW16] in some detail. They show that the polynomial $x_1x_2 + x_3x_4 + \cdots + x_{15}x_{16}$ cannot be computed by a width-2 ABP. First, they classify the matrices occurring in the ABP according to their determinants.

Definition 5.3.1 (Classification of 2×2 matrices, [AW16])**.** *A matrix $M \in \mathbb{F}[\boldsymbol{x}]^{2\times 2}$ is classified according to the value of its determinant, $|M|$. If $|M| = 0$, then it is called a* degenerate layer*. If the determinant of the matrix M is a nonzero constant, i.e. $0 \neq |M| \in \mathbb{F}$, then it is called an* invertible layer*. Otherwise, the determinant of the matrix M is a non-constant polynomial, i.e. $|M| \in \mathbb{F}[\boldsymbol{x}] \setminus \mathbb{F}$. In that case, M is called as a* potentially degenerate layer.

To show that $x_1x_2 + \cdots + x_{15}x_{16}$ cannot be computed by an ABP, it is enough to prove that $x_1x_2 + x_3x_4 + \ell$ cannot be computed by an ABP with only invertible layers, for any linear polynomial ℓ. This follows from the the first two points of the following Theorem.

Theorem 5.3.2 (Allender and Wang [AW16])**.** *1. If there exists a width-2 ABP with degenerate layers that computes $x_1x_2 + \cdots + x_{15}x_{16}$, then there exists a width-2 ABP without degenerate layers that computes $x_1x_2 + \cdots + x_{15}x_{16}$.*

2. *If there exists a width-2 ABP with potentially degenerate layers that computes* $x_1x_2 + \cdots + x_{15}x_{16}$*, then there exists a width-2 ABP with only invertible layers that computes* $x_1x_2 + x_3x_4 + \ell$*, for some linear polynomial* ℓ*.*
3. *A width-2 ABP with only invertible layers cannot compute* $x_1x_2 + x_3x_4 + \ell$ *for any linear polynomial* ℓ*.*

For this thesis, we wish to bring the invertible width-2 ABP into a canonical form. We start with the form of the invertible ABP, as given by Allender and Wang [AW16] and modify it into a simpler form.

Lemma 5.3.3 (Allender and Wang [AW16])**.** *Every invertible ABP can be transformed so that its layers* $M_2, M_3, \ldots, M_{d-1}$ *are only of the form* $\left[\begin{smallmatrix} a & 0 \\ \ell & d \end{smallmatrix}\right]$ *or* $\left[\begin{smallmatrix} a & \ell \\ 0 & d \end{smallmatrix}\right]$ *or* $\left[\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right]$*, where* ℓ *is a nonzero linear polynomial in* $\mathbb{F}[\boldsymbol{x}]$*,* $a, b, c, d \in \mathbb{F}$ *with* $a, d \neq 0$*,* b *and* c *are not both* 0 *and the layers are invertible.*

The first layer, M_1 *is of the form* $\left[\begin{smallmatrix} a & \ell \end{smallmatrix}\right]$ *and the last layer* M_d *is of the form* $\left[\begin{smallmatrix} b \\ \ell \end{smallmatrix}\right]$ *or* $\left[\begin{smallmatrix} \ell \\ b \end{smallmatrix}\right]$*.*

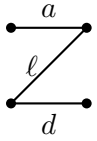
Definition 5.3.4 (AW-ABP)**.** *We call an invertible width-2 ABP of the above form as an AW-ABP.*

In this thesis, we will call these matrix forms as Z, mirror-Z *and* X respectively. The name comes from the appearance of these layers in the graphical form:

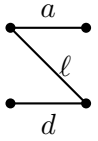
Definition 5.3.5 (Z, mirror-Z and X layers)**.**

$$\begin{aligned} Z &= \left\{ \left[\begin{smallmatrix} a & 0 \\ \ell & d \end{smallmatrix}\right] \mid a, d \in \mathbb{F} \setminus \{0\}, \ell \in \mathcal{L} \setminus \{0\} \right\}, \\ \text{mirror-Z} &= \left\{ \left[\begin{smallmatrix} a & \ell \\ 0 & d \end{smallmatrix}\right] \mid a, d \in \mathbb{F} \setminus \{0\}, \ell \in \mathcal{L} \setminus \{0\} \right\}, \\ X &= \left\{ \left[\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right] \mid a, d \neq 0, (b, c) \neq (0, 0), ad - bc \neq 0 \right\}. \end{aligned}$$

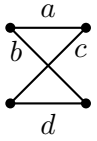
The Z layer, $\left[\begin{smallmatrix} a & 0 \\ \ell & d \end{smallmatrix}\right]$



The mirror-Z layer, $\left[\begin{smallmatrix} a & \ell \\ 0 & d \end{smallmatrix}\right]$



The X layer, $\left[\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right]$



Observe that $Z \cap X \neq \emptyset$. Similarly, mirror-Z $\cap X \neq \emptyset$. But, $Z \cap$ mirror-Z $= \emptyset$.

Observation 5.3.6 ([AW16])**.** *Without loss of generality, an AW-ABP contains no two consecutive Z layers (*mirror-Z *and X layers respectively).*

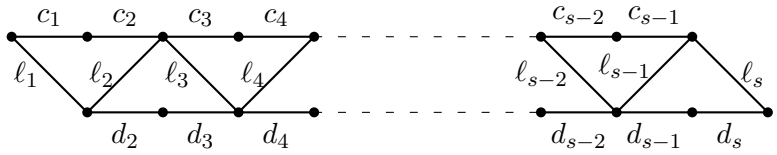
That is because two consecutive Z layers (mirror-Z and X layers respectively) can be multiplied together to give another Z layer (mirror-Z and X layer respectively). Since, the determinants of the above matrices are nonzero constants, the determinant of the product matrix is also a nonzero constant. When we multiply two X layers, it means that $a, d \neq 0$ or $b, c \neq 0$. This ABP can be modified, so that without loss of generality, $a, d \neq 0$. But, (b, c) may be $(0, 0)$. In the case of the product of two Z (mirror-Z respectively) layers, since the determinant of the product matrix is a nonzero constant, in the product matrix, $a, d \neq 0$. But, ℓ may now become 0. In such cases, the layer $\left[\begin{smallmatrix} a & 0 \\ 0 & d \end{smallmatrix}\right]$ can be multiplied into the consecutive layer.

5.3.1 Canonical form of a width-2 invertible ABP: Triangular ABP

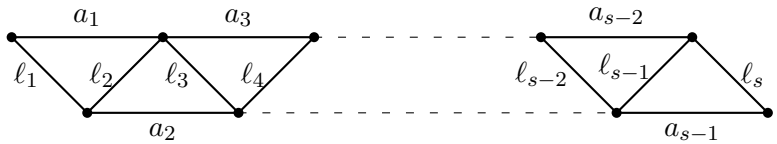
We will now show that the X matrix is not required. I.e. any polynomial which can be computed by an invertible ABP can also be computed by an invertible ABP that consists of only the Z and mirror-Z layers. Let us first see what such an ABP would look like.

Definition 5.3.7 (Triangular ABP)**.** *A* width-2 triangular ABP *contains $Z \setminus X$ and* mirror-Z$\setminus X$ *layers alternately.*

Thus, the linear polynomials in the Z and mirror-Z layers are not constants. We use this name because of the graphical appearance of such an ABP:



Since every path which goes through c_i has to go through c_{i+1} (for an odd i), we can replace the consecutive $c_i c_{i+1}$ with a constant a_i. Similarly, we can replace the consecutive $d_i d_{i+1}$ (for an even i) with a constant a_i.



A triangular ABP has a simple representation as a graph, as can be seen in the diagram above. It is represented as an iterated matrix product by assuming that the Z layers are of the form $\left[\begin{smallmatrix} 1 & 0 \\ \ell & a \end{smallmatrix}\right]$ and mirror-Z layers are of the form $\left[\begin{smallmatrix} a & \ell \\ 0 & 1 \end{smallmatrix}\right]$.

In Lemma 5.3.11, we will see how to get rid of the X layer. Before that, we need to define a few operations on the AW-ABP which will help us simplify the AW-ABP.

Definition 5.3.8 (Unitize$_i(A)$)**.** *This operation takes an AW-ABP $A = M_1 M_2 \cdots M_d$. If $M_1 = [\, a \;\; \ell \,]$, then* $\text{Unitize}_1(A) = \text{Mul}\left(A, 1, \left[\begin{smallmatrix} \frac{1}{a} & 0 \\ 0 & 1 \end{smallmatrix}\right]\right)$*, so that the first layer is of the form $[\, 1 \;\; \ell \,]$.*

For $2 \leq i \leq d-1$,

- *If $M_i = \left[\begin{smallmatrix} c & 0 \\ \ell & d \end{smallmatrix}\right] \in Z$,* $\text{Unitize}_i(A) = \text{Mul}\left(A, i, \left[\begin{smallmatrix} \frac{1}{c} & 0 \\ 0 & \frac{1}{d} \end{smallmatrix}\right]\right)$*, so that the ith layer is of the form $\left[\begin{smallmatrix} 1 & 0 \\ \ell' & 1 \end{smallmatrix}\right]$.*
- *If $M_i = \left[\begin{smallmatrix} c & \ell \\ 0 & d \end{smallmatrix}\right] \in$* mirror-Z, $\text{Unitize}_i(A) = \text{Mul}\left(A, i, \left[\begin{smallmatrix} \frac{1}{c} & 0 \\ 0 & \frac{1}{d} \end{smallmatrix}\right]\right)$*, so that the ith layer is of the form $\left[\begin{smallmatrix} 1 & \ell' \\ 0 & 1 \end{smallmatrix}\right]$.*
- *If $M_i = \left[\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right] \in X \setminus Z$, i.e. if $a, b \neq 0$,* $\text{Unitize}_i(A) = \text{Mul}\left(A, i, \left[\begin{smallmatrix} \frac{1}{a} & 0 \\ 0 & \frac{1}{b} \end{smallmatrix}\right]\right)$*, so that the ith layer is of the form $\left[\begin{smallmatrix} 1 & 1 \\ c' & d' \end{smallmatrix}\right]$.*

Lemma 5.3.9 (Introducing a swap layer)**.** *Let $A = M_1 M_2 \cdots M_d$ be an AW-ABP that computes the polynomial $p(\boldsymbol{x})$. Define the* swap layer $S = \left[\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right]$*. Then, for every $1 \leq i < d$, there is an ABP $A' = M_1 M_2 \cdots M_i S M'_{i+1} \cdots M'_d$ that computes the same polynomial $p(\boldsymbol{x})$.*

Moreover, for all j such that $i+1 \leq j \leq d-1$,

- *if M_j is a Z layer, then M'_j is a* mirror-Z *layer,*
- *if M_j is a* mirror-Z *layer, then M'_j is a Z layer and*
- *if M_j is an X layer, then M'_j is also an X layer.*

Proof. First, we observe that $S^{-1} = S$. We then introduce $SS^{-1} = SS$ between M_j and M_{j+1} for all $i \leq j < d$. Define $M'_j = SM_jS$ for $i < j < d$. Now, if M_j is a Z layer, then SM_jS is a mirror-Z layer. If M_j is a mirror-Z layer, then SM_jS is a Z layer and if M_j is an X layer, then SM_jS is also an X layer. □

So, in the graphical representation of the ABPs, the layer M'_j is the 'reflection' of the layer M_j for all j such that $i+1 \leq j \leq d$.

Definition 5.3.10 ($\text{Swap}_i(A)$)**.** *The operation* $\text{Swap}_i(A)$ *returns the ABP* A' *from the above lemma.*

In the proof of Lemma 5.3.11, we would be multiplying the layers M_i and S, obtained after applying a Swap operation, into a single layer of some desirable form, so that the polynomial computed is again an AW-ABP. Observe that the operations $\text{Unitize}_i(A)$ and $\text{Swap}_i(A)$ affect only the ith and later layers. Hence, we can start from the leftmost layer M_1 of the ABP and bring it to some desirable form. We can then do the same to the next layer, and so on.

Let us now see how to convert an AW-ABP into a triangular ABP.

Lemma 5.3.11. *Every polynomial computed by a width-*2 *invertible ABP can be computed by a triangular width-*2 *ABP.*

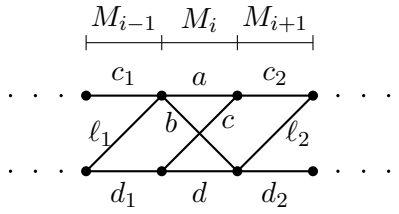
Proof. Let $A = M_1M_2\cdots M_d$ be the width-2 AW-ABP, where M_1 is a 1×2 matrix, $M_2, M_3\ldots, M_{d-1}$ are 2×2 matrices and M_d is a 2×1 matrix. We will modify this ABP, such that the polynomial computed by the ABP remains the same. So, we have to get rid of all the X layers. Let $M_i := \left[\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right]$ be the first 2×2 layer of the form X.

We can assume without loss of generality that M_{i-1} is a Z layer. (If not, then, M_{i-1} has to be a mirror-Z layer. Then, we consider $\text{Swap}_1(A)$ as our ABP A). So, $M_{i-1}\in Z\backslash X$.

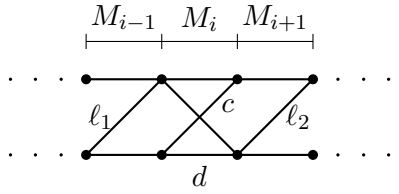
If $i=2$, we can assume that M_1 is of the form $[\,\ell\ a\,]$ without loss of generality (ℓ is not a constant). We can also assume that $a,b,d\neq 0$. I.e. $M_i\notin Z$. That is because, if M_i were a Z layer, then, we could have multiplied M_i into M_{i-1} (by Observation 5.3.6).

Now, $M_{i+1}\notin X$ by Observation 5.3.6. We now claim that ABP A can be modified so that $M_{i+1}\in Z\setminus X$. To prove that, we consider the following two cases: the first case is if $M_i\in$ mirror-Z$\cap X$, i.e. if $c=0$, then, M_{i+1} anyway should be a Z layer by Observation 5.3.6. The second case is when $M_i\in X\setminus\{Z\cup\text{mirror-Z}\}$. I.e. $a,b,c,d\neq 0$. The constants a and d are nonzero by the definition of the X layer. Then, consider $\text{Swap}_i(A) = M_1M_2\cdots M_{i-1}M_iSM'_{i+1}\cdots M'_d$. Define $M'_i = M_iS = \left[\begin{smallmatrix} b & a \\ d & c \end{smallmatrix}\right]$, which is an X layer. Then $A' = M_1M_2\cdots M_{i-1}M'_iM'_{i+1}\cdots M'_d$ is an ABP in the required form with $M'_{i+1}\in Z\setminus X$ by Lemma 5.3.9. Since we will only work with this new ABP, let us call this the ABP A. We also reuse the matrix names and the edge weight names.

Thus, without loss of generality, the scenario is that $M_{i-1} \in Z \setminus X$, $M_i := \left[\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right] \in X$ with $a, b, d \neq 0$ and $M_{i+1} \in Z \setminus X$.

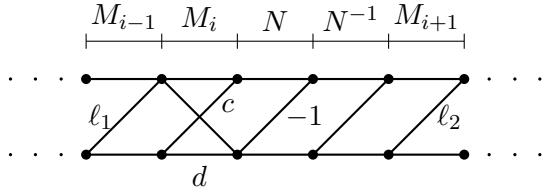


We further simplify the ABP for our purpose. By applying the operations $A = \text{Unitize}_{i-1}(A)$, $A = \text{Unitize}_i(A)$ and $A = \text{Unitize}_{i+1}(A)$, we can assume that $c_1 = d_1 = a = b = c_2 = d_2 = 1$. Pre-multiplying a Z matrix or an X matrix with a diagonal matrix yields a Z matrix or an X matrix, respectively. This operation will change the values on the other edges. E.g. ℓ_1 now becomes ℓ_1/c_1. But we reuse the names $\ell_1, \ell_2, c, d, M_{i-1}, M_i, M_{i+1}, \ldots$.

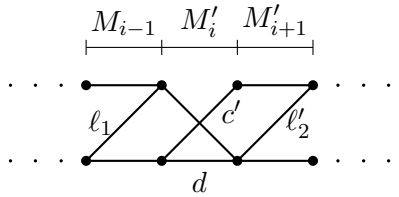


Observe that $c \neq d$, because, otherwise, $\det(M_i) = 0$. But by Lemma 5.2.5, that is not possible.

Let $N := \left[\begin{smallmatrix} 1 & 0 \\ -1 & 1 \end{smallmatrix}\right]$. Thus, $N^{-1} = \left[\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}\right]$. Let $A = \text{Mul}(A, i, N)$.



N^{-1} is a Z layer (like M_{i+1}) and can be multiplied into M_{i+1}. Let $M'_{i+1} = N^{-1}M_{i+1}$. Let $M'_i = M_iN = \left[\begin{smallmatrix} 0 & 1 \\ c-d & d \end{smallmatrix}\right]$. Let $A' = M_1M_2 \cdots M_{i-1}M'_iM'_{i+1}M_{i+2} \cdots M_d$.

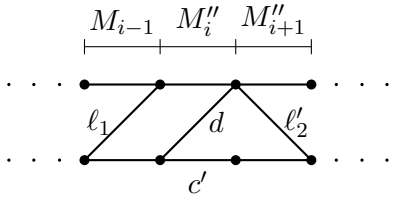


Here, $c' = c - d \neq 0, d \neq 0$ and $\ell'_2 = \ell_2 + 1$.

Recall that $M_{i+1} \in Z \setminus X$. I.e. ℓ_2 is not a constant. Hence, $M'_{i+1} \in Z \setminus X$.

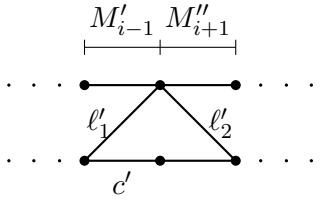
Now, M'_i is not an AW-ABP (Lemma 5.3.3). We introduce a swap layer between M'_i and M'_{i+1}, Then, $\text{Swap}_i(A') = M_1M_2 \cdots M_{i-1}M'_iSM''_{i+1} \cdots M'_d$. Note that $M'_{i+2}, \ldots, M'_d$

are valid AW-ABP layers.



Observe that $M_i'' = M_i'S = \left[\begin{smallmatrix} 0 & 1 \\ c-d & d \end{smallmatrix}\right]\left[\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right] = \left[\begin{smallmatrix} 1 & 0 \\ d & c-d \end{smallmatrix}\right] \in Z$.

Since $M_{i-1} \in Z \setminus X$ and $M_i'' \in Z$, we can multiply them together to get matrix $M_{i-1}' \in Z \setminus X$.



Here, $\ell_1' = \ell_1 + d$ is not a constant.

We thus got rid of M_i, the first X layer in the AW-ABP, by modifying the layers to the right of this occurrence. We modified the previous layer M_{i-1}, but ensured that it remains of the form $Z \setminus X$. So, we got an AW-ABP computing the same polynomial such that the first i layers are not of the form X. We repeat this process until we lose all the X layers. □

There are many equivalent manipulations which give us the above result.

Properties of a triangular ABP

Thus, every polynomial computed by an invertible ABP is computable by a triangular ABP. The *diagonal edge weights* are of the form $\ell = \ell_{[1]} + \ell_{[0]}$, where $\ell_{[1]}$ is a linear form and $\ell_{[0]}$ is a field constant.

Definition 5.3.12 (L)**.** *Let L be the* multi-set *of all the* linear forms *appearing as weights in the ABP.*

Thus, $|L| = d$ is the degree of the polynomial computed by the triangular ABP.

Lemma 5.3.13 (Polynomial computed by a width-2, invertible ABP)**.** *The polynomial A*

computed by an invertible ABP is of the form

$$A = \sum_{S \subseteq L} c_S \prod_{l \in S} l,$$

and the degree δ homogeneous part of the polynomial A, $A_{[\delta]}$ is also of a similar form

$$A_{[\delta]} = \sum_{S \subseteq L, |S| = \delta} c_S \prod_{l \in S} l,$$

where, $c_S \in \mathbb{F}$ is a constant.

Thus the polynomials $A, A_{[\delta]} \in \{f(l_1, l_2, \ldots, l_d)\}_{f \in \mathbb{F}[z_1, z_2, \ldots, z_d]}$.

From the triangular representation of the invertible ABP, it is easy to see that the coefficient corresponding to the highest degree homogeneous form, $c_L = 1$. Thus,

Lemma 5.3.14. *The highest degree homogeneous part of the polynomial computed by an invertible ABP factorizes into linear forms.*

Thus, any polynomial whose highest degree part does not factorize completely cannot be computed by an invertible width-2 ABP. In particular, $x_1x_2 + x_3x_4$ cannot be computed by such a model.

5.4 Sum of Two Triangular ABPs

Now, let us turn our attention to the question that this chapter attempts to answer. The question is can a sum of two invertible ABPs compute $x_1x_2 + x_3x_4 + x_5x_6$?

5.4.1 Dimension of the linear forms

We first show that the set of linear forms appearing in the two ABPs has full rank.

Lemma 5.4.1 (Lower bound on the dimension of the linear forms)**.** *Let $p = x_1x_2 + x_3x_4 + x_5x_6$ be a polynomial computed as the sum $A + B$ of two triangular width-2 ABPs, A and B. Let L_A be the multi-set of all the linear forms appearing as the diagonal weights in the triangular ABP A. Let L_B be the multi-set of all the linear forms appearing as the diagonal weights in the triangular ABP B. Then,* $\dim\{L_A \cup L_B\} = 6$.

Proof. It is easy to see that the dimension of the set of all first order partial derivatives of the polynomial $p = x_1x_2 + x_3x_4 + x_5x_6$ is 6.

$$\dim\left(\partial_{(=1)}(p)\right) = 6. \tag{5.1}$$

Let $L = L_A \cup L_B$ be the union of the two sets of linear forms, L_A and L_B. We now prove that for any degree 2 polynomial p computed as the sum of two triangular width-2 ABPs, $\dim\left(\partial_{(=1)}(p)\right) \leq \dim L$. Since p is the sum of two triangular ABPs, whose degree 2 homogeneous parts are given by Lemma 5.3.13,

$$p_{[2]} = \sum_{l_1,l_2 \in L} c_{\{l_1,l_2\}} l_1 l_2,$$

where $c_{\{l_1,l_2\}}$s are field constants. Thus, the partial derivative with respect to a variable is some linear combination of the linear forms in L. Hence, the dimension of the set of all order 1 partial derivatives is upper bounded by the dimension of the set L.

$$\dim\left(\partial_{(=1)}(p)\right) \leq \dim L. \tag{5.2}$$

Thus, from Equations (5.2) and (5.1), if the polynomial p is computed as a sum of triangular ABPs, then the dimension of its linear forms should be ≥ 6. But, there are only 6 variables. Hence, the dimension of L is 6. □

5.4.2 Comparison of linear forms in the two ABPs

Let the polynomials computed by the two ABPs be p and q respectively. Then, it can happen that their highest degree homogeneous parts completely factorize into linear polynomials. However, when added together, we get $x_1x_2 + x_3x_4 + x_5x_6$; the homogeneous parts of the polynomials p and q with degree ≥ 3 cancel each other.

If there exist two triangular ABPs A and B such that $A+B = x_1x_2+x_3x_4+x_5x_6$, then, there also exist two triangular ABPs A' and B' such that $A' - B' = x_1x_2 + x_3x_4 + x_5x_6$. Henceforth, we will consider the problem of the existence of ABPs A and B such that $A - B = x_1x_2 + x_3x_4 + x_5x_6$.

Let us look at a few easy properties of this model. Let A and B be two width-2,

triangular ABPs.

If $A - B = x_1x_2 + x_3x_4 + x_5x_6$, and $\deg(A) \geq 3$, then we can see that

$$A_{[\delta]} = B_{[\delta]}, \text{ for all } \delta \geq 3. \tag{5.3}$$

Let L_A be the multi-set of the *linear forms* occurring in the ABP A and L_B be the multi-set of the linear forms occurring in the ABP B. Then, the multi-sets L_A and L_B of the linear forms occurring in ABP A and in ABP B have to be the same.

Lemma 5.4.2. *Let A and B be two triangular ABPs are such that $A - B = x_1x_2 + x_3x_4 + x_5x_6$. If $\deg(A) \geq 3$, then there exists another triangular ABP B' such that $A - B' = x_1x_2 + x_3x_4 + x_5x_6$ and $L_A = L_{B'}$.*

Proof. Since $A_{[\delta]} = B_{[\delta]}$ for all $\delta \geq 3$, and since $\deg(A) \geq 3$, $\deg(B) = \deg(A)$. Let the degrees of the two polynomials computed by the triangular ABPs, A and B be d, i.e. $\deg(A) = \deg(B) = d$. Let the linear forms in the ABPs A and B appear in the order $(l_1, l_2, \ldots, l_d)$ and $(l'_1, l'_2, \ldots, l'_d)$ respectively. I.e. L_A is the multi-set $\{l_1, l_2, \ldots, l_d\}$ and L_B is the multi-set $\{l'_1, l'_2, \ldots, l'_d\}$. By Equation 5.3, the highest degree homogeneous parts of the ABPs A and B cancel, $A_{[d]} = B_{[d]}$. But, $A_{[d]} = \prod_{i\in[d]} l_i$ and $B_{[d]} = \prod_{i\in[d]} l'_i$. Thus, $\prod_{i\in[d]} l_i = \prod_{i\in[d]} l'_i$. Since $\mathbb{F}[\boldsymbol{x}]$ is a unique factorization domain, there exists a permutation π on $[d]$ such that for every i, $l'_i = c_i l_{\pi(i)}$, where $c_i \in \mathbb{F} \setminus \{0\}$ and $\prod_{i\in[d]} c_i = 1$.

We apply the operations $B = \text{Mul}(B, i, N_i)$, for appropriate N_is, in the order $i = 1$ to $(d-1)$ to get that $L_A = L_{B'}$.

□

We will rename the ABP B' from the above lemma as B and use it henceforth.

Using that $\dim\{L_A \cup L_B\} = 6$ (Lemma 5.4.1) and if $\deg(A) \geq 3$ or $\deg(B) \geq 3$, then $\deg(A) = \deg(B) = d$ and $L_A = L_B$ (Lemma 5.4.2), we can say that $|L_A| = |L_B| = d \geq 6$.

5.4.3 Restricting the ABP

Now, let k be the rank of the multi-set of the linear forms, L. Define $\boldsymbol{y} = \{y_1, y_2, \ldots, y_k\}$. Then, by Lemma 5.2.3, $\{f(l_1, l_2, \ldots, l_d)\}_{f\in\mathbb{F}[z_1,z_2,\ldots,z_d]}$ is isomorphic to $\mathbb{F}[\boldsymbol{y}]$. There are at least $k!$ such isomorphisms from $\{f(l_1, l_2, \ldots, l_d)\}_{f\in\mathbb{F}[z_1,z_2,\ldots,z_d]}$ to $\mathbb{F}[\boldsymbol{y}]$.

Definition 5.4.3 ($\psi_{\mathcal{B}}$)**.** *Let $\mathcal{B} = (l_{j_1}, l_{j_2}, \ldots, l_{j_k})$ be an ordered basis*[2] *of the set of linear forms, L. Then, let $\psi_{\mathcal{B}} : \{f(l_1, l_2, \ldots, l_d)\}_{f \in \mathbb{F}[z_1, z_2, \ldots, z_d]} \longrightarrow \mathbb{F}[\boldsymbol{y}]$ be the ring isomorphism defined by $\psi_{\mathcal{B}} : l_{j_i} \longmapsto y_i$ and $\psi_{\mathcal{B}} : a \longmapsto a$.*

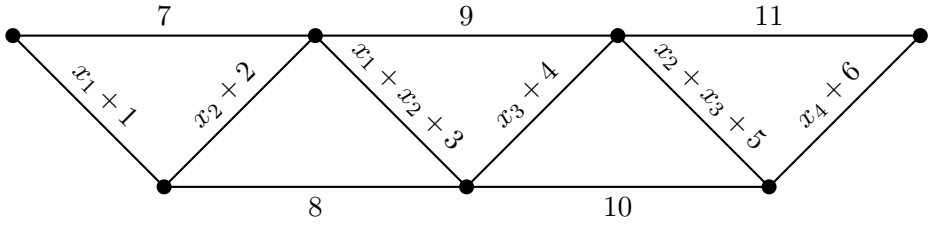
Definition 5.4.4 ($A|_S$, Restriction of ABP A to a subspace of linear forms)**.** *Let $A = M_1 M_2 \cdots M_d$ be a triangular ABP. Let $S \subseteq L$ be a subset of the linear forms occurring in the ABP A. Then, for each layer M, we perform the following operation. Let $l(M)$ be the linear form occurring in the layer M. If $l(M) \in \operatorname{span}\{S\}$, then define $M|_S = M$. Else, the layer $M|_S$ is obtained from the layer M by replacing its linear form $l(M)$ with 1 and the other constants in the layer M (the $(1,1)$th entry, the $(2,2)$th entry and the constant part from the linear polynomial, ℓ) with 0.*

Then, $A|_S$ is obtained by replacing each M_i with $M_i|_S$.

This operation on the triangular ABP A is like taking a partial derivative of the circuit with respect to $L \setminus \operatorname{span}\{S\}$. It does not preserve the computed polynomial.

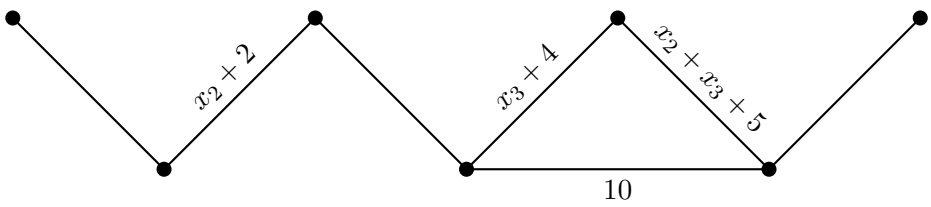
A picture is worth a thousand words. We elaborate the definition with the following example.

ABP A:



Then,

ABP $A|_{\{x_2,x_3\}}$:



The following is the main theorem of this chapter. We show that if $A - B = x_1x_2 + x_3x_4 + x_5x_6$, then, the two polynomials, under various restrictions, are equal.

[2] A basis with some order defined on it, denoted by round brace.

Theorem 5.4.5. *Let $S \subseteq L$ be a subset of the multi-set of the linear forms appearing in the ABPs A and B. If $|L \setminus \text{span}\{S\}| \geq 3$ and $A - B = x_1x_2 + x_3x_4 + x_5x_6$, then $A|_S = B|_S$.*

Proof. Let $p = A|_S$ and $q = B|_S$. We will show that $p_{[i]} = q_{[i]}$ for all i. Fix any i.

Since the higher degree parts of the two polynomials computed by the two ABPs cancel, $A_{[\delta]} = B_{[\delta]}$ for all $\delta \geq 3$.

We choose $\delta = i + |L \setminus \text{span}\{S\}|$.

Recall that for a polynomial $f(\boldsymbol{y}) \in \mathbb{F}[\boldsymbol{y}]$, when the set of variables, $\boldsymbol{y} = \boldsymbol{z}_1 \sqcup \boldsymbol{z}_2$, is the disjoint union of $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$, then, $f_{(\boldsymbol{z}_2,m)}$ is the partial coefficient of the monomial m with respect to the variables $\boldsymbol{z}_2$ in the polynomial f. We will show that there exists a basis $\mathcal{B}$, a subset $\boldsymbol{z}_2$ of the new variables $\boldsymbol{y}$ under $\psi_{\mathcal{B}}$ and a monomial m in $\boldsymbol{z}_2$ such that:

$$\psi_{\mathcal{B}}{}^{-1}\left(\left(\psi_{\mathcal{B}}\left(A_{[\delta]}\right)\right)_{(\boldsymbol{z}_2,m)}\right) = c \cdot p_{[i]} \tag{5.4}$$

and

$$\psi_{\mathcal{B}}{}^{-1}\left(\left(\psi_{\mathcal{B}}\left(B_{[\delta]}\right)\right)_{(\boldsymbol{z}_2,m)}\right) = c \cdot q_{[i]}, \tag{5.5}$$

where c is a nonzero constant. Since $A_{[\delta]} = B_{[\delta]}$, and $\psi_{\mathcal{B}}^{-1}$ and $\psi_{\mathcal{B}}$ are isomorphisms, from the above equations, we get that $p_{[i]} = q_{[i]}$.

Let $\mathcal{B}_1$ be an ordered basis of S. And $\mathcal{B}_2$ be a basis of $L \setminus \text{span}\{S\}$. Then, $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ is an ordered basis of L.

Define $\boldsymbol{z}_1 = \{\psi_{\mathcal{B}}(l)\}_{l \in \mathcal{B}_1}$ and $\boldsymbol{z}_2 = \{\psi_{\mathcal{B}}(l)\}_{l \in \mathcal{B}_2}$. Thus, $\boldsymbol{y}$ is the *disjoint* union of $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$. Now,

$$A_{[\delta]} = \left(p_{[i]} \cdot \prod_{l \in L \setminus \text{span}\{S\}} l\right) + \text{Terms with smaller degree in } L \setminus \text{span}\{S\}.$$

We will apply $\psi_{\mathcal{B}}$ on both sides. Then,

$$\psi_{\mathcal{B}}\left(A_{[\delta]}\right) = \left(\psi_{\mathcal{B}}\left(p_{[i]}\right) \cdot \prod_{l \in L \setminus \text{span}\{S\}} \psi_{\mathcal{B}}\left(l\right)\right) + \text{Terms with smaller degree in } \boldsymbol{z}_2.$$

Here, $\psi_{\mathcal{B}}\left(p_{[i]}\right) \in \mathbb{F}[\boldsymbol{z}_1]$. If $l \in L \setminus \text{span}\{S\}$, then $\psi_{\mathcal{B}}(l)(\mod \boldsymbol{z}_1)$ is a nonzero linear form over $\boldsymbol{z}_2$.

Now, we pick a monomial $m \in \mathbb{F}[\boldsymbol{z}_2]$ such that its coefficient in $\psi_{\mathcal{B}}\left(A_{[\delta]}\right)$ is a constant

multiple of $p_{[i]}$. We pick any monomial with nonzero coefficient from $\prod_{l \in L \setminus \text{span}\{S\}} \psi_{\mathcal{B}}(l)$. Then, $\left(\psi_{\mathcal{B}}\left(A_{[\delta]}\right)\right)_{(\boldsymbol{z}_2,m)} = \text{coeff}_m\left(\prod_{l \in L \setminus \text{span}\{S\}} \psi_{\mathcal{B}}(l)\right) \cdot \psi_{\mathcal{B}}\left(p_{[i]}\right)$. Similarly, we get that $\left(\psi_{\mathcal{B}}\left(B_{[\delta]}\right)\right)_{(\boldsymbol{z}_2,m)} = \text{coeff}_m\left(\prod_{l \in L \setminus \text{span}\{S\}} \psi_{\mathcal{B}}(l)\right) \cdot \psi_{\mathcal{B}}\left(q_{[i]}\right)$.

By setting $c = \text{coeff}_m\left(\prod_{l \in L \setminus \text{span}\{S\}} \psi_{\mathcal{B}}(l)\right)$, we get Equations 5.4 and 5.5. □

5.4.4 3-wise independent linear forms

We will now see an application of the above lemma.

Let the multi-set of linear forms occurring in each ABP be $L_A = L_B = L = \{l_i\}_{i=1}^{d}$. Let the multi-set of linear *polynomials* occurring on the ABPs A and B be $\{l_1 + c_1, l_2 + c_2, \ldots, l_d + c_d\}$ and $\{l_1 + c'_1, l_2 + c'_2, \ldots, l_d + c'_d\}$ respectively. We will next show that, under some conditions, $c_i = c'_i$ for all $1 \leq i \leq d$.

Definition 5.4.6 (3-wise independent linear forms)**.** *A multi-set of linear forms L is 3-independent if every subset $S \subseteq L$ of size 3 is linearly independent.*

Thus, for all subsets $S = \{l_i, l_j, l_k\} \subseteq L$ of size 3, $c_i l_i + c_j l_j + c_k l_k = 0$ implies $c_i = c_j = c_k = 0$.

Lemma 5.4.7. *If $A - B = x_1x_2 + x_3x_4 + x_5x_6$, and if the linear forms in L are 3-independent, then, $c_i = c'_i$ for all $1 \leq i \leq d$.*

Proof. Since L is 3-independent, it is also 2-independent and hence, each linear form occurs only once in each ABP. Let the set $S_i = \{l_i\}$. Then, $A|_{S_i} = l_i + c_i$ and $B|_{S_i} = l_i + c'_i$. By Lemma 5.4.1, $|L \setminus \text{span}\{S_i\}| = 5 \geq 3$. By Theorem 5.4.5, $A|_{S_i} = B|_{S_i}$. Hence, $c_i = c'_i$. □

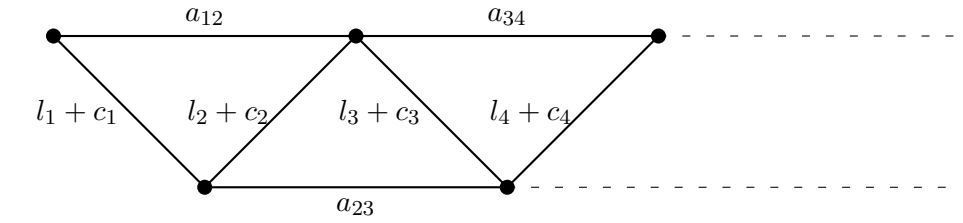
Until now, we know that the linear polynomials occurring on the diagonal edges of ABPs A and B are the same. But they can occur in different sequences in the two ABPs. We will now show that under some conditions, if $A - B = x_1x_2 + x_3x_4 + x_5x_6$, then, the two polynomials are the same, a contradiction.

Lemma 5.4.8. *If $A - B = x_1x_2 + x_3x_4 + x_5x_6$, and if the linear forms in L are 3-independent, then, the polynomials computed by the two ABPs A and B are the same.*

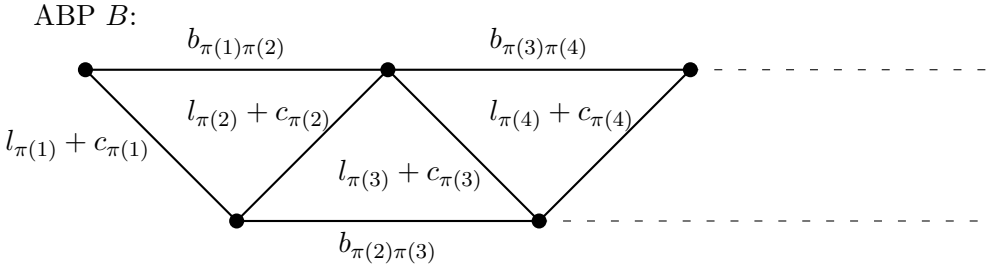
Proof. By Lemma 5.4.7, the multi-set (or, in this case, *set*) of linear *polynomials* occurring in the ABPs A and B are equal. Let that set be $\{l_1 + c_1, l_2 + c_2, \ldots l_d + c_d\}$. Let $A =$

$M_1 M_2 \cdots M_d$, with the linear form l_i occurring in the matrix M_i. Let $B = N_1 N_2 \cdots N_d$, with the linear form $l_{\pi(i)}$ occurring in the matrix N_i, where π is a permutation on the set $[d]$.

ABP A:



ABP B:



Let a_{ij} be the weight between the linear forms l_i and l_j in ABP A. If l_i and l_j are not neighbors in ABP A, then $a_{ij} = 0$. E.g. if $j \neq i+1$, then $a_{ij} = 0$. Similarly, let b_{ij} be the weight between l_i and l_j in ABP B. We will show that $b_{ij} = a_{ij}$ for all i, j. Recall that $a_{ij} \neq 0$ if and only if l_i and l_j are neighbours in ABP A. This means that $b_{ij} = a_{ij} \neq 0$ if and only if l_i and l_j are neighbours in ABP B. Thus, the linear forms l_i and l_{i+1} occur on adjacent edges in B. The ABPs A and B are the same.

It now remains to show that $b_{ij} = a_{ij}$ for all i, j. Consider the set $S_i = \{l_i, l_{i+1}\}$ for $1 \leq i \leq d-1$. Then, $A|_{S_i} = (l_i + c_i)(l_{i+1} + c_{i+1}) + a_{i,i+1}$. This is because of the 3-wise independence. Similarly, $B|_{S_i} = (l_i + c_i)(l_{i+1} + c_{i+1}) + b_{i,i+1}$. By Lemma 5.4.1, $|L \setminus \text{span}\{S_i\}| = 4 \geq 3$. By Theorem 5.4.5, $A|_{S_i} = B|_{S_i}$. Hence, $a_{i,i+1} = b_{i,i+1}$.

□

5.5 Discussion

We have studied the sum of two triangular ABPs. We have not been able to prove an impossibility result for this model yet. But we have made partial progress in this direction. It seems Theorem 5.4.5 is particularly strong. For example, it can be used to show that

If there exists a point on the two ABPs such that the span of the linear forms to its left is disjoint from the span of the linear forms to its right, and both spans have dimension 3*, then, the two ABPs compute the same polynomial.*

The part on which some light needs to be shed is the 'order of the linear forms in the two ABPs'.

Question: If the degree ≥ 3 parts of the two triangular ABPs cancel, is it that the linear forms in the two ABPs occur in the same order?

For 3-wise independent linear forms, this is true (Lemma 5.4.8). Whereas, the answer to the above question is 'No' when the span of the linear forms has dimension 2. we have an example of two different ABPs that compute the same polynomial when the span of the linear forms has dimension 2.

Eventually, we would like to see a proof of the following conjecture:

Conjecture 5.5.1. *A sum of* c *width-*2 *ABPs cannot compute* $x_1x_2+x_3x_4+\cdots+x_{c'-1}x_{c'}$*, where* $c' = 2(c+1)$*.*

Chapter 6

Conclusion

There has been a lot of progress in recent years in the field of arithmetic circuits. We will discuss here a few important open questions related to our thesis.

Connection between lower bounds and PIT

It has been observed that the models which have lower bounds also have PIT and vice versa. For example, ROABPs have lower bounds and PIT, both based on the relation between the width of the ROABP and the partial evaluation dimension. Can we formalize this connection for other circuit families within the family of ABPs?

Polynomial time PIT for ROABP

The next big aim is to get a polynomial time hitting set for ROABP. If not polynomial time, then, what about $n^{O(\log \log n)}$ time hitting set? Polynomial time hitting set for read-k ABPs, special ROABPs like set-multilinear circuits, diagonal circuits, commutative ROABPs - all of these are open questions.

The ROABP model is fairly well-understood now, because of its characterization in terms of partial evaluation dimension and a polynomial time algorithm should be available to us within the next few years.

Connection between PIT for ROABPs and pseudo-random generators for Boolean branching programs

Would better PIT algorithms for ROABPs give better pseudo-random generators for read-once oblivious Boolean branching programs (ROBPs), and vice versa? The progress in the world of read-once oblivious Boolean branching programs is being matched by the world of read-once oblivious arithmetic branching programs, e.g. invertible ROABPs, sum of ROABPs and equality of ROBPs, width-2 ROABPs. Can we formalize this connection?

Bibliography

[AB03] Manindra Agrawal and Somenath Biswas. Primality and identity testing via Chinese remaindering. *J. ACM*, 50(4):429–443, July 2003.

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, New York, NY, USA, 1st edition, 2009.

[AFS+16] Matthew Anderson, Michael A. Forbes, Ramprasad Saptharishi, Amir Shpilka, and Ben Lee Volk. Identity testing and lower bounds for read-k oblivious algebraic branching programs. In *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, pages 30:1–30:25, 2016.

[AGKS13] Manindra Agrawal, Rohit Gurjar, Arpita Korwar, and Nitin Saxena. Hitting-sets for low-distance multilinear depth-3. *Electronic Colloquium on Computational Complexity (ECCC)*, 20:174, 2013.

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

[Agr05] Manindra Agrawal. Proving lower bounds via pseudo-random generators. In *FSTTCS*, volume 3821 of *Lecture Notes in Computer Science*, pages 92–105, 2005.

[AJMV98] Eric Allender, Jia Jiao, Meena Mahajan, and V. Vinay. Non-commutative arithmetic circuits: Depth reduction and size lower bounds. *Theoretical Computer Science*, 209(1-2):47–86, 1998.

[AKS04] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. Primes is in P. *Annals of Mathematics*, 2:781–793, 2004.

[AL86] L M Adleman and H W Lenstra. Finding irreducible polynomials over finite fields. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, STOC '86, pages 350–355, New York, NY, USA, 1986. ACM.

[Alo99] Noga Alon. Combinatorial nullstellensatz. *Combinatorics, Probability and Computing*, 8(1-2):7–29, January 1999.

[ASS13] Manindra Agrawal, Chandan Saha, and Nitin Saxena. Quasi-polynomial hitting-set for set-depth-Δ formulas. In *STOC*, pages 321–330, 2013.

[ASSS12] Manindra Agrawal, Chandan Saha, Ramprasad Saptharishi, and Nitin Saxena. Jacobian hits circuits: hitting-sets, lower bounds for depth-d occur-k formulas & depth-3 transcendence degree-k circuits. In *STOC*, pages 599–614, 2012.

[AvMV15] Matthew Anderson, Dieter van Melkebeek, and Ilya Volkovich. Deterministic polynomial identity tests for multilinear bounded-read formulae. *Computational Complexity*, 24(4):695–776, 2015.

[AW16] Eric Allender and Fengming Wang. On the power of algebraic branching programs of width two. *Computational Complexity*, 25(1):217–253, 2016.

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

[BOC92] Michael Ben-Or and Richard Cleve. Computing algebraic formulas using a constant number of registers. *SIAM J. Comput.*, 21(1):54–58, 1992.

[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*, STOC '88, pages 301–309, New York, NY, USA, 1988. ACM.

[Bür00] Peter Bürgisser. Cook's versus valiant's hypothesis. *Theor. Comput. Sci.*, 235(1):71–88, 2000.

[CK00] Zhi-Zhong Chen and Ming-Yang Kao. Reducing randomness via irrational numbers. *SIAM J. Comput.*, 29(4):1247–1256, 2000.

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

[DL78] Richard A. Demillo and Richard J. Lipton. A probabilistic remark on algebraic program testing. *Information Processing Letters*, 7(4):193 – 195, 1978.

[dOSV16] Rafael Mendes de Oliveira, Amir Shpilka, and Ben Lee Volk. Subexponential size hitting sets for bounded depth multilinear formulas. *Computational Complexity*, 25(2):455–505, 2016.

[DS07] Zeev Dvir and Amir Shpilka. Locally decodable codes with two queries and polynomial identity testing for depth 3 circuits. *SIAM J. Comput.*, 36(5):1404–1434, 2007.

[DSY09] Zeev Dvir, Amir Shpilka, and Amir Yehudayoff. Hardness-randomness tradeoffs for bounded depth arithmetic circuits. *SIAM J. Comput.*, 39(4):1279–1293, 2009. (Extended abstract appeared in STOC '08).

[FS12] Michael A. Forbes and Amir Shpilka. On identity testing of tensors, low-rank recovery and compressed sensing. In *STOC*, pages 163–172, 2012.

[FS13a] Michael A. Forbes and Amir Shpilka. Explicit Noether normalization for simultaneous conjugation via polynomial identity testing. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 16th International Workshop, APPROX 2013, and 17th International Workshop, RANDOM 2013, Berkeley, CA, USA, August 21-23, 2013. Proceedings*, pages 527–542, 2013.

[FS13b] Michael A. Forbes and Amir Shpilka. Quasipolynomial-time identity testing of non-commutative and read-once oblivious algebraic branching programs. In *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 *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 867–875, 2014.

[GKKS16] Ankit Gupta, Pritish Kamath, Neeraj Kayal, and Ramprasad Saptharishi. Arithmetic circuits: A chasm at depth 3. *SIAM J. Comput.*, 45(3):1064–1079, 2016.

[GKS17] Rohit Gurjar, Arpita Korwar, and Nitin Saxena. Identity testing for constant-width, and any-order, read-once oblivious arithmetic branching programs. *Theory of Computing*, 13(2):1–21, 2017.

[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 2015, June 17-19, 2015, Portland, Oregon, USA*, pages 323–346, 2015. To appear in Computational Complexity (JCC).

[HS80] Joos Heintz and Claus-Peter Schnorr. Testing polynomials which are easy to compute (extended abstract). In *Proceedings of the 12th Annual ACM Symposium on Theory of Computing (STOC 1980)*, page 262272, 1980.

[JQS10] Maurice J. Jansen, Youming Qiao, and Jayalal Sarma. Deterministic identity testing of read-once algebraic branching programs. *Electronic Colloquium on Computational Complexity (ECCC)*, 17:84, 2010.

[Kay09] Neeraj Kayal. The complexity of the annihilating polynomial. In *Conference on Computational Complexity (CCC)*. IEEE, 2009.

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

[KMSV13] Zohar Shay Karnin, Partha Mukhopadhyay, Amir Shpilka, and Ilya Volkovich. Deterministic identity testing of depth-4 multilinear circuits with bounded top fan-in. *SIAM J. Comput.*, 42(6):2114–2131, 2013.

[KNP11] Michal Koucký, Prajakta Nimbhorkar, and Pavel Pudlák. Pseudorandom generators for group products: extended abstract. In *STOC*, pages 263–272, 2011.

[KNS16] Neeraj Kayal, Vineet Nair, and Chandan Saha. Separation between read-once oblivious algebraic branching programs (ROABPs) and multilinear depth three circuits. In *33rd Symposium on Theoretical Aspects of Computer Science, STACS 2016, February 17-20, 2016, Orléans, France*, pages 46:1–46:15, 2016.

[Kro82] Leopold Kronecker. *Grundzuge einer arithmetischen Theorie der algebraischen Grossen*. Berlin, G. Reimer, 1882.

[KS01] Adam Klivans and Daniel A. Spielman. Randomness efficient identity testing of multivariate polynomials. In *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 *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.

[LFKN92] Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *J. ACM*, 39(4):859–868, 1992.

[LMS15] Nutan Limaye, Guillaume Malod, and Srikanth Srinivasan. Lower bounds for non-commutative skew circuits. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:22, 2015.

[Lov79] László Lovász. On determinants, matchings, and random algorithms. In *FCT*, pages 565–574, 1979.

[LV98] Daniel Lewin and Salil P. Vadhan. Checking polynomial identities over any field: Towards a derandomization? In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 438–447, 1998.

[Mil07] Elizabeth Million. The Hadamard product. http://buzzard.ups.edu/courses/2007spring/projects/million-paper.pdf, [Online; accessed 9-June-2017], 2007.

[Mit13] Johannes Mittmann. *Independence in Algebraic Complexity Theory*. PhD thesis, Rheinischen Friedrich-Wilhelms-Universität, Bonn, 2013.

[MV97] Meena Mahajan and V. Vinay. Determinant: Combinatorics, algorithms, and complexity. *Chicago J. Theor. Comput. Sci.*, 1997.

[MVV87] Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7:105–113, 1987.

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

[NW96] Noam Nisan and Avi Wigderson. Lower bounds on arithmetic circuits via partial derivatives. In *Computational Complexity*, volume 6(3), pages 217–234, 1996.

[Raz05] Ran Raz. Lower bounds on algebraic circuits. 2005 Barbados Workshop on Computational Complexity, 2005.

[RR99] Ran Raz and Omer Reingold. On recycling the randomness of states in space bounded computation. In *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing, May 1-4, 1999, Atlanta, Georgia, USA*, pages 159–168, 1999.

[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.

[Sap16] Ramprasad Saptharishi. A survey of lower bounds in arithmetic circuit complexity. https://github.com/dasarpmar/lowerbounds-survey, 2016.

[Sax08] Nitin Saxena. Diagonal circuit identity testing and lower bounds. In *ICALP*, volume 5125 of *Lecture Notes in Computer Science*, pages 60–71. Springer, 2008.

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

[Sax14] Nitin Saxena. Progress on polynomial identity testing- II. In *Perspectives in Computational Complexity*, volume 26 of *Progress in Computer Science and Applied Logic*, pages 131–146. Springer International Publishing, 2014.

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

[Sha92] Adi Shamir. IP = PSPACE. *J. ACM*, 39(4):869–877, October 1992.

[SS11] Nitin Saxena and Comandur Seshadhri. An almost optimal rank bound for depth-3 identities. *SIAM J. Comput.*, 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 J. Comput.*, 41(5):1285–1298, 2012.

[SS13] Nitin Saxena and Comandur Seshadhri. From Sylvester-Gallai configurations to rank bounds: Improved blackbox identity test for depth-3 circuits. *J. ACM*, 60(5):33, 2013.

[SSS09] Chandan Saha, Ramprasad Saptharishi, and Nitin Saxena. The power of depth 2 circuits over algebras. In *FSTTCS*, pages 371–382, 2009.

[SSS13] Chandan Saha, Ramprasad Saptharishi, and Nitin Saxena. A case of depth-3 identity testing, sparse factorization and duality. *Computational Complexity*, 22(1):39–69, 2013.

[Ste12] Thomas Steinke. Pseudorandomness for permutation branching programs without the group theory. *Electronic Colloquium on Computational Complexity (ECCC)*, 19:83, 2012.

[SV11] Shubhangi Saraf and Ilya Volkovich. Black-box identity testing of depth-4 multilinear circuits. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 421–430, 2011.

[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.

[Thi98] Thomas Thierauf. The isomorphism problem for read-once branching programs and arithmetic circuits. *Chicago Journal of Theoretical Computer Science*, 1998(1), 1998.

[Tut47] William T. Tutte. The Factorization of Linear Graphs. *Journal of the London Mathematical Society*, s1-22(2):107–111, 1947.

[VSBR83] Leslie G. Valiant, Sven Skyum, S. Berkowitz, and Charles Rackoff. Fast parallel computation of polynomials using few processors. *SIAM J. Comput.*, 12(4):641–644, 1983.

[Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. In *Proceedings of the International Symposiumon on Symbolic and Algebraic Computation*, EUROSAM '79, pages 216–226, London, UK, UK, 1979. Springer-Verlag.

Index