

# Structural Results on Sparse Factoring and Identity Testing

*A Thesis Submitted*

in Partial Fulfillment of the Requirements

for the Degree of

*Doctor of Philosophy*

*by*

Pranav Bisht

17111268



*to the*

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

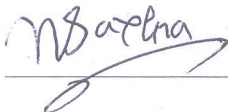
**INDIAN INSTITUTE OF TECHNOLOGY KANPUR**

**July, 2022**



# CERTIFICATE

It is certified that the work contained in the thesis entitled “Structural Results on Sparse Factoring and Identity Testing”, by “Pranav Bisht”, has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

 28 Jul'22

Nitin Saxena

Department of Computer Science and Engineering  
Indian Institute of Technology Kanpur

July, 2022



### Declaration

This is to certify that the thesis titled Structural Results on  
Sparse Factoring and Identity Testing  
has been authored by me. It presents the research conducted by me under  
the supervision of... Prof. Nitin Saxena.....

To the best of my knowledge, it is an original work, both in terms of  
research content and narrative, and has not been submitted elsewhere, in  
part or in full, for a degree. Further, due credit has been attributed to the  
relevant state-of-the-art and collaborations (if any) with appropriate  
citations and acknowledgements, in line with established norms and  
practices.

Pranav  
Signature

Name: PRANAV BISHT.....

Programme: ✓ PhD/MTech/MDes

Department: CSE.....

Indian Institute of Technology Kanpur  
Kanpur 208016



# SYNOPSIS

The area of Algebraic Complexity Theory deals with various problems related to polynomials and their respective computational complexity. In this thesis, we work on two fundamental problems in this area — Polynomial Factoring and Polynomial Identity Testing (PIT). These problems have well-known connections to each other and to other important computational problems. Both the problems admit polynomial-time randomized algorithms, while efficient deterministic algorithms are yet to be found. However, these problems have been derandomized for various restricted polynomial classes which often rely on crucial structural observations. In this work, we prove some new structural results and show how they lead to efficient factoring and identity testing algorithms for certain polynomial classes.

**Sparse Factoring:** Given an input polynomial  $f \in \mathbb{F}[x_1, \dots, x_n]$ , the complete factorization of  $f$  is a representation of  $f$  as  $f_1^{e_1} \cdot f_2^{e_2} \cdot \dots \cdot f_k^{e_k}$ , where  $f_1, \dots, f_k$  are co-prime irreducible polynomials and  $e_1, \dots, e_k$  are positive integers. Sparsity of a polynomial  $f$  is defined as the number of monomials with non-zero coefficients in  $f$ . A list of these monomials along with their non-zero coefficients is called the sparse representation of  $f$ . We call  $f$  a sparse polynomial if its sparsity is a small number, typically bounded by  $\text{poly}(n)$ . In sparse factoring, the input polynomial is provided in the sparse representation and one is asked to factor the given polynomial and output its irreducible factors, also in the sparse representation. In general, sparse polynomials are known to have dense factors. Therefore, the class of sparse polynomials with constant individual degrees is studied, which is widely believed to be closed under factors.

The key combinatorial object studied for factor-sparsity is the Newton polytope. In this work, we introduce a new notion of *min-entropy* for a set of vectors. We prove that for a Newton polytope with a ‘low’ min-entropy vertex set, all the integral points inside the polytope will also have low min-entropy. We use this structural result to bound the total number of integral points in the Newton polytope of a *symmetric* polynomial. This eventually leads to a polynomial-sized upper bound on the sparsity of the factors of a symmetric, sparse polynomial with bounded individual degree. This in turn yields an efficient deterministic factoring algorithm for such polynomials.

We also study other variants of this problem. For a polynomial  $f$  which factors as  $f = g \cdot h$ , we call  $g$  a co-factor of  $h$  and vice-versa. We prove a new sparsity bound for the co-factor of some multilinear factor of a sparse polynomial. We also study a decision problem — which we call exact power testing — in which we design an efficient deterministic algorithm to test whether a given sparse polynomial  $f$  is an exact power of some other polynomial  $g$ , i.e. test whether  $f = g^e$ , for some positive constant  $e$ .

**Polynomial Identity Testing:** Given an input polynomial  $f$ , the Polynomial Identity Testing (PIT) problem asks to determine if  $f$  is an identically zero polynomial. In general, the input polynomial  $f$  is given as an algebraic circuit. There are two flavors here — whitebox and blackbox PIT. In whitebox PIT, one is allowed to look inside the circuit, whereas in blackbox setting, the input is provided in the form of a blackbox which can only be evaluated at field points. Despite the general problem being still open, various restricted classes have been solved and gradually stronger models are introduced and attacked. In this work, we study PIT for two different classes — sum of ROABPs (read-once oblivious algebraic branching programs) and  $\Sigma^{[2]}\Pi\Sigma\Pi^{[\text{ind-deg } d]}$  circuits.

PIT for ROABPs has been extensively studied and quasi-polynomial time blackbox algorithms are known for this class. However, PIT for sum of ROABPs is a much tougher problem as there are explicit polynomials that can be expressed as sum of just two ROABPs but require exponential size when expressed as a single ROABP. In this work, we give a new reduction from PIT of sum of ROABPs to PIT of a single ROABP, which improves over



the previous known reduction in the log-variate setting (when the number of variables is logarithmic to the size of input). More generally, we extend our reduction to the border class of sum of ROABPs. We also show a poly-time blackbox PIT for a single log-variate ROABP of constant width. Combined with our reduction, we get a poly-time blackbox PIT for sum of constant-many, log-variate, constant-width ROABPs. The PIT for single ROABP here is a consequence of the new structural result that we prove — an ROABP computing a homogeneous polynomial can be syntactically homogenized in the same width. The standard homogenization tricks incur a blow-up in width, which is fatal in the constant-width regime.

The depth-4  $\Sigma\Pi\Sigma\Pi$  circuits are widely studied due to the famous depth-4 and depth-3 chasm results, which show that it is sufficient to solve PIT for these classes in order to get non-trivial PIT for general circuits. Not surprisingly therefore, depth-4 circuits are difficult to solve and various restricted subclasses are studied, for example restricting the top and bottom fan-ins to constants. In this work, we study the  $\Sigma^{[2]}\Pi\Sigma\Pi^{[\text{ind-deg } d]}$  class, for which the PIT question effectively asks whether  $\prod_{i=1}^r f_i = \prod_{j=1}^m g_j$ , where every  $f_i$  and  $g_j$  is a sparse polynomial of constant individual degree  $d$ . In the whitebox setting, this problem reduces to the sparse factoring problem as one can simply factor every polynomial in LHS and RHS and compare each irreducible factor. Even then, it will require quasi-polynomial time, which is the best known time-complexity for sparse factoring. Moreover, this idea does not work in the blackbox setting. In this work, we develop a new method which does not depend on solving sparse factoring and yet it yields a poly-time blackbox PIT. This is achieved via a structural result that links the subresultant of two polynomials with their gcd and resultant of their co-prime parts.



# Acknowledgements

I feel extremely fortunate to have been advised by Prof. Nitin Saxena as my Ph.D supervisor. During my MTech and PhD at IIT Kanpur, he has trained me from a student to a researcher. He has mentored and continues to guide several students in Algebraic complexity theory, which in itself is a significant contribution for the future of this area, besides his independent research output. He teaches essential courses for any beginner in complexity theory, which makes it easy to understand state-of-the-art of this field. His disciplined lifestyle and unyielding style of research are sources of inspiration for me. I am grateful to Prof. Ilya Volkovich for being my collaborator and friend. His technical expertise and the ability to abstract out and simplify complicated concepts is mesmerizing. I would also like to thank Prof. Manindra Agrawal, Prof. Rajat Mittal, Prof. Satyadev Nandakumar, Prof. Raghunath Tewari and other faculty in the department for imparting their knowledge through helpful courses.

I thank my partner-in-crime Ashish Dwivedi for being a true friend and companion throughout this long journey. His purity of thoughts and youthful zeal for research inspire me. I hope we continue to keep in touch as friends and researchers. I would like to thank Samik Some for being friends and helping me multiple times during my stay here. I will remember the three of us hanging out in campus every now and then. I thank my bachelor buddy Vishesh Jindal for regularly calling me and staying in touch. I also thank my master's buddy Chandan Pandey for his discussions and wish him good times in the coming future. I would like to thank my caring seniors - Amit and Sumanta for guiding me multiple times. Finally, I would like to thank my theory peers - Rajendra, Priyanka, Mahesh, Pranjal, Bhargav, Prateek, Diptajit, Subhayan, Sanyam.

Most importantly, I dedicate this thesis to my mother, who encouraged me to pursue higher studies from the very beginning. I am grateful to my father for being patient and helping me through tough times.

# Contents

<b>Acknowledgements</b>	<b>ix</b>
<b>List of Publications</b>	<b>xv</b>
<b>List of Figures and Tables</b>	<b>xvii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Algebraic models of computation . . . . .	3
1.1.1 Algebraic circuits . . . . .	3
1.1.2 ABPs . . . . .	4
1.1.3 ROABPs . . . . .	5
1.2 Sparse polynomial factorization . . . . .	6
1.3 Polynomial Identity Testing . . . . .	9
1.4 Contribution of this thesis . . . . .	11
1.4.1 Sparse factoring results . . . . .	11
1.4.2 PIT results . . . . .	16
1.5 Organization of the thesis . . . . .	21
<b>I Sparse Polynomial Factorization</b>	<b>23</b>
<b>2 Factoring Preliminaries</b>	<b>25</b>
2.1 Notations . . . . .	25
2.2 Definitions . . . . .	26

2.2.1	Content and Primitive parts . . . . .	27
2.2.2	Multivariate Reversal operation . . . . .	29
2.3	The sparsity connection with Newton polytopes . . . . .	30
2.4	Symmetric Polytopes . . . . .	31
<b>3</b>	<b>Sparse Factoring via Symmetric Polytopes</b>	<b>35</b>
3.1	Polytope Entropy . . . . .	36
3.2	Polytope point counting . . . . .	40
3.2.1	Point counting . . . . .	41
3.3	Factor sparsity bounds . . . . .	43
3.3.1	Tightness of sparsity bounds . . . . .	45
3.4	Factoring algorithms . . . . .	46
3.5	Discussion . . . . .	48
<b>4</b>	<b>Exact Power Testing</b>	<b>51</b>
4.1	Exact Power Testing . . . . .	52
4.1.1	Reverse Monic Case . . . . .	52
4.1.2	General Case . . . . .	55
4.2	Discussion . . . . .	60
<b>5</b>	<b>Co-factor Sparsity via Unique Projections</b>	<b>61</b>
5.1	Multilinear co-Factor Motivation . . . . .	62
5.2	Co-Factor Polynomial Sparsity . . . . .	64
5.2.1	Unique Projections . . . . .	64
5.2.2	Co-factor sparsity bounds . . . . .	66
5.3	Discussion . . . . .	71
<b>II</b>	<b>Polynomial Identity Testing</b>	<b>73</b>
<b>6</b>	<b>PIT Preliminaries</b>	<b>75</b>
6.1	Notations . . . . .	75

6.2	Hitting set generator (HSG) . . . . .	76
6.3	GCD, Resultants and Subresultants . . . . .	80
6.4	ROABPs . . . . .	84
6.5	Border PIT . . . . .	89
<b>7</b>	<b>PIT for <math>\Sigma^{[2]}\Pi\Sigma\Pi^{[\text{ind-deg } d]}</math> circuits</b>	<b>91</b>
7.1	Proof Technique . . . . .	92
7.2	Structure Theorem . . . . .	94
7.3	PIT for $\Sigma^{[2]}\Pi\Sigma\Pi^{[\text{ind-deg } d]}$ Circuits . . . . .	97
7.3.1	The $\Sigma^{[k]}\Pi\Sigma\Pi^{[\text{ind-deg } d]}$ Model . . . . .	97
7.3.2	Vectors and Interlacing . . . . .	98
7.3.3	The PIT Algorithm . . . . .	99
7.4	Discussion . . . . .	106
<b>8</b>	<b>PIT for sum of ROABPs and its Border Class</b>	<b>109</b>
8.1	Previous works and motivation . . . . .	110
8.2	PIT for Degree-preserving sum of ROABPs . . . . .	113
8.2.1	Syntactically Homogeneous ROABP . . . . .	113
8.2.2	PIT for single ROABP . . . . .	116
8.2.3	PIT for Degree-Preserving Sum . . . . .	119
8.3	PIT for Sum of ROABPs . . . . .	120
8.3.1	Sum of two ROABPs . . . . .	122
8.3.2	Sum of $c$ ROABPs . . . . .	128
8.4	PIT for Border . . . . .	132
8.5	Discussion . . . . .	135
<b>9</b>	<b>Conclusion</b>	<b>139</b>
	<b>Bibliography</b>	<b>142</b>
	<b>Index</b>	<b>151</b>





# List of Publications

[BS22] **Derandomization via symmetric polytopes: Poly-time factorization of certain sparse polynomials**

*with Nitin Saxena.*

42nd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS), 2022.

[BV22] **On Solving Sparse Polynomial Factorization Related Problems**

*with Ilya Volkovich.*

42nd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS), 2022.

[BS21] **Blackbox identity testing for sum of special ROABPs and its border class**

*with Nitin Saxena.*

Journal of computational complexity, Volume 30:8, 2021.

Chapter 3 is based on [BS22]. Chapters 4, 5 & 7 are based on [BV22]. Chapter 8 is based on [BS21]. The content related to [BS21] in this thesis has been reproduced with permission from Springer Nature.



# List of Figures

1.1	A circuit computing the polynomial $x^2 + y + 1$ . . . . .	4
1.2	ABP of width 2, depth 3 computing $f = x_2x_3$ . . . . .	5
1.3	ROABP computing $f = (x_1 + y_1)(x_2 + y_2) \cdots (x_n + y_n)$ . . . . .	6

# List of Tables

8.1	Time complexities of different PIT algorithms related to $n$ -variate, degree $d$ and width $r$ ROABP model. . . . .	111
-----	--	-----



# Chapter 1

## Introduction

Computational complexity theory is a branch of theoretical computer science that aims to classify various computational problems according to their inherent complexity. The complexity of the problem is measured by the amount of resources required to solve them in a well defined mathematical model. The traditional model is the well known Turing machine model, in which time and space are the two main resources. The celebrated P vs NP problem can be considered the driving force for this area. As this problem continues to be elusive, various other computational models have been studied in the quest of proving hardness of explicit problems in these models. Communication complexity, quantum computing, circuit complexity etc. are now independent yet connected areas of research based on different computational models.

Another sub-area is that of randomized algorithms, which is modeled using the probabilistic Turing machine, that has an additional resource of randomness. Here the most important class is BPP, which represents the class of problems having efficient randomized algorithms. BPP acts as the randomized equivalent of class P. This leads to the following philosophical question: *Is randomness necessary?* or formally is  $P = BPP$ ? It is widely believed to be true. In other words, it is believed that any randomized algorithm can be converted into a deterministic algorithm with only a polynomial blow-up in its time-complexity. This is called derandomization. Many problems have been derandomized, for example the famous primality testing algorithm of [AKS04]. Yet there are

many interesting problems that admit efficient randomized algorithms but for which no efficient deterministic algorithms are known. We study two such fundamental problems in this thesis: Polynomial Factorization and Polynomial Identity Testing (PIT). Given a polynomial  $f \in \mathbb{F}[x_1, \dots, x_n]$ , *multivariate* polynomial factorization asks to output all its irreducible factors, while PIT asks to determine if  $f$  is identically zero. The form in which the input polynomial  $f$  is provided is significant in the complexity of these problems.

This thesis will focus on the model of sparse polynomials for factoring. Sparsity of a polynomial  $f \in \mathbb{F}[x_1, \dots, x_n]$  is defined as the number of monomials with non-zero coefficients in  $f$ . A polynomial is called  $s$ -sparse if it has at most  $s$ -many terms. Degree of a monomial is the sum of degrees of the variables in it. Degree (total-degree) of a polynomial is the maximum degree of any monomial in it. *Individual degree* of a polynomial is the maximum degree of any variable in the polynomial. Note that sparsity for an  $n$ -variate, individual degree  $d$  polynomial can be exponentially high – at most  $(d+1)^n$ . While for the class of sparse polynomials it is low, typically  $\text{poly}(n)$ . For PIT in this thesis, we study the models of depth four circuits and ROABPs (read-once oblivious algebraic branching programs). We define these along with other algebraic models in the section below.

It turns out that both these fundamental problems are connected to each other. It is easy to show that PIT reduces to factorization. Observe that the polynomial  $x^2 + y \cdot f$  factorizes if and only if  $f = 0$ . The other direction is non-trivial and shown in [KSS14]. They showed that the problem of derandomizing multivariate polynomial factoring reduces to derandomizing PIT, for general algebraic circuits (assuming deterministic univariate factoring). Showing this equivalence for other models was left as an open problem by the authors. The classes which admit an efficient deterministic PIT algorithm are particularly interesting because if we show the equivalence for these classes, we also derandomize factoring for them. Sparse polynomials is one such natural class, for which we do have poly-time deterministic PIT algorithms [KS01] but poly-time factoring is open.

## 1.1 Algebraic models of computation

Polynomials are best studied using algebraic models of computation. The area of algebraic complexity theory analyzes polynomials in terms of their complexity and studies related computational problems such as factoring or PIT. Below, we define various algebraic models for computation of a polynomial. For details, we refer the reader to the excellent survey of [SY10].

### 1.1.1 Algebraic circuits

An *algebraic circuit* is a directed acyclic graph where computation is done bottom-up, with input leaves at the bottom and a single output node at top. The leaves are labeled with variables or field constants while rest of the nodes are either addition or multiplication nodes. The directed edges  $u \rightarrow v$  are labeled with field constants, which get multiplied to the polynomial computed at node  $u$  before feeding it to node  $v$ . The in-degree of a node is called its *fan-in* and out-degree is called *fan-out*. *Size* of the circuit is simply size of the directed graph. *Depth* of the circuit is length of the longest path from a leaf to the output node. *Degree* of the circuit is maximum degree of a polynomial computed at any node in the circuit. The much more general class of  $\text{poly}(n)$ -sized and  $\text{poly}(n)$  degree algebraic circuits is called **VP**, which is considered the algebraic analog of complexity class **P**. The class **VNP** is considered the algebraic analog of complexity class **NP**. It is the class of polynomials which can be expressed as an exponential sum of a projection of a **VP** circuit family. Like the **P** vs **NP** question in the classical model, here the main aim is to prove  $\text{VP} \neq \text{VNP}$ . An algebraic circuit where fan-out of every node is one is called an *algebraic formula*. The class of polynomial sized formulas is called **VF**.

Algebraic circuits can be assumed to be layered with alternating layer of  $+$  and  $\times$  nodes, with the root node to be addition gate. A size- $s$  *depth-2*  $\Sigma\Pi$  circuit computes a sum of  $s$ -many monomials. Thus, depth-2 circuits compute the class of sparse polynomials.

A *depth-3*  $\Sigma\Pi\Sigma$  circuit computes polynomials of the form  $f = \sum_{i=1}^k \prod_{j=1}^{d_i} \ell_{ij}$ , where each  $\ell_{ij}$  is a *linear* polynomial. A sub-model of depth-3 circuits called *diagonal depth-3*

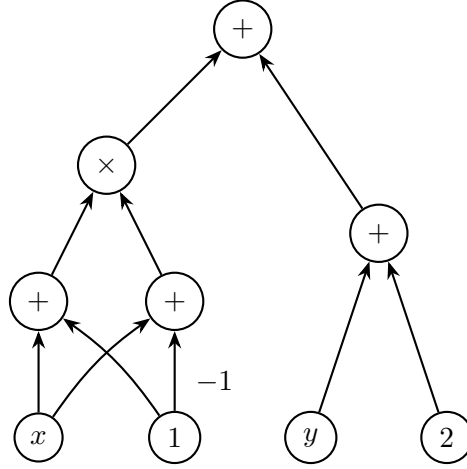


Figure 1.1: A circuit computing the polynomial  $x^2 + y + 1$ .

circuits  $\Sigma \wedge \Sigma$ , computes polynomials of the form  $f = \sum_{i=1}^k \ell_i^{d_i}$ , where each  $\ell_i$  is a linear polynomial.

A *depth-4* circuit is of the form  $\Sigma \Pi \Sigma \Pi$ . Thus, a size- $s$  circuit in this class computes polynomials of the form  $f = \sum_{i=1}^k \prod_{j=1}^{d_i} f_{ij}$ , where each  $f_{ij}$  is an  $s$ -sparse polynomial. In this thesis, we also consider a sub-model of depth-4 circuits, which we call  $\Sigma^{[k]} \Pi \Sigma \Pi^{[\text{ind-deg } d]}$ . Here,  $k, d$  are considered constants and each  $f_{ij}$  is a sparse polynomial of individual degree  $d$ .

### 1.1.2 ABPs

An *algebraic branching program* (ABP) is a layered directed graph with a unique source vertex  $s$  and sink vertex  $t$ . The ABP of *depth- $d$*  has  $d + 1$  layers—  $V_0, V_1, \dots, V_d$ , where first layer  $V_0 =: \{s\}$ , and last layer  $V_d =: \{t\}$ . The directed edges go from  $V_i$  to  $V_{i+1}$ , for  $0 \leq i \leq d - 1$  and are labeled with *linear* polynomials from  $\mathbb{F}[\mathbf{x}]$ . The *weight of a path*  $p$  is  $W(p) := \prod_{e \in p} W(e)$ , where  $W(e)$  denotes the weight (or label) of an edge. The final polynomial  $f(\mathbf{x})$  *computed by the ABP* is then simply the sum of weight of all paths from source to sink:  $f(\mathbf{x}) := \sum_{\text{path } p : s \rightsquigarrow t} W(p)$ . The *length* of the ABP is the number of layers from  $s$  to  $t$ . The ABP has *width*  $w$ , if for  $0 \leq i \leq d$ ,  $|V_i| \leq w$ . *Size* of the ABP is its graph size, which is product of its length and width. **VBP** is the class of all polynomial-sized ABPs.



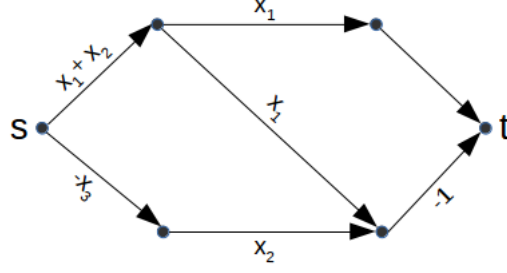


Figure 1.2: ABP of width 2, depth 3 computing  $f = x_2x_3$ .

ABP also has an alternate algebraic representation in terms of matrix product. Let the set of vertices in  $i^{th}$ -layer  $V_i$  be  $V_i =: \{v_{i,j} \mid j \in [w]\}$ . Then,  $f(\mathbf{x}) = \prod_{i=1}^d D_i$ , where  $D_1 \in \mathbb{F}^{1 \times w}[\mathbf{x}]$ ,  $D_i \in \mathbb{F}^{w \times w}[\mathbf{x}]$  (for  $2 \leq i \leq d-1$ ), and  $D_d \in \mathbb{F}^{w \times 1}[\mathbf{x}]$  such that the entries are:

$$D_1(j) := W(s, v_{1,j}) , \text{ for } j \in [w]$$

$$D_i(j, k) = W(v_{i-1,j}, v_{i,k}) , \text{ for } j, k \in [w] \text{ and } 2 \leq i \leq d-1$$

$$D_d(k) = W(v_{d-1,k}, t) , \text{ for } k \in [w] .$$

By default  $W(u, v) := 0$ , if there is no edge  $(u, v)$  in the ABP. For the ABP in Figure 1.2, the matrix product representation is:

$$\begin{bmatrix} 1 & -1 \end{bmatrix} \begin{bmatrix} x_1 + x_2 & 0 \\ 0 & x_3 \end{bmatrix} \begin{bmatrix} x_1 & x_1 \\ 0 & x_2 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

### 1.1.3 ROABPs

An ABP is called *read-once oblivious* ABP (ROABP) if each variable appears in only one layer and instead of linear polynomials, edge weights are univariate polynomials. So, ROABP has length equal to the number of variables  $n$ . The *variable order*  $(x_{\pi(1)}, \dots, x_{\pi(n)})$  of ROABP is the order of variables as they appear in edge weights between the layers  $i-1$

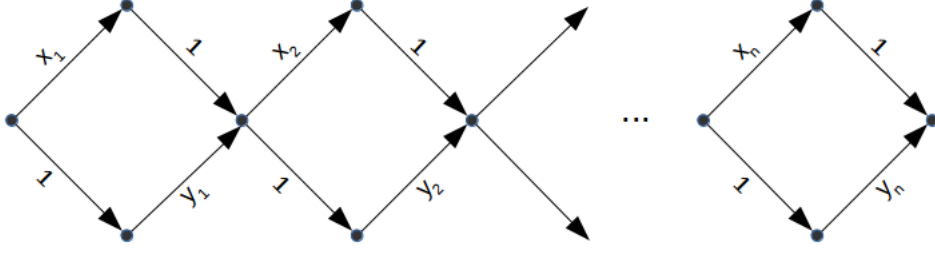


Figure 1.3: ROABP computing  $f = (x_1 + y_1)(x_2 + y_2) \cdots (x_n + y_n)$ .

to  $i$  in the ROABP, where  $i$  ranges from 1 to  $n$ . *Size* of the ROABP is the sum of its graph size and the individual degrees (of the univariate edge-labels). The three size parameters here are width, degree and number of variables.

In the matrix product form, ROABP  $D(\mathbf{x}) = \prod_{i=1}^n D_i$ , where  $D_1 \in \mathbb{F}^{1 \times w}[x_{\pi(1)}]$ ,  $D_i \in \mathbb{F}^{w \times w}[x_{\pi(i)}]$  for  $2 \leq i \leq n-1$ , and  $D_n \in \mathbb{F}^{w \times 1}[x_{\pi(n)}]$ . One can also view  $D_i$  as a univariate polynomial with coefficients coming from  $w$ -dimensional vectors or  $w \times w$  matrices. For ROABP  $D(\mathbf{x})$ ,  $D_{\leq i}$  denotes the sub product  $\prod_{j=1}^i D_j$ , and  $D_{> i}$  denotes  $\prod_{j=i+1}^n D_j$ .

## 1.2 Sparse polynomial factorization

Polynomial factorization is an important problem with interesting connections to circuit lower bounds [KI04], list decoding [Sud97, GS98] and cryptography [CR88]. It admits an efficient randomized algorithm [Kal89, KT90] when the input is given in the form of an algebraic circuit. See [Kal03, vzG06, vzGG13, Sud98] for a detailed exposition.

In sparse polynomial factorization, we give a sparse polynomial as input and ask output factors also in the sparse representation. This problem was first studied in [vzGK85], where a randomized algorithm for the same was provided. The time complexity of their algorithm was polynomial in the sparsity of factors and exponential in the number of factors. Naturally, [vzGK85] raised the question of proving better sparsity bounds for factors of sparse polynomials. Note that the time-complexity of factoring algorithm will be lower bounded by the sparsity of factors, just to write the output factors in sparse representation

alone. Unfortunately, sparse polynomials can have dense factors. For characteristic zero fields, Example 1.2.1 gives an  $s$ -sparse polynomial which has a factor of sparsity  $s^{\log d}$ , where  $d$  is the individual degree.

**Example 1.2.1** ([vzGK85]). Consider the following polynomial  $f$  with individual degree  $d$ , which factorizes as  $f = gh$ , where

$$f = \prod_{i=1}^n (x_i^d - 1), \quad g = \prod_{i=1}^n (1 + x_i + \dots + x_i^{d-1}), \quad h = \prod_{i=1}^n (x_i - 1).$$

For finite fields, the situation is even worse. Here, Example 1.2.2 gives an  $s$ -sparse polynomial with a factor of  $s^d$  sparsity.

**Example 1.2.2** ([BSV20]). Over finite fields we have the following polynomial which has a higher exponential blowup in factor-sparsity. Consider the following polynomial  $f \in \mathbb{F}_p[x_1, \dots, x_n]$  with individual degree  $p$ , for some prime  $p$  and let  $0 < d < p$ . We have  $f = gh$ , where

$$\begin{aligned} f &= x_1^p + x_2^p + \dots + x_n^p = (x_1 + x_2 + \dots + x_n)^p, \\ g &= (x_1 + x_2 + \dots + x_n)^d, \\ h &= (x_1 + x_2 + \dots + x_n)^{p-d}. \end{aligned}$$

These examples give the best known lower bounds on sparsity of factors. The best known factor-sparsity upper bound for an  $s$ -sparse polynomial is  $s^{O(d^2 \log n)}$  due to [BSV20]. It is conjectured that the true bound depends only on  $d$  in the exponent and does not have this  $\log n$  dependence.

**Conjecture 1.2.3.** *Let  $f$  be an  $s$ -sparse polynomial with individual degree  $d$  and  $g$  be a factor of  $f$ . Then, sparsity of  $g$  is upper bounded by  $s^{\mu(d)}$ , where  $\mu : \mathbb{N} \rightarrow \mathbb{N}$  is any function.*

Therefore, researchers study the sparse factoring problem in the setting where individual degree  $d$  is considered constant. Even designing factoring algorithms for sparse polynomials of individual degree one or two required non-trivial insights. An efficient deterministic algorithm for factorization of sparse multilinear polynomials ( $d = 1$ ) was

provided in [SV10]. This result was further generalized to sparse polynomials that factorize into multilinear factors in [Vol15]. The model of sparse multiquadratic polynomials ( $d = 2$ ) was solved in [Vol17].

In Chapter 3, we make progress towards this conjecture by proving an  $s^{O(d^2 \log d)}$  factor-sparsity bound for the class of  $s$ -sparse symmetric polynomials. Symmetric polynomials is a natural class which is studied extensively in both computer science and mathematics. Many multivariate polynomials  $f \in \mathbb{F}[x_1, \dots, x_n]$  are constructed by ‘boosting’ a univariate polynomial  $a(X)$  by product or addition as shown below:

$$f = \prod_{i=1}^n a(x_i) \quad \text{or} \quad f = \left( \sum_{i=1}^n a(x_i) \right)^d \quad \text{for some } d \geq 1.$$

Such polynomials are actually symmetric by construction. We note that the famous examples (Example 1.2.1 and Example 1.2.2) fall under this category. Besides symmetric polynomials, we also prove polynomial-sized sparsity bounds for another class of polynomials, which we call *low min-entropy* polynomials (See Definition 2.2.1).

Given an input  $f$  in a particular representation, the *Factor Closure problem* asks for the best bound on size of factors in the same representation. The foundational work of [Kal86, Kal87, Kal89] showed that if  $f$  is a size- $s$ , degree- $\text{poly}(n)$  algebraic *circuit*, then its factors also have  $\text{poly}(s, n)$ -sized algebraic circuits, i.e. VP is closed under factoring. [DSS18] proved factor closure for the classes of VF, VBP, VNP with a quasi-polynomial blowup in size, and gave analogous whitebox algorithms (see Section 1.1 for definitions of these classes). VNP was shown to be properly closed under factoring (with only poly blowup in size) in [CKS19]. Recently, [ST21b] showed that size- $s$  ABPs have factors of size  $\text{poly}(s)$ , thus proving factor closure for VBP class. For algebraic formulas, [Oli16] showed that if  $f$  is computed by a depth- $\Delta$ , size- $s$  algebraic formula, then its factors can be computed by depth- $(\Delta + 5)$  and size  $\text{poly}(s)$  formulas, provided that individual degree of  $f$  is constant. Observe that sparse polynomials can also be seen as depth-2 algebraic formulas, thus [Oli16] showed that factors of bounded individual degree sparse polynomials can be computed by depth-7 formulas. However, [BSV20] showed that these factors can be computed in depth-2 itself with a quasi-polynomial blowup in size. We show that if  $f$

is also symmetric, then the factors can even be computed in depth-2 efficiently, with only a polynomial blowup in size.

The work of [BSV20] also gives a deterministic factorization algorithm for sparse polynomials with constant individual degree which runs in  $s^{O(\log n)}$  time. With our new  $\text{poly}(s)$ -sparsity bound, we use their algorithm to get a deterministic  $\text{poly}(s)$ -time factorization algorithm for symmetric,  $s$ -sparse polynomials and also for low min-entropy polynomials, in the bounded individual degree regime.

### 1.3 Polynomial Identity Testing

Polynomial Identity Testing (PIT) is the problem of testing whether a given multivariate polynomial is identically zero or not. The input polynomial to be tested is usually given in a compact representation – like an *algebraic circuit* or an *algebraic branching program* (ABP). The PIT algorithm is said to be efficient if its time complexity is polynomial in the input size of algebraic circuit resp. ABP. There are two main types of PIT algorithms – *blackbox* or *whitebox*. A blackbox PIT algorithm tests the zeroness of input polynomial using only evaluations of circuit, resp. ABP, over field points. However, a whitebox algorithm is allowed additional access to look inside the circuit or ABP. The set of points  $\mathcal{H}$  over which a blackbox PIT algorithm evaluates is also commonly known as a *hitting-set*. PIT admits a simple yet efficient randomized blackbox algorithm due to *Polynomial Identity Lemma* [Sch80, Zip79, DL77, Ore22]. The primary focus of research in PIT is to derandomize it and get a poly-time deterministic blackbox algorithm. The problem of PIT also has interesting connections with circuit lower bounds [HS80, KI04, Agr05, AGS19], geometric complexity theory [Mul12b, Mul12a] and many other well known problems like matching [MVV87, FGT17], primality testing [AKS04] and polynomial factoring [KSS14]. Refer to [SY10, Sax09, Sax14, Sap16] for detailed surveys on PIT and lower bounds.

The first PIT model we study in this thesis is a sub-class of depth-4 algebraic circuits. The depth-4  $\Sigma\Pi\Sigma\Pi$  circuit class is extremely important in the context of the PIT problem, as it is known that a polynomial-time black-box PIT for this class implies a

quasi-polynomial-time black-box PIT for general VP circuits [AV08, AGS19]. For a long time, no PIT algorithm better than the trivial  $d^{O(n)}$  time algorithm was known for this class, until the recent breakthrough lower bound result of [LST22], which also gives a sub-exponential time PIT algorithm. Various restricted versions of depth-4 circuits are studied to get close to polynomial-time PIT algorithms. For example, [PS21] give a polynomial-time PIT algorithm for  $\Sigma^{[3]}\Pi\Sigma\Pi^{[2]}$  circuits, where the top fan-in is 3 and the bottom fan-in is 2. Recently, [DDS21] gave a quasi-polynomial-time PIT for  $\Sigma^{[k]}\Pi\Sigma\Pi^{[d]}$  circuits, where the top fan-in  $k$  and bottom fan-in  $d$  are allowed to be any fixed constants. In this model, the restriction on bottom fan-in implies that the bottom  $\Sigma\Pi$  computes polynomials of total degree at most  $d$ . *We give polynomial-time PIT algorithm for  $\Sigma^{[2]}\Pi\Sigma\Pi^{[\text{ind-deg } d]}$  model*, where the top fan-in is 2 and the bottom  $\Sigma\Pi$  computes polynomials with *individual* degree at most  $d$ . We note that the individual degree restriction is much weaker than the total degree restriction. Indeed, even for the case of individual degree bounded by 1 (i.e. multilinear polynomials) the total degree can still be  $\Omega(n)!$  [SV18] gave a polynomial-time PIT algorithm for the class of multilinear  $\Sigma^{[k]}\Pi\Sigma\Pi$  circuits, with constant top fan-in  $k$ , where every gate in the circuit computes a multilinear polynomial. Yet, even a white-box polynomial-time PIT for *general*  $\Sigma^{[2]}\Pi\Sigma\Pi$  circuits is still open.

In [vzGK85], another problem was posed alongside the sparse factorization problem, in the hope that it might be easier. This problem is referred to as *testing sparse factorization*. Given  $m + 1$  sparse polynomials  $f, g_1, \dots, g_m$ , it asks to test whether  $f = g_1 \cdot \dots \cdot g_m$ . The work of [SSS13] gives a polynomial-time algorithm for this problem, in the special case where every  $g_i$  is a sum of univariate polynomials. [Vol17] gives a polynomial-time algorithm when  $f$  (and therefore every  $g_i$ ) has constant individual degree and each  $g_i$  is an irreducible polynomial. Our PIT result for  $\Sigma^{[2]}\Pi\Sigma\Pi^{[\text{ind-deg } d]}$  is connected to this problem. We give a polynomial-time algorithm to test whether  $\prod_{i=1}^r f_i = \prod_{j=1}^m g_j$ , where each  $f_i, g_j$  is a sparse polynomial with constant individual degree. Note that now LHS is also a product of polynomials. Moreover, there is no restriction placed on  $g_j$ -s except that they have bounded individual degree.

The second PIT model that we study in this thesis is that of read-once oblivious

ABPs (ROABPs), defined earlier. In the whitebox regime, ROABP has a well known poly-time PIT algorithm [RS05]. However, in the blackbox regime, we only have quasi-poly-time PIT algorithms [FS13b, AGKS15] and no known poly-time algorithms. Thus, one can also ask for blackbox PIT of ROABPs with restriction on the width parameter. In [GKS17] they address this question and give a poly-time graybox (known variable order) PIT for constant width ROABPs. The constant width setting can be considered a necessary stepping stone before solving PIT for general width ROABPs. The sum of ROABPs is another interesting model for PIT. For a constant number of ROABPs, [GKST17] give the first poly-time whitebox, and only a *quasi*-poly time blackbox PIT algorithm. One can then ask for *poly-time* blackbox PIT for sum of ROABPs under the restriction of constant width. This problem is also open. What if we also restrict the number of variables? It is a nontrivial model as the degree remains arbitrary. This brings us to the question of poly-time blackbox PIT for sum of constantly-many, constant-width, log-variate ROABPs. We give a positive answer for this question in this thesis. We show that *blackbox PIT for sum of constantly-many, log-variate constant-width ROABPs is in polynomial time*.

## 1.4 Contribution of this thesis

In what follows, let  $\mathbb{F}$  be any arbitrary field, finite or otherwise.

### 1.4.1 Sparse factoring results

We prove the following structural result in sparse factoring: A symmetric sparse polynomial of constant individual degree has sparse factors.

**Theorem** (Theorem 3.3.3). *Let  $f \in \mathbb{F}[x_1, \dots, x_n]$  be an  $s$ -sparse, symmetric polynomial with individual degree  $d$ . Then, every factor of  $f$  has its sparsity bounded by  $s^{O(d^2 \log d)}$ .*

For constant  $d$ , the above bound is  $\text{poly}(s)$ . Previous best sparsity upper bound for factors of  $f$  was  $s^{O(d^2 \log n)}$  due to [BSV20], which is super polynomial even for constant  $d$ . Our

result shows a positive evidence for Conjecture 1.2.3, and it suggests that general factor-sparsity bound should be independent of this  $\log n$  dependence. We can also work with a general (possibly *non-symmetric*)  $f$  and get the same sparsity bound for any *symmetric* factor of  $f$  (if it exists). See Theorem 3.3.1.

Given a polynomial  $f$ , the *complete factorization* of  $f$  is a representation of  $f$  as  $f_1^{e_1} f_2^{e_2} \cdots f_k^{e_k}$ , where  $f_1, \dots, f_k$  are co-prime, irreducible polynomials and  $e_1, \dots, e_k$  are positive integers. Let  $c_{\mathbb{F}}(d)$  denote the best known time complexity for factoring a univariate polynomial of degree  $d$  over the field  $\mathbb{F}$ . For  $\mathbb{F} = \mathbb{Q}$ ,  $c_{\mathbb{F}}(d) \leq \text{poly}(d, t)$  where  $t$  is the maximum bit-complexity of the coefficients of  $f$  [LLL82]. For a finite field  $\mathbb{F} = \mathbb{F}_{p^\ell}$ ,  $c_{\mathbb{F}}(d) \leq \text{poly}(\ell \cdot p, d)$  [Ber67, CZ81]. We get the following sparse factoring algorithm as a direct corollary of our factor-sparsity bound above.

**Corollary** (Corollary 3.4.2). *Let  $f \in \mathbb{F}[x_1, \dots, x_n]$  be an  $s$ -sparse, symmetric polynomial with individual degree  $d$ . Then, there is a deterministic algorithm that computes the complete factorization of  $f$  in at most  $\text{poly}(s^{d^7 \log d} \cdot n^{d^2} \cdot c_{\mathbb{F}}(d^2))$ -time.*

Throughout this thesis, we get the same results if we replace symmetric polynomials with a more general class of polynomials, which we call *symmetric-support* polynomials. We can identify the support of polynomial  $f$ , with the subset  $\text{supp}(f) \subseteq \{0, 1, \dots, d\}^n$ , as the set of exponent vectors corresponding to each monomial in  $f$ . We call  $f \in \mathbb{F}[x_1, \dots, x_n]$  a *symmetric-support* polynomial if for each monomial  $x_1^{e_1} \cdots x_n^{e_n} \in \text{supp}(f)$ , we also have that  $x_1^{e_{\sigma(1)}} \cdots x_n^{e_{\sigma(n)}} \in \text{supp}(f)$  for every permutation  $\sigma \in S_n$ . For eg.,  $f = x_1^2 x_2 x_3 + 2x_1 x_2^2 x_3 - x_1 x_2 x_3^2$  is a symmetric-support polynomial that is not symmetric. While  $f = x_1^2 x_2 x_3 + x_1 x_2^2 x_3$  is not symmetric-support.

We introduce the notion of *min-entropy* as follows. We say that an  $n$ -dimensional vector  $\mathbf{v}$  is of  $\delta$ -*min-entropy* if  $(n - \delta)$  of its coordinates have the same value, where we consider minimum  $\delta$  across all distinct elements in  $\mathbf{v}$ . We call a set of vectors  $A$ , a  $\delta$ -*min-entropy set*, if every vector in  $A$  has min-entropy  $\leq \delta$ . We call  $f$  a  $\delta$ -*min-entropy polynomial*, if  $\text{supp}(f)$  is of  $\delta$ -min-entropy. For eg.,  $f = x_1 x_2 x_3^3 x_4^5 + 2x_1^2 x_2^4 x_3^6 x_4^6 + 3x_1^7 x_2^9 x_3^8 x_4^9$  is a 2-min-entropy polynomial (also, 3-min-entropy but not 1-min-entropy). Moreover, it is not symmetric. For  $\delta$ -min-entropy polynomials, we get  $\text{poly}(n)$  factor-sparsity bound



below, when  $d, \delta$  are constants.

**Theorem** (Theorem 3.3.4). *Let  $f \in \mathbb{F}[x_1, \dots, x_n]$  be a  $\delta$ -min-entropy polynomial with individual degree  $d$  such that  $d, \delta$  are constants. Then, every factor of  $f$  has its sparsity bounded by  $(nd)^{O(d\delta)}$ .*

We get the following factoring algorithm as a direct corollary of the sparsity bound above. When  $\delta$  and  $d$  are constants, we get a  $\text{poly}(n, c_{\mathbb{F}}(d))$ -time algorithm.

**Corollary** (Corollary 3.4.3). *Let  $f \in \mathbb{F}[x_1, \dots, x_n]$  be a  $\delta$ -min-entropy polynomial with individual degree  $d$ . Then, there is a deterministic algorithm that computes the complete factorization of  $f$  in at most  $\text{poly}((nd)^{d^4\delta} \cdot c_{\mathbb{F}}(d^2))$ -time.*

All the factor-sparsity results mentioned above crucially rely on our structure theorem stated below. It shows that if we have a  $\delta$ -min-entropy set of exponent vectors, then the integral-points in its convex hull, have min-entropy at most  $O(d\delta)$ . Trivially, such a thing is false for  $\mathbb{Z}$ -linear-span (resp.  $\mathbb{Q}$ -linear-span) of vertices. Yet, surprisingly, a *convex-span* (CS) preserves the low min-entropy of its vertices!

**Theorem** (Theorem 3.1.4). *Let  $V \subseteq \{0, 1, \dots, d\}^n$  be a  $\delta$ -min-entropy set, then  $CS(V) \cap \mathbb{Z}^n$  is a  $(2d\delta)$ -min-entropy set.*

**Exact power testing.** Our next result pertains to exact powers of polynomials. A polynomial  $f \in \mathbb{F}[x_1, x_2, \dots, x_n]$  is an *exact power* if there exists (another) polynomial  $g \in \mathbb{F}[x_1, x_2, \dots, x_n]$  and  $e \in \mathbb{N}$  such that  $f = g^e$ . We note that despite the rich structure, the best known sparsity bound for exact roots of  $s$ -sparse polynomials is the general sparsity bound of size  $s^{O(d^2 \log n)}$  by [BSV20]. Hence, one can use the factorization algorithm of [BSV20] to test if a given sparse polynomial is an exact power, in quasi-polynomial time. Here we solve the decision version of this problem in polynomial time. We provide a polynomial-time algorithm for exact-power testing that **does not** rely on this sparsity bound. Moreover our algorithm *does not* reduce to univariate factorization and we give a polynomial-time algorithm in the bit-complexity of field elements. Thus, for finite fields we only get  $\text{poly}(\log |\mathbb{F}|)$  dependence instead of  $\text{poly}(|\mathbb{F}|)$  bottleneck in univariate factoring.

**Theorem** (Theorem 4.1.7). *There is a deterministic algorithm that given a sparse polynomial  $f \in \mathbb{F}[x_1, x_2, \dots, x_n]$  of individual degree  $d$  as an input, decides whether  $f = g^e$  for some polynomial  $g \in \mathbb{F}[x_1, x_2, \dots, x_n]$  and  $e \in \mathbb{N}$ , in time  $\text{poly}(s^{d^2}, n)$ .*

The above algorithm is poly-time for constant  $d$ . We note that  $s^{d/2}$  is a lower bound on the sparsity of exact roots. Consider a modification of Example 1.2.2 with  $f = (x_1 + \dots + x_n)^{p+1} = g^2$ , where  $g = (x_1 + \dots + x_n)^{(p+1)/2}$  over the finite field  $\mathbb{F}_p$ . Here  $f$  is  $O(n^2)$ -sparse while  $g$  is  $n^{\Omega(p)}$ -sparse.

**Co-factor sparsity bound.** Given two polynomials  $f, h \in \mathbb{F}[x_1, x_2, \dots, x_n]$  such that  $f = gh$ ,  $g$  is called a *quotient polynomial* or a *co-factor* of  $h$ . We study the problem of multilinear co-factor sparsity: suppose  $f$  is  $s$ -sparse and  $h$  is multilinear. How sparse/dense can  $g$  be? We remark that any (even non-constructive) efficient upper bound on the sparsity of  $g$  allows us to compute  $g$  efficiently by interpolating the ratio  $f/h$  using a reconstruction algorithm for sparse polynomials (e.g. [KS01]) and verifying the result. In literature, we encounter a decision version of this problem called *divisibility testing*. It gives two multivariate polynomials  $f$  and  $h$  and asks to decide whether  $h$  divides  $f$ . [For15] gives a quasi-polynomial time algorithm when  $f$  is sparse and  $h$  is a constant degree polynomial (and hence also sparse). Here, we consider  $f, h$  that have any constant individual degree. That is, we relax the constant total degree restriction on  $h$  to much weaker constant individual degree restriction, and put a constant individual degree restriction for  $f$ . [Vol17] gives a polynomial-time algorithm for divisibility testing for such  $f, h$ . Here, we consider a multilinear  $h$  (which can have degree  $\Omega(n)$ ) and solve the ‘search’ version of the problem itself, i.e. we actually compute  $f/h$  in quasi-polynomial time, when  $f$  is sparse with constant individual degree and  $h$  is a multilinear factor of  $f$ .

**Theorem** (Corollary 5.2.16). *Let  $f \in \mathbb{F}[x_1, x_2, \dots, x_n]$  be a polynomial of sparsity  $s$  and individual degree at most  $d$  such that  $f = gh$ . Suppose, in addition, that  $h$  is a multilinear polynomial. Then the sparsity of  $g$  is bounded by  $s^{O(d \log s)}$ .*

We beat the general sparsity bound of  $s^{O(d^2 \log n)}$  by [BSV20] when  $s = \text{poly}(n)$ . The motivation to study this problem is two-fold: first of all, a multilinear factor of an  $s$ -sparse

polynomial (of any degree) is itself  $s$ -sparse (see Lemma 5.2.14). This suggests more structure for multilinear co-factors we could potentially exploit. Second, a polynomial-size sparsity bound on multilinear co-factors  $g$  (even when the individual degree of  $g$  is  $d = 2$ ) would imply a polynomial-size sparsity bound for (**all** factors of) polynomials with individual degree  $d = 3$ . We note that the multicubic ( $d = 3$ ) case is the first instance where we do not have a polynomial-size factor-sparsity bound yet. Indeed, multilinear co-factors can be seen as the “bottle-neck” for this case (Lemma 5.1.1).

**Our main contribution here is to identify a combinatorial property – the length of the shortest unique projection – that governs the bound on the sparsity of multilinear co-factors.** We say that a polynomial  $h \in \mathbb{F}[x_1, x_2, \dots, x_n]$  has a *unique projection of length  $k$*  if there exist  $k$  variables  $x_{i_1}, x_{i_2}, \dots, x_{i_k}$  and  $k$  corresponding exponents  $e_1, e_2, \dots, e_k$  such that  $h$  has a unique monomial that contains the pattern  $x_{i_1}^{e_1} x_{i_2}^{e_2} \dots x_{i_k}^{e_k}$  (see Definition 5.2.1 for details). We show the following structural result: the shortest unique projection for an  $s$ -sparse polynomial has length at most  $\log s$ . This result is free of any restriction on individual degree.

**Lemma** (Lemma 5.2.5). *Let  $h \in \mathbb{F}[x_1, \dots, x_n]$  be an  $s$ -sparse polynomial. Then  $h$  has a unique projection of length at most  $\log s + 1$ .*

This gives the factor of  $\log s$  in our sparsity bound for multilinear co-factors. However, more generally we derive sparsity bound wrt length of the multilinear polynomial as follows.

**Theorem** (Theorem 5.2.15). *Let  $f \in \mathbb{F}[x_1, x_2, \dots, x_n]$  be a polynomial of sparsity  $s$  and individual degree at most  $d$  such that  $f = gh$ . Suppose, in addition, that  $h$  is a multilinear polynomial with a unique projection of length  $k$ . Then the sparsity of  $g$  is bounded by  $s^{O(dk)}$ .*

Example 1.2.2 has  $h$  with a unique projection of length 1. Consider  $g$  with  $d = p - 1$ . Using our result above, we derive a sparsity bound of  $n^{O(p)}$  for  $g$ . Since  $f$  is  $n$ -sparse, this shows the tightness of our sparsity bound for  $g$ . We can also extend these sparsity bounds

to the case of a co-factor of a power of a multilinear polynomial. See Theorem 5.2.19 for the formal statement.

### 1.4.2 PIT results

**PIT for  $\Sigma^{[2]}\Pi\Sigma\Pi^{[\text{ind-deg } d]}$  circuits.** Our first result here is an efficient (deterministic) identity testing algorithm for the class of  $\Sigma^{[2]}\Pi\Sigma\Pi^{[\text{ind-deg } d]}$  circuits, a circuit  $C$  of size  $s$  in this class computes a polynomial of the form:

$$C = \prod_{i=1}^r g_i + \prod_{j=1}^m h_j$$

where each polynomial ( $g_i$  and  $h_j$ ) is an  $s$ -sparse polynomial with individual degree at most  $d$  (for some fixed  $d$ ). Note, though, that  $r$  and  $m$ , and hence the total degree of  $C$ , can be arbitrary (i.e. polynomially) large. In particular, the polynomial computed by  $C$  may not itself be sparse.

Observe that the identity testing problem for this circuit class reduces to polynomial factorization of sparse polynomials with bounded individual degree. Therefore, by invoking the factorization algorithm of [BSV20], we can get a quasi-polynomial-time algorithm. Our result gives a *polynomial-time* algorithm for this model without depending on this factor-sparsity bound. In addition, our algorithm operates in the *black-box* setting, whereas the described factorization-based algorithm is a white-box algorithm.

**Theorem** (Theorem 7.3.9). *There exists a deterministic algorithm that given a black-box access to a  $\Sigma^{[2]}\Pi\Sigma\Pi^{[\text{ind-deg } d]}$  circuit  $C$  of size  $s$  determines if  $C \equiv 0$ , in time  $\text{poly}((sd)^{d^3}, n)$ .*

For constant  $d$ , the above PIT algorithm is polynomial-time. In [vzGK85], another problem was posed alongside the sparse factorization problem. This problem is referred to as *testing sparse factorization*. Given  $m + 1$  sparse polynomials  $f, g_1, \dots, g_m$ , it asks to test whether  $f = g_1 \cdot \dots \cdot g_m$ . The work of [SSS13] gives a polynomial-time algorithm for this problem, in the special case where every  $g_i$  is a sum of univariate polynomials. [Vol17] gives a polynomial-time algorithm when  $f$  (and therefore every  $g_i$ ) has constant individual degree and each  $g_i$  is an irreducible polynomial. Our PIT result is connected to this

problem. Here, we give a polynomial-time algorithm to test (even in blackbox) whether  $\prod_{i=1}^r f_i = \prod_{j=1}^m g_j$ , where each  $f_i, g_j$  is a sparse polynomial with constant individual degree. Note that now LHS is also a product of polynomials. Moreover, there is no restriction placed on  $g_j$ -s except that they have bounded individual degree.

Our algorithm relies on a structural result stated below. It links the gcd of two polynomials, their subresultant and the resultant of their coprime parts - in the **multivariate** setting. See Section 6.3 for the formal definitions.

**Theorem** (Theorem 7.2.1). *Let  $A, B \in \mathbb{F}[x_1, x_2, \dots, x_n]$  be two polynomials such that  $A = f \cdot g$  and  $B = h \cdot g$  and let  $x_i$  be a variable. Then*

$$S_{x_i}(d, A, B) = g \cdot \text{Res}_{x_i}(f, h) \cdot \text{lc}_{x_i}(g)^{m'+n'-1}$$

here  $m = \deg_{x_i}(A)$ ,  $n = \deg_{x_i}(B)$ ,  $d = \deg_{x_i}(g)$ ,  $m' = \deg_{x_i}(f) = m - d$  and  $n' = \deg_{x_i}(h) = n - d$ . In addition:

- $\text{Res}_{x_i}(f, h)$  is the resultant of  $f$  and  $h$  w.r.t the variable  $x_i$ .
- $\text{lc}_{x_i}(g)$  is the leading coefficient of  $g$  when written as a polynomial in  $x_i$
- And finally,  $S_{x_i}(d, A, B)$  is the  $d$ -th subresultant of  $A$  and  $B$ .

To put the result in context, consider two univariate polynomials  $A, B \in \mathbb{F}[x]$ . A classical result in the Theory of Resultants (see e.g. [GCL92, vzGG13, CLO15]) states that:

There exist a non-zero field element  $\alpha \in \mathbb{F}$  such that  $S(j, A, B) = \alpha \cdot \gcd(A, B)$ ,

when  $j = \deg(\gcd(A, B))$ . In the multivariate setting one can always regard multivariate polynomials as polynomials in a single variable with coefficients being rational functions in the remaining variables. Yet, in this case  $\alpha$  is no longer a mere 'field element' as it can now be an arbitrary rational function in the remaining variables! From that perspective, the above theorem can be seen as explicitly expressing  $\alpha$  as a polynomial (and not even a rational function) in the remaining variables. We believe that this explicit relation could be of interest in its own right. To elaborate on this, let us write the polynomials  $A$  and

$B$ , in the statement of theorem above, as  $A = (uf) \cdot (g/u)$  and  $B = (uh) \cdot (g/u)$ , where  $u$  is a rational function that **does not** depend on  $x_i$ . Observe that the introduction of  $u$  does not affect the degrees of  $x_i$ . We obtain the following invariant:

$$\begin{aligned} S_{x_i}(d, A, B) &= \frac{g}{u} \cdot \text{Res}_{x_i}(uf, uh) \cdot \text{lc}_{x_i}\left(\frac{g}{u}\right)^{m'+n'-1} \\ &= \frac{g}{u} \cdot \text{Res}_{x_i}(f, h) \cdot u^{m'+n'} \cdot \frac{\text{lc}_{x_i}(g)^{m'+n'-1}}{u^{m'+n'-1}} = g \cdot \text{Res}_{x_i}(f, h) \cdot \text{lc}_{x_i}(g)^{m'+n'-1}. \end{aligned}$$

**PIT for sum of ROABPs** Our next PIT result is for sum of ROABPs. We show a reduction from designing a blackbox PIT algorithm for sum of ROABPs to designing a blackbox PIT algorithm for a single ROABP.

**Theorem** (Theorem 8.3.7). *Let  $T(r, n, d)$  be the time complexity of a blackbox PIT algorithm for a single ROABP of width  $r$  and degree  $d$  in  $n$  variables over any field  $\mathbb{F}$ . Then, blackbox PIT for sum of  $c$ -many ROABPs, each of width  $r$  and degree  $d$  in  $n$  variables, can be solved in time  $T'(r, n, d, c) = (2^n \cdot T(2^c r^{3^c}, n, d))^{O(c)}$  over  $\mathbb{F}$ .*

This reduction is poly-time when number of ROABPs,  $c$  is constant and number of variables is logarithmic in the input size, that is  $n = O(\log(rd))$ . Thus, in the log-variate setting, if we have a poly-time blackbox PIT for a single ROABP, then we show a poly-time blackbox PIT for sum of  $c$  ROABPs. Though, sum of even two ROABPs is provably harder than a single ROABP (see Fact 8.1.1), we still get an efficient PIT for sum of ROABPs.

**Corollary** (Corollary 8.3.8). *Let  $\mathcal{P}$  be a set of  $n$ -variate polynomials, computed by a sum of  $c$ -many ROABPs, each of width  $r$  and degree  $d$ . Then, blackbox PIT for  $\mathcal{P}$  can be solved in  $\text{poly}(d^c, r^{nc3^c})$  time.*

If  $c, r = O(1)$  and  $n = O(\log d)$ , then input size is  $O(d)$  and the stated time-complexity is  $\text{poly}(d)$ . The trivial time complexity for blackbox PIT of  $\mathcal{P}$  is  $d^n = d^{O(\log d)}$ . [GKST17] gave a blackbox PIT algorithm for sum of  $c$  ROABPs in  $(ndr)^{O(c2^c \log(ndr))}$  time. It is super-polynomial time, even under the restrictions of constant- $c$ , constant-width and log-variate. [GKS17] gave a blackbox PIT algorithm for a single ROABP in  $O(ndr^{\log n})$  time. It is

poly-time for constant width without the log-variate restriction. However, their algorithm assumes the knowledge of variable order. Moreover, it works only for fields of characteristic either zero or larger than  $ndr^{\log n}$ . Our algorithm is efficient in the log-variate setting, does not require knowledge of variable order and works for all fields.

For some  $k \in \mathbb{N}$ , let  $f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_k(\mathbf{x})$  be any  $k$  polynomials in  $\mathbb{F}[x_1, \dots, x_n]$ . We call  $\sum_{i=1}^k f_i(\mathbf{x})$ , a *deg-preserving sum*, if for  $f(\mathbf{x}) = \sum_{i=1}^k f_i(\mathbf{x})$ , we have  $\deg(f) = \max_i \deg(f_i)$ . In the previous PIT algorithm number of ROABPs  $c$ , is assumed to be constant to get efficient blackbox PIT. We could allow an arbitrary  $c$ , if the sum is deg-preserving. In other words, with the additional restriction of a deg-preserving sum, we bring down the double exponential dependence on  $c$  to polynomial dependence on  $c$  in the theorem below.

**Theorem** (Theorem 8.2.9). *Let  $\mathcal{P}$  be a set of  $n$ -variate polynomials computed by a degree-preserving sum of  $c$  ROABPs, each of width  $r$  and degree  $d$ . Then, blackbox PIT for  $\mathcal{P}$  can be solved in  $\text{poly}(d, c \cdot r^n)$  time.*

If  $r = O(1)$  and  $n = O(\log d)$ , then the stated time complexity is  $\text{poly}(cd)$ – polynomial in the input-size. Consider the class of polynomials which can be computed by a sum of  $c$  ROABPs, where each ROABP computes a homogeneous polynomial. In Section 8.2, we show that such a sum can be expressed as a deg-preserving sum. Thus, we get a blackbox PIT for this class in the same time. If we could get a  $\text{poly}(d, cr^n)$  time PIT in the above theorem without the deg-preserving sum restriction, then we get poly-time PIT for the model of  $\Sigma \wedge \Sigma$  circuits (See Lemma 6.4.5).

All the PIT algorithms above rely on efficient PIT for a single log-variate ROABP of constant width. We indeed prove one in Lemma 8.2.8. This PIT in turn relies on our structural result which shows that an ROABP computing a homogeneous polynomial can be syntactically homogenized in exactly the same width. See Section 8.2.1 for definition of syntactic homogeneity.

**Theorem** (Theorem 8.2.4). *Let  $f(\mathbf{x}) \in \mathbb{F}[\mathbf{x}]$  be a degree  $d$  homogeneous polynomial computed by an ROABP  $C(\mathbf{x})$  of width  $w$  in the variable order  $(y_1, \dots, y_n)$ . Then,  $f$  also has*

a syntactically homogeneous ROABP  $D(\mathbf{x}) = \prod_{i=1}^n D_i(y_i)$  of optimal width  $r \leq w$  in the same variable order. Moreover,  $\forall i \in [n]$ , each entry in  $D_i(y_i)$  is merely a monomial in  $y_i$ .

If we use [AGKS15]’s blackbox PIT algorithm for single ROABP in our PIT reduction, we get another efficient PIT for sum of ROABPs as a corollary below.

**Corollary** (Corollary 8.3.9). *Let  $\mathcal{P}$  be a set of  $n$ -variate polynomials, over a field  $\mathbb{F}$ , computed by a sum of  $c$ -many ROABPs, each of width  $r$  and degree  $d$ . Also, the variable order of each ROABP is unknown. Then, blackbox PIT for  $\mathcal{P}$  can be solved in  $\text{poly}(2^{cn} \cdot n^{c \log n}, d^{c \log n}, r^{3^c \log n})$  time.*

For  $c = O(1)$  and  $n = O(\log(rd))$ , the stated time complexity is  $(rd)^{O(\log \log(rd))}$ . Thus, in the log-variate setting, we give a more efficient PIT than the result of [GKST17], which only yields a  $\text{poly}(rd)^{O(\log(rd))}$  time algorithm, even with  $n = O(\log(rd))$ .

**PIT for border of sum of ROABPs.** Let  $\mathcal{C}$  be an algebraic class over field  $\mathbb{F}$ , like arithmetic circuit or ABP or ROABP. An *approximation closure* or *border* of class  $\mathcal{C}$ , denoted as  $\overline{\mathcal{C}}$  is defined as follows: a family  $(f_n)$  is in  $\overline{\mathcal{C}}$  if there are polynomials  $f_{n,1}, \dots, f_{n,t} \in \mathbb{F}[\mathbf{x}]$  such that the family  $(g_n)$  defined by

$$g_n(\mathbf{x}, \epsilon) := f_n(\mathbf{x}) + \epsilon f_{n,1}(\mathbf{x}) + \epsilon^2 f_{n,2}(\mathbf{x}) + \dots + \epsilon^t f_{n,t}(\mathbf{x})$$

is in  $\mathcal{C}$  over the field  $\mathbb{F}(\epsilon)$ , where  $t$  is called the error degree. Here  $\epsilon$  is a new indeterminate and  $\lim_{\epsilon \rightarrow 0} g_n(\mathbf{x}, \epsilon) = f_n(\mathbf{x})$ . In other words,  $f_n$  is approximated by a polynomial  $g_n$  which has a circuit in class  $\mathcal{C}$  over  $\mathbb{F}(\epsilon)$ . Although the circuit  $C \in \mathcal{C}$  computing  $g_n$  might involve internal computations with  $\epsilon$  in the denominator but the final output does not and is a polynomial over  $\mathbb{F}[\epsilon][\mathbf{x}]$ . Border classes can be more complicated because the degree of  $\epsilon$  involved can be super-polynomial. So, poly-time PIT algorithms for border classes are rare. Even though single ROABPs are known to be closed under border (see Lemma 6.5.1), it is not the case with sum of ROABPs. Below, we give a reduction from blackbox PIT for the border class of sum of  $c$  ROABPs to blackbox PIT for a single ROABP.

**Theorem** (Theorem 8.4.1). *Let  $T(r, n, d)$  be the time complexity of a blackbox PIT algorithm for a single ROABP of width  $r$  in  $n$  variables and degree  $d$  over any field  $\mathbb{F}$ . Then*



*blackbox PIT for border of sum of  $c$  ROABPs, each of width  $r$  and degree  $d$  in  $n$  variables, can be solved in time  $(2^n \cdot T(2^c r^{3^c}, n, d))^{O(c)}$  over  $\mathbb{F}$ .*

This reduction yields PIT algorithms for the border class of sum of ROABPs analogous to Corollary 8.3.8 and Corollary 8.3.9 above.

## 1.5 Organization of the thesis

We divide this thesis into two parts. Part-I deals with sparse polynomial factorization and Part-II deals with PIT results. Both the parts have their own preliminaries covered in chapters 2 & 6 respectively. Part-I covers Chapters 3,4 & 5 while Part-II covers Chapters 7 & 8. We discuss the sparsity bound for sparse, symmetric polynomials in Chapter 3. Chapter 4 deals with the exact power testing and Chapter 5 deals with sparsity bounds for cofactors of multilinear polynomials. In PIT, we start with PIT for  $\Sigma^{[2]}\Pi\Sigma\Pi^{[\text{ind-deg } d]}$  circuits in Chapter 7 and then move on to sum of ROABPs in Chapter 8. Each chapter ends with a discussion, where we also mention future directions for the problem discussed there. Finally, we conclude this thesis in Chapter 9.



## Part I

# Sparse Polynomial Factorization



## Chapter 2

# Factoring Preliminaries

### 2.1 Notations

We use shorthand  $[n]$  for the set  $\{1, 2, \dots, n\}$ . We denote a vector  $v = (v_1, \dots, v_n)$  in short by  $\mathbf{v}$  (as a column vector). We will use the terms vector or *point* interchangeably. We denote the  $n$ -fold Cartesian product of a set  $H$  by  $H^n$ . We will use  $\log x$  for  $\log_2 x$  and  $\ln x$  for  $\log_e x$ . We use the  $:=$  symbol for defining things. We will sometimes use  $\mathbb{F}[\mathbf{x}]$  as short for  $\mathbb{F}[x_1, \dots, x_n]$ . The finite *symmetric group* on  $n$  elements, which contains all the permutations of  $n$  elements, is denoted by  $S_n$ . For a vector  $\mathbf{v}$  and a permutation  $\sigma \in S_n$ , we denote  $\sigma$ -permutation of  $\mathbf{v}$  by  $\sigma \circ \mathbf{v} := (v_{\sigma(1)}, \dots, v_{\sigma(n)})$ . We call a set of points symmetric, if for each point  $\mathbf{v}$  in it,  $\sigma \circ \mathbf{v}$  is also in the set, for every permutation  $\sigma \in S_n$ . We often use the short-hand wrt to mean “with respect to”.

Let  $f \in \mathbb{F}[\mathbf{x}]$  be an  $n$ -variate polynomial. Individual degree of a variable  $x_i$ , denoted by  $\deg_{x_i}(f)$  is defined as the maximum degree of that variable in  $f$ , while *individual degree* of a polynomial is the maximum among all the individual degrees,  $\max_{i \in [n]} \deg_{x_i}(f)$ . We will use  $\mathbf{x}^{\mathbf{e}}$  to denote the monomial  $x_1^{e_1} x_2^{e_2} \cdots x_n^{e_n}$ . We define  $\text{coeff}(\mathbf{x}^{\mathbf{e}})(f)$  as the coefficient of monomial  $\mathbf{x}^{\mathbf{e}}$  in polynomial  $f$ . We define *support* of  $f$  as  $\text{supp}(f) = \{\mathbf{e} \mid \text{coeff}(\mathbf{x}^{\mathbf{e}})(f) \neq 0\}$ . Let us denote *sparsity* of  $f$  as  $\|f\|$ , which is the same as  $|\text{supp}(f)|$ .

For a set  $I \subseteq [n]$ , we use  $x_I$  to denote the set of variables  $\{x_i \mid i \in I\}$  and  $x_{[n] \setminus I}$  to denote the set of remaining variables. We use the symbol  $f|_{x_I=0_I}$  to denote the polynomial

resulting from substituting 0 at all the  $x_I$  variables in  $f$ . For two polynomials  $g, h$ , we use the symbol  $\gcd(g, h)$  to denote their greatest common divisor.

## 2.2 Definitions

**Polytopes:** For a finite set of points  $\mathbf{v}_1, \dots, \mathbf{v}_k \in \mathbb{R}^n$ , their *convex combination* is defined as an  $\mathbb{R}$ -linear combination of the points:  $\alpha_1 \mathbf{v}_1 + \dots + \alpha_k \mathbf{v}_k$ , such that  $\alpha_i \geq 0$  for each  $i \in [k]$  and  $\sum_{i=1}^k \alpha_i = 1$ . We define *convex-span* (or convex hull)  $CS(\mathbf{v}_1, \dots, \mathbf{v}_k)$  as the set of all possible convex combinations of  $\mathbf{v}_i$ ,  $i \in [k]$ . A set  $P \subseteq \mathbb{R}^n$  is called a (bounded) *polytope* if there is a finite set of points  $\mathbf{v}_1, \dots, \mathbf{v}_k$  such that  $P = CS(\mathbf{v}_1, \dots, \mathbf{v}_k)$ . A point  $\mathbf{a} \in P$  is called a *vertex* of  $P$  if it cannot be written as  $\mathbf{a} = \alpha \mathbf{u} + (1 - \alpha) \mathbf{v}$  for any  $\mathbf{u}, \mathbf{v} \in P \setminus \{\mathbf{a}\}$  and  $\alpha \in [0, 1]$ . It is equivalent to saying that vertices are *corner* points of a polytope  $P$  which cannot be expressed as convex combination of any other set of points in  $P$ . We use  $V(P)$  to denote the set of vertices of  $P$ . It is easy to verify that for a polytope  $P$ ,  $P = CS(V(P))$ . Moreover, if  $P = CS(\mathbf{v}_1, \dots, \mathbf{v}_k)$  then  $V(P) \subseteq \{\mathbf{v}_1, \dots, \mathbf{v}_k\}$ .

*Minkowski sum* of two polytopes  $A, B \in \mathbb{R}^n$  is defined as the following set of points  $A + B = \{\mathbf{a} + \mathbf{b} \mid \mathbf{a} \in A, \mathbf{b} \in B\}$ . A basic fact is that Minkowski sum of two polytopes is itself a polytope. The vertices of Minkowski sum have some very useful properties. It is known that every vertex of  $A + B$  can be expressed *uniquely* as a sum  $\mathbf{u} + \mathbf{v}$ , where  $\mathbf{u}$  is a vertex of  $A$  and  $\mathbf{v}$  is a vertex of  $B$ . Additionally, one can show that  $|V(A + B)| \geq \max\{|V(A)|, |V(B)|\}$  (See [BSV20, Prop. 3.2] or [DdO14, Cor. 3.13]). We refer the readers to [Zie12, Sch00] for a detailed discussion on polytopes.

For a polynomial  $f \in \mathbb{F}[x_1, \dots, x_n]$ , the *Newton polytope* of  $f$  is defined as  $P_f := CS(\text{supp}(f))$ . In this work, we crucially exploit its *integral-points*  $P'_f := P_f \cap \mathbb{Z}^n$ . We denote the vertex set  $V(P_f)$  by  $V_f$ . We also note that  $V_f \subseteq \text{supp}(f) \subseteq \{0, \dots, d\}^n$  (integral-points).

**Hyperplanes and Halfspaces:** A hyperplane is the generalization of a line in higher dimensions. A hyperplane  $H$  is defined as  $H := \{\mathbf{y} \in \mathbb{R}^n \mid \mathbf{a}^\top \mathbf{y} = b\}$ , for some vector  $\mathbf{a} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$  and a number  $b \in \mathbb{R}$ . The hyperplane divides the space into two parts

$\mathbf{a}^\top \mathbf{y} \geq b$  and  $\mathbf{a}^\top \mathbf{y} \leq b$ . These are called halfspaces.

### Min-entropy:

**Definition 2.2.1** ( $\delta$ -min-entropy). A vector  $\mathbf{v} \in \{0, 1, \dots, d\}^n$  has  $\delta$ -min-entropy if a single value  $c \in \{0, \dots, d\}$  appears in exactly  $(n - \delta)$  coordinates of  $\mathbf{v}$ . We consider the  $c$  appearing with maximum frequency in  $\mathbf{v}$  and hence the smallest  $\delta$ . In coding theory language, a  $\delta$ -min-entropy vector has Hamming distance  $\delta$  from a constant vector  $(c, \dots, c)$  for some  $c \in \{0, 1, \dots, d\}$ . We also extend this definition of min-entropy to sets and polynomials. We call  $A \subseteq \{0, 1, \dots, d\}^n$  a  $\delta$ -min-entropy set, if for every vector  $\mathbf{v} \in A$ ,  $\mathbf{v}$  has min-entropy  $\leq \delta$ . We call  $f$  a  $\delta$ -min-entropy polynomial, if its support set  $\text{supp}(f)$  is a  $\delta$ -min-entropy set.

**Example 2.2.2.** The polynomial  $f = x_1^d + x_2^d + \dots + x_n^d$  has 1-min-entropy as

$$\text{supp}(f) = \left\{ \begin{bmatrix} d \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ d \\ \vdots \\ 0 \end{bmatrix}, \dots, \begin{bmatrix} 0 \\ 0 \\ \vdots \\ d \end{bmatrix} \right\},$$

where each vector in it has element 0 occurring with frequency  $(n - 1)$ .

**Example 2.2.3.** Let us now consider a polynomial of high min-entropy. The polynomial  $f = x_1 \cdots x_{n/3} + x_{n/3+1} \cdots x_n$ , has min-entropy  $n/3$ . To clarify the notation, we can also call  $f$  to be of  $(n/3 + 1)$ -min-entropy but we cannot call it  $(< n/3)$ -min-entropy vector.

### 2.2.1 Content and Primitive parts

We use the short-hand UFD to denote a unique factorization domain. For polynomials over UFDs and more generally over their field of fractions, we define content and primitive parts as follows.

**Definition 2.2.4** (Chapter 6 [vzGG13]). Let  $R$  be a unique factorization domain and  $K$  be its field of fractions. Let  $g \in K[y]$  be a polynomial over  $K$  such that  $g = \sum_{i=0}^m (g_i/b) \cdot y^i \in K[y]$ , where  $b \in R$  is the common denominator. The content of  $g$  is defined as

$\text{cont}(g) = \gcd(g_0, \dots, g_m)/b$ . We define primitive part of  $g$  as  $\text{pp}(g) = g/\text{cont}(g)$ . Observe that  $\text{cont}(g) \in K$ , while  $\text{pp}(g) \in R[y]$ .

**Example 2.2.5.** For  $R = \mathbb{Z}$  and  $K = \mathbb{Q}$ , consider  $g = 3y+9/2 \in \mathbb{Q}[y]$ . Then  $g = (6y+9)/2$  and  $\text{cont}(g) = \gcd(6, 9)/2 = 3/2 \in \mathbb{Q}$ . And  $\text{pp}(g) = g/\text{cont}(g) = 2y + 3 \in \mathbb{Z}[y]$ .

**Example 2.2.6.** Let us now consider a bi-variate example. Let  $R = \mathbb{F}[x]$  and  $K = \mathbb{F}(x)$ . Consider  $g = (x^2 - 1) \cdot y + (x - 1)/(x + 1) \in \mathbb{F}(x)[y]$ . Then,  $\text{cont}(g) = \gcd((x^2 - 1)(x + 1), (x - 1))/(x + 1) = (x - 1)/(x + 1) \in \mathbb{F}(x)$ . And  $\text{pp}(g) = (x + 1)^2 \cdot y + 1 \in \mathbb{F}[x][y]$ .

We use the above definitions to prove the following lemma which will be useful to us later on.

**Lemma 2.2.7.** *Let  $R$  be a unique factorization domain and  $K$  be its field of fractions. Let  $f \in R[y]$  and  $g \in K[y]$  such that  $f = g^e$ . Then,  $g \in R[y]$ .*

*Proof.* By definitions of content and primitive parts in Definition 2.2.4, we know that  $\text{cont}(g) \in K$ , while  $\text{pp}(g) \in R[y]$  for  $g = \text{cont}(g) \cdot \text{pp}(g)$ .

Gauss's Lemma states that the product of two primitive polynomials is also primitive. From this, one can derive that for two polynomials  $g, h \in K[y]$ ,  $\text{cont}(gh) = \text{cont}(g) \cdot \text{cont}(h)$  and  $\text{pp}(gh) = \text{pp}(g) \cdot \text{pp}(h)$ . In particular,  $\text{cont}(g^e) = \text{cont}(g)^e$  and  $\text{pp}(g^e) = \text{pp}(g)^e$ . Since  $f = g^e$ , we get that

$$\text{cont}(f) = \text{cont}(g)^e. \quad (2.1)$$

Since  $f \in R[y]$ , we know that  $\text{cont}(f) \in R$  by definition. We will now use this to prove that  $\text{cont}(g) \in R$  also. This will suffice to prove that  $g = \text{cont}(g) \cdot \text{pp}(g) \in R[y]$ . Note that we can write  $\text{cont}(g)$  in the simplest form as  $\text{cont}(g) = \frac{a}{b}$ , where  $a, b \in R$  and  $\gcd(a, b) = 1$ . Let  $d := \text{cont}(f) \in R$ . Using (2.1), we get that  $d = \left(\frac{a}{b}\right)^e = \frac{a^e}{b^e}$ . Now, let  $p$  be an irreducible factor of  $b$  in  $R$ . Since  $d \in R$ ,  $p$  must divide the numerator  $a^e$ . If  $p$  divides  $a^e$ , then  $p$  must divide  $a$  also. This contradicts with the fact that  $\gcd(a, b) = 1$ . This means, that the denominator  $b$  must be one and hence,  $\text{cont}(g) \in R$ . Thus,  $g \in R[y]$ .  $\square$



### 2.2.2 Multivariate Reversal operation

Let  $f \in \mathbb{F}[x_1, \dots, x_n]$ . For some  $i \in [n]$ , let  $x_i$  be a variable in the support of  $f$ . Let  $f = \sum_{j=0}^d f_j \cdot x_i^j$  such that  $\forall j$ ,  $f_j$  is a polynomial in rest of the variables and  $f_d \neq 0$ . The leading coefficient of  $f$  wrt  $x_i$  is defined as  $\text{lc}_{x_i}(f) := f_d$ . Polynomial  $f$  is called monic wrt variable  $x_i$ , if  $\text{lc}_{x_i}(f) = 1$ .

**Definition 2.2.8** (Reverse Monic, Reverse Pseudo-Monic). *We say that a polynomial  $f \in \mathbb{F}[x_1, \dots, x_n]$  is reverse monic if there exists a variable  $x_i \in \text{supp}(f)$  such that  $f|_{x_i=0} = 1$ . If we know the variable beforehand, we say that  $f$  is  $x_i$ -reverse monic. We can extend this definition to a set of variables  $x_I$ , for some arbitrary  $I \subseteq [n]$ . We say that  $f$  is  $I$ -reverse monic, if  $f|_{x_I=0_I} = 1$ . We say that  $f$  is reverse pseudo-monic,  $x_i$ -reverse pseudo-monic,  $I$ -reverse pseudo-monic respectively, when instead of 1 the result of setting variables to 0 is a non-zero field element or a single monomial.*

In other words, the constant term of a reverse monic polynomial is 1, when regarded as a polynomial in the remaining variables. The following are immediate connections between some of the previously defined concepts.

**Observation 2.2.9.** Let  $h \in \mathbb{F}[x_1, x_2, \dots, x_n]$  be a polynomial,  $i \in [n]$  and  $I \subseteq [n]$ . Then:

- $\text{supp}(h|_{x_i=0}) = \{\mathbf{e} \in \text{supp}(h) \mid e_i = 0\}$ .
- $h$  is  $I$ -reverse pseudo-monic if and only if  $|\text{supp}(h|_{x_I=0_I})| = 1$ .

The following transformation will be useful later on to convert specific polynomials into  $I$ -reverse pseudo-monic polynomials. It is a generalization of the standard reversal operation for univariate polynomials.

**Definition 2.2.10** (Reversal Transformation). *Let  $f \in \mathbb{F}[x_1, \dots, x_n]$  be a polynomial and let  $\ell \in \mathbb{N}$ . We define the reversal operation on  $f$  with respect to a variable  $x_i$  as follows:*

$$\text{rev}_i^\ell[f] := x_i^\ell \cdot f|_{x_i=\frac{1}{x_i}} = x_i^\ell \cdot f(x_1, \dots, x_{i-1}, \frac{1}{x_i}, x_{i+1}, \dots, x_n).$$

*By iteration, we can extend this definition to a set of variables  $x_I$ , for some arbitrary*

$$I = \{i_1, \dots, i_r\} \subseteq [n].$$

$$\text{rev}_I^\ell[f] := \text{rev}_{i_1}^\ell \left[ \text{rev}_{i_2}^\ell [\dots \text{rev}_{i_r}^\ell [f]] \right].$$

*Alternatively:*

$$\begin{aligned} \text{rev}_I^\ell[f] &:= x_{i_1}^\ell \cdots x_{i_r}^\ell \cdot f(y_1, \dots, y_n), \\ \text{where } y_j &= \begin{cases} \frac{1}{x_j}, & \text{if } x_j \in I \\ x_j, & \text{if } x_j \notin I. \end{cases} \end{aligned}$$

For intuition, express  $f$  as a polynomial in  $x_i$  such that  $f = f_d x_i^d + f_{d-1} x_i^{d-1} + \dots + f_1 x_i + f_0$ , where each coefficient  $f_j$  is a polynomial in variables other than  $x_i$ . Then,  $\text{rev}_i^d[f]$  reverses the order of coefficients in this representation. That is,  $\text{rev}_i^d[f] = f_0 x_i^d + f_1 x_i^{d-1} + \dots + f_{d-1} x_i + f_d$ . In particular, if  $f$  is monic in  $x_i$  then  $\text{rev}_i^d[f]$  is  $x_i$ -reverse monic.

The following lemma summarizes some of the basic, yet useful properties of the reversal transformation. Subsequently, we will use these properties implicitly.

**Lemma 2.2.11.** *Let  $f, g, h \in \mathbb{F}[x_1, x_2, \dots, x_n]$  such that  $f = g \cdot h$ . Let  $i \in [n]$  and suppose that  $d \geq \deg_{x_i}(f)$ . Then:*

1.  $\text{rev}_i^d[f]$  is a polynomial (and not a rational function).
2.  $\deg_{x_i}(\text{rev}_i^d[f]) \leq d$ .
3.  $\|\text{rev}_i^d[f]\| = \|f\|$ .
4. Let  $a, b$  such that  $d = a + b$ . Then  $\text{rev}_i^d[f] = \text{rev}_i^a[g] \cdot \text{rev}_i^b[h]$ .

## 2.3 The sparsity connection with Newton polytopes

For a polynomial  $f \in \mathbb{F}[x_1, x_2, \dots, x_n]$ , we define the *Newton polytope* of  $f$ , denoted by  $P_f$ , as the convex hull (or convex-span) of all the points in  $\text{supp}(f)$ . Recall the definition of Minkowski sum of two polytopes. Minkowski sum is well studied in polytope literature and

comes with some nice properties. The Minkowski sum  $A + B$  is itself a convex polytope. Let  $V(P)$  denote the set of *vertices* (corner points) of a polytope  $P$ , then one can show a certain ‘incompressibility’ property:  $|V(A + B)| \geq \max\{|V(A)|, |V(B)|\}$ . See the exposition in [BSV20, Prop. 3.2] or [DdO14, Cor. 3.13] for a proof of this.

The following classical fact about Minkowski sum of Newton polytopes, was first observed by [Ost21] a century ago.

**Proposition 2.3.1** ([Ost21]). *Let  $f, g, h \in \mathbb{F}[x_1, \dots, x_n]$  be polynomials such that  $f = g \cdot h$ . Then*

$$P_f = P_g + P_h.$$

Further, we know that  $\|f\| \geq |V_f|$ , since  $\text{supp}(f)$  is in the convex hull of  $V_f$ . Therefore, we are able to connect sparsity of  $f$  with its factors as follows:

**Proposition 2.3.2** ([BSV20, DdO14]). *Let  $f, g, h \in \mathbb{F}[x_1, \dots, x_n]$  be polynomials such that  $f = g \cdot h$ . Then*

$$\|f\| \geq |V_f| \geq \max\{|V_g|, |V_h|\}. \quad (2.2)$$

Another way to think about sparsity bound problem for  $f = g \cdot h$  is to determine how many monomials of  $g$  survive on multiplication with  $h$ . Equation (2.2) tells us that at least  $|V_g|$  many monomials survive. These vertices of  $P_g$  are in a sense *extremal* monomials in  $g$  that never get canceled on multiplication (with any  $h$ ).

## 2.4 Symmetric Polytopes

In this section, we discuss definitions and few properties of symmetric-support polynomials and symmetric polytopes.

**Definition 2.4.1** (Symmetric-support polynomials). *We call  $f \in \mathbb{F}[x_1, \dots, x_n]$  a symmetric polynomial if  $f(x_1, \dots, x_n) = f(x_{\sigma(1)}, \dots, x_{\sigma(n)})$ , for any permutation  $\sigma \in S_n$ . It is equivalent to saying:  $f$  is symmetric if and only if  $\text{coeff}(x_1^{e_1} \cdots x_n^{e_n})(f) = \text{coeff}(x_1^{e_{\sigma(1)}} \cdots x_n^{e_{\sigma(n)}})(f)$ , for all  $\sigma \in S_n$ . We call  $f \in \mathbb{F}[x_1, \dots, x_n]$  a symmetric-support polynomial if for each monomial  $x_1^{e_1} \cdots x_n^{e_n}$ , we have  $\text{coeff}(x_1^{e_1} \cdots x_n^{e_n})(f) \neq 0 \Rightarrow \text{coeff}(x_1^{e_{\sigma(1)}} \cdots x_n^{e_{\sigma(n)}})(f) \neq 0$ , for*

every  $\sigma \in S_n$ . Note that all symmetric polynomials are also symmetric-support polynomials.

For example,  $f = x_1^2 x_2 x_3 + x_1 x_2^2 x_3 + x_1 x_2 x_3^2$  is a symmetric polynomial and hence, also symmetric-support. In contrast,  $f = x_1^2 x_2 x_3 + 2x_1 x_2^2 x_3 - x_1 x_2 x_3^2$  is a symmetric-support polynomial that is not symmetric. While  $f = x_1^2 x_2 x_3 + x_1 x_2^2 x_3$  is not even symmetric-support.

**Definition 2.4.2.** *We say that a polytope  $P$  is symmetric if for each point  $(v_1, \dots, v_n)$  in  $P$ ,  $(v_{\sigma(1)}, \dots, v_{\sigma(n)})$  is also in  $P$ , for every permutation  $\sigma \in S_n$ .*

We now observe that Newton polytopes of symmetric-support polynomials are in fact symmetric polytopes!

**Lemma 2.4.3** (Symmetric polynomials  $\Rightarrow$  symmetric polytopes). *Let  $f$  be a symmetric-support polynomial. Then the Newton polytope  $P_f$  of  $f$  is also symmetric.*

*Proof.* Let  $\text{supp}(f) = \{\mathbf{v}_1, \dots, \mathbf{v}_k\}$ . Let  $\mathbf{v} \in P_f$  be an arbitrary point. We need to show that  $\sigma \circ \mathbf{v} \in P_f$ , for every  $\sigma \in S_n$ . Since  $P_f = CS(\text{supp}(f))$ , we can express  $\mathbf{v}$  as:

$$\mathbf{v} = \sum_{i=1}^k \alpha_i \cdot \mathbf{v}_i.$$

Then for an arbitrary  $\sigma \in S_n$ , observe that:

$$\sigma \circ \mathbf{v} = \sum_{i=1}^k \alpha_i \cdot \sigma \circ \mathbf{v}_i.$$

Now, since  $f$  is a symmetric-support polynomial,  $\sigma \circ \mathbf{v}_i$  is also in  $\text{supp}(f)$ , for each  $i \in [k]$ .

The above equation then implies that  $\sigma \circ \mathbf{v} \in CS(\text{supp}(f))$ , which means that  $\sigma \circ \mathbf{v} \in P_f$ .  $\square$

It turns out that symmetric polytopes have a nice property: a point is a vertex in a symmetric polytope  $P$  if and only if all its  $S_n$  permutations are also vertices in  $P$ .

**Lemma 2.4.4** (Vertices of symmetric polytope). *Let  $P$  be a symmetric polytope with  $V(P)$  as its vertex set. Then, for any  $\sigma \in S_n$ :  $\mathbf{v} \in V(P)$  if and only if  $\sigma \circ \mathbf{v} \in V(P)$ .*

*Proof.* ( $\Rightarrow$ ) Suppose  $V(P) := \{\mathbf{v}_1, \dots, \mathbf{v}_k\}$ . Let  $\mathbf{v} \in V(P)$ . Consider any permutation  $\sigma \in S_n$ . Since  $P$  is a symmetric polytope, we at least know that  $\sigma \circ \mathbf{v} \in P$ . For the sake of contradiction, suppose  $\sigma \circ \mathbf{v} \notin V(P)$ . Thus, it is an internal point which can be expressed as a nontrivial convex combination of vertices. There exist, for  $i \in [k]$ ,  $0 \leq \alpha_i < 1$ ,  $\sum_{i=1}^k \alpha_i = 1$  such that:

$$\begin{aligned}\sigma \circ \mathbf{v} &= \alpha_1 \cdot \mathbf{v}_1 + \dots + \alpha_k \cdot \mathbf{v}_k \\ \sigma^{-1} \circ (\sigma \circ \mathbf{v}) &= \sigma^{-1} \circ (\alpha_1 \cdot \mathbf{v}_1 + \dots + \alpha_k \cdot \mathbf{v}_k) \\ \mathbf{v} &= \alpha_1 \cdot (\sigma^{-1} \circ \mathbf{v}_1) + \dots + \alpha_k \cdot (\sigma^{-1} \circ \mathbf{v}_k).\end{aligned}$$

Observe that  $\sigma^{-1} \in S_n$  and since  $P$  is symmetric  $\sigma^{-1} \circ \mathbf{v}_i \in P$  for all  $i \in [k]$ . This means,  $\mathbf{v}$  is a nontrivial convex combination of other points in  $P$ , which contradicts the fact that  $\mathbf{v}$  is a vertex. Therefore,  $\sigma \circ \mathbf{v}$  must also be a vertex.

( $\Leftarrow$ ) Now we wish to prove that all permutations of a non-vertex point must also be non-vertices. Let  $\mathbf{v} \notin V(P)$ . Then, for any  $\sigma \in S_n$ , we get a nontrivial convex combination:

$$\begin{aligned}\mathbf{v} &= \alpha_1 \cdot \mathbf{v}_1 + \dots + \alpha_k \cdot \mathbf{v}_k \\ \sigma \circ \mathbf{v} &= \alpha_1 \cdot \sigma \circ \mathbf{v}_1 + \dots + \alpha_k \cdot \sigma \circ \mathbf{v}_k\end{aligned}$$

Again, since  $P$  is symmetric  $\sigma \circ \mathbf{v}_i \in P$  for all  $i \in [k]$ . Thus,  $\sigma \circ \mathbf{v}$  is a nontrivial convex combination of other points, hence it must be a non-vertex.  $\square$



## Chapter 3

# Sparse Factoring via Symmetric Polytopes

In this chapter, we will show how to deterministically factor sparse polynomials that are symmetric. A polynomial is called symmetric if it is invariant under the permutation of variables. The work of Bhargava et al [BSV20] showed that the problem of sparse factoring reduces to proving factor-sparsity bounds (See Lemma 3.4.1). In this chapter, we shall prove that  $s$ -sparse symmetric polynomials with individual degree  $d$  have factors of at most  $s^{O(d^2 \log d)}$  sparsity and use this bound to get  $\text{poly}(s)$ -time deterministic factoring algorithm for such polynomials, when  $d$  is constant. As discussed earlier also, the restriction on individual degree being constant is necessary as there are  $s$ -sparse symmetric polynomials having factors of sparsity  $s^{\log d}$  in characteristic zero fields and  $s^d$  in finite fields (See examples 1.2.1 and 1.2.2 respectively). We use the polytope connection with factor-sparsity outlined in Section 2.3 by showing that Newton polytopes of symmetric polynomials have high number of vertices. In other words, we show that the gap between number of vertices and total number of integral points in such symmetric polytopes is low. After developing the necessary tools, we prove our factor-sparsity bound of  $s^{O(d^2 \log d)}$  formally in Theorem 3.3.3. To achieve this bound, we define a new notion of *min-entropy* for a set of vectors and our underlying structural result (Theorem 3.1.4) analyzes min-entropy for the convex span of a given set. We also remark that all the results in this chapter hold

for any field  $\mathbb{F}$ , finite or otherwise.

### 3.1 Polytope Entropy

In this chapter, we shall be mainly interested in vectors that have a single coordinate of high frequency. We say that such vectors have low *min-entropy*. Recall the definition of min-entropy in Definition 2.2.1. The main result of this section is: *Convex span preserves low min-entropy*. Formally, if we have a  $\delta$ -min-entropy set of vectors/points from  $\{0, 1, \dots, d\}^n$ , then every integral point in its convex span has min-entropy  $\leq O(d\delta)$ . This structural observation stated in Theorem 3.1.4 will act as a foundation for all the main results in this chapter.

The driving intuition is that internal points in a convex span must derive some nice properties from the vertices as they can be expressed as their convex combination. In this case, the nice property will be low min-entropy. In order to prove this, we first design a set of symmetric hyperplane equations such that for each hyperplane, the entire  $\delta$ -min-entropy set  $V$  lies on one side of its halfspace (Lemma 3.1.1). Thus, every point in convex-span of  $V$  must also lie on the same side (Observation 3.1.2). Secondly, the hyperplane equations are so designed that any *integral*-point lying on that side must have min-entropy at most  $O(d\delta)$  (Lemma 3.1.3). This design of the hyperplane equations is the novelty of this chapter. We will apply Theorem 3.1.4 in obtaining efficient factor-sparsity bounds later in this chapter. However, we believe that it could be of independent interest also in convex geometry.

Let  $P_f$  be the Newton polytope of  $f$  and  $V_f$  be its set of vertices such that  $P_f = CS(V_f)$ . We prove few claims below which will help us prove Theorem 3.1.4. We are given  $V_f$  to be a  $\delta$ -min-entropy set. Let  $m := n - \delta$ . We first design a hyperplane  $\mathbf{u}^\top \cdot \mathbf{y} + d\delta = 0$  that will help us prove very useful properties later. Define vector  $\mathbf{u} \in \{-1, +1\}^n$  such that,

$$u_i := \begin{cases} +1 & \text{if } i \leq \delta + m/2 \\ -1 & \text{otherwise.} \end{cases} \quad (3.1)$$



The first  $\delta + m/2$  coordinates of  $\mathbf{u}$  are  $+1$  and the remaining  $n - (\delta + m/2) = m/2$  coordinates are  $-1$ .

**Lemma 3.1.1** (Hyperplane cover). *Let  $V \subseteq \{0, 1, \dots, d\}^n$  be a  $\delta$ -min-entropy set and  $\mathbf{u}$  be as defined in (3.1). Let  $S_n$  be the finite symmetric group. Then, every point  $\mathbf{y} \in V$  satisfies each of the following inequalities:*

$$(\sigma \circ \mathbf{u})^\top \cdot \mathbf{y} + d\delta \geq 0, \quad \text{for each } \sigma \in S_n. \quad (3.2)$$

*Proof.* Let  $\mathbf{y}$  be any  $\leq \delta$ -min-entropy point from  $V$ ; so assume that  $\mathbf{y}$  has some majority element  $i$  with frequency  $\geq m = n - \delta$ . Let  $\sigma \in S_n$  be any permutation. Define  $l := (\sigma \circ \mathbf{u})^\top \cdot \mathbf{y}$ . It suffices to show that  $l \geq -d\delta$ . We claim that to minimize  $l$  by varying  $(\sigma \circ \mathbf{u})$ , we can place at most  $\delta$  many  $d$ 's in the coordinates corresponding to  $-1$  and rest of the  $\geq m$  coordinates must be filled by  $i$ . Since we have a total of  $(\delta + m/2)$  coordinates equal to  $+1$  and remaining  $(m/2)$  coordinates equal to  $-1$ , the minimum value of  $l$  in this case is  $(\delta + m/2) \cdot i - (m/2 - \delta) \cdot i - (\delta) \cdot d = (2i - d)\delta \geq -d\delta$ , since  $i \geq 0$ . Thus,  $l \geq -d\delta$  in this configuration.

We now see why this is the optimal configuration to minimize  $l$ . Note that the majority element  $i$  will always have a non-negative contribution in  $l$ , since its frequency in the positions corresponding to  $+1$  will always be greater than or equal to its frequency in the positions corresponding to  $-1$ . This is because there are only  $m/2$  positions corresponding to  $-1$  and  $i$  has frequency  $\geq m$ . Therefore to minimize its contribution, majority element  $i$  should be fixed to  $0$ . To get lowest possible value from non-majority elements in  $\mathbf{y}$ , we should set all of them to  $d$  and place them in positions corresponding to  $-1$ . This will give minimum value of  $l$  to be  $-d\delta$ . Rest of the configurations will only have higher values. Thus,  $l + d\delta \geq 0$  for each  $\mathbf{y} \in V$  and for every  $\sigma \in S_n$ .  $\square$

If for a set of points, each point lies on one side of a hyperplane, then all the points in their convex-span also lie on that side. We prove this observation below. This will imply that, Lemma 3.1.1 holds for every point in  $CS(V)$  also.

**Observation 3.1.2** (Halfspace is convex). Let  $V = \{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k\} \subseteq \mathbb{R}^n$  be a set of  $k$

vectors. Suppose for some  $(\mathbf{a}, b) \in \mathbb{R}^n \times \mathbb{R}$  and for each  $i \in [k]$ ,  $\mathbf{a}^\top \cdot \mathbf{v}_i + b \geq 0$ . Then, for any  $\mathbf{v} \in CS(V)$ ,  $\mathbf{a}^\top \cdot \mathbf{v} + b \geq 0$ .

*Proof.* This follows because  $\mathbf{v}$  is a convex combination of vectors in  $V$ . Let  $\mathbf{v} = \alpha_1 \cdot \mathbf{v}_1 + \dots + \alpha_k \cdot \mathbf{v}_k$ , where  $\sum_{i=1}^k \alpha_i = 1$  and  $\alpha_i \in \mathbb{R}_{\geq 0}$  for each  $i \in [k]$ . Thus,

$$\begin{aligned} \mathbf{a}^\top \cdot \mathbf{v} + b &= \mathbf{a}^\top \cdot \left( \sum_{i=1}^k \alpha_i \cdot \mathbf{v}_i \right) + b \\ &= \left( \sum_{i=1}^k \alpha_i \cdot \mathbf{a}^\top \cdot \mathbf{v}_i \right) + \sum_{i=1}^k \alpha_i \cdot b \\ &= \sum_{i=1}^k \alpha_i \cdot (\mathbf{a}^\top \cdot \mathbf{v}_i + b) \geq 0. \end{aligned}$$

In the second step, we use  $\sum_{i=1}^k \alpha_i = 1$ ; while in the last step we use the hypothesis and  $\alpha_i \geq 0$  for each  $i \in [k]$ .  $\square$

Suppose we are given a sorted point  $\mathbf{y}$  that satisfies the inequality  $\mathbf{u}^\top \cdot \mathbf{y} + d\delta \geq 0$ . Below, we prove that such a point has low min-entropy.

**Lemma 3.1.3** (Integral-point in the cover). *Let  $\mathbf{u}$  be as defined in (3.1). Let  $\mathbf{y} \in \{0, 1, \dots, d\}^n$  be a point with  $0 \leq y_1 \leq y_2 \leq \dots \leq y_n \leq d$  such that  $\mathbf{u}^\top \cdot \mathbf{y} + d\delta \geq 0$  holds. Then,  $\mathbf{y}$  is a  $(2d\delta)$ -min-entropy point.*

*Proof.* Recall  $n = \delta + m$ . Let us call, expectedly, the first  $(\delta + m/2)$  coordinates, the ‘positive zone’ of  $\mathbf{y}$  and the remaining  $(m/2)$  coordinates, the ‘negative zone’ of  $\mathbf{y}$ ; this corresponds to positions of  $+1$ ’s and  $-1$ ’s in  $\mathbf{u}$  respectively. Let  $\mathbf{y} =: \mathbf{y}_+ + \mathbf{y}_-$ , where we define  $\mathbf{y}_+$  and  $\mathbf{y}_-$  as follows,

$$\begin{aligned} (\mathbf{y}_+)_j &:= \begin{cases} y_j & \text{if } j \leq \delta + m/2 \\ 0 & \text{otherwise.} \end{cases} \\ (\mathbf{y}_-)_j &:= \begin{cases} 0 & \text{if } j \leq \delta + m/2 \\ y_j & \text{otherwise.} \end{cases} \end{aligned}$$

Suppose the first coordinate in the negative zone of  $\mathbf{y}$  is  $i$  for some  $i \in \{0, 1, \dots, d\}$ . We

then claim that  $\mathbf{y}$  has at least  $n - 2d\delta$  coordinates with this same value  $i$ , proving  $\mathbf{y}$  to be a  $2d\delta$ -min-entropy point. Let  $p_+$  and  $p_-$  denote the frequency of  $i$  in positive and negative zones of  $\mathbf{y}$  respectively, for some integers  $p_+, p_- \geq 0$ . Note that  $\mathbf{u}^\top \cdot \mathbf{y} = \|\mathbf{y}_+\|_1 - \|\mathbf{y}_-\|_1$ , where  $\|\cdot\|_1$  is the  $\mathbf{L}^1$ -norm. We first upper bound  $\|\mathbf{y}_+\|_1$ . Since  $\mathbf{y}$  is sorted, the last  $p_+$  coordinates in positive zone must be  $i$  and all coordinates preceding it are of value at most  $i - 1$ . This gives us

$$\|\mathbf{y}_+\|_1 \leq (i - 1) \cdot (\delta + m/2 - p_+) + i \cdot (p_+) = i\delta + im/2 - \delta - m/2 + p_+ . \quad (3.3)$$

Now, we lower bound  $\|\mathbf{y}_-\|_1$ . Since  $\mathbf{y}$  is sorted, the first  $p_-$  coordinates in negative zone must be  $i$  and all subsequent coordinates are of value at least  $(i + 1)$ . This gives us

$$\|\mathbf{y}_-\|_1 \geq i \cdot (p_-) + (i + 1) \cdot (m/2 - p_-) = im/2 + m/2 - p_- . \quad (3.4)$$

Let  $l := \mathbf{u}^\top \cdot \mathbf{y} + d\delta$ . By hypothesis,  $l \geq 0$ . Then using (3.3) and (3.4) together, we observe that

$$\begin{aligned} 0 \leq l &= \mathbf{u}^\top \cdot \mathbf{y} + d\delta = \|\mathbf{y}_+\|_1 - \|\mathbf{y}_-\|_1 + d\delta \\ 0 \leq &i\delta + im/2 - \delta - m/2 + p_+ - (im/2 + m/2 - p_-) + d\delta \\ &= (i + d)\delta - \delta - m + p_+ + p_- \\ &\leq 2d\delta - n + p_+ + p_- \quad (\text{since } i \leq d \text{ and } n = m + \delta) \\ n - 2d\delta &\leq p_+ + p_- . \end{aligned}$$

Recall that total frequency of the value  $i$  in  $\mathbf{y}$  is  $p_+ + p_- \geq n - 2d\delta$ . This proves that  $\mathbf{y}$  is a  $(2d\delta)$ -min-entropy point; finishing the proof. (We note that the calculations in (3.3) and (3.4) are technically for  $i \in [d - 1]$ . For the corner cases of  $i = 0$  or  $i = d$ , the same logic will work and in fact with a better lower bound on frequency of  $i$  in  $\mathbf{y}$ .)  $\square$

We are now ready to prove the main theorem of this section.

**Theorem 3.1.4** (Polytope Entropy Theorem). *Let  $V \subseteq \{0, 1, \dots, d\}^n$  be a  $\delta$ -min-entropy set, then  $CS(V) \cap \mathbb{Z}^n$  is a  $(2d\delta)$ -min-entropy set.*

*Proof.* In Lemma 3.1.1, we showed that every point  $\mathbf{v} \in V$  belongs to the halfspace

$\mathbf{u}^\top \cdot \mathbf{v} + d\delta \geq 0$ . In fact, we proved it for every permutation  $\sigma \in S_n$  of  $\mathbf{u}$ . Let  $\mathbf{y}$  be any integral-point in  $CS(V)$ . Thus by Observation 3.1.2,  $(\sigma \circ \mathbf{u})^\top \cdot \mathbf{y} + d\delta \geq 0$ , for every  $\sigma \in S_n$ .

Note that since  $V \in \{0, 1, \dots, d\}^n$ , any integral point  $\mathbf{y} \in CS(V)$  also lies in  $\{0, 1, \dots, d\}^n$ . Further, let  $\pi \in S_n$  be the permutation which sorts  $\mathbf{y}$ , i.e.  $y_{\pi(1)} \leq y_{\pi(2)} \leq \dots \leq y_{\pi(n)}$ . Since the hyperplane cover holds for every permutation, in particular it holds for  $\pi^{-1} \in S_n$  also, i.e.  $(\pi^{-1} \circ \mathbf{u})^\top \cdot \mathbf{y} + d\delta \geq 0$ . Thus,

$$(\pi^{-1} \circ \mathbf{u})^\top \cdot \mathbf{y} + d\delta = \mathbf{u}^\top \cdot (\pi \circ \mathbf{y}) + d\delta \geq 0.$$

Now by Lemma 3.1.3, as  $\pi \circ \mathbf{y}$  is sorted, we deduce that  $\pi \circ \mathbf{y}$  is a  $(2d\delta)$ -min-entropy vector. Since min-entropy of a vector does not change on permuting it, we deduce that  $\mathbf{y}$  is also a  $(2d\delta)$ -min-entropy point.  $\square$

**Remark.** Note that Theorem 3.1.4 is almost tight as a blow-up in min-entropy of internal integral points by a factor of  $d$  is inevitable. To observe this, consider  $V$  to be the support set in Example 2.2.2 with min-entropy  $\delta = 1$ . It is easy to see that the integral vector  $\mathbf{v} = \sum_{i=1}^d e_i$  is in  $CS(V)$ , where  $e_i$  is the standard unit vector with a single 1 in position  $i$  and rest all 0. Thus,  $\mathbf{v}$  is a vector with 1 in first  $d$  coordinates and remaining all 0, and it has min-entropy exactly  $d$ , for  $d \leq n/2$ . Therefore,  $CS(V) \cap \mathbb{Z}^n$  is of min-entropy at least  $d\delta$ . However, in this work we are concerned with  $d = O(1)$ , so even  $2d\delta$  blowup is perfectly fine.

## 3.2 Polytope point counting

In the last section, we proved that a low min-entropy vertex set implies low min-entropy for the whole polytope. Now, we shall see that a low min-entropy polytope has low number of total integral points. We start by observing that a sparse, symmetric polynomial cannot have high min-entropy. This will then help us prove that a symmetric polytope has low number of integral points. Recall the definition of symmetric-support polynomials and symmetric polytopes from Section 2.4.

### 3.2.1 Point counting

We now come to the counting part of this work, where we shall connect symmetry, min-entropy and sparsity with each other. We will also make use of Theorem 3.1.4 here to achieve efficient bounds on the count of integral points of symmetric and low min-entropy polytopes. The lemma below mentions few standard bounds that we will make use of later.

**Lemma 3.2.1** (Counting estimates). *1. For positive integers  $a, b$  with  $a \geq b$ ,  $(a/b)^b \leq \binom{a}{b}$ .*

*2. For positive integers  $a, b, c$  with  $a \geq bc$ ,  $\binom{a}{bc} \leq \left(\binom{a}{c}\right)^b$ .*

*3. For positive real  $x$ ,  $\log(1+x) > \ln(1+x) > x - \frac{x^2}{2}$ .*

*Proof.* For (1), see that  $\binom{a}{b} = \frac{a(a-1)\cdots(a-b+1)}{b(b-1)\cdots 1} \geq \left(\frac{a}{b}\right)^b$ .

For (2), we make use of  $\binom{a}{c+b'} \leq \binom{a}{b'} \cdot \binom{a-b'}{c} \leq \binom{a}{b'} \cdot \binom{a}{c}$ . Use this  $b$  times to get  $\binom{a}{bc} \leq \left(\binom{a}{c}\right)^b$ .

For (3), let  $f(x) = \ln(1+x) - (x - \frac{x^2}{2})$ . Then,

$$f'(x) = \frac{1}{1+x} - (1-x) = \frac{x^2}{1+x}.$$

Note that for  $x = 0$ ,  $f(0) = 0$  and  $f'(x) > 0$  for  $x > 0$ . Therefore,  $f(x) > 0$ , for  $x > 0$ .  $\square$

Using a simple counting argument below, we bound the total number of points in a given  $\delta$ -min-entropy set of integral points. Thus, low min-entropy implies small cardinality for a set.

**Lemma 3.2.2** (min-entropy-sparsity upper bound). *Let  $T \subseteq \{0, 1, \dots, d\}^n$  be a  $\delta$ -min-entropy set. Then,  $|T| \leq \binom{n}{\delta} \cdot (d+1)^{\delta+1}$ .*

*Proof.* Any  $\mathbf{v} \in T$  has min-entropy  $\leq \delta$  and thus has at least  $n - \delta$  coordinates that are equal. To specify  $\mathbf{v}$ , one needs to specify the indices of these repeated coordinates ( $\leq \binom{n}{\delta}$  possibilities) and specify a total of  $\delta+1$  values for all the coordinates of  $\mathbf{v}$  ( $d+1$  possibilities for each). Hence, we deduce that  $|T| \leq \binom{n}{\delta} \cdot (d+1)^{\delta+1}$ .  $\square$

We now show that a symmetric set of integral points is of somewhat low min-entropy.

**Lemma 3.2.3** (symmetry  $\Rightarrow$  low min-entropy). *Let  $V \subseteq \{0, 1, \dots, d\}^n$  be any symmetric set. Then  $V$  is a  $\delta$ -min-entropy set, for some  $\delta < 2d \cdot \log |V|$ .*

*Proof.* Consider the smallest possible  $\delta$ , for which  $V$  can be a  $\delta$ -min-entropy set. By pigeonhole principle, for any point in  $V$ , there exists a coordinate-value with frequency  $\geq n/(d+1)$ . Therefore,

$$\delta \leq n - \frac{n}{d+1} = n \left( 1 - \frac{1}{d+1} \right). \quad (3.5)$$

Since  $\delta$  is chosen to be the minimum possible, a non-empty  $V$  contains a vector  $\mathbf{v}$  having some integral value with maximum-frequency exactly  $n - \delta$ . Since  $V$  is symmetric, it must also contain all the distinct  $S_n$ -permutations of this vector  $\mathbf{v}$ , which are at least  $\binom{n}{\delta}$  many. This implies that  $|V| \geq \binom{n}{\delta}$  and further using Lemma 3.2.1, we get that  $|V| \geq \left(\frac{n}{\delta}\right)^\delta$ . Then,

$$\begin{aligned} \log |V| &\geq \delta \cdot \log \left( \frac{n}{\delta} \right) \\ &\geq \delta \cdot \log \left( \frac{n}{n \left( 1 - \frac{1}{d+1} \right)} \right) \quad [\text{Using (3.5)}] \\ &= \delta \cdot \log \left( 1 + \frac{1}{d} \right) \\ &> \delta \cdot \left( \frac{1}{d} - \frac{1}{2d^2} \right) \quad [\text{Using Lemma 3.2.1}] \\ &\geq \frac{\delta}{2d} \quad [\text{As } d \geq 1]. \end{aligned}$$

This proves that  $\delta < 2d \cdot \log |V|$ . □

Finally, we come to our polytope bounds. Using Theorem 3.1.4 and the counting observations above, we now show that convex span of a low min-entropy set has low number of integral points.

**Theorem 3.2.4** (low min-entropy polytope count). *Let  $V \subseteq \{0, 1, \dots, d\}^n$  be a  $\delta$ -min-entropy set. Then,  $|CS(V) \cap \mathbb{Z}^n| \leq \binom{n}{\delta}^{2d} \cdot (d+1)^{2d\delta+1}$ .*

*Proof.* Theorem 3.1.4 proves that  $CS(V) \cap \mathbb{Z}^n$  is a  $(2d\delta)$ -min-entropy set. Note that  $CS(V) \cap \mathbb{Z}^n$  is also a subset of  $\{0, \dots, d\}^n$ , since  $V$  is. Then by Lemma 3.2.2,  $|CS(V) \cap \mathbb{Z}^n| \leq \binom{n}{2d\delta} \cdot (d+1)^{2d\delta+1}$ . The conclusion then follows by using Lemma 3.2.1 to observe that  $\binom{n}{2d\delta} \leq \binom{n}{\delta}^{2d}$ .  $\square$

We now show that the gap between number of vertices and total number of integral points in a symmetric polytope, is low.

**Theorem 3.2.5** (Symmetric polytope count). *Let  $V \subseteq \{0, 1, \dots, d\}^n$  be the vertices of a symmetric polytope  $P$ . Then,  $|P \cap \mathbb{Z}^n| \leq |V|^{O(d^2 \log d)}$ .*

*Proof.* Consider the smallest possible  $\delta$ , for which  $V$  can be a  $\delta$ -min-entropy set. Since  $P$  is a symmetric polytope, its vertex set  $V$  is also symmetric by Lemma 2.4.4. By Lemma 3.2.3, the min-entropy of  $V$  is at most  $\delta := O(d \cdot \log |V|)$ . Then by Theorem 3.2.4,  $|CS(V) \cap \mathbb{Z}^n| \leq \binom{n}{\delta}^{2d} \cdot (d+1)^{2d\delta+1}$ . Observe that  $(d+1)^{2d\delta+1} \leq d^{O(d^2 \log |V|)} = |V|^{O(d^2 \log d)}$ . Since  $V$  is non-empty, it contains at least one  $\delta$  min-entropy vector. Since  $V$  is symmetric, it also contains all the permutations of this vector, which are at least  $\binom{n}{\delta}$ -many. Hence,  $|V| \geq \binom{n}{\delta}$ . Thus,  $\binom{n}{\delta}^{2d} \leq |V|^{2d}$  and we get the desired conclusion.  $\square$

**Remark 3.2.6.** We can also view Theorem 3.2.5 in a different way. Let  $P$  be a symmetric polytope formed by taking convex hull of some symmetric set  $E \subseteq \{0, 1, \dots, d\}^n$ . Then,  $P$  must have *many* vertices (corner points), at least  $|E|^{\Omega(1/(d^2 \log d))}$ -many, to be precise. We get this *incompressibility result* by substituting  $P = CS(E)$  in Theorem 3.2.5 and observing that  $E \subseteq P \cap \mathbb{Z}^n$ .

### 3.3 Factor sparsity bounds

In Section 2.3, we discussed the broad polytope approach to getting factor-sparsity bounds, in particular Equation (2.2). We proved Theorem 3.2.4 and Theorem 3.2.5 in the previous section, which give us the required polytope bounds, for using this method. In this section, we will reap benefits of our hard labor done earlier to show that factors of symmetric or low min-entropy sparse polynomials are also sparse.

**Remark.** Throughout this section, whenever we talk about symmetric polynomials, we can replace them with the larger class of symmetric-support polynomials. The proofs remain the same. See Definition 2.4.1 to recap what we mean by symmetric-support.

We start by proving sparsity bound for a symmetric factor  $g$  of some  $s$ -sparse polynomial  $f$ , which may or may not be symmetric. To get a  $\text{poly}(s)$  sparsity bound, we only require individual degree of  $g$  to be constant, while  $f$  may have arbitrary individual degree. Of course, this is only useful when  $f$  has some symmetric factor.

**Theorem 3.3.1** (Symmetric factor sparsity bound). *Let  $f \in \mathbb{F}[x_1, \dots, x_n]$  be an  $s$ -sparse polynomial. Let  $g$  be any symmetric factor of  $f$ , with individual degree at most  $d$ . Then,  $g$  has sparsity at most  $s^{O(d^2 \log d)}$ .*

*Proof.* Let  $P_g$  be the Newton polytope of  $g$  and  $V_g$  be its vertex set. Since  $g$  is symmetric,  $P_g$  is a symmetric polytope. Now invoke Theorem 3.2.5 with  $P = P_g$  to note that  $|P_g \cap \mathbb{Z}^n| \leq |V_g|^{O(d^2 \log d)}$ . Observe that  $\text{supp}(g) \subseteq P_g \cap \mathbb{Z}^n$  and  $|V_g| \leq |V_f| \leq s$ . Therefore,  $\|g\| \leq s^{O(d^2 \log d)}$ .  $\square$

We will be needing the following observation for our main proofs below.

**Observation 3.3.2.** Let  $f, g, h \in \mathbb{F}[x_1, \dots, x_n]$  be polynomials such that  $f = g \cdot h$ . Let  $P_f$  be the Newton polytope of  $f$ . Then,  $\|g\| \leq |P_f \cap \mathbb{Z}^n|$ .

*Proof.* By Minkowski sum property, we have that  $P_f = P_g + P_h$ . For every  $\mathbf{u} \in \text{supp}(g)$ , consider a fixed point  $\mathbf{v} \in \text{supp}(h)$ . Observe that  $\mathbf{u} + \mathbf{v} \in P_f$  for every  $\mathbf{u}$ , since  $\mathbf{u} \in P_g$  and  $\mathbf{v} \in P_h$ . Moreover, since the support vectors are integral,  $\mathbf{u} + \mathbf{v} \in P_f \cap \mathbb{Z}^n$ , for every  $\mathbf{u}$ . In other words,  $P_f \cap \mathbb{Z}^n$  contains a translated copy of  $\text{supp}(g)$ . This means that  $\|g\| \leq |P_f \cap \mathbb{Z}^n|$ .  $\square$

We have sufficient tools now to prove our main theorems of this chapter.

**Theorem 3.3.3** (Symmetric sparsity bound). *Let  $f \in \mathbb{F}[x_1, \dots, x_n]$  be an  $s$ -sparse, symmetric polynomial with individual degree  $d$ . Then, every factor of  $f$  has its sparsity bounded by  $s^{O(d^2 \log d)}$ .*



*Proof.* Since  $f$  is symmetric, its Newton polytope  $P_f$  will also be symmetric. Let  $V_f$  be the vertex set of  $P_f$ . Then, invoke Theorem 3.2.5 with  $P = P_f$  to deduce that  $|P_f \cap \mathbb{Z}^n| \leq |V_f|^{O(d^2 \log d)}$ . The conclusion then follows from Observation 3.3.2 and  $|V_f| \leq s$ .  $\square$

The above factor-sparsity bound is  $\text{poly}(s)$  when  $d$  is constant. For  $\delta$ -min-entropy polynomials, we get  $\text{poly}(n)$  factor-sparsity bound below, when  $d, \delta$  are constants.

**Theorem 3.3.4** (Factors of low min-entropy polynomials). *Let  $f \in \mathbb{F}[x_1, \dots, x_n]$  be a  $\delta$ -min-entropy polynomial with individual degree  $d$ . Then, every factor of  $f$  has its sparsity bounded by  $(nd)^{O(d\delta)}$ .*

*Proof.* Let  $P_f$  be the Newton polytope of  $f$  and  $V_f$  be its vertex set. Since  $f$  is of  $\delta$ -min-entropy, so is  $V_f$ , as  $V_f \subseteq \text{supp}(f)$ . Then by Theorem 3.2.4, we deduce that  $|P_f \cap \mathbb{Z}^n| \leq \binom{n}{\delta}^{2d} \cdot (d+1)^{2d\delta+1} \leq (nd)^{O(d\delta)}$ . The conclusion then follows from Observation 3.3.2.  $\square$

**Remark.** In Theorem 3.3.1 (resp. Theorem 3.2.5), we do not require the whole of  $g$  (resp.  $P$ ) to be symmetric, rather we only need its vertex set  $V_g$  (resp.  $V$ ) to be symmetric, which is a much weaker requirement. Similarly, we don't need  $f$  to be symmetric in Theorem 3.3.3 but only  $V_f$  to be symmetric. In these cases, we will not require use of Lemma 2.4.4.

### 3.3.1 Tightness of sparsity bounds

We consider the two examples discussed earlier where factors of a sparse polynomial have significantly high sparsity, unless the individual degree is bounded.

In Example 1.2.1, observe that  $\|f\| = 2^n$ , while  $\|g\| = d^n$ . If we let  $s := \|f\|$ , then  $\|g\| = s^{\log d}$ . Over fields of characteristic 0, this is an example which exhibits highest known blowup in sparsity. Moreover, it also shows that  $s^{\log d}$  is a lower bound on factor-sparsity. Note that  $f$  is a symmetric polynomial and Theorem 3.3.3 shows that  $\|g\| \leq s^{O(d^2 \log d)}$ .

In Example 1.2.2, observe that  $\|f\| = n$ , while  $\|g\| = \binom{n+d-1}{d} \approx n^d$ . If we let  $s := \|f\|$ , then  $\|g\| \approx s^d$ . The factor-sparsity bounds in this work hold for any field  $\mathbb{F}$ , finite or otherwise. Moreover,  $f$  is also symmetric and falls under the purview of Theorem 3.3.3, which shows that  $\|g\| \leq s^{O(d^2 \log d)}$ . Hence, this example shows that in Theorem 3.3.3,

we cannot do better than  $s^d$ . Note that this  $f$  is also a low min-entropy polynomial, in fact with  $\delta = 1$  as all the exponent vectors in  $\text{supp}(f)$  have  $n - 1$  coordinates having the same value 0. Thus, we can use Theorem 3.3.4 for this  $f$  to get a factor-sparsity bound of  $(np)^{O(p)}$  which is really close to the actual sparsity  $n^d$ .

### 3.4 Factoring algorithms

Given a polynomial  $f$ , the *complete factorization* of  $f$  is a representation of  $f$  as  $f_1^{e_1} f_2^{e_2} \cdots f_k^{e_k}$ , where  $f_1, \dots, f_k$  are co-prime irreducible polynomials and  $e_1, \dots, e_k$  are positive integers. This representation is unique up to a reordering of  $f_i$ 's. A nice feature of the results in [BSV20] is that once you prove a nice factor-sparsity bound, they also show how to get a deterministic factoring algorithm for sparse polynomials which runs in time polynomial in the sparsity bound proven. We restate this as Lemma 3.4.1 below. Here  $c_{\mathbb{F}}(d)$  is the best known time complexity for factoring a univariate polynomial of degree  $d$  over the field  $\mathbb{F}$ . For  $\mathbb{F} = \mathbb{Q}$ ,  $c_{\mathbb{F}}(d) \leq \text{poly}(d, t)$  where  $t$  is the maximum bit-complexity of the coefficients of  $f$  [LLL82]. For a finite field  $\mathbb{F} = \mathbb{F}_{p^\ell}$ ,  $c_{\mathbb{F}}(d) \leq \text{poly}(\ell \cdot p, d)$  [Ber67, CZ81].

**Lemma 3.4.1** (Theorem 5.8 in [BSV20]). *Let  $f \in \mathbb{F}[x_1, \dots, x_n]$  be an  $s$ -sparse polynomial with individual degrees at most  $d$ . Let  $\xi(n, d, s)$  be the upper bound on sparsity for every factor of  $f$ . Then given  $f$ , there is a deterministic algorithm that computes complete factorization of  $f$  in  $(n \cdot \xi(n, d^2, s^d))^{O(d^2)} \cdot \text{poly}(c_{\mathbb{F}}(d^2))$  field operations.*

In other words, the above theorem shows a deterministic reduction from multivariate to univariate factoring over any field  $\mathbb{F}$ . In [BSV20], they showed that  $\xi(n, d, s) \leq s^{O(d^2 \log n)}$  and then used Lemma 3.4.1 to get a factoring algorithm of  $s^{\text{poly}(d) \log n}$  time complexity, which is quasi-polynomial in the bounded individual degree setting. In this work, we deal with symmetric and constant-min-entropy input polynomials, for which we show that  $\xi(n, d, s) \leq (ns)^{\text{poly}(d)}$ . Thus, we can use Lemma 3.4.1 to get polynomial time factoring algorithms in the bounded individual degree setting. The finer time complexities are discussed in the proofs of Corollary 3.4.2 and Corollary 3.4.3 below. Also, note that  $\xi(n, d, s)$  is a lower bound on time complexity for sparse factoring algorithms as we output

each factor as an explicit list of monomials.

**Corollary 3.4.2** (Symmetric factoring algorithm). *Let  $f \in \mathbb{F}[x_1, \dots, x_n]$  be an  $s$ -sparse, symmetric polynomial with individual degree  $d$ . Then, there is a deterministic algorithm that computes the complete factorization of  $f$  in at most  $\text{poly}(s^{d^7 \log d} \cdot n^{d^2} \cdot c_{\mathbb{F}}(d^2))$ -time.*

*Proof.* Let  $f \in \mathbb{F}[x_1, \dots, x_n]$  be an  $s$ -sparse, symmetric polynomial with individual degree  $d$ . In Theorem 3.3.3, we show that  $\xi(n, d, s) \leq s^{O(d^2 \log d)}$ . Plugging this value in Lemma 3.4.1, we get a deterministic factoring algorithm for  $f$ . The time complexity  $T(n, s, d)$  for this algorithm is:

$$\begin{aligned} T(n, s, d) &= (n \cdot \xi(n, d^2, s^d))^{O(d^2)} \cdot \text{poly}(c_{\mathbb{F}}(d^2)) \\ &= \left( n \cdot s^{d \cdot O(d^4 \log d^2)} \right)^{O(d^2)} \cdot \text{poly}(c_{\mathbb{F}}(d^2)) \\ &= s^{O(d^7 \log d)} \cdot n^{O(d^2)} \cdot \text{poly}(c_{\mathbb{F}}(d^2)). \end{aligned} \quad \square$$

The time complexity of the above factoring algorithm is  $\text{poly}(s, n, c_{\mathbb{F}}(d))$  when  $d$  is a constant. For a constant-min-entropy polynomial with constant individual degree, we can factor it in  $\text{poly}(n, c_{\mathbb{F}}(d))$ -time as shown below.

**Corollary 3.4.3** (Low min-entropy factoring algorithm). *Let  $f \in \mathbb{F}[x_1, \dots, x_n]$  be a  $\delta$ -min-entropy polynomial with individual degree  $d$ . Then, there is a deterministic algorithm that computes the complete factorization of  $f$  in at most  $\text{poly}((nd)^{d^4 \delta} \cdot c_{\mathbb{F}}(d^2))$ -time.*

*Proof.* Let  $f \in \mathbb{F}[x_1, \dots, x_n]$  be a  $\delta$ -min-entropy polynomial with individual degree  $d$ . In Theorem 3.3.3, we show that  $\xi(n, d, s) \leq (nd)^{O(d\delta)}$ . Plugging this value in Lemma 3.4.1, we get a deterministic factoring algorithm for  $f$ . The time complexity  $T(n, s, d)$  for this algorithm is:

$$\begin{aligned} T(n, s, d) &= (n \cdot \xi(n, d^2, s^d))^{O(d^2)} \cdot \text{poly}(c_{\mathbb{F}}(d^2)) \\ &= \left( n \cdot (nd^2)^{O(d^2 \delta)} \right)^{O(d^2)} \cdot \text{poly}(c_{\mathbb{F}}(d^2)) \\ &= (nd)^{O(d^4 \delta)} \cdot \text{poly}(c_{\mathbb{F}}(d^2)). \end{aligned} \quad \square$$

### 3.5 Discussion

**Comparison with previous techniques:** The fascinating approach of connecting sparsity with Newton polytopes was first presented in [DdO14], which was summarized in Equation (2.2) earlier. The more important part of actually getting a non-trivial factor-sparsity bound using this polytope approach was done in [BSV20]. For  $f = gh$ , [BSV20] proved a lower bound on  $|V(P_g)|$  in terms of  $\|g\|$  by using an approximate version of Carathéodory’s theorem [BSV20, Theorem 3.6] and showed that  $|V(P_g)| \geq \|g\|^{\frac{1}{O(d^2 \log n)}}$ . Thus using (2.2), they get  $\|g\| \leq \|f\|^{O(d^2 \log n)}$ . Although the use of approximate Carathéodory’s theorem gives the first non-trivial factor-sparsity bound, it also brings in an  $O(\log n)$  term in the exponent of their bound. In this work, we make use of the sparsity connection in [DdO14] but replace the lower bound method of [BSV20] with our new techniques of designing a specific set of *hyperplane equations* in Structure Theorem 3.1.4, and exploiting the symmetry of polytopes in Theorem 3.2.5. These new techniques help us get rid of the unwanted  $\log n$  dependence in exponent of the factor-sparsity bound. If  $g$  is symmetric, we show a much better lower bound of  $|V(P_g)| \geq \|g\|^{\frac{1}{O(d^2 \log d)}}$ . This gives us  $\|g\| \leq \|f\|^{O(d^2 \log d)}$  using (2.2).

The result of [BSV20] and Theorem 3.2.5 in our work, both show a lower bound on  $|V(P_g)|$ . Remark 4.3 and Claim 4.4 of [BSV20] show that this particular proof strategy cannot get a sparsity upper bound better than  $s^{O(\log n)}$ , for general factors of a general sparse polynomial. They show that the general polytope approach hits a dead end there. However, we derive the results in this work despite these known limitations and show a positive evidence for the sparsity conjecture Conjecture 1.2.3. We note that the polytope example in Claim 4.4 of [BSV20] is not symmetric and is of high min-entropy and therefore, it is not a hurdle for the results of this work. We utilize structural properties like symmetry or low min-entropy of input  $f$  to get sparsity upper bound for its factors, even though the factors themselves might not be symmetric or of low min-entropy.

**Future Directions:** This work proved polynomial-sized factor-sparsity bounds for low min-entropy polynomials. The next task is to prove efficient bounds for factors of high-min-

entropy polynomials. We note that the example in Claim 4.4 of [BSV20] is a polynomial with its vertices having exactly  $n/2$ -min-entropy such that its Newton polytope has at least  $n^{\Omega(\log n)}$ -many integral points. Therefore, a new technique which goes beyond counting number of internal points in a polytope is required. Otherwise, if one does not believe in the sparsity-conjecture, then one needs to come up with a sparse high-min-entropy polynomial which has dense factors.



## Chapter 4

# Exact Power Testing

In this chapter, we consider a subproblem of sparse factoring. We restrict the input sparse polynomial  $f$  to be an *exact power*. A polynomial  $f \in \mathbb{F}[x_1, x_2, \dots, x_n]$  is an exact power if there exists (another) polynomial  $g \in \mathbb{F}[x_1, x_2, \dots, x_n]$  and  $e \in \mathbb{N}$  such that  $f = g^e$ . The motivating question here is can one obtain a sparsity bound for  $g$ , better than that of a general factor? Despite the rich structure, this question is still open. The best known sparsity bound for exact roots (i.e.  $\|g\|$  in terms of  $\|f\|$ ) is still the general sparsity bound of size  $s^{O(d^2 \log n)}$  by [BSV20]. However, we can still ask the decision problem of testing whether  $f$  is an exact power. Observe that one can use the factorization algorithm of [BSV20] to test if a given sparse polynomial is an exact power, in quasi-polynomial time. Similarly, a polynomial-size sparsity bound, even for the case of exact roots, would imply a polynomial-time algorithm for exact-power testing problem. In this chapter, we provide a polynomial-time algorithm for exact-power testing that **does not** rely on this sparsity bound. We show that, *there is a deterministic algorithm that given a sparse polynomial  $f \in \mathbb{F}[x_1, x_2, \dots, x_n]$  of constant individual degree as an input, decides whether  $f = g^e$  for some polynomial  $g \in \mathbb{F}[x_1, x_2, \dots, x_n]$  and  $e \in \mathbb{N}$ , in time  $\text{poly}(s, n)$* . It is formally stated in Theorem 4.1.7.

We remark that the algorithm only performs exact-power testing and **does not** output a “witness” polynomial  $g$ . Indeed, a polynomial-time algorithm that actually outputs  $g$  would imply a polynomial-size sparsity bound on exact roots! In addition, the runtime

of our algorithm is polynomial in the bit-complexity of the field elements since it does not rely on univariate polynomial factorization. For instance, for finite fields we get the runtime of  $\text{poly}(\log |\mathbb{F}|)$  vs  $\text{poly}(|\mathbb{F}|)$ .

## 4.1 Exact Power Testing

In this section we will design our algorithm for exact power testing. We want to test whether  $f = g^e$  for some  $g \in \mathbb{F}[x_1, x_2, \dots, x_n]$  and  $e \in \mathbb{N}$ . We first show an  $s^{O(d)}$  sparsity bound for  $g$ , when  $f$  is an  $s$ -sparse, reverse-monic polynomial of individual degree  $d$  (Lemma 4.1.1). Moreover, we also get an algorithm to compute  $g$  for this case in Algorithm 1.

In general though, our input polynomial  $f$  may not be reverse-monic, but we show that sparsity bound obtained for the reverse-monic case suffices for exact power testing. We first convert  $f$  into a reverse-monic polynomial  $\hat{f}$  with respect to some variable  $x_i$ , using a known standard transformation (see Definition 4.1.4). This step only incurs a slight sparsity blow-up of  $s^d$ . One important property of this transformation is that it preserves the “exact power” structure. That is, if  $f = g^e$ , then  $\hat{f} = h^e$ , for some polynomial  $h$ . We then compute this  $e$ -th root of the reverse-monic  $\hat{f}$ , as mentioned previously.

However, we are still not quite done. It can happen that a polynomial  $f$  which was not an exact power, may become an exact power after the reverse-monic transformation. We need an additional condition to get the converse implication. We show that if both  $\hat{f}$  and  $f|_{x_i=0}$  are exact powers, then we can correctly conclude that  $f$  is also an exact power (Claim 4.1.6). This gives us a recursive algorithm, as  $f|_{x_i=0}$  is a polynomial in  $(n - 1)$  variables. This procedure is described formally in Algorithm 2.

### 4.1.1 Reverse Monic Case

We start with the case when our input polynomial  $f$  is reverse-monic w.r.t. some variable. We generalize it to the general case in Section 4.1.2.

We use Newton’s Binomial Theorem to get the sparsity bound for  $f^{1/e}$  below. This tool



has been used before to bound the size of  $e$ -th roots for the classes of algebraic circuits, formulas and ABPs. See for example [Dut18, ST21b]. Although, these works assume  $\text{char}(\mathbb{F})$  to be 0 or a non-divisor of  $e$ , we prove our result below for fields of arbitrary characteristic.

**Lemma 4.1.1.** *Let  $f \in \mathbb{F}[x_1, \dots, x_n]$  be an  $s$ -sparse polynomial of individual degree  $d$  which is  $x_i$ -reverse monic, for some  $i \in [n]$ . If  $f = g^e$  for some polynomial  $g \in \mathbb{F}[x_1, \dots, x_n]$  and  $e \in \mathbb{N}$ , then  $g$  is  $s^{d/e+1}$ -sparse.*

*Proof.* We can write  $g$  as  $g = f^{1/e} = (1 + (f - 1))^{1/e}$ . By Newton's Binomial Theorem, this gives

$$g = (1 + (f - 1))^{1/e} = \sum_{i=0}^{\infty} \binom{1/e}{i} (f - 1)^i. \quad (4.1)$$

We first focus on the case when  $\text{char}(\mathbb{F})$  is either zero or it does not divide  $e$ . In that case  $1/e$  is well defined in  $\mathbb{F}$  and so are all the binomial coefficients appearing in (4.1). Since  $f$  is reverse-monic in  $x_i$ ,  $f|_{x_i=0} = 1$ . This means that  $(f - 1)$  has  $x_i$ -degree  $\geq 1$ . Since  $g$  has  $x_i$ -degree  $= d/e$ , (4.1) becomes a finite sum modulo the ideal  $\langle x_i \rangle^{d/e+1}$ . Thus,

$$g = \sum_{i=0}^{d/e} \binom{1/e}{i} (f - 1)^i \pmod{\langle x_i \rangle^{d/e+1}}. \quad (4.2)$$

Since  $\|f - 1\| \leq s$ , it is easy to see that  $G = \sum_{i=0}^{d/e} \binom{1/e}{i} (f - 1)^i$  is  $s^{d/e+1}$ -sparse. Since  $g = G \pmod{\langle x_i \rangle^{d/e+1}}$  and going  $\pmod{\langle x_i \rangle^{d/e+1}}$  can only decrease sparsity, we get that  $g$  is also  $s^{d/e+1}$ -sparse.

Now we handle the case when  $\text{char}(\mathbb{F})$  divides  $e$ . Let  $p$  be the characteristic of  $\mathbb{F}$ , for some prime  $p$ . Let  $e = p^k \cdot q$ , where  $p^k$  is the highest power of  $p$  which divides  $e$ , for some integer  $k \geq 1$  and  $p \nmid q$ . Then by the famous *Frobenius endomorphism*, we know that:

$$\begin{aligned} g(x_1, \dots, x_n)^p &= g(x_1^p, \dots, x_n^p) \\ g(x_1, \dots, x_n)^{p^k \cdot q} &= \left( g(x_1^{p^k}, \dots, x_n^{p^k}) \right)^q. \end{aligned} \quad (4.3)$$

Since  $f = g^e = g^{p^k \cdot q}$ , we can use the variable transformation  $y_j \leftarrow x_j^{p^k}$ , for each  $j \in [n]$  to get that

$$f = g(y_1, \dots, y_n)^q.$$

Observe that  $x_i$ -degree in every non-zero monomial of  $f$  is a multiple of  $e = p^k \cdot q$ , therefore  $f$  is a proper polynomial in  $\mathbb{F}[y_1, \dots, y_n]$ . Moreover  $f$  is still  $s$ -sparse as the transformation does not affect sparsity. We further note that if  $f$  was reverse-monic in  $x_i$ , it will also be reverse-monic in  $y_i$ . Thus, we have reduced to the previous case, since  $p$  does not divide  $q$ . Moreover, the individual degree of  $f$  is now reduced, specifically  $y_i$ -degree of  $f$  is  $d' = d/p^k$ . This implies that  $g(y_1, \dots, y_n)$  has sparsity  $\leq s^{d'/q+1} = s^{d/e+1}$ . Since this transformation does not affect sparsity, we deduce that our original  $g$  is also  $s^{d/e+1}$ -sparse.  $\square$

**Remark 4.1.2.** Lemma 4.1.1 is also true for a monic  $f$  of sparsity  $s$  such that  $f = g^e$ . This is because we can convert a monic  $f$  into a reverse-monic  $\hat{f}$  by the reversal transformation,  $\hat{f} = \text{rev}_i^d[f]$  (see Definition 2.2.10). Observe that if  $f$  is monic in  $x_i$ , then  $\hat{f}$  is reverse-monic in  $x_i$ . By definition, this transformation is invertible. In fact,  $f = \text{rev}_i^d[\hat{f}]$ , thus given  $\hat{f}$ , we can recover  $f$ . We also get that  $\|f\| = \|\hat{f}\| = s$ . Since  $\hat{f} = \hat{g}^e$  and  $\hat{g}$  is  $s^{d/e+1}$ -sparse using Lemma 4.1.1, we also get that  $g$  is  $s^{d/e+1}$ -sparse.

In fact, Lemma 4.1.1 gives rise to an algorithm to compute the  $e^{\text{th}}$  root of a reverse-monic  $f$ , as shown below.

**Lemma 4.1.3.** *Let  $f \in \mathbb{F}[x_1, \dots, x_n]$  be an  $s$ -sparse polynomial of individual degree  $d$  which is  $x_i$ -reverse monic for some  $i \in [n]$ . If  $f = g^e$  for some polynomial  $g \in \mathbb{F}[x_1, \dots, x_n]$  and  $e \in \mathbb{N}$ , then there is a deterministic algorithm to compute  $g$  in  $\text{poly}(s^{d/e}, n, d)$   $\mathbb{F}$ -operations.*

*Proof.* Algorithm 1 below computes the required  $g$  when  $f = g^e$ , for some reverse-monic  $f$ .

**Correctness:** Follows from Lemma 4.1.1.

**Time Complexity:** Steps 2 to 6, except Step 4 can be done in  $O(d)$  time. Step 4 will take  $\text{poly}(s, n, d)$  time as  $f$  is  $s$ -sparse. Steps 8 and 9, each take  $\text{poly}(s^{d/e})$   $\mathbb{F}$ -operations as  $\|g\| \leq \|G\| \leq s^{d/e+1}$ . Thus, total complexity is  $\text{poly}(s^{d/e}, n, d)$   $\mathbb{F}$ -operations.  $\square$

---

**Algorithm 1:** To compute  $e^{th}$  root of  $f$ :

---

**Input:** Polynomial  $f \in \mathbb{F}[x_1, \dots, x_n]$  of individual degree  $\leq d$ ,  $s$ -sparse and reverse-monic in variable  $x_i$  such that  $f = g^e$ .

**Output:** Root  $g$ .

```

1 Let  $p := \text{char}(\mathbb{F})$ .
2 if  $p > 0$  and  $p \mid e$  then
3   Let  $e = p^k \cdot q$ , where  $p^k$  is the highest power of  $p$  that divides  $e$  and  $p \nmid q$ .
4    $f \leftarrow f(x_1^{1/p^k}, \dots, x_n^{1/p^k})$ .
5    $d \leftarrow d/p^k$ .
6    $e' \leftarrow e$  and  $e \leftarrow e/p^k$ . /* Saving value of  $e$  in  $e'$  and updating  $e$  to  $q$  */
7 end
8  $G \leftarrow \sum_{i=0}^{d/e} \binom{1/e}{i} (f-1)^i$ .
9  $g \leftarrow G \pmod{x_i^{d/e+1}}$ .
10 if  $p > 0$  and  $p \mid e'$  then
11    $g \leftarrow g(x_1^{p^k}, \dots, x_n^{p^k})$ .
12 end
13 return  $g$ .
```

---

**Remark.** Using Remark 4.1.2, Lemma 4.1.3 also works for a monic  $f$  by first making it reverse-monic, computing its  $e^{th}$  root and then returning the reversal of that.

#### 4.1.2 General Case

Now, we handle the case where input  $f$  is not monic or reverse-monic in any variable. In this case, we are not able to compute the exact root, but we can solve the decision version of this problem, that is we show how to efficiently test if  $f = g^e$ , for some  $g$  and  $e \geq 1$ .

We first give a standard trick to convert a polynomial into a reverse-monic polynomial. The properties mentioned below are fairly straightforward to prove, see for example [BSV20, Lem 5.5]. For the sake of convenience, we slightly abuse the notation  $\mathbf{x}$  to denote the set  $\{x_1, \dots, x_n\} \setminus \{x_i\}$  throughout Section 4.1.2.

**Definition 4.1.4** (Reverse-monic transformation). *Let  $f \in \mathbb{F}[x_1, \dots, x_n]$  be an  $s$ -sparse polynomial of individual degree at most  $d$ . Pick any variable  $x_i \in \text{supp}(f)$  such that*

$f|_{x_i=0} \neq 0$ . Set  $f_0 := f|_{x_i=0}$ . We define

$$\hat{f} = \frac{1}{f_0} \cdot f|_{x_i=f_0 \cdot y}.$$

This transformation has some nice properties:

1.  $\hat{f}$  is reverse-monic in  $y$ . Moreover  $\hat{f}$  is a proper polynomial in  $\mathbb{F}[\mathbf{x}][y]$  (instead of a rational function in  $\mathbb{F}(x)[y]$ ).
2.  $\|\hat{f}\| \leq s^d$ .
3. Individual degree of  $\hat{f}$  is at most  $d^2$ . However,  $\deg_y(\hat{f}) = \deg_{x_i}(f) \leq d$ .

We remark that it could be the case that the trailing coefficient  $f_0 = 0$  for every  $x_i$  above. We show how to handle that case in Step 3 of Algorithm 2. So without loss of generality, we can always convert our polynomial  $f$  into reverse-monic  $\hat{f}$ .

By definition of this transformation, one can easily show that if  $f = g^e$ , then  $\hat{f} = h^e$ , for some suitable  $h$ . However, the converse is not always true. For example, consider  $f(x, z) = z(x+1)^2$ . It is not an exact power but if we make it reverse-monic w.r.t.  $x$  we get  $\hat{f}(y, z) = 1/z \cdot f(zy, z)$ . It turns out to be  $\hat{f} = (zy+1)^2$ , which is an exact power. For testing whether  $f$  is an exact power, we need a converse also. In the two claims below, we find the extra condition on trailing coefficient, which gives us a suitable converse. This will amount to a recursive algorithm for exact power testing in Algorithm 2.

**Claim 4.1.5** ( $\Rightarrow$ ). *If  $f = g^e$  in  $\mathbb{F}[\mathbf{x}, x_i]$ , then  $\hat{f} = h^e$  in  $\mathbb{F}[\mathbf{x}, y]$  for some polynomial  $h$  and  $f_0 = g_0^e$  in  $\mathbb{F}[\mathbf{x}]$ , where  $g_0 := g|_{x_i=0}$ .*

*Proof.* Let  $f = f_k \cdot x_i^k + \dots + f_1 \cdot x_i + f_0$  and  $g = g_m \cdot x_i^m + \dots + g_1 \cdot x_i + g_0$ . If  $f = g^e$ , then  $k = em$  and  $f_0 = g_0^e$ . Thus,

$$\begin{aligned} \hat{f} &= \frac{1}{f_0} \cdot f(\mathbf{x}, f_0 \cdot y) = \frac{1}{f_0} \cdot g(\mathbf{x}, f_0 \cdot y)^e \\ &= \left( \frac{g(\mathbf{x}, f_0 \cdot y)}{g_0} \right)^e = h^e, \end{aligned}$$

for  $h := \frac{g(\mathbf{x}, f_0 \cdot y)}{g_0}$ . By definition of the reverse-monic transformation,  $\hat{f} \in \mathbb{F}[\mathbf{x}, y]$  is a proper

polynomial in this ring. Clearly,  $h$  is in  $\mathbb{F}(\mathbf{x})[y]$  by definition. Also  $\hat{f} = h^e \in \mathbb{F}[\mathbf{x}, y]$ , therefore  $h$  also belongs to  $\mathbb{F}[\mathbf{x}, y]$ , by Lemma 2.2.7.  $\square$

**Claim 4.1.6** ( $\Leftarrow$ ). *If  $\hat{f} = h^e$  in  $\mathbb{F}[\mathbf{x}, y]$  and  $f_0 = b^e$  in  $\mathbb{F}[\mathbf{x}]$ , then  $f = g^e$  in  $\mathbb{F}[\mathbf{x}, x_i]$ .*

*Proof.* Observe that,

$$\begin{aligned} f(\mathbf{x}, x_i) &= f_0 \cdot \hat{f}\left(\mathbf{x}, \frac{x_i}{f_0}\right) = f_0 \cdot h\left(\mathbf{x}, \frac{x_i}{f_0}\right)^e \\ &= \left(b \cdot h\left(\mathbf{x}, \frac{x_i}{f_0}\right)\right)^e = (g(\mathbf{x}, x_i))^e, \end{aligned}$$

for  $g := b \cdot h(\mathbf{x}, \frac{x_i}{f_0})$ . Clearly,  $g \in \mathbb{F}(\mathbf{x})[x_i]$  from above. But since  $f \in \mathbb{F}[\mathbf{x}][x_i]$  and  $f = g^e$ , this implies  $g \in \mathbb{F}[\mathbf{x}, x_i]$  by Lemma 2.2.7.  $\square$

Let  $r_{\mathbb{F}}(a, e)$  denote the time complexity of deciding whether  $a = b^e$ , for  $a, b \in \mathbb{F}$  and for some  $e \in \mathbb{N}$ . Then,

- For a finite field  $\mathbb{F} = \mathbb{F}_q$ ,  $r_{\mathbb{F}} = \text{poly}(\log q)$   $\mathbb{F}$ -operations (Lemma 4.1.8).
- For the field of rationals  $\mathbb{F} = \mathbb{Q}$ ,  $r_{\mathbb{F}} = \text{poly}(e, \log a)$   $\mathbb{F}$ -operations. For an integer (or rational number), it is easy to even compute the  $e$ -th root by binary search, or one can simply invoke univariate factorization ([LLL82]) for  $x^e - a$  to compute  $a^{1/e}$ .

We finally come to our main theorem of the chapter.

**Theorem 4.1.7.** *Let  $f \in \mathbb{F}[x_1, \dots, x_n]$  be an  $s$ -sparse polynomial of individual degree  $d$ . There is a deterministic algorithm to test whether  $f = g^e$  for some polynomial  $g \in \mathbb{F}[x_1, \dots, x_n]$  and  $e \in \mathbb{N}$ . The algorithm takes  $\text{poly}(s^{d^2}, n, d) + r_{\mathbb{F}}(f(0, \dots, 0), e)$   $\mathbb{F}$ -operations.*

*Proof.* The  $e = 1$  case is trivial. Run the Algorithm 2 below for each  $e \in \{2, \dots, d\}$ . If any such  $e$  exists such that  $f = g^e$ , that is Algorithm 2 outputs YES, then  $f$  is an exact power. Otherwise, if for every  $e$  Algorithm 2 outputs NO, then  $f$  is not an exact power.

We discuss the correctness and time-complexity of Algorithm 2 below.

---

**Algorithm 2:** Exact power testing
 

---

**Input:** An  $s$ -sparse polynomial  $f \in \mathbb{F}[x_1, \dots, x_n]$  with individual degree  $d$  and an integer  $e \in \{2, \dots, d\}$ .

**Output:** YES, if  $f = g^e$  for some polynomial  $g$  and NO, otherwise.

- 1 For each  $i \in [n]$ , check whether  $f$  is reverse-monic in variable  $x_i$ . If such an  $i$  exists, then set  $\hat{f} := f, y := x_i$  and go to Step 8 directly, else go to Step 2.
  - 2 Choose any  $i \in [n]$ . Set  $f_0 := f|_{x_i=0}$  and  $x_i := y$ .
  - 3 **if**  $f_0 = 0$  **then**
    - 4     Let  $k$  be the highest power of  $x_i$  such that  $x_i^k$  divides  $f$ .
    - 5     If  $e \nmid k$  then output NO and return, otherwise set  $f = f/x_i^k$  and  $f_0 = f|_{x_i=0}$ .
  - 6 **end**
  - 7 Define  $\hat{f} := \frac{1}{f_0} \cdot f(\mathbf{x}, f_0 \cdot y)$ .
  - 8 Invoke Algorithm 1 for  $\hat{f}$  which is reverse-monic in variable  $y$  to get candidate root  $\hat{g}$ .
  - 9 Check whether  $\hat{f} = \hat{g}^e$ , by multiplying out. If it is, go to Step 10, otherwise output NO.
  - 10 For the  $(n-1)$ -variate polynomial  $f_0 := f|_{x_i=0} \in \mathbb{F}[\mathbf{x}]$ , recursively check whether  $f_0$  is  $e^{\text{th}}$  power of some polynomial. If it is, then output YES, otherwise output NO.
- 

**Correctness:** If  $f$  is indeed equal to  $g^e$  for some  $e \in \{2, \dots, d\}$ , then by Claim 4.1.5,  $\hat{f} = h^e$  for some  $h$  and  $f_0 = b^e$ , for  $b = g|_{x_i=0}$ . Thus, by Lemma 4.1.3, Step 8 will compute the correct root  $\hat{g}$  and in Step 10, the algorithm will output YES. If  $f$  is not an exact power for any  $e \in [d]$ , the algorithm will output NO in either Step 5 or Step 9 or Step 10. This follows due to Claim 4.1.6 (consider contrapositive).

**Time Complexity:** Step 1 takes  $\text{poly}(s, n, d)$   $\mathbb{F}$ -operations as we only have to check whether  $f|_{x_i=0} = 1$  at most  $n$  times. In Steps 2-6, we are required to compute the trailing coefficient of  $f$  w.r.t  $x_i$ -variable, which takes  $\text{poly}(s, n, d)$   $\mathbb{F}$ -operations. Step 7 takes at most  $\text{poly}(s^d, n, d)$  time. Step 8 takes at most  $\text{poly}(\|\hat{f}\|^{d/e}, n, d)$   $\mathbb{F}$ -operations by Lemma 4.1.3 as  $y$ -degree of  $\hat{f}$  is still  $d$ . Since  $\|\hat{f}\| \leq s^d$ , this step takes at most  $\text{poly}(s^{d^2/e}, n, d)$   $\mathbb{F}$ -operations. Multiplying out in Step 9 will take at most  $\text{poly}(s^{(d^2/e) \cdot e}) = \text{poly}(s^{d^2})$   $\mathbb{F}$ -operations as  $\|\hat{g}\| \leq s^{d^2/e+1}$ . Note that in Step 10, we recurse on  $f_0$ , which has sparsity  $\leq s$ , therefore there is no blow-up of sparsity in the recursion. Hence, this step takes  $\text{poly}(s^{d^2}, n, d)$   $\mathbb{F}$ -operations to reach the base case of deciding whether the field element  $f(0, \dots, 0)$  has an  $e$ -th root. The total complexity is thus,  $\text{poly}(s^{d^2}, n, d) +$

$r_{\mathbb{F}}(f(0, \dots, 0), e)$   $\mathbb{F}$ -operations. □

**Remark.** Note that for a finite field  $\mathbb{F} = \mathbb{F}_q$ , the time-complexity of Algorithm 2 is  $\text{poly}(s^{d^2}, n, \log q)$  time. For the field of rationals  $\mathbb{F} = \mathbb{Q}$ , the time-complexity is  $\text{poly}(s^{d^2}, n, t)$ , where  $t$  is the bit-complexity of the coefficients of input polynomial.

Using the standard theory of finite fields, we show how to test whether a finite field element is an exact power. In other words, we show that  $r_{\mathbb{F}} = \text{poly}(\log q)$  for a finite field  $F_q$ . This is required for the base case of Algorithm 2, when working over finite fields.

**Lemma 4.1.8** (Folklore). *For a finite field  $\mathbb{F}_q$ , we can decide whether an element  $a \in \mathbb{F}_q$  is a  $k^{\text{th}}$  power residue, i.e.  $a = b^k$  for some  $b \in \mathbb{F}_q$  in  $\text{poly}(\log q)$   $\mathbb{F}_q$ -operations.*

*Proof.* We will focus on  $\mathbb{F}_q^*$  since  $0 = 0^k$  trivially. We will prove that  $a = b^k$  for some  $b \in \mathbb{F}_q^*$  and  $k \geq 1$ , if and only if  $a^{\frac{q-1}{d}} = 1$  in  $\mathbb{F}_q^*$ , where  $d = \gcd(k, q-1)$ . Having proved that, we can simply test this by computing  $a^{\frac{q-1}{d}}$  in  $\text{poly}(\log q)$   $\mathbb{F}_q$ -operations using repeated squaring. Now we prove both the directions for

$$a = b^k \Leftrightarrow a^{\frac{q-1}{d}} = 1.$$

( $\Rightarrow$ ) Observe that  $a = b^k \Rightarrow a^{\frac{q-1}{d}} = b^{\frac{k(q-1)}{d}}$ . Since  $d = \gcd(k, q-1)$ ,  $d$  divides  $k$ , hence  $\frac{k}{d}$  is an integer. Thus, we get that  $a^{\frac{q-1}{d}} = b^{\frac{k(q-1)}{d}} = 1$  by using the generalization of Fermat's Little Theorem, i.e.  $x^{q-1} = 1$  for all  $x \in \mathbb{F}_q^*$ .

( $\Leftarrow$ ) We know that  $\mathbb{F}_q^*$  is a cyclic group of order  $q-1$ . Let  $g$  be its generator such that  $a = g^r$ , for some  $r \in [q-2]$  (For  $r = 0$ , we know that  $1 = 1^k$  trivially). Now, if  $a^{\frac{q-1}{d}} = 1$ , we get that  $g^{\frac{r(q-1)}{d}} = 1$ . This implies that  $\frac{r}{d}$  is an integer or that  $d$  divides  $r$ . By Bezout's identity, we know that  $d = \gcd(k, q-1) = sk + t(q-1)$  for some integers  $s, t$ . Since  $d \mid r$ , we get that  $r = s'k + t'(q-1)$ . This proves that  $a$  is a  $k^{\text{th}}$  power residue as:

$$a = g^r = g^{s'k + t'(q-1)} = g^{s'k} = b^k$$

in  $\mathbb{F}_q^*$  for  $b = g^{s'}$ . □

## 4.2 Discussion

In this chapter, we designed a poly-time algorithm to test whether  $f = g^e$ , for some polynomial  $g$  and  $e \in \mathbb{N}$ , where  $f$  has bounded individual degree. The search version of this problem is open, i.e. compute the exact root  $g$  in poly-time. Note that Lemma 3.4.1 discussed earlier implies that we only need to prove a polynomial-size sparsity bound for  $g$  in order to compute it. Thus, the next open problem in the context of this chapter is to prove a polynomial-size sparsity bound (Conjecture 1.2.3) for the special case of exact-roots  $f = g^e$ , where  $f$  has constant individual degree  $d$ . The smallest open case here is  $d = 4$  and  $e = 2$ , in other words prove that square-root of a sparse multi-quartic polynomial is also sparse.



## Chapter 5

# Co-factor Sparsity via Unique Projections

In this chapter, we consider a variant of sparse factorization problem. In sparse factoring, we are given an input polynomial  $f \in \mathbb{F}[x_1, \dots, x_n]$ . If  $f$  factors as  $f = gh$ , we are asked to prove that *every* factor of  $f$  is sparse. Now suppose you are promised that one of the factors, say  $h$  is also sparse. Then, can one prove that  $g$  is sparse? We call  $g$  the *quotient polynomial* or *co-factor* of  $h$ . We remark that any (even non-constructive) efficient upper bound on the sparsity of  $g$  allows us to compute  $g$  efficiently by interpolating the ratio  $f/h$  using a reconstruction algorithm for sparse polynomials (e.g. [KS01]) and verifying the result.

To state our result we need the following technical definition. We say that a polynomial  $h \in \mathbb{F}[x_1, x_2, \dots, x_n]$  has a *unique projection of length  $k$*  if there exist  $k$  variables  $x_{i_1}, x_{i_2}, \dots, x_{i_k}$  and  $k$  corresponding exponents  $e_1, e_2, \dots, e_k$  such that  $h$  has a unique monomial that contains the pattern  $x_{i_1}^{e_1} x_{i_2}^{e_2} \dots x_{i_k}^{e_k}$  (see Definition 5.2.1 for more details).

*Let  $f$  be an  $s$ -sparse polynomial of individual degree at most  $d$  such that  $f = gh$ . Suppose, in addition, that  $h$  is a multilinear polynomial with a unique projection of length  $k$ . Then  $g$  is  $s^{O(dk)}$ -sparse.*

This is formally proved in Theorem 5.2.15. Note that for constant  $k$ , we get the desired  $s^{O(d)}$  bound in sparsity conjecture. We remark that Example 1.2.2 with  $d = p-1$  (resulting

in a lower bound of  $n^{\Omega(p)}$ ) showcases the tightness of our result as here  $f$  is  $n$ -sparse and  $h = x_1 + \dots + x_n$  has a unique projection of length 1 (e.g.  $x_1$ ) which results in an upper bound of  $n^{O(p)}$  for  $g$ . We can also extend the above result to the case of a co-factor of a power of a multilinear polynomial. See Theorem 5.2.19 for the formal statement. In general  $k$  may not be constant but we show that every multilinear  $s$ -sparse polynomial always has a unique projection of length  $O(\log s)$  (see Lemma 5.2.5). Using this, we obtain a new sparsity bound of size  $s^{O(d \log s)}$  for all multilinear co-factors.

*Let  $f$  be an  $s$ -sparse polynomial of individual degree  $d$  such that  $f = gh$ . Suppose, in addition, that  $h$  is a multilinear polynomial. Then  $g$  is  $s^{O(d \log s)}$ -sparse.*

This is formally proved in Corollary 5.2.16. The obtained bound is slightly better than the general sparsity bound of size  $s^{O(d^2 \log n)}$  by [BSV20] when  $s = \text{poly}(n)$ . Although our overall improvement may seem incremental (e.g. it does not allow us to “get rid” of the  $\log n$  in the exponent) our main contribution here is conceptual: identifying a combinatorial property - the length of the multilinear polynomial - that governs the bound on the sparsity of multilinear co-factors.

## 5.1 Multilinear co-Factor Motivation

The results which are informally stated above apply to the factorization scenario of  $f = gh$  where  $f$  is  $s$ -sparse and  $h$  is multilinear. First of all, note that by previous results (see [BSV20] and references within)  $h$  itself is  $s$ -sparse. So we are looking to bound the sparsity of  $g$ . As it turns out, this pattern is the “bottleneck” case for multicubic polynomials. In other words, showing a polynomial-size sparsity bound on  $g$  in this scenario would imply a polynomial-size sparsity bound on factors of general multicubic polynomials! In fact, it is sufficient to consider the case when the degree of  $g$  in every variable is exactly 2! We remark that getting polynomial-size sparsity bound is open for  $d \geq 3$ . The following lemma summarizes this formally.

**Lemma 5.1.1.** *Suppose there exists an absolute constant  $a \geq 1$  such that for any multicubic polynomial  $f$ : if  $g \mid f$  **and**  $f/g$  is multilinear then  $\|g\| \leq \|f\|^a$ . Then for any*

multicubic polynomial  $f$  if  $g \mid f$  then  $\|g\| \leq \|f\|^a$ .

*Proof.* We prove the following claim: Let  $f \in \mathbb{F}[x_1, x_2, \dots, x_n]$  be a multicubic polynomial such that  $f = uv$  and  $v \not\equiv 0$ . Then  $\|u\| \leq \|f\|^a$ . Note that the claim also covers the case when  $f = u \equiv 0$ . The proof is by induction on  $n$  (the number of variables in  $f$ ). The base case is when  $n = 0$  (i.e.  $u, f, v \in \mathbb{F}$ ) where the claim follows trivially. Suppose  $n \geq 1$ . We have the following cases to consider:

- There exists a variable  $x_i$  s.t.  $\deg_{x_i}(u) \geq 1$  but  $\deg_{x_i}(v) = 0$ . Let  $1 \leq d \leq 3$  be the degree of  $x_i$  in  $u$ . In this case we can write:

$$(u_d x_i^d + \dots + u_0)v = uv = f = f_d x_i^d + \dots + f_0.$$

Here,  $u_j, f_j$  and  $v$  do not depend on  $x_i$ . Formally:  $f_j = u_j v$  for  $j \in \{0, \dots, d\}$ . By the induction hypothesis, we have that  $\|u_j\| \leq \|f_j\|^a$  for  $j \in \{0, \dots, d\}$  and hence:

$$\|u\| = \sum_{j=0}^d \|u_j\| \leq \sum_{j=0}^d \|f_j\|^a \leq \left( \sum_{j=0}^d \|f_j\| \right)^a = \|f\|^a.$$

- There exists a variable  $x_i$  s.t.  $\deg_{x_i}(v) \geq 1$ , but  $\deg_{x_i}(u) = 0$ . Pick  $\alpha \in \mathbb{F}$  such that  $v|_{x_i=\alpha} \not\equiv 0$ . We have that:

$$u \cdot v|_{x_i=\alpha} = u|_{x_i=\alpha} \cdot v|_{x_i=\alpha} = f|_{x_i=\alpha}.$$

By the induction hypothesis:  $\|u\| \leq \|f|_{x_i=\alpha}\|^a \leq \|f\|^a$ .

- There exists a variable  $x_i$  s.t.  $\deg_{x_i}(u) = 1$ . Wlog  $\deg_{x_i}(v) \geq 1$ . We can write

$$(u_1 x_i + u_0)(v_d x_i^d + \dots + v_e x_i^e) = uv = f = (f_{d+1} x_i^{d+1} + \dots + f_e x_i^e).$$

Here,  $d > e$  and  $v_d, v_e \not\equiv 0$ . In particular, we have that  $u_1 v_d = f_{d+1}$  and  $u_0 v_e = f_e$ .

By the induction hypothesis:  $\|u\| = \|u_1\| + \|u_0\| \leq \|f_{d+1}\|^a + \|f_e\|^a \leq \|f\|^a$ .

- WLOG we are left with the case that for each  $i \in [n]$  we have that:  $\deg_{x_i}(u) = 2$  and  $\deg_{x_i}(v) = 1$ . Based on our assumption, in this case  $\|u\| \leq \|f\|^a$  and we are done.  $\square$

## 5.2 Co-Factor Polynomial Sparsity

In this section we will prove our main results on co-factor sparsity in Theorem 5.2.15 and Corollary 5.2.16. We begin with some technical definitions.

### 5.2.1 Unique Projections

**Definition 5.2.1.** Let  $V \subseteq \mathbb{N}^n$ . A unique projection of  $V$  of length  $k$  is a set  $\{(i_1, e_1), \dots, (i_k, e_k)\}$  such that there exists a unique vector  $\mathbf{v} \in V$  satisfying  $\forall j \in [k] : v_{i_j} = e_j$ .

A unique projection of a polynomial  $h \in \mathbb{F}[x_1, x_2, \dots, x_n]$  is defined as a unique projection of  $\text{supp}(h)$ .

In other words, there exists a unique monomial in the monomial representation of  $h$  that contains the pattern  $x_{i_1}^{e_1} x_{i_2}^{e_2} \dots x_{i_k}^{e_k}$ . The following is immediate from the definition.

**Observation 5.2.2.** Let  $h \in \mathbb{F}[x_1, x_2, \dots, x_n]$  be a polynomial and let  $\{(i_1, e_1), \dots, (i_k, e_k)\}$  be a unique projection of  $h$ . Pick  $j \in [k]$  and let  $\ell \geq e_j$ . Then

$$\{(i_1, e_1), (i_2, e_2), \dots, (i_{j-1}, e_{j-1}), (i_j, \ell - e_j), (i_{j+1}, e_{j+1}), \dots, (i_k, e_k)\}$$

is a unique projection of  $\text{rev}_{i_j}^\ell[h]$ .

Subsequently, we demonstrate the usefulness of unique projections.

**Lemma 5.2.3.** Let  $h \in \mathbb{F}[x_1, x_2, \dots, x_n]$  be a polynomial with a unique projection of the form  $\{(i_1, 0), (i_2, 0), \dots, (i_k, 0)\}$  (i.e.  $\forall j \in [k] : e_k = 0$ ). Then  $h$  is  $\{i_1, i_2, \dots, i_k\}$ -reverse pseudo-monic.

*Proof.* Let  $I = \{i_1, i_2, \dots, i_k\}$ . By iterative application of Part 1 of Observation 2.2.9, we obtain that

$$\text{supp}(h|_{x_I=0_I}) = \{\mathbf{e} \in \text{supp}(h) \mid \forall j \in [k] : e_{i_j} = 0\}$$

As  $I$  corresponds to a unique projection, the set of the RHS contains exactly one vector and the claim follows from Part 2 of Observation 2.2.9.  $\square$

**Lemma 5.2.4.** Let  $h \in \mathbb{F}[x_1, x_2, \dots, x_n]$  be a multilinear polynomial and let  $\{(i_1, e_1), \dots, (i_k, e_k)\}$  be a unique projection of  $h$ . Furthermore, let  $J = \{i_j \mid e_j = 1\}$ . That is, the set of all

indices  $i_j$  for which  $e_j = 1$ . Then  $\tilde{h} := \text{rev}_J^1[h]$  is  $\{i_1, i_2, \dots, i_k\}$ -reverse pseudo-monic.

*Proof.* First, note that since  $h$  is a multilinear polynomial, we have that  $e_j = 0$  for indices  $j \in \{i_1, \dots, i_k\} \setminus J$ . Subsequently, by iterative application of Observation 5.2.2, we obtain that  $\tilde{h}$  is a multilinear polynomial with a unique projection  $\{(i_1, 0), (i_2, 0), \dots, (i_k, 0)\}$ . Note that  $\tilde{h}$  is a proper polynomial (and not a rational function) by iterative application of Part 1 in Lemma 2.2.11. The claim then follows from Lemma 5.2.3.  $\square$

We conclude this section by showing that every set contains a unique projection of (at most) logarithmic size and a relation of unique projections with  $\delta$ -min-entropy polynomials that were defined in [BS22].

**Lemma 5.2.5.** *Let  $V \subseteq \mathbb{N}^n$  be a set of size  $|V| \leq s$ . Then  $V$  has a unique projection of length at most  $\log s + 1$ .*

*Proof.* The proof is by induction on the size of  $V$ . For the base case  $|V| = 1$  there exists a unique projection of length 1. Now assume  $|V| \geq 2$ . Therefore,  $V$  contains at least two different vectors  $\mathbf{u} \neq \mathbf{w}$ . Let  $i$  be such that  $u_i \neq w_i$ . Let us denote  $a = u_i$  and  $b = w_i$ . Partition  $V$  into  $V_a := \{\mathbf{v} \in V \mid v_i = a\}$  and  $V_b := \{\mathbf{v} \in V \mid v_i = b\}$ . We have that  $|V_a| + |V_b| \leq |V|$ . Hence,  $\log 1 \leq |V_a| \leq s/2$ . By the induction hypothesis,  $V_a$  has a unique projection of length at most  $\log(s/2) + 1 = \log s$ . We now add the index  $i$  and  $e_i = a$  to the set to obtain a unique projection for  $V$  of size  $\log s + 1$ .  $\square$

Recently, [BS22] defined a class of polynomials called ‘low min-entropy’ polynomials and showed an  $(nd)^{O(d\delta)}$  sparsity upper bound for the factors of a  $\delta$ -min-entropy polynomial. We quickly give their definition of a  $\delta$ -min-entropy set and then show a combinatorial connection of min-entropy with our notion of unique projections below. We note this connection between these two combinatorial concepts but our results are incomparable from those in [BS22].

**Definition 5.2.6** ([BS22]). *A vector  $\mathbf{v} \in \mathbb{N}^n$  has min-entropy  $\delta$  if it has the same value in  $(n - \delta)$  of its coordinates. A set  $V \subseteq \mathbb{N}^n$  is called a  $\delta$ -min-entropy set if for every  $\mathbf{v} \in V$ ,  $\mathbf{v}$  has min-entropy  $\leq \delta$ .*

**Lemma 5.2.7.** *Let  $V \subseteq \mathbb{N}^n$  be a  $\delta$ -min-entropy set. Then  $V$  has a unique projection of length at most  $2\delta + 1$ .*

*Proof.* Let  $\mathbf{u} \in V$  be a vector with maximum min-entropy in  $V$ . Let  $m(\mathbf{u})$  denote the majority value in  $\mathbf{u}$ . In other words,  $\mathbf{u}$  has the largest number of non-majority values among all vectors in  $V$ . Let  $u_{i_1}, \dots, u_{i_k}$  be all the non-majority values in  $\mathbf{u}$ . Since  $\mathbf{u}$  has min-entropy  $\leq \delta$ , the length of this sequence  $k$  is at most  $\delta$ . We note that the remaining elements of  $\mathbf{u}$  outside this sequence have the same value (equal to  $m(\mathbf{u})$ ). We extend the sequence to length  $k + \delta + 1$  by adding any  $\delta + 1$  elements from these remaining elements of  $\mathbf{u}$  to get:  $u_{i_1}, \dots, u_{i_k}, u_{i_{k+1}}, \dots, u_{i_{k+\delta+1}}$ . We claim that  $\{(i_1, u_{i_1}), \dots, (i_{k+\delta+1}, u_{i_{k+\delta+1}})\}$  is a unique projection of the set  $V$ .

We need to show that if  $\mathbf{v}$  is another vector in  $V$  such that  $v_{i_j} = u_{i_j}$ , for each  $j \in [k + \delta + 1]$  (values agree on projection), then  $\mathbf{v} = \mathbf{u}$ . Observe that  $u_{i_{k+1}} = \dots = u_{i_{k+\delta+1}} = m(\mathbf{u})$ , by definition of  $k$ . This means  $v_{i_{k+1}} = \dots = v_{i_{k+\delta+1}} = m(\mathbf{u})$  also. We deduce that at least  $\delta + 1$  coordinates of  $\mathbf{v}$  have value  $m(\mathbf{u})$ . We also know that  $\mathbf{v}$  has min-entropy  $\leq \delta$  and note that for a  $\leq \delta$ -min-entropy vector, if any  $\delta + 1$  coordinates have the same value, that value is the majority value. Hence,  $m(\mathbf{v}) = m(\mathbf{u})$ . Now suppose for the sake of contradiction that there exists some coordinate  $i_r$  outside the projection ( $r > k + \delta + 1$ ) for which  $v_{i_r} \neq u_{i_r}$ . Since all the non-majority values of  $\mathbf{u}$  have already appeared in the projection coordinates, we deduce that  $u_{i_r} = m(\mathbf{u}) = m(\mathbf{v})$ . This means that  $v_{i_r} \neq m(\mathbf{v})$ . In that case,  $v_{i_r}$  is another element in  $\mathbf{v}$  apart from  $v_{i_1}, \dots, v_{i_k}$  which is distinct from  $m(\mathbf{v})$ . This is a contradiction to our assumption that  $\mathbf{u}$  is a vector with maximum min-entropy. Hence,  $v_{i_j} = u_{i_j}$  for all  $j > k + \delta + 1$ . By our premise, they also agree on the  $k + \delta + 1$  projection coordinates. Hence  $v_j = u_j$ , for all  $j \in [n]$  and thus,  $\mathbf{v} = \mathbf{u}$ . Moreover, since  $k \leq \delta$ , length of this unique projection is  $k + \delta + 1 \leq 2\delta + 1$ .  $\square$

### 5.2.2 Co-factor sparsity bounds

We now state and prove the technical results of this section in increasing order of generality. We will culminate in our main results towards the end in Theorem 5.2.15 and Corollary 5.2.16. In what follows, let  $f, h \in \mathbb{F}[x_1, x_2, \dots, x_n]$  be two  $s$ -sparse polynomials

such that  $f = gh$ .

**Lemma 5.2.8.** *Suppose that  $h$  is reverse monic and the individual degree of  $g$  is at most  $d$ . Then  $g$  is  $s^{d+2}$ -sparse.*

*Proof.* By hypothesis, let  $h$  be reverse monic w.r.t. some variable  $x_i \in \text{supp}(h)$ . Express  $h$  as a univariate in  $x_i$  with coefficients as polynomials in the remaining variables. Since  $h$  is  $x_i$ -reverse monic, the constant term,  $h|_{x_i=0}$  is 1. Therefore, every term in  $(1-h)$  has  $x_i$ -degree  $\geq 1$ . We use this observation in a division-elimination argument as follows:

$$g = \frac{f}{h} = \frac{f}{1 - (1-h)} = \sum_{j=0}^{\infty} f(1-h)^j. \quad (5.1)$$

Let  $x_i$ -degree of  $g$  be  $d_i$ . Then, we can safely truncate the infinite sum in Equation (5.1) as follows:

$$g = \sum_{j=0}^{d_i} f(1-h)^j \bmod \langle x_i^{d_i+1} \rangle. \quad (5.2)$$

Equation (5.2) helps us in bounding sparsity of  $g$ . Note that going  $\bmod \langle x_i^{d_i+1} \rangle$  can only decrease sparsity, so we focus only on the sparsity of finite sum in (5.2). Since  $g$  is a factor of  $f$ , its individual degree  $d_i$  is also upper bounded by  $d$ . Also note that both  $\|f\|, \|(1-h)\| \leq s$ . Therefore, we get that  $\|g\| \leq \sum_{j=0}^d s^{j+1} \leq s^{d+2}$ .  $\square$

Generalizing this observation we obtain:

**Lemma 5.2.9.** *Suppose that  $h$  is  $I$ -reverse monic and the individual degrees of the variables of  $g$  in  $x_I$  are at most  $d$ . Then  $g$  is  $s^{d|I|+2}$ -sparse.*

*Proof.* We follow the same template as in proof of Lemma 5.2.8, with the change that  $h$  is reverse monic with respect to a set  $I$  of variables instead of just a single variable. Express  $h$  as a polynomial in  $x_I$  variables with coefficients as polynomials in the remaining  $n - |I|$  variables. Since  $h$  is  $I$ -reverse monic,  $h|_{x_I=0_I}$  (the constant term of  $h$ ) is 1. Therefore, every term in  $(1-h)$  has total  $x_I$ -degree  $\geq 1$ . We then get the same Equation (5.1) for  $g$ . Let  $I = \{i_1, \dots, i_k\} \subseteq [n]$ , where  $k = |I|$ . Let individual degree of variable  $x_{i_j}$  in  $g$  be  $d_j$

for each  $j \in [k]$ . Then, we can truncate the infinite sum as follows:

$$g = \sum_{j=0}^{dk} f(1-h)^j \bmod \langle x_{i_1}^{d_1+1}, \dots, x_{i_k}^{d_k+1} \rangle. \quad (5.3)$$

By the premises, for each  $j \in [k]$  each individual degree  $d_j$  is upper bounded by  $d$ . Therefore, we only need to sum up to  $j = dk$  in (5.3) as the total degree in  $x_I$  variables is upper bounded by  $dk$ . Therefore, we get that  $\|g\| \leq \sum_{j=0}^{dk} s^{j+1} \leq s^{dk+2} = s^{d|I|+2}$ .  $\square$

The next lemma transforms a pseudo-monic polynomial into a monic polynomial while maintaining the sparsity and the multiplicative properties.

**Lemma 5.2.10.** *Let  $f = gh$ . Suppose that  $h$  is  $I$ -reverse pseudo-monic and the individual degrees of the variables of  $g$  in  $x_I$  are at most  $d$ . Then there exists polynomials  $\tilde{f}, \tilde{g}, \tilde{h} \in \mathbb{F}[x_1, x_2, \dots, x_n]$  such that:*

1.  $\tilde{h}$  is  $I$ -reverse monic.
2.  $\tilde{f} = \tilde{g}\tilde{h}$ .
3.  $\|\tilde{f}\| = \|f\|, \|\tilde{g}\| = \|g\|, \|\tilde{h}\| = \|h\|$ .
4. The individual degrees of the variables of  $\tilde{g}$  in  $x_I$  are at most  $d$ .

*Proof.* Let  $\alpha := h|_{x_I=0_I}$ . We first define  $\hat{f}, \hat{g}$  and  $\hat{h}$  by setting  $x_i := x_i \cdot \alpha$  for all  $i \in I$ , into  $f, g$  and  $h$ , respectively. Next, we set  $\tilde{f} := \hat{f}$ ,  $\tilde{g} := \hat{g} \cdot \alpha$  and  $\tilde{h} = \hat{h}/\alpha$ . We will now prove each part of the claim.

1. First, observe that  $\tilde{h}$  is, indeed, a polynomial (and not a rational function). This is due to the fact that  $\alpha$  divides  $\hat{h}$ . Next,  $\tilde{h}|_{x_I=0_I} = \hat{h}|_{x_I=0_I}/\alpha = h|_{x_I=0_I}/\alpha = 1$ .
2.  $\tilde{f} = \hat{f} = \hat{g}\hat{h} = (\hat{g} \cdot \alpha)(\hat{h}/\alpha) = \tilde{g}\tilde{h}$ .
3. Since  $\alpha$  is a monomial or a field element there is 1 – 1 correspondence between the monomials of  $f, g, h$  and  $\tilde{f}, \tilde{g}, \tilde{h}$ , respectively.
4. By definition,  $\alpha \in \mathbb{F}[x_{[n] \setminus I}]$ . Hence, multiplication or division by  $\alpha$  does not affect the degrees of the variables in  $I$ .  $\square$



By transforming a pseudo-monic polynomial into a monic polynomial we can generalize Lemma 5.2.9 to the pseudo-monic case.

**Corollary 5.2.11.** *Suppose that  $h$  is  $I$ -reverse pseudo-monic and the individual degrees of the variables of  $g$  in  $x_I$  are at most  $d$ . Then  $g$  is  $s^{d|I|+2}$ -sparse.*

*Proof.* Apply Lemma 5.2.9 on  $\tilde{f}, \tilde{g}$  and  $\tilde{h}$  from Lemma 5.2.10. We obtain that  $\tilde{g}$  and hence  $g$  is  $s^{d|I|+2}$ -sparse.  $\square$

**Remark 5.2.12.** In the context of exact-root sparsity, we can extend the result of Lemma 4.1.1 from the reverse monic to the  $I$ -reverse pseudo-monic case. It is done in the exact same fashion as we moved from Lemma 5.2.8 to Corollary 5.2.11 above. The formal statement is given in Theorem 5.2.13 below. In addition, we observe that if  $f = g^e$  then  $f$  is  $I$ -reverse pseudo-monic iff  $g$  is  $I$ -reverse pseudo-monic (for the exact same  $I$ ).

**Theorem 5.2.13.** *Let  $f \in \mathbb{F}[x_1, x_2, \dots, x_n]$  be a polynomial of sparsity  $s$  and individual degree at most  $d$  such that  $f = g^e$  for some (other) polynomial  $g \in \mathbb{F}[x_1, x_2, \dots, x_n]$  and  $e \in \mathbb{N}$ . In addition, suppose that  $f$  is  $I$ -reverse pseudo-monic for some  $I \subseteq [n]$ . Then the sparsity of  $g$  is bounded by  $s^{O(d \cdot |I|/e)}$ .*

Any multilinear factor of a sparse polynomial is also sparse. This is a known result, which can be found, for example in [BSV20].

**Lemma 5.2.14.** *Let  $f, h \in \mathbb{F}[x_1, x_2, \dots, x_n]$  where  $h$  is a multilinear polynomial and  $h \mid f$ . Then  $\|h\| \leq \|f\|$ .*

Now we finally prove the main results of this chapter.

**Theorem 5.2.15.** *Let  $f \in \mathbb{F}[x_1, \dots, x_n]$  be an  $s$ -sparse polynomial, with a multilinear factor  $h$  such that  $f = g \cdot h$ . Suppose that the individual degree of  $g$  is at most  $d$  and  $h$  has a unique projection of length at most  $k$ . Then  $g$  is  $s^{dk+2}$ -sparse.*

*Proof.* Let  $\{(i_1, e_1), (i_2, e_2), \dots, (i_k, e_k)\}$  be the guaranteed unique projection of  $h$  and let  $J = \{i_j \mid e_j = 1\}$ . We define:

$$\tilde{f} := \text{rev}_J^{d+1}[f], \quad \tilde{g} := \text{rev}_J^d[g] \quad \text{and} \quad \tilde{h} := \text{rev}_J^1[h].$$

By Lemma 2.2.11, we have that  $\tilde{f} = \tilde{g} \cdot \tilde{h}$ , where  $\tilde{f}$  is an  $s$ -sparse polynomial,  $\tilde{g}$  is a polynomial with individual degree at most  $d$  and  $\tilde{h}$  is a multilinear polynomial. Furthermore, by Lemma 5.2.14,  $\|\tilde{h}\| \leq \|\tilde{f}\| \leq s$ . Finally, by Lemma 5.2.4,  $\tilde{h}$  is  $\{i_1, i_2, \dots, i_k\}$ -reverse pseudo-monic. Consequently, by Corollary 5.2.11, we obtain that  $\tilde{g}$  and hence  $g$  are  $s^{dk+2}$ -sparse.  $\square$

**Corollary 5.2.16.** *Let  $f \in \mathbb{F}[x_1, \dots, x_n]$  be an  $s$ -sparse polynomial, such that  $f = g \cdot h$  where  $h$  is multilinear polynomial and  $g$  is a polynomial with individual degree at most  $d$ . Then  $g$  is  $s^{d(\log s + 1) + 2}$ -sparse.*

*Proof.* By Lemma 5.2.14,  $\|h\| \leq \|f\| \leq s$ . Consequently, by Lemma 5.2.5,  $h$  has a unique projection length at most  $\log s + 1$ . Further using Theorem 5.2.15, we deduce that  $\|g\| \leq s^{d(\log s + 1) + 2}$ .  $\square$

Similarly, by plugging in Lemma 5.2.7 into Theorem 5.2.15 we obtain the following relation to low min-entropy polynomials.

**Corollary 5.2.17.** *Let  $f \in \mathbb{F}[x_1, \dots, x_n]$  be an  $s$ -sparse polynomial, such that  $f = g \cdot h$  where  $h$  is a  $\delta$ -min-entropy, multilinear polynomial and  $g$  is a polynomial with individual degree at most  $d$ . Then  $g$  is  $s^{d(2\delta+1)+2}$ -sparse.*

**Remark 5.2.18.** By using the formal expansion:

$$\frac{1}{(1-x)^\ell} = \sum_{j=0}^{\infty} \binom{j+\ell-1}{j} x^j$$

for division elimination in the proof of Lemma 5.2.9, we can get somewhat stronger versions of our results below.

**Theorem 5.2.19.** *Let  $f \in \mathbb{F}[x_1, x_2, \dots, x_n]$  be a polynomial of sparsity  $s$  and individual degree at most  $d$  such that  $f = gh^\ell$  for some  $\ell \in \mathbb{N}$ . Suppose, in addition, that  $h$  is a multilinear polynomial with a unique projection of length  $k$ . Then the sparsity of  $g$  is bounded by  $s^{O((d-\ell)k)}$ .*

### 5.3 Discussion

In this chapter, we discussed the problem of bounding sparsity for the quotient polynomial  $f/h$ , where  $f$  is  $s$ -sparse and  $h$  is a multilinear polynomial. We prove  $s^{O(d \log s)}$  bound here, where  $n$  is the number of variables. Can one improve this sparsity bound? Ideally, get rid of the  $\log s$  term in the exponent. One can start by studying the structure of multilinear polynomials with non-constant or log-sized unique projections. In a different direction, can one extend our  $s^{O(d \log s)}$  bound for any quotient polynomial  $f/h$ , where both  $f, h$  are  $s$ -sparse polynomials of individual degree  $d$  and  $h$  can be non-multilinear. Note that Lemma 5.2.5 is proved for any sparse polynomial, not just for multilinear polynomials. However, transforming  $h$  to a reverse pseudo-monic  $\tilde{h}$  step breaks down (see Lemma 5.2.4), as the reversal operation for a non-multilinear  $h$  can produce a rational function  $\tilde{h}$  instead of a polynomial.



## Part II

# Polynomial Identity Testing



## Chapter 6

# PIT Preliminaries

### 6.1 Notations

We require some new notations for Part-II of the thesis, in addition to those used in Part-I (Section 2.1). We borrow some of the notations here from [GKST17]. Let  $\mathbf{x}_k$  or  $\mathbf{x}_{\leq k}$  denote the tuple of first  $k$  variables  $(x_1, x_2, \dots, x_k)$  and  $\mathbf{x}_{>k}$  denote the tuple of remaining variables  $(x_{k+1}, x_{k+2}, \dots, x_n)$ . Let  $\pi$  denote the *variable order* of an ROABP, where  $\pi : [n] \rightarrow [n]$  is some permutation. This means the variables are read in the order  $(x_{\pi(1)}, x_{\pi(2)}, \dots, x_{\pi(n)})$ . Let  $\mathbb{F}^{w \times w}[\mathbf{x}]$  denote the ring of polynomials in  $n$ -variables over the matrix algebra of  $w \times w$  matrices.

Let  $A(\mathbf{x}) \in \mathbb{F}[\mathbf{x}]$  be a polynomial in  $n$  variables of degree  $d$ . For an exponent vector  $\mathbf{a}$  let  $|\mathbf{a}|_1$  denote the degree of the monomial  $\mathbf{x}^{\mathbf{a}}$ . Let  $\text{coeff}(A)(\mathbf{x}^{\mathbf{a}}) \in \mathbb{F}$  denote the coefficient of the monomial  $\mathbf{x}^{\mathbf{a}}$  in  $A(\mathbf{x})$ . We use  $A^{[d]}$  to denote the degree- $d$  *homogeneous part* of  $A(\mathbf{x})$  and  $A^{<[d]}$  to denote the remaining lower-degree terms. Let  $\mathbf{y}$  and  $\mathbf{z}$  be a *partition* of  $\mathbf{x}$  such that  $|\mathbf{y}| = k$ , then the *coefficient polynomial*  $A_{(\mathbf{y}, \mathbf{a})}$ , denotes the coefficient of monomial  $\mathbf{y}^{\mathbf{a}}$  in  $A(\mathbf{x})$  which is a polynomial in  $\mathbb{F}[\mathbf{z}]$ . Similarly  $A_{(\mathbf{z}, \mathbf{b})} \in \mathbb{F}[\mathbf{y}]$  is the coefficient of monomial  $\mathbf{z}^{\mathbf{b}}$  in  $A(\mathbf{x})$ . Observe that  $A_{(\mathbf{x}, \mathbf{a})}$  and  $\text{coeff}(A)(\mathbf{x}^{\mathbf{a}})$  are different. For example if  $A(\mathbf{x}) = x_1x_2 + x_2^2 + 2x_1$ , then  $A_{(x_1, 1)} = x_2 + 2$  while  $\text{coeff}(A)(x_1) = 2$ .

A polynomial  $A(\mathbf{x}) \in \mathbb{F}^{w \times w}[\mathbf{x}]$  is called a *matrix polynomial*, where the coefficients are  $w \times w$  matrices of field constants. The *coefficient space* of  $A(\mathbf{x})$  is defined as the span of

all the coefficients of  $A$ :  $\text{span}_{\mathbb{F}}\{\text{coeff}(A)(\mathbf{x}^{\mathbf{a}}) \mid \mathbf{a} \in \{0, 1, \dots, d\}^n\}$ . We can also define it for any prefix of variables.

For a set of polynomials  $\mathcal{P}$ , their  $\mathbb{F}$ -span is defined as:  $\text{span}_{\mathbb{F}}\mathcal{P} := \left\{ \sum_{A \in \mathcal{P}} \alpha_A \cdot A \mid \alpha_A \in \mathbb{F} \right\}$ . The set  $\mathcal{P}$  is called  $\mathbb{F}$ -linearly independent if  $\sum_{A \in \mathcal{P}} \alpha_A \cdot A = 0$  implies  $\alpha_A = 0$  for all  $A \in \mathcal{P}$ .  $\text{Dim}_{\mathbb{F}}\mathcal{P}$  is then defined as cardinality of the largest  $\mathbb{F}$ -linearly independent subset of  $\mathcal{P}$ .

## 6.2 Hitting set generator (HSG)

The problem of PIT asks for determining whether a given input polynomial is identically zero or not. The input polynomial is given in the form of some algebraic circuit. In *white-box* PIT, one can look ‘inside’ the input circuit while in *black-box* PIT, the input is given as a black-box and one can only evaluate the given circuit on field points. Therefore, in black-box PIT for a class of  $n$ -variate polynomials  $\mathcal{C}$ , we are asked to provide a set  $\mathcal{H} \in \mathbb{F}^n$  such that for any non-zero  $f \in \mathcal{C}$ , there exists at least one point  $\alpha \in \mathcal{H}$  such that  $f(\alpha) \neq 0$ . Such a set  $\mathcal{H}$  is called *hitting-set* for class  $\mathcal{C}$ . In general, we always have the following brute-force hitting-set, which is efficient when  $n$  is small.

**Lemma 6.2.1** ([Alo99]). *Let  $f \in \mathbb{F}[x_1, \dots, x_n]$  be a polynomial with  $\deg_{x_i}(f) \leq d_i$ , for  $i \in [n]$ . Let  $W_i \subseteq \mathbb{F}$  be a set of size at least  $d_i + 1$ . If  $f \neq 0$ , then there exists  $\mathbf{a} \in W_1 \times \dots \times W_n$  such that  $f(\mathbf{a}) \neq 0$ .*

There is also a notion of hitting set generator (HSG) or simply generator in short, which is equivalent to a hitting set and is easier to work with PIT algorithms. We frame the PIT result in this work using generators. We give the formal definition of a generator below.

**Definition 6.2.2** (Generator). *Let  $\mathcal{C}$  be a class of  $n$ -variate polynomials. Consider  $\mathcal{G} = (\mathcal{G}_1, \mathcal{G}_2, \dots, \mathcal{G}_n) : \mathbb{F}^k \rightarrow \mathbb{F}^n$ , an  $n$ -tuple of  $k$ -variate polynomials where for each  $i \in [n]$ ,  $\mathcal{G}_i \in \mathbb{F}[t_1, t_2, \dots, t_k]$ . Let  $f(x_1, \dots, x_n)$  be an  $n$ -variate polynomial. We define action of  $\mathcal{G}$  on polynomial  $f$  by  $f(\mathcal{G}) = f(\mathcal{G}_1, \dots, \mathcal{G}_n) \in \mathbb{F}[t_1, \dots, t_k]$ . We call  $\mathcal{G}$  a  $k$ -seeded generator for class  $\mathcal{C}$  if for every non-zero  $f \in \mathcal{C}$ ,  $f(\mathcal{G}) \neq 0$ . Degree of generator  $\mathcal{G}$  is defined as  $\deg(\mathcal{G}) := \max\{\deg(\mathcal{G}_i)\}_{i=1}^n$ .*



Sometimes, we will use the symbol  $f \circ \mathcal{G}$  in place of  $f(\mathcal{G})$  to avoid parenthesis in complicated equations. For a polynomial-time PIT algorithm,  $k$  is kept constant. A generator  $\mathcal{G}$  acts as a variable reduction map which converts an input polynomial  $f \in \mathbb{F}[x_1, \dots, x_n]$  to  $f(\mathcal{G}) \in \mathbb{F}[t_1, \dots, t_k]$  such that  $f \equiv 0$  if and only if  $f(\mathcal{G}) \equiv 0$ . Let  $D$  be the degree of  $\mathcal{G}$  and  $d$  be the individual degree of  $f$ . Then  $\mathcal{G}$  gives us a brute-force hitting-set of size  $(ndD)^k$  (Lemma 6.2.3). In other words, we get a polynomial-time black-box PIT algorithm for  $f$  when  $k$  is constant,  $\mathcal{G}$  can be designed in polynomial time and its degree is also polynomially bounded.

The following lemma shows how to obtain hitting set from a generator. It basically follows from Lemma 6.2.1. One can also obtain a generator from a given hitting set. See [SV15, For14] for equivalence of hitting-sets and generators.

**Lemma 6.2.3** (Generator  $\implies$  hitting-set, [SV15]). *Let  $\mathcal{G} = (\mathcal{G}_1, \dots, \mathcal{G}_n) : \mathbb{F}^k \rightarrow \mathbb{F}^n$  be a generator for a circuit class  $\mathcal{C}$  such that  $\deg(\mathcal{G}) := D$ . Let  $W \subseteq \mathbb{F}$  be any set of size  $ndD$ . Then,  $H := \mathcal{G}(W^k)$  is a hitting set, of size  $|H| \leq (ndD)^k$ , for polynomials  $f \in \mathcal{C}$  of individual degrees  $< d$ .*

Generator for a class of a polynomials is also a generator for the product of polynomials from that class, since the generator will hit each polynomial in the product.

**Lemma 6.2.4** (Folklore). *Let  $f = \prod_{i=1}^k f_i$  be a non-zero polynomial, where for each  $i \in [k]$ ,  $f_i \in \mathcal{C}$ , for some circuit class  $\mathcal{C}$ . Let  $\mathcal{G}$  be a generator for class  $\mathcal{C}$ . Then,  $f(\mathcal{G}) \neq 0$ .*

Generator for a class of polynomials is also generator for the factors of a polynomial in that class. This is because the polynomial ring is an integral domain.

**Lemma 6.2.5** (Folklore). *Let  $f \in \mathcal{C}$  be a non-zero polynomial in circuit class  $\mathcal{C}$ . Let  $\mathcal{G}$  be a generator for  $\mathcal{C}$ . If  $f$  has a non-zero factor  $g$ , then  $g(\mathcal{G}) \neq 0$ .*

Below we state the folklore trick of polynomial interpolation which recovers coefficients of a univariate polynomial from sufficiently many evaluations of the polynomial.

**Lemma 6.2.6** (Lagrange Interpolation). *Let  $\alpha_1, \dots, \alpha_k$  be any  $k$  distinct points in  $\mathbb{F}$ . Suppose we are given evaluations of a polynomial  $f(x) \in \mathbb{F}[x]$  of degree  $k - 1$  at these  $k$*

points,  $\beta_i = f(\alpha_i)$  for each  $i \in [k]$ . Then we can recover  $f$  as follows:

$$\begin{aligned}\ell_i(x) &= \prod_{\substack{1 \leq j \leq k \\ j \neq i}} \frac{x - \alpha_j}{\alpha_i - \alpha_j} \\ f(x) &= \sum_{i=1}^k \beta_i \ell_i(x)\end{aligned}$$

*Proof.* Observe that  $\ell_i(\alpha_j) = 0$  for  $i \neq j$  and 1 when  $i = j$ . Thus,  $f(\alpha_i) = \beta_i$ . Also  $f(x)$  is the unique degree  $k - 1$  polynomial with these evaluations, since if there is another polynomial  $g \neq f$  with same  $k$  evaluations, then  $f - g$  is a non-zero polynomial of degree  $\leq k - 1$  having  $k$  roots, which is a contradiction.  $\square$

Often we have a set of candidate maps for a class of polynomials  $\mathcal{P}$ , such that for each polynomial  $f \in \mathcal{P}$ , one of the maps in the set acts as a generator for that particular  $f$ . The following lemma shows that we can replace these set of candidate generators with a single generator for class  $\mathcal{P}$ .

**Lemma 6.2.7** (Generator Interpolation). *Let  $G = \{\Phi_1, \dots, \Phi_\ell\}$  where each  $\Phi_i : \mathbb{F}^k \rightarrow \mathbb{F}^n$  is of seed-length  $k$ . Suppose  $G$  is a set of candidate generators for a class of  $n$ -variate polynomials  $\mathcal{P}$  such that for any non-zero  $f \in \mathcal{P}$ , there exists  $i \in [\ell]$ ,  $f(\Phi_i) \neq 0$ . Then, there exists a single generator  $\Psi : \mathbb{F}^{k+1} \rightarrow \mathbb{F}^n$  of seed-length  $k + 1$  such that for every non-zero  $f \in \mathcal{P}$ ,  $f(\Psi) \neq 0$ . Moreover,  $\deg(\Psi) = \max\{\ell - 1, \deg(\Phi_1), \dots, \deg(\Phi_\ell)\}$ .*

*Proof.* Let  $\{\alpha_1, \dots, \alpha_\ell\}$  be an arbitrary set of distinct constants. Let  $t_1, \dots, t_k$  be the seed variables for every  $\Phi_i$  and let  $y$  be the new seed variable. Define  $\Psi : \mathbb{F}^{k+1} \rightarrow \mathbb{F}^n$  to be the Lagrange interpolation polynomial as follows:

$$\Psi = \sum_{i=1}^{\ell} \left( \prod_{\substack{1 \leq j \leq \ell \\ j \neq i}} \frac{y - \alpha_j}{\alpha_i - \alpha_j} \right) \Phi_i.$$

Observe that  $\Psi|_{y=\alpha_i} = \Phi_i$  for each  $i \in [\ell]$ . We know that for any non-zero  $f \in \mathcal{P}$ , there exists  $i \in [\ell]$  such that  $f(\Phi_i) \neq 0$ . Therefore  $f(\Psi)|_{y=\alpha_i} \neq 0$ . Hence  $f(\Psi) \neq 0$  as a polynomial in  $\mathbb{F}[y, t_1, \dots, t_k]$  since its evaluation at  $y = \alpha_i$  is non-zero. Note that  $\deg_y(\Psi) = \ell - 1$ .  $\square$

We mention the famous Kronecker map below, which preserves non-zeros of any multivariate polynomial.

**Lemma 6.2.8** ([Kro82]). *Let  $f(\mathbf{x}) \in \mathbb{F}[x_1, \dots, x_n]$  be a non-zero polynomial of individual degree  $< d$ . Let the Kronecker HSG,  $\Phi : \mathbb{F} \rightarrow \mathbb{F}^n$  be defined as  $(t^{d^0}, t^{d^1}, \dots, t^{d^{n-1}})$ . Then,  $f(\Phi) \neq 0$ .*

We now show how to replace a  $k$ -seeded HSG with a single seed HSG of appropriate degree.

**Lemma 6.2.9.** *Let  $\Phi : \mathbb{F}^k \rightarrow \mathbb{F}^n$  be a  $k$ -seeded hitting set generator for some class  $\mathcal{P}$  of  $n$ -variate, degree  $d$  polynomials in  $\mathbb{F}[\mathbf{x}]$  such that  $\deg(\Phi) = D$ . Then, there is a single-seed generator  $\Psi : \mathbb{F} \rightarrow \mathbb{F}^n$  for class  $\mathcal{P}$  with  $\deg(\Psi) \leq (dD + 1)^k$ .*

*Proof.* Let  $f \in \mathcal{P}$  be a non-zero polynomial. Since  $\Phi$  is a generator for  $f$ ,  $f(\Phi) \neq 0$ . Now, consider the Kronecker HSG  $\Gamma : \mathbb{F} \rightarrow \mathbb{F}^k = (z^{B^0}, z^{B^1}, \dots, z^{B^{k-1}})$  for the polynomial  $f(\Phi) \in \mathbb{F}[t_1, \dots, t_k]$ , where we set  $B := dD + 1$ . Since  $\deg(f(\Phi)) < B$ , by Lemma 6.2.8,  $f(\Phi \circ \Gamma) \neq 0$ . Thus, we get a univariate HSG  $\Psi := \Phi \circ \Gamma : \mathbb{F} \rightarrow \mathbb{F}^n$  such that for a non-zero  $f \in \mathcal{P}$ ,  $f(\Psi) \neq 0$ . Observe that  $\deg(\Psi) = \deg(\Gamma) \cdot \deg(\Phi) = B^{k-1} \cdot D \leq (dD + 1)^k$ .  $\square$

blackbox PIT for the class of sparse polynomials is achieved by the following lemma.

**Lemma 6.2.10** (Sparse HSG; [KS01]). *Let  $f \in \mathbb{F}[x_1, \dots, x_n]$  be a non-zero  $m$ -sparse polynomial of individual degree at most  $d$ . Let  $p$  be a prime larger than  $\max(d, mn + 1)$ . Then, there is some  $k \in [mn + 1]$  such that the univariate polynomial  $f'(y) := f(y, y^{k^1 \bmod p}, \dots, y^{k^{n-1} \bmod p})$  is non-zero. This yields a HSG  $\Psi : \mathbb{F} \rightarrow \mathbb{F}^n$  for the class of  $m$ -sparse polynomials such that  $\deg(\Psi) = \text{poly}(m, n, d)$ .*

*Proof.* We refer reader to [SY10, Theorem 4.12] for the proof of  $f'(y)$  being non-zero. Observe that this gives us a set  $G$  of HSGs for the class  $\mathcal{P}$  of  $m$ -sparse polynomials,  $G = \{\Phi_k\}_{k \in [mn+1]}$ , where  $\Phi_k : \mathbb{F} \rightarrow \mathbb{F}^n = (y, y^{k^1 \bmod p}, \dots, y^{k^{n-1} \bmod p})$ . Using Lemma 6.2.7, we get a single two-seeded HSG  $\Phi : \mathbb{F}^2 \rightarrow \mathbb{F}^n$  such that  $f(\Phi) \neq 0$  for a non-zero  $f \in \mathcal{P}$ . By using Lemma 6.2.9, we also get a univariate HSG  $\Psi$  for class  $\mathcal{P}$ . Degree of  $f(\Psi)$  is at most

$\text{poly}(m, n, d)$ . We get a deterministic poly-time blackbox PIT for class  $\mathcal{P}$  by evaluating  $f(\Psi)$  on its (degree +1)-many distinct points.  $\square$

Using the above lemma, we can also design a generator for a polynomial whose leading homogeneous part is sparse.

**Lemma 6.2.11** (Top Sparse). *Let  $f \in \mathbb{F}[x_1, \dots, x_n]$  be a polynomial of degree  $d$  such that  $f^{[d]}$  is  $m$ -sparse, where  $f^{[d]}$  denotes the top degree- $d$  homogeneous part of  $f$ . Then, there is a hitting set generator  $\Psi : \mathbb{F} \rightarrow \mathbb{F}^n$  for  $f$  with  $\deg(\Psi) = \text{poly}(m, n, d)$ .*

*Proof.* Note that if  $f$  is a non-zero polynomial of degree  $d$ , then  $f^{[d]} \neq 0$ . Moreover sparsity of  $f^{[d]}$  is at most  $m$ . Thus by Lemma 6.2.10, we have an HSG  $\Phi : \mathbb{F} \rightarrow \mathbb{F}^n$  for  $f^{[d]}$  with  $\deg(\Phi) = \text{poly}(m, n, d)$  such that  $f^{[d]}(\Phi) \neq 0$ . Let  $z$  be a new variable. Observe that,

$$f^{[d]}(\mathbf{x}) = \text{coeff}(f(zx_1, zx_2, \dots, zx_n))(z^d).$$

Let HSG  $\Phi = (g_1(y), \dots, g_n(y))$ . This gives us a two-seeded HSG  $\Phi' := (zg_1(y), \dots, zg_n(y))$  for  $f$ . This is because,

$$\begin{aligned} f(zx_1, \dots, zx_n) &= f^{[d]}(\mathbf{x}) \cdot z^d + f^{[d-1]}(\mathbf{x}) \cdot z^{d-1} + \dots + f^{[0]} \cdot z^0 \\ f(zg_1, \dots, zg_n) &= f^{[d]}(g_1, \dots, g_n) \cdot z^d + \dots + f^{[0]} \cdot z^0 \\ f(zg_1, \dots, zg_n) &= f^{[d]}(\Phi) \cdot z^d + \dots + f^{[0]} \cdot z^0 \end{aligned}$$

Since  $f^{[d]}(\Phi) \neq 0$ , this implies  $f(\Phi') = f(zg_1, \dots, zg_n) \neq 0$ . Note that  $\deg(\Phi') = \text{poly}(m, n, d)$ . Now, by Lemma 6.2.9 we get a single-seed HSG  $\Psi : \mathbb{F} \rightarrow \mathbb{F}^n$  for  $f$  such that  $f(\Psi) \neq 0$  and  $\deg(\Psi) = \text{poly}(m, n, d)$ .  $\square$

### 6.3 GCD, Resultants and Subresultants

In this section we discuss the classical tool of resultants and subresultants. In Theorem 7.2.1 we prove an interesting connection between subresultant of two polynomials with their gcd and resultant of their co-prime parts. This result will be crucial in the design of our PIT algorithm in Chapter 7.

The polynomial ring  $\mathbb{F}[x_1, \dots, x_n]$  is a unique factorization domain (UFD). Hence, the gcd of two polynomials is well defined up to a multiplication by field element. We can also define gcd with respect to a single variable  $x_i$ , where we treat rest of the variables as field elements. That is,  $\gcd_{x_i}(f, g)$  is well defined up to multiplication by a rational function depending on the remaining variables. By convention, we choose the normalized gcd whenever we consider  $\gcd_{x_i}(f, g)$  in this work. For example, let  $f = x_1^2 x_2 + x_1 x_2^2$  and  $g = x_1^2 x_2^2$ , then  $\gcd(f, g) = x_1 x_2$  while  $\gcd_{x_2}(f, g) = x_2$ . Technically, the former is gcd in  $\mathbb{F}[x_1, x_2]$  and the latter is normalized gcd in  $\mathbb{F}(x_1)[x_2]$ .

We now consider a somewhat more general scenario. Let  $A(y), B(y) \in \mathcal{R}[y]$  be two non-zero polynomials of  $y$ -degree  $d$  and  $e$ , respectively in an arbitrary UFD  $\mathcal{R}$ . Suppose  $A(y) = \sum_{i=0}^d a_i \cdot y^i$  and  $B(y) = \sum_{j=0}^e b_j \cdot y^j$ . Consider the  $(d+e) \times (d+e)$  *Sylvester matrix*  $M$  whose first  $e$  rows are the  $e$  shifts of the row vector  $(a_d, \dots, a_0, 0, \dots, 0)$  and next  $d$  rows are the  $d$  shifts of the row vector  $(b_e, \dots, b_0, 0, \dots, 0)$ .

$$M = \begin{bmatrix} a_d & a_{d-1} & \dots & a_1 & a_0 & & & \\ & a_d & a_{d-1} & \dots & a_1 & a_0 & & \\ & & \dots & \dots & \dots & \dots & & \\ & & & a_d & a_{d-1} & \dots & a_1 & a_0 \\ b_e & b_{e-1} & \dots & b_1 & b_0 & & & \\ & b_e & b_{e-1} & \dots & b_1 & b_0 & & \\ & & \dots & \dots & \dots & \dots & & \\ & & & b_e & b_{e-1} & \dots & b_1 & b_0 \end{bmatrix}.$$

**Definition 6.3.1** (Resultant). *The resultant  $\text{Res}_y(A, B) \in \mathcal{R}$  is defined to be the determinant of this Sylvester matrix. That is,  $\text{Res}_y(A, B) = \det(M)$ .*

In our setting,  $\mathcal{R}$  will be a polynomial ring, say  $\mathbb{F}[x_1, \dots, x_n]$  and  $\text{Res}_y(A, B)$  will be a polynomial free of  $y$ -variable. The following is the most fundamental property of the Resultant:

**Lemma 6.3.2** (See e.g. [GCL92, vzGG13, CLO15]). *Let  $A, B \in \mathbb{F}[y, x_1, \dots, x_n]$  be two*

polynomials. Then  $\gcd_y(A, B) \neq 1$  if and only if  $\text{Res}_y(A, B) \equiv 0$ . That is,  $A$  and  $B$  have a non-trivial factor that depends on the variable  $y$  ( $\deg_y(\gcd(A, B)) > 0$ ) iff the Resultant of  $A, B$  w.r.t.  $y$  is the identically zero polynomial.

We state the following important connection between projection of resultant and resultant of projections.

**Lemma 6.3.3.** *Let  $f, g \in \mathbb{F}[y, \mathbf{x}]$  be two polynomials and let  $\mathbf{a} \in \mathbb{F}^n$ . Then,*

$$\text{Res}_y(f, g)(\mathbf{a}) \neq 0 \implies \text{Res}_y(f(\mathbf{a}), g(\mathbf{a})) \neq 0.$$

*Proof.* Let  $d := \deg_y(f)$ ,  $e := \deg_y(g)$ ,  $r := \deg_y(f(\mathbf{a}))$  and  $t := \deg_y(g(\mathbf{a}))$ . Then with some easy determinant calculations, one can show that:

$$\text{Res}_y(f, g)(\mathbf{a}) = \begin{cases} \text{Res}_y(f(\mathbf{a}), g(\mathbf{a})) & r = d, t = e \\ (\text{lc}_y(f)(\mathbf{a}))^{e-t} \cdot \text{Res}_y(f(\mathbf{a}), g(\mathbf{a})) & r = d, t < e \\ (-1)^{e(d-r)} \cdot (\text{lc}_y(g)(\mathbf{a}))^{d-r} \cdot \text{Res}_y(f(\mathbf{a}), g(\mathbf{a})) & r < d, t = e \\ 0 & r < d, t < e \end{cases}$$

Note that if  $\text{Res}_y(f, g)(\mathbf{a}) \neq 0$ , then  $\text{Res}_y(f(\mathbf{a}), g(\mathbf{a}))$  divides it and hence the conclusion follows.  $\square$

Lemma 6.3.2 and Lemma 6.3.3 together give us the following useful Corollary:

**Corollary 6.3.4.** *Let  $f(y, \mathbf{x})$  and  $g(y, \mathbf{x})$  be two polynomials in  $\mathbb{F}[y, \mathbf{x}]$ . Let  $\mathbf{a} \in \mathbb{F}^n$ . Then,*

$$\text{Res}_y(f, g)(\mathbf{a}) \neq 0 \implies \gcd_y(f(y, \mathbf{a}), g(y, \mathbf{a})) = 1.$$

We also require multiplicative property of the Resultant that essentially follows from the definition:

**Lemma 6.3.5.** *Let  $A, B, u, v \in \mathbb{F}[y, x_1, \dots, x_n]$  be polynomials. Then  $\text{Res}_y(A, B) \mid \text{Res}_y(uA, vB)$ .*

We now study few useful sub-matrices of the Sylvester matrix below.

**Definition 6.3.6** (*j*-th principal resultant). Let  $M_j$  be the submatrix of  $M$  formed by deleting last  $j$  rows of  $A$  terms, last  $j$  rows of  $B$  terms and the last  $2j$  columns. We call  $M_j$  to be the *j*-th principal resultant of  $A$  and  $B$ . Note that  $\text{Res}_y(A, B) = M = M_0$ .

We can now define the subresultant polynomial as follows.

**Definition 6.3.7** (Subresultant). Let  $M_{ij}$  be the  $(d + e - 2j) \times (d + e - 2j)$  submatrix of Sylvester matrix  $M$  formed by deleting:

- rows  $e - j + 1$  to  $e$  (each having coefficients of  $A(y)$ ),
- rows  $d + e - j + 1$  to  $d + e$  (each having coefficients of  $B(y)$ ),
- columns  $d + e - 2j$  to  $d + e$ , except for column  $d + e - i - j$ .

Note that the *j*-th principal resultant  $M_j$  is exactly  $M_{jj}$ .

For  $0 \leq j \leq e$ , the *j*-th subresultant of  $A(y), B(y) \in \mathcal{R}[y]$  is the polynomial in  $\mathcal{R}[y]$  of degree  $j$  defined by

$$S_y(j, A, B) = \det(M_{0j}) + \det(M_{1j}) \cdot y + \dots + \det(M_{jj}) \cdot y^j.$$

We state below known results in the theory of subresultants, which will be useful for us.

**Lemma 6.3.8** (Lem 7.1 of [GCL92]). Let  $A(x), B(x) \in \mathcal{R}[x]$  be two polynomials over an arbitrary UFD  $\mathcal{R}$ . Let  $\mathcal{K}$  be the field of fractions of  $\mathcal{R}$ . Suppose

$$A(x) = Q(x) \cdot B(x) + R(x),$$

for some polynomials  $Q, R \in \mathcal{K}[x]$  such that  $\deg_x(A) = m$ ,  $\deg_x(B) = n$ ,  $\deg_x(Q) = m - n$ ,  $\deg_x(R) = k$  and  $m \geq n > k$ . Let  $b$  and  $r$  denote the leading coefficients of  $B(x)$  and  $R(x)$

respectively. Then

$$S_x(j, A, B) = (-1)^{(m-j)(n-j)} \times \begin{cases} b^{m-k} \cdot S_x(j, B, R) & 0 \leq j < k \\ b^{m-k} \cdot r^{n-k-1} \cdot R(x) & j = k \\ 0 & k < j < n-1 \\ b^{m-n+1} \cdot R(x) & j = n-1. \end{cases}$$

That is,  $S_x(j, A, B)$  equals to one of the above four expressions multiplied by the corresponding sign  $(-1)^{(m-j)(n-j)}$ .

We conclude this section making an important observation that any subresultant (and hence the Resultant) of two sparse polynomials of individual degree at most  $d$  is a sum of at most  $d+1$  determinants of  $2d \times 2d$  matrices where each entry is a coefficient of a sparse polynomial and, hence is itself a (somewhat) sparse polynomial of a small individual degree.

**Observation 6.3.9.** Let  $A, B \in \mathbb{F}[y, x_1, \dots, x_n]$  be two  $s$ -sparse polynomials with individual degrees at most  $d$ . Then for any  $j$ ,  $S_y(j, A, B)$  is an  $(2ds)^{2d+1}$ -sparse polynomial with individual degrees at most  $2d^2$ .

## 6.4 ROABPs

**Characterizing dependencies:** We state the definition of *characterizing dependencies* which defines an ROABP layer by layer.

**Definition 6.4.1** ([GKST17], Defn. 2.7). Let  $A(\mathbf{x})$  be a polynomial of individual degree  $d$  with variable-order  $(x_{\pi(1)}, x_{\pi(2)}, \dots, x_{\pi(n)})$ . Suppose, for each  $k \in [n]$  and  $\mathbf{y} = (x_{\pi(1)}, \dots, x_{\pi(k)})$ ,  $\dim_{\mathbb{F}}\{A_{(\mathbf{y}, \mathbf{a})} \mid \mathbf{a} \in \{0, 1, \dots, d\}^k\} \leq w$ . For  $k \in [n]$ , we define the spanning set  $\text{span}_k(A)$  and the dependency set  $\text{depend}_k(A)$  as subsets of  $\{0, 1, \dots, d\}^k$  as follows. For  $k = 0$ , let  $\text{depend}_0(A) := \phi$  and  $\text{span}_0(A) := \{\epsilon\}$ , where  $\epsilon = ()$  denotes the empty tuple. For  $k \in [n]$ , let

- $\text{depend}_k(A) := \{(\mathbf{a}, j) \mid \mathbf{a} \in \text{span}_{k-1}(A) \text{ and } 0 \leq j \leq d\}$ , i.e.  $\text{depend}_k(A)$  contains



all possible extensions of the tuples in  $\text{span}_{k-1}(A)$ .

- $\text{span}_k(A) \subseteq \text{depend}_k(A)$  is a subset of size  $\leq w$ , such that for any  $\mathbf{b} \in \text{depend}_k(A)$ , the polynomial  $A_{(\mathbf{y}, \mathbf{b})}$  is in the span of  $\{A_{(\mathbf{y}, \mathbf{a})} \mid \mathbf{a} \in \text{span}_k(A)\}$ .

Such dependencies of  $\{A_{(\mathbf{y}, \mathbf{a})} \mid \mathbf{a} \in \text{depend}_k(A)\}$  over  $\{A_{(\mathbf{y}, \mathbf{a})} \mid \mathbf{a} \in \text{span}_k(A)\}$  comprise the characterizing set of dependencies (certifying the width of  $A$ ).

**Nisan's characterization:** [Nis91] gave an exact width characterization for ROABPs. We follow the presentation of [GKST17] for this characterization.

**Lemma 6.4.2** ([GKST17], Lem. 2.4, 2.8). *Let  $A(\mathbf{x})$  be a polynomial of individual degree  $d$ , computed by an ROABP of width  $w$  with variable order  $(x_{\pi(1)}, x_{\pi(2)}, \dots, x_{\pi(n)})$ . For  $k \in [n]$ , let  $\mathbf{y} = (x_{\pi(1)}, x_{\pi(2)}, \dots, x_{\pi(k)})$  be the prefix of length  $k$  and  $\mathbf{z}$  be the suffix of length  $n - k$ . Then,  $\dim_{\mathbb{F}}\{A_{(\mathbf{y}, \mathbf{a})} \mid \mathbf{a} \in \{0, 1, \dots, d\}^k\} \leq w$ .*

*Conversely, let  $A(\mathbf{x})$  be a polynomial of individual degree  $d$ , with  $\mathbf{x} = \{x_1, \dots, x_n\}$  and  $w \geq 1$ , such that for any  $k \in [n]$  and  $\mathbf{y}_k = (x_{\pi(1)}, x_{\pi(2)}, \dots, x_{\pi(k)})$ , we have  $\dim_{\mathbb{F}}\{A_{(\mathbf{y}_k, \mathbf{a})} \mid \mathbf{a} \in \{0, 1, \dots, d\}^k\} \leq w$ . Then, there exists an ROABP of width  $w$  for  $A(\mathbf{x})$  in the variable order  $(x_{\pi(1)}, x_{\pi(2)}, \dots, x_{\pi(n)})$ .*

**Remark.** Here, we are taking  $w$  to be the maximum width across all layers in an ROABP. Moreover, we are talking about width of any ROABP computing  $A$ . To be more precise, Nisan's characterization says that for an *optimal* width ROABP for  $A$ , the width in layer  $k$  will be exactly the same as dimension of the coefficient polynomials in that layer. Let  $w_k$  be the width of layer  $k$  for any  $k \in [n]$  in the optimal ROABP. Then,  $w_k = \dim_{\mathbb{F}}\{A_{(\mathbf{y}_k, \mathbf{a})} \mid \mathbf{a} \in \{0, 1, \dots, d\}^k\}$ .

We need the following lemma later in Section 8.3.1, which is not difficult to prove (simply inspect the required coefficient).

**Lemma 6.4.3** ([GKST17], Lem. 2.6). *Let  $A(\mathbf{x})$  be a polynomial of individual degree  $d$ , computed by an ROABP of width  $w$ . Let  $\mathbf{y} = (x_{i_1}, x_{i_2}, \dots, x_{i_k})$  be any  $k$  variables of  $\mathbf{x}$ . Then, the coefficient polynomial  $A_{(\mathbf{y}, \mathbf{a})}$  can be computed by an ROABP of width  $w$ ,*

for every  $\mathbf{a} \in \{0, 1, \dots, d\}^k$ . Moreover, all these ROABPs have the same variable order, inherited from the variable order of the ROABP for  $A$ .

**Known results for ROABPs:** We mention few folklore and known results for ROABPs below. The width is additive for sum of ROABPs that compute in the same variable order.

**Lemma 6.4.4** (Parallel Sum). *Let  $A$  and  $B$  be polynomials computable by ROABPs of width  $w_1$  and  $w_2$  respectively, in the same variable order. Then  $A + B$  can be computed by an ROABP of width  $w_1 + w_2$  in this order.*

*Proof.* We compute the sum by joining the two ROABPs in parallel. Let  $s_1, t_1$  be the source and sink vertices of ROABP  $A$  respectively and  $s_2, t_2$  be that of  $B$ . Create a new source vertex  $s$  and a sink vertex  $t$  for  $A + B$ . Draw an edge from  $s \rightarrow s_1$  with unit label and an edge  $s \rightarrow s_2$  with unit label. Similarly, join  $t_1 \rightarrow t$  and  $t_2 \rightarrow t$  with unit labels. Clearly, the new ROABP is of width  $w_1 + w_2$ .  $\square$

The lemma below shows that PIT for  $\Sigma \wedge \Sigma$  circuits reduces to PIT for log-variate ROABPs.

**Lemma 6.4.5** ( $\Sigma \wedge \Sigma$  to ROABP; [FSS14], [Sax08]). *If we have poly-time blackbox PIT for log-variate (commutative) ROABPs, then we have poly-time blackbox PIT for (standard multivariate) diagonal depth-3 circuits. Moreover, if we have poly-time blackbox PIT for sum of log-variate, constant width (commutative) ROABPs, then also we get the same conclusion.*

*Proof sketch.* [FS13a, FS13b] exploited the fact that diagonal depth-3 circuits have low dimension partial-derivative space to show that non-zero polynomials computed by them have a nonzero *log-support* monomial. That is, a degree  $d$  polynomial  $f = \sum_{i=1}^k \ell_i^{d_i}$  has  $\dim_{\mathbb{F}}\{\partial^{<\infty}(f)\} = \text{poly}(k, n, d) = \text{poly}(s)$ , where  $s$  is the size of circuit,  $\ell_i$ 's are linear polynomials and  $\{\partial^{<\infty}(f)\}$  denotes the space of all partial derivatives of  $f$ . Note that a simple monomial like  $x_1 x_2 \dots x_n$  with support  $n$  has  $\dim_{\mathbb{F}}\{\partial^{<\infty}(x_1 \dots x_n)\} = 2^n$ . With

few additional observations, they prove that a non-zero  $f$  must compute a monomial with non-zero coefficient, that is supported on at most  $O(\log s)$  variables.

Under the promise of such a log-support monomial, we can apply the generator  $\mathcal{G}$  of [SV09], used in [FSS14] or the map of [Vai15], both of seed-length  $O(\log n)$ . Both these maps preserve non-zerosness of  $f$ , i.e  $f(\mathcal{G}) \not\equiv 0$ .

The maps/generators are such defined that after applying either of them, we will get to ‘power of sums of univariates’ form which we can convert to ‘sum of products of univariates’ form using the duality-trick of [Sax08], i.e  $f(\mathcal{G}) = \sum_{i=1}^t \prod_{j=1}^{n'} f_{ij}(x_j)$  where  $t = \text{poly}(s, d)$  and number of variables in  $f(\mathcal{G})$  are  $n' = O(\log s)$ . Observe that each product-of-univariates  $\prod_{j=1}^{n'} f_{ij}(x_j)$ , has a width-1 ROABP in any variable order (commutative). Thus by Lemma 6.4.4,  $f(\mathcal{G})$  can be computed by an  $O(\log s)$ -variate, width  $t = \text{poly}(s, d)$  ROABP. Thus, solving PIT for log-variate ROABP will solve PIT for  $f(\mathcal{G})$  and hence for  $f$ . Second part follows by observing that  $f(\mathcal{G})$  is also a sum of  $t$ -many width-1, log-variate ROABPs.  $\square$

**Prefix and suffix maps:** In Chapter 8, we will need the notation of prefix and suffix maps which are defined below.

**Definition 6.4.6** (Prefix/suffix map). *Suppose  $A$  is an ROABP in the variable order  $(y_1, \dots, y_n)$ . For some  $\ell \leq n$ , let  $\mathcal{G} : \mathbb{F} \rightarrow \mathbb{F}^\ell = (\mathcal{G}_1, \dots, \mathcal{G}_\ell)$  be a generator for  $\ell$ -variate ROABPs of some fixed width, with a single seed variable  $t$ . We call  $\Psi : \mathbb{F}^{n-\ell+1} \rightarrow \mathbb{F}^n = (\mathcal{G}_1, \dots, \mathcal{G}_\ell, y_{\ell+1}, \dots, y_n)$  a prefix map with respect to generator  $\mathcal{G}$ . Note that  $A(\Psi) \in \mathbb{F}[t, y_{\ell+1}, \dots, y_n]$ . In a similar fashion, we define suffix map with respect to  $\mathcal{G}$  as  $\Phi : \mathbb{F}^{n-\ell+1} \rightarrow \mathbb{F}^n = (y_1, \dots, y_{n-\ell}, \mathcal{G}_1, \dots, \mathcal{G}_\ell)$ .*

Basically, a prefix map projects the initial  $\ell$  variables of a polynomial using  $\mathcal{G}$  and leaves the remaining variables as it is, while a suffix map projects the last  $\ell$  variables using  $\mathcal{G}$  and leaves the initial variables untouched. Below, we show that any linear combination of polynomials computed in an internal layer of an ROABP can also be computed by an ROABP of same width.

**Lemma 6.4.7** (Partial ROABP). *Let  $D = \prod_{i=1}^n D_i(y_i)$  be an ROABP of width  $w$  in the variable order  $(y_1, \dots, y_n)$ . For some  $k \in [n-1]$ , consider the decomposition of  $D$  at  $k$ -th layer as  $D = \sum_{i=1}^w P_i \cdot Q_i$ . Then for any  $c_1, \dots, c_w \in \mathbb{F}$ , the polynomials  $\sum_{i=1}^w c_i \cdot P_i$  and  $\sum_{i=1}^w c_i \cdot Q_i$  can be computed by ROABPs of width  $w$  each.*

*Proof.* Note that  $D = D_{\leq k} \cdot D_{> k} = (P_1, \dots, P_w)(Q_1, \dots, Q_w)^\top$ . Observe that  $D_{\leq k}$  is an ROABP of width  $w$  with  $w$ -many output nodes computing the polynomials  $P_1, \dots, P_w$ . We can add a single sink node  $t$  and edges  $(P_i, t)$  with weights  $c_i$ , for each  $i \in [w]$ . This modified ROABP of width  $w$  computes the polynomial  $\sum_{i=1}^w c_i \cdot P_i$ . Proof for  $\sum_{i=1}^w c_i \cdot Q_i$  follows similarly.  $\square$

Below, we show that a generator for an ROABP in  $\ell$  variables, can be extended to a generator for an ROABP in  $n \geq \ell$  variables of same width, if the remaining variables are untouched.

**Lemma 6.4.8.** *Let  $\mathcal{G} : \mathbb{F} \rightarrow \mathbb{F}^\ell$  be a generator for  $\ell$ -variate ROABPs of width  $w$ . Then the prefix map  $\Psi$  with respect to  $\mathcal{G}$  is a generator for  $n$ -variate ROABPs of width  $w$ , for some  $n \geq \ell$ .*

*Proof.* Suppose  $\mathcal{G} = (\mathcal{G}_1, \dots, \mathcal{G}_\ell)$ . Let  $f \in \mathbb{F}[x_1, \dots, x_n]$  be a non-zero polynomial computed by an ROABP of width  $w$  in the variable order  $(y_1, \dots, y_n)$ . For the sake of contradiction, suppose  $f(\Psi) =: f(\mathcal{G}_1, \dots, \mathcal{G}_\ell, y_{\ell+1}, \dots, y_n) \equiv 0$ . Then decomposing  $D$  at  $\ell$ -th layer we get  $f = \sum_{i=1}^w P_i \cdot Q_i$ , where for each  $i \in [w]$ ,  $P_i \in \mathbb{F}[\mathbf{y}_{\leq \ell}]$  and  $Q_i \in \mathbb{F}[\mathbf{y}_{> \ell}]$ . By Nisan's characterization (Lemma 6.4.2),  $Q_1, \dots, Q_w$  are  $\mathbb{F}$ -linearly independent. Observe that

$$f(\Psi) = \sum_{i=1}^w P_i(\mathcal{G}) \cdot Q_i \equiv 0. \quad (6.1)$$

Let  $t$  be the single seed variable for  $\mathcal{G}$ . For each  $i$ ,  $P_i(\mathcal{G})$  is a non-zero univariate polynomial in  $t$ , as  $P_i$  is a non-zero  $\ell$ -variate polynomial computed by a width  $w$  ROABP and  $\mathcal{G}$  is a generator for such polynomials. Therefore, there exists some  $i \in [w]$  and  $\alpha \in \mathbb{F}$  such that  $P_i(\mathcal{G})|_{t=\alpha} \neq 0$ . Let  $c_i := P_i(\mathcal{G})|_{t=\alpha}$  for all  $i \in [w]$ . Then  $\sum_{i=1}^w c_i \cdot Q_i \equiv 0$  by (6.1). Since there is some non-zero  $c_i$ , this implies a non-trivial dependency among  $Q_i$ 's, which contradicts their linear independence. Hence,  $f(\Psi) \neq 0$ .  $\square$

Similarly, one can show that the suffix map  $\Phi$  wrt generator  $\mathcal{G}$  is also a hitting set for  $n \geq \ell$  variate-ROABPs.

## 6.5 Border PIT

Recall the definition of a border class from Section 1.4.2. It turns out that for a single ROABP, border does not add any power, i.e.  $\overline{\text{ROABP}} = \text{ROABP}$ . We give a proof exposition below.

**Lemma 6.5.1** ([For16]). *A polynomial  $f \in \mathbb{F}[\mathbf{x}]$  in the border class of width  $w$  ROABPs can also be computed by an ROABP of width at most  $w$ .*

*Proof.* Let  $g = f + \epsilon h$ , where  $g$  has an ROABP of width  $w$  over  $\mathbb{F}(\epsilon)$ . We need to show that the limit polynomial  $f$  also has ROABP-width  $\leq w$  over  $\mathbb{F}$ . Let the unknown variable order of  $g$  be  $(y_1, \dots, y_n)$ . By applying Nisan's characterization [Nis91] on  $g$ , we know that for all  $k \in [n]$ , the matrix defined in [Nis91] for each layer,  $M_k$  has rank at most  $w$  over  $\mathbb{F}(\epsilon)$ . This means determinant of any  $(w+1) \times (w+1)$  minor of  $M_k$  is identically zero. Observe that entries of  $M_k$  are coefficients of monomials of  $g$  which are in  $\mathbb{F}[\epsilon][\mathbf{x}]$ . Thus, determinant polynomial will remain zero even under the limit,  $\epsilon \rightarrow 0$ . Hence, for  $f = \lim_{\epsilon \rightarrow 0} g$ , each matrix  $M_k$  also has rank at most  $w$  over  $\mathbb{F}$ . Thus by Nisan's characterization,  $f$  also has an ROABP of width at most  $w$ . This matrix is now commonly called as partial derivative matrix. The notion of rank of partial derivative matrix is equivalent to the notion of dimension of the space spanned by the coefficient polynomials as defined in Lemma 6.4.2 and used in this work. We refer the reader to chapter on The Partial Derivative Matrix in [Sap16] for details on this matrix and its connection with coefficient polynomials.  $\square$

Let  $f$  be a degree  $d$  polynomial computed by an ROABP of width  $w$  and let  $f^{[d]}$  be its leading homogeneous degree- $d$  part. Lemma 8.2.6 states that  $f^{[d]}$  can also be computed an ROABP of width  $w$ . This fact also has a nice alternate proof via border complexity as follows. It is not difficult to show that for a polynomial  $f$  of degree- $d$  in class  $\mathcal{C}$ ,  $f^{[d]}$  can be computed in the border class  $\overline{\mathcal{C}}$ . Since for ROABPs the border is the same,  $f^{[d]}$  can also be computed by an ROABP of width  $\leq w$ .

For a class  $\mathcal{C}$ , where the border class  $\overline{\mathcal{C}}$  is same as  $\mathcal{C}$ , the PIT algorithm will be same for both  $\mathcal{C}$  and  $\overline{\mathcal{C}}$ . For example, the class of sparse polynomials and also the class of single ROABPs (Lemma 6.5.1). However, for classes where  $\overline{\mathcal{C}} \neq \mathcal{C}$ , blackbox PIT algorithms that work for  $\mathcal{C}$  may not work for the border class  $\overline{\mathcal{C}}$ . The only thing we can say for such classes, in general, is that PIT for  $\overline{\mathcal{C}}$  is in PSPACE, as PIT for  $\overline{\text{VP}}$  is in PSPACE [GSS19, FS18]. PIT algorithms, which rely on a rank based measure, usually work for the border class also, since the rank based measure also works for the border class. An example of this are the PIT algorithms for the class  $\mathcal{C}$  of diagonal depth-3 circuits, even though for this class it is unknown whether  $\overline{\mathcal{C}}$  is same as  $\mathcal{C}$ .

**Lemma 6.5.2** (Border  $\Sigma \wedge \Sigma$ ; [For16]). *The blackbox PIT algorithms in [FS13a], [FSS14] and [FGS18] for the class of diagonal depth-3 circuits also solve blackbox PIT for its border class in their same respective times.*

*Proof Sketch.* We discussed in the first part of the proof of Lemma 6.4.5 that for a polynomial  $f$  computed by size- $s$   $\Sigma \wedge \Sigma$  circuit,  $\dim\{\partial^{<\infty}(f)\} = \text{poly}(s)$  which helps in proving that  $f$  has a non-zero monomial of  $O(\log s)$  support. All of the works - [FS13a], [FSS14] and [FGS18] build on this property to give efficient PIT algorithms for  $f$ .

Actually, one can also show that a polynomial  $g$  computed in the border of size- $s$   $\Sigma \wedge \Sigma$  circuit also has  $\dim\{\partial^{<\infty}(g)\} = \text{poly}(s)$  as discussed in [For16]. Its proof is very similar to the proof of Lemma 6.5.1 below. This then proves that  $g$  also has a non-zero monomial of  $O(\log s)$  support. Thus, the above mentioned PIT algorithms also work for  $g$ . □

## Chapter 7

# PIT for $\Sigma^{[2]}\Pi\Sigma\Pi^{[\text{ind-deg } d]}$ circuits

In this chapter, we focus on the circuit model for PIT. The depth-4  $\Sigma\Pi\Sigma\Pi$  circuit class (see Section 1.1 for definition) is extremely important in the context of the PIT problem, as it is known that a polynomial-time black-box PIT for this class implies a quasi-polynomial-time black-box PIT for general  $\text{VP}$  circuits [AV08, AGS19]. It is no surprise therefore, that this model has been tough to crack. Even very restricted subclasses of  $\Sigma\Pi\Sigma\Pi$  are open, for example poly-time PIT for  $\Sigma^{[k]}\Pi\Sigma\Pi^{[\delta]}$ , even when  $k = \delta = 3$ . Our main result here is an efficient (deterministic) identity testing algorithm for the class of  $\Sigma^{[2]}\Pi\Sigma\Pi^{[\text{ind-deg } d]}$  circuits, where a  $\Sigma^{[2]}\Pi\Sigma\Pi^{[\text{ind-deg } d]}$  circuit  $C$  of size  $s$  computes a polynomial of the form:

$$C = \prod_{i=1}^r g_i + \prod_{j=1}^m h_j$$

where each polynomial ( $g_i$  and  $h_j$ ) is an  $s$ -sparse polynomial with individual degree at most some constant  $d$ . Note, though, that  $r$  and  $m$ , and hence the total degree of  $C$ , can be arbitrary (i.e. polynomially) large. In particular, the polynomial computed by  $C$  may not itself be sparse. This class generalizes the model considered in [Vol17], where  $m = 1$  and the  $g_i$ -s are irreducible polynomials.

It turns out this problem is related to the sparse factorization problem discussed in Chapter 3. Observe that the identity testing problem for this circuit class reduces to polynomial factorization of sparse polynomials with bounded individual degree. Therefore, by invoking the factorization algorithm of [BSV20], we can get an algorithm whose runtime

is efficient in terms of the sparsity bound. Plugging in the best bound of [BSV20], results in a quasi-polynomial-time algorithm. Here, we give a *polynomial-time* algorithm for this model. In addition, our algorithm operates in the black-box setting, whereas the described factorization-based algorithm is a white-box algorithm. Let  $f \in \mathbb{F}[x_1, \dots, x_n]$  be a polynomial computed by a  $\Sigma^{[2]}\Pi\Sigma\Pi^{[\text{ind-deg } d]}$  of size  $s$ , where  $d$  is some fixed constant. Then, we show that *there exists a deterministic algorithm that given a black-box access to  $f$  determines if  $f \equiv 0$ , in time  $\text{poly}(s, n)$* . See Theorem 7.3.9 for the formal version.

## 7.1 Proof Technique

Let  $C = \prod_{i=1}^r g_i + \prod_{j=1}^m h_j$  where  $g_i$ -s and  $h_j$ -s are  $s$ -sparse polynomials in  $\mathbb{F}[x_1, x_2, \dots, x_n]$  of individual degree at most  $d$ . Clearly, if  $C \equiv 0$  then it will evaluate to zero on any input. Now suppose  $C \not\equiv 0$ . Our goal is to find a point  $\mathbf{a} \in \mathbb{F}^n$  such that  $C(\mathbf{a}) \neq 0$ . Our approach relies on the uniqueness of factorization property of the ring of multivariate polynomials. Specifically, we have that

$$\prod_{i=1}^r g_i \neq - \prod_{j=1}^m h_j$$

Consequently, wlog there exists an irreducible polynomial (factor)  $u$  and  $\ell > 0$  such that  $u^\ell$  divides the LHS but does not divide the RHS. Our goal is to preserve this “situation” while reducing the number of variables. Clearly, a random projection will be sufficient. However, we wish to obtain a deterministic algorithm. To this end, we are looking for a projection that does not introduce new dependencies between factors. That is, for every  $i, j$ : if  $v \mid g_i$  and  $u \mid h_j$  satisfying  $\gcd(u, v) = 1$  we need to ensure that  $\gcd(u', v') = 1$ , when  $u'$  and  $v'$  are the projections of  $u$  and  $v$ , respectively. The main tool for that is the *Resultant*. Indeed, one of the fundamental properties of the resultant is that

$$\text{Res}(A, B) \neq 0 \text{ if and only if } \gcd(A, B) = 1.$$

In the multivariate setting, this condition roughly translates into:

$$[\forall x_k : \text{Res}_{x_k}(u, v) \neq 0] \implies \gcd(u', v') = 1.$$



In other words, we need to hit all the resultants of the form  $\text{Res}_{x_k}(u, v)$  when  $v \mid g_i$  and  $u \mid h_j$ . By definition,  $\text{Res}_{x_k}(u, v)$  is a determinant of  $2d \times 2d$  matrix where each entry is a coefficient of  $u$  or  $v$ . Hence,  $\text{Res}_{x_k}(u, v)$  is  $t^{O(d)}$ -sparse polynomial with individual degree at most  $O(d^2)$ , where  $t$  is an upper bound on the sparsities of  $u$  and  $v$ . Consequently, we can use a hitting set generator for sparse polynomials (e.g. [KS01]) to hit the resultant. As  $u$  and  $v$  are factors of  $s$ -sparse polynomials of individual degree  $d$ , the best upper bound by [BSV20] will be  $t = s^{O(d^2 \log s)}$ . This will result in a quasi-polynomial-time algorithm.

Another idea would be to use the multiplicative properties of the resultant and hit  $\text{Res}_{x_k}(h_j, g_i)$  instead. Indeed,  $\text{Res}_{x_k}(h_j, g_i) \neq 0 \implies \text{Res}_{x_k}(u, v) \neq 0$  and since  $g_i$  and  $h_j$  are  $s$ -sparse,  $\text{Res}_{x_k}(h_j, g_i)$  is  $s^{O(d)}$ -sparse and this would get a polynomial-time algorithm. The main issue is that we could have  $\text{Res}_{x_k}(u, v) \neq 0$  while  $\text{Res}_{x_k}(h_j, g_i) \equiv 0$ . For example, if  $h_j = uf$  and  $g_i = vf$  for the same polynomial  $f$ . Going back, one may ask whether we could show a better sparsity bound on  $\text{Res}_{x_k}(u, v)$ . While we do not quite do that, we instead show that  $\text{Res}_{x_k}(u, v)$  is a *factor* of *some*  $s^{O(d)}$ -sparse polynomial of individual degree at most  $O(d^2)$ . As the ring of polynomials forms an integral domain, this allows us to use a polynomial-size hitting set generator for sparse polynomials.

To achieve the above goal, suppose for simplicity that  $g_i = u^{a_1} \cdot v^{b_1}$  and  $h_j = u^{a_2} \cdot v^{b_2}$ , for some non-negative integers  $a_1, b_1, a_2, b_2$ . If all these numbers are strictly positive, we run into the same issue we have encountered earlier. That is,  $\text{Res}_{x_k}(u, v) \neq 0$  while  $\text{Res}_{x_k}(h_j, g_i) \equiv 0$ . To address that, we apply Theorem 7.2.1 (our structural result) which allows us to “extract” the gcd. For example, if  $g_i = uv^2$  and  $h_j = u^2v$ , we can write  $g_i = v \cdot uv$  and  $h_j = u \cdot uv$  and obtain that  $\text{Res}_{x_k}(u, v)$  is a factor of  $S_{x_k}(\deg_{x_k}(uv), g_i, h_j)$ , which is an  $s^{O(d)}$ -sparse polynomial (see Observation 6.3.9). However, a sole gcd extraction may be insufficient. Consider the case when  $g_i = uv^2$  and  $h_j = uv$ . Repeating the same argument will just yield a trivial statement that  $\text{Res}_{x_k}(v, 1) = 1$  is a factor of a sparse polynomial. To overcome this difficulty, we apply the previous argument on powers of  $g_i$  and  $h_j$ . That is, on  $g_i^z = u^{za_1} \cdot v^{zb_1}$  and  $h_j^t = u^{ta_2} \cdot v^{tb_2}$ . The idea now would be to isolate the powers of  $u$  from the powers of  $v$ . Within the same example, consider  $g_i^2 = u^2v^4 = v \cdot u^2v^3$  and  $h_j^3 = u^3v^3 = u \cdot u^2v^3$ . Now, by Theorem 7.2.1,  $\text{Res}_{x_k}(u, v)$

is a factor of  $S_{x_k}(\deg_{x_k}(u^2v^3), g_i^2, h_j^3)$ . More generally, we show how to find appropriate “small”  $z$  and  $t$  using linear algebra.

Unfortunately, though, this could be made possible only when  $h_j$  and  $g_i$  satisfy certain “non-degeneracy” condition w.r.t  $u$  and  $v$ . More formally, when the matrix  $E := \begin{bmatrix} a_1 & a_2 \\ b_1 & b_2 \end{bmatrix}$  has full rank (see Lemma 7.3.6). Our final crucial observation is that we can actually ignore “degenerate” pairs  $u, v$ . To this end, we prove a technical lemma (Lemma 7.3.2) which could be of independent interest.

## 7.2 Structure Theorem

In this section, we prove our main technical result which links the gcd of two polynomials, their subresultant and the resultant of their coprime parts. It could be of interest in its own right.

**Theorem 7.2.1.** *Let  $A(x), B(x) \in \mathcal{R}[x]$  be two polynomials over an arbitrary UFD  $\mathcal{R}$ . Suppose  $A(x) = f(x) \cdot g(x)$  and  $B(x) = h(x) \cdot g(x)$  with  $\deg_x(A) = m$ ,  $\deg_x(B) = n$ ,  $\deg_x(g) = d$ ,  $\deg_x(f) = m' = m - d$  and  $\deg_x(h) = n' = n - d$ . Then*

$$S_x(d, A, B) = g \cdot \text{Res}_x(f, h) \cdot \text{lc}_x(g)^{m'+n'-1}$$

*Proof.* Let  $\mathcal{K}$  be the field of fractions of UFD  $\mathcal{R}$ . Consider Euclidean division of  $A$  by  $B$  in  $\mathcal{K}[x]$  so that we get  $A(x) = Q(x) \cdot B(x) + R(x)$ , for some polynomials  $Q, R \in \mathcal{K}[x]$  such that  $\deg_x(R) < \deg_x(B)$ . Note that since  $g$  divides both  $A$  and  $B$ , it must also divide  $R$ . Therefore,  $R = g \cdot p$  for some polynomial  $p(x) \in \mathcal{K}[x]$ . Thus, we also get

$$f(x) = Q(x) \cdot h(x) + p(x) \tag{7.1}$$

Let  $\deg_x(R) = k$  for some  $k < n$  and let  $\deg_x(p) = k' = k - d$ . Now, we prove the theorem by induction on  $\deg_x(p)$ .

**Base case:**  $\deg_x(p) = k' = 0$ . In other words,  $\deg_x(R) = k = d$ . Thus using second case of Lemma 6.3.8, we get that:

$$\begin{aligned}
S_x(d, A, B) &= (-1)^{(m-d)(n-d)} \cdot b^{m-k} \cdot r^{n-k-1} \cdot R \\
&= (-1)^{m'.n'} \cdot \text{lc}_x(h)^{m-k} \cdot \text{lc}_x(g)^{m-k} \cdot \text{lc}_x(p)^{n-k-1} \cdot \text{lc}_x(g)^{n-k-1} \cdot pg \\
&= (-1)^{m'.n'} \cdot g \cdot \text{lc}_x(h)^{m-k} \cdot \text{lc}_x(p)^{n-k} \cdot \text{lc}_x(g)^{m+n-2k-1} \\
S_x(d, A, B) &= (-1)^{m'.n'} \cdot g \cdot \text{lc}_x(h)^{m-k} \cdot \text{lc}_x(p)^{n-k} \cdot \text{lc}_x(g)^{m'+n'-1} \tag{7.2}
\end{aligned}$$

The second last step above follows because  $p = \text{lc}_x(p)$  when  $\deg_x(p) = 0$ . Now, we shall compute  $\text{Res}_x(f, h)$ . Note that  $\text{Res}_x(f, h) = S_x(0, f, h)$  by definition of subresultant. Considering (7.1) with  $\deg_x(p) = 0$ , we can use second case of Lemma 6.3.8 to get:

$$\begin{aligned}
S_x(0, f, h) &= (-1)^{(\deg_x(f)-0) \cdot (\deg_x(h)-0)} \cdot \text{lc}_x(h)^{\deg_x(f)-\deg_x(p)} \cdot \text{lc}_x(p)^{\deg_x(h)-\deg_x(p)-1} \cdot p \\
&= (-1)^{m'.n'} \cdot \text{lc}_x(h)^{m'} \cdot \text{lc}_x(p)^{n'-1} \cdot p \quad [\text{as } \deg_x(p) = 0] \\
&= (-1)^{m'.n'} \cdot \text{lc}_x(h)^{m'} \cdot \text{lc}_x(p)^{n'} \quad [\text{as } p = \text{lc}_x(p)] \\
\text{Res}_x(f, h) &= (-1)^{m'.n'} \cdot \text{lc}_x(h)^{m-k} \cdot \text{lc}_x(p)^{n-k} \tag{7.3}
\end{aligned}$$

(7.2) and (7.3) together yield  $S_x(d, A, B) = g \cdot \text{Res}_x(f, h) \cdot \text{lc}_x(g)^{m'+n'-1}$  for the base case.

**Induction step:** Now, we assume  $\deg_x(p) = k' > 1$ . In other words,  $\deg_x(R) = k > d$ . Therefore, by first case of Lemma 6.3.8:

$$\begin{aligned}
S_x(d, A, B) &= (-1)^{(m-d)(n-d)} \cdot b^{m-k} \cdot S_x(d, B, R) \\
&= (-1)^{m'.n'} \cdot \text{lc}_x(h)^{m-k} \cdot \text{lc}_x(g)^{m-k} \cdot S_x(d, B, R) \tag{7.4}
\end{aligned}$$

Now consider Euclidean division of  $B$  by  $R$  in  $\mathcal{K}[x]$  to get

$$B(x) = Q'(x) \cdot R(x) + R'(x) \tag{7.5}$$

for some polynomial  $R'(x) \in \mathcal{K}[x]$  with  $\deg_x(R') < \deg_x(R)$ . Since  $g$  divides both  $B$  and  $R$ , we deduce that  $g$  must also divide  $R'$ . Let  $R' = g \cdot p'$  for some polynomial  $p' \in \mathcal{K}[x]$ .

Thus from (7.5), we also get

$$h(x) = Q'(x) \cdot p(x) + p'(x) \quad (7.6)$$

In (7.5) since  $\deg_x(R') < \deg_x(R)$  or equivalently  $\deg_x(p') < \deg_x(p)$ , we can use induction hypothesis to deduce that,

$$S_x(d, B, R) = g \cdot \text{Res}_x(h, p) \cdot \text{lc}_x(g)^{n'+k'-1} \quad (7.7)$$

Note that  $\deg_x(p) = k' > 0$  in induction step, thus we can use first case of Lemma 6.3.8 on (7.1) to get

$$\begin{aligned} \text{Res}_x(f, h) &= S_x(0, f, h) \\ &= (-1)^{(\deg_x(f)-0)(\deg_x(h)-0)} \cdot \text{lc}_x(h)^{\deg_x(f)-\deg_x(p)} \cdot S_x(0, h, p) \\ &= (-1)^{m' \cdot n'} \cdot \text{lc}_x(h)^{m'-k'} \cdot \text{Res}_x(h, p) \\ \text{Res}_x(h, p) &= \frac{\text{Res}_x(f, h)}{(-1)^{m' \cdot n'} \cdot \text{lc}_x(h)^{m'-k'}}. \end{aligned} \quad (7.8)$$

Substituting (7.8) in (7.7), we get:

$$S_x(d, B, R) = g \cdot \frac{\text{Res}_x(f, h)}{(-1)^{m' \cdot n'} \cdot \text{lc}_x(h)^{m'-k'}} \cdot \text{lc}_x(g)^{n'+k'-1} \quad (7.9)$$

Substituting (7.9) back into (7.4), we get

$$\begin{aligned} S_x(d, A, B) &= (-1)^{m' \cdot n'} \cdot \text{lc}_x(h)^{m-k} \cdot \text{lc}_x(g)^{m-k} \cdot g \cdot \frac{\text{Res}_x(f, h)}{(-1)^{m' \cdot n'} \cdot \text{lc}_x(h)^{m'-k'}} \cdot \text{lc}_x(g)^{n'+k'-1} \\ &= \text{lc}_x(g)^{m-k} \cdot g \cdot \text{Res}_x(f, h) \cdot \text{lc}_x(g)^{n'+k'-1} \quad [\text{as } m-k = m'-k'] \\ &= g \cdot \text{Res}_x(f, h) \cdot \text{lc}_x(g)^{m-k+n'+k'-1} \\ &= g \cdot \text{Res}_x(f, h) \cdot \text{lc}_x(g)^{m'+n'-1} \quad [\text{as } m-k+k' = m-d = m'] \end{aligned}$$

This completes the proof of induction step, as well as that of the theorem.  $\square$

### 7.3 PIT for $\Sigma^{[2]}\Pi\Sigma\Pi^{[\text{ind-deg } d]}$ Circuits

In this section we prove our main result in Theorem 7.3.9. We refer the reader to Section 1.1 for the formal definition of an algebraic circuit and PIT algorithm. Also, recall the definition of hitting set generator in Definition 6.2.2.

#### 7.3.1 The $\Sigma^{[k]}\Pi\Sigma\Pi^{[\text{ind-deg } d]}$ Model

A size  $s$ , depth-4  $\Sigma\Pi\Sigma\Pi$  circuit computes a polynomial of the form  $f = \sum_{i=1}^k \prod_{j=1}^m f_{ij}$ , where  $f_{ij}$  are  $s$ -sparse polynomials for each  $i \in [k], j \in [m]$ . For  $s = \text{poly}(n)$ , [LST22] gives the first deterministic sub-exponential time PIT for constant-depth (depth-4 also) which runs in  $(sn)^{O(n^\mu)}$ -time, where  $\mu > 0$  is any real number. While a polynomial-time PIT algorithm for general depth-4 circuit continues to be elusive, various restricted versions of this model have been attacked. One such restriction is to make the top fan-in  $k$  constant. For  $k = 2$ , even white-box PIT for  $\Sigma^{[2]}\Pi\Sigma\Pi$  circuits is still open. A more restricted model is the class of  $\Sigma^{[k]}\Pi\Sigma\Pi^{[d]}$  circuits, where the top fan-in  $k$  and the bottom fan-in  $d$  are constants. For a size- $s$  circuit of this class,  $f_{ij}$ 's are  $s$ -sparse polynomials of constant total degree at most  $d$ . Even this restricted model seems to be quite non-trivial. Only very recently, [DDS21] gave a quasi-polynomial-time black-box PIT algorithm for this model. For  $k = 3$  and  $d = 2$  ( $f_{ij}$ 's are quadratic polynomials), [PS21] give a polynomial-time black-box PIT algorithm. For  $k = 3$  and  $d > 2$ , coming up with a polynomial PIT algorithm remains an open question.

We now introduce, what we call the  $\Sigma^{[k]}\Pi\Sigma\Pi^{[\text{ind-deg } d]}$  model. In the  $\Sigma^{[k]}\Pi\Sigma\Pi^{[d]}$  model, the sparse polynomials  $f_{ij}$ 's have constant total degree  $\leq d$ . We relax this restriction to  $f_{ij}$ 's being constant *individual* degree  $\leq d$  polynomials in  $\Sigma^{[k]}\Pi\Sigma\Pi^{[\text{ind-deg } d]}$  model. This is a more general model, since  $f_{ij}$ 's can now have much higher total degree, like  $O(n)$ . In Section 7.3.3, we give a deterministic polynomial-time black-box PIT algorithm for this model when  $k = 2$  and  $d$  is any constant. We also note that our PIT algorithm works for any field  $\mathbb{F}$ , while the works of [PS21, DDS21] do have certain field restrictions.

### 7.3.2 Vectors and Interlacing

In this section we will state and prove some useful properties of vectors that will be useful later. It may also be of independent interest.

**Definition 7.3.1** (Interlacing). *Let  $a, b, c, d \in \mathbb{R}$ . We say that two pair of points  $(a, b)$  and  $(c, d)$  interlace if  $(a - c)(b - d) < 0$ . Equivalently,  $(a - c)$  and  $(b - d)$  have opposite signs.*

**Remark:** In particular, interlacing implies that at least two of  $a, b, c, d$  are non-zero.

The following result relates the interlacing property with linear dependence.

**Lemma 7.3.2.** *Let  $\mathbf{a}, \mathbf{b} \in \mathbb{R}_{\geq 0}^n$  and  $\mathbf{c}, \mathbf{d} \in \mathbb{R}_{\geq 0}^m$  such that  $\left(\sum_{i=1}^n a_i, \sum_{i=1}^n b_i\right)$  interlaces with  $\left(\sum_{j=1}^m c_j, \sum_{j=1}^m d_j\right)$ . Then there exists  $i \in [n], j \in [m]$  such that the matrix  $\begin{bmatrix} a_i & c_j \\ b_i & d_j \end{bmatrix}$  has full rank.*

*Proof.* Without loss of generality, suppose  $\sum_{i=1}^n a_i > \sum_{j=1}^m c_j$  and  $\sum_{i=1}^n b_i < \sum_{j=1}^m d_j$ . In particular, there exists  $s \in [n], t \in [m]$  such that  $a_s, d_t > 0$ . Consider the  $2 \times (n + m)$  matrix,

$$E := \begin{bmatrix} a_1 & \cdots & a_n & c_1 & \cdots & c_m \\ b_1 & \cdots & b_n & d_1 & \cdots & d_m \end{bmatrix}.$$

Suppose  $E$  is not full-rank. Since  $\mathbf{a}$  and  $\mathbf{d}$  are non-zero vectors,  $E$  is not the zero matrix and hence it must have rank 1. In that case, the first row is linearly dependent on the second row. Since all  $E$ 's entries are non-negative, there exist  $\alpha, \beta > 0$  such that  $\alpha \cdot \mathbf{a} = \beta \cdot \mathbf{b}$  and  $\alpha \cdot \mathbf{c} = \beta \cdot \mathbf{d}$ . This implies:

$$\alpha \cdot \left(\sum_{i=1}^n a_i\right) = \beta \cdot \left(\sum_{i=1}^n b_i\right), \quad \alpha \cdot \left(\sum_{j=1}^m c_j\right) = \beta \cdot \left(\sum_{j=1}^m d_j\right).$$

Which in turn implies:

$$\beta \cdot \left(\sum_{i=1}^n b_i\right) = \alpha \cdot \left(\sum_{i=1}^n a_i\right) > \alpha \cdot \left(\sum_{j=1}^m c_j\right) = \beta \cdot \left(\sum_{j=1}^m d_j\right) \implies \sum_{i=1}^n b_i > \sum_{j=1}^m d_j$$

This contradicts the interlacing property. Hence,  $E$  must be full-rank. Let  $E'$  be a

$2 \times 2$  rank-2 minor of  $E$ . If  $E'$  is of the form  $E' = \begin{bmatrix} a_i & c_j \\ b_i & d_j \end{bmatrix}$  for some  $i, j$  we are done.

Otherwise, suppose if  $E'$  is of the form  $E' = \begin{bmatrix} a_i & a_j \\ b_i & b_j \end{bmatrix}$  or  $E' = \begin{bmatrix} c_i & c_j \\ d_i & d_j \end{bmatrix}$  then by the

exchange property, we can exchange one of the columns in  $E'$  with non-zero columns  $\begin{bmatrix} c_t \\ d_t \end{bmatrix}$  or  $\begin{bmatrix} a_s \\ b_s \end{bmatrix}$ , respectively, to get a rank-2 minor of the required form.  $\square$

### 7.3.3 The PIT Algorithm

For a polynomial  $f$  and an irreducible polynomial  $u$ , let  $e_u(f)$  denote the highest power of  $u$  in  $f$ . In other words,  $f = u^{e_u(f)} \cdot g$ , such that  $u \nmid g$ . If  $u \nmid f$ , then  $e_u(f) = 0$ . We define a polynomial  $\Phi$  with respect to two non-zero polynomials  $P, Q$  as follows:

**Definition 7.3.3.** Let  $P, Q \in \mathbb{F}[x_1, \dots, x_n]$  be two non-zero polynomials. Define a non-zero polynomial  $\Phi_{P,Q} \in \mathbb{F}[x_1, \dots, x_n]$  as:

$$\Phi_{P,Q} := \prod_{\substack{u,v \mid PQ \\ i \in [n]}} \text{Res}_{x_i}(u, v) \cdot \prod_{i \in [n]} \text{lc}_{x_i}(P) \cdot \prod_{i \in [n]} \text{lc}_{x_i}(Q),$$

where  $u, v$  are irreducible factors of  $P$  or  $Q$  such that  $(e_u(P), e_v(P))$  interlaces with  $(e_u(Q), e_v(Q))$ . Moreover, we only consider non-zero multiplicands.

The next Lemma shows that a non-zero of  $\Phi$  preserves non-similarity of polynomials.

**Lemma 7.3.4.** Let  $P, Q \in \mathbb{F}[x_1, x_2, \dots, x_n]$  be two polynomials such that  $P \approx Q$  and let  $\mathbf{a} \in \mathbb{F}^n$  such that  $\Phi_{P,Q}(\mathbf{a}) \neq 0$ . Then, there exists an  $i \in [n]$  such that  $P(x_i, \mathbf{a}_{-i}) \approx Q(x_i, \mathbf{a}_{-i})$ .

*Proof.* By our premise, we have  $P \approx Q$ . By uniqueness of factorization, without loss of generality, there exists an irreducible factor  $u$  of  $P$ , appearing with higher power in  $P$  than in  $Q$ . That is,  $e_u(P) > e_u(Q)$ . Let  $k = e_u(P)$  and  $\ell = e_u(Q)$ . We have  $k > \ell \geq 0$ . We get  $\ell = 0$ , when  $u$  does not divide  $Q$ . Let  $P = u^k \cdot G$  and  $Q = u^\ell \cdot H$ , for some

polynomials  $G, H$  such that  $u$  does not divide either of them. Define set  $T := \{v \mid v \text{ is an irreducible factor of } H \text{ and } e_v(P) \geq e_v(Q)\}$ . Then let

$$\begin{aligned} P &= u^k \cdot \left( \prod_{v \in T} v^{e_v(P)} \right) \cdot G' \\ Q &= u^\ell \cdot \left( \prod_{v \in T} v^{e_v(Q)} \right) \cdot H', \end{aligned}$$

where  $G', H'$  are the product of remaining polynomials from  $G$  and  $H$  respectively.

We choose  $i$  to be any element in  $\text{var}(u)$ , i.e.  $u$  depends on  $x_i$ . Note that  $\text{lc}_{x_i}(P) \neq 0$ , since  $u$  is a factor of  $P$  which depends on  $x_i$ . Since  $\text{lc}$  is multiplicative, we get that  $\text{lc}_{x_i}(P(x_i, \mathbf{a}_{-i})) = \text{lc}_{x_i}(u(x_i, \mathbf{a}_{-i}))^k \cdot \text{lc}_{x_i}(G(x_i, \mathbf{a}_{-i}))$ . From our premise, we also know that  $\Phi_{P,Q}(\mathbf{a}) \neq 0$ . Then by the definition of  $\Phi_{P,Q}$ , we get that  $\text{lc}_{x_i}(P(x_i, \mathbf{a}_{-i})) \neq 0$ , which implies that  $\text{lc}_{x_i}(u(x_i, \mathbf{a}_{-i})) \neq 0$ . Together with the fact that  $u$  has  $x_i$ -degree at least one, we conclude that  $u(x_i, \mathbf{a}_{-i})$  also has  $x_i$ -degree at least one. Suppose for the sake of contradiction that  $P(x_i, \mathbf{a}_{-i}) \sim Q(x_i, \mathbf{a}_{-i})$ . Then, we get

$$\begin{aligned} u(x_i, \mathbf{a}_{-i})^k \cdot \left( \prod_{v \in T} v^{e_v(P)}(x_i, \mathbf{a}_{-i}) \right) \cdot G'(x_i, \mathbf{a}_{-i}) &\sim u(x_i, \mathbf{a}_{-i})^\ell \cdot \left( \prod_{v \in T} v^{e_v(Q)}(x_i, \mathbf{a}_{-i}) \right) \cdot H'(x_i, \mathbf{a}_{-i}) \\ \implies u(x_i, \mathbf{a}_{-i})^{k-\ell} \cdot \left( \prod_{v \in T} v^{e_v(P)-e_v(Q)}(x_i, \mathbf{a}_{-i}) \right) \cdot G'(x_i, \mathbf{a}_{-i}) &\sim H'(x_i, \mathbf{a}_{-i}). \end{aligned}$$

Since  $k > \ell$  and  $\forall v \in T : e_v(P) \geq e_v(Q)$ , LHS is a proper polynomial in the above equation. Moreover,  $u(x_i, \mathbf{a}_{-i})$  divides LHS. Now since  $\text{LHS} \sim H'(x_i, \mathbf{a}_{-i})$  and  $u(x_i, \mathbf{a}_{-i})$  depends on  $x_i$ , we deduce that  $H'(x_i, \mathbf{a}_{-i})$  also depends on  $x_i$ . By uniqueness of factorization, we also deduce that  $u(x_i, \mathbf{a}_{-i})$  divides  $H'(x_i, \mathbf{a}_{-i})$ .

Let  $H' = v_1^{e_1} \cdots v_m^{e_m}$  be the irreducible factorization of  $H'$ , for some  $m \geq 1$  and  $e_j \geq 1$  for all  $j \in [m]$ , where each  $v_j$  is irreducible. Here  $e_j = e_{v_j}(Q)$  for each  $j \in [m]$ . Then  $H'(x_i, \mathbf{a}_{-i}) = v_1(x_i, \mathbf{a}_{-i})^{e_1} \cdots v_m(x_i, \mathbf{a}_{-i})^{e_m}$ , where  $v_j(x_i, \mathbf{a}_{-i})$ 's may not be irreducible anymore due to substitution. Recall that  $u$  does not divide  $H$  and hence it does not divide  $H'$  either. Since  $u$  is irreducible, we get that  $\gcd_{x_i}(u, v_j) = 1$ , for all  $j \in [m]$ . At the same time, recall that  $u(x_i, \mathbf{a}_{-i})$  divides  $H'(x_i, \mathbf{a}_{-i})$ . Since  $H'(x_i, \mathbf{a}_{-i})$  depends on  $x_i$ , this implies that  $u(x_i, \mathbf{a}_{-i})$  shares a non-trivial factor with some  $v_j(x_i, \mathbf{a}_{-i})$  which depends



on  $x_i$ . Thus, there exists some  $j \in [m]$  such that  $\gcd_{x_i}(u(x_i, \mathbf{a}_{-i}), v_j(x_i, \mathbf{a}_{-i})) \neq 1$ . By definition of  $H'$ ,  $v_j \notin T$  and hence  $e_{v_j}(P) < e_{v_j}(Q) = e_j$  for all  $j \in [m]$ . Recall that  $e_u(P) = k > \ell = e_u(Q)$ . Hence  $(e_u(P), e_{v_j}(P))$  interlaces with  $(e_u(Q), e_{v_j}(Q))$ . By our premise,  $\Phi_{P,Q}(\mathbf{a}) \neq 0$ . Then by the definition of  $\Phi_{P,Q}$ , we get that  $\text{Res}_{x_i}(u, v_j)(\mathbf{a}_{-i}) \neq 0$ . By further applying Corollary 6.3.4, we deduce that  $\gcd_{x_i}(u(x_i, \mathbf{a}_{-i}), v_j(x_i, \mathbf{a}_{-i})) = 1$ , which gives us a contradiction. Hence,  $P(x_i, \mathbf{a}_{-i}) \approx Q(x_i, \mathbf{a}_{-i})$ .  $\square$

The following is a technical lemma that will be used subsequently.

**Lemma 7.3.5.** *Let  $u, v \in \mathbb{F}[x_1, x_2, \dots, x_n]$  be two coprime and irreducible polynomials such that  $\text{var}(u) \cap \text{var}(v)$  is non-empty. And suppose we have two polynomials  $g = u^{a_1} \cdot v^{b_1}$  and  $h = u^{a_2} \cdot v^{b_2}$ , for some non-negative integers  $a_1, b_1, a_2, b_2$ . Define  $z := a_2 + b_2$  and  $t := a_1 + b_1$ . For any  $i \in \text{var}(u) \cap \text{var}(v)$ , let  $W := \gcd_{x_i}(g^z, h^t)$ . Finally, let  $E$  be the following matrix:*

$$E := \begin{bmatrix} a_1 & a_2 \\ b_1 & b_2 \end{bmatrix}.$$

Then

$$\frac{g^z}{W} = \begin{cases} u^{\det(E)} & \text{if } \det(E) \geq 0 \\ v^{-\det(E)} & \text{otherwise.} \end{cases} \quad \frac{h^t}{W} = \begin{cases} v^{\det(E)} & \text{if } \det(E) \geq 0 \\ u^{-\det(E)} & \text{otherwise.} \end{cases}$$

*Proof.* We have that:

$$\begin{aligned} g^z &= (u^{a_1} \cdot v^{b_1})^z = u^{a_1 a_2 + a_1 b_2} \cdot v^{a_2 b_1 + b_1 b_2} \\ h^t &= (u^{a_2} \cdot v^{b_2})^t = u^{a_1 a_2 + a_2 b_1} \cdot v^{a_1 b_2 + b_1 b_2} \end{aligned}$$

If  $\det(E) \geq 0$ , then  $a_1 b_2 \geq a_2 b_1$  and consequently  $W = u^{a_1 a_2 + a_2 b_1} \cdot v^{a_2 b_1 + b_1 b_2}$ . In that case,  $g^z/W = u^{a_1 b_2 - a_2 b_1} = u^{\det(E)}$  and  $h^t/W = v^{a_1 b_2 - a_2 b_1} = v^{\det(E)}$ . Otherwise, if  $\det(E) < 0$ , then  $a_2 b_1 > a_1 b_2$  and consequently  $W = u^{a_1 a_2 + a_1 b_2} \cdot v^{a_1 b_2 + b_1 b_2}$ . Then  $g^z/W = v^{a_2 b_1 - a_1 b_2} = v^{-\det(E)}$  and  $h^t/W = u^{a_2 b_1 - a_1 b_2} = u^{-\det(E)}$ .  $\square$

We now show that under certain non-degeneracy condition, a resultant of two factors of sparse polynomials is itself a factor of a (somewhat) sparse polynomial.

**Lemma 7.3.6.** *Let  $u \approx v \in \mathbb{F}[x_1, x_2, \dots, x_n]$  be two irreducible polynomials. Suppose there exist  $s$ -sparse, individual degree- $d$  polynomials  $g, h \in \mathbb{F}[x_1, x_2, \dots, x_n]$  such that the matrix*

$$E := \begin{bmatrix} e_u(g) & e_u(h) \\ e_v(g) & e_v(h) \end{bmatrix}$$

*has full rank. Then for any  $i \in [n]$  :  $\text{Res}_{x_i}(u, v)$  is a factor of a non-zero  $(sd)^{\mathcal{O}(d^3)}$ -sparse,  $\mathcal{O}(d^4)$ -individual degree polynomial.*

*Proof.* Consider any  $i \notin \text{var}(u) \cup \text{var}(v)$ . Then  $\text{Res}_{x_i}(u, v)$  is defined to be 1, which is trivially a factor of any sparse polynomial. Now consider any  $i \in \text{var}(u) \setminus \text{var}(v)$ . Then by definition,  $\text{Res}_{x_i}(u, v) = v^{\deg_{x_i}(u)}$ . Note that both  $e_u(g)$  and  $e_v(h)$  cannot be zero, as  $E$  has full rank. Therefore,  $v$  is factor of  $g$  or  $h$ , which are both  $s$ -sparse. Similarly,  $u$  is also a factor of  $g$  or  $h$  which implies  $\deg_{x_i}(u) \leq d$ . We deduce that  $\text{Res}_{x_i}(u, v)$  is a factor of an  $s^d$ -sparse polynomial. Similarly, we get the same conclusion for any  $i \in \text{var}(v) \setminus \text{var}(u)$ . We are now left with  $i \in \text{var}(u) \cap \text{var}(v)$ , for which we shall prove below.

Let us write  $g = u^{e_u(g)} \cdot v^{e_v(g)} \cdot A$  and  $h = u^{e_u(h)} \cdot v^{e_v(h)} \cdot B$ , for some polynomials  $A, B \in \mathbb{F}[x_1, x_2, \dots, x_n]$  co-prime to both  $u$  and  $v$ . Let  $g' = u^{e_u(g)} \cdot v^{e_v(g)}$  and  $h' = u^{e_u(h)} \cdot v^{e_v(h)}$ . Further, let  $z = e_u(h) + e_v(h)$  and  $t = e_u(g) + e_v(g)$ . Consider polynomials  $g^z = (g')^z \cdot A^z$  and  $h^t = (h')^t \cdot B^t$ . Since both  $g, h$  have individual degree  $d$ , we know that  $e_u(g), e_v(g), e_u(h), e_v(h) \leq d$  and hence  $s, t \leq 2d$ . Pick any  $i \in \text{var}(u) \cap \text{var}(v)$  and consider  $\gcd_{x_i}(g^z, h^t)$ . Define  $W := \gcd_{x_i}((g')^z, (h')^t)$  and  $Y := \gcd_{x_i}(A^z, B^t)$ . Since  $g', h'$  are co-prime to both  $A$  and  $B$ , we deduce that

$$\gcd_{x_i}(g^z, h^t) = W \cdot Y.$$

By our premise, we have  $\det(E) \neq 0$ . Without loss of generality, let us assume  $\det(E) > 0$ . The other case follows similarly. Using Lemma 7.3.5, we get that  $(g')^z/W = u^{\det(E)}$  and  $(h')^t/W = v^{\det(E)}$ . Therefore, we can write

$$\begin{aligned} g^z &= W \cdot Y \cdot u^{\det(E)} \cdot \frac{A^z}{Y}, \\ h^t &= W \cdot Y \cdot v^{\det(E)} \cdot \frac{B^t}{Y}. \end{aligned}$$

Note that  $A^z/Y$  and  $B^t/Y$  are proper polynomials by definition of  $Y$ . Let  $\ell = \deg_{x_i}(\gcd(g^z, h^t))$ . By Theorem 7.2.1, there exists  $k \geq 0$  such that:

$$S_{x_i}(\ell, g^z, h^t) = W \cdot Y \cdot \text{Res}_{x_i} \left( u^{\det(E)} \cdot \frac{A^z}{Y}, v^{\det(E)} \cdot \frac{B^t}{Y} \right) \cdot \text{lc}_{x_i}(W \cdot Y)^k.$$

Since both  $g^z$  and  $h^t$  are  $s^{2d}$ -sparse with individual degree at most  $2d^2$ , by Observation 6.3.9,  $S_{x_i}(\ell, g^z, h^t)$  is  $(sd)^{\mathcal{O}(d^3)}$ -sparse with individual degree at most  $8d^4$ . Furthermore, observe that by definition:

$$\begin{aligned} \gcd_{x_i} \left( u^{\det(E)} \cdot \frac{A^z}{Y}, v^{\det(E)} \cdot \frac{B^t}{Y} \right) = 1 &\implies \text{Res}_{x_i} \left( u^{\det(E)} \cdot \frac{A^z}{Y}, v^{\det(E)} \cdot \frac{B^t}{Y} \right) \neq 0 \\ &\implies S_{x_i}(\ell, g^z, h^t) \neq 0. \end{aligned}$$

Finally, since  $\det(E)$  is a positive integer, we use Lemma 6.3.5 to deduce that

$$\text{Res}_{x_i}(u, v) \mid \text{Res}_{x_i} \left( u^{\det(E)} \cdot \frac{A^z}{Y}, v^{\det(E)} \cdot \frac{B^t}{Y} \right).$$

Hence, we conclude that  $\text{Res}_{x_i}(u, v)$  is a factor of a non-zero  $(sd)^{\mathcal{O}(d^3)}$ -sparse sub-resultant polynomial.  $\square$

Using the above, we conclude that while the multiplicands in the polynomial  $\Phi_{P,Q}$  may not themselves be sparse, they are factors of (some) sparse polynomials. Consequently,  $\Phi_{P,Q}$  can be hit by a hitting set generator for sparse polynomials.

**Lemma 7.3.7.** *Let both  $P, Q \in \mathbb{F}[x_1, x_2, \dots, x_n]$  be products of  $s$ -sparse, individual degree- $d$  polynomials and let  $\mathcal{G}$  be a generator for  $(sd)^{\mathcal{O}(d^3)}$ -sparse,  $\mathcal{O}(d^4)$ -individual degree polynomials. Then  $\Phi_{P,Q}(\mathcal{G}) \neq 0$ .*

*Proof.* Let  $P = \prod_{j \in [r]} g_j$  and  $Q = \prod_{k \in [m]} h_k$ , where  $g_j, h_k$  are  $s$ -sparse polynomials of individual degree  $d$ , for all  $j \in [r], k \in [m]$ . By definition,  $\Phi_{P,Q}$  has two types of multiplicands. We will show that  $\mathcal{G}$  hits both types.

For the first type, let  $u, v$  be any irreducible factors of  $P$  or  $Q$  such that  $(e_u(P), e_v(P))$  interlaces with  $(e_u(Q), e_v(Q))$ . We wish to show that  $\text{Res}_{x_i}(u, v) \neq 0 \implies \text{Res}_{x_i}(u, v)(\mathcal{G}) \neq$

0. Observe that,

$$\begin{aligned} e_u(P) &= \sum_{j=1}^r e_u(g_j) & e_u(Q) &= \sum_{k=1}^m e_u(h_k) \\ e_v(P) &= \sum_{j=1}^r e_v(g_j) & e_v(Q) &= \sum_{k=1}^m e_v(h_k). \end{aligned}$$

By definition, each of these e-values is a non-negative integer. Therefore by Lemma 7.3.2, there exists  $j \in [r], k \in [m]$  such that  $E := \begin{bmatrix} e_u(g_j) & e_u(h_k) \\ e_v(g_j) & e_v(h_k) \end{bmatrix}$  has full rank. Then by Lemma 7.3.6, for any  $i \in [n] : \text{Res}_{x_i}(u, v)$  is factor of some  $(sd)^{\mathcal{O}(d^3)}$ -sparse,  $\mathcal{O}(d^4)$ -individual degree polynomial. Since,  $\mathcal{G}$  is a generator for such polynomials, we deduce that  $\text{Res}_{x_i}(u, v)(\mathcal{G}) \neq 0$ .

For the second type, by multiplicative property of lc, we know that for any  $i \in [n]$ ,

$$\text{lc}_{x_i}(P) = \prod_{j \in [r]} \text{lc}_{x_i}(g_j) \quad \text{and} \quad \text{lc}_{x_i}(P)(\mathcal{G}) = \prod_{j \in [r]} \text{lc}_{x_i}(g_j)(\mathcal{G}).$$

Note that  $\text{lc}_{x_i}(g_j)$  is also  $s$ -sparse with individual degree  $d$ . Hence,  $\text{lc}_{x_i}(g_j)(\mathcal{G}) \neq 0$ , for all  $j \in [r], i \in [n]$ . This implies  $\text{lc}_{x_i}(P)(\mathcal{G}) \neq 0$ , for all  $i \in [n]$  (whenever  $\text{lc}_{x_i}(P) \neq 0$ ). Similarly, we can show  $\text{lc}_{x_i}(Q)(\mathcal{G}) \neq 0$ , for all  $i \in [n]$ . We conclude that  $\Phi_{P,Q}(\mathcal{G}) \neq 0$ .  $\square$

For a generator  $\mathcal{G} = (\mathcal{G}_1, \dots, \mathcal{G}_n)$ , we define  $\mathcal{G}_{-i} := (\mathcal{G}_1, \dots, \mathcal{G}_{i-1}, \mathcal{G}_{i+1}, \dots, \mathcal{G}_n)$ . For a polynomial  $f \in \mathbb{F}[x_1, x_2, \dots, x_n]$ , we define  $f(x_i, \mathcal{G}_{-i}) := f(\mathcal{G}_1, \dots, \mathcal{G}_{i-1}, x_i, \mathcal{G}_{i+1}, \dots, \mathcal{G}_n)$ . By combining the result with Lemma 7.3.4 we obtain the following:

**Corollary 7.3.8.** *Let  $P, Q \in \mathbb{F}[x_1, x_2, \dots, x_n]$  be products of  $s$ -sparse, individual degree- $d$  polynomials such that  $P \approx Q$ . Let  $\mathcal{G} = (\mathcal{G}_1, \dots, \mathcal{G}_n) : \mathbb{F}^t \rightarrow \mathbb{F}^n$  be a generator for  $(sd)^{\mathcal{O}(d^3)}$ -sparse,  $\mathcal{O}(d^4)$ -individual degree polynomials. Then there exists an  $i \in [n]$  such that  $P(x_i, \mathcal{G}_{-i}) \approx Q(x_i, \mathcal{G}_{-i})$ .*

*Proof.* By Lemma 7.3.7, we get that  $\Phi_{P,Q}(\mathcal{G}) \neq 0$ . From Lemma 6.2.3, we know that there exists a set  $W \subseteq \mathbb{F}$  of large enough size such that  $\mathcal{G}(W^t) \subseteq \mathbb{F}^n$  is a hitting set for  $\Phi_{P,Q}$ . In particular, there exists  $\mathbf{b} \in W^t$  such that for  $\mathbf{a} := \mathcal{G}(\mathbf{b})$ , we have  $\Phi_{P,Q}(\mathbf{a}) \neq 0$ .

By Lemma 7.3.4, there exists an  $i \in [n]$  such that  $P(x_i, \mathbf{a}_{-i}) \approx Q(x_i, \mathbf{a}_{-i})$ . Now suppose  $P(x_i, \mathcal{G}_{-i}) \sim Q(x_i, \mathcal{G}_{-i})$ . Then have that

$$P(\mathcal{G}_1(\mathbf{b}), \dots, \mathcal{G}_{i-1}(\mathbf{b}), x_i, \mathcal{G}_{i+1}(\mathbf{b}), \dots, \mathcal{G}_n(\mathbf{b})) \sim Q(\mathcal{G}_1(\mathbf{b}), \dots, \mathcal{G}_{i-1}(\mathbf{b}), x_i, \mathcal{G}_{i+1}(\mathbf{b}), \dots, \mathcal{G}_n(\mathbf{b})).$$

This implies that  $P(x_i, \mathbf{a}_{-i}) \sim Q(x_i, \mathbf{a}_{-i})$ , which is a contradiction. Hence,  $P(x_i, \mathcal{G}_{-i}) \approx Q(x_i, \mathcal{G}_{-i})$ .  $\square$

Finally, we can prove the main result of the section.

**Theorem 7.3.9.** *There exists a deterministic algorithm that given  $n, d, s$  and a black-box access to a  $\Sigma^{[2]}\Pi\Sigma\Pi^{[\text{ind-deg } d]}$  circuit of size  $s$  determines if  $C \equiv 0$ , in time  $\text{poly}((sd)^{d^3}, n)$ . Algorithm 3 provides the outline.*

---

**Algorithm 3:** Black-box PIT algorithm for class  $\Sigma^{[2]}\Pi\Sigma\Pi^{[\text{ind-deg } d]}$

---

**Input:** A black-box access to a polynomial  $f(x_1, \dots, x_n)$  computed by

$\Sigma^{[2]}\Pi\Sigma\Pi^{[\text{ind-deg } d]}$  circuit

**Output:** “ZERO”, if  $f$  is identically zero and “NON-ZERO”, otherwise.

- 1 Call Lemma 6.2.10 to get generator  $\mathcal{G}$  of seed-length 1 for  $n$ -variate polynomials of sparsity  $\leq (sd)^{\mathcal{O}(d^3)}$  and individual degree  $\leq \mathcal{O}(d^4)$ .
  - 2 **for**  $i \leftarrow 1$  **to**  $n$  **do**
    - 3     Compute the bivariate polynomial  $f(x_i, \mathcal{G}_{-i})$ .
    - 4     Call Lemma 6.2.1 to do brute-force black-box PIT for  $f(x_i, \mathcal{G}_{-i})$ .
    - 5     **if**  $f(x_i, \mathcal{G}_{-i}) \not\equiv 0$  **then return** “NON-ZERO”.
  - 6 **end**
  - 7 **return** “ZERO”.
- 

*Proof.* We now analyze the correctness and runtime complexity of Algorithm 3.

**Correctness:** Note that  $f \equiv 0 \implies f(x_i, \mathcal{G}_{-i}) \equiv 0$  trivially, for all  $i \in [n]$ . Thus, the algorithm outputs “ZERO” in this case, as desired. Now suppose  $f \not\equiv 0$ . Let  $f = P + Q$ , where both  $P, Q$  are product of  $s$ -sparse, individual degree  $d$  polynomials. If  $P \approx Q$ , then Corollary 7.3.8 implies that there exists an  $i \in [n]$  such that  $P(x_i, \mathcal{G}_{-i}) \approx Q(x_i, \mathcal{G}_{-i})$ .

In particular,  $P(x_i, \mathcal{G}_{-i}) \neq -Q(x_i, \mathcal{G}_{-i})$ . Since  $f(x_i, \mathcal{G}_{-i}) = P(x_i, \mathcal{G}_{-i}) + Q(x_i, \mathcal{G}_{-i})$ , we deduce that  $f(x_i, \mathcal{G}_{-i}) \neq 0$  in this case. Now suppose  $f \neq 0$  but  $P \sim Q$ . Let  $P = cQ$ , for some  $c \in \mathbb{F}$ . Then  $f = (c+1)Q$ , where  $c \neq -1$  and  $Q \neq 0$ . This means that there exists an  $i \in [n]$  such that  $\text{lc}_{x_i}(Q) \neq 0$ . Using Lemma 7.3.7, we know that  $\Phi_{P,Q}(\mathcal{G}) \neq 0$  and thus by definition of  $\Phi_{P,Q}$ , we get  $\text{lc}_{x_i}(Q)(\mathcal{G}) \neq 0$ . This implies that  $Q(x_i, \mathcal{G}_{-i}) \neq 0$ . We conclude that  $f(x_i, \mathcal{G}_{-i}) = (c+1)Q(x_i, \mathcal{G}_{-i}) \neq 0$ . Thus whenever  $f \neq 0$ , the algorithm outputs “NON-ZERO”.

**Time complexity:** By Lemma 6.2.10, degree of generator  $\mathcal{G}$  is  $\text{poly}((sd)^{d^3}, n)$ . Note that  $f$  has individual degree at most  $sd$  and thus  $f(x_i, \mathcal{G}_{-i})$  has individual degree  $\leq sd \cdot \deg(\mathcal{G})$ . Then by Lemma 6.2.1, testing non-zeroness of the bivariate polynomial  $f(x_i, \mathcal{G}_{-i})$  takes only  $\text{poly}((sd)^{d^3}, n)$  time. The  $n$  iterations only add a factor of  $n$ .  $\square$

## 7.4 Discussion

We introduced a new model for PIT in this chapter, called the  $\Sigma^{[k]}\Pi\Sigma\Pi^{[\text{ind-deg } d]}$  circuits. This is a generalization of the  $\Sigma^{[k]}\Pi\Sigma\Pi^{[\delta]}$  model studied in [DDS21]. Instead of sparse, constant total-degree  $\delta$  polynomials in the bottom  $\Sigma\Pi$ , this model has sparse, constant individual-degree  $d$  polynomials in the bottom  $\Sigma\Pi$ .

In this chapter, we solved this new model for  $k = 2$  and any constant  $d$ . For  $k = 2$ , the question of PIT reduces to testing whether a product of sparse, constant individual degree polynomials equals another such product. Here uniqueness of factorization can be exploited. We don’t need to preserve the exact factorization pattern of these two products, but it suffices to preserve just the co-primality of the distinct irreducible factors in these products (under a variable reduction map). Towards that end, the broad idea here is to construct a generator which hits all the appropriate resultants. The main problem in this approach is that these irreducible factors may not be polynomially sparse. Factors of sparse polynomials are not known to be polynomially sparse, unless one proves the sparsity conjecture (Conjecture 1.2.3). However, in this chapter we use the problem against itself! We don’t show that each resultant is sparse, but we show that it is a factor of some another

sparse polynomial (the subresultant). Thus, we are able to hit the resultants indirectly, which gives us a poly-time blackbox PIT for  $\Sigma^{[2]}\Pi\Sigma\Pi^{[\text{ind-deg } d]}$  circuits.

For  $k = 3$ , we have three products which is a more complex scenario and seems to require new ideas. We leave it as an open problem to design a poly-time PIT algorithm for  $\Sigma^{[k]}\Pi\Sigma\Pi^{[\text{ind-deg } d]}$  circuits with bounded  $k$  and  $d$ , for any  $k \geq 3$ . To the best of our knowledge, even whitebox PIT for  $k = 3$  and  $d = 1$  is open!





## Chapter 8

# PIT for sum of ROABPs and its Border Class

The main model of computation for PIT in this chapter will be sum of ROABPs. Recall the definition of ROABPs and sum of ROABPs from Chapter 6. For a single ROABP, we have a well known poly-time PIT algorithm in the whitebox setting [RS05]. However, in the blackbox regime, we only have quasi-polynomial time PIT algorithms [FS13b, AGKS15] and no known poly-time algorithms. Therefore, various restrictions of ROABPs are studied. There are three size parameters for an ROABP: width, number of variables, degree. [GKS17] developed a poly-time graybox (known variable order) PIT for constant width ROABPs. The blackbox setting here is still open. One can also restrict the number of variables to  $O(\log s)$ , where  $s$  is the overall size of ROABP. Lemma 6.4.5 shows that PIT for log-variate ROABP subsumes PIT for  $\sum \wedge \sum$  circuits. However, even this PIT is still open. In this chapter, for few results we shall restrict both width and number of variables, however we will work for the more general model of sum of ROABPs.

The sum of ROABPs model was first studied by [GKST17]. For a constant number of ROABPs, they give the first poly-time whitebox, and only a *quasi*-poly time blackbox PIT algorithm. In this chapter we prove the following main PIT result.

*Blackbox Polynomial Identity Testing for sum of constantly-many, log-variate*

*constant-width ROABPs is in polynomial time.*

This is formally stated in Corollary 8.3.8. In addition, with just a single restriction of log-variate, we still get an improvement over [GKST17] in Corollary 8.3.9. More generally, we also give efficient PIT for the *border* version. In subsequent works, the combined restrictions on width and degree have also been studied for the orbit of ROABPs [ST21a, BG21].

All the results in this chapter hold for any field  $\mathbb{F}$ . Moreover, all our PIT algorithms here are completely blackbox and assume no knowledge about the variable order of an ROABP.

## 8.1 Previous works and motivation

In this section, we will discuss the major motivations behind this work by showing connections of ROABP with other algebraic models of computation. We refer the reader to Chapter 6 for formal definitions of these models.

**ABPs:** It is well known that ABPs subsume determinants and formulas. In turn, algebraic formulas subsume constant depth circuits (see [BOC92] and [Nis91, Lem. 1]). The combined restrictions on variables and width still gives interesting sub-models for ABPs. For example [AGS19, Thm.22] show that even solving PIT for log-variate width-2 ABPs will almost solve the complete PIT problem. The ROABP model is also quite nontrivial as a poly-time blackbox PIT is still open. Table 8.1 gives a comprehensive comparison between the time complexities of previous works on ROABP model and this work. In the table, algorithms of [FS13b] and [GKS17] work only for known variable orders. Moreover, the PIT of [GKS17] works only for fields of characteristic either zero or larger than  $ndr^{\log n}$ . The work of [FSS14] gives quasi-poly time PIT under the restriction of multilinearity or constant individual degree. We only mention our border algorithms in Table 8.1. Naturally, we also solve PIT for the base class of sum of  $c$  ROABPs.

Model	Setting	Time	Reference
$\sum \wedge \sum$	Blackbox	$(nd)^{O(\log n)}$	[ASS13]
$\sum \wedge \sum$	Blackbox	$\text{poly}(d, 2^n)$	[FGS18]
ROABP	Graybox	$(ndr)^{O(\log n)}$	[FS13b]
ROABP	Blackbox	$n^{O(d \log r \log n)}$	[FSS14]
ROABP	Blackbox	$(ndr)^{O(\log n)}$	[AGKS15]
ROABP	Graybox	$O(ndr^{\log n})$	[GKS17]
Sum of $c$ ROABPs	Blackbox	$(ndr)^{O(c 2^c \log(ndr))}$	[GKST17]
Border of sum of $c$ ROABPs	Blackbox	$(2^n (nd)^{\log n} r^{3^c \log n})^{O(c)}$	This work
Border of sum of $c$ ROABPs	Blackbox	$\text{poly}(d^c, r^{nc 3^c})$	This work

Table 8.1: Time complexities of different PIT algorithms related to  $n$ -variate, degree  $d$  and width  $r$  ROABP model.

**Log-variate:** There has been a recent line of work on ‘Bootstrapping variables’ in algebraic circuits. [AGS19] prove that solving blackbox PIT for circuits that depend only on the first  $\log^{oc}$  variables is sufficient to solve blackbox PIT for general circuits. Here  $c$  is a constant and  $\log^{oc}$  is a composition of  $c$  logarithms. [KST19] and [GKSS19] further showed that even saving on *one* evaluation point from the brute-force hitting-set of constant-variate algebraic circuits would solve general PIT. Although such bootstrapping results are not known for ROABPs, nonetheless log-variate ROABP is still an open interesting model for the reasons discussed below.

The well studied diagonal depth-3 model ( $\sum \wedge \sum$ ) is one of the lower hanging fruits in PIT. [FGS18] were able to utilize low-variate setting to give the first poly-time blackbox PIT for log-variate diagonal depth-3 circuits. The natural extension is to solve blackbox PIT for log-variate ROABPs. In fact, it can be shown that PIT for log-variate *commutative* ROABPs implies PIT for the general multivariate diagonal depth-3 model using the results of [FS13a, FSS14]. See Lemma 6.4.5 for details. Making progress in this direction has been the key motivation behind the tools and techniques developed in this work.

**Constant width:** The sum of constant-width ROABPs model is more expressive than that of a single constant-width ROABP. [KNS16] observed that even a sum of two width-3 ROABPs cannot be computed by a single constant-width ROABP which is stated as Fact 8.1.1 here. Thus, this model is nontrivial and blackbox PIT for it is still open. Even in the log-variate setting, sum of two width-3 ROABPs will require a single ROABP of

super-constant width. Thus, sum of constantly many, constant width, log-variate ROABPs lacked a poly-time blackbox PIT, which we solve in Corollary 8.3.8.

**Fact 8.1.1** ([KNS16], Thm. 7). There is an explicit family of  $3n$ -variate multilinear polynomials  $\{g_n\}_{n \geq 1}$  which is computable by sum of two width-3 ROABPs of size  $\Theta(n)$ , but any single ROABP computing  $g$  must have width  $2^{\Omega(n)}$ .

**PIT for Border:** Border of a class can offer additional computational power. [For16] gives an interesting example. Consider the class of polynomials which are of the form  $\alpha \ell_1^d + \beta \ell_2^d$ , where  $\ell_1, \ell_2 \in \mathbb{F}[\mathbf{x}]$  are homogeneous linear polynomials. For  $d \geq 3$ , it can be shown that  $x^{d-1}y$  cannot be expressed as  $\alpha \ell_1^d + \beta \ell_2^d$ . However,  $x^{d-1}y$  can be computed in the border of this class, as shown below

$$g := \frac{1}{d\epsilon}((x + \epsilon y)^d - x^d)$$

$$\lim_{\epsilon \rightarrow 0} g = x^{d-1}y$$

The fundamental problem in border complexity is to understand this difference in the computational power of a class  $\mathcal{C}$  and its border class  $\overline{\mathcal{C}}$ . Generally, one would like to understand whether  $\text{VP} = \overline{\text{VP}}$ , where VP is the class of polynomials with polynomial sized algebraic circuits. However, this question is open even for more restricted classes like diagonal depth-3 circuits, depth-3 circuits, ABPs etc. Border classes play an important role in Mulmuley's 'GCT approach' to attack  $P \neq NP$  conjecture [MS01, MS08], and in 'GCT Chasm' [Mul12b, Mul12a]. In particular, showing  $\overline{\text{VP}} \neq \text{VNP}$  is an important structural question. We refer the reader to [BIZ18] for an excellent discussion on border complexity.

An interesting question in border complexity is to come up with PIT algorithm for a border class. Of course to solve PIT for a border class  $\overline{\mathcal{C}}$ , we necessarily need to solve PIT for  $\mathcal{C}$ . However for those circuit classes where we have an efficient PIT algorithm for  $\mathcal{C}$  and it is not known whether  $\mathcal{C} = \overline{\mathcal{C}}$ , it is an interesting problem to solve PIT for  $\overline{\mathcal{C}}$ .

For the algebraic class of ROABPs, one can show that the border does not offer any additional computation power, that is,  $\overline{\text{ROABP}} = \text{ROABP}$  (Lemma 6.5.1). However, it is

not clear if sum of constantly many ROABPs is equal to its border class. See Section 8.4 for a discussion on this. Therefore PIT for the border class of sum of ROABPs is an interesting problem. We solve it for sum of constantly-many, constant width, log-variate ROABPs.

## 8.2 PIT for Degree-preserving sum of ROABPs

In this section, we first give PIT for a single log-variate constant width ROABP (Lemma 8.2.8). This is a consequence of our main structure theorem that concerns with homogenizing an ROABP in same width (Theorem 8.2.4). More generally, we get PIT for what we call a degree-preserving sum of ROABPs, where each ROABP is of constant width and log-variate (Theorem 8.2.9). We formally define degree-preserving sum below.

**Definition 8.2.1** (Degree-preserving sum). *For  $k \in \mathbb{N}$ , let  $f_1(\mathbf{x}), f_2(\mathbf{x}) \dots f_k(\mathbf{x})$  be any  $k$  polynomials in  $\mathbb{F}[x_1, \dots, x_n]$ . We call  $\sum_{i=1}^k f_i(\mathbf{x})$ , a degree-preserving sum, if for  $f(\mathbf{x}) := \sum_{i=1}^k f_i(\mathbf{x})$ , we have  $\deg(f) = \max_i \deg(f_i)$ .*

In other words, the highest degree term does not cancel in a degree-preserving sum. We remark that the number of ROABPs in Theorem 8.2.9 is arbitrary and bounded only by the size of input. This is in contrast with the results in next section and that of [GKST17], where the number of ROABPs is kept constant due to the double exponential dependence on it in the runtime. However, there the sum is not restricted to be degree-preserving.

### 8.2.1 Syntactically Homogeneous ROABP

An ROABP is called *syntactically homogeneous*, if for any two nodes  $(u, v)$  in the ROABP, as source and sink respectively, the polynomial computed from  $u \rightsquigarrow v$  is homogeneous. Clearly, every syntactically homogeneous ROABP is an ROABP computing a homogeneous polynomial, but every ROABP computing a homogeneous polynomial is not syntactically homogeneous. This is because some edge label in the ROABP may be inhomogeneous or some intermediate path may be computing an inhomogeneous polynomial, which cancels out in the end.

Throughout this chapter, we work with unknown variable order of ROABP. If the ROABP computes a polynomial over  $\mathbb{F}[\mathbf{x}]$ , we assume an arbitrary variable order  $(y_1, \dots, y_n)$ , where for all  $i \in [n]$ ,  $y_i = x_{\pi(i)}$  for some unknown permutation  $\pi : [n] \rightarrow [n]$ .

**Definition 8.2.2** (Syntactic homogeneity). *Let  $f(\mathbf{x}) \in \mathbb{F}[\mathbf{x}]$  be computed by an ROABP  $D(\mathbf{x})$  of width  $r$  in the variable order  $(y_1, \dots, y_n)$ . Let  $D(\mathbf{x}) =: \prod_{i=1}^n D_i(y_i)$ , where  $D_1(y_1) \in \mathbb{F}^{1 \times r}[y_1]$ ,  $D_n(y_n) \in \mathbb{F}^{r \times 1}[y_n]$ , and  $D_i \in \mathbb{F}^{r \times r}[y_i]$  for  $1 < i < n$ .*

*We call ROABP  $D(\mathbf{x})$ , syntactically homogeneous, if for all  $1 \leq i < n$ , each entry in the subproduct row-vector  $D_{\leq i} := \prod_{j=1}^i D_j \in \mathbb{F}^{1 \times r}[\mathbf{y}_{\leq i}]$ , is a homogeneous polynomial and so is each entry in the subproduct column-vector  $D_{> i} := \prod_{j=i+1}^n D_j \in \mathbb{F}^{r \times 1}[\mathbf{y}_{> i}]$ .*

Although in this work, we only need Definition 8.2.2, it can be shown that it is equivalent to the informal definition of syntactic homogeneity stated at the start of this section. To see this observe that, Definition 8.2.2 clearly follows from homogeneity of the polynomials  $[u \rightsquigarrow v]$  computed from  $u$  to  $v$ , for *any* two nodes  $(u, v)$  in ROABP. For the other side, let  $V$  be the vertex set of the layer which contains  $u$ . Then, note that  $[s \rightsquigarrow v] = \sum_{w:w \in V} [s \rightsquigarrow w] \cdot [w \rightsquigarrow v]$ . By Definition 8.2.2,  $[s \rightsquigarrow v]$  is homogeneous and so is each  $[s \rightsquigarrow w]$ . Also, the set of polynomials  $\{[s \rightsquigarrow w] \mid w \in V\}$  is  $\mathbb{F}$ -linearly independent by Nisan's characterization. Now, apply Lemma 8.2.3 below to get each  $[w \rightsquigarrow v]$  (in particular  $[u \rightsquigarrow v]$ ) to be homogeneous.

We first prove the following lemma, which we will need in the proof of Theorem 8.2.4.

**Lemma 8.2.3.** *Let  $\mathbf{y}$  and  $\mathbf{z}$  be a partition of variable set  $\mathbf{x}$ . Suppose  $f \in \mathbb{F}[\mathbf{x}]$  is a homogeneous polynomial of degree  $d$  having a variable disjoint decomposition as  $f = \sum_{i=1}^r f_i g_i$ , where for all  $i \in [r]$ ,  $f_i \in \mathbb{F}[\mathbf{y}]$  and  $g_i \in \mathbb{F}[\mathbf{z}]$ . Suppose  $f_1, \dots, f_r$  are  $\mathbb{F}$ -linearly independent and each  $f_i$  is also a homogeneous polynomial. Then, for each  $i \in [r]$ ,  $g_i$  is also a homogeneous polynomial.*

*Proof.* For the sake of contradiction, suppose there exists a  $g_k$ , for some  $k \in [r]$ , which is not homogeneous. Let  $f_k$  be its corresponding polynomial which is homogeneous and has degree, say  $d_k$ . Since  $f$  is homogeneous of degree  $d$ , let  $g_k = g_k^{[d-d_k]} + g_k^{[\neq(d-d_k)]}$ , where

$g_k^{[d-d_k]}$  is the degree  $(d - d_k)$  homogeneous part of  $g_k$  and  $g_k^{[\neq(d-d_k)]}$  is the rest of the polynomial. We will prove that the latter part has to be zero.

Let  $\mathbf{z}^{\mathbf{a}}$  be any monomial in  $g_k^{[\neq(d-d_k)]}$  with coefficient, say  $c_k \neq 0$ , where degree of monomial  $|\mathbf{a}|_1 \neq d - d_k$ . The nonzero term  $f_k \cdot c_k \mathbf{z}^{\mathbf{a}}$  in  $f$  has to get canceled since it is of degree  $d_k + |\mathbf{a}|_1 \neq d$ . Observe that this term can get canceled only by product of  $\mathbf{z}^{\mathbf{a}}$  with those  $f_i$  that have degree  $d_k$  (simply by variable disjointedness & degree comparison). For  $\ell \leq r$ , let  $f_{i_1}, f_{i_2}, \dots, f_{i_\ell}$  be the polynomials in  $\{f_1, \dots, f_r\}$  of degree exactly  $d_k$ . Let  $\text{coeff}(g_i)(\mathbf{z}^{\mathbf{a}}) =: c_i$ , for  $i \in [r]$ , where  $c_i$  can be possibly zero except for  $c_k$ . Then,

$$\begin{aligned} f_{i_1} \cdot (c_{i_1} \mathbf{z}^{\mathbf{a}}) + f_{i_2} \cdot (c_{i_2} \mathbf{z}^{\mathbf{a}}) + \dots + f_{i_\ell} \cdot (c_{i_\ell} \mathbf{z}^{\mathbf{a}}) &= 0 \\ \Rightarrow c_{i_1} f_{i_1} + c_{i_2} f_{i_2} + \dots + c_{i_\ell} f_{i_\ell} &= 0. \end{aligned}$$

Since  $c_k \neq 0$ , this contradicts  $\mathbb{F}$ -linear independence of  $f_1, \dots, f_r$ . Thus,  $g_k^{[\neq(d-d_k)]}$  is zero. Hence,  $\forall k \in [r]$ ,  $g_k$  is a homogeneous polynomial of degree  $d - \deg(f_k)$ .  $\square$

If a homogeneous polynomial  $f$  is computed by an ROABP of width  $w$ , then the optimal width ROABP for  $f$  constructed using Nisan's characterization (Lemma 6.4.2) has width  $r \leq w$ . In the following theorem, we prove that it is also syntactically homogeneous.

**Theorem 8.2.4** (Structure Theorem). *Let  $f(\mathbf{x}) \in \mathbb{F}[\mathbf{x}]$  be a degree  $d$  homogeneous polynomial computed by an ROABP  $C(\mathbf{x})$  of width  $w$  in the variable order  $(y_1, \dots, y_n)$ . Then,  $f$  also has a syntactically homogeneous ROABP  $D(\mathbf{x}) = \prod_{i=1}^n D_i(y_i)$  of optimal width  $r \leq w$  in the same variable order. Moreover,  $\forall i \in [n]$ , each entry in  $D_i(y_i)$  is merely a monomial in  $y_i$ .*

*Proof.* If  $f$  is computed by a width  $w$  ROABP  $C(\mathbf{x})$ , it will also have an optimal ROABP  $D(\mathbf{x})$  of width  $r \leq w$  constructed using Nisan's characterization. Here, we follow the construction as presented in [GKST17, Lem. 2.8]. For all  $i \in [n - 1]$ , we can write  $f$  as  $f = D_{\leq i} \cdot D_{> i} = \sum_{j=1}^r g_j(\mathbf{y}_{\leq i}) h_j(\mathbf{y}_{> i})$ . Fix  $i$ . Nisan's characterization picks the entries of  $D_{\leq i}$  to be  $r$   $\mathbb{F}$ -linearly independent polynomials  $g_1, \dots, g_r \in \mathbb{F}[\mathbf{y}_{\leq i}]$  and entries of  $D_{> i}$  to another  $r$   $\mathbb{F}$ -linearly independent polynomials  $h_1, \dots, h_r$ . Recall definition of  $\text{span}_i(f)$  in Definition 6.4.1. By construction, for each  $j \in [r]$ ,  $h_j =: f_{(\mathbf{y}_{\leq i}, \mathbf{e}_j)}$ , where

$\{\mathbf{e}_1, \dots, \mathbf{e}_r\} := \text{span}_i(f)$ . Observe that if  $f$  is a homogeneous polynomial, then so is each coefficient polynomial  $h_j = f_{(\mathbf{y}_{\leq i}, \mathbf{e}_j)}$ . Since  $f$  is a homogeneous polynomial and  $h_j$ 's are  $\mathbb{F}$ -linearly independent homogeneous polynomials, this forces each  $g_j$  to be also homogeneous, as proved in Lemma 8.2.3. Thus, for all  $i \in [n-1]$ , each entry in  $D_{\leq i}$  and  $D_{>i}$  is a homogeneous polynomial.

We now prove the second part of the theorem, that every entry in each intermediate matrix  $D_i(y_i)$  is a monomial in  $y_i$ . For  $D_1$ , consider the partition  $D = D_1 \cdot D_{>1}$ . By syntactic homogeneity proved above, each entry of  $D_1$  is homogeneous and thus is of the form  $y_1^{b_j}$  (single monomial), for some  $b_j \geq 0$ . Similarly, each entry of  $D_n$  is also homogeneous, when considering the partition  $D = D_{\leq n-1} \cdot D_n$ . For  $1 < i < n$ , consider  $D = D_{<i} \cdot D_i \cdot D_{>i}$ .

$$D = \begin{bmatrix} f_1 & f_2 & \cdots & f_r \end{bmatrix} \begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1r} \\ g_{21} & g_{22} & \cdots & g_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ g_{r1} & g_{r2} & \cdots & g_{rr} \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_r \end{bmatrix} \quad (8.1)$$

Let entries of  $D_{<i}$  be  $f_1, \dots, f_r \in \mathbb{F}[\mathbf{y}_{<i}]$ . By syntactic homogeneity of  $D_{<i}$ , each  $f_k$  for  $k \in [r]$ , is homogeneous. Also, by Nisan's characterization  $f_1, \dots, f_r$  are  $\mathbb{F}$ -linearly independent. Note that each entry of  $D_{\leq i}$  is inner product of  $D_{<i}$  with appropriate column of  $r \times r$  matrix  $D_i$ . Without loss of generality, let us consider the inner product with the first column whose entries are  $g_{11}, g_{21}, \dots, g_{r1} \in \mathbb{F}[y_i]$ . By syntactic homogeneity of  $D_{\leq i}$ , we know that  $G := f_1 g_{11} + f_2 g_{21} + \dots + f_r g_{r1}$  is homogeneous. Then, by Lemma 8.2.3 again, for each  $k \in [r]$ ,  $g_{k1}$  is also a homogeneous polynomial in  $y_i$ . Similarly, for every other column in  $D_i$ . This shows that each matrix entry  $g_{ij}$  is homogeneous (& univariate) and hence it is a monomial.  $\square$

## 8.2.2 PIT for single ROABP

As a simple corollary of the structure theorem, we get the following sparsity bound for a homogeneous polynomial computable by a width  $r$  ROABP.



**Lemma 8.2.5** (Sparsity bound). *Let  $f(\mathbf{x}) \in \mathbb{F}[x_1, \dots, x_n]$  be a homogeneous polynomial such that it can be computed by an ROABP of width  $r$ . Then,  $\|f\| \leq r^n$ .*

*Proof.* Let  $D(\mathbf{x})$  be the width- $r$  ROABP computing  $f$  over the field  $\mathbb{F}$ . By Theorem 8.2.4, without loss of generality, we can assume  $D(\mathbf{x})$  to be syntactically homogeneous with width  $\leq r$ . Thus, each edge label in the ROABP  $D$  is a univariate monomial. In that case, each path from source to sink computes only a single monomial. The number of paths from source to sink is at most  $r^n$ , as there are  $n$  layers and for each layer we have  $r$  choices. Hence the polynomial computes a sum of at most  $r^n$  monomials.  $\square$

We extend the above methods to prove another important property below: if a polynomial has ROABP width- $r$ , then so does its lead homogeneous part. For our work, we only require proof for the highest degree homogeneous component, which we state below but the same proof works for the lowest degree homogeneous component as well. Let  $\text{width}(f)$  denote the *minimum* width in which  $f$  can be computed by an ROABP.

**Lemma 8.2.6** (Homogeneous-part width). *Let  $f(\mathbf{x}) = f^{[d]} + f^{<[d]}$  be a polynomial of degree- $d$ , in  $\mathbb{F}[x_1, \dots, x_n]$ , where  $f^{[d]}$  is the (lead) degree- $d$  homogeneous component of  $f$ , and  $f^{<[d]}$  is rest of the polynomial  $f$ . Then,  $\text{width}(f^{[d]}) \leq \text{width}(f)$ , in the same variable order.*

*Proof.* Let  $f^{[d]}$  have an ROABP of  $\text{width}(f^{[d]}) =: k$  in unknown variable order  $(y_1, \dots, y_n)$ . For any fixed  $\ell \in [n]$ , consider the partition  $\{y_1, \dots, y_\ell\} \sqcup \{y_{\ell+1}, \dots, y_n\}$ . Without loss of generality, there are  $k$  coefficient polynomials of  $f^{[d]} - g_1, \dots, g_k \in \mathbb{F}[\mathbf{y}_{>\ell}]$  – that are  $\mathbb{F}$ -linearly independent. For some  $\mathbf{e}_1, \dots, \mathbf{e}_k \in \{0, 1, \dots, d\}^\ell$ , these are precisely  $g_i =: (f^{[d]})_{(\mathbf{y}_{\leq \ell}, \mathbf{e}_i)}$ , for each  $i \in [k]$ . We claim that the  $k$  coefficient-operators  $\mathbf{e}_1, \dots, \mathbf{e}_k \in \{0, 1, \dots, d\}^\ell$ , that worked for  $f^{[d]}$ , will also work for  $f$ .

Formally, the set of polynomials  $\{f_{(\mathbf{y}_{\leq \ell}, \mathbf{e}_1)}, \dots, f_{(\mathbf{y}_{\leq \ell}, \mathbf{e}_k)}\}$  will also be  $\mathbb{F}$ -linearly independent. This will mean that  $\dim_{\mathbb{F}}\{A_{(\mathbf{y}_\ell, \mathbf{a})} \mid \mathbf{a} \in \{0, 1, \dots, d\}^\ell\} \geq k$  and hence  $\text{width}(f) \geq k$ , by Lemma 6.4.2. We prove the linear independence now. For each  $i \in [k]$ ,

let  $g_i := (f^{[d]})_{(\mathbf{y}_{\leq \ell}, \mathbf{e}_i)}$  and  $h_i := (f^{[<d]})_{(\mathbf{y}_{\leq \ell}, \mathbf{e}_i)}$ . Then,

$$f_{(\mathbf{y}_{\leq \ell}, \mathbf{e}_i)} = (f^{[d]})_{(\mathbf{y}_{\leq \ell}, \mathbf{e}_i)} + (f^{[<d]})_{(\mathbf{y}_{\leq \ell}, \mathbf{e}_i)} =: g_i + h_i.$$

Here,  $\forall i \in [k]$ ,  $h_i$  is of degree strictly less than that of  $g_i$ . Observe that the coefficient-operators  $(f^{[d]})_{(\mathbf{y}_{\leq \ell}, \mathbf{e}_i)}$  respect homogeneity. Therefore,  $\forall i \in [k]$ ,  $g_i$  is a nonzero *homogeneous* polynomial of degree  $d_i := d - |\mathbf{e}_i|_1$ . Since  $g_1, \dots, g_k$  are  $\mathbb{F}$ -linearly independent, any  $\mathbb{F}$ -linear combination  $c_1 g_1 + c_2 g_2 + \dots + c_k g_k$  is nonzero, whenever  $c_i \in \mathbb{F}$  are not all zero. Now, we prove our claim that the polynomials  $g_1 + h_1, \dots, g_k + h_k$  are  $\mathbb{F}$ -linearly independent.

Suppose not, then there exist  $c_1, \dots, c_k \in \mathbb{F}$  not all zero such that

$$\begin{aligned} c_1(g_1 + h_1) + \dots + c_k(g_k + h_k) &= 0 \\ c_1 g_1 + \dots + c_k g_k &= -(c_1 h_1 + \dots + c_k h_k). \end{aligned} \tag{8.2}$$

Let  $d' := \max_i \{\deg(g_i) \mid c_i \neq 0\}$ . We show that the LHS in (8.2) is a nonzero polynomial of degree exactly  $d'$ . This is because  $g_i$  are homogeneous. So, if degree of LHS is  $< d'$ , then all the  $g_i$  of degree  $d'$  have to cancel among themselves. This cannot happen since they are linearly independent. Thus, LHS is of degree  $d'$  but RHS in (8.2) is a polynomial of degree  $< d'$ , since  $\deg(h_i) < \deg(g_i) \leq d'$ , for each  $i \in [k]$ . This contradicts (8.2), thus proving  $\{g_1 + h_1, \dots, g_k + h_k\}$  to be  $\mathbb{F}$ -linearly independent. We conclude that  $\text{width}(f) \geq k$ .  $\square$

We get a nice structural result as a consequence: the leading homogeneous part of a polynomial  $f \in \mathbb{F}[x_1, \dots, x_n]$  computed by an ROABP of width  $r$  is at most  $r^n$ -sparse.

**Corollary 8.2.7** (Leading sparsity). *Let  $f \in \mathbb{F}[x_1, \dots, x_n]$  be a degree- $d$  polynomial computed by an ROABP of width- $w$ . Let  $f^{[d]}$  be the leading homogeneous component of  $f$ . Then,  $\|f^{[d]}\| \leq w^n$ .*

*Proof.* Let  $\text{width}(f^{[d]}) := r$ . Then by Lemma 8.2.6,  $r \leq w$ . Since  $f^{[d]}$  is a homogeneous polynomial, Lemma 8.2.5 implies that  $\|f^{[d]}\| \leq r^n \leq w^n$ .  $\square$

This immediately gives us blackbox PIT for a log-variate constant-width ROABP (possibly inhomogeneous), as stated below.

**Lemma 8.2.8** (Single ROABP). *Let  $\mathcal{P}$  be a set of polynomials, over a field  $\mathbb{F}$ , computed by an ROABP of width- $r$  and degree- $d$  in  $n$  variables and unknown variable order. Then, we can design a hitting set generator  $\mathcal{G} : \mathbb{F} \rightarrow \mathbb{F}^n$  for  $\mathcal{P}$  with  $\deg(\mathcal{G}) = \text{poly}(d, r^n)$ . Thus, we get a blackbox PIT algorithm for  $\mathcal{P}$  in  $\text{poly}(d, r^n)$  time.*

*Proof.* Let  $f \in \mathcal{P}$  be of a non-zero polynomial of degree exactly  $d$ . Let  $f^{[d]}$  be the top homogeneous part of  $f$ , which will also be non-zero. By Corollary 8.2.7, we get  $\|f^{[d]}\| \leq r^n$ . Thus by Lemma 6.2.11, we get a single-seed HSG  $\mathcal{G} : \mathbb{F} \rightarrow \mathbb{F}^n$  for  $f$  such that  $f(\mathcal{G}) \neq 0$  and  $\deg(\mathcal{G}) = \text{poly}(d, r^n)$ . Thus,  $f(\mathcal{G})$  is a univariate polynomial with degree at most  $\text{poly}(d, r^n)$ . Evaluating  $f(\mathcal{G})$  on  $\deg(f(\mathcal{G})) + 1$ -many points yields a  $\text{poly}(d, r^n)$  time blackbox PIT.  $\square$

### 8.2.3 PIT for Degree-Preserving Sum

In a similar fashion as above, we also get a blackbox PIT for a degree-preserving sum of  $c$  ROABPs. Recall the definition of degree-preserving sum in Definition 8.2.1.

**Theorem 8.2.9** (Degree-preserving sum). *Let  $\mathcal{P}$  be a set of  $n$ -variate polynomials computed by a degree-preserving sum of  $c$  ROABPs, each of width  $r$  and degree  $d$ . Then, blackbox PIT for  $\mathcal{P}$  can be solved in  $\text{poly}(d, c \cdot r^n)$  time.*

*Proof.* Let  $f(\mathbf{x})$  be a degree  $d$  polynomial computed by a degree-preserving sum,  $f(\mathbf{x}) = \sum_{i=1}^c f_i(\mathbf{x})$ , where for each  $i \in [c]$ ,  $f_i(\mathbf{x})$  is computed by a width  $r$  ROABP. For each  $i \in [c]$ , let  $d_i := \deg(f_i)$ . By Corollary 8.2.7, the top degree homogeneous part of each  $f_i$ ,  $f_i^{[d_i]}$  has sparsity at most  $r^n$ . For a non-zero  $f$  of degree  $d$ , the (leading) degree- $d$  part of  $f$ ,  $f^{[d]} \neq 0$ . Since the sum is degree-preserving,  $d = \max_{i \in [c]} d_i$ . In other words, there exists a subset of indices  $S \subseteq [c]$  such that for all  $j \in S$ ,  $d_j = d$ . This yields the homogeneous sum:

$$f^{[d]} = \sum_{j \in S} f_j^{[d]}.$$

Since  $|S| \leq c$ ,  $f^{[d]}$  is non-zero with sparsity at most  $c \cdot r^n$ . Thus, we get a single-seed HSG  $\mathcal{G} : \mathbb{F} \rightarrow \mathbb{F}^n$  for  $f$  with  $\deg(\mathcal{G}) = \text{poly}(d, c \cdot r^n)$ , using Lemma 6.2.11. Thus,  $f(\mathcal{G})$  is a univariate polynomial of degree  $\text{poly}(d, c \cdot r^n)$ , which implies a blackbox PIT for  $f$  in  $\text{poly}(d, c \cdot r^n)$  time.  $\square$

Consider the class of polynomials which can be computed by a sum of  $c$  ROABPs, where each ROABP computes a homogeneous polynomial. Such a sum can be expressed as a degree-preserving sum. Thus, we get a blackbox PIT for this class in the same time. We prove it as follows. Let  $f$  be computed by a sum of ROABPs, where each ROABP computes a homogeneous polynomial. Suppose  $f(\mathbf{x}) = \sum_{i=1}^c f_i(\mathbf{x})$ , where for each  $i \in [c]$ ,  $f_i(\mathbf{x})$  is a homogeneous polynomial, say of degree  $d_i$ , computed by an ROABP of width  $r$ . Let degree of  $f$  be  $d$ . Let us consider the subset of indices  $S \subseteq [c]$ , defined as  $S = \{j \mid j \in [c], d_j \leq d\}$ . Since  $f$  is of degree  $d$ , observe that  $f = \sum_{j \in S} f_j$ , since homogeneous polynomials of degree  $> d$  must cancel out among themselves. Thus,  $f$  is computed by a degree-preserving sum  $\sum_{j \in S} f_j$ .

### 8.3 PIT for Sum of ROABPs

In this section, we discuss the general sum of ROABPs model, where the sum may not be degree-preserving. Note that we cannot apply the same idea now because the leading-degree components may cancel out in the sum and we do not have any analog of Lemma 8.2.6 for intermediate degree homogeneous parts. Therefore, a different idea is needed. For ease of exposition, we first explain the main idea of our PIT algorithm for sum of two ROABPs. We then generalize it to sum of  $c$  ROABPs recursively. Even though sum of ROABPs model is provably harder than a single ROABP (Fact 8.1.1), yet we show a reduction from PIT of sum of ROABPs to PIT of a single ROABP in Theorem 8.3.7. As an application of our reduction, we get new PIT algorithms in Corollary 8.3.8 and Corollary 8.3.9, which are efficient in the log-variate setting.

We start by showing below that any hitting-set map for a prefix of variables, also preserves the coefficient space dimension up to all subsequent variables. This will help us

work directly with hitting sets of single ROABPs in our PIT and it will be used critically in proof of Claim 8.3.2 later. Recall the definition of prefix and suffix maps in Definition 6.4.6.

**Lemma 8.3.1** (Dim. preservation). *Let  $D(\mathbf{y})$  be a matrix-product polynomial:*

$$D(\mathbf{y}) = D_1(y_1) \cdots D_{k-1}(y_{k-1}) \cdot D_k(y_k) \cdot D'(\mathbf{y}_{>k}),$$

where  $D_1 \in \mathbb{F}^{1 \times r}[y_1]$ ,  $D_i \in \mathbb{F}^{r \times r}[y_i]$  for  $2 \leq i < k$ ,  $D_k \in \mathbb{F}^{r \times r'}[y_k]$  with  $r \leq r'$  and  $D' \in \mathbb{F}^{r' \times 1}[\mathbf{y}_{>k}]$ . Suppose  $\mathcal{G} : \mathbb{F} \rightarrow \mathbb{F}^{k-1}$  is a generator for any  $r$ -width,  $(k-1)$ -variate ROABP with seed variable  $t$ . Further, let  $\Psi$  be the prefix map with respect to  $\mathcal{G}$ . Then,  $\Psi$  preserves the  $k$ -prefix coefficient space dimension of  $D$ , i.e.

$$\dim_{\mathbb{F}}\{D_{(\mathbf{y}_{\leq k}, \mathbf{a})} \mid \mathbf{a} \in \{0, 1, \dots, d\}^k\} = \dim_{\mathbb{F}}\{D(\Psi)_{((t, y_k), \mathbf{a})} \mid \mathbf{a} \in \{0, 1, \dots, d\}^2\}.$$

*Proof.* Consider the matrix product for  $D(\mathbf{y})$  at  $(k-1)^{th}$  layer:  $D = D_{<k} \cdot D_k \cdot D'$ , where  $D_{<k} := \prod_{i=1}^{k-1} D_i \in \mathbb{F}^{1 \times r}[\mathbf{y}_{<k}]$ . Without loss of generality, let the entries of  $D_{<k} \cdot D_k$  be the  $r'$   $\mathbb{F}$ -linearly independent polynomials given by Nisan's characterization (Lemma 6.4.2). Similarly entries of  $D'$  are  $r'$  linearly independent polynomials given by coefficient-extraction of  $D$ :

$$\begin{aligned} D_{<k} \cdot D_k &=: [P_1, P_2, \dots, P_{r'}], \\ D' &=: [Q_1, Q_2, \dots, Q_{r'}]^{\top}. \end{aligned}$$

View  $\Psi$  as mapping the first  $k$  variables to  $\mathbb{F}[t, y_k]$  (keeping the rest  $n-k$  variables unchanged). For  $c_i \in \mathbb{F}$  ( $i \in [r']$ ) not all zero, we have:

$$c_1 P_1 + c_2 P_2 + \dots + c_{r'} P_{r'} \neq 0. \quad (8.3)$$

Note that the polynomial  $c_1 P_1 + c_2 P_2 + \dots + c_{r'} P_{r'} =: P$  has width  $r'$  by Lemma 6.4.7.

Suppose the given generator  $\mathcal{G} = (\mathcal{G}_1, \dots, \mathcal{G}_{k-1})$ . Define  $\mathcal{G}' := (\mathcal{G}_1, \dots, \mathcal{G}_{k-1}, y_k)$ . Since  $\mathcal{G}$  is an HSG (with seed  $t$ ) for any width- $r$   $(k-1)$ -variate ROABP,  $\mathcal{G}'$  is an HSG (with seeds  $t, y_k$ ) for any width  $\leq r'$   $k$ -variate ROABP (having a width  $r$  ROABP in first  $k-1$  layers) using Lemma 6.4.8. Since  $P$  is indeed of width  $r'$  (&  $k$ -variate), therefore  $\mathcal{G}'$  preserves

the non-zeroneess of (8.3), implying  $\mathbb{F}$ -linear independence of  $P_1(\mathcal{G}'), \dots, P_{r'}(\mathcal{G}') \in \mathbb{F}[t, y_k]$ . Moreover,

$$D(\Psi) = [P_1(\mathcal{G}'), \dots, P_{r'}(\mathcal{G}')] \cdot [Q_1, Q_2, \dots, Q_{r'}]^\top.$$

Hence,  $\dim_{\mathbb{F}}\{D(\Psi)_{((t, y_k), \mathbf{a})} \mid \mathbf{a} \in \{0, 1, \dots, d\}^2\} = \dim_{\mathbb{F}}\{D_{(\mathbf{y}_{\leq k}, \mathbf{a})} \mid \mathbf{a} \in \{0, 1, \dots, d\}^k\} = r'$ , since otherwise if the dimension reduces on applying  $\Psi$ , we get a new non-trivial dependency among  $\{P_i(\mathcal{G}')\}_{i=1}^{r'}$ , say  $c_1 P_1(\mathcal{G}') + \dots + c_{r'} P_{r'}(\mathcal{G}') = 0$ . But these coefficients will not form a dependency among  $\{P_i\}_{i=1}^{r'}$ , i.e.  $c_1 P_1 + \dots + c_{r'} P_{r'} \neq 0$  and since  $\mathcal{G}'$  preserves non-zeroneess of (8.3), it leads to a contradiction.  $\square$

### 8.3.1 Sum of two ROABPs

We start with the sum of two ROABPs  $A + B$ . The blackbox PIT developed here would be extended to sum of  $c$  ROABPs in Section 8.3.2. Testing  $A + B = 0$  is same as testing equivalence of  $A$  and  $B$ . Let  $A, B \in \mathbb{F}[\mathbf{x}]$  be polynomials of individual degree  $d$ , computed by width- $r$  ROABPs, each of size  $s$  in  $n$  variables. Suppose  $A$  is computed in some unknown variable order  $(y_1, y_2, \dots, y_n)$ , where for all  $i \in [n]$ ,  $y_i = x_{\pi(i)}$  for some unknown permutation  $\pi : [n] \rightarrow [n]$ . We can assume that variable order of  $B$  is different from  $A$ , since otherwise by Lemma 6.4.4,  $A + B$  can be computed by a single ROABP of width  $\leq 2r$ . In that case, since we will be applying a hitting set map for a single ROABP of width  $O(r^3)$  in Lemma 8.3.4, we are already done. The main idea in [GKST17] is to construct an ROABP for  $B$  in the variable order of  $A$ , by using the characterizing dependencies of  $A$  (Definition 6.4.1). Note that the width of an ROABP can blow up exponentially when expressed in a different variable order (see [For14]). We can assume that  $B$  does not have ROABP of width  $r$  in the variable order of  $A$  since otherwise, we will again get a single ROABP of width  $2r$  computing  $A + B$  in which case, we are done.

Thus we are in the setting:  $A \neq -B$ , and  $B$  does not have a width  $r$  ROABP in the

order  $(y_1, y_2, \dots, y_n)$ . By Lemma 6.4.2, there will be a minimum index  $k \in [n]$  such that

$$\dim_{\mathbb{F}}\{A_{(\mathbf{y}_{\leq k}, \mathbf{a})} \mid \mathbf{a} \in \{0, 1, \dots, d\}^k\} \leq r, \text{ and}$$

$$\dim_{\mathbb{F}}\{B_{(\mathbf{y}_{\leq k}, \mathbf{a})} \mid \mathbf{a} \in \{0, 1, \dots, d\}^k\} > r$$

In simple terms,  $k$  is the first layer in the variable order of  $A$ , where  $B$  does not have an ROABP of width  $\leq r$ . Let us consider the dependency equations at this  $k^{th}$  layer for both  $A$  and  $B$ . Observe that there exists an exponent  $\mathbf{b} \in \text{depend}_k(A)$  such that  $A$  will satisfy its dependency equation while  $B$  will violate its dependency equation for this exponent. This is because, if  $B$  satisfies dependency equations for all  $\mathbf{b} \in \text{depend}_k(A)$ , then  $B$  also has a width  $\leq r$  ROABP till layer  $k$ , by Lemma 6.4.2. This contradicts our choice of  $k$ . Thus, we have some  $\mathbf{b} \in \text{depend}_k(A)$  such that

$$A_{(\mathbf{u}, \mathbf{b})} =: \sum_{\mathbf{a} \in \text{span}_k(A)} \alpha_{\mathbf{b}, \mathbf{a}} \cdot A_{(\mathbf{u}, \mathbf{a})} \quad (8.4)$$

$$B_{(\mathbf{u}, \mathbf{b})} \neq \sum_{\mathbf{a} \in \text{span}_k(A)} \alpha_{\mathbf{b}, \mathbf{a}} \cdot B_{(\mathbf{u}, \mathbf{a})} \quad (8.5)$$

where  $\mathbf{u} := \mathbf{y}_{\leq k} = (y_1, y_2, \dots, y_k)$ , and  $\alpha_{\mathbf{b}, \mathbf{a}} \in \mathbb{F}$  are the dependency coefficients defined by (8.4). Note that  $B$  may violate the dependency equations of  $A$  before layer  $k$ , while having a width  $\leq r$  representation. Unlike [GKST17], in our proof, we are ignoring such a layer and care only about the layer, where we witness blow-up of width in ROABP of  $B$ . Such a layer will exist under our assumption that  $B$  does not have a width  $\leq r$  ROABP in the variable order of  $A$ .

In the whitebox setting, one can essentially search for this violation/non-zeroness certificate and verify the satisfiability of dependency equations in poly-time. But in the blackbox setting, the unknown variable order creates a hurdle in searching for this certificate. Guessing the variable order by brute force takes  $n! \approx n^n$  time. We will show later that for the purpose of PIT, we can get around this obstacle in  $2^n$  time. We are okay with  $2^n$  overhead in the log-variate setting.

For a polynomial  $f \in \mathbb{F}[\mathbf{y}]$ , for our convenience, let us use a short-hand  $f_{(y_1^{a_1} y_2^{a_2})} \in \mathbb{F}[y_3, \dots, y_n]$  to denote the coefficient polynomial of monomial  $y_1^{a_1} y_2^{a_2}$  in  $f$ , which is same

as  $f_{((y_1, y_2), (a_1, a_2))}$  in the earlier notation. Let  $\mathcal{G} : \mathbb{F} \rightarrow \mathbb{F}^{k-1}$  be an HSG for  $(k-1)$ -variate ROABPs of width  $r$ , with a single seed variable, say  $t_1$ . Let  $\Phi_1$  be the prefix map wrt  $\mathcal{G}$  such that  $f(\Phi_1) \in \mathbb{F}[t_1, y_k, \dots, y_n]$ . We now show in Claim 8.3.2 below that  $B$  continues to violate the dependency equation (8.5) of  $A$  at the  $y_k$ -layer, under the image of map  $\Phi_1$ . We get (8.6) and (8.7) below, analogous to (8.4) and (8.5), respectively.

**Claim 8.3.2** (Prefix map). *Suppose  $\mathcal{G}^1 : \mathbb{F} \rightarrow \mathbb{F}^{k-1}$  is a generator for any  $(k-1)$ -variate ROABP of width  $\leq r$ , with single seed variable  $t_1$ . Let  $\Phi_1$  be the prefix map with respect to  $\mathcal{G}^1$ . Let  $d' := \max\{\deg_{t_1}(A(\Phi_1)), \deg_{t_1}(B(\Phi_1))\}$  and  $E := \{0, 1, \dots, d'\}$ . Let  $\text{span}_2(A(\Phi_1))$  be a basis of size  $\leq r$  such that there exists a set of constants  $\{\gamma_{\mathbf{b}, \mathbf{a}}\}$ , with  $\mathbb{F}$ -dependencies for every two-tuple  $\mathbf{b} \in E^2$ , defined as*

$$A(\Phi_1)_{(t_1^{b_1} y_k^{b_2})} =: \sum_{\mathbf{a} \in \text{span}_2(A(\Phi_1))} \gamma_{\mathbf{b}, \mathbf{a}} \cdot A(\Phi_1)_{(t_1^{a_1} y_k^{a_2})}. \quad (8.6)$$

Then, there exists  $\mathbf{b} \in E^2$  with a dependency violation in  $B(\Phi_1)$ , that is,

$$B(\Phi_1)_{(t_1^{b_1} y_k^{b_2})} \neq \sum_{\mathbf{a} \in \text{span}_2(A(\Phi_1))} \gamma_{\mathbf{b}, \mathbf{a}} \cdot B(\Phi_1)_{(t_1^{a_1} y_k^{a_2})}. \quad (8.7)$$

*Proof.* For the sake of contradiction, suppose  $B(\Phi_1)$  follows the dependency equations of  $A(\Phi_1)$  in the  $y_k$  layer. This means  $\forall \mathbf{b}$ , LHS equals RHS in (8.7). Since  $|\text{span}_2(A(\Phi_1))| \leq r$ , this means  $B(\Phi_1)$  has a width  $r$  ROABP in the first two layers ( $t_1$  and  $y_k$ ). In other words, coefficient space dimension for first two layers of  $B(\Phi_1)$  is  $\leq r$ . Observe that then by Lemma 8.3.1,  $\Phi_1$  preserves the coefficient space dimension of the first  $k$  layers of  $B$  too. Thus,

$$\dim_{\mathbb{F}}\{B(\mathbf{y}_{\leq k}, \mathbf{a}) \mid \mathbf{a} \in E^k\} = \dim_{\mathbb{F}}\{B(\Phi_1)_{((t_1, y_k), \mathbf{a})} \mid \mathbf{a} \in E^2\} \leq r.$$

This contradicts our choice of  $k$  being the first variable up to which  $B$  does not have a width  $r$  representation, which meant  $\dim_{\mathbb{F}}\{B(\mathbf{y}_{\leq k}, \mathbf{a}) \mid \mathbf{a} \in E^k\} > r$  (as in Lemma 6.4.2).  $\square$

In the following claim, we pick an HSG  $\mathcal{G}^2 : \mathbb{F} \rightarrow \mathbb{F}^{n-k}$  for  $(n-k)$ -variate ROABPs of appropriate width, with a single seed variable, say  $t_2$ . We consider the suffix map  $\Phi_2$  wrt  $\mathcal{G}^2$  and show that the image of  $B(\Phi_1)$  under  $\Phi_2$  continues to violate a dependency equation of the image of  $A(\Phi_1)$  under  $\Phi_2$ . For a polynomial  $f$ , we use the notation  $f \circ \Phi_2$



below to denote  $f(\Phi_2)$ . Note that  $\Phi_2$  maps the last  $(n - k)$  variables of  $f$  using  $\mathcal{G}^2$  and leaves the remaining variables as it is. In the setup of Claim 8.3.2,  $A(\Phi_1)$  is a polynomial in  $\mathbb{F}[t_1, y_k, \dots, y_n]$ . Then observe that  $(A(\Phi_1))(\Phi_2) =: A(\Phi_1) \circ \Phi_2$  is a polynomial in  $\mathbb{F}[t_1, y_k, t_2]$ .

**Claim 8.3.3** (Suffix map). *Assume the setup of Claim 8.3.2. Suppose  $\mathcal{G}^2 : \mathbb{F} \rightarrow \mathbb{F}^{n-k}$  is a generator for any  $(n - k)$ -variate ROABP of width  $\leq r^2(r + 1)$ , with single seed variable  $t_2$ . Let  $\Phi_2$  be the suffix map with respect to  $\mathcal{G}^2$ . Then, there exists  $\mathbf{b} \in E^2$  such that:*

$$(A(\Phi_1) \circ \Phi_2)_{(t_1^{b_1} y_k^{b_2})} = \sum_{\mathbf{a} \in \text{span}_2(A(\Phi_1))} \gamma_{\mathbf{b}, \mathbf{a}} \cdot (A(\Phi_1) \circ \Phi_2)_{(t_1^{a_1} y_k^{a_2})} \quad (8.8)$$

$$(B(\Phi_1) \circ \Phi_2)_{(t_1^{b_1} y_k^{b_2})} \neq \sum_{\mathbf{a} \in \text{span}_2(A(\Phi_1))} \gamma_{\mathbf{b}, \mathbf{a}} \cdot (B(\Phi_1) \circ \Phi_2)_{(t_1^{a_1} y_k^{a_2})}. \quad (8.9)$$

*Proof.* (8.8) in this claim directly follows by applying map  $\Phi_2$  on (8.6) and it is true  $\forall \mathbf{b} \in E^2$ . Now we shall prove that in (8.7) the difference polynomial  $g$  defined as,

$$g := B(\Phi_1)_{(t_1^{b_1} y_k^{b_2})} - \sum_{\mathbf{a} \in \text{span}_2(A(\Phi_1))} \gamma_{\mathbf{b}, \mathbf{a}} \cdot B(\Phi_1)_{(t_1^{a_1} y_k^{a_2})} \neq 0$$

can be computed using a single ROABP of width at most  $r^2(r + 1)$ .

Let  $(x_{\sigma(1)}, \dots, x_{\sigma(n)})$  be the original variable order of  $B$ , for some permutation  $\sigma$ . By assumption,  $B$  had a width  $r$  representation for the first  $(k - 1)$  layers in the variable order of  $A(y_1, \dots, y_n)$ , implying

$$B = [P_1, P_2, \dots, P_r] \cdot [Q_1, Q_2, \dots, Q_r]^\top,$$

where  $\forall i \in [r]$ ,  $P_i \in \mathbb{F}[\mathbf{y}_{<k}]$  and  $Q_i \in \mathbb{F}[\mathbf{y}_{\geq k}]$ . Recall that, by construction (Lemma 6.4.2),  $Q_i$ 's are certain coefficient polynomials of  $B$ ,  $Q_i = B_{(\mathbf{y}_{<k}, \mathbf{a}_i)}$  where  $\{\mathbf{a}_1, \dots, \mathbf{a}_r\} = \text{span}_{k-1}(A)$ . Now, by Lemma 6.4.3, each  $Q_i$  has a width  $r$  ROABP in the variable order inherited from  $B$ , that is,  $\sigma(\mathbf{y}_{\geq k})$ . Clearly, for each  $a \in E$ ,

$$B(\Phi_1)_{(t_1, a)} = \sum_{i=1}^r \text{coeff}(\Phi_1(P_i))(t_1^a) \cdot Q_i,$$

implying  $B(\Phi_1)_{(t_1, a)} \in \text{span}_{\mathbb{F}}\{Q_1, \dots, Q_r\}$ . For each  $a \in E$ , let  $B(\Phi_1)_{(t_1, a)} =: Q'_a$ , where  $Q'_a$  is the suitable  $\mathbb{F}$ -linear combination of  $Q_1, \dots, Q_r$ . Observe that by Lemma 6.4.4, any

$\mathbb{F}$ -linear combination  $\sum_{i=1}^r c_i Q_i$ , where each  $c_i \in \mathbb{F}$ , can be computed by a single ROABP of width  $r^2$  by placing ROABP of each  $Q_i$  in parallel. Thus, for each  $a \in E$ ,  $Q'_a$  also has an ROABP of width  $r^2$ .

Moving one variable forward, again by applying Lemma 6.4.3 on each  $Q'_a$ , we know that for each  $b \in E$ ,  $Q'_{a(y_k, b)} := (Q'_a)_{(y_k, b)}$  also has an ROABP of width  $r^2$  in the variable order  $\sigma(\mathbf{y}_{>k})$ . We can rewrite our polynomial  $g$  as

$$g = Q'_{b_1(y_k, b_2)} - \sum_{\mathbf{a} \in \text{span}_2(A(\Phi_1))} \gamma_{\mathbf{b}, \mathbf{a}} \cdot Q'_{a_1(y_k, a_2)}.$$

The number of summands in  $g$  is  $|\text{span}_2(A(\Phi_1))| + 1 \leq r + 1$ , and each summand in  $g$  has a width  $r^2$  ROABP. Hence, again by Lemma 6.4.4,  $g$  can be computed by a single ROABP of width  $\leq r^2(r + 1)$  by placing each of the width  $r^2$  ROABPs in parallel.

By premise  $\mathcal{G}^2$  is an HSG for any  $(n - k)$ -variate ROABP of width  $\leq r^2(r + 1)$  and  $\Phi_2$  is the suffix map wrt  $\mathcal{G}^2$ . Moreover,  $g$  by its definition is an  $(n - k)$ -variate polynomial and therefore  $g(\Phi_2) = g(\mathcal{G}^2) \neq 0$ . This yields the inequality in (8.9).  $\square$

Suppose we are given a correct guess of the variable order of  $A$ , say  $(y_1, y_2, \dots, y_n)$ . Now, let us combine both the prefix map  $\Phi_1$  and suffix map  $\Phi_2$  together, using their underlying generators  $\mathcal{G}^1, \mathcal{G}^2$ , respectively. Let  $\mathcal{G}^1 = (\mathcal{G}_1^1, \dots, \mathcal{G}_{k-1}^1)$  and  $\mathcal{G}^2 = (\mathcal{G}_1^2, \dots, \mathcal{G}_{n-k}^2)$ . Define  $\Phi : \mathbb{F}^3 \rightarrow \mathbb{F}^n = (\mathcal{G}^1, y_k, \mathcal{G}^2) := (\mathcal{G}_1^1, \dots, \mathcal{G}_{k-1}^1, y_k, \mathcal{G}_1^2, \dots, \mathcal{G}_{n-k}^2)$ . Note that  $A(\Phi_1) \circ \Phi_2 = A(\Phi)$ . Then, using Claim 8.3.2 and Claim 8.3.3, we show that  $\Phi$  is a generator for  $A + B$ , because the dependency violation by  $B$  is preserved under the image of  $\Phi$ . Moreover,  $(A + B)(\Phi)$  is a trivariate polynomial which is easy to test for non-zerosness.

Now we handle the case where variable order of  $A$  is unknown. Instead of going through all  $n!$  permutations, we only go over all  $k$ -sized subsets of  $[n]$ . This is because we are applying an HSG  $\mathcal{G}^1$  of a single ROABP (of unknown variable order) on the prefix variables and hence the order within the prefix-subset does not matter. For each choice of subset, we go over additional  $n$  choices by trying each variable as  $y_k$ . This process will lead to constructing a set  $G$  of HSGs such that for an input polynomial  $A + B \neq 0$ , there is an HSG  $\Phi \in G$  such that  $A + B(\Phi) \neq 0$ . Moreover, our set  $G$  has size  $\leq n \cdot 2^n$  which

is  $\text{poly}(r, d)$  in the log-variate setting. Then, using Lemma 6.2.7 we can combine all the generators into a single generator. We formalize this construction in the lemma below.

**Lemma 8.3.4** (Sum of two). *Let  $A(\mathbf{x})$  and  $B(\mathbf{x})$  be two polynomials of individual degree  $d$ , each computed by an ROABP of width  $r$ . Let  $\Gamma : \mathbb{F} \rightarrow \mathbb{F}^n$  be a hitting set generator for the class of width  $r$ ,  $n$ -variate,  $d$ -degree ROABPs with degree of HSG  $\Gamma$ ,  $\deg(\Gamma) =: T(r, n, d)$ . Then, one can design a hitting set generator  $\Psi : \mathbb{F} \rightarrow \mathbb{F}^n$  for the sum  $A + B$  in  $(n2^n \cdot T(2r^3, n, d))^{O(1)}$  time such that  $\deg(\Psi) = (n2^n \cdot T(2r^3, n, d))^{O(1)}$ .*

*Proof.* We start with a non-zero input polynomial  $A + B$ . Suppose we know the correct variable order of  $A$ :  $(y_1, \dots, y_n)$  and the correct layer where  $B$  violates dependency equation of  $A$ , say the  $y_k$  layer. Invoke HSG  $\Gamma$  with appropriate parameters to get HSG  $\mathcal{G}^1 : \mathbb{F} \rightarrow \mathbb{F}^{k-1}$  for  $(k-1)$ -variate, width  $\leq r$  ROABPs and HSG  $\mathcal{G}^2 : \mathbb{F} \rightarrow \mathbb{F}^{n-k}$  for  $(n-k)$ -variate, width  $\leq r^2(r+1)$  ROABPs as used in Claim 8.3.2 and Claim 8.3.3 respectively. Let  $\Phi : \mathbb{F}^3 \rightarrow \mathbb{F}^n := (\mathcal{G}^1, y_k, \mathcal{G}^2)$  be the concatenation of the two HSGs. We now show that if we guessed the correct order and layer, then  $\Phi$  is an HSG for  $A + B$ , that is,  $A + B(\Phi) \neq 0$ .

Claim 8.3.2 and Claim 8.3.3 together prove that  $B(\Phi)$  violates a dependency equation of  $A(\Phi)$  in the  $y_k$  layer. In other words  $A(\Phi)$  is an ROABP of width  $r$ , in the variable order  $(t_1, y_k, t_2)$ , where  $t_1, t_2$  are the seed variables of  $\mathcal{G}^1, \mathcal{G}^2$  respectively. At the same time, (8.9) points out that  $B(\Phi)$  does not have width  $r$  ROABP in the same variable order. Thus,  $(A + B)(\Phi) = A(\Phi) + B(\Phi) \neq 0$ .

Let us now work with unknown variable order. For Claim 8.3.2 to hold, we only need to ‘guess’ the prefix set  $\{y_1, \dots, y_k\}$  and the variable  $y_k$ . For each  $k \in [n]$ , we go over all  $k$ -sized subsets of  $[n]$ , and try  $k$  choices of  $y_k$  for each subset. The number of possibilities is at most  $\sum_{k=1}^n k \binom{n}{k} = \sum_{k=1}^n n \binom{n-1}{k-1} \leq n2^{n-1}$ . We try  $\Phi$  for each guess. For the  $\Phi$  corresponding to correct guess of prefix and variable  $y_k$ , the above argument guarantees that  $\Phi$  is an HSG for  $A + B$ . Thus, we get a collection  $G$  of candidate HSGs, one for each guess, with  $|G| \leq n2^n$  such that one of them is guaranteed to work. Using Lemma 6.2.7, we get a single generator  $\Psi' : \mathbb{F}^4 \rightarrow \mathbb{F}^n$ , which we can redefine to a single-seed generator  $\Psi : \mathbb{F} \rightarrow \mathbb{F}^n$  for  $A + B$  using Lemma 6.2.9.

We now calculate the degree of HSG  $\Psi$  in terms of degree of the HSG  $\Gamma$  for a single ROABP. Observe that  $\deg(\Phi_1) = T(r, k-1, d)$  and  $\deg(\Phi_2) = T(r^2(r+1), n-k, d)$ . Thus,

$$\deg(\Phi) = \max\{\deg(\Phi_1), \deg(\Phi_2)\} \leq T(2r^3, n, d).$$

Now, degree for the single generator  $\Psi'$  using Lemma 6.2.7 is

$$\deg(\Psi') = \max\{\deg(\Phi), |G|\} = \max\{T(2r^3, n, d), n2^n\}.$$

Thus, by Lemma 6.2.9, degree of the single-seed generator  $\Psi$  is

$$\deg(\Psi) \leq (n2^n \cdot T(2r^3, n, d))^{O(1)}.$$

The time complexity for designing  $\Psi$  is also similarly bounded.  $\square$

Using the efficient HSG designed for a single constant-width, log-variate ROABP in Lemma 8.2.8, we get an efficient HSG for sum of two such ROABPs below.

**Corollary 8.3.5.** *Let  $\mathcal{P}$  be a set of  $n$ -variate polynomials over a field  $\mathbb{F}$ , computed by a sum of two ROABPs, each of width  $r$  and degree  $d$  in unknown variable order. Then, blackbox PIT for  $\mathcal{P}$  can be solved in  $\text{poly}(d, r^n)$  time.*

*Proof.* We have HSG  $\Phi$  for single ROABP of width  $r$  with  $\deg(\Phi) = \text{poly}(d, r^n)$  using Lemma 8.2.8. Now, by Lemma 8.3.4, we have HSG  $\Psi$  for  $\mathcal{P}$  with  $\deg(\Psi) = \text{poly}(d, 2^n, r^{3n}) = \text{poly}(d, r^n)$ . Thus,  $\deg(f(\Psi))$  for some  $f \in \mathcal{P}$  is also  $\text{poly}(d, r^n)$ . By evaluating  $f(\Psi)$  on  $\deg+1$  points, we can check non-zerosness of  $f(\Psi)$ . This gives blackbox PIT for  $f$ .  $\square$

### 8.3.2 Sum of $c$ ROABPs

Let the input be sum of  $c$  polynomials  $A_1(\mathbf{x}), A_2(\mathbf{x}), \dots, A_c(\mathbf{x})$ , each of individual degree  $d$ , computed by ROABPs of width  $r$ . Again, we will assume the variable orders for each ROABP to be different, lest we reduce to a smaller sum instance. The simple recursive strategy used in [GKST17] is to reduce it to an instance of sum of two ROABPs. Let  $A := A_1$  and  $B := A_2 + \dots + A_c$ . Suppose  $A$  has  $r$ -width ROABP in some unknown

variable order  $(y_1, y_2, \dots, y_n)$ . Thus, we get dependency equations for  $A$ , as in (8.4). If the input sum is non-zero, then  $B$  will not follow some dependency of  $A$ .

Note that unlike the sum of two ROABPs case,  $B$  is not computed by a single ROABP of width  $r$ . This is not a cause of worry, as we shall see now. Define  $Q := B_{(\mathbf{u}, \mathbf{b})} - \sum_{\mathbf{a} \in \text{span}_k(A)} \gamma_{\mathbf{b}, \mathbf{a}} \cdot B_{(\mathbf{u}, \mathbf{a})}$ , where  $\mathbf{u} := (y_1, \dots, y_k)$ . Since  $B = A_2 + \dots + A_c$ , we get

$$Q = \sum_{i=2}^c \left( A_{i(\mathbf{u}, \mathbf{b})} - \sum_{\mathbf{a} \in \text{span}_k(A)} \gamma_{\mathbf{b}, \mathbf{a}} \cdot A_{i(\mathbf{u}, \mathbf{a})} \right). \quad (8.10)$$

Now each  $A_{i(\mathbf{u}, \mathbf{a})}$  and  $A_{i(\mathbf{u}, \mathbf{b})}$  have ROABPs of width  $r$  individually, by Lemma 6.4.3. Thus, for each of the  $c - 1$  summands in (8.10), we have an ROABP of width  $r(r + 1)$ , by Lemma 6.4.4. We apply prefix map/generator  $\Phi_1$  on first  $k - 1$  variables and get analogous dependency equations for  $A(\Phi_1)$ ,  $B(\Phi_1)$  (and  $Q(\Phi_1) \neq 0$ ) as in Claim 8.3.2.

$$Q(\Phi_1)_{(t_1^{b_1} y_k^{b_2})} = \sum_{i=2}^c \left( A_i(\Phi_1)_{(t_1^{b_1} y_k^{b_2})} - \sum_{\mathbf{a} \in \text{span}_2(A_i(\Phi_1))} \gamma_{\mathbf{b}, \mathbf{a}} \cdot A_i(\Phi_1)_{(t_1^{a_1} y_k^{a_2})} \right). \quad (8.11)$$

In (8.11), each of the  $c - 1$  summands has an ROABP of width  $r^2(r + 1) \leq 2r^3$  (See proof of Claim 8.3.3). This effectively reduces the problem, of designing suffix map/generator  $\Phi_2$ , to an instance of blackbox PIT for sum of  $c - 1$  ROABPs of width  $O(r^3)$ , which can be solved recursively. We formalize this process in the following lemma.

**Lemma 8.3.6** (Sum of  $c$ ). *Let  $A_1(\mathbf{x}), A_2(\mathbf{x}), \dots, A_c(\mathbf{x})$  be  $c$  polynomials of individual degree  $d$ , each computed by an ROABP of width  $r$ . Let  $\Gamma : \mathbb{F} \rightarrow \mathbb{F}^n$  be a hitting set generator for the class of width  $r$ ,  $n$ -variate,  $d$ -degree ROABPs with degree of HSG  $\Gamma$ ,  $\deg(\Gamma) =: T(r, n, d)$ . Then, one can design a hitting set generator  $\Psi : \mathbb{F} \rightarrow \mathbb{F}^n$  for the sum  $\sum_{i=1}^c A_i(\mathbf{x})$  in  $(2^n \cdot T(2^c r^{3^c}, n, d))^{O(c)}$  time, with  $\deg(\Psi) = (2^n \cdot T(2^c r^{3^c}, n, d))^{O(c)}$ .*

*Proof.* We prove by induction on  $c$ . Base case for  $c = 2$  has been proved earlier in Lemma 8.3.4. Suppose  $A = A_1$  has the unknown variable order  $(y_1, \dots, y_n)$  where  $y_i = x_{\pi(i)}$  for each  $i \in [n]$ . Let  $y_k$  be the first layer where  $B = A_2 + \dots + A_c$  deviates from  $A$ . Suppose we have correctly guessed the variable order and the variable  $y_k$ . Then, we employ HSG  $\mathcal{G}^1 : \mathbb{F} \rightarrow \mathbb{F}^{k-1}$  for first  $k - 1$  variables as used in Claim 8.3.2 and we get (8.6) for  $A$  while we get (8.11) for  $B = \sum_{i=2}^c A_i$ . Since  $B$  violates dependency equation of  $A$ ,

(8.11) is non-zero polynomial, which can be computed as a sum of  $c-1$  ROABPs of width  $\leq 2r^3$ . By induction hypothesis, we can design a (single-seed) HSG  $\mathcal{G}^2 : \mathbb{F} \rightarrow \mathbb{F}^{n-k}$  for  $Q(\Phi_1)_{(t_1^{b_1} y_k^{b_2})}$ , which acts on the remaining  $(n-k)$  variables, and preserves non-zerosness of the polynomial. Thus, altogether  $\Phi : \mathbb{F}^3 \rightarrow \mathbb{F}^n := (\mathcal{G}^1, y_k, \mathcal{G}^2)$  will preserve non-zerosness of  $A+B$ , since  $A(\Phi)$  will satisfy all its dependency equations but  $B(\Phi)$  would continue to violate one. This implies  $(A+B)(\Phi) \neq 0$  when  $A+B \neq 0$ .

Now, suppose the variable order is unknown. Then, observe that we only need to guess the prefix subset of variables and the correct variable  $y_k$  in it. We simply brute-force search for them. In other words, for each  $k$ -sized subset of  $[n]$  and for each variable as  $y_k$ , we apply  $\Phi$ . Thus, we get a set of candidate generators  $G$  of size  $|G| \leq n2^n$  such that for any non-zero  $A+B$ , there is some generator  $\Phi \in G$  for which  $(A+B)(\Phi) \neq 0$ . Using Lemma 6.2.7, we can combine the generators into a single generator  $\Psi' : \mathbb{F}^4 \rightarrow \mathbb{F}^n$ . Using Lemma 6.2.9, we get a single-seed generator  $\Psi : \mathbb{F} \rightarrow \mathbb{F}^n$  for  $\sum_{i=1}^c A_i$ .

Now, we calculate  $\deg(\Psi)$ . Observe that

$$\begin{aligned}
 \deg(\mathcal{G}^1) &= T(r, k-1, d) \leq T(r, n, d) \\
 \deg(\mathcal{G}^2) &= (2^{n-k} \cdot T(2^{c-1}r^{3^c}, n-k, d))^{O(c-1)} \\
 &\leq (2^n \cdot T(2^c r^{3^c}, n, d))^{O(c-1)} \\
 \deg(\Phi) &\leq (2^n \cdot T(2^c r^{3^c}, n, d))^{O(c-1)} \\
 \deg(\Psi') &\leq (2^n \cdot T(2^c r^{3^c}, n, d))^{O(c-1)} \\
 \deg(\Psi) &\leq (d \cdot n2^n)^{O(1)} \cdot (2^n \cdot T(2^c r^{3^c}, n, d))^{O(c-1)} \\
 &\leq (2^n \cdot T(2^c r^{3^c}, n, d))^{O(c)}
 \end{aligned}$$

The first step follows since  $\mathcal{G}^1$  is HSG for single ROABP. Degree of  $\mathcal{G}^2$  is given by induction hypothesis. Degree of  $\Phi$  is simply the maximum between  $\deg(\mathcal{G}^1)$  and  $\deg(\mathcal{G}^2)$ . Degree of  $\Psi'$  is calculated using Lemma 6.2.7. Finally, by Lemma 6.2.9, we get degree of  $\Psi$ .

Let  $S(c, r, n, d)$  denote the time complexity of constructing  $\Psi$ . Similar to the argument above, we get the following recursive formula for designing  $\Psi$ , where  $S(1, r, n, d) =$

$T(r, n, d)^{O(1)}$  and

$$S(c, r, n, d) \leq n2^n \cdot T(r, n, d) \cdot S(c-1, 2r^3, n, d).$$

As a solution, we get  $S(c, r, n, d) \leq (2^n \cdot T(2^c r^{3^c}, n, d))^{O(c)}$ .  $\square$

Finally after having developed all the machinery, we come to our main result of this chapter — showing a reduction from designing a blackbox PIT algorithm for sum of ROABPs to designing a blackbox PIT algorithm for a single ROABP.

**Theorem 8.3.7** (Reduction to one). *Let  $T(r, n, d)$  be the time complexity of a blackbox PIT algorithm for a single ROABP of width  $r$  and degree  $d$  in  $n$  variables over any field  $\mathbb{F}$ . Then, blackbox PIT for sum of  $c$ -many ROABPs, each of width  $r$  and degree  $d$  in  $n$  variables, can be solved in time  $T'(r, n, d, c) = (2^n \cdot T(2^c r^{3^c}, n, d))^{O(c)}$  over  $\mathbb{F}$ .*

*Proof.* The proof simply follows from Lemma 8.3.6 because of equivalence between hitting set generator and blackbox PIT. That is, if we have an HSG of degree  $T(r, n, d)$  which can also be constructed in the same time, then we have a  $\text{poly}(T(r, n, d))$  time blackbox PIT and vice versa. See Section 6.2 and [SY10, Lemma 4.1] for exact equivalence between HSG and blackbox PIT.  $\square$

**Blackbox** PIT for a single ROABP, over any field  $\mathbb{F}$ , has time complexity  $(ndr)^{O(\log n)}$ , which is only quasi-poly time. In the log-variate setting, a  $d^n = d^{O(\log(rd))}$  (quasi-poly) time algorithm for sum of ROABPs is already trivial via brute force derandomization based on the Polynomial Identity Lemma. Thus, to extract a poly-time PIT for sum using above theorem, we need poly-time blackbox PIT for single ROABP. We indeed get one, when width  $r$  is constant and  $n = O(\log d)$ , in Lemma 8.2.8. This then gives us the following corollary.

**Corollary 8.3.8** (Sum of ROABPs). *Let  $\mathcal{P}$  be a set of  $n$ -variate polynomials, computed by a sum of  $c$ -many ROABPs, each of width  $r$  and degree  $d$ . Then, blackbox PIT for  $\mathcal{P}$  can be solved in  $\text{poly}(d^c, r^{nc3^c})$  time.*

*Proof.* Let  $f \in \mathbb{F}[\mathbf{x}]$  be an  $n$ -variate polynomial computed by sum of  $c$  ROABPs, each of width  $r$  and degree  $d$ . Using Lemma 8.2.8, we have a blackbox PIT for a single ROABP of width  $r$  in  $\text{poly}(d, r^n)$  time. Now, in Theorem 8.3.7, set  $T(r, n, d) := \text{poly}(d, r^n)$  to get  $T'(r, n, d, c) = (2^n \cdot d \cdot r^{n3^c})^{O(c)} = \text{poly}(d^c, r^{nc3^c})$ . This gives us blackbox PIT algorithm for  $f$  with the required time complexity.  $\square$

If we use blackbox PIT algorithm of [AGKS15] for single ROABP in Theorem 8.3.7, we get another efficient PIT for the sum of ROABPs as a corollary below.

**Corollary 8.3.9** (Improved Sum PIT). *Let  $\mathcal{P}$  be a set of  $n$ -variate polynomials computed by a sum of  $c$ -many ROABPs, each of width  $r$  and degree  $d$ . Then, blackbox PIT for  $\mathcal{P}$  can be solved in  $\text{poly}(2^{cn} \cdot n^{c \log n}, d^{c \log n}, r^{3^c \log n})$  time.*

*Proof.* Using result of [AGKS15], we have a  $(ndr)^{O(\log n)}$  time blackbox PIT for a single ROABP of width  $r$  and degree  $d$  in  $n$  variables. Therefore, we can set  $T(r, n, d) := (ndr)^{O(\log n)}$  in Theorem 8.3.7 to get a  $\text{poly}(2^{cn} \cdot n^{c \log n}, d^{c \log n}, r^{3^c \log n})$  time blackbox PIT for border of sum of  $c$  ROABPs.  $\square$

**Remark.** In a subsequent work, [GG20] improved the blackbox PIT for a single ROABP. One can then also use that in conjunction with our Theorem 8.3.7 to get a further improved blackbox PIT for sum of ROABPs in the log-variate regime. This has been stated as Theorem 1.2 in [GG20] by citing an earlier version of Theorem 8.3.7 in this work.

## 8.4 PIT for Border

In this section, we give the reduction from PIT for border class of sum of  $c$  ROABPs to PIT for a single ROABP (Theorem 8.4.1). Although a single ROABP is closed under border, it is not clear if sum of constantly many ROABPs is equal to its border class. Let  $A$  and  $B$  be two ROABPs of width  $w$  in different variable orders. Let  $f$  be a polynomial computed in the border class of sum of two ROABPs. Then we can write  $g = f + \epsilon h$ , where  $g$  is computed by  $A + B$  over  $\mathbb{F}(\epsilon)$ . One might question whether  $f$  can be expressed as  $f_1 + f_2$ , where  $f_1$  is computed in the border of  $A$  and  $f_2$  in the border of  $B$ . If this



were true, then  $f$  could also be computed by sum of two ROABPs since both  $f_1$  and  $f_2$  individually would be computable by ROABPs of width  $w$  as stated in Lemma 6.5.1. But this line of thought is false for the following reason. Note that the polynomial  $g$  which approximates  $f$  is computed by sum of two ROABPs  $A + B$  over  $\mathbb{F}(\epsilon)$ . In the edge weights of  $A$ , there maybe coefficients involving  $\epsilon$  in denominator which get canceled only in the sum but stay individually, and therefore  $f$  is not directly expressible as  $f_1 + f_2$ , where both  $f_1$  and  $f_2$  are individually computable in the border class of single ROABP.

Moreover,  $g = A + B$  over  $\mathbb{F}(\epsilon)$  may not have a single ROABP of same width over  $\mathbb{F}(\epsilon)$ , since width can blow up exponentially, as stated in Fact 8.1.1. Thus, border of sum of ROABPs cannot be directly expressed as border of a single ROABP of similar width. Hence, in all likelihood, the border class of sum of  $c$  ROABPs is more powerful than the class of sum of  $c$  ROABPs. This makes it an interesting candidate for the PIT question. In Theorem 8.4.1, we solve it along the same lines as Theorem 8.3.7 by showing an efficient reduction from PIT of border of sum to PIT of border of a single ROABP (in log-variate regime). We are able to achieve this because the proof technique of Theorem 8.3.7 is compatible with border, essentially because at the core they rely on rank based measure of Nisan's width characterization.

**Theorem 8.4.1** (Reduction for border). *Let  $T(r, n, d)$  be the time complexity of a blackbox PIT algorithm for a single ROABP of width  $r$  in  $n$  variables and degree  $d$  over any field  $\mathbb{F}$ . Then blackbox PIT for border of sum of  $c$  ROABPs, each of width  $r$  and degree  $d$  in  $n$  variables, can be solved in time  $(2^n \cdot T(2^c r^{3^c}, n, d))^{O(c)}$  over  $\mathbb{F}$ .*

*Proof.* Let  $f(\mathbf{x})$  be a polynomial in the border of  $A_1 + A_2 + \dots + A_c$ . That is,  $g(\mathbf{x}, \epsilon) = f(\mathbf{x}) + \epsilon \cdot h(\mathbf{x}, \epsilon)$ , where  $g(\mathbf{x}, \epsilon)$  is computed by  $A_1 + \dots + A_c$  over  $\mathbb{F}(\epsilon)$  and  $\lim_{\epsilon \rightarrow 0} g = f$ . Our aim is to design an HSG  $\Psi$  for  $f(\mathbf{x})$  such that  $f \not\equiv 0 \Rightarrow f(\Psi) \not\equiv 0$ .

Let us first work over the function field  $\mathbb{F}(\epsilon)$ . We follow a similar inductive strategy as Lemma 8.3.6 for  $g$ . Let  $A = A_1$  and  $B := A_2 + \dots + A_c$ . Assume  $A$  has width  $r$  ROABP while  $f$  may not. Write  $f$  in the variable order of  $A$ . If  $f$  has an ROABP of width  $r$  in the variable order of  $A$ , then the HSG from Lemma 8.2.8 will work in the promised time, since border is closed for a single ROABP.

By Lemma 6.4.2, if  $f$  cannot be written as ROABP of width  $r$  in variable order of  $A$ , then there is a layer  $k$ , where  $f$  has width greater than  $r$  and hence  $f$  does not follow the dependency equations of  $A$  in that layer<sup>1</sup>. Thus, there exists  $k \in [n]$ , prefix  $\mathbf{u}$ ;  $S \subset \{0, 1, \dots, d\}^k$ ; and constants  $\{\alpha_{\mathbf{a}} \in \mathbb{F}[\epsilon] \mid \mathbf{a} \in S\}$  such that

$$0 = \sum_{\mathbf{a} \in S} \alpha_{\mathbf{a}} \cdot A_{(\mathbf{u}, \mathbf{a})} \quad (8.12)$$

$$0 \neq \sum_{\mathbf{a} \in S} \alpha_{\mathbf{a}} \cdot f_{(\mathbf{u}, \mathbf{a})}. \quad (8.13)$$

The equality (8.12) above is derived from (8.4) by collecting all the terms on one side and considering size of  $S$  to be at most  $r + 1$ . Similarly, (8.13) is derived by considering dependency equations for  $f$  at layer  $k$ . Observe that here, unlike Lemma 8.3.6, we are not working with  $B$  directly as in (8.5). This is because the inequality for  $B$  in (8.5) may become an equality in the limit  $\epsilon \rightarrow 0$ . Therefore, we work indirectly via  $f$  in (8.13) because we are assuming input polynomial  $f = \lim_{\epsilon \rightarrow 0} g$  to be non-zero and use that to derive a nontrivial relationship as shown below.

Recalling  $A + B = g = f + \epsilon \cdot h$ , we use (8.12) to deduce:

$$\sum_{\mathbf{a} \in S} \alpha_{\mathbf{a}} \cdot B_{(\mathbf{u}, \mathbf{a})} = \sum_{\mathbf{a} \in S} \alpha_{\mathbf{a}} \cdot f_{(\mathbf{u}, \mathbf{a})} + \epsilon \cdot h'$$

for the  $h' \in \mathbb{F}[\epsilon, \mathbf{x}]$  that depends on  $h$ . Without loss of generality, we can assume that  $\epsilon$  does not divide  $\alpha_{\mathbf{a}}$  for some  $\mathbf{a} \in S$ , because if it does divide all  $\alpha_{\mathbf{a}}$ , then we can divide the above equation by  $\epsilon$  to get a new equation of the same form. Moreover,  $\{f_{(\mathbf{u}, \mathbf{a})} \mid \mathbf{a} \in S\}$  are linearly independent polynomials over  $\mathbb{F}$  (equivalently over  $\mathbb{F}(\epsilon)$ ) and  $|S| \leq r + 1$  in the above equation.

Using  $\alpha \equiv \alpha(0) \bmod \epsilon$ , we write down a nontrivial relationship

$$\sum_{\mathbf{a} \in S} \alpha_{\mathbf{a}} \cdot B_{(\mathbf{u}, \mathbf{a})} = \sum_{\mathbf{a} \in S} \alpha_{\mathbf{a}}(0) \cdot f_{(\mathbf{u}, \mathbf{a})} + \epsilon \cdot h''. \quad (8.14)$$

It is nontrivial because its RHS does not vanish on setting  $\epsilon = 0$  and it remains well-defined, while LHS is a sum of  $(c - 1)$  ROABPs over  $\mathbb{F}(\epsilon)$ , each of width  $r(r + 1)$ . If

---

<sup>1</sup>If no such layer exists, then  $f$  is in the border of a single ROABP of width  $r$  and we are done by Lemma 6.5.1

our input  $f = \lim_{\epsilon \rightarrow 0} g$  is non-zero, then LHS above must also be non-zero in the limit  $\epsilon \rightarrow 0$ , by considering (8.14) in conjunction with (8.13). Thus, we reduce to the problem of designing HSG for border of sum of  $c - 1$  ROABPs.

Following the proof in Section 8.3, this reduces PIT of  $f$  to PIT of border of sum of  $c - 1$  ROABPs each of width  $O(r^3)$  (via the prefix and suffix maps). This can be solved recursively, till we reach border of a single ROABP of appropriate width. Since border of single ROABP is same as ROABP by Lemma 6.5.1, we can call HSG of Lemma 8.2.8 with appropriate parameters. The exact and formal details of unfolding the recursion are same as that in the proof of Lemma 8.3.6.  $\square$

**Remark.** Since the above theorem achieves the same time complexity as in Theorem 8.3.7, our results for PIT of sum of  $c$  ROABPs extend to their border versions also. Thus, we also get  $\text{poly}(d^c, r^{nc3^c})$  and  $\text{poly}(2^{cn} \cdot n^{c \log n}, d^{c \log n}, r^{3^c \log n})$ -time blackbox PIT algorithms for the border class of sum of  $c$  ROABPs analogous to Corollary 8.3.8 and Corollary 8.3.9, respectively. We will also get a new blackbox PIT for the border class with the time complexity achieved in Theorem 1.2 of [GG20].

## 8.5 Discussion

We first discuss the proof techniques we employed in this chapter, followed by open problems.

**Syntactic homogeneity for ROABP:** Inspired by circuits we defined *syntactic homogeneity* for ROABP (Definition 8.2.2). We proved that if a degree- $d$  homogeneous polynomial has an ROABP of width- $r$ , then it also has a syntactic homogeneous ROABP of the *same* width, and in the same variable order (Theorem 8.2.4). Note that if one applies the usual *homogenization* trick, for circuits/ABP [SY10, Thm.2.2], then the ROABP width blows up to  $O(rd^2)$ , making the width non-constant! Our new technique helps solve blackbox PIT for a constant-width log-variate ROABP, but also seems independently interesting. Moreover, Theorem 8.2.4 is independent of any restrictions on width or number

of variables but we make use of these restrictions only in deriving efficient PIT algorithms.

**Reduction from many to one:** In Theorem 8.3.7, we gave a reduction from designing a polynomial time blackbox PIT for sum of constantly many ROABPs to designing a polynomial time blackbox PIT for a single ROABP, in the log-variate setting. This reduction does not assume any restriction on width of ROABPs. This already gives us an improvement over [GKST17] in Corollary 8.3.9. We need constant width in Corollary 8.3.8 only because poly-time blackbox PIT for an unbounded-width (log-variate) ROABP is yet to be found.

**Remark.** In a recent subsequent work, [GG20, Thm. 1.1] constructed explicit hitting sets of polynomial size for a single log-variate ROABP of width upto  $2^{O(\log d / \log \log d)}$  over field  $\mathbb{F}$  with  $\text{char}(\mathbb{F}) = 0$  or  $\text{char}(\mathbb{F}) > d$ . In their second theorem, they directly call our Theorem 8.3.7 to extend their result to sum of constantly many such ROABPs.

**Comparison with [GKST17]:** Previously, the sum of constantly-many ROABPs was studied by [GKST17]. For this model, they give  $(ndr)^{O(\log(ndr))}$ -time blackbox PIT. We improve the time complexity for this model with respect to width  $r$  and degree  $d$  parameters by trading it with exponential dependence on number of variables. We get blackbox PIT with time complexity  $2^{O(n)} \cdot (ndr)^{O(\log n)}$  in Corollary 8.3.9. In the log-variate setting alone, this is an improvement over [GKST17]. If we allow worse dependence on both width and number of variables, then we show how to get down to polynomial dependence on  $d$  in Corollary 8.3.8 by giving a  $\text{poly}(d, r^n)$ -time blackbox PIT. For a comprehensive comparison, see Table 8.1 with  $c = O(1)$ .

[GKST17] viewed Nisan's characterization in terms of characterizing dependencies (Definition 6.4.1). They used this technique to get a reduction from sum of ROABPs to a single ROABP. This main idea of using characterizing dependencies for reduction is summarized at the start of Section 8.3.1 in this chapter. This is inherently a whitebox reduction and indeed [GKST17] use it to give a  $\text{poly}(n, d, r)$ -time whitebox PIT for sum of constantly-many ROABPs. For blackbox PIT, they give an indirect reduction, which takes

quasi-polynomial time. Moreover, instead of a hitting set for single ROABP, they require something stronger, which a hitting set might not always provide. They need an efficient shift that  $l$ -concentrates a single ROABP. A polynomial is said to be  $l$ -concentrated if all of its coefficients are in the linear span of its coefficients corresponding to monomials having variable support  $< l$ . They prove that this efficient shift also  $l$ -concentrates sum of constantly-many ROABPs. This method yields a quasi-poly time reduction, since they fix  $l$  to  $O(\log s)$  ( $s$  is size of ROABP) and apply brute-force hitting set after the shift. The main contribution of this chapter is to convert the whitebox reduction of [GKST17] to a direct blackbox reduction, that from a hitting set for sum of ROABPs to a hitting set for a single ROABP (Theorem 8.3.7).

**Future Directions:** In the context of this work and previous related works, a variety of open problems arise:

- Design a  $\text{poly}(s)$ -time blackbox PIT algorithm for  $(\log s)$ -variate, size- $s$  ROABP. This will also solve standard multivariate diagonal depth-3 model [FSS14]. Without loss of generality, ROABP can also be assumed to be syntactically homogeneous (Theorem 8.2.4 and Lemma 8.2.6).
- In Theorem 8.2.9, can we remove the restriction of degree-preserving sum? If so, then that would solve diagonal depth-3 model (Lemma 6.4.5). Design a  $\text{poly}(r^n, c, d)$ -time blackbox PIT for sum of  $c$  ROABPs, each of width  $r$  computing an  $n$ -variate polynomial of degree  $d$ .
- Bring down  $2^n$  dependence in Theorem 8.3.7 to  $\text{poly}(n)$ . Currently, the dependence on  $c$  is doubly exponential, both in this reduction and also in [GKST17]. One would also like to bring it down to  $\text{poly}(c)$  or even just to single exponential like  $(ndr)^{O(c)}$ .



## Chapter 9

# Conclusion

We discussed two fundamental problems in algebraic complexity theory in this thesis – Polynomial factoring and identity testing. For polynomial factoring, we consider the class of sparse polynomials, while for PIT we consider  $\Sigma^{[2]}\Pi\Sigma\Pi^{[\text{ind-deg } d]}$  circuits and sum of ROABPs. Our underlying theme was to make structural observations for these classes and use them to derive the necessary algorithms.

We explored the problem of sparse multivariate factoring, in which we are asked to completely factor a given sparse polynomial. We note that runtime of any factoring algorithm here will be lower bounded by sparsity of factors, as we need to output the factors in sparse representation itself. It is known that sparse polynomials can have dense factors (see examples 1.2.1 & 1.2.2). However these factors are sparse, when we consider individual degrees to be constant. This brings us to the sparsity conjecture stated in Conjecture 1.2.3, that  $s$ -sparse polynomials of constant individual degree have  $\text{poly}(s)$ -sparse factors. Making progress towards the sparsity conjecture is the underlying motivation for Part-I of this thesis. Moreover, [BSV20] showed that factor-sparsity bound also gives us a upper bound on the run time of sparse factoring in the bounded individual degree regime (see Lemma 3.4.1). Therefore, it suffices to prove sparsity conjecture to get poly-time sparse factoring algorithm.

We proved the sparsity conjecture for the class of symmetric polynomials. For an  $s$ -sparse, symmetric polynomial  $f$  of individual degree  $d$ , we showed a sparsity bound of

$s^{O(d^2 \log d)}$  for any factor of  $f$ . This gives us the desired  $\text{poly}(s)$  bound for constant  $d$ . We achieve this bound by observing that symmetric polynomials have somewhat ‘low’ min-entropy and then using it with our structural result that all integral points in the Newton polytope also have low min-entropy. The main open problem here of course is to prove the sparsity conjecture for general sparse polynomials. We could ask a possibly simpler question. Let  $f$  be an  $s$ -sparse polynomial of constant individual degree  $d$  such that  $f = g^e$  for some polynomial  $g$ , and a constant  $e$ . Then can one prove that  $g$  is  $\text{poly}(s)$ -sparse? While we don’t fully solve this problem here, we instead develop a  $\text{poly}(s)$ -time algorithm to test whether  $f$  is of the form  $g^e$ , for any  $e \in \{2, \dots, d\}$ . Another variant of the sparsity conjecture is to show that the cofactor is sparse. i.e. Let  $f, h$  be two  $s$ -sparse polynomials of constant individual degree  $d$  such that  $f = gh$ . Then is  $g$   $s^{\text{poly}(d)}$ -sparse? We were able to show that  $g$  is  $s^{O(d \log s)}$ -sparse, when  $h$  is a multilinear polynomial.

In Part-II, we first give a poly-time blackbox PIT for  $\Sigma^{[2]}\Pi\Sigma\Pi^{[\text{ind-deg } d]}$  circuits. This model is more general than  $\Sigma^{[2]}\Pi\Sigma\Pi^{[d]}$  circuits, where the bottom  $\Sigma\Pi$  computes polynomials of constant total degree  $d$ , while in  $\Sigma^{[2]}\Pi\Sigma\Pi^{[\text{ind-deg } d]}$  circuits the bottom  $\Sigma\Pi$  computes polynomials of constant individual degree  $d$ . We were able to design this PIT by hitting appropriate resultants, where we made use of our structural result (Theorem 7.2.1), which shows that sub-resultant of two sparse polynomials is a multiple of their GCD and resultant of their co-prime parts. The main open problem here is to remove the individual degree restriction and show (even whitebox) PIT for  $\Sigma^{[2]}\Pi\Sigma\Pi$  circuits.

In our second PIT result, we show a deterministic poly-time blackbox PIT for sum of constantly-many, log-variate, constant-width ROABPs. We showed a direct reduction from blackbox PIT for sum of ROABPs to blackbox PIT for a single ROABP. This reduction is poly-time when number of variables is logarithmic in the input size and number of summands is constant. We used this reduction in conjunction with our structural result on syntactic homogeneity of ROABPs, to derive our poly-time blackbox PIT. Our reduction also improves over the previous best PIT algorithm of [GKST17] in the log-variate regime. An additional benefit of our reduction is that if one can show a poly-time blackbox PIT for log-variate ROABPs, then one also gets a poly-time blackbox PIT for sum of constantly-



many, log-variate ROABPs (without the width restriction). We note that blackbox PIT for log-variate ROABPs is the main open problem here, which will also solve PIT for general  $\Sigma\wedge\Sigma$  circuits. The finer open problems related to each result above, are mentioned in the Discussion section of each chapter.



# Bibliography

- [AGKS15] Manindra Agrawal, Rohit Gurjar, Arpita Korwar, and Nitin Saxena. Hitting-sets for ROABP and sum of set-multilinear circuits. *SIAM Journal on Computing*, 44(3):669–697, 2015. [11](#), [20](#), [109](#), [111](#), [132](#)
- [Agr05] Manindra Agrawal. Proving lower bounds via pseudo-random generators. In *FSTTCS*, volume 3821 of *Lecture Notes in Computer Science*, pages 92–105, 2005. [9](#)
- [AGS19] Manindra Agrawal, Sumanta Ghosh, and Nitin Saxena. Bootstrapping variables in algebraic circuits. *Proceedings of the National Academy of Sciences*, 116(17):8107–8118, 2019. [9](#), [10](#), [91](#), [110](#), [111](#)
- [AKS04] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. *Annals of mathematics*, pages 781–793, 2004. [1](#), [9](#)
- [Alo99] N. Alon. Combinatorial nullstellensatz. *Combinatorics, Probability and Computing*, 8:7–29, 1999. [76](#)
- [ASS13] Manindra Agrawal, Chandan Saha, and Nitin Saxena. Quasi-polynomial hitting-set for set-depth- $\Delta$  formulas. In *Symposium on Theory of Computing Conference, STOC, Palo Alto, CA, USA, June 1-4*, pages 321–330, 2013. [111](#)
- [AV08] Manindra Agrawal and V Vinay. Arithmetic circuits: A chasm at depth four. In *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 67–75. IEEE, 2008. [10](#), [91](#)
- [Ber67] Elwyn R Berlekamp. Factoring polynomials over finite fields. *Bell System Technical Journal*, 46(8):1853–1859, 1967. [12](#), [46](#)
- [BG21] Vishwas Bhargava and Sumanta Ghosh. Improved hitting set for orbit of roabps. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021. [110](#)
- [BIZ18] Karl Bringmann, Christian Ikenmeyer, and Jeroen Zuiddam. On algebraic branching programs of small width. *Journal of the ACM (JACM)*, 65(5):1–29, 2018. [112](#)

- [BOC92] Michael Ben-Or and Richard Cleve. Computing algebraic formulas using a constant number of registers. *SIAM Journal on Computing*, 21(1):54–58, 1992. (Preliminary version in STOC’88). [110](#)
- [BS21] Pranav Bisht and Nitin Saxena. Blackbox identity testing for sum of special roabps and its border class. *computational complexity*, 30(1):1–48, 2021. [xv](#)
- [BS22] Pranav Bisht and Nitin Saxena. Derandomization via symmetric polytopes: Poly-time factorization of certain sparse polynomials. 2022. (to appear in the proceedings of the 42nd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS), 2022). [xv](#), [65](#)
- [BSV20] Vishwas Bhargava, Shubhangi Saraf, and Ilya Volkovich. Deterministic factorization of sparse polynomials with bounded individual degree. *Journal of the ACM (JACM)*, 67(2):1–28, 2020. (Preliminary version in FOCS 2018). [7](#), [8](#), [9](#), [11](#), [13](#), [14](#), [16](#), [26](#), [31](#), [35](#), [46](#), [48](#), [49](#), [51](#), [55](#), [62](#), [69](#), [91](#), [92](#), [93](#), [139](#)
- [BV22] Pranav Bisht and Ilya Volkovich. On solving sparse polynomial factorization related problems. *Electron. Colloquium Comput. Complex.*, TR22-070, 2022. (to appear in the proceedings of the 42nd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS), 2022). [xv](#)
- [CKS19] Chi-Ning Chou, Mrinal Kumar, and Noam Solomon. Closure results for polynomial factorization. *Theory of Computing*, 15(1):1–34, 2019. [8](#)
- [CLO15] D. A. Cox, J. Little, and D. O’Shea. *Ideals, varieties, and algorithms - an introduction to computational algebraic geometry and commutative algebra (4. ed.)*. Undergraduate texts in mathematics. Springer, 2015. [17](#), [81](#)
- [CR88] Benny Chor and Ronald L Rivest. A knapsack-type public key cryptosystem based on arithmetic in finite fields. *IEEE Transactions on Information Theory*, 34(5):901–909, 1988. [6](#)
- [CZ81] David G Cantor and Hans Zassenhaus. A new algorithm for factoring polynomials over finite fields. *Mathematics of Computation*, pages 587–592, 1981. [12](#), [46](#)
- [DdO14] Zeev Dvir and Rafael Mendes de Oliveira. Factors of sparse polynomials are sparse, 2014. [26](#), [31](#), [48](#)
- [DDS21] Pranjal Dutta, Prateek Dwivedi, and Nitin Saxena. Deterministic identity testing paradigms for bounded top-fanin depth-4 circuits. In *36th Conference on Computational Complexity (CCC 2021)*, volume 5, page 9, 2021. [10](#), [97](#), [106](#)
- [DL77] Richard A DeMillo and Richard J Lipton. A probabilistic remark on algebraic program testing. Technical report, Georgia Inst of Tech Atlanta School of Information and Computer science, 1977. [9](#)

- [DSS18] Pranjali Dutta, Nitin Saxena, and Amit Sinhababu. Discovering the roots: Uniform closure results for algebraic classes under factoring. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1152–1165, 2018. [8](#)
- [Dut18] P. Dutta. Discovering the roots: Unifying and extending results on multivariate polynomial factoring in algebraic complexity. *Master’s thesis, Chennai Mathematical Institute*, 2018. [53](#)
- [FGS18] Michael A Forbes, Sumanta Ghosh, and Nitin Saxena. Towards blackbox identity testing of log-variate circuits. In *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018. [90](#), [111](#)
- [FGT17] Stephen A. Fenner, Rohit Gurjar, and Thomas Thierauf. Guest column: Parallel algorithms for perfect matching. *SIGACT News*, 48(1):102–109, 2017. [9](#)
- [For14] Michael Andrew Forbes. *Polynomial identity testing of read-once oblivious algebraic branching programs*. PhD thesis, Massachusetts Institute of Technology, 2014. [77](#), [122](#)
- [For15] M. A. Forbes. Deterministic divisibility testing via shifted partial derivatives. In *FOCS*, 2015. [14](#)
- [For16] Michael Forbes. Some concrete questions on the border complexity of polynomials. presentation given at the workshop on algebraic complexity theory wact 2016 in tel aviv, 2016. [89](#), [90](#), [112](#)
- [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*, pages 527–542. Springer, 2013. [86](#), [90](#), [111](#)
- [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. [11](#), [86](#), [109](#), [110](#), [111](#)
- [FS18] Michael A Forbes and Amir Shpilka. A pspace construction of a hitting set for the closure of small algebraic circuits. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1180–1192, 2018. [90](#)
- [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), New York, NY, USA, May 31 - June 03, 2014*, pages 867–875, 2014. [86](#), [87](#), [90](#), [110](#), [111](#), [137](#)
- [GCL92] K. O. Geddes, S. R. Czapor, and G. Labahn. *Algorithms for computer algebra*. Kluwer, 1992. [17](#), [81](#), [83](#)

- [GG20] Zeyu Guo and Rohit Gurjar. Improved explicit hitting-sets for roabps. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020. [132](#), [135](#), [136](#)
- [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. (Preliminary version in CCC’16). [11](#), [18](#), [109](#), [110](#), [111](#)
- [GKSS19] Zeyu Guo, Mrinal Kumar, Ramprasad Saptharishi, and Noam Solomon. Derandomization from algebraic hardness: Treading the borders. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 147–157. IEEE, 2019. [111](#)
- [GKST17] Rohit Gurjar, Arpita Korwar, Nitin Saxena, and Thomas Thierauf. Deterministic identity testing for sum of read-once oblivious arithmetic branching programs. *Computational Complexity*, pages 1–46, 2017. (Conference version in CCC 2015). [11](#), [18](#), [20](#), [75](#), [84](#), [85](#), [109](#), [110](#), [111](#), [113](#), [115](#), [122](#), [123](#), [128](#), [136](#), [137](#), [140](#)
- [GS98] V. Guruswami and M. Sudan. Improved decoding of reed-solomon and algebraic-geometric codes. In *Proceedings 39th Annual Symposium on Foundations of Computer Science (Cat. No.98CB36280)*, pages 28–37, 1998. [6](#)
- [GSS19] Zeyu Guo, Nitin Saxena, and Amit Sinhababu. Algebraic dependencies and pspace algorithms in approximative complexity over any field. *Theory of Computing*, 15(1):1–30, 2019. [90](#)
- [HS80] Joos Heintz and Claus P. Schnorr. Testing polynomials which are easy to compute (extended abstract). In *Proceedings of the Twelfth Annual ACM Symposium on Theory of Computing*, STOC ’80, pages 262–272, New York, NY, USA, 1980. ACM. [9](#)
- [Kal86] Erich Kaltofen. Uniform closure properties of p-computable functions. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 330–337, 1986. [8](#)
- [Kal87] Erich Kaltofen. Single-factor hensel lifting and its application to the straight-line complexity of certain polynomials. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 443–452, 1987. [8](#)
- [Kal89] Erich Kaltofen. Factorization of polynomials given by straight-line programs. *Adv. Comput. Res.*, 5:375–412, 1989. [6](#), [8](#)
- [Kal03] Erich Kaltofen. Polynomial factorization: a success story. In *Proceedings of the 2003 international symposium on Symbolic and algebraic computation*, pages 3–4, 2003. [6](#)
- [KI04] Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *computational complexity*, 13(1-2):1–46, 2004. [6](#), [9](#)

- [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)*, pages 46:1–46:15, 2016. [111](#), [112](#)
- [Kro82] Leopold Kronecker. *Grundzuge einer arithmetischen Theorie der algebraischen Grossen*. Berlin, G. Reimer, 1882. [79](#)
- [KS01] Adam R Klivans and Daniel Spielman. Randomness efficient identity testing of multivariate polynomials. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 216–223, 2001. [2](#), [14](#), [61](#), [79](#), [93](#)
- [KSS14] Swastik Kopparty, Shubhangi Saraf, and Amir Shpilka. Equivalence of polynomial identity testing and deterministic multivariate polynomial factorization. In *2014 IEEE 29th Conference on Computational Complexity (CCC)*, pages 169–180. IEEE, 2014. [2](#), [9](#)
- [KST19] Mrinal Kumar, Ramprasad Satharishi, and Anamay Tengse. Near-optimal bootstrapping of hitting sets for algebraic circuits. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 639–646. Society for Industrial and Applied Mathematics, 2019. [111](#)
- [KT90] Erich Kaltofen and Barry M Trager. Computing with polynomials given by black boxes for their evaluations: Greatest common divisors, factorization, separation of numerators and denominators. *Journal of Symbolic Computation*, 9(3):301–320, 1990. [6](#)
- [LLL82] Arjen K Lenstra, Hendrik Willem Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische annalen*, 261(ARTICLE):515–534, 1982. [12](#), [46](#), [57](#)
- [LST22] Nutan Limaye, Srikanth Srinivasan, and Sébastien Tavenas. Superpolynomial lower bounds against low-depth algebraic circuits. In *FOCS’21*, 2022. [10](#), [97](#)
- [MS01] Ketan D Mulmuley and Milind Sohoni. Geometric complexity theory i: An approach to the p vs. np and related problems. *SIAM Journal on Computing*, 31(2):496–526, 2001. [112](#)
- [MS08] Ketan D Mulmuley and Milind Sohoni. Geometric complexity theory ii: Towards explicit obstructions for embeddings among class varieties. *SIAM Journal on Computing*, 38(3):1175–1206, 2008. [112](#)
- [Mul12a] Ketan D. Mulmuley. The GCT Program toward the P vs. NP problem. *Commun. ACM*, 55(6):98–107, June 2012. [9](#), [112](#)
- [Mul12b] Ketan D. Mulmuley. Geometric Complexity Theory V: Equivalence between blackbox derandomization of polynomial identity testing and derandomization of Noether’s Normalization Lemma. In *FOCS*, pages 629–638, 2012. [9](#), [112](#)
- [MVV87] Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7:105–113, 1987. [9](#)

- [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. 85, 89, 110
- [Oli16] Rafael Oliveira. Factors of low individual degree polynomials. *computational complexity*, 25(2):507–561, 2016. 8
- [Ore22] Øystein Ore. Über höhere kongruenzen. *Norsk Mat. Forenings Skrifter*, 1(7):15, 1922. 9
- [Ost21] AM Ostrowski. Über die bedeutung der theorie der konvexen polyeder für die formale algebra. *Jahresberichte Deutsche Math. Verein*, 20:98–99, 1921. (English translated version republished in ACM SIGSAM Bulletin, 33(1):5, 1999). 31
- [PS21] Shir Peleg and Amir Shpilka. Polynomial time deterministic identity testing algorithm for  $\sigma$  [3]  $\pi\sigma\pi$  [2] circuits via edelstein–kelly type theorem for quadratic polynomials. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 259–271, 2021. 10, 97
- [RS05] Ran Raz and Amir Shpilka. Deterministic polynomial identity testing in non-commutative models. *Computational Complexity*, 14(1):1–19, 2005. 11, 109
- [Sap16] Ramprasad Saptharishi. A survey of lower bounds in arithmetic circuit complexity. Technical report, <https://github.com/dasarpmar/lowerbounds-survey/>, 2016. 9, 89
- [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. 86, 87
- [Sax09] Nitin Saxena. Progress on polynomial identity testing. *Bulletin of the EATCS*, 99:49–79, 2009. 9
- [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. 9
- [Sch80] Jacob T Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM (JACM)*, 27(4):701–717, 1980. 9
- [Sch00] Andrzej Schinzel. *Polynomials with special regard to reducibility*, volume 77. Cambridge University Press, 2000. 26
- [SSS13] C. Saha, R. Saptharishi, and N. Saxena. A case of depth-3 identity testing, sparse factorization and duality. *Computational Complexity*, 22(1):39–69, 2013. 10, 16
- [ST21a] Chandan Saha and Bhargav Thankey. Hitting sets for orbits of circuit classes and polynomial families. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021. 110



- [ST21b] A. Sinhababu and T. Thierauf. Factorization of polynomials given by arithmetic branching programs. volume 30, pages 1–47. Springer, 2021. [8](#), [53](#)
- [Sud97] Madhu Sudan. Decoding of reed solomon codes beyond the error-correction bound. *Journal of complexity*, 13(1):180–193, 1997. [6](#)
- [Sud98] Madhu Sudan. Algebra and computation. lecture notes, 1998. [6](#)
- [SV09] Amir Shpilka and Ilya Volkovich. Improved polynomial identity testing for read-once formulas. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 700–713. Springer, 2009. [87](#)
- [SV10] Amir Shpilka and Ilya Volkovich. On the relation between polynomial identity testing and finding variable disjoint factors. In *International Colloquium on Automata, Languages, and Programming*, pages 408–419. Springer, 2010. [8](#)
- [SV15] A. Shpilka and I. Volkovich. Read-once polynomial identity testing. *Computational Complexity*, 24(3):477–532, 2015. [77](#)
- [SV18] S. Saraf and I. Volkovich. Blackbox identity testing for depth-4 multilinear circuits. *Combinatorica*, 38(5):1205–1238, 2018. [10](#)
- [SY10] Amir Shpilka and Amir Yehudayoff. *Arithmetic circuits: A survey of recent results and open questions*. Now Publishers Inc, 2010. [3](#), [9](#), [79](#), [131](#), [135](#)
- [Vai15] Rishabh Vaid. Blackbox identity testing for simple depth 3 circuits. Master’s thesis, Indian Institute of Technology Kanpur, 2015. [87](#)
- [Vol15] Ilya Volkovich. Deterministically factoring sparse polynomials into multilinear factors and sums of univariate polynomials. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2015)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015. [8](#)
- [Vol17] Ilya Volkovich. On some computations on sparse polynomials. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017. [8](#), [10](#), [14](#), [16](#), [91](#)
- [vzG06] Joachim von zur Gathen. Who was who in polynomial factorization: 1. In *Proceedings of the 2006 international symposium on Symbolic and algebraic computation*, page 2, 2006. [6](#)
- [vzGG13] Joachim von zur Gathen and Jürgen Gerhard. *Modern computer algebra*. Cambridge university press, 2013. [6](#), [17](#), [27](#), [81](#)
- [vzGK85] Joachim von zur Gathen and Erich Kaltofen. Factoring sparse multivariate polynomials. *Journal of Computer and System Sciences*, 31(2):265–287, 1985. [6](#), [7](#), [10](#), [16](#)
- [Zie12] Günter M Ziegler. *Lectures on polytopes*, volume 152. Springer Science & Business Media, 2012. [26](#)

- [Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. In *International symposium on symbolic and algebraic manipulation*, pages 216–226. Springer, 1979. [9](#)

# Index

- NP, 1, 3
- P, 1, 3
- VNP, 3, 8
- VP, 3, 8, 10, 91
- bit-complexity, 12, 13, 46, 52, 59
- blackbox PIT, 9, 11, 18–20, 79, 90, 109–112, 119, 120, 122, 131, 132, 135, 136, 140, 141
- co-factor sparsity, 14, 64, 66
- depth-2 circuits, 3
- depth-3 circuits, 3, 112
- depth-4 circuits, 4, 10
- derandomization, 1, 131
- diagonal depth-3 circuits, 86, 90, 111, 112
- greatest common divisor, 26
- hitting set generator, 76, 93, 97, 103
- hyperplane, 1, 26, 36, 37, 48
- interlacing, 98
- interpolation, 77, 78
- log-variate, 11, 18–20, 86, 109–113, 119, 120, 123, 127, 128, 131–133, 135, 136, 140, 141
- Minkowski sum, 26, 30, 31
- multilinear polynomial, 10, 15, 16, 61, 62, 71, 140
- multivariate polynomial factoring, 2
- Newton polytope, 26, 30, 31, 35, 36, 49, 140
- Nisan’s characterization, 85, 88, 89, 114, 115
- resultant, 17, 80–82, 92–94, 101, 140
- sparse polynomial factorization, 6, 21
- subresultant, 17, 80, 83, 84, 94, 95
- symmetric group, 25, 37
- syntactically homogeneous, 113, 115, 117, 137
- unique factorization domain, 27, 81
- unique projections, 64, 65, 71