Treading the Borders

for Explicitness, Circuit Factoring, and Identity Testing

A *thesis* submitted

in partial fulfilment of the requirements

for the degree of

Doctor of Philosophy

by

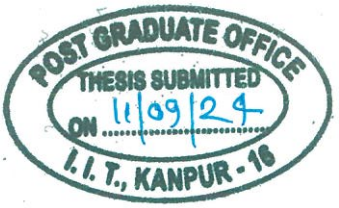
Prateek Dwivedi

18111269

Department of Computer Science and Engineering

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

September 2024



Certificate

It is certified that the work contained in this thesis entitled **"Treading the Borders for Explicitness, Circuit Factoring, and Identity Testing"** by **Prateek Dwivedi** has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Prof. Nitin Saxena
N.Rama.Rao.Chair Professor and J.C.Bose Fellow
Department of Computer Science and Engineering
Indian Institute of Technology Kanpur

September 2024

Declaration

This is to certify that the thesis titled **"Treading the Borders for Explicitness, Circuit 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 with appropriate citations and acknowledgments, in line with established norms and practices.

Prateek Dwivedi

Roll No. 18111269

Program: Doctor of Philosophy

Department of Computer Science and Engineering

Indian Institute of Technology Kanpur

September 2024

SYNOPSIS

Name of the student: **Prateek Dwivedi** Roll No: **18111269**

Degree for which submitted: **PhD** Department: **CSE**

Thesis title: **Treading the Borders for Explicitness, Circuit Factoring, and Identity Testing**

Thesis supervisors: **Prof. Nitin Saxena**

Month and year of thesis submission: **September 2024**

Polynomials are an indispensable mathematical object of study in theoretical computer science, where many computational problems can be formulated using them. Algebraic complexity theory focuses on understanding these computations using standard operations such as additions and multiplications. In this doctoral thesis, we study approximate (border) computation of polynomials to address some unresolved questions in classical algebraic complexity theory. In this thesis, we de-border and de-randomize some problems using newly introduced paradigms. Additionally, the work proposes a presentable model for approximate computation and motivates its study with applications to problems in polynomial factoring.

Algebraic circuits are a natural and structured computational model pioneered by Leslie G. Valiant in 1979 to study the complexity of polynomials. In the seminal research, Valiant posed one of the most central questions of algebraic complexity: can a small-size algebraic circuit compute the permanent polynomial? It is conjectured to be false and is famously formulated as $\mathsf{VP} \neq \mathsf{VNP}$. Mulmuley and Sohoni introduced Geometric Complexity Theory to use algebraic geometry and representation theory to solve big conjectures of computational complexity including P vs. NP. In the process, they strengthened the Valiant's conjecture to $\overline{\mathsf{VP}} \neq \mathsf{VNP}$. For an algebraic complexity class $\mathcal{C}$, we say that a

polynomial f over a field $\mathbb{F}$ is in the border class $\overline{\mathcal{C}}$, if there is a polynomial g over $\mathbb{F}(\varepsilon)$ such that $g = f + \varepsilon \cdot Q$, where Q is over $\mathbb{F}[\varepsilon]$ and $g \in \mathcal{C}$ over $\mathbb{F}(\varepsilon)$. The notion of approximating f stems from observing that g comes arbitrarily close to f as $\varepsilon \to 0$. The primary theme of our work is to upper bound the circuit size complexity of f given the size complexity of its approximating polynomial g. Within this framework, such questions cannot be immediately answered a priori.

Explicitness of Border Classes

The first part of the thesis studies the approximation of polynomials using algebraic circuits. The interest in border complexity and its connection to classical algebraic complexity has gained popularity in recent years. Motivated by the universality of border depth-3 circuit of constant top fan-in due to Mrinal Kumar, we study the border of restricted depth-4 circuits. In the first part of the thesis, we develop a de-bordering paradigm called DiDIL—Divide, Derive, Induct, with Limits, to de-border restricted depth-4 circuit $\overline{\Sigma^{[k]}\Pi\Sigma\wedge}$ and prove that when the top fan-in is constant, there are small Algebraic Branching Program (ABP) which can compute these polynomials. Before this work, it was not clear whether these polynomials were even in a believably larger class of explicit polynomials called VNP.

Most of the known de-bordering techniques, including the one we develop in this thesis, are designed to work on specific models, and they do not scale to general circuits. One of the reasons for the limited understanding of border circuits is their inherent non-constructive definition. In the thesis, we develop a constructive, or a presentable version of border circuits, and over finite fields prove that presentable border classes like $\overline{\mathsf{VP}_\varepsilon}$ and $\overline{\mathsf{VNP}_\varepsilon}$ lie in VNP. Such strong de-bordering results are currently open for classical border classes.

Circuit Factoring

In a series of works from the 1980s, Kaltofen proved that the complexity class VP is closed under factoring, which initiated the study of factor closure of algebraic complexity classes. It is believed that almost all the natural algebraic classes of multivariate polynomials are 'well behaved' if they are closed under factoring. It was a long-time conjecture of Bürgisser that the class VNP is closed under factoring over all fields, which was resolved in 2018 by Chou, Kumar, and Solomon over fields of large characteristics. In this thesis, we make progress on the open problem by proving that for all finite fields, and all factors, VNP is closed under factoring. Consequently, the factors of VP are also in VNP in fields of small characteristics. The prime characteristic cases were open before due to the inseparability obstruction, that is, when the multiplicity is not co-prime to the field characteristic.

The study of presentable border classes in the first part of the thesis helped us make significant progress on the quarter-century old *factor conjecture* of Bürgisser. In this thesis, we prove that the irreducible, separable, and polynomial degree factors of a polynomial-size circuit are in VNP. The conjecture posits that small circuits can compute these low-degree factors, that is, they are in VP. Before this work, these factors were proved to be in $\overline{\mathsf{VP}}$ by Bürgisser. Moreover, it is open whether $\overline{\mathsf{VP}}$ is in VNP.

Identity Testing

A natural and fundamental computational problem is to ask if the circuit computes an identically zero polynomial. The problem has a simple randomized algorithms due to the PIT Lemma, and de-randomization of it is closely related to algebraic and boolean circuit lower bounds. The input to the algorithm is an algebraic circuit computing a multivariate polynomial, and the problem is studied in two settings—blackbox and whitebox. While in the former, we decide only from circuit evaluation, in the latter, we have complete access to the internals of the circuit. For a long time, in the deterministic world, nothing better than an exponential time brute force algorithm was known for even constant depth

circuits. Agrawal and Vinay proved that an efficient PIT algorithm for the depth-4 circuit $\Sigma\Pi\Sigma\Pi$ would almost solve the general case, highlighting the significance of the constant depth regime.

In a breakthrough work by Limaye, Tavenas, and Srinivasan in 2021, a subexponential time algorithm for general constant depth circuits, including $\Sigma\Pi\Sigma\Pi$, was obtained from the hardness vs. randomness trade-off. In this thesis, we give the first polynomial-time whitebox algorithm to solve the PIT on restricted depth-4 circuits $\Sigma^{[k]}\Pi\Sigma\wedge$ when k is constant. These circuits compute polynomials of the form — sum of k-many products of sum of univariates. In addition, we extend our algorithms to solve PIT on border classes $\overline{\Sigma^{[k]}\Pi\Sigma\wedge}$ and $\overline{\Sigma^{[k]}\Pi\Sigma\Pi^{[\delta]}}$ in quasi-polynomial time when the top fan-in is constant.

ACKNOWLEDGEMENTS

This thesis is the outcome of a long and meaningful journey—one that would not have been possible without the support, guidance, and kindness of many people. I take this opportunity to express my heartfelt gratitude to all those who have stood by me along the way. I am also deeply thankful to the countless researchers in theory who work tirelessly to make many meaningful dents in our understanding of the world—any research done today stands on the shoulders of their efforts.

I express my heartfelt gratitude to my advisor, Prof. Nitin Saxena, for his unwavering guidance throughout my Ph.D. journey. When I first started as his student, he asked me "Why do you want to work in theory?" and I answered, "Because it's exciting". Now, as I reflect on all these years, I realize that he turned that simple response into the best possible answer. Since then, he has taught me countless lessons, from tackling complex problems to making consistent progress week by week. He instilled in me the importance of discipline in research, even emphasizing the value of negative results as progress. I am especially grateful for his encouragement to participate in workshops and conferences, significantly enriching my academic experience. If it were not for his support and understanding during the various family medical emergencies that I faced during my Ph.D., I would not have been able to finish my dissertation. In addition to research, you also taught me to be a good teacher. Thank you for entrusting me with multiple teaching assistant responsibilities. Especially for the last experience, where, in addition to imparting knowledge, we worked together to ensure fairness for everyone and restoring the students' faith in the system. Thank you, Prof. Saxena, for being a constant source of inspiration and encouragement.

The work in this thesis is based on joint research with Pranjal Dutta, C.S. Bhargav, and my advisor. I want to express my gratitude to Pranjal Dutta, with whom I embarked on my PhD journey. We started working together on a monumental project that, despite the challenges of the pandemic, materialised well. Your patience in teaching me 'fundamental' topics of complexity theory and algebra have been invaluable. I have learned a lot from you, both in research and beyond. I truly enjoyed our endless debates on academics, life, and politics. Although we were on opposite ends of the spectrum most of the time, our long and intense discussions were always insightful and enriching. Thanks to my other collaborator, C.S. Bhargav, with whom I have had countless deep and engaging discussions on research problems. These discussions helped solidify so many of my ideas and were instrumental in our progress. Our endless conversations, whether about old papers or new results, were

always engaging and productive. We tirelessly debated, found errors in the proofs, and enthusiastically worked to fix it, always pushing the boundaries of our work. I truly cherish our association, both as researchers and as friends.

I would like to extend my gratitude to several people who have supported me throughout my PhD journey. Mahesh, thank you for helping me overcome academic challenges—your grit and determination have been truly inspiring. I am grateful to Rajendra and Priyanka for advice on how to drive my academic career forward; I have fond memories of our time together. My seniors Sumanta, Amit, Ashish, and Pranav thank you for guiding me in the early days of my research. Sharing the lab with Muzafar has been a delight. Our discussions have taught me so much, especially about Kashmir, and your joyful nature has brought so much energy to the lab. Thank you, Akhil, for all the philosophical discussions on the deepest meta-questions. I truly admire the clarity and purity of your thoughts. I also thank my other lab buddies, Foram, Tufan, Anindya, Ras, and Rizwan.

Outside academia, I owe so much to Anurag, who has been by my side since our school days, always encouraging me to be the best version of myself. I am grateful for keeping a check on me when it all became tough for me. Govind and Ankit, without your push, I might never have found the courage to sit for GATE and eventually discover my passion for research. The joy, laughter and endless roasting sessions we shared made everything lighter. I am thankful to have stayed connected through it all, making even the hardest moments easier to bear.

I would also like to thank Avideep, my closest friend during the Ph.D. years. Living next door, he was a constant source of warmth, humour, and thoughtful conversation. His presence made the journey not only bearable but truly enjoyable. Through him, I developed a deep appreciation for Bengali culture, which has since become very dear to me. Without our slow, shared Sundays, life at IIT Kanpur would have felt incomplete. I am also grateful to Soumya, with whom Avideep and I shared many wonderful moments that made this journey special. Your sincerity and dedication left a lasting impression on me. Thank you for the calm strength you brought to our little circle.

I owe so much to my family. To my brothers—thank you for always believing in me. Amit Bhaiya was the one who dropped me off to Kanpur when I first began this journey. It breaks my heart that he's not here to see me finish it. But I know he would have been proud of his little brother for making it through. Thank you, Manish Bhaiya, for your unwavering faith in me, and Nikhil Bhaiya, for being more of a friend than a brother. I am deeply grateful to Mummy and Papa for their unconditional love, and for standing by

me through countless sleepless nights and difficult days with endless patience and quiet strength.

Lastly, I want to thank my fiancée, Shweta, for countless things that could fill pages. You stood by me through long nights, tough moments, and endless drafts, always being my companion. Your love gave me strength and dispelled the fear of failure. Meeting you on that trip to Lucknow for the IPL match was nothing short of a blessing. Your honesty made me humble and a better person. Thank you for being my anchor, my partner, and my biggest supporter. This achievement is as much yours as it is mine.

वर्णानामर्थसंघानां रसानां छन्दसामपि।
मङ्गलानां च कर्त्तारौ वन्दे वाणीविनायकौ॥

I pray to Saraswati and Ganesha, who create letters, meanings, emotions, rhythms, and all that is auspicious.

Contents

List of Publications

The thesis is based on the following list of publications in chronological order. The names of the authors are in alphabetical order.

1. **Deterministic Identity Testing Paradigms for Bounded Top-Fanin Depth-4 Circuits.**
   Pranjal Dutta, Prateek Dwivedi, and Nitin Saxena.
   In 36th Computational Complexity Conference (CCC 2021).

2. **Demystifying the border of depth-3 algebraic circuits.**
   Pranjal Dutta, Prateek Dwivedi, and Nitin Saxena.
   In 62nd Annual Symposium on Foundations of Computer Science (FOCS), 2021.
   Invited in the special issue of SIAM Journal on Computing (SICOMP).

3. **Learning the coefficients: A presentable version of border complexity and applications to circuit factoring.**
   C.S. Bhargav, Prateek Dwivedi, and Nitin Saxena.
   In 56th Annual ACM Symposium on Theory of Computing (STOC), 2024.

The following list of publications is not included in the thesis.

1. **Lower Bounds for the Sum of Small-Size Algebraic Branching Programs.**
   C.S. Bhargav, Prateek Dwivedi, and Nitin Saxena.
   In 19th Annual Conference of Theory and Applications of Models of Computation (TAMC), 2024.
   Invited in the special issue of Theoretical Computer Science(Theor. Comput. Sci.)

List of Figures

List of Tables

Dedicated to Mummy Papa.

Chapter 1

Introduction

The essence of mathematics lies in its freedom.

Georg Cantor

Theoretical Computer Science has formed the mathematical backbone of computation long before computers came into existence. For much of its history, the thirst for solving problems of interest efficiently has been quenched by an ever-growing understanding of computational models, aided by mathematical constructs. In hindsight, it was only natural to study these problems collectively, where they were classified based on the various notions of *efficiency* – for instance, time and space. Developing the mathematical machinery necessary to formally explore the differences between these classes constitutes the area we call *Computational Complexity Theory.*

The work in this thesis pertains to a subfield of Computation Complexity where the object of our interest is Polynomials. These algebraic objects defined over a field $\mathbb{F}$ using variables $\{x_1, \ldots, x_n\}$ are expressed as follows in the fully expanded form:

$$f(\boldsymbol{x}) \quad \coloneqq \quad \sum_{\boldsymbol{a} \in \mathbb{N}^n} c_{\boldsymbol{a}} \cdot x_1^{a_1} \cdots x_n^{a_n},$$

where $\boldsymbol{a} = (a_1, \ldots, a_n)$ is a tuple of non-negative integers representing exponents of variables, and $c_{\boldsymbol{a}}$ is constant from the field $\mathbb{F}$. The summation in the above polynomial expression is finite, where in each term $x_1^{a_1} \cdots x_n^{a_n}$ is called a monomial. The expanded form in some cases may not be the most efficient way to express it — for example,

$f = x_1x_2 - 5x_1 - 3x_2 + 15 = (x_1 - 3) \cdot (x_2 - 5)$. Traditional computational models cannot capture such structural properties so well, so we will work with a more abstract and succinct model called *Algebraic Circuits*.

Definition 1.1 (Algebraic Circuits and Formula)**.** *An algebraic circuit is a directed acyclic graph comprising alternating layers of addition gates (+) and multiplication gates (×). The leaves of the circuit are input variables or constants from the field $\mathbb{F}$. The circuit computes an n-variate polynomial in $\mathbb{F}[x_1, \ldots, x_n]$ at the output node of the graph. A circuit is called a formula if the out-degree of every gate is at most one.*

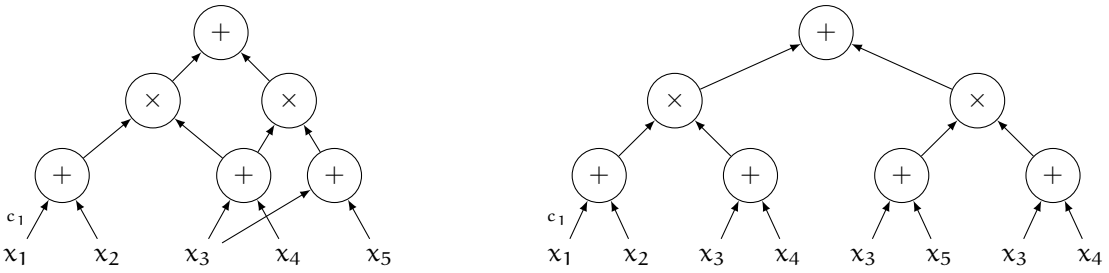


FIGURE 1.1: Algebraic circuit (left) and formula (right) computing $f = (c_1x_1 + x_2)(x_3 + x_4) + (x_3 + x_5)(x_3 + x_4)$.

Computing a polynomial by a circuit always refers to computing a family of polynomials $\{f_n\}$, one for each $n \in \mathbb{N}$. The measure of efficiency of computation by a circuit is the number of vertices and edges of the graph, which we refer to as the size of the circuit.

Definition 1.2 (Efficiency Measures of Circuits)**.** *The size of the smallest circuit over $\mathbb{F}$ computing the polynomial f is denotes by* $\text{size}_{\mathbb{F}}(f)$*. The number of edges in the longest path from the root to the leaf is called* $\text{depth}(f)$*.*

This thesis is divided into three parts: explicitness, circuit factoring, and identity testing. In each part, we will delve into a problem in algebraic complexity of a different flavor. We will discuss the work of this thesis formally after fixing a few notations in the next section. We will also state some results with relevant references that will be essential in our proofs later. The section additionally provides a brief overview of algebraic complexity theory.

1.1 Algebraic Computation and Classes

We start with fixing some helpful notations for this thesis. We use $[n]$ to denote the set $\{1, \ldots, n\}$, and $\mathbf{x} = (x_1, \ldots, x_n)$ for n variables. We use $\mathbb{F}[\mathbf{x}]$ to denote ring of polynomials

with coefficients from the field $\mathbb{F}$ on the variables $\mathbf{x}$. The fraction field consisting of rational polynomials of the form g/h where $g, h \in \mathbb{F}[\mathbf{x}]$ such that $h \neq 0$, is denoted by $\mathbb{F}(\mathbf{x})$. For vectors $\mathbf{a} = (a_1, \ldots, a_n), \mathbf{b} = (b_1, \ldots, b_n) \in \mathbb{F}^n$, and a variable t, we define $\mathbf{a} + t \cdot \mathbf{b} \coloneqq (a_1 + t \cdot b_1, \cdots, a_n + t \cdot b_n)$. For a mathematical object a, we denote its *boolean* encoding by $\langle a \rangle$. Finally, in the upcoming chapters we will frequently use formal power series rings and use $\mathbb{F}[\![\mathbf{x}]\!]$ to denote them.

As discussed in the previous section, polynomials can be classified on the basis of the efficiency parameters of the algebraic circuit. We formally define some of these classes that are relevant for this thesis. Formally defined by Valiant, we start with the class of efficiently computable polynomials [Val79].

Definition 1.3 (Valiant's P)**.** *The class* VP *is the set of all polynomials* $f \in \mathbb{F}[x_1, \ldots, x_n]$ *of degree* $\mathrm{poly}(n)$ *that can be computed by algebraic circuits of size* $\mathrm{poly}(n)$*.*

Similarly, we can define the class VF using algebraic formulas instead. An algebraic analogue of NP is defined using an exponential sum of VP polynomials.

Definition 1.4 (Valiant's NP)**.** *The class* VNP *is the set of all polynomials* $f \in \mathbb{F}[x_1, \ldots, x_n]$ *such that there exists a polynomial* $g \in \mathbb{F}[x_1, \ldots, x_n, y_1, \ldots, y_m]$ *in* VP *with* $m = \mathrm{poly}(n)$ *and*

$$f(\mathbf{x}) \quad = \quad \sum_{\mathbf{a} \in \{0,1\}^m} g(\mathbf{x}, \mathbf{a}).$$

We call $y_1, \ldots, y_m$ the *witness* (or *hypercube*) variables and $g(\mathbf{x}, \mathbf{y})$ as the *verifier circuit.* It is clear from the definitions that VP is contained in VNP, and Valiant's conjecture is that the inclusion is strict. Valiant proved that VNP contains *explicit* polynomials (also see [Bü00, Prop. 2.20]), making it a de facto class of *interesting* polynomials.

Proposition 1.5 (Valiant's criterion)**.** *Let* f *be a polynomial in* n *variables of degree* $\mathrm{poly}(n)$ *over a field* $\mathbb{F}$ *such that* $f = \sum_e c_e \mathbf{x}^e$*. Suppose that there exists a string function* $\phi : \{0,1\}^* \mapsto \{0,1\}^*$ *in* $\#\mathsf{P}/\mathrm{poly}$ *such that* $\phi(\langle e \rangle) = \langle c_e \rangle$*. Then, the polynomial* f *is in* VNP *over the field* $\mathbb{F}$*.*

Unlike the usual definition of $\#\mathsf{P}$ which consists of functions mapping $\{0,1\}^*$ to $\mathbb{N}$, we find it more convenient to consider functions that output binary strings. Over finite fields, we use a weaker version of the criterion in our proofs, where instead of assuming coefficient function $\phi \in \#\mathsf{P}/\mathrm{poly}$, we assume $\phi \in \#_p\mathsf{P}/\mathrm{poly}$ [Bü00, Section 4.3]. Formally that means, there exists a function $\psi \in \#\mathsf{P}/\mathrm{poly}$ such that $\phi(\langle e \rangle) = \psi(\langle e \rangle) \bmod p$ [1]. We

[1] In a slight abuse of notation, we assume the function ψ maps $\{0,1\}^*$ to $\mathbb{N}$.

omit this subtlety wherever it is clear from the context. For a formal definition, refer to Section 1.6.

Between algebraic circuits and formulas, another interesting model of polynomial computation is *Algebraic Branching Programs*.

Definition 1.6 (Algebraic Branching Programs (ABP))**.** *An algebraic branching program is a layered and directed graph with a source vertex* s *and a sink vertex* t*. All edges connect vertices from layer* i *to* $i+1$*. Further, the edges are labeled with linear polynomials over the underlying field* $\mathbb{F}$*. For every path* γ *from* s *to* t*,* $\mathrm{wt}(\gamma)$ *is the product of labels on the edges of the path* γ*. The polynomial computed by the ABP is defined as*

$$f := \sum_{path\ \gamma : s \rightsquigarrow t} \mathrm{wt}(\gamma)$$

The *depth* of an ABP is defined as the number of layers in the graph, and the *width* is the maximum number of nodes in a layer across the graph. The number of vertices used in the graph is *size* of an ABP. It is straightforward to see that the ABP is closed under addition and multiplication.

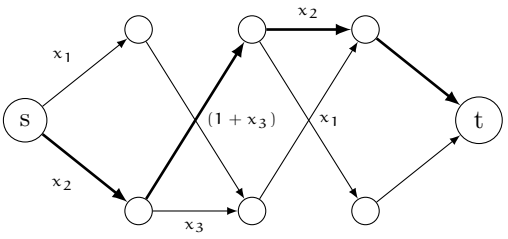


FIGURE 1.2: An example of Algebraic Branching Program. The highlighted path computes the polynomial $x_2^2 \cdot (1 + x_3)$.

Definition 1.7 (Valiant's Branching Program)**.** *The class* VBP *is the set of all polynomials* $f \in \mathbb{F}[x_1, \ldots, x_n]$ *that can be computed by an algebraic branching program of size* $\mathrm{poly}(n)$*.*

It is easy to see that formulas can be simulated by ABPs. However, to see circuits simulating them, we need to alternatively view ABPs as products of matrices. Structurally, the connection between an ABP and an algebraic circuit is more apparent when one realises that they are equivalent to *skew circuits* — at most one child of every product gate is not an input gate (see [MV97] and [MR08, Section 5]).

Nisan studied ABPs in a non-commutative setting and proved strong lower bounds against it using width characterization [Nis91]. Forbes and Shpilka extended that result

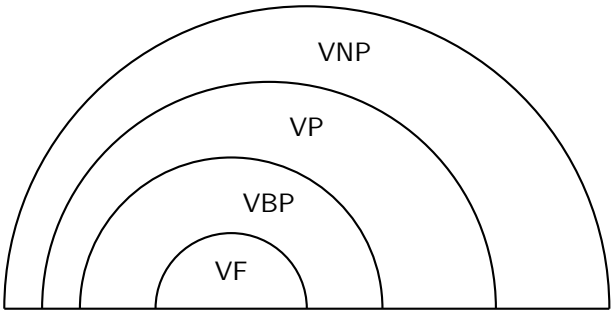


FIGURE 1.3: Containments of algebraic complexity classes.

to a special ABP which mimics non-commutative operations in the commutative world by restricting the number of time and the order of reading the variables [FS13b].

Definition 1.8 (Read-once Oblivious Algebraic Branching Program)**.** *For a fixed permutation* $\sigma : [n] \to [n]$ *on the variables, a* RO *over* σ *is an* ABP *where in the* i*-th layer the edge labels are univariate polynomials over* $x_{\sigma(i)}$*.* $\mathsf{ARO}(s)$ *defines a class of polynomials that can be computed by an* RO *of size at most* s *in every permutation.*

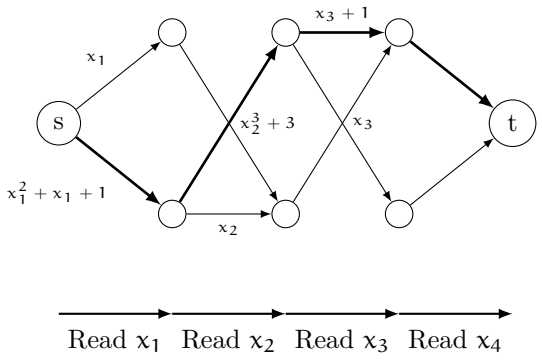


FIGURE 1.4: An example of ROABP. The high path computes the polynomial $(x_1^2 + x_1 + 1) \cdot (x_2^3 + 3) \cdot (x_3 + 1)$.

ROABPs are one of the few models that we understand quite well due to the structural characterization due to Nisan [Nis91]. We will discuss various properties and known results for polynomials computable by ROABP in later sections. Before moving on to formally discussing the work presented in the thesis, we will fix some notation to denote constant-depth circuits.

For constant-depth circuits, we use $\Sigma, \Pi, \wedge$ to denote the layers of summation, product, and powering gates, respectively.

1. **Depth-2 Circuits.** They are denoted by $\Sigma\Pi$ and compute sparse polynomials of the form $\Sigma_i c_i \boldsymbol{x}^{e_i}$, where e_i is the exponent vector.

2. **Depth-3 Circuits.** $\Sigma\Pi\Sigma$ circuits compute polynomials that are of the form— summation of products of linear polynomials. When the number of product gates is restricted to k, then we use the notation $\Sigma^{[k]}\Pi^{[d]}\Sigma$, where d denotes the degree. $\Sigma\wedge\Sigma$ compute sum of powers of linear polynomials and historically called depth-3 diagonal circuits.

3. **Depth-4 Circuits.** In general, we use $\Sigma\Pi\Sigma\Pi$ to denote the sum of product of sparse polynomials. In this thesis, we will be mostly interested in $\Sigma^{[k]}\Pi\Sigma\wedge$ which computes the polynomial of the form

$$f \quad = \quad \sum_{i=1}^{k} \prod_{j} \left(g_{ij_1}(x_1) + \cdots + g_{ij_n}(x_n)\right),$$

where $g_{ij_\ell} \in \mathbb{F}[x_\ell]$. We use $\Sigma\wedge\Sigma\Pi^{\delta}$ to denote polynomials of the form $\sum_i f_i(\boldsymbol{x})^{e_i}$ where $\deg(f_i) \leq \delta$. Similar to diagonal depth-3 circuits, a diagonal depth-4 circuit is denoted by $\Sigma\wedge\Sigma\wedge$. Finally, a bounded depth-4 circuit of the form $\sum_{i\in[k]} \prod_j g_{ij}(\boldsymbol{x})$ where $\deg(g_{ij}) \leq \delta$ is denoted by $\Sigma^{[k]}\Pi\Sigma\Pi^{[\delta]}$.

The surveys of [SY10, CKW11, Mah14, Sap13] provide an excellent overview of algebraic complexity theory and the current state of lower bounds. For a more extensive but slightly dated review, see [BCS97, Bü00].

1.2 Border Complexity

The notion of approximation is known to be useful in algorithms and in the wider spectrum of complexity theory, where the output is mildly corrupted with an error. There is a natural way to associate a Euclidean (or Zariski) topology with the polynomial ring. This confers a notion of limit and, thereby, a way of approximating a polynomial by a sequence of polynomials (see, e.g., [BIZ18, Section 2.3]). The topological notion of definition has been extensively explored to design algorithms for matrix multiplication [Str74, BCRL79, Bin80, CW90, LO15a]. However, in Valiant's framework, the simplest definition for *algebraic* approximation and border complexity was given by Búrgisser [Bür04].

We say that a polynomial $g(\boldsymbol{x}, \varepsilon) \in \mathbb{F}(\varepsilon)[x_1, \ldots, x_n]$ approximates a polynomial $f(\boldsymbol{x}) \in \mathbb{F}[x_1, \ldots, x_n]$ if $g = f + \varepsilon Q$, where $Q(\boldsymbol{x}, \varepsilon) \in \mathbb{F}[\varepsilon][x_1, \ldots, x_n]$. Refer to Definitions 2.1 and 3.1 for formal statements. The size of the smallest circuit that computes g and thereby approximates f, defines the *border complexity* of f, which is denoted by $\overline{\text{size}}(f)$. In

an algebraically closed field, like $\mathbb{C}$, the topological notion of approximation is known to be equivalent to the algebraic notion [Bür04, Theorem 2.4].

Mulmuley and Sohoni introduced Geometric Complexity Theory (GCT) to settle major open problems in complexity theory [MS01], and initiated the study of border complexity to settle Valiant's conjecture and strengthened it to $\mathsf{VNP} \not\subseteq \overline{\mathsf{VBP}}$. The philosophy of the GCT program is to employ advanced concepts of Algebraic Geometry and Representation Theory to prove strong lower bound results. The increasing popularity of GCT since its inception is due to its strong connection to computational invariant theory [FS13a, Mul12a, GGdOW16, BGdO$^+$18, IQS18], algebraic natural proofs [GKSS17, BIL$^+$21, CKR$^+$20, KRST22], lower bounds [BI13, Gro15a, LO15b], optimization [AGL$^+$18, BFG$^+$19] and many more. We encourage the reader to refer to [BLMW11, Sec. 9] and [Mul12a, Mul17] for expository references.

The algebraic notion of an approximation gives rise to multitudes of natural questions. Understanding the relationship of classical circuit size complexity with its border counterparts is one such question, called De-bordering, that has gained lot of interests in recent years (refer Question 2.2). At the core of it, the problem asks whether the auxiliary variable ε is essential in the approximation, or it can always be eliminated to obtain f effectively.

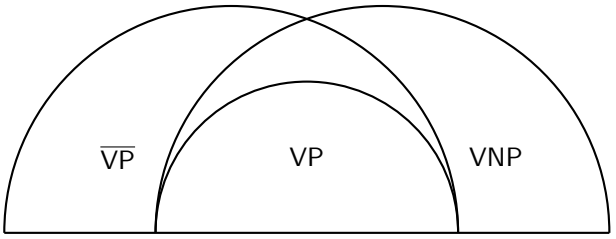


FIGURE 1.5: Containments of Border complexity classes.

1.2.1 Importance of De-bordering

Historically, de-bordering of matrix tensors helped Bini in designing faster matrix multiplication algorithm [Bin80]. In algebraic complexity, the biggest motivation comes from a fundamental conjecture from GCT program [Bür04, Problem 4.3].

Conjecture 1.9 (De-bordering Small Circuits)**.** $\mathsf{VP} = \overline{\mathsf{VP}}$

If the conjecture is proven to be true, then any proof that separates VP from VNP will also prove the Mulmuley-Sohoni conjecture $\mathsf{VNP} \not\subseteq \overline{\mathsf{VP}}$. Moreover, if $\mathsf{VP} \neq \overline{\mathsf{VP}}$, then

any proof of $\mathsf{VP} \neq \mathsf{VNP}$ using continuous lower bounds would have to separate VNP from $\overline{\mathsf{VP}} \setminus \mathsf{VP}$ [Gro15b, BIZ18]. Mulmuley takes it further to motivate de-bordering using its significance in *flip* — an approach to prove stronger lower bounds from the theory and understanding of upper bounds [Mul07, Mul12b].

In its nascent history, border complexity has demonstrated immense potential in helping us understand complexity classes better and thereby motivate us to explore them further. For example, recently it was proved that $\overline{\mathsf{VBP}_2} = \overline{\mathsf{VF}}$ [BIZ18], a relation that does not hold in classical complexity, where it is known that $\mathsf{VBP}_2 \subsetneq \mathsf{VBP}_3 = \mathsf{VF}$ [BC92, AW16]. Furthermore, in a rather surprising result Mrinal Kumar proved that over $\mathbb{C}$, $\mathsf{VF} \subseteq \overline{\Sigma^{[2]}\Pi^{[D]}\Sigma}$ when the degree is inevitably exponential, and proved the universality of a seemingly simple model in the classical setting [Kum20].

It is extremely challenging and non-trivial to de-border classes using known tools and techniques. The problem gained traction from an expository seminar by Michael Forbes, where it was demonstrated that $\overline{\Sigma\wedge\Sigma} \subseteq \mathsf{VBP}$ and conjectured that the class can be completely debordered [For16].

Conjecture 1.10 (De-bordering Waring Rank)**.** $\overline{\Sigma\wedge\Sigma} = \Sigma\wedge\Sigma$

As we will later see in Chapter 2, resolving the above conjecture has far reaching consequences in improving de-bordering results of depth restricted circuits. However, most the known techniques to de-border are tailor made for restricted models and they do not scale to resolve big conjectures.

1.3 Circuit Factoring

Polynomial factorisation is one of the most natural problems in algebraic complexity, which has led to foundational insights in understanding polynomials. Although integer factorisation is believed to be a difficult problem on the classical computational models, various efficient algorithms for its algebraic analogue have been discovered in the last half a century. The problem requires non-trivial ideas even in the dense representation, where the input is given as a list of monomials respecting a fixed ordering. However, we are interested in studying the problem over a more succinct representation of Algebraic Circuits.

Question 1.11 (Factor Closure)**.** *Let $\mathbb{F}$ be a field. Consider a polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ in a circuit class $\mathcal{C}$, and let u be its arbitrary factor. Then show that u is $\mathcal{C}$.*

A complexity class is called *uniformly* factor closed if there is an efficient algorithm to output the factors. The factor of a polynomial could be of a much bigger size complexity than that of the polynomial itself, making it a highly non-trivial problem where each class offers a different challenge. Consider the following example, the number of monomials of $f = \prod_{i\in[n]}(x_i^d - 1) = g \cdot h$ is $s := 2^n$ such that

$$g = \prod_{i=1}^{n}\left(1 + x_i + \cdots + x_i^{d-1}\right), \quad h = \prod_{i=1}^{n}(x_i - 1),$$

meanwhile the number of monomials of g is $s^{\log d}$ [FS15].

In a series of highly influential papers, Kaltofen showed that over fields of characteristic zero, the class VP is closed under factoring [Kal85, Kal86, Kal87, Kal89, KT90]. In fact, if a polynomial factorizes as $f = u^e v$ with u and v co-prime, then Kaltofen [Kal87] showed that size(u) is at most poly$(e, \deg(u), \text{size}(f))$. For the exponential degree polynomial f, one might expect that the size of u depends only on its degree and the size of f, and that the dependence on multiplicity e can be completely removed. Búrgisser formulated it as the Factor Conjecture [Bü00, Conjecture 8.3].

Conjecture 1.12 (Factor Conjecture)**.** *Let $\mathbb{F}$ be a field. Consider a polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$, and let u be its arbitrary factor. Then*

$$\text{size}(u) \leq (\text{size}(f) + \deg(u) + n)^{O(1)}.$$

In other words, we expect that any poly(n)-degree factor of a poly(n)-size circuit (with no restrictions on the degree) is in VP. In case $f = u^e$, Kaltofen [Kal87] showed that it is true and u is VP.

In the last few decades, fascinating progress has been made in restricted algebraic classes. We summarise them in the table below in a natural order of restriction.

There have been many proofs of the original VP closure result itself. See [Bü00, KSS15, Oli16, CKS19a, DSS22] for some alternate ones. From the earlier example, we know that the sparse polynomials ΣΠ are not closed under factoring. However, in a breakthrough work Bhargava, Saraf, and Volkovich studied the problem on sparse polynomials and gave a near polynomial bound on the sparsity, when the individual degree is constant [BSV20a] (see also [vzGK85, Len99, Gre16]). Then a quasi-polynomial time deterministic algorithm was presented to output all constant degree factors of sparse polynomials in the work of Kumar, Ramanathan, and Saptharishi [KRS24]. Lastly, most of the closure results

Table 1.1: Factor Closure Results on Restricted Algebraic Circuits

Model	Main Idea	Ref.
VNP	Hensel Lifting with VNP composition	[CKS19b]
VP	Hensel Lifting	[KT90]
VBP	Hensel Lifting and solving Linear System	[ST21]
Quasi–VF	Newton Iteration (All Roots)	[DSS22]
Constant depth and constant individual degree	Newton Iteration	[Oli20]

discussed in Table 1.1 continue to hold for analogous border classes (see [DSS22, Section 6.1]).

1.3.1 Significance of Factor Closure

An algebraic complexity class which is closed under factorisation is in some sense resilient to nonzero multiplication, which crucially helps to prove lower bound for certain algebraic proof systems [FSTW21].

The proofs and techniques developed in proving factor closure results have found profound applications in various areas of computer science and have gained a lot of independent interest in the past few decades. For example, factorisation is helpful in *Hardness versus Randomness* trade-off which establishes strong connections between lower bounds and deterministic algorithms for problems that have efficient randomised algorithms [KI04, DSY10, AGS19, CKS19b, KST19a, KS19, GKSS22].Other application of it includes coding theory [Sud97, GS99], cryptography [CR88], convex optimisation [Oli20], de-randomisation of the Noether Normalization Lemma [Mul17], circuit reconstruction [Sin16, KS09, BSV20b] and more.

Polynomial factorisation has deep-rooted links to deterministic identity testing, since the two problems are known to be equivalent in a certain sense [SV10, KSS15]. Thus factorisation becomes a helpful tool to prove strong circuit lower bounds [KI04]. We will discuss the problem of identity testing at length in the upcoming section together with its applications.

1.4 Polynomial Identity Testing

On a compact and an abstract model like an algebraic circuit, we can ask several algorithmic questions pertaining to polynomials — adding or multiplying two circuits, evaluating it at certain point, testing equivalence, etc. In a similar spirit, it is natural to ask whether all the coefficients of an input polynomial are zero, formally known as an identically zero polynomial.

Question 1.13 (Polynomial Identity Testing)**.** *Given an algebraic circuit* C *over a field* $\mathbb{F}$ *as input, test if* C *computes an identically zero polynomial.*

Broadly, there are two categories in which the problem is studied — whitebox and blackbox. In the whitebox PIT, we can look inside the circuit and access internal computations of a circuit. A blackbox PIT algorithm decides if the input circuit computes a zero polynomial using only evaluations, and has no access to the internal structure of the circuit. Naturally, from the definitions, blackbox PIT is a stronger notion since it immediately implies whitebox testing. A simple randomised blackbox algorithm for PIT is known for a long time and runs in polynomial time due to the following PIT lemma [Ore22, DL78, Zip79, Sch80].

Lemma 1.14 (PIT Lemma)**.** *Let* $f(\mathbf{x}) \in \mathbb{F}[x_1, \ldots, x_n]$ *be a non-zero polynomial of degree* d*. Then for any set* $S \subseteq \mathbb{F}$*,*

$$\Pr_{a_i \in S}[f(a_1, \ldots, a_n) = 0] \quad \leq \quad \frac{d}{|S|}.$$

The algorithm evaluates the input circuit at a randomly chosen point to test the polynomial. The size of the set S governs the one-sided error probability of the algorithm. It is worthwhile to note that there have been several attempts to improve the lemma [CK00, LV98, KS01, AB03, BHS08, BP20]. However, the most intriguing question is if the algorithm can be de-randomised.

Question 1.15 (Determinisitc PIT)**.** *Design a deterministic algorithm to test whether any* n *variate, degree* d *polynomial computed by a size* s *circuit is identically zero in time* $\mathrm{poly}(n, d, s)$*?*

Lemma 1.14 reveals that if we consider a set $\mathcal{H}$ of size $(d+1)^n$, then with non-zero probability, there exists a point $\mathbf{a} = (a_1, \ldots, a_n) \in \mathcal{H}$ which hits the polynomial, that is $f(\mathbf{a}) \neq 0$. This set is called the *Hitting Set*, which is defined formally as follows.

Definition 1.16 (Hitting Set)**.** *Let $\mathcal{C}$ be a class of n-variate polynomials. A set $\mathcal{H} \subseteq \mathbb{F}^n$ is called a* hitting-set *for $\mathcal{C}$, if for all non-zero $f \in \mathcal{C}$, there exists a non-zeroness certificate point $a \in \mathcal{H}$ such that $f(a) \neq 0$. If the hitting set can be generated in time $T(n)$, then we call it a $T(n)$-time hitting set.*

A hitting set is called *explicit* if it can be efficiently generated by an algorithm. In the blackbox setting, Question 1.15 is equivalent to asking for a $\text{poly}(n, d, s)$ size explicit hitting set [For14, Section 3.2]. Heintz and Schnorr [HS80] showed that a hitting set of size $\text{poly}(n, d, s)$ exists, which was improved in [Mit13, Theorem 2.7.3]. Such existential evidences are promising, but to completely answer Question 1.15 in the blackbox setting, we need an explicit hitting set in time $\text{poly}(n, d, s)$.

For general algebraic circuits, we do not know anything better than an exponential time brute-force algorithm for PIT. Even with years of effort, it was not until recently that the first subexponential time PIT algorithm was obtained for constant-depth circuits via the Hardness vs. Randomness tradeoff [LST21]. Over the past two decades, significant advances have been made in PIT for restricted classes, ranging from bounded-depth circuits to read-once oblivious algebraic branching programs. Refer to Section 6.1 for a comprehensive discussion on the current state of affairs of PIT on models that are relevant to this thesis. For a more broader overview on the topic, refer to extensive surveys filled with exposition of various PIT results [Sax09a, AS09, SY10, Sax14b].

By reducing the problem of checking if a set is not a hitting set to the satisfiability of a set of polynomials, Mulmuley observed that the problem of constructing hitting set for VP is in PSPACE. Assuming the generalised Riemann hypothesis, Koiran brought it down to PH [Koi96]. Refer to [Mul17, Proposition 2.9] for more details. Mulmuley asked if the same is true for the hitting sets of the border class $\overline{\text{VP}}$, and showed that the problem is in EXPSPACE using the Gröbner basis. In a subsequent work, the problem was placed in PSPACE over all fields [FS18, GSS19]. It initiated the quest to construct the hitting set for border classes, which is seemingly a different avenue for exploration.

Definition 1.17. *Let $\overline{\mathcal{C}}$ be a border class of n-variate polynomials. A set $\mathcal{H} \subseteq \mathbb{F}^n$ is called a robust hitting set for $\overline{\mathcal{C}}$, if for all non-zero $f \in \overline{\mathcal{C}}$ there exists a non-zeroness certificate point $a \in \mathcal{H}$ such that $f(a) \neq 0$. In other words, if $g \in \mathcal{C}_{\mathbb{F}(\varepsilon)}$ approximates f, then $g(x = a, \varepsilon) \notin \varepsilon \cdot \mathbb{F}[\varepsilon]$.*

We emphasise that $a \in \mathbb{F}^n$ such that $g(x = a, \varepsilon) \neq 0$, may not hit the polynomial f, since $g(a, \varepsilon)$ might be in $\varepsilon \cdot \mathbb{F}[\varepsilon]$. Intrinsically, this property makes it harder to construct

an explicit hitting set for border classes. The existential nature of the definition of border complexity invalidates a *whitebox* identity test of border classes. Although our newly introduced *presentable* border classes open up a new avenue to study border PIT in the *whitebox* setting as well.

1.4.1 Prominence of Identity Testing

Polynomial Identity Testing has found numerous applications and connections to various algorithmic problems. Some notable examples include algorithms for finding perfect matchings in graphs [Lov79, MVV87, FGT21, GT20, ST17], primality testing [AKS04], polynomial factoring [KSS15, DSS22], polynomial equivalence [DOS14], reconstruction algorithms [KS06, Shp09, KS09] and the existence of algebraic natural proofs [CKR+20, KRST22]. Remarkably, PIT reoccurs naturally in several key results in complexity theory, including $\mathsf{IP} = \mathsf{PSPACE}$ [Sha92], and the PCP theorem [ALM+98, AS98].

Hardness vs Randomness. The theory of using hard objects to derandomise algorithms is extensively studied in complexity theory. In algebraic complexity, similar strong connections are known between PIT and circuit lower bounds. Heintz-Shnorr [HS80] and Agrawal [Agr05] proved that a complete de-randomisation of blackbox PIT algorithm on circuits imply a exponential circuit lower bounds. In the whitebox setting, Kabanets and Impagliazzo [KI04] took inspiration from [NW94] to prove that a subexponential time white-box PIT algorithm for circuits over integers would imply either $\mathsf{VP} \neq \mathsf{VNP}$ (algebraic lower bound) or $\mathsf{NEXP} \not\subseteq \#\mathsf{P}/\mathsf{poly}$ (boolean lower bound).

In the other direction, Kabanets and Impagliazzo showed that a polynomial family which requires super-polynomial circuit size gives a sub-exponential size explicit hitting set. Their framework can derandomise PIT up to quasi-polynomial time. Dvir, Shpilka, and Yehudayoff proved that hard polynomials of low individual degree on constant depth circuits give an efficient hitting set for a constant depth circuit [DSY10, CKS19b]. Then Guo, Kumar, Saptharishi, and Solomon took it further and proved that constant-variate hard polynomials give efficient hitting sets for general circuits and thereby use hardness to completely resolve Question 1.15 [GKSS22, And20]. More recently, Andrew showed that the hardness of matrix multiplication can be used to obtain efficient hitting sets for circuits [And22].

Border PIT. Mulmuley first asked the question of designing a hitting set for $\overline{\mathsf{VP}}$ for its application in Noether's Normalization Lemma (NLL) problem. The paper proved that constructing explicit normalization maps reduces to constructing small hitting sets for

$\overline{\mathsf{VP}}$, which can be solved efficiently with randomness, and deterministically, the problem is in EXPSPACE. Further, it was observed that a certain formulation of NLL is closely connected to proving circuit lower bounds. In fact, derandomizing a special instance of NLL has given a deterministic polynomial time black-box PIT algorithm for a class of restricted depth-4 circuits [Muk16]. Another reason for studying border PIT is to obtain explicit *robust* hitting sets — a stronger notion that resolves the discrepancy between hitting sets of classical and border classes [FS18, MS21].

1.5 Contribution of Thesis

The central theme of this thesis is to study the border complexity of various relevant problems in algebraic complexity. Our model of interest throughout the thesis will be restricted depth-4 circuits. We will now present a summary of the main results of this thesis informally first, and then provide a more detailed discussion in the corresponding chapters.

1.5.1 Demystifying the Border with Explicitness

It is believed that the power harnessed from the auxiliary variable to approximate a polynomial is essential, and removal of it could potentially blow up the size complexity exponentially. Any non-trivial improvement over the exponential degree upper bound of ε proved in the seminal work of Búrgisser [Bür04] would give evidence to refute such a belief. In Section 2.2 we discuss more de-bordering results in detail. However, the universality of the border depth-3 circuit proved by Kumar [Kum20] makes it imperative to investigate constant depth circuits. Until recently, we had little reason to believe if polynomials which can be approximated by $\overline{\Sigma^{[2]}\Pi^{[d]}\Sigma}$ are explicit, that is, in VNP. Using a first-of-its-kind proof, we proved that $\overline{\Sigma^{[k]}\Pi^{[d]}\Sigma} \subseteq \mathsf{VBP}$, for all $k = O(1)$ [DDS22]. We call our iterative de-bordering technique DiDIL (refer Section 2.3).

It is natural to ask whether the technique introduced to de-border depth-3 circuits is robust enough to de-border depth-4 circuits. In Chapter 2 we give a detailed introduction of our novel de-bordering technique DiDIL, followed by applying it to de-border depth-4 circuits $\overline{\Sigma^{[k]}\Pi\Sigma\wedge}$ and prove the following result:

$$\text{For any constant } k \text{ and size } s = \text{poly}(n),\ \overline{\Sigma^{[k]}\Pi\Sigma\wedge}(s) \subseteq \mathsf{VBP}.$$

Although we do not improve the upper bound on the degree of ε, we still manage to make a humble progress towards proving $\overline{\mathsf{VP}} = \mathsf{VP}$.

Main Idea. The novel idea to de-border restricted depth-4 circuits is to use an iterative application of Division, Derivation, and Integration with Limits (DiDIL) to systematically reduce the top fan-in. At a very high level, DiDIL reduces the top Π gate to a $\wedge$ gate, and as a result, it obtains a tractable model that can be de-bordered using known techniques. We give a more formal overview of our proof technique in Section 2.3.

Although the philosophy of de-bordering makes us believe that border classes are perhaps not too much bigger than their classical counterparts, it is far from clear if that is the case from the definitions. The use of arbitrary functions in the auxiliary variable for approximation makes the border circuits highly existential, with no compact representation possible in reality. In fact, we do not know whether polynomials that are easy to approximate are also explicit, that is, $\overline{\mathsf{VP}}$ contained in VNP. To make the concept of an approximation *constructive*, while retaining the core essence, in this thesis, we propose a natural restriction called *presentability*. A presentable border class $\overline{\mathsf{VP}}_\varepsilon$ is the same as the border class $\overline{\mathsf{VP}}$ (Section 1.2), with the additional restriction that all polynomials in ε used in the approximate circuit are of small size (see Definition 3.3 and 5.1). We use this definition to prove the following de-bordering results.

> Over finite fields, the polynomials in presentable border of VNP are explicit.

Since $\mathsf{VP} \subseteq \mathsf{VNP}$, this gives us an interesting tower of containment $\mathsf{VP} \subseteq \overline{\mathsf{VP}}_\varepsilon \subseteq \mathsf{VNP}$. The first of its kind, general debordering at this level, makes the *presentable* border a worthwhile restriction for further investigation. In addition, we generalise Valiant's conjecture and Conjecture 1.9 to a presentable world.

Conjecture 1.18 (Presentable Border Collapse)**.** $\mathsf{VP} = \overline{\mathsf{VP}}_\varepsilon \neq \mathsf{VNP}$

Main Idea. Interpolation, which seemed unhelpful at first, is crucially used to show a structural modification that de-border the presentable model (see Section 2.2). To prove that $\overline{\mathsf{VNP}}_\varepsilon \subseteq \mathsf{VNP}$, instead of using the definition, we employ Valiant's criterion, which essentially states that low-degree polynomials whose coefficients are effectively computable in the boolean world are in VNP (refer to Proposition 1.5). By carefully choosing the evaluation points, interpolation helps obtain the required coefficient function to invoke Valiant's criterion.

1.5.2 Demonstrating Factor Closure

Ever since Kaltofen's closure result of VP [KT90], there was a quest to prove a similar strong closure result for other algebraic complexity classes. In particular, Búrgisser conjectured that explicit polynomials are closed under factorisation [Bü00, Conjecture 2.1]. Chou, Kumar, and Solomon then used their techniques of factor closure of low-degree and depth-restricted circuits to resolve Búrgisser's conjecture f fields of large characteristics [CKS18]. In this thesis, we resolved the conjecture for fields of small characteristic.

> Over any finite field, the class VNP is closed under factorisation.

As a corollary of the above theorem, we find that over finite fields, the factors of polynomials in VP are explicit. This partially resolved the question whether the class is factor closed, over fields of positive characteristic as well [Bü00, Problem 2.1].

Main Idea. The key contribution of this thesis is to handle the *separable* case where the characteristic of the field does not divide the multiplicity. Over finite fields, it is possible to obtain the coefficient function of a factor because of its natural relation to the coefficients of the given polynomial due to simple Frobenius action. Moreover, such a coefficient function will be efficient because the converse of Valiant's criterion is true over finite fields. Although the converse has been remarked in older papers, here we give an independent proof of it by taking inspiration from our earlier proofs. Refer to Chapter 4 for a more detailed proof overview.

Among all the reasons for studying the border complexity we discussed in Section 1.2, its appearance in circuit factoring is perhaps the most organic. In Búrgisser's attempt to resolve Conjecture 1.12 it was proved that any poly(n) degree separable factors of a poly(n)-size circuit are in $\overline{\mathsf{VP}}$. We observe that these separable factors of low degree are, in fact, in the presentable border class $\overline{\mathsf{VP}}_\varepsilon$ (see Section 3.1). Although the conjecture was to show that they are in VP, our presentable de-bordering results prove that they are explicit.

> Over finite fields, poly(n) degree separable factors u of an n-variate polynomial f computable by a circuit of poly(n) size are explicit. That is, u is in VNP.

Main Idea. We first show that Búrgisser in fact proved that u is in $\overline{\mathsf{VP}}_\varepsilon$. Moreover, our presentable de-bordering has proved that over finite fields $\overline{\mathsf{VP}}_\varepsilon \subseteq \mathsf{VNP}$.

1.5.3 Derandomising Polynomial Identity Testing

In an astonishing structural result, Agrawal and Vinay proved that the depth of any algebraic circuit can be squished to a constant with a reasonable increase in the size of the circuit [AV08, Koi12, Tav15].

In particular, they established that the complete derandomization of PIT for depth-4 circuits implies near-complete derandomization of PIT for general circuits. The bootstrapping phenomenon takes the connection even further and expects an efficient PIT algorithm for a very restricted depth-4 circuit to nearly solve Question 1.15 in the blackbox setting [AGS19, KST19b, GKSS22, And20]. The connections have encouraged us for decades to pursue an efficient deterministic algorithm for restricted depth-4 circuits. In this thesis, we will study $\Sigma^{[k]}\Pi\Sigma\wedge$ circuits, defined as follows:

Definition 1.19. *A $\Sigma^{[k]}\Pi\Sigma\wedge$ formula computes a polynomial $f(\mathbf{x}) \in \mathbb{F}[x_1, \ldots, x_n]$ of the form:*

$$f \quad = \quad \sum_{i=1}^{k} \prod_{j} (g_{ij_1}(x_1) + \cdots + g_{ij_n}(x_n)),$$

where $g_{ij_\ell} \in \mathbb{F}[x_\ell]$.

Saha, Saptharishi, and Saxena were the first to initiate the study on the above model and solved the PIT completely on it when $k = 2$ through factoring [SSS13]. The identity testing of $\Sigma^{[k]}\Pi\Sigma\wedge$ can be viewed as a humble generalisation from PIT on bounded-top fan-in depth-3 circuits $\Sigma^{[k]}\Pi^{[d]}\Sigma$ [KS07, SS12] towards PIT on depth-4 circuits. In this thesis, we consider $\Sigma^{[k]}\Pi\Sigma\wedge$ when k is constant and prove the following.

> There is a polynomial-time whitebox PIT algorithm for the $\Sigma^{[k]}\Pi\Sigma\wedge$ formula, when k is constant.

We also give a quasipolynomial time blackbox PIT algorithm for the same model using a different technique [DDS21]. Refer to [Dut22] for more details.

Main Idea. In our pursuit of designing the PIT algorithm, we discover a technique called DiDI. Similar to DiDIL discussed earlier for de-bordering, DiDI is an iterative application of Division and Derivation to reduce the top fan-in. Naturally, these operations distort the model, but with our careful analysis, it can be established that the non-zeroness is preserved in the reduced model. In the end, we show that identity testing on the reduced

model suffices, which is possible using known PIT algorithms. At the core of our identity testing algorithm, the idea is primal: if a bivariate polynomial $g(x, y) \neq 0$, then either its derivative $\partial_y g$ is non-zero, or its constant term $g(x, 0)$ is non-zero.

In the blackbox setting, most of the hitting sets are not known to be robust enough to work for respective border classes. If a class $\mathcal{C}$ has an efficient hitting set $\mathcal{H}$, then naturally showing border closure $\overline{\mathcal{C}} = \mathcal{C}$ immediately proves that $\mathcal{H}$ is a *robust* hitting set. Although known PIT techniques are not known to help, the structural understanding attained from de-bordering circuit classes helped us in derandomising PIT for the same class.

> There exists an explicit quasi-polynomial time robust hitting set for $\overline{\Sigma^{[k]}\Pi\Sigma\wedge}$, when k is constant.

Although we were unable to de-border $\overline{\Sigma^{[k]}\Pi\Sigma\Pi^{[\delta]}}$, our technique and structural understanding scaled well enough for near complete derandomization of PIT for the class.

> There exists an explicit quasi-polynomial time robust hitting set for $\overline{\Sigma^{[k]}\Pi\Sigma\Pi^{[\delta]}}$, when k is constant.

Main Idea. We combine the de-bordering and de-randomisation techniques from earlier discussion to construct an efficient hitting set for $\overline{\Sigma^{[k]}\Pi\Sigma\wedge}$ and $\overline{\Sigma^{[k]}\Pi\Sigma\Pi^{[\delta]}}$. The idea is once again to reduce the PIT problem to a tractable model which is known to have an efficient hitting set. Recall that at a very high level DiDI converts the top product gate to a powering gate. Therefore, all we need in addition to our technique is the hitting set for $\overline{\Sigma\wedge\Sigma\wedge}$ for $\overline{\Sigma^{[k]}\Pi\Sigma\wedge}$ and $\overline{\Sigma\wedge\Sigma\Pi^{[\delta]}}$ for $\overline{\Sigma^{[k]}\Pi\Sigma\Pi^{[\delta]}}$ which are known from earlier results.

1.6 Preliminaries

In this section, we introduce various standard concepts from algebraic complexity theory and summarize a few past results that help to prove our main theorems. Although we do not give the complete proof of all of the stated results, we will do our best to provide relevant references for a comprehensive discussion on them.

Counting and functional complexity classes

We will review some of the computational complexity classes used in our proofs and discuss some standard closure results. For details refer to [Bür04, Section 4.3] and [KP11, Section 2.2]. For a more comprehensive discussion refer to [Pap94]. For a natural number r, $\langle r \rangle \in \{0,1\}^*$ denotes the binary encoding of r.

Definition 1.20 (#P and FP)**.** *The complexity class* #P *is defined as the set of string functions* $\psi : \{0,1\}^* \to \{0,1\}^*$ *such that there is a language* $\chi \in \mathsf{P}$ *satisfying* $\psi(x) = \langle |S| \rangle$ *where*

$$S = \left\{ y \in \{0,1\}^{\mathrm{poly}(|x|)} \ : \ (x,y) \in \chi \right\}.$$

Further, a function ψ *is in* FP *if there exists a Turing machine that computes* $\psi(x)$*, for all inputs* $x \in \{0,1\}^*$*, in time* $\mathrm{poly}(|x|)$*.*

It is easy to show that FP is contained in #P (refer [SK12, Lemma 8]). Further, any counting class can be extended to its corresponding non-uniform version where the functions accept an advice string, in addition to the input string, for computation.

Definition 1.21 (Non-uniform complexity classes)**.** *The complexity class* C/poly *is defined as the set of functions* $\phi : \{0,1\}^* \to \{0,1\}^*$ *such that there exists a* ψ *in class* C *and a polynomial length advice function* $\alpha : \mathbb{N} \to \{0,1\}^*$ *satisfying* $\phi(x) = \psi(x, \alpha(|x|))$*.*

We remark that the advice function α in the definition above only depends on the length of the input. Moreover, for all $n \in \mathbb{N}$, $|\alpha(n)| \leq \mathrm{poly}(n)$. The following lemma shows that the complexity classes of our interest are closed under usual operations.

Lemma 1.22 (Closure Properties)**.** *For a positive integer* r*, consider a set of functions* $\phi_1, \ldots, \phi_r$ *in* #P/poly*. Consider an input string* $x \in \{0,1\}^*$*. Then the following closure properties can be shown:*

1. Addition and Multiplication: *Let* $\phi_+(x) := \sum_{i \in [r]} \phi_i(x)$ *and* $\phi_\times := \prod_{i \in [r]} \phi_i(x)$*. Then,* ϕ_+ *and* $\phi_\times$ *are also in* #P/poly *for* $r \leq \mathrm{poly}(|x|)$*.*
2. Repeated Squaring: *For all* $i \in [r]$*,* $\phi_i(x)^t$ *is in* #P/poly *for* $t \leq 2^{\mathrm{poly}(|x|)}$*.*
3. Projection: *Let* $\Phi_i(x) := \sum_{b \in \{0,1\}^\ell} \phi_i(x,b)$*, where* $\ell \leq \mathrm{poly}(|x|)$*. Then* Φ_i *is in* #P/poly*.*

Proof. For every #P/poly function ϕ_i, let α_i be its advice function and χ_i be its associated language in P defining the counting set S_i, see Theorem 1.20.

1. Addition and Multiplication: Define an advice function $\alpha(|x|, \langle i \rangle) = \alpha_i(|x|)$, and two sets as follows:

$$S_+ := \left\{ (i, y) \in \{0,1\}^{\mathrm{poly}(|x|)+\log r} \ : \ (x, \alpha(|x|, i), y) \in \chi_i \right\}, \text{ and}$$
$$S_\times := \left\{ (y_1, \ldots, y_r) \in \{0,1\}^{r\,\mathrm{poly}(|x|)} \ : \ \forall i \in [r], (x, \alpha(|x|, \langle i \rangle), y_i) \in \chi_i \right\}.$$

   For input $x \in \{0,1\}^*$, let $\bar{\alpha}(|x|) = (\alpha(|x|, \langle 1 \rangle), \ldots, \alpha(|x|, \langle r \rangle))$ be the advice function. Then, it is easy to verify that

$$\phi_+(x) = \psi_+(x, \bar{\alpha}(|x|)) := \langle |S_+| \rangle, \text{ and}$$
$$\phi_\times(x) = \psi_\times(x, \bar{\alpha}(|x|)) := \langle |S_\times| \rangle.$$

   Due to the upper bound on r, the length of the advice string $\bar{\alpha}$ is bounded by $\mathrm{poly}(|x|)$. Moreover, ψ_+ and $\psi_\times$ are in $\#\mathsf{P}$ by definition. Hence, ϕ_+ and $\phi_\times$ are in $\#\mathsf{P}/\mathsf{poly}$.

2. Repeated Squaring: Note that $\phi_i(x)^2$ is in $\#\mathsf{P}/\mathsf{poly}$ from the discussion on multiplication above. Then the claim follows by repeatedly multiplying $\#\mathsf{P}/\mathsf{poly}$ function, $\log r$ many times.

3. Projection: The proof is in the same line as *addition*, which was discussed earlier. Since the advice function depends solely on the length of the input x, it will be same throughout the hypercube-sum. This essentially, lets us add exponentially many $\#\mathsf{P}/\mathsf{poly}$ function. Let $\Psi_i(x, \alpha_i(|x|)) = \langle |S_P| \rangle$ where

$$S_P := \left\{ (b, y) \in \{0,1\}^{\mathrm{poly}(|x|)+\ell} \ : \ (x, \alpha_i(|x|), b, y) \in \chi_i \right\}.$$

   Given the advice string $\alpha_i(|x|)$ as input, clearly Ψ_i is in $\#\mathsf{P}$. Observe that $\Phi_i(x) = \Psi_i(x, \alpha_i(|x|))$, hence Φ_i belongs to $\#\mathsf{P}/\mathsf{poly}$.

□

Structural Results

For a degree-d polynomial f, we denote its degree-k homogeneous components by $\mathrm{Hom}_k(f)$. Similarly, we define $\mathrm{Hom}_{\le k}(f)$ equal to $\sum_{i \in [k]} \mathrm{Hom}_k(f)$.

The following structural theorem proves that, with black-box access to a circuit, it is feasible to construct a circuit that can compute all its homogeneous components.

Refer [SY10, Theorem 2.2] for the proof.

Lemma 1.23 (Homogenisation). *Consider an* n*-variate polynomial* $f := \sum_{i\in[d]} c_i(\mathbf{y})x^i$ *computable by a circuit of size* s *over* $\mathbb{F}$*. Then* $\text{size}(c_i)$ *is at most* $\text{poly}(s, n, d)$*, for all* $i \in [d]$*. Moreover,* $\text{size}(\text{Hom}_{\leq d}(f))$ *is at most* $\text{poly}(s, n, d)$*.*

A straight-forward application of the Homogenisation lemma is the elimination of division gates and taking derivatives of circuits. Refer [SY10, Theorem 2.11] and [DSS22, Lemma 5] for the proof.

Lemma 1.24 (Division Elimination on Circuits). *Consider an* n*-variate polynomial* $f(\mathbf{x}) \in \mathbb{F}[x_1, \ldots, x_n]$ *such that* $\text{size}_{\mathbb{F}}(f) = s$*. Then*

$$\text{size}_{\mathbb{F}}\left(f \mod \langle \mathbf{x}\rangle^{d+1}\right) \leq \text{poly}(s, d).$$

Lemma 1.25 (Derivatives). *Consider an* n*-variate polynomial* $f \in \mathbb{F}[y, x_1, \ldots, x_n]$ *such that* $\text{size}_{\mathbb{F}}(f) = s$*. Then for any* k*,*

$$\text{size}_{\mathbb{F}}\left(\partial_y^k f\right) \leq \text{poly}(s, k).$$

In a similar way, we can eliminate division in algebraic branching programs as well. However, to help us in de-bordering restricted depth-4 circuit we the following expression [Dut22, Lemma 2.6.4].

Lemma 1.26 (Division Elimination on ABPs). *Consider polynomials* $g(\mathbf{x}, y), h(\mathbf{x}, y)$ *of degree at most* d *computable by ABP of size* s *over* $\mathbb{F}$ *such that* $h(\mathbf{x}, 0) \neq 0$*. Then,*

$$\frac{g}{h} \mod y^d \quad = \quad \sum_{i=0}^{d-1} \left(\frac{C_{i1}}{C_{i2}}\right) \cdot y^i,$$

where each C_{ij} *is computable by an ABP of size at most* $O(sd^2)$*. Moreover, if* g/h *is a polynomial then it can be computed by an ABP of size* $O(sd^2)$*.*

Hypercube Sum of Formulas

In Section 1.1 we define the class VNP as hypercube sum of small size circuits. In a subsequent work, Valiant proved that the polynomials in VNP can be equivalently computed by a hypercube-sum of small size formulas [Val82]. Refer [Bü00, Theorem 2.13] and [MP08, Theorem 2] for the proof.

Lemma 1.27 (Verifier formula)**.** *Consider an* n*-variate polynomial* f *of degree* d *computable by a* circuit *of size* s *over* $\mathbb{F}$*. Then, there is a verifier polynomial* h*, with* m *and the* formula *size of* h *both bounded by* $\mathrm{poly}(s, n, d)$*, satisfying the hypercube-sum expression*

$$\sum_{a \in \{0,1\}^{\ell}} h(x_1, \ldots, x_n, a_1, \ldots, a_\ell) = f.$$

The equivalence from the above lemma helps in proving various closure properties of polynomial in VNP. Some of these properties are crucially required in our factoring results in Chapters 4 and 5.

Lemma 1.28 (VNP closure properties)**.** *For all* $i \in [t]$*, let* $f_i \in \mathbb{F}[x_1, \ldots, x_n, y_1, \ldots, y_m]$ *be polynomials in* VNP *over* $\mathbb{F}$*, where* t *is at most* $\mathrm{poly}(n, m)$*. Then the following closure properties hold:*

1. Addition and Multiplication: *Let* $f_+ := \sum_{i \in [t]} f_i$*, and* $f_\times := \prod_{i \in [t]} f_i$*. Then* f_+ *and* $f_\times$ *are in* VNP*.*
2. Coefficient Extraction: *For all* $i \in [t]$*, let* $f_i = \sum_{e} c_e(x) \cdot y^e$*. Then for all exponent vectors* e*, the coefficient* c_e *is also a polynomial in* VNP*.*
3. Composition: *Let* g *be a* t*-variate polynomial in* VNP*. Then* $g(f_1, \ldots, f_t)$ *is in* VNP*.*

Proof. The statements can be proved directly from the Definition of VNP Definition 1.4. For all $i \in [t]$, let $(m_i, \mathrm{size}(h_i))$ be the size parameters for f_i in VNP over $\mathbb{F}$, where both the parameters and $\deg(f_i)$ are bounded by $\mathrm{poly}(n, m)$. Then the properties can be proved as follows.

Addition and Multiplication: Observe that

$$\begin{aligned} f_+ = \sum_{i \in [t]} f_i &= \sum_{i \in [t]} \left(\sum_{a_i \in \{0,1\}^{m_i}} h_i(x, a_i) \right) \\ &= \sum_{(a_1, \ldots, a_t) \in \{0,1\}^{\ell_+}} h_+(x, a_1, \ldots, a_t), \end{aligned}$$

where $\ell_+ := \sum_{i \in [t]} m_i$ and $h_+ := \sum_{i \in [t]} h_i(x, a_i)$. Since both t and m_i, are bounded by $\mathrm{poly}(n, m)$, the length of the witness ℓ_+ is atmost $\mathrm{poly}(n, m)$. Moreover, $\mathrm{size}(h_+) =$

$3 + t + \sum_{i\in[t]} \text{size}(h_i) \le \text{poly}(n, m)$. Similarly for multiplication we have

$$f_\times = \prod_{i\in[t]} f_i = \prod_{i\in[t]} \left(\sum_{a_i\in\{0,1\}^{m_i}} h_i(x, a_i) \right) = \sum_{(a_1,\ldots,a_t)\in\{0,1\}^{\ell_\times}} h_\times(x, a_1, \ldots, a_t).$$

A similar analysis reveals that VNP size parameters $(\ell_\times, \text{size}(h_\times))$ of $f_\times$ are bounded by $\text{poly}(n, m)$.

Coefficient Extraction: The standard interpolation suffices for the proof over large characteristic. Later in Chapter 3 we give a more robust proof to prove one of our main results (see Lemma 3.4).

Composition: We will follow the proof sketch of [CKS19b, Claim 8.4]. Suppose that g is hypercube sum of verifier polynomials v. It is enough to prove the statement for $v \in \mathsf{VP}$. Invoke Lemma 1.27 on the circuit C for the verifier polynomial v to obtain a polynomial h and $\ell \le \text{poly}(t, d)$ satisfying

$$C = \sum_{a\in\{0,1\}^\ell} h(z_1, \ldots, z_t, a_1, \ldots, a_\ell).$$

Let T be the formula computing h of size $\text{poly}(t, d)$. Composing with the VNP polynomials gives

$$C(f_1, \ldots, f_t) = \sum_{a\in\{0,1\}^\ell} h(f_1, \ldots, f_t, a_1, \ldots, a_\ell).$$

We claim that feeding the verifier circuits h_i of the VNP polynomials f_i, into the formula T gives the required hypercube-sum representation.

$$C(f_1, \ldots, f_t) = \sum_{a,a_i\in\{0,1\}^{\ell'}} T(h_1(x, a_i), \ldots, h_t(x, a_t), a),$$

where $\ell' = \ell + \sum_i m_i \le \text{poly}(t, d, n, m)$. Moreover, the size of the circuit computing T composed with $h_1, \ldots, h_t$ is at most $O(\text{size}(T) + \sum_i \text{size}(h_i)) \le \text{poly}(t, d, n, m)$. The correctness of the expression above, follows from an easy induction on the depth of the formula T. Along the layers, from bottom to the top, we repeatedly invoke the additive and multiplicative closure properties which were discussed earlier. Since T is a formula, the verifier circuits for each of the h_i's receive their unique copy of the witnesses and this is preserved throughout the computation. The last part is crucial for the correctness because

simply plugging in the h_i's to the *circuit* C could result in the same witnesses being reused and it may not be the intended computation [2]. □

Properties of Restricted Circuits

Read-once Oblivious ABPs (Definition 1.8) are among the most well-understood models in algebraic complexity. Their properties play a key role in our de-bordering results, which we will discuss next. We start with defining partial derivative matrix which are helpful in characterizing the width of ARO.

Definition 1.29 (Partial Derivative Matrix)**.** *Consider a polynomial* $f(\mathbf{x}) \in \mathbb{F}[x_1, \ldots, x_n]$ *of individual degree* d *and a partition of variables* $\mathbf{x}$ *into two parts* $\mathbf{y} = (y_1, \ldots, y_k)$ *and* $\mathbf{z} = \mathbf{x} \backslash \mathbf{y}$*. The* $n^k \times n^{d-k}$ *matrix* M_f *whose rows are indexed by monomials* p_i *from* $\mathbf{y}$ *and columns are indexed by monomials* q_i *from* $\mathbf{z}$ *is called partial derivative matrix if its* (i, j)*-th entry is* $\mathrm{coef}_{p_i \cdot q_i}(f)$*.*

Partial derivative matrices capture space of $\mathrm{coef}_{\mathbf{y}^a}(f)$ for all $\mathbf{a} \in \{0, \ldots, d\}^k$, which can be expressed as partial derivative $\partial f / \partial \mathbf{y}^{\mathbf{a}}$ evaluated at $\mathbf{y} = 0$. The following two lemmas characterizes the width of an ARO using partial derivative matrix [For14, Lemma 4.5.8].

Lemma 1.30. *Let* $f(\mathbf{x}) \in \mathbb{F}[x_1, \ldots, x_n]$ *be a polynomial of individual degree* d *which can be computed by and ARO of width* w*. For any variable partitioning of variable* $\mathbf{y} = (y_1, \ldots, y_k)$ *and* $\mathbf{z} = \mathbf{x} \backslash \mathbf{y}$*, consider partial derivative matrix* M_f*. Then* $\mathrm{rank}_{\mathbb{F}} M_f \leq w$*.*

Lemma 1.31. *Let* $f(\mathbf{x}) \in \mathbb{F}[x_1, \ldots, x_n]$ *be a polynomial of individual degree* d*. For any variable partitioning of variable* $\mathbf{y} = (y_1, \ldots, y_k)$ *and* $\mathbf{z} = \mathbf{x} \backslash \mathbf{y}$*, consider partial derivative matrix* M_f *such that* $\mathrm{rank}_{\mathbb{F}} M_f \leq w$*. Then* f *is computable by an ARO of width* w*.*

ROABPs are highly restrictive computational model, yet they can capture certain class of restricted constant depth circuits. In particular depth-3 and depth-4 diagonal circuits can be computed by ARO due to the duality trick.

Lemma 1.32 (Duality trick [Sax08])**.** *The polynomial* $f = (x_1 + \ldots + x_n)^d$ *over a field of large characteristic can be written as*

$$f = \sum_{i \in [t]} f_{i1}(x_1) \cdots f_{in}(x_n),$$

where $t = O(nd)$*, and* f_{ij} *is a univariate polynomial of degree at most* d*.*

[2] Consider a pedagogical example, $C(z) = z^2$ from [CKS19b, Footnote 9].

The idea is to convert $\wedge\Sigma\wedge$ into $\Sigma\Pi\Sigma\wedge$ using the duality trick, which in turn is subsumed by ARO [Gur15, Section 2.5.2].

Lemma 1.33 ($\Sigma\wedge\Sigma\wedge$ as ARO)**.** *Let* $f(\mathbf{x}) \in \mathbb{F}[x_1, \ldots, x_n]$ *be computable by* $\Sigma\wedge\Sigma\wedge$ *circuit of size* s *and syntactic degree* D*. Then* f *is computable by an* ARO *of size* $O(sn^2D^2)$*.*

Several properties of the diagonal depth-4 circuit will help us in our de-bordering results. We discuss them below.

Lemma 1.34 (Waring identity for a monomial [CCG12])**.** *Let* $M = x_1^{b_1} \cdots x_k^{b_k}$*, where* $1 \le b_1 \le \cdots \le b_k$*, and* roots of unity $\mathcal{Z}(i) := \{z \in \mathbb{C} : z^{b_i+1} = 1\}$*. Then,*

$$M = \sum_{\varepsilon(i)\in\mathcal{Z}(i):i=2,\cdots,k} \gamma_{\varepsilon(2),\ldots,\varepsilon(k)} \cdot (x_1 + \varepsilon(2)x_2 + \ldots + \varepsilon(k)x_k)^d \ ,$$

where $d := \deg(M) = b_1 + \cdots + b_k$*, and* $\gamma_{\varepsilon(2),\cdots,\varepsilon(k)}$ *are* $\prod_{i=2}^{k}(b_i + 1)$ *many scalars.*

Remark. For fields other than $\mathbb{F} = \mathbb{C}$: We can go to a small extension (at most d^k), for a monomial of degree d, to make sure that $\varepsilon(i)$ exists. We use the above lemma to show that $\Sigma\wedge\Sigma\wedge$ is closed under constant many multiplication.

Lemma 1.35 ($\Sigma\wedge\Sigma\wedge$ closed under multiplication)**.** *Let* $f_i(\mathbf{x}) \in \mathbb{F}[x_1, \ldots, x_n]$ *be a polynomial of syntactic degree at most* d_i *and computed by a* $\Sigma\wedge\Sigma\wedge$ *circuit of size* s_i*, for all* $i \in [k]$*. Then,* $f_1 \cdots f_k$ *has* $\Sigma\wedge\Sigma\wedge$ *circuit of size* $O((d_2 + 1)\cdots(d_k + 1) \cdot s_1 \cdots s_k)$*.*

Proof Sketch. Let $f_i =: \sum_j f_{ij}^{e_{ij}}$; by assumption $e_{ij} \le d_i$. Use Lemma 1.34 to express $f_{ij_1}^{e_{ij_1}} \cdots f_{ij_k}^{e_{ij_k}}$ as $\Sigma\wedge\Sigma\wedge$. □

Using standard interpolation the coefficients of polynomials computable by $\Sigma\wedge\Sigma\wedge$ can be extracted. Moreover, then taking derivatives is easy for $\Sigma\wedge\Sigma\wedge$ as well.

Lemma 1.36 ($\Sigma\wedge\Sigma\wedge$ coefficient extraction)**.** *Let* $f(\mathbf{x}, y) \in \mathbb{F}[x_1, \ldots, x_n][y]$ *be computed by a* $\Sigma\wedge\Sigma\wedge$ *circuit of size* s *and degree* d*. Then,* $\mathrm{coef}_{y^e}(f) \in \mathbb{F}[\mathbf{x}]$ *is computable by a* $\Sigma\wedge\Sigma\wedge$ *circuit of size* $O(sd)$*, over* $\mathbb{F}[\mathbf{x}]$*.*

Proof sketch. Let $f =: \sum_i \alpha_i \cdot f_i^{e_i}$, with $e_i \le s$ and $\deg_y(f) \le d$. Thus, write $f =: \sum_{i=0}^{d} c_i \cdot y^i$, where $c_i \in \mathbb{F}[\mathbf{x}]$. Interpolate using $(d + 1)$-many distinct points, and conclude that f_i has a $\Sigma\wedge\Sigma\wedge$ circuit of size $O(sd)$. □

Lemma 1.37 ($\Sigma\wedge\Sigma\wedge$ differentiation)**.** *Let* $f(\mathbf{x}, y) \in \mathbb{F}[\mathbf{x}][y]$ *be computed by a* $\Sigma\wedge\Sigma\wedge$ *circuit of size* s *and degree* d*. Then,* $\partial_y(f)$ *is a* $\Sigma\wedge\Sigma\wedge$ *circuit of size* $O(sd^2)$*, over* $\mathbb{F}[\mathbf{x}][y]$*.*

Proof sketch. Lemma 1.36 shows that each f_e has $O(sd)$ size circuit where $f =: \sum_e f_e y^e$. Doing this for each $e \in [0, d]$ gives a blowup of $O(sd^2)$ and the representation:

$$\partial_y(f) = \sum_e f_e \cdot e \cdot y^{e-1}.$$

□

1.7 Structure of the thesis

In the previous section we have fixed some notations to follow for the rest of the thesis and discussed some results we need for the proofs. The thesis is divided into three sections — Explicitness of Border Classes, Circuit Factoring, and Identity Testing. The de-bordering of restricted depth-4 circuits is discussed in Chapter 2. Presentable border classes and its explicitness is proved in Chapter 3. The second section of the thesis is dedicated to factoring results. Chapter 4 proves that VNP is closed under factoring over finite fields, and in Chapter 5 we make progress on factor conjecture. In the final section, our thesis gives the identity testing algorithms for restricted depth-4 circuits. Whitebox identity testing algorithm is given in Chapter 6, and robust hitting sets for border classes is constructed in Chapter 7. Finally we close it in Chapter 8 with a few concluding remarks, open problems, and future directions.

Part I

Explicitness of Border Classes

Chapter 2

De-bordering Depth 4 Circuits

An equation for me has no meaning,
unless it represents a thought of God

Srinivasa Ramanujan

Border complexity and polynomial approximation play a pivotal role in Geometric Complexity Theory (GCT) program, which aims to resolve P vs. NP. In addition, it has also found applications in circuit factoring, which we discuss extensively in Section 1.2. It raises a natural question if approximation is essential, or if it can be inexpensively removed — de-bordering. In this chapter, we will de-border restricted depth-4 circuit $\overline{\Sigma^{[k]}\Pi\Sigma\wedge}$. Formally, by the end of this chapter we will prove the following theorem.

Theorem 2.18 (De-bordering $\overline{\Sigma^{[k]}\Pi\Sigma\wedge}$)**.** *A polynomial* $f(\mathbf{x}) \in \mathbb{F}[x_1, \ldots, x_n]$ *approximated by a* $\Sigma^{[k]}\Pi\Sigma\wedge$ *circuit of size* s *can be exactly computed by an ABP over* $\mathbb{F}$ *of size* $s^{O(k \cdot 7^k)}$*. In particular, for any constant* k*,*

$$\overline{\Sigma^{[k]}\Pi\Sigma\wedge}(s) \subseteq \mathsf{VBP}.$$

We call our de-bordering paradigm DiDIL—Divide, Derive, Integrate, with limits. This inductive process described in Section 6.2, in some sense, converts the top Π gate to $\wedge$ gate. We start with some necessary background for the proof.

2.1 Border Complexity Preliminaries

Border (approximative) complexity encapsulates the central theme of this thesis. We begin this chapter by introducing the concept of Border classes and related discussion.

Definition 2.1 (Border Class)**.** *For an algebraic complexity class $\mathcal{C}$, over a field $\mathbb{F}$, we define the border class $\overline{\mathcal{C}}$ as a set of polynomials $f(\mathbf{x}) \in \mathbb{F}[x_1, \ldots, x_n]$ for which there exists a polynomial $g(\mathbf{x}, \varepsilon) := f(\mathbf{x}) + \varepsilon \cdot Q(\mathbf{x}, \varepsilon)$ in class $\mathcal{C}$ over $\mathbb{F}(\varepsilon)$, and $Q \in \mathbb{F}[\varepsilon, x_1, \ldots, x_n]$.*

In the literature, we equivalently say that $f \in \overline{\mathcal{C}}$, the border closure of $\mathcal{C}$, if g belongs to a circuit class $\mathcal{C}$ over $\mathbb{F}(\varepsilon)$. It is important to note that the ε-function used in the computation of g is treated as constants in the circuit and is not considered in $\text{size}_{\mathbb{F}(\varepsilon)}(g)$. When $\mathbb{F} = \mathbb{R}$, under Euclidean topology, we can analytically think of the above approximation as $\lim_{\varepsilon \to 0} g = f$. Although the limit exists, evaluating g at $\varepsilon = 0$ may not be valid because of possible negative powers of ε in the circuit. Hence, given the circuit computing g, we do not have any information about the circuit complexity of the polynomial it is approximating.

Recall that $\text{size}(f)$ denotes the size of the smallest circuit that computes f. In a similar spirit, we use $\overline{\text{size}}(f)$ to denote the border size complexity of f , i.e., $\overline{\text{size}}(f) = \text{size}_{\mathbb{F}(\varepsilon)}(g)$. From the previous discussion, it is evident that $\overline{\text{size}}(f) \leq \text{size}(f)$. A natural question then is to ask what is the upper bound of $\text{size}(f)$ in terms of $\overline{\text{size}}(f)$. The question can be formulated as follows.

Question 2.2 (De-bordering)**.** *Let $\overline{\mathcal{C}}$ be the border class of $\mathcal{C}_{\mathbb{F}}$ as per the Definition 2.1. Find the smallest algebraic class of polynomials $\mathcal{D}$ such that $\overline{\mathcal{C}} \subseteq \mathcal{D}$.*

When $\mathcal{D} = \mathcal{C}$, we say that $\mathcal{C}$ is closed under the border. The importance of de-bordering naturally stems from the efforts to improve understanding of border classes, which was introduced to prove the strengthened separation $\overline{\mathsf{VP}} \neq \mathsf{VNP}$ and resolve Valiant's original conjecture $\mathsf{VP} \neq \mathsf{VNP}$. Although at the current nascent stage of study, we do not have evidence to support the conjecture that $\overline{\mathsf{VP}} \neq \mathsf{VP}$. As we will see in the upcoming section, we have partial and complete de-bordering results for various restricted models. Read more about the importance of de-bordering results in Section 1.2 and [Dut22, Section 6.1]

2.2 Current Status of De-bordering

Although the study of de-bordering classes is at a rudimentary stage, we have seen some interesting de-bordering results for restricted classes in recent years. We discuss some of these results in this section to solidify our understanding of the problem and the challenges of scaling them to general classes.

One interesting property of approximation is that the border distributes over product and division operations. Note that proving a similar result for addition would almost trivialise de-bordering depth-restricted circuits. Refer [Dut22, Lemma 6.2.1] for the proof.

Lemma 2.3 (Distributive Property of Border)**.** *Let $\mathcal{C}$ and $\mathcal{D}$ be class of polynomials in $\mathsf{R}[x_1, \ldots, x_n]$. Then, $\overline{\mathsf{C} \cdot \mathsf{D}} = \overline{\mathsf{C}} \cdot \overline{\mathsf{D}}$ and $\overline{\mathsf{C}/\mathsf{D}} = \overline{\mathsf{C}} \,/\, \overline{\mathsf{D}}$.*

De-bordering using Interpolation. From the previous discussion, we know that the border complexity of a polynomial does not immediately reveal its *exact* or *vanilla* circuit complexity. Suppose g approximates f such that $g(\boldsymbol{x}, \varepsilon) = f(\boldsymbol{x}) + \varepsilon \cdot Q(\boldsymbol{x}, \varepsilon)$. The natural idea is to isolate f via *interpolation* by evaluating g on random values from $\mathbb{F}$ with respect to ε. Given that g is defined over $\mathbb{F}(\varepsilon)$, the random choice of values for ε would ensure the denominator of the ε-function remains non-zero upon evaluation. This simple idea seems hard to execute since, a priori, the degree of ε in g could be arbitrarily large. However, in a foundational work, Búrgisser [Bür04, Theorem 5.7] proved that the degree of ε in g is at most exponential in $\overline{\mathrm{size}}(f)$, and gave the first general de-bordering result where $\mathrm{size}(f) \leq \exp\left(\overline{\mathrm{size}}(f)\right)$.

Improving the degree bound would significantly improve the de-bordering upper bound (refer [Bür04, Problem 4.3] and the discussion afterwards). In that spirit, [BIZ18, Corollary 3.10] considers an alternative measure of approximation called the 'error degree' to de-border polynomial-sized formulas using interpolation. Refer to [BIZ18, Figure 8] for similar results on various algebraic classes and their border closure.

De-bordering in a non-commutative world. To prove strong lower bounds for restricted classes, Nisan [Nis91] characterised the size of the smallest Algebraic Branching Program (Definition Definition 1.6) computing a polynomial in the non-commutative setting, using the rank of the *Partial Derivatives Matrix* (ref Definition 1.29). Michael Forbes [For16] observed that such a rank-based characterisation can de-border without any loss in the size. To see it in action, consider a variant of ABP, which plays a crucial role in the

proofs presented in the upcoming sections: the Read-once Oblivious Algebraic Branching Program in any order (ARO) (see Definition 1.8). Although ARO computation is commutative, their restrictive nature of reading the variables once, in a fixed order, makes them morally similar to ABP computation in a non-commutative setting. Moreover, they exhibit a similar rank-based size characterisation (refer Lemma 1.30 and Lemma 1.31). Using it, Forbes sketched the following de-bordering lemma [For16]. With a slight abuse of notation, let $\mathsf{ARO}(s)$ denote a set of polynomials computable by an ARO of size at most s.

Lemma 2.4 (De-bordering restricted branching programs)**.** $\overline{\mathsf{ARO}(s)} = \mathsf{ARO}(s)$.

Proof. One direction is vacuously true from the definition. For the other direction, consider a polynomial $g(\mathbf{x}, \varepsilon)$ which is computable by an ARO of size s over $\mathbb{F}(\varepsilon)$, and approximates $f(\mathbf{x}) \in \mathbb{F}[x_1, \ldots, x_n]$ such that

$$g := f + \varepsilon Q \tag{2.1}$$

where $Q(\mathbf{x}, \varepsilon) \in \mathbb{F}[\varepsilon][x_1, \ldots, x_n]$. Let M_g be the partial derivative matrix with respect to the variable partition $\mathbf{x} = \mathbf{y} \sqcup \mathbf{z}$. Note that from the right-hand side of Equation (2.1) and the definition of the partial derivative matrix (Definition 1.29), the elements of M_g are coefficients of monomials of g from $\mathbb{F}[\varepsilon]$.

Lemma 1.30 shows that $\operatorname{rank}_{\mathbb{F}(\varepsilon)} M_g \leq w$, where $w \leq s$ is the width of the ARO computing g. Equivalently, the determinant polynomial of any $(w+1) \times (w+1)$ minor of M_g is identically zero. Since M_g has polynomial entries, the determinant of these minors remains zero under the homomorphism $\phi : \mathbb{F}[\varepsilon] \to \mathbb{F}$ such that $\varepsilon \to 0$. Although $\phi(g)$ may be undefined, note that $\phi(M_g) = \phi(M_{f+\varepsilon Q}) = M_f$. Therefore, we can conclude that $\operatorname{rank}_{\mathbb{F}} M_f \leq w$. The converse Lemma 1.31 proves that an ARO of width w over $\mathbb{F}$ exists, which computes f exactly. □

De-bordering in the monotone world. Let the underlying field be real, $\mathbb{F} = \mathbb{R}$. An algebraic circuit, formula, or branching program is called *monotone* if all the constants used in the computation are positive, and hence, computations performed by them cannot feature cancellation. The monotone analogue of $\mathsf{VP}, \mathsf{VBP}$, and VNP is denoted by $\mathsf{mVP}, \mathsf{mVBP}$, and mVNP, respectively. One of the reasons to believe that approximation can help in computing polynomials more efficiently is that there could be possibly heavy cancellation with the help of auxiliary variable ε. In a heavily restricted monotone computation, where cancellations are not permitted, approximation naturally does not help.

In particular, [BIM+20, Theorem 3] and [CL24, Section 3] proved that $\overline{\mathsf{mVP}} = \mathsf{mVP}$, $\overline{\mathsf{mVBP}} = \mathsf{mVBP}$, and $\overline{\mathsf{mVNP}} = \mathsf{mVNP}$. Their de-bordering paradigm is explicitly compared to existential de-bordering for ARO, primarily using structural results that reduce computation iteratively. The proofs are beyond the scope of this thesis. We encourage interested readers to find more details in the referred papers.

De-bordering restricted depth circuits. Though de-bordering polynomially-bounded *circuits* $\overline{\mathsf{VP}}$ seems distant, for the *formulas* class $\overline{\mathsf{VF}}$ it suffices to upper bound the border of width-restricted ABPs, in particular, $\overline{\mathsf{VBP}}_2$ [BIZ18, Corollary 3.9]. Bounded depth circuits are one of the most interesting restrictions on circuits. They are equivalent to formulas with at most polynomial increase in size. Depth-2 circuits $\Sigma\Pi$ and $\Pi\Sigma$ can be explicitly de-bordered easily (refer [BIZ18, Proposition A.12] and [Dut22, Lemma 6.2.2]). Let $\Pi\Sigma\wedge(s)$ denote a class of polynomials computable by $\Pi\Sigma\wedge$ of size at most s.

Lemma 2.5 (De-bordering Product of Sum of Univariates)**.** *A polynomial* $f(\mathbf{x}) \in \mathbb{F}[x_1, \ldots, x_n]$ *approximated by a* $\Pi\Sigma\wedge$ *circuit of size* s *can be completely de-bordered. In particular,*

$$\overline{\Pi\Sigma\wedge}(s) = \Pi\Sigma\wedge(s).$$

Proof Sketch. Use Lemma 2.3 to show that $\overline{\Pi\Sigma\wedge}(s) = \Pi\,\overline{\Sigma\wedge}(s)$. The lemma follows from easy de-bordering of the sum of univariates. □

Restricted depth-3 and depth-4 circuits like $\Sigma\wedge\Sigma$ and $\Sigma\wedge\Sigma\wedge$ (respectively), which are contained in ARO Lemma 1.33, can be partially de-bordered using Lemma 2.4.

Lemma 2.6 (De-bordering Diagonal Circuits)**.** *A polynomial* $f(\mathbf{x}) \in \mathbb{F}[x_1, \ldots, x_n]$ *approximated by* $\Sigma\wedge\Sigma\wedge$ *circuit of size* s *and syntactic degree* D*, can be exactly computed by an* ARO *of size* $O(s\,n^2, D^2)$*. In particular,*

$$\overline{\Sigma\wedge\Sigma\wedge}(s) \subseteq \mathsf{ARO}\left(O(s\,n^2, D^2)\right)$$

Until recently, there was no clarity if the polynomials in the border of general depth-3 circuit $\overline{\Sigma^{[k]}\Pi^{[d]}\Sigma}$ are in VNP even when $k = 2$. The importance of the model stems from Kumar's universality result of depth-4 circuits [Kum20] (Refer [Dut22, Theorem 6.3.1]).

Theorem 2.7 (Universality of Border Depth-3)**.** *Let* $f(\mathbf{x}) \in \mathbb{C}[x_1, \ldots, x_n]$ *be a polynomial of degree* d*. Then* $f \in \overline{\Sigma^{[2]}\Pi^{[D]}\Sigma}$*, where* $D = O\left(\binom{n+d}{d-1}\right)$.

Such a statement is not true in a strong sense for classical depth-3 circuits, making it arguably the most intriguing model to study. The strenuous efforts to de-border depth-3 circuits in [DDS22], gave birth to a novel de-bordering paradigm called DiDIL (Di = Divide, D =Derive, I = Induct, L = Limit). The paradigm proved that the polynomials in $\overline{\Sigma^{[k]}\Pi^{[d]}\Sigma}$ are *explicit*, moreover, they are computable by small ABPs [DDS22, Theorem 1].

Theorem 2.8 (De-bordering Depth-3 Circuit)**.** *For any constant* k*, a polynomial* $f(\mathbf{x}) \in \mathbb{F}[x_1, \ldots, x_n]$ *approximated by a* $\Sigma^{[k]}\Pi\Sigma$ *circuit of size* s*, can be exactly computed by an ABP of size* $\mathrm{poly}(s)$*. In particular,*

$$\overline{\Sigma^{[k]}\Pi^{[d]}\Sigma}(\mathrm{poly}(n)) \subseteq \mathsf{VBP}.$$

In the next section, we will discuss our novel de-bordering paradigm in detail, and prove a similar de-bordering result for depth-4 circuits.

2.3 Gentle Introduction to DiDIL

Most of the known de-bordering techniques are tailor-made for the model of interest, and they do not easily scale to other models. For instance, consider the de-bordering of diagonal circuits $\overline{\Sigma\wedge\Sigma} \to \overline{\mathsf{ARO}} \to \mathsf{ARO}$, which uses duality to transform it into a model which we understand fairly well Lemma 1.33. However, the approach is infeasible on a general depth-3 circuit $\Sigma\Pi\Sigma$ because of possibly exponential blow in top fan-in to transform the Π-gates to $\wedge$-gates. Refer to our paper for a comprehensive discussion on the limitation of known de-bordering techniques [DDS22, Section C]. These impediments in extending the paradigms necessitate a new approach for our model of interest.

Overview We start by giving a high-level working sketch of our paradigm, followed by some concrete discussion, and finally give the complete proof in the next section. DiDIL is a culmination of a four-step inductive process which harnesses the power of *logarithmic derivative* and *power-series* to reduce the model inductively to the one we understand better, specifically to the one we can de-border.

Definition 2.9 (Logarithmic Derivative)**.** *Let* R *be a ring. The logarithmic derivative is a map* $\mathrm{dlog}_y : \mathsf{R}[y] \to \mathsf{R}(y)$ *defined as* $\mathrm{dlog}_y(f) = \partial_y f/f$*, where* $\partial_y f$ *denotes the partial derivative of* f *with respect to* y*.*

While the derivatives will be primarily used to reduce the fan-in, the logarithms will aid us with identities for analysis. For instance, dlog linearises the product of two polynomials: $\mathrm{dlog}_y(f \times g) = \mathrm{dlog}_y(f) + \mathrm{dlog}_y(g)$, and similarly, $\mathrm{dlog}_y(f/g) = \mathrm{dlog}_y(f) - \mathrm{dlog}_y(g)$. We emphasise here that, with some courage, the de-bordering using DiDIL is possible without logarithms. The second tool that we need for de-bordering is the ring of *Formal Power Series*, denoted by $R[\![\mathbf{x}]\!]$ [Sin19, Section 2.2.1]. These rings contain infinite series with non-zero coefficients, such as $f = \sum_{\mathbf{a}\in\mathbb{N}^n} c_{\mathbf{a}}\mathbf{x}^{\mathbf{a}}$. Our paradigm uses this ring for the inverse identity

$$\frac{1}{1-x} = \sum_{i\geq 0} x^i, \tag{2.2}$$

which is unavailable in $R[x]$ (refer [Sin19, Lemma 2.2.2]). While these tools help reduce the top fan-in, they in a way transform the top Π-gates to $\wedge$-gates.

We first demonstrate DiDIL in a pedagogical setup where suppose $g(\mathbf{x}, \varepsilon) \coloneqq T_1 + T_2$, and T_i is a product of linear affine forms with non-zero constant terms. Let g approximate f as per the Definition 2.1. Note that this is the smallest case which seemingly cannot be solved by known techniques. We have deliberately kept the discussion vague and incomplete to a certain extent to convey the overall picture, though in principle the idea is correct. DiDIL heavily distorts the model, so much that we need auxiliary variables to keep track of the degree of the polynomials. We do that by scaling each variable x_i by $z \cdot x_i$. The goal from here is to reduce the fan-in two case to one (single summand). We do that most naturally by first dividing g by T_1 to get: $g/T_1 = 1 + T_2/T_1$ and then taking partial derivative with respect to z to obtain

$$\partial_z\left(\frac{T_2}{T_1}\right) = \partial_z\left(\frac{f}{T_1}\right) + \varepsilon \cdot \partial_z\left(\frac{Q}{T_1}\right). \tag{2.3}$$

Although the two operations reduced the number of terms, the polynomial looks complicated and the computational model is completely distorted. Using the logarithmic derivative identity from Definition 2.9 we write:

$$\partial_z\left(\frac{T_2}{T_1}\right) = \frac{T_2}{T_1}\,\mathrm{dlog}\left(\frac{T_2}{T_1}\right) = \frac{T_2}{T_1}\left(\mathrm{dlog}(T_2) - \mathrm{dlog}(T_1)\right).$$

Since T_i is the product of linear affine forms, the dlog the operator linearises it further. Since we assumed that constant terms of T_i are non-zero, we can use the inverse identity

of a formal power series ring. Let T_i be of the form $(1 - a \cdot z)$, then

$$\frac{1}{1 - a \cdot z} \equiv 1 + a \cdot z + \cdots + a^{d-1} z^{d-1} \mod z^d.$$

The algebraic jugglery proves that the left side of Equation (2.3) can be computed by a small size depth-restricted circuit with powering ($\wedge$) gates. We know from our discussion in Section 2.2, these models can be de-bordered easily using known techniques. We remark here that the derivative is computable by a more general *Bloated* model that we will introduce later.

De-bordering a model has a meaning only when it approximates a polynomial, as in the Definition 2.1. This constitutes an important part of our proof, where we show that when $\partial_z(T_2/T_1)$ is truncated to a high enough power of z, it correctly approximates a polynomial related to f. In particular, there exists a polynomial $t \in \mathbb{F}[x_1, \ldots, x_n]$, such that $\partial_z(T_2/T_1) = \partial_z(f/t) + \varepsilon \cdot Q$. The de-bordering discussion in the previous paragraph reveals the exact circuit complexity of $\partial_z(f/t)$, together with the complexity of the polynomial t, we use interpolation to upper bound the size of f. Moreover, our careful analysis proves that $f \in \mathsf{VBP}$.

The model obtained from a single iteration of DiDIL may appear garbled on the surface, but exhibits two interesting properties. First, it is closed under the DiDIL operations, and in effect, the form of the circuit does not change with the iterative application of operations like division and derivation. For this reason, we define a bloated model for the repeated application of DiDIL.

Definition 2.10 (DiDIL model)**.** *Let* $R(\mathbf{x})$ *be the ring of rational functions. Denote* $\mathsf{Gen}(k, s)$ *as a class of circuits* $\mathcal{C}$ *over* R *which computes polynomial* $f(\mathbf{x}) \in R(x_1, \ldots, x_n)$ *of the form:* $f = \sum_{i=1}^{k} T_i$ *such that*

$$T_i = \left(\frac{U_i}{V_i}\right) \cdot \left(\frac{P_i}{Q_i}\right),$$

where $U_i, V_i \in \Pi\Sigma\wedge$ *and* $P_i, Q_i \in \Sigma\wedge\Sigma\wedge$ *are polynomials in* $R[x_1, \ldots, x_n]$*. The size of the circuit is defined naturally as* $\mathrm{size}(\mathcal{C}) := \sum_{i=1}^{k} \mathrm{size}(T_i) \leq s$ *where*

$$\mathrm{size}(T_i) = \mathrm{size}(U_i) + \mathrm{size}(V_i) + \mathrm{size}(P_i) + \mathrm{size}(Q_i).$$

Further, the syntactic degree of the circuit is defined as the maximum syntactic degree of the numerator and denominator.

The second reason for considering a bloated model for DiDIL is that circuits like $\Sigma^{[k]}\Pi^{[d]}\Sigma$ and $\Sigma^{[k]}\Pi\Sigma\wedge$ of size s are in $\mathsf{Gen}(k,s)$. Let us begin by de-bordering this bloated model in the simplest case. Let R be a commutative ring, which will be suitably fixed later for our proof.

Lemma 2.11 (De-bordering simple DiDIL model)**.** *Consider a polynomial* $f(\mathbf{x}) \in \mathsf{R}[x_1,\dots,x_n]$ *in* $\overline{\mathsf{Gen}(1,s)}$ *of syntactic degree* d*. Then* f *is computable by ratio of two* ABP *of size* $O(s\cdot d^4\cdot n)$ *and syntactic degree* $O(n\cdot d)$*.*

Proof. From Definition 2.10, and invoking Lemma 2.3 we have,

$$f \in \overline{\left(\frac{\Pi\Sigma\wedge}{\Pi\Sigma\wedge}\right)\cdot\left(\frac{\Sigma\wedge\Sigma\wedge}{\Sigma\wedge\Sigma\wedge}\right)} = \left(\frac{\overline{\Pi\Sigma\wedge}\cdot\overline{\Sigma\wedge\Sigma\wedge}}{\overline{\Pi\Sigma\wedge}\cdot\overline{\Sigma\wedge\Sigma\wedge}}\right) \subseteq \frac{\Pi\Sigma\wedge}{\Pi\Sigma\wedge}\cdot\frac{\mathsf{ARO}}{\mathsf{ARO}}.$$

The last containment and the size claim follows from Lemma 2.5 and Lemma 2.6. The syntactic degree claim follows from Lemma 1.33. □

In the lemma above, although we can eliminate the division using Lemma 1.26 to obtain a single ABP computing f exactly, we differ it until the last to avoid the repetitive expensive overhead.

For safe division, we will use the notion of *valuation*. The map will help us along the way to pick the correct summand for division.

Definition 2.12 (Valuation)**.** *Let* R *be a ring. The valuation of a polynomial is a map* $\mathrm{val}_y : \mathsf{R}[y] \to \mathbb{Z}_{\geq 0}$ *defined as the maximum power of* y *which divides* f*. Over fraction field* $\mathsf{R}(y)$*,* $\mathrm{val}_y(p/q) = \mathrm{val}_y(p) - \mathrm{val}_y(q)$*.*

A positive valuation of an element in the fraction field indicates that it belongs to a power series ring. The following property tells you which term to choose for the division. Refer [DDS22, Lemma 2.2.2] for the proof.

Proposition 2.13 (Power Series Indicator)**.** *Let* $f \in \mathsf{R}(x,y)$ *such that* $\mathrm{val}_y(f) > 0$*. Then* $f \in \mathsf{R}[x][\![y]\!] \cap \mathsf{R}(x,y)$*.*

The final ingredient of our paradigm is a homomorphism map, which ensures that terms can be reciprocated, and it aids the proof in derivation.

Definition 2.14 (DiDIL homomorphism)**.** *Pick* $\alpha_1,\dots,\alpha_n$ *uniformly at random from* $\mathbb{F}$*. Define a map*

$$\Phi : \mathbb{F}[\varepsilon][x_1,\dots,x_n] \to \mathbb{F}[\varepsilon][z,x_1,\dots,x_n]$$

such that for all $i \in [n]$, $x_i \mapsto z \cdot x_i + \alpha_i$.

Choosing $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$ randomly from $\mathbb{F}^n$ will suffice to ensure that $\Phi(T_i)(\mathbf{x} = \mathbf{0}) = T_i(\mathbf{x} = \boldsymbol{\alpha}) \neq 0$, which is crucial for safe division and inverse identity.

2.4 Debordering $\Sigma^{[k]}\Pi\Sigma\wedge$ using DiDIL

After familiarising ourselves with the preliminaries of our novel paradigm, we are ready to discuss the proof of our main theorem.

Note that Theorem 2.8 is subsumed in Theorem 2.18. From Definition 2.10 we know that the polynomial f is in $\overline{\mathsf{Gen}(k, s)}$. We start by applying the DiDIL homomorphism from Definition 2.14 on $f_0 := f$. Since the upper bound on f can be obtained by applying Φ^{-1}, we will give the circuit size upper bound on $\Phi(f_o)$.

Lemma 2.15 (First Reduction)**.** *Let* $\Phi(f_0) \in \overline{\Sigma^{[k]}\Pi\Sigma\wedge}(s_0)$ *over* $R_0 := \mathbb{F}[z]/\langle z^d \rangle$*, where* d *is the syntactic degree. Then* $f_1 := \partial_z(\Phi(f_0)/t_{k,0}) \in \overline{\mathsf{Gen}(k-1, s_1)}$ *over* $R_1 := \mathbb{F}[z]/\langle z^{d_1} \rangle$*, where* $t_{k,0} \in R_0(\mathbf{x})$*,* $d_1 \leq d$*, and* $s_1 = O(d^3 \cdot s)$.

Proof. Let $\Phi(f_0)$ be approximated by $\Phi(g_0) := \sum_{i\in[k]} \Phi(T_i) \in \mathsf{Gen}(k, s_0)$ over R_0, as per the Definition 2.1. Define $\nu_i := \mathrm{val}_z(\Phi(T_i))$, using Definition 2.12. Note that from the properties of the DiDIL homomorphism, $\nu_i \geq 0$, for all $i \in [k]$. Assume that $\Phi(T_i) = \varepsilon^{a_i} \cdot \tilde{T}_i$ where $\tilde{T}_i := t_i + \varepsilon \cdot \tilde{t}_i(\mathbf{x}, z, \varepsilon)$ and $t_i = \tilde{T}_i(\mathbf{x}, z, \varepsilon = 0)$. Lastly, without loss of generality, assume that

$$\min_{i\in[k]} \mathrm{val}_z\left(\tilde{T}_i\right) = \nu_k.$$

Then we Divide and Derive as discussed in Section 2.3.

$$\sum_{i=1}^{k} \Phi(T_i) = \Phi(f_0) + \varepsilon \cdot \Phi(S_0) \tag{2.4}$$

$$\varepsilon^{a_k} + \sum_{i=1}^{k-1} \frac{\Phi(T_i)}{\tilde{T}_k} = \frac{\Phi(f_0)}{\tilde{T}_k} + \varepsilon \cdot \frac{\Phi(S_0)}{\tilde{T}_k} \qquad \text{(Divide)}$$

$$\sum_{i=1}^{k-1} \partial_z\left(\frac{\Phi(T_i)}{\tilde{T}_k}\right) = \partial_z\left(\frac{\Phi(f_0)}{\tilde{T}_k}\right) + \varepsilon \cdot \partial_z\left(\frac{\Phi(S_0)}{\tilde{T}_k}\right) \qquad \text{(Derive)}$$

$$\sum_{i=1}^{k-1} \left(\frac{\Phi(T_i)}{\tilde{T}_k}\right) \cdot \mathrm{dlog}\left(\frac{\Phi(T_i)}{\tilde{T}_k}\right) = \partial_z\left(\frac{\Phi(f_0)}{\tilde{T}_k}\right) + \varepsilon \cdot \partial_z\left(\frac{\Phi(S_0)}{\tilde{T}_k}\right)$$

Define $d_1 \coloneqq d - \nu_k - 1$, and let $g_1 \coloneqq \sum_{i\in[k-1]} T_{i,1}$ where for all $i \in [k-1]$

$$T_{i,1} \coloneqq \left(\frac{\Phi(T_i)}{\tilde{T}_k}\right) \cdot \operatorname{dlog}\left(\frac{\Phi(T_i)}{\tilde{T}_k}\right).$$

Recall that $R_1 = \mathbb{F}[z]/\langle z^{d_1}\rangle$. We will prove that g_1 is well-defined in $R_1(\varepsilon, \mathbf{x})$ and approximates $f_1 \coloneqq \partial_z(\Phi(f_0)/t_k)$.

Division by minimum valuation ensures $\operatorname{val}_z(\Phi(T_i) \,/\, \tilde{T}_k \geq 0$. Then, using Proposition 2.13 we infer that

$$\Phi(T_i) \,/\, \tilde{T}_k \in \mathbb{F}(\mathbf{x}, \varepsilon)[\![z]\!] \implies T_{i,1} \in \mathbb{F}(\mathbf{x}, \varepsilon)[\![z]\!].$$

On the other side $\operatorname{val}_z(\Phi(f_0)) \geq \nu_k$ because

$$\operatorname{val}_z\left(\Phi(f_0) + \varepsilon \cdot \Phi(S_0)\right) = \operatorname{val}_z\left(\sum_{i=1}^{k} \Phi(T_i)\right) \geq \nu_k.$$

Once again using Proposition 2.13, we get $\Phi(f_0)/\tilde{T}_k \in \mathbb{F}(\mathbf{x}, \varepsilon)[\![z]\!]$. Therefore, both f_1 and $T_{i,1}$ are well-defined power series. Moreover, note that Equation (2.4) hold over mod z^d. Then division by minimum valuation and derivation implies that the equation holds over mod z^{d_1} where $d_1 = d - \nu_k - 1$. Finally, this shows that f_1 and $T_{i,1}$ are well defined in $R_1(\mathbf{x}, \varepsilon)$. Finally, since $\tilde{T}_k(\mathbf{x}, z, \varepsilon = 0)$ is well-defined, g_1 correctly approximates f_1 over $R_1(\mathbf{x})$ as required.

Size Blow-up. To prove that g_1 is in $\mathsf{Gen}(k-1, \cdot)$, and thereby conclude that $f_1 \in \overline{\mathsf{Gen}(k-1, \cdot)}$, we have to understand the linearising effect of dlog on $(\Phi(T_i) \,/\, \tilde{T}_k)$ (refer discussion after Definition 2.9). Taking the linearisation of the product into account, observe that it suffices to study its effect on $\Sigma\wedge$. From the properties of Φ, we know that such a $\Sigma\wedge$ is of the form $A - z \cdot B$, where $A \in \mathbb{F}(\varepsilon)\backslash\{0\}$ and B is a univariate polynomial in $\mathbb{F}(\varepsilon)[\mathbf{x}, z]$. Using the power series identity (such as Equation (2.2)) we have the following in $R_1(\mathbf{x}, \varepsilon)$:

$$\begin{aligned}
\operatorname{dlog}(A - z \cdot B) &= \frac{-\partial_z(z \cdot B)}{A(1 - z \cdot B \,/\, A)} \\
&= \frac{-\partial_z(z \cdot B)}{A} \cdot \sum_{j=0}^{d_1-1} \left(\frac{z \cdot B}{A}\right)^j \\
&=: C_1 \cdot \sum_{j=0}^{d_1-1} (C_2)^j \qquad (2.5)
\end{aligned}$$

Observe that C_1 and $(C_2)^j$ have trivial $\wedge\Sigma\wedge$ circuits, of size $O(s_0)$ and $O(j \cdot s_0)$ respectively. Using the multiplicative closure of $\Sigma\wedge\Sigma\wedge$ circuit (Lemma 1.35), we obtain the final $\Sigma\wedge\Sigma\wedge$ circuit computing $\mathrm{dlog}(A - z \cdot B)$, of size $s_1 = O(d^3 \cdot s)$ using the fact that $d_1 \leq d$. Clearly, the syntactic degree blows up to $O(d^2)$. □

The *First Reduction* lemma above formally describes the effect of Division on Derivation on the $\Sigma^{[k]}\Pi\Sigma\wedge$, which reduces the problem to a bloated model with reduced fan-in. In the following lemma, we show that the bloated model is closed under these operations and the fan-in continues to reduce with its repeated application.

Lemma 2.16 (Inductive Reduction)**.** *For a positive integer* $j < k$*, let* $f_j \in \overline{\mathsf{Gen}(k-j, s_j)}$ *of syntactic degree* D_j*, over* $R_j := \mathbb{F}[z]\langle z^{d_j}\rangle$*, where* $d_j \leq d$*.*

- *(Valuation) Suppose* $g_j = \sum_i T_{i,j}$ *approximates* f_j*. For all* $i \in [k-j]$ *assume that* $\nu_{i,j} := \mathrm{val}_z(T_{i,j}) \geq 0$*.*
- *(Invertibility) Further, assume that* $U_{i,j}(\mathbf{x}, z = 0, \varepsilon)$ *and* $V_{i,j}(\mathbf{x}, z = 0, \varepsilon) \in \mathbb{F}(\varepsilon)\backslash\{0\}$*.*

Then, $f_{j+1} := \partial_z(f_j/t_{k-j,j}) \in \overline{\mathsf{Gen}(k-j-1, s_{j+1})}$ *over* $R_{j+1} := \mathbb{F}[z]/\langle z^{d_{j+1}}\rangle$*, where* $t_{k-j,j} \in R_j(\mathbf{x})$*,* $d_{j+1} \leq d_j \leq d$*, and* $s_{j+1} = s_j^7 \cdot d^{O(j)}$*. Moreover, the valuation and invertibility properties above continue to hold with respect to* g_{j+1} *approximating* f_{j+1}*.*

Proof. We proceed as earlier with Division and Derivation without applying the DiDIL homomorphism. Note that the assumption on the valuation with respect to z ensures safe operations. Once again, without loss of generality, assume $\min_i \nu_{i,j} = \nu_{k-j,j}$. Let $T_{k-j,j} = \varepsilon^{a_{j-k,j}} \cdot \tilde{T}_{k-j,j}$ where $\tilde{T}_{k-j,j} := t_{k-j,j} + \varepsilon \cdot \tilde{t}_{k-j,j}(\mathbf{x}, z, \varepsilon)$ and $t_{k-j,j} = \tilde{T}_{k-j,j}(\mathbf{x}, z, \varepsilon = 0)$.

$$\sum_{i=1}^{k-j} T_{i,j} = f_j + \varepsilon \cdot S_j \tag{2.6}$$

$$\varepsilon^{a_{k-j,j}} + \sum_{i=1}^{k-j-1} \frac{T_{i,j}}{\tilde{T}_{k-j,j}} = \frac{f_j}{\tilde{T}_{k-j,j}} + \varepsilon \cdot \frac{S_j}{\tilde{T}_{k-j,j}} \qquad \text{(Divide)} \tag{2.7}$$

$$\sum_{i=1}^{k-j-1} \partial_z\left(\frac{T_{i,j}}{\tilde{T}_{k-j,j}}\right) = \partial_z\left(\frac{f_j}{\tilde{T}_{k-j,j}}\right) + \varepsilon \cdot \partial_z\left(\frac{S_j}{\tilde{T}_{k-j,j}}\right) \qquad \text{(Derive)}$$

$$\sum_{i=1}^{k-j-1} \left(\frac{T_{i,j}}{\tilde{T}_{k-j,j}}\right) \cdot \mathrm{dlog}\left(\frac{T_{i,j}}{\tilde{T}_{k-j,j}}\right) = \partial_z\left(\frac{f_j}{\tilde{T}_{k-j,j}}\right) + \varepsilon \cdot \partial_z\left(\frac{S_j}{\tilde{T}_{k-j,j}}\right)$$

Define $d_{j+1} \coloneqq d_j - \nu_{k-j,j} - 1$ and let $g_{j+1} = \sum_{i\in[k-j-1]} T_{i,j+1}$, where for all $i \in [k-j-1]$

$$T_{i,j+1} \coloneqq \left(\frac{T_{i,j}}{\tilde{T}_{k-j,j}}\right) \cdot \operatorname{dlog}\left(\frac{T_{i,j}}{\tilde{T}_{k-j,j}}\right)$$

Valuation $\nu_{i,j+1} \coloneqq \operatorname{val}_z(T_{i,j+1}) \geq 0$ follows trivially. Recall that $R_{j+1} = \mathbb{F}[z]/\langle z^{d_{j+1}}\rangle$. We will prove that g_{j+1} is well-defined in $R_{j+1}(\varepsilon, \boldsymbol{x})$ and approximates $f_{j+1} \coloneqq \partial_z(f_j/t_{k-j,j})$.

Note that f_j and $T_{i,j}$ have non-zero valuation and hence belong to $\mathbb{F}(\boldsymbol{x}, \varepsilon)[\![z]\!]$. Division by minimum valuation, and using arguments similar to Lemma 2.15, we can conclude that f_{j+1} and $T_{i,j+1}$ are elements in $\mathbb{F}(\boldsymbol{x}, z, \varepsilon) \cap \mathbb{F}(\boldsymbol{x}, \varepsilon)[\![z]\!]$ and hence in $R_{j+1}(\boldsymbol{x})$ and $R_{j+1}(\boldsymbol{x}, \mathtt{epsilon})$ (respectively). Since $\tilde{T}_{k-j}(\boldsymbol{x}, z, \varepsilon = 0)$ is well-defined, g_{j+1} correctly approximates f_{j+1} over R_{j+1} as required.

Invertibility. To show that $g_{j+1} \in \mathsf{Gen}(k-j-1, \cdot)$, it remains to show that $\Pi\Sigma\wedge$ circuit in $T_{i,j+1}$ are invertible. Note that this was implicit in Lemma 2.15. Borrowing the notations from Definition 2.10, we simplify $T_{i,j+1}$ as follows:

$$\frac{T_{i,j}}{\tilde{T}_{k-j,j}} = \varepsilon^{-a_{j-k,j}} \cdot \frac{U_{i,j} \cdot V_{k-j,j}}{V_{i,j} \cdot U_{k-j,j}} \cdot \frac{P_{i,j} \cdot Q_{k-j,j}}{Q_{i,j} \cdot P_{k-j,j}}. \tag{2.8}$$

Define $U_{i,j+1} \coloneqq \varepsilon^{-a_{j-k,j}} U_{i,j} \cdot V_{k-j,j}$ and $V_{i,j+1} \coloneqq V_{i,j} \cdot U_{k-j,j}$. Then clearly

$$U_{i,j+1}(\boldsymbol{x}, z = 0, \varepsilon), V_{i,j+1}(\boldsymbol{x}, z = 0, \varepsilon) \in \mathbb{F}(\varepsilon)\backslash\{0\}.$$

The P's and the Q's will be analysed with the action of dlog on Equation (2.2) in the upcoming discussion on the size blow up. We will essentially bring together $\Sigma\wedge\Sigma\wedge/\Sigma\wedge\Sigma\wedge$ to define the final $P_{i,j+1}$ and $Q_{i,j+1}$.

Size Bound. The size analysis is different from Lemma 2.15 for two reasons. First, we don't use the DiDIL homomorphism in the inductive reduction, and second, the analysis of bloated model demands more care. We begin the analysis by considering the overall expression.

$$\begin{aligned} T_{i,j+1} &= \frac{T_{i,j}}{\tilde{T}_{k-j,j}} \cdot \operatorname{dlog}\left(\frac{T_{i,j}}{\tilde{T}_{k-j,j}}\right) \\ &= \underbrace{\frac{U_{i,j+1}}{V_{i,j+1}}}_{(1)} \cdot \underbrace{\left[\underbrace{\frac{P_{i,j}\cdot Q_{k-j,j}}{Q_{i,j}\cdot P_{k-j,j}}}_{(2)} \cdot \underbrace{\operatorname{dlog}\left(\frac{T_{i,j}}{\tilde{T}_{k-j,j}}\right)}_{(3)}\right]}_{(4)} \\ &= \frac{U_{i,j+1}}{V_{i,j+1}} \cdot \frac{P_{i,j+1}}{Q_{i,j+1}} \end{aligned} \tag{2.9}$$

It is easy to note that in the equation above, (1) is computable by $\Pi\Sigma\wedge/\Pi\Sigma\wedge$ with constant blow up in size. Similarly, (2) is computable by

$$\frac{\Pi^{[2]}(\Sigma\wedge\Sigma\wedge)}{\Pi^{[2]}(\Sigma\wedge\Sigma\wedge)} \subseteq \frac{\Sigma\wedge\Sigma\wedge}{\Sigma\wedge\Sigma\wedge}$$

of size $(D_j \cdot s_j^2)$, using the properties (Lemma 1.35). And finally, because of linearisation, $\operatorname{dlog}(\cdot)$ is computable by:

$$\sum \operatorname{dlog}(\Sigma\wedge) \pm \sum^{[4]} \operatorname{dlog}(\Sigma\wedge\Sigma\wedge)$$

Using analysis similar to Equation (2.5) obtain a single $\Sigma\wedge\Sigma\wedge$ circuit of size $O(D_j^2 \cdot d_j \cdot s_j) \leq O(D_j^3 \cdot s_j)$ for the first summand. Since $\Sigma\wedge\Sigma\wedge$ is closed under derivation (Lemma 1.37), $\operatorname{dlog}(\Sigma\wedge\Sigma\wedge)$ is computable by $\Sigma\wedge\Sigma\wedge/\Sigma\wedge\Sigma\wedge$ circuit of size $O(D_j^2 \cdot s_j)$ and syntactic degree $O(D_j)$. Re-indexing the sum, and using additive closure, we obtain a single $\Sigma\wedge\Sigma\wedge/\Sigma\wedge\Sigma\wedge$ circuit of size $O(D_j^12 \cdot s_j^4)$. Lastly adding it to $\Sigma\wedge\Sigma\wedge$ gives the final $\Sigma\wedge\Sigma\wedge/\Sigma\wedge\Sigma\wedge$ for (3) in Equation (2.9) of size $O(D_j^{16} \cdot d \cdot s_j^5)$.

Adding (2) and (3), once again by additive closure, gives $\Sigma\wedge\Sigma\wedge/\Sigma\wedge\Sigma\wedge$ for (4) of size $s_{j+1} \coloneqq O(D_j^{O(1)} \cdot d \cdot s_j^7)$, and syntactic degree $D_{j+1} \coloneqq O(d \cdot D_j)$. □

We will use the two previous lemmas to inductively reduce $\mathsf{Gen}(k, s)$ to $\mathsf{Gen}(1, \cdot)$, which we know how to de-border using Lemma 2.11. We observe that $\mathsf{Gen}(1, \cdot)$ will approximate a derivative polynomial related to f, so we need a way to lift our de-bordering to essentially reconstruct the original polynomial f. We prove the following claim, which borrows the notations from the previous two lemmas.

Claim 2.17 (Integration). *Let* $f_{j+1} = \partial_z(f_j/t_{k-j,j}) \in \mathsf{ABP}/\mathsf{ABP}$ *over* R_{j+1} *of size* S_{j+1} *and syntactic degree* D'_{j+1}. *Then, the expressions on the left of the following table are computable by the respective model.*

	Model	*Size*	*Syntactic Degree*
$\left(\frac{f_j}{t_{k-j,j}}\right)_{z=0}$	$\frac{\mathsf{ABP}}{\mathsf{ABP}}$	$S'_j \coloneqq O(s_j^{O(k-j)} \cdot D_j'^4)$	$O(D_{j+1})$
f_j	$\sum_{i=0}^{d_j-1}\left(\frac{\mathsf{ABP}}{\mathsf{ABP}} \cdot z^i\right)$	$S_j = d_{j+1} \cdot S_{j+1} D_{j+1}'^2 + S'_j$	$D'_j = D'_{j+1} + O(D_{j+1})$

Proof. Recall Equation (2.7):

$$\varepsilon^{a_{k-j,j}} + \sum_{i=1}^{k-j-1} \frac{T_{i,j}}{\tilde{T}_{k-j,j}} = \frac{f_j}{\tilde{T}_{k-j,j}} + \varepsilon \cdot \frac{S_j}{\tilde{T}_{k-j,j}}.$$

The *Valuation* and *Invertibility* of Lemma 2.16, together with Proposition 2.13, gives that the $z = 0$ evaluation of the expression above is computable by

$$\left(\sum_{i=1}^{k-j} \mathbb{F}(\varepsilon) \cdot \frac{\Sigma\wedge\Sigma\wedge}{\Sigma\wedge\Sigma\wedge}\right)_{z=0} = \frac{f_j}{t_{k-j,j}}(\mathbf{x}, \varepsilon, z = 0) + \varepsilon \cdot Q'_{j+1}(\mathbf{x}, \varepsilon, z = 0).$$

Using arguments similar to the proof of Lemma 2.16 we can conclude that the expression above is a well-defined approximation. Then using Lemma 2.6 we conclude that

$$\left(\frac{f_j}{t_{k-j,j}}\right)_{z=0} \in \frac{\mathsf{ABP}}{\mathsf{ABP}}.$$

For the size of the ABP, observe that we take the sum of $(k-j)$-many $\Sigma\wedge\Sigma\wedge/\Sigma\wedge\Sigma\wedge$, each of size s_j (Lemma 1.35), followed by conversion to ARO (Lemma 1.33).

For the second part of the proof, use division elimination from Lemma 1.26, to express $\mathsf{ABP}/\mathsf{ABP}$ computing f_{j+1} as follows:

$$f_{j+1} = \sum_{i=0}^{d_{j+1}-1} C_{i,j+1} \cdot z^i,$$

where $C_{i,j+1} \in \mathsf{ABP}/\mathsf{ABP}$ of size $O\left(S_{j+1} \cdot D'^2_{j+1}\right)$. Definite-Integration implies:

$$\frac{f_j}{t_{k-j,j}} = \left(\frac{f_j}{t_{k-j,j}}\right)_{z=0} + \sum_{i=1}^{d_{j+1}} \left(\frac{C_{i,j+1}}{i}\right) \cdot z^i \in \sum_{i=0}^{d_{j+1}} \left(\frac{\mathsf{ABP}}{\mathsf{ABP}}\right) \cdot z^i. \tag{2.10}$$

Moreover, recall that

$$t_{k-j,j} \in \overline{\left(\frac{\Pi\Sigma\wedge}{\Pi\Sigma\wedge}\right) \cdot \left(\frac{\Sigma\wedge\Sigma\wedge}{\Sigma\wedge\Sigma\wedge}\right)} \subseteq \frac{\mathsf{ABP}}{\mathsf{ABP}}$$

of size s_j and degree D_j. Since the $\mathrm{val}_z(t_{k-j,j}) \geq \nu_{k-j,j} = d_j - d_{j+1} - 1$, we multiply $t_{k-j,j}$ in Equation (2.10) and truncate it at $d_j - 1$ to compute f_j to required precision. For the size blow up, note that

$$S_j = d_{j+1} \cdot S_{j+1} D'^2_{j+1} + S'_j,$$

moreover, the degree will be $D'_j = D'_{j+1} + O(D_{j+1})$. □

We bring all the pieces together to prove the main theorem of the chapter.

Theorem 2.18 (De-bordering $\overline{\Sigma^{[k]}\Pi\Sigma\wedge}$)**.** *A polynomial $f(\mathbf{x}) \in \mathbb{F}[x_1, \ldots, x_n]$ approximated by a $\Sigma^{[k]}\Pi\Sigma\wedge$ circuit of size s can be exactly computed by an ABP over $\mathbb{F}$ of size $s^{O(k \cdot 7^k)}$. In particular, for any constant k,*

$$\overline{\Sigma^{[k]}\Pi\Sigma\wedge}(s) \subseteq \mathsf{VBP}.$$

Proof. When $k = 1$, Lemma 2.11 de-borders and finishes the proof. Assume $f_0 = f \in \overline{\mathsf{Gen}(k, s_0)}$ where $s_0 = s$. We start by applying DiDIL map from Definition 2.14, and prove upper bound on $\Phi(f_0)$. Lemma 2.15, followed by Lemma 2.16 gives a sequence of polynomials $f_1, f_2, \ldots, f_{k-1}$ such that they are approximated by $g_i \in \mathsf{Gen}(k-i, s_i)$, for all $i \in [k-1]$. Further, we bound the syntactic degree $D_{k-1} = O(d \cdot D_{k-2}) = d^{O(d)}$, and then the size

$$s_{k-1} = O(d^{O(k-1)} \cdot s_{k-2}^7) \leq s^{O(k \cdot 7^k)}.$$

That implies $f_{k-1} \in \overline{\mathsf{Gen}(1, s_{k-1})} \subseteq \mathsf{ABP}/\mathsf{ABP}$ using Lemma 2.11 of size

$$O(s^{O(k \cdot 7^k)} \cdot D_{k-1}^4 \cdot n) \leq s^{O(k \cdot 7^k)},$$

and syntactic degree $n \cdot D_{k-1}^2 \leq d^{O(d)}$. Now we will Integrate to lift this quantity to recover $f_{k-2}, f_{k-3}, \cdots, f_0$ using Claim 2.17.

Note that we do not have to do division elimination in each step of integration after doing it the first time. The ABP/ABP coefficient in the computation of f_j are z-free, hence do not incur a size blow up due to integration. The main blow-up is in the computation of $z = 0$ part in the proof of Claim 2.17. First observe that $D'_j = D'_{j+1} + O(D_{j+1}) = d^{O(j)}$. Then using the recurrence for S'_j we obtain the upper bound:

$$S'_j = s^{O(k-j)\cdot j\cdot 7^j} \leq s^{O(k\cdot 7^k)},$$

because $\max_{j\in[k-1]} j\cdot(k-j)\cdot 7^j = (k-1)\cdot 7^{k-1}$ by differentiating and computing maxima.

Let $S_{k-1} = s_{k-1}$ be the size of ratio of ABP computing f_{k-1} then f_0 is computable by ABP/ABP of size $S_j = d_{j+1}\cdot S_{j+1}D'^2_{j+1} + S'_j \leq s^{O(k\cdot 7^k)}$. Use the degree bound on z to eliminate the division to obtain a single ABP computing f_0 of size $s^{O(k\cdot 7^k)}$. Apply Φ^{-1} to obtain the required ABP which *exactly* computes the polynomial f. □

Chapter 3

De-bordering Presentable Border Classes

In mathematics you don't understand things. You just get used to them.

John von Neumann

The non-explicit nature of the definition of border complexity makes de-bordering results strenuous, and it is one of the central reasons for the lack of de-bordering results on unrestricted algebraic circuits. In this chapter, we will introduce a natural restriction on approximation which makes the de-bordering a tractable pursuit. We begin with station an alternative but equivalent definition of approximation (refer Definition 2.1 as well).

Definition 3.1 (Approximation)**.** *A polynomial* $f(\mathbf{x}) \in \mathbb{F}[x_1, \ldots, x_n]$ *is approximated by a polynomial* $g(\mathbf{x}, \varepsilon) \in \mathbb{F}[\varepsilon][x_1, \ldots, x_n]$ *to an* order of approximation M *if*

$$g(\mathbf{x}, \varepsilon) = \varepsilon^M f(\mathbf{x}) + \varepsilon^{M+1} Q(\mathbf{x}, \varepsilon),$$

for some $Q(\mathbf{x}, \varepsilon) \in \mathbb{F}[\varepsilon][x_1, \ldots, x_n]$*. The* border size *of* f *denoted* $\overline{\mathsf{size}}_M(f)$*, is defined as* $\mathrm{size}_{\mathbb{F}[\varepsilon]}(g)$*, the size of the polynomial* g *over the ring* $\mathbb{F}[\varepsilon]$*.*

Note that analytically the approximation gives $\lim_{\varepsilon \to 0} \varepsilon^{-M} g(\mathbf{x}, \varepsilon) = f(\mathbf{x})$. Furthermore, arbitrary polynomials in ε are treated as 'free constants' in the circuit of g. It is not hard to see via scaling arguments ($g' := \varepsilon^{-M} g$) that the notions is equivalent to Definition 2.1. For a discussion of the different notions of approximation and their equivalence, see [Bür04, Lemma 5.6], [BIZ18, Section 2] and also [Mum76, Theorem 2.33].

Proposition 3.2. *Consider a polynomial* $f(\mathbf{x}) \in \mathbb{F}[x_1, \ldots, x_n]$. *Then,* $\overline{\text{size}}(f) = \Theta\left(\overline{\text{size}}_M(f)\right)$.

Hence, for the rest of the chapter we can safely drop the subscript M from $\overline{\text{size}}_M(f)$, or use them interchangeably in the subsequent chapters.

3.1 Presentable Border and its Efficacy

In the definition of approximation, we allow arbitrary polynomials in ε, of arbitrary complexity to be used as free constants. This arbitrariness makes the definition of approximation inherently *existential* and intractable for de-bordering. Read more on this in Section 2.2. As a way of making approximation more constructive, while retaining its essence, in this thesis we propose and study a natural restriction on the definition of approximation, called *presentability*, that restricts the complexity of ε polynomials used for approximation.

The *presentable* class $\overline{\mathsf{VP}}_\varepsilon$ is the same as $\overline{\mathsf{VP}}$ but with the additional condition that all the polynomials in ε used as 'constants' in the approximating circuit $g(\mathbf{x}, \varepsilon)$, have polynomial-size circuits themselves (see Definition 5.1). We can extend our concept of presentable border to $\overline{\mathsf{VNP}}_\varepsilon$ over any field $\mathbb{F}$.

Definition 3.3 (Presentable $\overline{\mathsf{VNP}}$)**.** *The presentable border class* $\overline{\mathsf{VNP}}_\varepsilon$*, over* $\mathbb{F}$*, is defined as the set of polynomials* $f \in \mathbb{F}[x_1, \ldots, x_n]$ *such that there is an approximating polynomial* $g \in \mathbb{F}[\varepsilon][x_1, \ldots, x_n]$ *expressing*

$$g(\mathbf{x}, \varepsilon) \;=:\; \varepsilon^M f(\mathbf{x}) + \varepsilon^{M+1} Q(\mathbf{x}, \varepsilon)\,,$$

for some error $Q \in \mathbb{F}[\varepsilon][x_1, \ldots, x_n]$ *and* order $M \in \mathbb{N}$*; moreover, there exists a* verifier *polynomial* $h \in \mathbb{F}[x_1, \ldots, x_n, y_1, \ldots, y_m, \varepsilon]$*, with* $m, \deg_{\mathbf{x},\mathbf{y}}(h)$ *and* $\text{size}_{\mathbb{F}}(h)$ *all bounded by* $\text{poly}(n)$*, satisfying a* hypercube-sum *expression*

$$\sum_{\mathbf{a} \in \{0,1\}^m} h(\mathbf{x}, \mathbf{a}, \varepsilon) \;=\; g(\mathbf{x}, \varepsilon)\,.$$

The pair $(m, \text{size}_{\mathbb{F}}(h))$ constitutes the size parameters for the polynomial family $f = f_n$ in $\overline{\mathsf{VNP}}_\varepsilon$. Crucially, although the bound on $\text{size}_{\mathbb{F}}(h)$ (instead of $\text{size}_{\mathbb{F}[\varepsilon]}(h)$) constrains the ε-polynomials to have small circuits, we do not restrict the degree of ε, which could be exponential in $\text{size}_{\mathbb{F}}(h)$. This makes this new class potentially harder than VNP. It is easy to see that $\mathsf{VNP} \subseteq \overline{\mathsf{VNP}}_\varepsilon \subseteq \overline{\mathsf{VNP}}$.

Since, presentable border classes allow for an exponential degree in ε, moving to $\overline{\mathsf{VP}}_\varepsilon$ and $\overline{\mathsf{VNP}}_\varepsilon$ does not lead to any ε-degree loss. Most of the de-bordering and separation results discussed in Section 2.2 are based on characterizations and properties of restricted classes that are not known for general classes such as $\overline{\mathsf{VP}}_\varepsilon$ and $\overline{\mathsf{VNP}}_\varepsilon$. Surprisingly, although interpolation seemed unhelpful on first glance, we show that a structural modification does indeed help in de-bordering when we move to presentable border classes.

Theorem 3.6 (Presentable is Explicit)**.** *Over* any *finite field,* $\overline{\mathsf{VNP}}_\varepsilon = \mathsf{VNP}$.

In the remaining section we briefly discuss the proof of the main theorem of the chapter, while discussing the obvious challenges which we can overcome due to the properties of *presentibility* of approximation.

Over finite fields, we will extract the coefficient of $\varepsilon^M \boldsymbol{x}^{\boldsymbol{e}}$ in approximating polynomial g by carefully choosing the interpolation points to be roots of unity, whose (multiplicative) *order* is 'only' exponential. Consequently, we show that the coefficient $c_{\boldsymbol{e}}$ can be obtained as a hypercube sum of an *exponential degree* algebraic circuit of *polynomial size* (Lemma 3.4) We enumerate two tricky issues that are handled in the proof.

1. It would not be possible to control the size of this extraction circuit (over the underlying field $\mathbb{F}_q$) if we were to use the usual definition of $\overline{\mathsf{VNP}}$, mainly because the ε-constants might truly require exponential *size* circuits. Working with $\overline{\mathsf{VNP}}_\varepsilon$ lets us keep the circuit size small while retaining the exponentially large degree of ε.

2. The choice of interpolation points must be careful; otherwise, just to write down the interpolation formula, we would need to invert an exponentially large matrix of *generic* constants, which would again require circuits of exponential size. In addition, we need the various points to eventually map to a suitable hypercube $\{0,1\}^\ell$, which places further constraints on the design of the points.

   We solve these problems by using the properties of finite fields that allow us to transfer to a much better-behaved Boolean computation model. In particular, we use a multiplicative generator ω of an exponentially large field $\mathbb{F}_{q'}$ to realize the hypercube points.

Using finite field arithmetic and the closure of the Boolean class $\#\mathsf{P}$ under exponential sums, we move from the algebraic world to the Boolean one (Lemma 3.5). Thus, we show that the algebraic circuit above (from Lemma 3.4) can be simulated by a (multi-output) Boolean circuit of polynomial size; furthermore, the hypercube sum computing

the coefficient function is demonstrated in $\#\mathsf{P}/\operatorname{poly}$. Valiant's criterion (Proposition 1.5) now implies that the polynomial f is indeed in VNP.

3.2 Presentable is Explicit

In this section we will prove that polynomials in $\overline{\mathsf{VNP}}_\varepsilon$ are explicit over finite fields. We will begin by stating two essential lemmas of our paper which will help us in designing *effective* coefficient functions of large degree polynomials. The following lemma shows that the polynomials computable by the hypercube-sum of small sized circuits are 'closed' under coefficient extraction, i.e. there is a similar algebraic expression for each coefficient. This is like interpolation, but as the degree and number of monomials is exponential, we desire to achieve an algebraic expression that is well structured.

Lemma 3.4 (Exponential interpolation)**.** *Let* $s \coloneqq \operatorname{poly}(r, \log q)$ *and let* $g = \sum_{\boldsymbol{e}} c_{\boldsymbol{e}} \boldsymbol{y}^{\boldsymbol{e}}$ *be an* r*-variate polynomial over* $\mathbb{F}_q$ *of degree* $D \coloneqq \exp(s)$ *such that* $g = \sum_{\boldsymbol{a} \in \{0,1\}^m} h(\boldsymbol{y}, \boldsymbol{a})$ *for some polynomial* h *with* $m, \operatorname{size}(h) \leq s$*.*

Then, taking $\boldsymbol{e}$ *as input there exists a polynomial* $t_{\boldsymbol{e}}$ *over a finite field extension* $\mathbb{F}_{q'}$*,* $q' \leq \operatorname{poly}(D)$*, such that the coefficient* $c_{\boldsymbol{e}} = \sum_{\boldsymbol{b} \in \{0,1\}^\ell} t_{\boldsymbol{e}}(b_1, \ldots, b_\ell)$ *, where* ℓ *and* $\operatorname{size}(t_{\boldsymbol{e}})$ *are at most* $\operatorname{poly}(s)$*.*

We will prove the above lemma in Section 3.2.1. In the subsequent lemma we show that the resulting hypercube sum above can be converted into a boolean function in $\#\mathsf{P}/\mathsf{poly}$. The two lemmas together build up the correct setup to invoke Valiant's criterion. Recall $s = \operatorname{poly}(r, \log q)$.

Lemma 3.5 (Algebraic to boolean complexity)**.** *For any exponent vector* $\boldsymbol{e} \in \{0, \ldots, D\}^r$*, let the coefficient of* $\boldsymbol{y}^{\boldsymbol{e}}$ *in* $g \in \mathbb{F}_q[y_1, \ldots, y_r]$*, denoted by* $c_{\boldsymbol{e}}$*, be computable by a polynomial* $t_{\boldsymbol{e}}$ *over a finite field extension* $\mathbb{F}_{q'}$*,* $q' \leq \operatorname{poly}(D) \leq 2^{O(s)}$*, as follows:*

$$c_{\boldsymbol{e}} = \sum_{\boldsymbol{b} \in \{0,1\}^\ell} t_{\boldsymbol{e}}(b_1, \ldots, b_\ell) \,, \tag{3.1}$$

where ℓ *and* $\operatorname{size}(t_{\boldsymbol{e}})$ *are at most* $\operatorname{poly}(s)$*. Then, with* s *as the input-size parameter, there exists a function* ϕ_g *in* $\#\mathsf{P}_p/\mathsf{poly}$ *that computes* $\phi_g(\langle \boldsymbol{e} \rangle) = \langle c_{\boldsymbol{e}} \rangle$*.*

We will defer the proof of the lemma until Section 3.2.2. Meanwhile, we will use the technical lemmas to give the complete proof of our first main result.

Theorem 3.6 (Presentable is Explicit)**.** *Over* any *finite field,* $\overline{\mathsf{VNP}}_\varepsilon = \mathsf{VNP}$.

Proof. Consider a polynomial (family) $f = f_n \in \mathbb{F}_q[x_1, \ldots, x_n]$ in $\overline{\mathsf{VNP}}_\varepsilon$ of degree d, which is approximated by $g \in \mathbb{F}_q[\varepsilon, x_1, \ldots, x_n]$ as per Definition 3.3. Let the $\overline{\mathsf{VNP}}_\varepsilon$ size parameters of g be (s, s), where $s := \mathrm{poly}(n)$ and $d := \deg_{\boldsymbol{x}}(g) \leq \mathrm{poly}(s)$. The size of the verifier circuit h from Definition 3.3 is bounded by s, hence the degree $D := \deg_\varepsilon(h) \leq 2^s$ (as, w.l.o.g., h has multiplication-fanin two).

Using Lemma 3.4 on g, followed by applying Lemma 3.5, gives a $\#\mathsf{P}/\mathsf{poly}$ function ϕ_g which computes the encoding of coefficients of g. The coefficient of a monomial $\boldsymbol{x}^{\boldsymbol{e}}$ in f is the coefficient of $\varepsilon^M \cdot \boldsymbol{x}^{\boldsymbol{e}}$ in the approximating polynomial g. Observe that if

$$f = \sum_{\boldsymbol{e} \in \{0,\ldots,d\}^n} c_{\boldsymbol{e}} \cdot \boldsymbol{x}^{\boldsymbol{e}}, \tag{3.2}$$

then $(c_{\boldsymbol{e}}) = \phi_g(M, e_1, \ldots, e_n)$. From the definition of $\overline{\mathsf{VNP}}_\varepsilon$, we know that $d, \log(M) \leq \mathrm{poly}(n)$. So, using Valiant's criterion (Proposition 1.5) we conclude that f is in VNP. □

3.2.1 Exponential interpolation technique

In this section we will give the proof of Lemma 3.4. We will show that the coefficients of the polynomial g from the lemma statement can be expressed as a hypercube sum of evaluation of small size circuits. Recall the size parameter $s = \mathrm{poly}(r + m, \log q)$ and $q =: p^a$ for prime p. We will induct on the number of variables r.

Consider a positive integer k such that $2^s = D < k < \Theta(D)$, and a primitive root of unity ω of order k. We know that $\omega \in \mathbb{F}_q$ if and only if k divides $q - 1$ (refer [vzGG13, Lemma 8.8]). Moreover, if $\mathbb{F}_q$ does not contain the particular primitive root of unity, we can obtain them in the multiplicative group of its finite field extension $\mathbb{F}_{q'}$, where $k < q' := p^{a'} = \Theta(D)$. Interested readers are encouraged to read more details in standard literature on Finite Fields, for instance refer to [vzGG13, Chapter 8] and [Sho09, Exercise 17.24]. For the rest of the section we will assume for simplicity that $\omega \in \mathbb{F}_q$; as an identical proof works over the extension $\mathbb{F}_{q'}$. Note that $1/k \in \mathbb{F}_q$, as $k|(q-1)$ implies that $p \nmid k$.

Base case. Suppose g is a univariate polynomial in $y = y_1$ and consider an exponent $e \leq k$. To extract the coefficient c_e in g, we will interpolate by evaluating g on a set of k distinct points $\{\omega^0, \omega^1, \ldots, \omega^{k-1}\}$, constituting all the powers of this primitive root

of unity. These evaluations of g form a linear system using Vandermonde matrix $V_\omega := \left(\omega^{ij}\right)_{0\le i,j<k}$ as follows:

$$\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{k-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{k-1} & \omega^{2(k-1)} & \cdots & \omega^{(k-1)^2} \end{pmatrix}_{k\times k} \begin{pmatrix} \vdots \\ c_e \\ \vdots \end{pmatrix}_{k\times 1} = \begin{pmatrix} \vdots \\ g\left(\omega^e\right) \\ \vdots \end{pmatrix}_{k\times 1} .$$

Vandermonde matrices are invertible if and only if its entries are all distinct. It is clear that ω^{-1} is also a primitive root of unity; moreover, the inverse matrix $(V_\omega)^{-1} = (1/k)\cdot V_{(\omega^{-1})}$ (refer [vzGG13, Theorem 8.13]). Therefore, we can express the required coefficient with the following equation:

$$\begin{aligned} c_e &= \sum_{j=0}^{k-1} \frac{\omega^{-ej}}{k} \cdot g\left(\omega^j\right) \\ &= \sum_{a\in\{0,1\}^m} \left(\sum_{j=0}^{k-1} \frac{\omega^{-ej}}{k} \cdot h\left(\omega^j, a\right) \right). \end{aligned} \tag{3.3}$$

A circuit that computes the inner sum in Equation (3.3) trivially, would be exponentially large in s because $k = \Theta(D)$. However, we can write this as a hypercube-sum by carefully encoding the powers of ω in a single polynomial using binary representation of the exponent. This encoding will design a verifier circuit, with a relatively small increase in the witness size. Let $\mathrm{wt}(k) := \lceil \log_2 k \rceil$ and use it to define a polynomial $\bar{h} \in \mathbb{F}_q[z, z_1, \ldots, z_{\mathrm{wt}(k)}]$ as follows:

$$\bar{h} := \prod_{i=1}^{\mathrm{wt}(k)} \left(z_i \cdot z^{2^{i-1}} + (1 - z_i)\cdot 1 \right) . \tag{3.4}$$

Let $\mathbf{j} := (j_1, \ldots, j_{\mathrm{wt}(k)})$ be the binary representation of j, then it is easy to verify that $\bar{h}(\omega, \mathbf{j}) = \omega^j$. Together with $\bar{h}$, Equation (3.3) can be re-written as follows:

$$\begin{aligned} c_e &= \sum_{a\in\{0,1\}^m} \sum_{\mathbf{j}\in\{0,1\}^{\mathrm{wt}(k)}} \frac{1}{k} \cdot \bar{h}(\bar{h}\left(\omega^{-1}, \langle e\rangle\right), \mathbf{j}) \cdot h\left(\bar{h}(\omega, \mathbf{j}), a\right) \\ &=: \sum_{a,\mathbf{j}\in\{0,1\}^\ell} t_e(a, \mathbf{j}) , \end{aligned}$$

where $\ell := m + \mathrm{wt}(k) \leq O(s)$. Observe that $\mathrm{size}\left(\bar{h}\right) \leq O(\mathrm{wt}(k)) \leq O(s)$, moreover, composition and multiplication have additive blow-up on size of the circuit. Since, $\mathrm{size}(h)$ was bounded by s, overall gluing the circuits together shows that $\mathrm{size}(t_e) \leq O(s)$.

Induction step. Let us assume that the lemma holds for all such $r-1$ variate polynomials. Now, suppose g is a r-variate polynomial in $\mathbb{F}_q[y_2, \ldots, y_r][y_1]$ such that

$$g = \sum_{i \leq D} g_i(y_2, \ldots, y_n) \cdot y_1^i,$$

where g_i is $(r-1)$-variate polynomial of degree at most D. With respect to the fixed exponent vector $\boldsymbol{e} = (e_1, e_2, \ldots, e_r) \in \mathbb{N}^r$, define $\boldsymbol{e}^- := (e_2, \ldots, e_r) \in \mathbb{N}^{r-1}$. From the equation above, observe that computing the coefficient $c_{\boldsymbol{e}}$ of $y_1^{e_1} y_2^{e_2} \cdots y_r^{e_r}$ in g is equivalent to computing the coefficient $c_{\boldsymbol{e}^-}$ of $y_2^{e_2} \cdots y_r^{e_r}$ in g_{e_1}. To invoke the induction hypothesis on g_{e_1}, we first need to show that, like g, it can be explicitly expressed as a hypercube-sum of a small sized circuit.

Once again interpolate on g to obtain the coefficient of $y_1^{e_1}$. Similar to the *base case*, begin by considering the evaluations of g on the set of powers $\{\omega^0, \omega^1, \ldots, \omega^{k-1}\}$. The equivalent linear system obtained using the Vandermonde matrix V_ω is as follows:

$$\begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{k-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{k-1} & \omega^{2(k-1)} & \cdots & \omega^{(k-1)^2} \end{pmatrix}_{k \times k} \begin{pmatrix} \vdots \\ g_{e_1} \\ \vdots \end{pmatrix}_{k \times 1} = \begin{pmatrix} \vdots \\ g\left(\omega^{e_1}, \boldsymbol{y}\right) \\ \vdots \end{pmatrix}_{k \times 1}.$$

As argued earlier, the matrix is invertible; more importantly, its elements are easily obtained from $(V_\omega)^{-1} = (1/k) \cdot V_{(\omega^{-1})}$. This results in the following expression for the $(r-1)$-variate coefficient polynomial:

$$\begin{aligned} g_{e_1} &= \sum_{j=0}^{k-1} \frac{\omega^{-e_1 j}}{k} \cdot g\left(\omega^j, y_2, \ldots, y_r\right) \\ &= \sum_{\boldsymbol{a} \in \{0,1\}^m} \left(\sum_{j=0}^{k-1} \frac{\omega^{-e_1 j}}{k} \cdot h\left(\omega^j, y_2, \ldots, y_r, \boldsymbol{a}\right) \right). \end{aligned} \tag{3.5}$$

To show that the inner summation has small size circuit, we encode the powers of root of unity using the polynomial $\bar{h}$ defined in Equation (3.4). All together, it gives the following compact expression:

$$\begin{aligned} g_{e_1} &= \sum_{a\in\{0,1\}^m} \sum_{j\in\{0,1\}^{wt(k)}} \frac{1}{k} \cdot \bar{h}(\bar{h}(\omega^{-1}, \langle e_1\rangle), j) \cdot h\left(\bar{h}(\omega, j), y_2, \ldots, y_r, a\right) \\ &=: \sum_{a,j\in\{0,1\}^\ell} h_{e_1}(y_2, \ldots, y_r, a, j)\ , \end{aligned} \tag{3.6}$$

where $\ell := m + wt(k) \le O(s)$. Further, $size(h_{e_1}) \le s + 2 \times size(\bar{h}) \le O(s)$.

Size analysis. We analyse the size of the verifier-circuit of c_e by unfolding the induction layers. Since g_{e_1} is now a $(r-1)$-variate polynomial, using induction hypothesis we get that there is a polynomial t_e, that computes the relevant coefficient c_{e^-} as follows:

$$c_{e^-} = \sum_{b\in\{0,1\}^\ell} t_e(b_1, \ldots, b_\ell)\ ,$$

whence we define $s(r) := size(t_e)$. Building on the insights from Equation (3.6), observe that in each iteration of the interpolation, the verifier is only evaluated and multiplied by the polynomial $\bar{h}$. This stems from the nature of interpolation, which extracts the coefficients as a linear combination of polynomial evaluations. So, we get a simple recurrence: $s(r) \le s(r-1) + 2 \cdot size(\bar{h})$, which implies that the final verifier-circuit size $s(r) \le O(rs)$. Analogously, the witness length increases by $wt(k)$ in each iteration, hence $\ell(r) \le m + r \cdot wt(k) \le O(rs)$. That concludes the proof of Lemma 3.4. □

3.2.2 Transfer algebraic complexity to boolean

In this section, we will show that the hypercube-sum of the evaluations of a small-size circuit can be transformed into a #P/poly function, which will prove Lemma 3.5. As described earlier, the proof goes via *booleanisation* of the algebraic circuit. Recall that $q = p^a$, and for a field element $b \in \mathbb{F}_q$, $\langle b\rangle \in \{0,1\}^s$ denotes the binary encoding of b. For a point $\mathbf{b} \in \mathbb{F}_q^\ell$, denote $\langle \mathbf{b}\rangle := (\langle b_1\rangle, \ldots, \langle b_\ell\rangle) \in \{0,1\}^{\ell s}$.

Claim 3.7 (Booleanisation)**.** *Consider a polynomial* $t \in \mathbb{F}_q[y_1, \ldots, y_\ell]$ *such that* $size(t) \le s$. *There exists an equivalent (multi-output) boolean circuit* T *of* $bitsize \le s \cdot poly(\log q)$, *such that for all inputs* $\mathbf{b} \in \mathbb{F}_q^\ell$ *we have* $T(\langle \mathbf{b}\rangle) = \langle t(\mathbf{b})\rangle$.

Proof. Let C be an algebraic circuit of size at most s which computes the polynomial t. Without loss of generality. we assume that the circuit has fan-in two. The idea is to build a Boolean circuit from the Algebraic circuit by replacing each of its field operation gates with equivalent Boolean gadgets. Following is a formal proof of it using induction on the depth of C.

In the base case, we have variables and constants at the input level. To construct the equivalent Boolean circuit T, split every input variable y_i into $\log q$ many gates which takes $\langle b_i \rangle$ as input. Similarly, every constant β in $\mathbb{F}_q$ can be split into $\log q$ many gates based on $\langle \beta \rangle$. Therefore $\text{bitsize}(T) \leq O(s \cdot \log q)$.

Let C_1, C_2 be sub-circuits of C, connected to an internal node C_{12}. From the induction hypothesis, there are equivalent Boolean circuits T_1, T_2 of bitsize at most $s \cdot \text{poly}(\log q)$ such that for all inputs $\mathbf{b} \in \mathbb{F}_q^\ell$ we get $T_i(\langle \mathbf{b} \rangle) = \langle C_i(\mathbf{b}) \rangle$, for $i \in [2]$. Arithmetic operations in a finite field, for instance, addition and multiplication, can be efficiently simulated by Boolean circuits (that have input and output as binary strings). In particular, there are $\text{poly}(\log q)$ size Boolean circuits T_+ and $T_\times$ such that for all $b_1, b_2 \in \mathbb{F}_q$, $\langle b_1 + b_2 \rangle = T_+(\langle b_1 \rangle, \langle b_2 \rangle)$ and $\langle b_1 \times b_2 \rangle = T_\times(\langle b_1 \rangle, \langle b_2 \rangle)$[1]. For a detailed discussion on computational complexity of finite field arithmetic refer [GS11, Section 2] and [AB09, Section A.4].

Based on the gate C_{12}, use either T_+ or $T_\times$ with T_1 and T_2 as inputs to obtain the circuit T_{12} such that for all inputs $\mathbf{b} \in \mathbb{F}_q^\ell$ we have $T_{12}(\langle \mathbf{b} \rangle) = \langle C_{12}(\mathbf{b}) \rangle$. Notice that $\text{bitsize}(T_{12}) = \text{bitsize}(T_1) + \text{bitsize}(T_2) + \max(\text{bitsize}(T_+), \text{bitsize}(T_\times))$. Proceeding this way in a level-by-level fashion, we obtain the complete Boolean circuit T which computes $\langle t(\mathbf{b}) \rangle$. Finally, for the bitsize claim we observe that every gate is replaced by either T_+ or $T_\times$ and thus $\text{bitsize}(T) \leq s \cdot \max(\text{bitsize}(T_+), \text{bitsize}(T_\times)) \leq s \cdot \text{poly}(\log q)$. □

We will use the above claim to convert the algebraic circuit in the hypercube sum of the coefficient into an efficiently computable Boolean function. Recall the hypercube-sum expression for coefficients from Lemma 3.5:

$$c_e = \sum_{\mathbf{b} \in \{0,1\}^\ell} t_e(b_1, \ldots, b_\ell)$$

where ℓ and $\text{size}(t_e)$ are at most $\text{poly}(s)$. Since c_e and $t_e(\mathbf{b})$ are elements of $\mathbb{F}_q$, their binary representation is an encoding of tuple of $\mathbb{F}_p$ elements. Refer the remark following Proposition 1.5.

[1] The Boolean encoding and the output of Boolean circuit are compared coordinate-wise.

Lemma 3.5 (restated.) *For any exponent vector $\boldsymbol{e} \in \{0,\ldots,D\}^r$, let the coefficient of $\mathbf{y}^{\boldsymbol{e}}$ in $g \in \mathbb{F}_q[y_1,\ldots,y_r]$, denoted by $c_{\boldsymbol{e}}$, be computable by a polynomial $t_{\boldsymbol{e}}$ over a finite field extension $\mathbb{F}_{q'}$, $q' \leq \mathrm{poly}(D) \leq 2^{O(s)}$, as follows:*

$$c_{\boldsymbol{e}} = \sum_{\mathbf{b}\in\{0,1\}^{\ell}} t_{\boldsymbol{e}}(b_1,\ldots,b_{\ell}) \,,$$

where ℓ and $\mathrm{size}(t_{\boldsymbol{e}})$ are at most $\mathrm{poly}(s)$. Then, with s as the input-size parameter, there exists a function ϕ_g in $\#\mathsf{P}_p/\mathsf{poly}$ that computes $\phi_g(\langle \boldsymbol{e}\rangle) = \langle c_{\boldsymbol{e}}\rangle$.

Proof. Consider the $\mathbb{F}_p$-basis representation of $\mathbb{F}_q$ element $t_{\boldsymbol{e}}(\mathbf{b}) = \sum_{i<a} t_{\boldsymbol{e},i}\alpha^i$, where $t_{\boldsymbol{e},i} \in \mathbb{F}_p$. Apply Claim 3.7 to the algebraic circuit that computes $t_{\boldsymbol{e}}$ to obtain a multi-output equivalent Boolean circuit $T_{\boldsymbol{e}}$ satisfying $\langle t_{\boldsymbol{e}}(\mathbf{b})\rangle = T_{\boldsymbol{e}}(\langle \mathbf{b}\rangle)$, for all $\mathbf{b} \in \{0,1\}^{\ell}$. The Boolean circuit $T_{\boldsymbol{e}}$ computes the encoding of $\mathbb{F}_q$ element as a tuple $(\langle t_{\boldsymbol{e},0}\rangle, \ldots, \langle t_{\boldsymbol{e},a-1}\rangle)$. Let $T_{\boldsymbol{e},i}$ denote a sub-circuit of $T_{\boldsymbol{e}}$ computing the string $\langle t_{\boldsymbol{e},i}\rangle$.

We claim that $T_{\boldsymbol{e},i}$ is in the complexity class $\mathsf{FP}/\mathsf{poly}$ (refer Section 1.6 for definitions). Define a Turing Machine M that takes $\langle T_{\boldsymbol{e}}\rangle$ as advice, and evaluates $T_{\boldsymbol{e}}$ at the input $\langle \mathbf{b}\rangle$ in time $\mathrm{poly}(s)$, for any $\mathbf{b} \in \{0,1\}^{\ell}$. The size of the advice string $\langle T_{\boldsymbol{e}}\rangle$ is independent of the input and depends only on the input length $\ell \leq \mathrm{poly}(s)$. Finally, the Turing machine outputs the i-th block of the evaluation. Clearly, the function computed by M is in FP, and hence $T_{\boldsymbol{e},i}$ is in $\mathsf{FP}/\mathsf{poly}$.

Let the $\mathbb{F}_p$-basis representation of the coefficient be $c_{\boldsymbol{e}} = \sum_{i<a} c_{\boldsymbol{e},i}\alpha^i$, where $c_{\boldsymbol{e},i} \in \mathbb{F}_p$. To design the coefficient function ϕ_g that computes the encoding of $c_{\boldsymbol{e}}$, it suffices to prove that there is a function $\phi_{g,i}(\langle \boldsymbol{e}\rangle)$ in $\#\mathsf{P}_p/\mathsf{poly}$ that computes $\langle c_{\boldsymbol{e},i}\rangle$, for all $i < a$. From Equation (3.1), we see that $c_{\boldsymbol{e},i} = \sum_{\mathbf{b}\in\{0,1\}^{\ell}} t_{\boldsymbol{e},i}$, where the sum is over $\mathbb{F}_p$. Therefore, we can express $\langle c_{\boldsymbol{e},i}\rangle$ as a hypercube sum of $\langle t_{\boldsymbol{e},i}\rangle$ reduced modulo p, and thus, also as a hypercube-sum of $T_{\boldsymbol{e},i}(\langle \mathbf{b}\rangle)$, modulo prime p.

Recall that $\mathsf{FP} \subseteq \#\mathsf{P}$ (Definition 1.20). Now, invoke Lemma 1.22(3) to obtain a $\#\mathsf{P}_p/\mathsf{poly}$ function that computes the hypercube-sum. □

Part II

Circuit Factoring

Chapter 4

Factor Closure of VNP over Finite Fields

Somewhere, something incredible is waiting to be known.

Carl Sagan

It is a natural question to ask if an algebraic complexity class is closed under factorization. Over any field $\mathbb{F}$, consider a polynomial family $f_n \in \mathsf{VNP}$, and its arbitrary factor g_n. Can we say that $g_n \in \mathsf{VNP}$. Over fields of characteristic zero, Chou, Kumar and Solomon [CKS19b] showed that this is indeed true.

Theorem 4.1. *Let $\mathbb{F}$ be a field of* zero *characteristic. Consider a polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ in the class* VNP *and let u be its arbitrary factor. Then, we have u in* VNP*.*

Chou, Kumar and Solomon resolved a long-standing conjecture by Búrgisser [Bü00, Conjecture 2.1] over fields of large characteristic.Inspired by the proof technique of Theorem 3.6, we prove that VNP closure under factoring holds over finite fields as well, thus settling Bürgisser's conjecture in an important regime.

Theorem 4.2 (Factor closure)**.** *Over* any *finite field, the class* VNP *is closed under factorization.*

Remark 4.3. As a corollary of the above theorem, we find that over finite fields, the factors of polynomials in VP are in VNP. This partially answers the question [Bü00, Problem 2.1]

whether VP is closed under taking factors over fields of positive characteristic. Recall that over fields of characteristic zero, we already know this to be true from the works of Kaltofen; but those methods fail in finite fields.

Overview. The two classical paradigms involved in factoring multivariate polynomials are *Hensel lifting* and *Newton iteration* (see, e.g. [vzG84, vzGG13]), which have historical origins in complex analysis. A crucial step in the proof of Theorem 4.1, which involves approximating a root of a polynomial to increasingly higher precision using Newton iteration, fails to work over finite fields (a more important case in computer science applications). To prove that the class VNP is closed under factoring over fields of positive characteristic p, we reduce the problem to two cases. Let f be a polynomial in VNP. Following [CKS19a], we have one of the following:

1. The polynomial $f = u^e$ is a power of a factor u.
2. The polynomial $f = u \cdot v$ is a product of co-prime polynomials u and v.

We would like to show that the factor u is in VNP in both cases. The proof of Case 2 (Lemma 4.5) uses slight modifications of standard techniques developed over the years [Kal87, KSS15, CKS19b]. We first transform the polynomial so that it is monic and bivariate. We start the Hensel lifting process with two coprime univariate factors and lift them to high enough precision (with respect to a degree measure). We use a version of the lift that automatically gives us the factors at the end. To finally show that the factor we obtain is in VNP, we use a one-shot analysis as in [CKS19b].

Over fields of characteristic zero, it can be shown that proving Case 2 is sufficient (see proof of [CKS19a, Lemma 1.3]). However, in a finite field $\mathbb{F}_q$, this reduction only works if the characteristic p of the field does not divide the exponent e (we can call this the *separable* case). Our main contribution is showing that if $f = u^{p^k}$ for some $k \geq 1$, then u is in VNP (Lemma 4.4). Using this result, we can then handle all powers (Lemma 4.6).

All previous known techniques fail in the case where the exponent e is a prime power. Inspired by the proof of Theorem 3.6, we take a completely different approach. Consider the simple case where $f = u^p$. The coefficients of u and coefficients of f are related by a simple Frobenius action. It turns out that Valiant's criterion (Proposition 1.5) for a polynomial being in VNP also has a converse (Lemma 4.7). It was remarked in [MP08, Section 6] that the fact has been observed before in [Pé04], though we could not find a

written reference [1]. We give an independent proof for finite fields in this paper by first noting that any coefficient of a VNP polynomial can be obtained as a hypercube-sum of evaluations of a VP circuit. As in the proof of Theorem 3.6 we similarly convert the algebraic expression thus obtained to a Boolean #P/poly circuit.

Since $f \in \mathsf{VNP}$, the inverse of Valiant's criterion gives us that its coefficient function is in #P/poly. We obtain the coefficients of u by performing an *inverse* Frobenius transformation, which we demonstrate in #P/poly. Finally, using Valiant's criterion in the forward direction, we see that the factor u is in VNP.

4.1 VNP is factor closed

Meanwhile we state the three technical lemmas that help us main result, specifically for the case of polynomial factoring in small characteristic fields. The first lemma is our main contribution that handles the 'pure' inseparable case of factoring.

Lemma 4.4 (Prime power)**.** *Let $f \in \mathbb{F}_q[x_1, \ldots, x_n]$ be a polynomial in* VNP*. If there is a polynomial u and a positive integer i such that $f = u^{p^i}$, then the factor u is in* VNP*.*

Lemma 4.5 (Coprime factors)**.** *Let $f \in \mathbb{F}_q[x_1, \ldots, x_n]$ be a polynomial in* VNP*. If there are co-prime polynomials u and v such that $f = u \cdot v$, then the factor u is in* VNP*.*

We defer the proof of the above fundamental lemmas to the subsequent two subsections. For now, we use them to prove an essential lemma that deals with the 'radical' computation in VNP.

Lemma 4.6 (Any power)**.** *Let $f \in \mathbb{F}_q[x_1, \ldots, x_n]$ be a polynomial in* VNP*. If there is a polynomial u and an arbitrary positive integer e such that $f = u^e$, then the factor u is in* VNP*.*

Proof. Let $e := p^i \cdot \widehat{e}$, and $u_1 := u^{p^i}$, such that p does not divide $\widehat{e}$. Note that, when $\widehat{e} = 1$ then Lemma 4.4 finishes the proof. When $\widehat{e} > 1$, we associate a polynomial $\widehat{f}$ with a new

[1] Perifel communicated to us a proof that over $\mathbb{Q}$, the coefficients of constant-free VNP families (see [Mal03]) are in GapP/poly.

variable z as follows:

$$\begin{aligned}\widehat{f} &:= z^{\widehat{e}} - f = z^{\widehat{e}} - u_1^{\widehat{e}} \\ &= (z - u_1) \cdot \left(z^{\widehat{e}-1} + z^{\widehat{e}-2} u_1 + \cdots + u_1^{\widehat{e}-1} \right) \\ &=: u_2(z) \cdot u_3(z) \,.\end{aligned}$$

For contradiction sake, assume that u_2 and u_3 share a factor, and hence are not co-prime. This implies that u_1 must be a root of u_3, which gives $u_3(u_1) = \widehat{e} \cdot u_1^{\widehat{e}-1} = 0$. However, since $\widehat{e} > 1$ and u_1 is non-zero, it follows that the characteristic p divides $\widehat{e}$, which contradicts our choice of $\widehat{e}$.

Observe that $z^{\widehat{e}}$ is trivially in VNP, hence we obtain that $\widehat{f}$ is in VNP. Since u_2 and u_3 are co-prime, we invoke Lemma 4.5 to shows that u_2 is in VNP, and therefore u_1 is in VNP. We finish the proof by using Lemma 4.4 on u_1 to finally prove that u is in VNP. □

With all the essential ingredients in place, we are now ready to prove the second main result of our paper. We will restate Theorem 4.2 formally, which proves the closure of VNP under factoring over all fields.

Theorem 4.2 (Formally restated). *Let $\mathbb{F}$ be a field of any characteristic. Consider a polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ in the class VNP and let u be its arbitrary factor. Then, we have u in VNP.*

Proof. Over fields of characteristic zero, [CKS19b, Theorem 2.8] proved that u is in VNP. Here we consider the hitherto unsolved case of small prime characteristic. In particular, when $\mathbb{F} = \mathbb{F}_q$, where $q =: p^a$ for some prime $p < \deg(f)$.

Pick the largest integer $e \geq 1$ and the polynomial $v \in \mathbb{F}_q[x_1, \ldots, x_n]$ satisfying $f =: u^e v$. If $v = 1$, then Lemma 4.6 proves that u is in VNP.

If u and v are coprime, then we conclude the proof using Lemma 4.5 and Lemma 4.6.

In the last case, there exists an irreducible polynomial $w \in \mathbb{F}_q[x_1, \ldots, x_n]$ that divides both u and v. Consider $u_1 := w^{e'}$ and $v_1 := (f/u_1)$ such that u_1, v_1 are coprime factors of f. Again, using Lemma 4.5 and Lemma 4.6 we get that w is in VNP. Repeat this for all the irreducible factors of u, and use the fact that VNP is closed under multiplication (Lemma 1.28); this concludes the proof of u being in VNP. □

4.1.1 Factoring prime powers using Valiant's converse

To prove Lemma 4.4, we show that the coefficients of the factor polynomial u can be computed effectively, and thus use Valiant's criterion to prove the claim. We will argue that coefficients of u can be obtained from the coefficient function of f. Therefore, it would suffice to design an effectively computable coefficient function for f, give that it is in VNP. To that effect, we prove the *converse* of Valiant's criterion, over finite fields.

Lemma 4.7 (Converse of Valiant's criterion)**.** *Let $f = \sum_e c_e \cdot x^e$ be a polynomial in* VNP *over $\mathbb{F}_q$. Then, there exists a function ϕ_f in $\#\mathsf{P}_p/\mathsf{poly}$ such that for all e, $\phi_f(\langle e \rangle) = \langle c_e \rangle$.*

Proof. Let $D := \deg(f)$ and the VNP size parameters of f be (s, s) where $s := \mathrm{poly}(n, \log q)$. Using the exponential-interpolation in Lemma 3.4, with $D = \mathrm{poly}(s)$, we can prove that each coefficient c_e of f is a hypercube-sum of small-circuit evaluations, with parameters $(\mathrm{poly}(s), \mathrm{poly}(s))$ [2]. That is, there is a polynomial t_e over a finite field extension $\mathbb{F}_{q'}$, $q' \leq \mathrm{poly}(s)$, such that

$$c_e = \sum_{b \in \{0,1\}^\ell} t_e(b_1, \ldots, b_\ell),$$

where ℓ and $\mathrm{size}(t_e)$ are at most $\mathrm{poly}(s)$. Next, moving to the boolean world, Lemma 3.5 shows that such an algebraic representation can be transformed to obtain the coefficient function $\phi_f \in \#\mathsf{P}_p/\mathsf{poly}$ such that $\phi_f(e) = \langle c_e \rangle$. □

As mentioned earlier, with the coefficient function of f in place, we need a way to map the coefficients of f to u. Following is a well-known claim from Algebra, that will help us map the coefficients.

Claim 4.8 (Frobenius Homomorphism)**.** *Let R be a commutative ring of characteristic p. Define a map $\rho : R \to R$ as $\rho(u) = u^{p^i}$. Then, ρ is a ring homomorphism. Moreover, when R is a finite field $\mathbb{F}_q$, then ρ is an automorphism that fixes $\mathbb{F}_{p^i}$.*

We now have all the necessary tools needed to prove the lemma.

Proof. Proof of Lemma 4.4 Given that $f = u^{p^i}$, let $u =: \sum_{a \in L} c_a x^a$, where the *support* L represents the set of exponent vectors associated to u. Essentially, Claim 4.8 allows us to

[2] The same conclusion can be made from VNP closure properties stated in Lemma 1.28.

distribute the prime power over addition as follows:

$$f = u^{p^i} = \left(\sum_{a \in L} c_a \cdot x^a\right)^{p^i} = \sum_{a \in L} (c_a)^{p^i} x^{p^i \cdot a} .$$

The last expression above clearly associates the coefficients of $x^{p^i \cdot a}$ in f to coefficients of x^a in u. Since f is in VNP, Lemma 4.7 guarantees a #P/poly function ϕ_f such that the following congruence, in the finite field $\mathbb{F}_q$, is true for all $a \in L$:

$$\left(\phi_f\left(p^i \cdot a\right)\right)^{1/p^i} = \phi_f(p^i \cdot a)^{q/p^i} = \phi_f(p^i \cdot a)^{p^{a-i}} =: \phi_u(a) = \langle c_a \rangle .$$

In Lemma 1.22 it was proved that #P/poly functions are closed under repeated-squaring, hence we conclude that $\phi_u \in$ #P/poly. Invoking Proposition 1.5 on ϕ_u proves that the factor $u \in$ VNP. □

4.1.2 Factoring co-prime factors

The proof of Lemma 4.5 adheres to the conventional template of factoring, pioneered by Kaltofen, using Hensel's lifting lemma. We will follow the presentation of [KSS15, ST21, Sud98]. It commences with a series of preprocessing procedures that brings the polynomial in the right setup to invoke the lifting lemma, which uniquely gives the factor. We will elucidate all the steps, and along the way analyse the VNP size parameters to ultimately conclude the proof.

Transformation to monic polynomial. Let $\alpha := (\alpha_1, \ldots, \alpha_n) \in \mathbb{F}_q^n$. Define a homogeneous *shift* map $\tau_\alpha : \mathbb{F}_q[x_1, \ldots, x_n] \to \mathbb{F}_q[x, x_1, \ldots, x_n]$ such that for all $i \in [n]$, it maps $x_i \mapsto x_i + \alpha_i \cdot x$. Let $f_\alpha := \tau_\alpha(f)$ and observe that $\deg(f_\alpha) = \deg(f) =: d$. Isolating the coefficient c_e of the leading term x^d of f_α gives

$$c_e =: \sum_{|e|=d} \widehat{c}_e \cdot \alpha_1^{e_1} \ldots \alpha_n^{e_n}.$$

PIT lemma guarantees that with high probability, a random choice of α ensures c_e is a non-zero field element (refer to [SY10, Lemma 4.2]). Then, f_α / c_e is a monic polynomial in x. Further, if (s, s) is the VNP size parameters of f, then the parameters for f_α are $(s, s + O(n))$. When the field is too small, to pick the right α, we can obtain it from a

field extension K of degree at most poly(deg(f)). Since arithmetic operations over K can be efficiently simulated in $\mathbb{F}$ (refer to [Bü00, Proposition 4.1]), we will assume $K = \mathbb{F}_q$ without loss of generality.

Multivariate to bi-variate factoring. We can reduce the problem of multivariate factoring to the bi-variate case. For notational convenience, we redefine f_α/c_e as f_α and associate a polynomial $\bar{f} \in \mathbb{F}_q[x_1, \ldots, x_n][x, y]$ as follows: $\bar{f}(x, y) := f_\alpha(x, yx_1 + a_1, yx_2 + a_2, \ldots, yx_n + a_n)$, where $\mathbf{a} \in \mathbb{F}_q^n$ is a point.

If f_α is monic and u_α is its monic irreducible factor, then $\bar{u} := u(x, yx_1 + a_1, \ldots, yx_n + a_n)$ is a monic irreducible factor of $\bar{f}$, see [ST21, Lemma 3.10]. In addition to this bi-variate transformation, the scaling and shifting of variables sets up the starting point for the lifting lemma. Refer to [DDS22, Section 2.2] and [ST21, Section 3.5].

Claim 4.9 (Initialize Hensel lifting)**.** *Let $f = u \cdot v$ be such that u, v are co-prime polynomials. Then the associated* uni*variate factors $\bar{u}(x, 0)$ and $\bar{v}(x, 0)$ of $\bar{f}(x, 0)$ are co-prime.*

Note that, the factor u can be recovered easily from $\bar{u}$ by performing an inverse linear-transformation of the coordinate shift. Further, the polynomial $\bar{f}(x, y)$ remains monic in x and is in VNP with size parameters $(s, s + O(n))$.

Hensel's Lifting. Let us re-assign $f = \bar{f}$ for notational simplicity. Recall that $f(x, y)$ is monic in x, therefore $f_0 := f(x, 0) \in \mathbb{F}_q[x]$ is a univariate polynomial of degree d. Since f_0 can have at most d factors, $u_0 := u(x, 0)$ and $v_0 := v(x, 0)$ are in VNP with parameters $(1, O(d))$. We will use the following ever-famous Hensel's Lifting lemma from number theory to lift the roots uniquely (mod y). For a detailed discussion on the specific monic version of the Lifting lemma required for our proof, we encourage the readers to refer [KSS15, Lemma 3.4]. For the rest of the section we assume $\mathbb{K} := \mathbb{F}_q[x_1, \ldots, x_n]$ as the base ring of the bivariate polynomials in x, y.

Lemma 4.10 (Monic Hensel's Lifting)**.** *Let $f = u \cdot v \in \mathbb{K}[x, y]$ be such that u, v are co-prime, and u is monic in x. Additionally, we are given $u_0 \equiv u \bmod y$ and $v_o \equiv v \bmod y$ such that $a_0u_0 + b_0v_0 \equiv 1 \bmod y$. Then for all natural numbers $k \geq 1$ there exist $u_k, v_k, a_k, b_k \in \mathbb{K}[x, y]$ satisfying the following:*

1. $u_k \equiv u_{k-1} \bmod y^{2^{k-1}}$ *and* $v_k \equiv v_{k-1} \bmod y^{2^{k-1}}$.
2. $f \equiv u_k \cdot v_k \bmod y^{2^k}$ *such that* $a_ku_k + b_kv_k \equiv 1 \bmod y^{2^k}$ *and* u_k *is* monic *in* x.
3. $u_k \equiv u \bmod y^{2^k}$ *and* $v_k \equiv v \bmod y^{2^k}$.

Moreover, for every k*, the lifted factors* u_k *and* v_k *are* unique *polynomials* $\bmod y^{2^k}$.

Hensel's Lifting is a technical, but a very powerful, tool which gives explicit formulas for the lifted factors. Its basic idea is to take the error of the previous step and *feed it back* to the next step. Consider the difference polynomial $m_k := f - u_{k-1}v_{k-1}$. Then the polynomials $\bar{u}_k := u_{k-1} + b_{k-1}m_k$ and $\bar{v}_k := v_{k-1} + a_{k-1}m_k$ are valid lifts of the factors u and v. However, to obtain monic, and therefore unique lifts, we need some correction. Let $q_k, r_k \in \mathbb{K}[x, y]$ be such that

$$(\bar{u}_k - u_{k-1}) =: y^{2^{k-1}} \cdot (q_k u_{k-1} + r_k),$$

where $\deg_x(r_k) \leq \deg_x(u_{k-1})$. The existence of these polynomials is guaranteed by Euclid's division algorithm. Then the unique, and monic, lifts are defined as follows:

$$u_k := u_{k-1} + y^{2^{k-1}} r_k \tag{4.1}$$

$$v_k := \bar{v}_k \left(1 + y^{2^{k-1}} q_k\right) . \tag{4.2}$$

It is easy to verify that they are the valid lifts as per Lemma 4.10. Refer [KSS15, Lemma 3.4] for rigorous calculations. In addition, let $w_k := a_{k-1}u_k + b_{k-1}v_k$, then the lifted factors remain (pseudo-)co-prime ($\bmod\ y^{2^k}$) with Bézout identity holding using the following polynomials:

$$a_k := a_{k-1}(1 - w_k)$$
$$b_k := b_{k-1}(1 - w_k).$$

Size analysis. We choose an integer $t \geq \log(\deg_y(u)) + 1$ and repeatedly use the Lifting lemma t times to obtain the factor $u_t \equiv u \mod y^{2^t}$. Since the lifted factors are unique, u can be obtained from u_t by truncating it to $\deg_y(u)$. Given that $f \in \mathsf{VNP}$, the factor $u \in \mathsf{VNP}$ can be proved using the following technical lemma. It proves that given the coefficients of polynomial f in variables $x_1, \ldots, x_n$, there is a small circuit which computes the lifted factor u.

Lemma 4.11 (Hensel in circuits)**.** *Let* $f = u \cdot v \in \mathbb{K}[x, y]$ *be a degree* d *polynomial such that* u, v *are co-prime and* u *is monic in* x*. The polynomials* u_0, v_0, a_0, b_0 *are defined as before. Let* L *be the set of exponent vectors of* f *such that* $f =: \sum_{e_i \in L} c_{e_i}(x_1, \ldots, x_n) \cdot x^{e_{i1}} y^{e_{i2}}$.

Given the coefficients $c_{e_1}, \ldots, c_{e_{|L|}}$ *as input, there exists a circuit* $C_u^{(t)}$ *over* $\mathbb{F}_q$ *which computes* $\mathrm{Hom}_{\leq d}(u_t)$[3]*. Further, there is a constant* $\beta \geq 2$ *such that the size of the circuit* $C_u^{(t)}$ *is at most* $\mathrm{poly}(d, \beta^t)$*, and intermediate degrees at most* $(d\beta^t)$*.*

Proof. Given all the coefficients of the polynomial f, observe that we can construct a sub-circuit C_f of size $s_f := \mathrm{poly}(d)$ that computes f. Then, the proof is an easy consequence of the following inductive analysis on t.

The base case is easy to analyse. Let $C_u^{(t-1)}, C_v^{(t-1)}, C_a^{(t-1)}$, and $C_b^{(t-1)}$ be the circuits that compute $u_{t-1}, v_{t-1}, a_{t-1}$ and b_{t-1} respectively, as described in Hensel's lifting Lemma 4.10. Let the size of all the circuits be at most $s_{t-1} := \mathrm{poly}(d, \beta^{t-1})$. Together with C_f, the difference polynomial m_k can be easily computed in size $s_f + O(s_{t-1})$ [4]. Then observe that $\mathrm{size}(\bar{u}_t)$ and $\mathrm{size}(\bar{v}_t)$ is at most $s_f + O(s_{t-1})$. To facilitate the lifting process, the quotient q_k and remainder r_k can be computed with additional $\mathrm{poly}(d)$ size (refer [KSS15, Lemma 2.8] and [vzGG13, Lemma 9.6]). Using these as sub-circuits, we obtain C_u^t and C_v^t with additional constant number of gates from Equations 4.1 and 4.2. Overall, the size of the lifted polynomials grows by a constant factor and, hence, the overall size of both the circuits is at most $s_t := s_f + O(s_{t-1}) + \mathrm{poly}(d) + O(\beta) \leq \mathrm{poly}(d, \beta^t)$. Almost the same argument works for circuits $C_a^{(t)}$ and $C_b^{(t)}$ computing a_t and b_t.

Lastly, we homogenize C_u^t using Lemma 1.23, to obtain the desired circuit which computes $\mathrm{Hom}_{\leq d}(u_t)$. The degree with respect to the lifting variable y is at most β^t due to constant growth in each iteration, moreover, with respect to x it is at most d due to the homogenization. Hence, the degree claim follows. □

We are now ready to give the complete proof of the following Lemma 4.5.

Lemma 4.5 (restated). *Let* $f \in \mathbb{F}_q[x_1, \ldots, x_n]$ *be a polynomial in* VNP*. If there are co-prime polynomials* u *and* v *such that* $f = u \cdot v$*, then the factor* u *is in* VNP*.*

Proof. Proof of Lemma 4.5 Assume that $f \in \mathbb{K}[x, y]$ after all the necessary invertible transformations discussed earlier in the section to apply Lemma 4.10. Let L be the support set of f such that $f =: \sum_{e_i \in L} c_{e_i}(x_1, \ldots, x_n) \cdot x^{e_{i1}} y^{e_{i2}}$.

Using Lemma 4.11 with $t \geq \log(\deg(f)) + 1$ gives a circuit $C_u^{(t)}$ that take the coefficients of f as input and outputs a circuit for the factor u. Moreover, the size of the circuit is at most $\mathrm{poly}(\deg(f))$ and degree is at most $O(\deg(f))$.

[3] This is the sum of the homogeneous parts of u_t up to degree d.
[4] For notations, refer to the discussion proceeding Lemma 4.10.

Since $f \in \mathsf{VNP}$, Lemma 1.28(2) shows that the coefficients $c_{e_i} \in \mathsf{VNP}$. Moreover, Lemma 1.28(3) will prove that $C_u^{(t)}$ composed with VNP polynomials, remains in VNP. Therefore, the factor u is in VNP. □

Chapter 5

Explicitness of Low-Degree Factors

> Problems worthy of attack, Prove their worth by fighting back
>
> Piet Hein

In an ambitious program to resolve the $\mathsf{P} \stackrel{?}{=} \mathsf{NP}$ question using methods from algebraic geometry and representation theory, Mulmuley and Sohoni [MS01] strengthened Valiant's conjecture by postulating that VNP is not contained in $\overline{\mathsf{VP}}$.

Completely independently and almost at the same time, Bürgisser [Bür04] (also see [Bü20]) introduced and used border complexity to factor multivariate polynomials. Bürgisser showed that for border complexity, the factor conjecture is indeed true – the factor u above, is in $\overline{\mathsf{VP}}$ (refer Conjecture 1.12). This makes factor conjecture an important stepping-stone towards understanding algebraic computation. We observe that it is in fact in a smaller presentable border class $\overline{\mathsf{VP}}_\varepsilon$ (refer Chapter 3). We formally define the class below.

Definition 5.1 (Presentable $\overline{\mathsf{VP}}$)**.** *The presentable border class* $\overline{\mathsf{VP}}_\varepsilon$ *is defined as the set of polynomials* $f \in \mathbb{F}[x_1, \ldots, x_n]$ *such that there is an approximating polynomial* $g \in \mathbb{F}[\varepsilon][x_1, \ldots, x_n]$ *satisfying*

$$g(\mathbf{x}, \varepsilon) = \varepsilon^M f(\mathbf{x}) + \varepsilon^{M+1} Q(\mathbf{x}, \varepsilon),$$

for some $Q \in \mathbb{F}[\varepsilon][x_1, \ldots, x_n]$ *and* $M \in \mathbb{N}$*. Moreover,* $\mathrm{size}_{\mathbb{F}}(g)$ *and* $\deg_{\mathbf{x}}(g)$ *is bounded by* $\mathrm{poly}(n)$*.*

Although, g has a small size circuit, we emphasise that the degree of ε-polynomials in g is unrestricted. Further, it is apparent from the definitions that $\mathsf{VP} \subseteq \overline{\mathsf{VP}}_\varepsilon \subseteq \overline{\mathsf{VNP}}_\varepsilon$. Búrgisser in [Bür04, Theorem 1.3] proved that the class $\overline{\mathsf{VP}}_\varepsilon$ contains all the low-degree *separable factors*[1] of circuits of small size.

Lemma 5.2. *Let* $q := p^a$ *and* e *be a positive integer coprime to* p. *Consider a polynomial (family)* $f \in \mathbb{F}_q[x_1, \ldots, x_n]$ *satisfying* $f = u^e v$, *where* u *is irreducible and coprime to* v, *such that* $\mathrm{size}(f)$ *and* $\deg(u)$ *is at most* $s := \mathrm{poly}(n, \log q)$. *Then we have* u *in* $\overline{\mathsf{VP}}_\varepsilon$.

Remark 5.3. We make a few observations.

1. In case $f = u^e$, Kaltofen [Kal87] showed that u is VP.
2. Búrgisser [Bür04] proved that u (in the lemma above) is in $\overline{\mathsf{VP}}$. Moreover, he remarked that, in his proof, the required polynomials in $\mathbb{F}[\varepsilon]$ do have small circuit-complexity (refer the remark following [Bür04, Definition 2.1]). For the sake of completeness, we will sketch the proof for $u \in \overline{\mathsf{VP}}_\varepsilon$ in the subsequent section.

As an application of the debordering result over finite fields in Theorem 3.6, we prove that the low-degree separable factors of small size circuits are explicit.

Corollary 5.4. *Let* $q := p^a$ *and* e *be a positive integer coprime to* p. *Consider a polynomial (family)* $f \in \mathbb{F}_q[x_1, \ldots, x_n]$ *and its irreducible factor* u *satisfying* $f = u^e v$, u *coprime to* v, *such that* $\mathrm{size}(f)$ *and* $\deg(u)$ *is* $\mathrm{poly}(n, \log q)$. *Then, the polynomial (family)* u *is in* VNP.

Proof. We learn from Lemma 5.2 that the polynomial family $u \in \overline{\mathsf{VP}}_\varepsilon$. Moreover, $\overline{\mathsf{VP}}_\varepsilon$ is contained in $\overline{\mathsf{VNP}}_\varepsilon$ by definition. As over $\mathbb{F}_q$, Theorem 3.6 proves $\overline{\mathsf{VNP}}_\varepsilon = \mathsf{VNP}$, hence $u \in \mathsf{VNP}$.

□

5.1 Low degree factors are easy to approximate

In this section we will sketch the proof of Lemma 5.2. Consider a polynomial $f \in \mathbb{F}_q[x_1, \ldots, x_n]$ of degree d_f from the lemma statement. For all $i \in [n]$, randomly pick

[1] i.e. factor u which is irreducible and has multiplicity coprime to the characteristic p. This isn't an issue in characteristic zero fields.

field elements $\alpha_i, \beta_i \in_r \mathbb{F}_q$ and define a map $\tau : x_i \mapsto x_i + \alpha_i y + \beta_i$, where y is a new variable. Under such a random invertible transformation, the polynomial completely splits over power series ring, see [DSS22, Theorem 17]. In particular, there exists $k \in \mathbb{F}_q^*, \gamma_i > 0$ and $h_i \in \mathbb{K}[\![x_1, \ldots, x_n]\!]$ satisfying

$$\tau(f) = k \cdot \prod_{i \in [d_f]} (y - h_i)^{\gamma_i},$$

where $\mathbb{K}$ is a field extension of $\mathbb{F}_q$ of degree at most d_f. Refer [DSS22, Section 3 and 6.2] for details. Further, $\mu_i \coloneqq h_i(\bar{0})$ are all distinct nonzero field elements. We assume $\mathbb{K} = \mathbb{F}_q$ without loss of generality (refer [Bü00, Proposition 4.1]). An immediate corollary of such a power series split is the following lemma (refer [DSS22, Corollary 18])

Lemma 5.5. *Let u be a factor of f of degree d_u, and $\tau(f)$ splits as before. Since $\tau(u)$ divides $\tau(f)$, we deduce that*

$$\tau(u) = k' \cdot \prod_{i \in [d_u]} (y - h_i)^{c_i},$$

where $0 \le c_i \le \gamma_i$, $k' \in \mathbb{F}_q^*$, and $h_i \in \mathbb{F}_q[\![x_1, \ldots, x_n]\!]$.

Recall the definition of $\mathrm{Hom}_{\le d_u}(h_i)$ from Section 1.6. Observe that

$$\tau(u) \equiv k' \cdot \prod_{i \in [d_u]} (y - \mathrm{Hom}_{\le d_u}(h_i))^{c_i} \bmod \langle x_1, \ldots, x_n \rangle^{d_u+1}.$$

Later we will show that due to the expression above it would suffice to give a complexity bound of the power series roots of $\tau(f)$ to uniquely recover the factor $\tau(u)$. The following proposition proves that all the power series roots can be easily approximated, see [Bür04, Proposition 3.4].

Proposition 5.6. *For all $i \in [d_f]$, there is an approximating polynomial $g_i \in \mathbb{F}_q[\varepsilon][x_1, \ldots, x_n]$ satisfying $g_i = \varepsilon^M \mathrm{Hom}_{\le d_u}(h_i) + \varepsilon^{M+1} Q_i(\boldsymbol{x}, \varepsilon)$, for some error $Q_i \in \mathbb{F}[\varepsilon][x_1, \ldots, x_n]$ and order $M \in \mathbb{N}$. Moreover $\deg_{\boldsymbol{x}}(g_i) \le d_u$ and $\mathrm{size}_{\mathbb{F}}(g_i) \le \mathrm{poly}(d_u, \mathrm{size}(f))$.*

Proof-Sketch Let $\tilde{f} \coloneqq \tau(f) \in \mathbb{F}_q[\boldsymbol{x}, y]$ and $\mu_i \coloneqq h_i(\bar{0})$. Define a perturbed polynomial $F \coloneqq \tilde{f}(\boldsymbol{x}, y + \mu_i + \varepsilon) - \tilde{f}(\boldsymbol{x} = 0, y = \mu_i + \varepsilon)$ over the ring $\mathbb{F}_q[\varepsilon]$. Since e is coprime to p, with appropriate coordinate shift it can be ensured that

$$F(\boldsymbol{x}, y, \varepsilon = 0) = \tilde{f}, \text{ and } F(\bar{0}, 0) = 0, \text{ but } \partial_y F(\bar{0}, 0) \neq 0,$$

see [Bür04, Equation 5]. Then the power series root H_j of the perturbed polynomial F can be obtained by classical Newton Iteration (refer [DSS22, Lemma 15]) as follows:

$$y_0 = 0, \quad y_{t+1} \coloneqq y_t - \frac{F(\mathbf{x}, y_t)}{\partial_y F(\mathbf{x}, y_t)} \tag{5.1}$$

where $y_t \equiv H_j \bmod \langle \mathbf{x} \rangle^{2^t}$. The quadratic convergence of degree in Newton iteration implies that it suffices to assume $t \leq \log d_u + 1$. An easy induction on t proves that y_t, and therefore H_j, is well defined over $\mathbb{F}_q[\![\varepsilon]\!]$. Hence, $H_j(\mathbf{x}, y, \varepsilon = 0) = h_i$, moreover $\mathrm{Hom}_{\leq d_u}(H_j)(\mathbf{x}, y, \varepsilon = 0) = \mathrm{Hom}_{\leq d_u}(h_i)$.

Let R be the subring of $\mathbb{F}(\varepsilon)$ consisting of rational functions defined at $\varepsilon = 0$. The preceding discussion then proves that an approximating polynomial $\tilde{g}_i \in R[\mathbf{x}]$ computes $A_t/B_t \equiv y_t \bmod \langle \mathbf{x} \rangle^{2^t}$ and satisfies the following:

$$\tilde{g}_i = \mathrm{Hom}_{\leq d_u}(h_i) + \varepsilon\, \tilde{Q}_i(\varepsilon, \mathbf{x}, y),$$

for some error $\tilde{Q}_i \in R[x_1, \ldots, x_n]$. Equivalently, there exists an approximating polynomial $g_i \in \mathbb{F}[\varepsilon][x_1, \ldots, x_n]$, order $M \in \mathbb{N}$, and error Q_i, such that $g_i = \varepsilon^M \mathrm{Hom}_{\leq d_u}(h_i) + \varepsilon^{M+1} Q_i(\varepsilon, \mathbf{x}, y)$ (refer discussion following Definition 3.1 and [Bür04, Lemma 5.6]). Moreover, $g_i = \varepsilon^M \tilde{g}_i$. Therefore, the proposition follows from showing

$$\mathrm{size}_{\mathbb{F}}(\tilde{g}_i) \leq \mathrm{poly}(d_u, \mathrm{size}(\tilde{f})),$$

and $M \leq 2^{\mathrm{poly}(d_u)}$. In case, $\deg_{\mathbf{x}}(g_i)$ is greater than d_u, homogenise and truncate the higher degree terms ([Bür04, Proposition 3.1]).

Size analysis. The circuit computing A_t/B_t is build iteratively using division gates following Equation (5.1). Treating ε as a variable, observe that $\mathrm{size}_{\mathbb{F}}(F) \leq s_0 \coloneqq \mathrm{size}(\tilde{f}) + 2$. Homogenise the circuit computing F using Lemma 1.23 with respect to y to obtain $\mathrm{Hom}_{\leq d_u}(F)$ of size $\mathrm{poly}(d_u, s_0)$. Use this homogenised circuit to obatin the circuit computing $\partial_y F$ of size $\mathrm{poly}(d_u, s_0)$. Using division and subtraction gates, compute A_1/B_1 and let its size be $s_1 \coloneqq \max(\mathrm{size}_{\mathbb{F}}(A_1), \mathrm{size}_{\mathbb{F}}(B_1)) \leq \mathrm{poly}(d_u, s_0)$. Let $t = \log d_u + 1$, then Newton iteration gives an easy recurrence on the size $s_t \leq c + s_{t-1} + \mathrm{poly}(d_u, s_0)$, where c is a small constant. Solving the recurrence gives $s_t \leq \mathrm{poly}(d_u, s_0) \leq \mathrm{poly}(d_u, \mathrm{size}(\tilde{f}))$. Finally, eliminate the division with respect to $\mathbf{x}, y$ variables using Lemma 1.24 to obtain the circuit computing $\tilde{g}_i$ of size at most $\mathrm{poly}(d_u, \mathrm{size}(\tilde{f}))$. The upper bound is preserved after the inverse transformation τ^{-1}.

Order analysis. Let the ε degree in A_1/B_1 be denoted by $d_1 \coloneqq \max(\deg_\varepsilon(A_1), \deg_\varepsilon(B_1))$. Observe that in each iteration the degree blows-up by a factor of d_u because of homogenisation in preprocessing. Thus, we get the recurrence $d_t \leq d_u \cdot d_{t-1}$, solving which gives $d_t \leq (d_u)^{\log d_u} \leq 2^{\mathrm{poly}(d_u)}$. Then to obtain g_i, set $M = d_t \leq 2^{\mathrm{poly}(d_u)}$. □

We are now ready to prove Lemma 5.2. For two arbitrary polynomials u and h, let $\mathrm{size}(u \mid h)$ denote the size of the circuit that computes u given h for free. The definition can be extended to $\overline{\mathrm{size}}(u \mid h)$ naturally.

Lemma 5.2 (restated). *Let $q \coloneqq p^a$ and e be a positive integer coprime to p. Consider a polynomial $f \in \mathbb{F}_q[x_1, \ldots, x_n]$ satisfying $f = u^e v$, where u is irreducible and coprime to v, such that $\mathrm{size}(f)$ and $\deg(u)$ is at most $s \coloneqq \mathrm{poly}(n, \log q)$. Then we have u in $\overline{\mathsf{VP}}_\varepsilon$.*

Proof. Using the map τ defined earlier, Lemma 5.5 proves that $\tau(u) = k' \cdot \prod_{i \in [d_u]} (y - h_i)^{c_i}$ where $h_i \in \mathbb{F}_q[\![x_1, \ldots, x_n]\!]$ and $d_u \coloneqq \deg(u)$. Suppose

$$H \coloneqq k' \cdot \prod_{i \in [d_u]} (y - \mathrm{Hom}_{\leq d_u}(h_i))^{c_i},$$

then observe that $\tau(u) \equiv H \bmod \langle \boldsymbol{x} \rangle^{d_u+1}$. The idea is to show that H can be easily approximated, hence the factor can be obtained accurately by eliminating division.

It is easy to verify that

$$\overline{\mathrm{size}}(H) \leq \overline{\mathrm{size}}\,(H \mid \mathrm{Hom}_{\leq d_u}(h_i)) + \overline{\mathrm{size}}\,(\mathrm{Hom}_{\leq d_u}(h_i)),$$

see [Bür04, Lemma 2.3(3)]. Since $d_u \leq s$, Proposition 5.6 proves that $\overline{\mathrm{size}}\,(\mathrm{Hom}_{\leq d_u}(h_i))$ is at most $\mathrm{poly}(s)$. Then, clearly presentable border complexity $\overline{\mathrm{size}}(H) \leq \mathrm{poly}(s)$. Suppose G approximates H, in the usual sense. Eliminate division in $G \bmod \langle \boldsymbol{x} \rangle^{d_u+1}$ using Lemma 1.24 to obtain the approximation of $\tau(u)$, moreover almost immediately we get that $\overline{\mathrm{size}}(\tau(u)) \leq \mathrm{poly}(s, d_u)$. Since shifting and scaling do not change the size complexity, apply the inverse transformation to conclude that $u \in \overline{\mathsf{VP}}_\varepsilon$. □

Part III

Identity Testing

Chapter 6

Whitebox Identity Testing of Depth-4 Circuits

> If a design is good enough to evolve once, the same design principle is good enough to evolve twice, from different starting points, in different parts of the animal kingdom.
>
> *Richard Dawkins, The Blind Watchmaker*

Polynomial Identity Testing (PIT) is one of central problems of Algebraic Complexity, which asks: given an algebraic circuit $\mathcal{C}$ over a field $\mathbb{F}$ and input variables $x_1, \ldots, x_n$, determine whether $\mathcal{C}$ computes the identically zero polynomial.

Theorem 6.1 ($\Sigma^k\Pi\Sigma\wedge$ Whitebox PIT)**.** *There is a deterministic, whitebox PIT algorithm for $\Sigma^{[k]}\Pi\Sigma\wedge$ circuits of size s over $\mathbb{F}[\mathbf{x}]$ running in time $s^{O(k7^k)}$.*

Remark 6.2.

1. Case $k \leq 2$ can be solved by invoking [SSS13, Theorem 5.2]; but $k \geq 3$ was open.
2. Our technique *necessarily* blows up the exponent exponentially in k. In particular, it would be interesting to design an efficient time algorithm when $k = \Theta(\log s)$.
3. It is not clear if the current technique gives PIT for $\Sigma^{[k]}\Pi\Sigma\mathsf{M}_2$ circuits, where $\Sigma\mathsf{M}_2$ denotes sum of *bi*variate monomials computed and fed into the top product gate.

6.1 State of Affairs

A very simple randomized algorithm for PIT is known for decades — evaluate the polynomial at a random point from a large enough domain. Due to the Polynomial Identity Lemma [Ore22, DL78, Zip79, Sch80], it is highly likely that a nonzero polynomial will have a nonzero evaluation. De-randomizing the algorithm has been an open problem for a long time, but a crude de-randomization gives the simplest PIT algorithm for any general circuit. When the number of variables is small, say constant, then this algorithm is very efficient. Refer to [Sap13, Corollary 2.2].

Lemma 6.3 (Trivial PIT)**.** *For a class of n-variate, individual degree at most d polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ there exists a deterministic PIT algorithm which runs in time $O(d^n)$.*

Bounded Depth PIT. For a long time the exponential-time algorithm from Lemma 6.3 was the fastest known algorithm for constant depth circuits. This changed with the breakthrough result of Limaye, Srinivasan, and Tavenas [LST21], which gave the first sub-exponential algorithm for small-depth circuits. They achieved it by proving super-polynomial lower bounds for bounded depth circuits and invoking hardness vs randomness trade-off result [CKS18].

Sparse PIT. A sparse polynomial has bounded number of monomials. For a long time, the only polynomial time algorithm for PIT was known for sparse polynomial due to Klivans and Spielman [KS01]. Refer [Sap13, Section 2.2.3] for the proof.

Theorem 6.4 (Sparse-PIT Map)**.** *Let $p(\mathbf{x}) \in \mathbb{F}[\mathbf{x}]$ with individual degree at most d and sparsity at most m. Then, there exists $1 \leq r \leq (mn \log d)^2$, such that*

$$p(y, y^d, \ldots, y^{d^{n-1}}) \neq 0, \bmod y^r - 1.$$

If p is computable by a size-s $\Sigma\Pi$ circuit, then there is a deterministic algorithm to test its identity which runs in time $\mathrm{poly}(s, m)$.

The algorithm can be easily extended for identity testing product of sparse polynomials, as described in [Sap13, Lemma 2.3]. See also [Dut22, Lemma 2.7.6].

Lemma 6.5 (Product of Sparse PIT)**.** *For a class of n-variate, degree d polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ computable by $\Pi\Sigma\Pi$ of size s, there is a deterministic PIT algorithm which runs in time $\mathrm{poly}(s, d)$.*

Depth-3 PIT. These circuits, denoted by $\Sigma^{[k]}\Pi^{[d]}\Sigma$, compute polynomial of the form $C = T_1 + \cdots + T_k$, where T_i is a product of linear terms. A deterministic whitebox PIT algorithm was given by Kayal and Saxena which runs in time $\text{poly}(n, d^k)$ [KS07]. And after a series of results, Saxena and Seshadhri [SS12] gave the blackbox algorithm with the same running time. For the Diagonal Depth-3 circuit $\Sigma^{[k]}\wedge\Sigma$, Saxena [Sax08] gave a poly-time whitebox algorithm using the duality trick, and later Agrawal, Saha, and Saxena [ASS13] introduced a blackbox algorithm which runs in time $\text{poly}(knd)^{\log(knd)}$.

Depth-4 PIT. In a surprising result, Agrawal and Vinay [AV08] showed that a complete derandomization of blackbox identity testing for just depth-4 algebraic circuits ($\Sigma\Pi\Sigma\Pi$) already implies a near complete derandomization for the general PIT problem. The bootstrapping phenomenon [AGS19, KST19a, GKSS22, And20] showed that achieving PIT for the restricted depth-4 circuits would have significant implications for PIT of general circuits. These results accentuate the importance and relevance of focusing on identity testing for depth-4 circuits. Although nothing better than sub-exponential is known in general, under various restrictions insightful and efficient algorithms have been discovered.

TABLE 6.1: Time complexity comparision of PIT algorithms related to $\Sigma\Pi\Sigma\Pi$ circuits

Model	Time	Ref.
Multilinear $\Sigma^{[k]}\Pi\Sigma\Pi$	$\text{poly}(n^{O(k^2)})$	[SV18, ASSS16]
$\Sigma\Pi\Sigma\Pi$ of bounded trdeg	$\text{poly}(s^{\text{trdeg}})$	[BMS13]
$\Sigma^{(k)}\Pi\Sigma\Pi^{[d]}$ of bounded *local* trdeg	$QP(n)$	[KS18]
$\Sigma^{[3]}\Pi\Sigma\Pi^{[2]}$	$\text{poly}(n, d)$	[PS21]
$\Sigma\Pi\Sigma\Pi$	SUBEXP(n)	[LST21]
$\Sigma^{[k]}\Pi\Sigma\wedge$	$s^{O(k \log\log s)}$	[DDS22]
$\Sigma^{[k]}\Pi\Sigma\Pi^{[]}\delta$	$s^{O(\delta^2 k \log s)}$	[DDS22]
Whitebox $\Sigma^{[k]}\Pi\Sigma\wedge$	$s^{O(k 7^k)}$	This chapter.

The table above is not an exhaustive list of all the depth restricted PIT results. We encourage interested readers to find more detailed discussions in surveys [SY10, Sax09b, Sax14a, AS09, DG24]. One PIT algorithm that we crucially need in our algorithm is that of Diagonal Depth-4 circuits $\Sigma\wedge\Sigma\wedge$. These circuits compute polynomials of the form $f(\mathbf{x}) = \sum_{i\in[s]} f_i^{e_i}$ where f_i is a sum of univariate polynomials. Using the duality trick Lemma 1.32 and the PIT results from [RS05, GKS17], an efficient PIT algorithm can be designed for $\Sigma\wedge\Sigma\wedge$ circuits.

Lemma 6.6 (PIT for $\Sigma\wedge\Sigma\wedge$)**.** *There exists a* $\mathrm{poly}(s)$ *time whitebox PIT algorithm for polynomials computable by* $\Sigma\wedge\Sigma\wedge$ *circuit of size* s*.*

Proof sketch. We show that any $g(\mathbf{x})^e = (g_1(x_1) + \ldots + g_n(x_n))^e$, where $\deg(g_i) \leq s$ can be written as $\sum_j h_{j1}(x_1)\cdots h_{jn}(x_n)$, for some $h_{j\ell} \in \mathbb{F}[x_\ell]$ of degree at most es. Define, $G := (y + g_1)\cdots(y + g_n) - y^n$. In its e-th power, notice that the leading-coefficient is $\mathrm{coef}_{y^{e(n-1)}}(G^e) = g^e$. So, interpolate on $e(n-1)+1$ many points $(y = \beta_i \in \mathbb{F})$ to get

$$\mathrm{coef}_{y^{e(n-1)}}(G^e) \;=\; \sum_{i=1}^{e(n-1)+1} \alpha_i\, G^e(\beta_i)\,.$$

Now, expand $G^e(\beta_i) = ((\beta_i + g_1)\cdots(\beta_i + g_n) - \beta_i^n)^e$, by binomial expansion (without expanding the inner n-fold product). The top-fanin can be atmost $s\cdot(e+1)\cdot(e(n-1)+1) = O(se^2n)$. The individual degrees of the intermediate univariates can be at most es. Thus, it can be computed by an ARO of size at most $O(s^2e^3n)$.

Now, if $f = \sum_{j\in[s]} f_j^{e_j}$ is computed by a $\Sigma\wedge\Sigma\wedge$ circuit of size s, then clearly, f can also be computed by an RARO of size at most $O(s^6)$. Thn the whitebox PIT follows from [RS05]. □

For a more comprehensive discussion refer to [For14, Section 8.6, and Corollary 8.6.9].

6.2 Gentle Introduction to DiDI

Techniques for de-randomizing PIT are mostly tailor-made for specific models, making them inextensible and with restricted applicability. We observed a similar limitation in the techniques of de-bordering results in Section 2.2, and we will later make a few comparative comments on why we believe *de-bordering* is perhaps harder than *de-randomization*. For instance, consider Kayal-Saxena whitebox identity testing algorithm for $\Sigma^{[k]}\Pi\Sigma$ circuits [KS01]. Although $\Sigma^{[k]}\Pi\Sigma$ is a restricted version of the model of our interest $\Sigma^k\Pi\Sigma\wedge$, their *Chinese Remaindering* over local rings loses applicability for a slightly general model (refer [SSS13] as well). Refer our paper for a more comprehensive discussion on limitation of known techniques [DDS21, Section 1.2]. In the rest of the section we introduce our novel technique to de-randomize whitebox PIT for depth-restricted circuit.

The de-bordering paradigm DiDIL, discussed in Section 2.3, and the upcoming discussion on the de-randomizing PIT technique DiDI, share the same underlying philosophy—

reduce the problem to a well-understood *powering* model ($\wedge$). Consider a pedagogical example, where we want to test the identity of $f := T_1 + T_2$, and T_i is a product of sum of univariates ($\Pi\Sigma\wedge$). The goal is to reduce the fan-in while preserving the non-zeroness, so that it suffices to test the reduced model. To achieve this, we Divide the expression by T_2 and take the derivative. Naturally, these operations pushes us to work with the fractional ring, further it also distorts the model as $\mathcal{T}_i$'s are no longer computable by simple $\Pi\Sigma\wedge$ circuits. However, with careful analytically analysis we establish that the non-zeroness is preserved in the reduced model.

The garbled model obtained from single iteration of DiDI is useful and closed under subsequent iterations of DiDI. We reuse the bloated computation mode $\mathsf{Gen}(k, s)$ from Section 2.3 for repeated application of DiDI.

Definition 2.10 (Restated). *Let $R(\mathbf{x})$ be the ring of rational functions. Denote $\mathsf{Gen}(k, s)$ as a class of circuits $\mathcal{C}$ over R which computes polynomial $f(\mathbf{x}) \in R(x_1, \ldots, x_n)$ of the form: $f = \sum_{i=1}^{k} T_i$ such that*

$$T_i = \left(\frac{U_i}{V_i}\right) \cdot \left(\frac{P_i}{Q_i}\right),$$

where $U_i, V_i \in \Pi\Sigma\wedge$ and $P_i, Q_i \in \Sigma\wedge\Sigma\wedge$ are polynomials in $R[x_1, \ldots, x_n]$.

The size of the circuit is defined naturally as $\mathrm{size}(\mathcal{C}) := \sum_{i=1}^{k} \mathrm{size}(T_i) \leq s$ where

$$\mathrm{size}(T_i) = \mathrm{size}(U_i) + \mathrm{size}(V_i) + \mathrm{size}(P_i) + \mathrm{size}(Q_i).$$

The syntactic degree of the circuit is defined as the maximum syntactic degree of the numerator and denominator.

We cannot directly perform identity testing on the simplest bloated model $\mathsf{Gen}(1, \cdot)$ because the denominator may not be invertible in the function field. While it may seem counterintuitive, given that the initial idea was to reduce the problem of testing $\Sigma^k\Pi\Sigma\wedge$ to testing $\mathsf{Gen}(1, \cdot)$, additional essential properties are required to provide a complete algorithm.

Finally we need a homomorphism similar to Definition 2.14 to perform safe division. Given the constructive nature of the algorithm, we cannot use random point for shifting and scaling as before. We will use the hitting set of product of sparse polynomials (refer Lemma 6.5) to obtain a point $\mathbf{a} = (a_1, \ldots, a_n) \in \mathbb{F}^n$ such that $T_{i,0}(\mathbf{x} = \mathbf{a}) \neq 0$, for all $i \in [k]$. Eventually this evaluation point will help in maintaining the invertibility of $\Pi\Sigma\wedge$.

Consider

$$g := \prod_{i\in[k]} T_{i,0} = \prod_{i\in[k]} T_{i,0} = \prod_{i\in[\ell]} \sum_{j\in[n]} f_{ij}(x_j),$$

where $f_{ij}(x_j)$ are univariate polynomials of degree at most d and $\ell \leq k \cdot s$. Note that $\deg(g) \leq d \cdot k \cdot s$ and g is computable by a $\Pi\Sigma\wedge$ circuit of size $O(s)$. Invoke Lemma 6.5 to obtain a hitting set $\mathcal{H}$, then evaluate g on every point of $\mathcal{H}$ to find an element $\mathbf{a} \in \mathcal{H}$ such that $g(\mathbf{x} = \mathbf{a}) \neq 0$.

Definition 6.7 (DiDI homomorphism)**.** *Pick* $\mathbf{a} = (a_1, \ldots, a_n)$ *from* $\mathbb{F}^n$ *using Sparse PIT as described above. Define a map*

$$\Phi : \mathbb{F}[x_1, \ldots, x_n] \to \mathbb{F}[z, x_1, \ldots, x_n]$$

such that for all $i \in [n]$, $x_i \mapsto z \cdot x_i + a_i$.

We emphasise that in whitebox setting all $T_{i,0}$, are readily available for evaluation. Since, the size of the set is poly(s) and each evaluation takes poly(s) time, this preliminary step will add poly(s) time to the overall time complexity. Moreover, we obtain the $\mathbf{a} \in \mathbb{F}^n$ which possess the non-zeroness preserving property.

6.3 De-randomizing PIT of $\Sigma^{[k]}\Pi\Sigma\wedge$ using DiDI

In this section, we develop the tools needed to give the complete poly-time algorithm for whitebox PIT of $\Sigma^k\Pi\Sigma\wedge$, when k is constant. We start by considering a polynomial $f(\mathbf{x}) \in \mathbb{F}[x_1, \ldots, x_n]$ that is computed by the $\Sigma^{[k]}\Pi\Sigma \wedge (s) \subseteq \mathsf{Gen}(k, s)$ circuit (Definition 2.10). Begin with defining $f_0 := f$ and $T_{i,0} := T_i$ where $T_{i,0} \in \Pi\Sigma\wedge$ such that

$$\sum_i T_{i,0} = f_0,$$

and size$(f_0) \leq s$. Assume $\deg(f) < d \leq s$, where we keep the parameter d separately to aid us later in complexity optimization. In every recursive call we work with $\mathsf{Gen}(\cdot)(\cdot)$ circuits. We start by applying DiDI homomorphism from Definition 6.7 on f_0. Invertibility implies that

$$f_0 = 0 \iff \Phi(f_0) = 0.$$

Lemma 6.8 (First Reduction). *Let* $\Phi(f_0)$ *be computable by* $\Sigma^{[k]}\Pi\Sigma\wedge$ *circuit of size* s_0 *over* $\mathbb{F}[z]$. *Then*

$$f_1 \coloneqq \partial_z\left(\frac{\Phi(f_0)}{\Phi(T_{k,0})}\right) \in \mathsf{Gen}(k-1, s_1)$$

over $R_1 \coloneqq \mathbb{F}[z]/\langle z^d\rangle$, *and* $s_1 = O(d^3 \cdot s)$. *Moreover,*

$$\Phi(f_0) \neq 0 \;\; \textit{over}\, \mathbb{F}[\mathbf{x}] \iff f_1 \neq 0 \;\; \textit{over}\, R_1(\mathbf{x}), \quad \textit{or} \;\; \Phi(f_0)(\mathbf{x}, z=0) \neq 0 \in \mathbb{F}\,.$$

Proof. We assume that $\sum_{i=1}^{k} T_{i,0} = f_0$ and $T_{k,0} \neq 0$. Start with division and derivation as follows:

$$\begin{aligned}\sum_{i\in[k]} \Phi(T_{i,0}) = \Phi(f_0) &\iff \sum_{i\in[k-1]} \frac{\Phi(T_{i,0})}{\Phi(T_{k,0})} + 1 = \frac{\Phi(f_0)}{\Phi(T_{k,0})} \\ &\implies \sum_{i\in[k-1]} \partial_z\left(\frac{\Phi(T_{i,0})}{\Phi(T_{k,0})}\right) = \partial_z\left(\frac{\Phi(f_0)}{\Phi(T_{k,0})}\right) \\ &\iff \sum_{i=1}^{k-1} \frac{\Phi(T_{i,0})}{\Phi(T_{k,0})} \cdot \mathrm{dlog}_z\left(\frac{\Phi(T_{i,0})}{\Phi(T_{k,0})}\right) = \partial_z\left(\frac{\Phi(f_0)}{\Phi(T_{k,0})}\right). \qquad (6.1)\end{aligned}$$

Here onwards we use dlog to denote dlog_z, unless stated otherwise. Note that, Equation (6.1) holds over $R_1(\mathbf{x})$. For all $i \in [k-1]$, define $T_{i,1} \in \mathbb{F}(\mathbf{x}, z)$ such that

$$T_{i,1} = \left(\frac{\Phi(T_{i,0})}{\Phi(T_{k,0})}\right) \cdot \mathrm{dlog}\left(\frac{\Phi(T_{i,0})}{\Phi(T_{k,0})}\right) \text{ over } R_1(\mathbf{x}).$$

Further, $f_1 \coloneqq \sum_{i\in[k-1]} T_{i,1}$ over $R_1(\mathbf{x})$. Due the the safe non-zero division, the expressions are well-defined.

Size Blow-up. To prove that f_1 is in $\mathsf{Gen}(k-1, \cdot)$, we have to understand the linearising effect of dlog on $(\Phi(T_{i,0}) / T_{k,0})$ (refer discussion after Definition 2.9). Taking the linearisation of the product into account, observe that it suffices to study its effect on $\Sigma\wedge$. From the properties of Φ, we know that such a $\Sigma\wedge$ is of the form $A - z \cdot B$, where $A \in \mathbb{F}\backslash\{0\}$ and B is a univariate polynomial in $R_1[\mathbf{x}]$. Using the power series identity (such

as Equation (2.2)) we have the following in $R_1(\mathbf{x})$:

$$\begin{aligned}\operatorname{dlog}(A - z\cdot B) &= \frac{-\partial_z(z\cdot B)}{A(1 - z\cdot B/A)}\\ &= \frac{-\partial_z(z\cdot B)}{A}\cdot\sum_{j=0}^{d_1-1}\left(\frac{z\cdot B}{A}\right)^j\\ &=: C_1\cdot\sum_{j=0}^{d_1-1}(C_2)^j \end{aligned} \tag{6.2}$$

Observe that C_1 and $(C_2)^j$ have trivial $\wedge\Sigma\wedge$ circuits, of size $O(s_0)$ and $O(j\cdot s_0)$ respectively. Using the multiplicative closure of $\Sigma\wedge\Sigma\wedge$ circuit (Lemma 1.35), we obtain the final $\Sigma\wedge\Sigma\wedge$ circuit computing $\operatorname{dlog}(A - z\cdot B)$, of size $s_1 = O(d^3\cdot s)$ using the fact that $d_1 = d$. Clearly, the syntactic degree blows up to $O(d^2)$.

Non-zeroness Preservation. We now prove that a single step of DiDI reduces the identity testing problem to $\mathsf{Gen}(k-1,\cdot)$. Note that $f_1 \neq 0$ implies $\operatorname{val}_z(f_1) < d =: d_1$. By assumption, $\Phi(T_{k,0})$ is invertible over $R_1(\mathbf{x})$. If $f_1 = 0$ over $R_1(\mathbf{x})$, it implies –

1. Either, $\Phi(f_0)/\Phi(T_{k,0})$ is z-free. Then $\Phi(f_0)/\Phi(T_{k,0}) \in \mathbb{F}(\mathbf{x})$, which further implies it is in $\mathbb{F}$, because of the map Φ (z-free implies $\mathbf{x}$-free, by substituting $z = 0$). Also, note that $f_0, T_{k,0} \neq 0$ implies $\Phi(f_0)/\Phi(T_{k,0})$ is a *nonzero* element in $\mathbb{F}$. Thus, it suffices to test non-zeroness of $\Phi(f_0)(\mathbf{x}, z = 0)$.

2. Or, $\partial_z(\Phi(f_0)/\Phi(T_{k,0})) = z^{d_1}\cdot p$ where $p \in \mathbb{F}(z,\mathbf{x})$ s.t. $\operatorname{val}_z(p) \geq 0$. Using Proposition 2.13 we get that $p \in \mathbb{F}(\mathbf{x})[\![z]\!]$. Hence, $\Phi(f_0)/\Phi(T_{k,0}) = z^{d_1+1}\cdot q$ where $q \in F(\mathbf{x})[[z]]$, i.e.

$$\frac{\Phi(f_0)}{\Phi(T_{k,0})} \in \langle z^{d_1+1}\rangle_{\mathbb{F}(\mathbf{x})[[z]]} \implies \operatorname{val}_z(\Phi(f_0)) \geq d+1,$$

   a contradiction.

Conversely, it is obvious that $f_0 = 0$ implies $f_1 = 0$. Thus, we have proved the following

$$\Phi(f_0) \neq 0 \text{ over } \mathbb{F}[\mathbf{x}] \iff f_1 \neq 0 \text{ over } R_1(\mathbf{x}), \text{ or } \Phi(f_0)(\mathbf{x}, z = 0) \neq 0 \in \mathbb{F}\,.$$

□

In Lemma 6.8 formally describes the effect of single Division and Derivation on the model. It further establishes the non-zeroness preserving property of DiDI, which is essential for reducing the identity testing problem on reduced fan-in. In the following lemma we show that it continues to hold with subsequent application of division and derivation.

Lemma 6.9 (Inductive Reduction)**.** *For a positive integer* $j < k$*, let* $f_j \in \mathsf{Gen}(k-j, s_j)$ *of syntactic degree* D_j*, over* $R_j := \mathbb{F}[z]\langle z^{d_j}\rangle$*, where* $d_j \le d$*.*

- *(Valuation) Suppose* $f_j = \sum_i T_{i,j}$*. For all* $i \in [k-j]$ *assume that* $T_{i,j} \neq 0$*, and* $\nu_{i,j} := \mathrm{val}_z(T_{i,j}) \ge 0$*. Without loss of generality assume* $\mathrm{val}_z(T_{k-j,j})$ *is the minimal valuation.*
- *(Invertibility) Further, assume that* $U_{i,j}(\mathbf{x}, z=0)$ *and* $V_{i,j}(\mathbf{x}, z=0) \in \mathbb{F}\backslash\{0\}$*.*

Then, $f_{j+1} := \partial_z(f_j/T_{k-j,j}) \in \mathsf{Gen}(k-j-1, s_{j+1})$ *of syntactic degree* $D_{j+1} = O(d \cdot D_j)$ *over* $R_{j+1} := \mathbb{F}[z]/\langle z^{d_{j+1}}\rangle$*,* $d_{j+1} \le d_j \le d$*, and* $s_{j+1} = s_j^7 \cdot d^{O(j)}$*. The valuation and invertibility properties above continue to hold with respect to* $f_{j+1} = \sum_i T_{i,j+1}$*. Moreover,*

$$f_j \neq 0 \ \text{over}\ R(\mathbf{x}) \iff f_{j+1} \neq 0 \ \text{over}\ R_{j+1}(\mathbf{x}),\ \text{or},\ 0 \neq \left(\frac{f_j}{T_{k-j,j}}\right)\Big|_{z=0} \in \mathbb{F}(\mathbf{x})\ .$$

Proof. We proceed as earlier by applying DiDI, without the map. We effectively reduce from top fan-in $(k-j)$ to $(k-j-1)$ using division and derivation. For all $i \in [k-j]$, let $\nu_{i,j} := \mathrm{val}_z(T_{i,j})$. Note that, since $\mathrm{val}_z(U_{i,j}) = \mathrm{val}_z(V_{i,j}) = 0$, without loss of generality we assume that

$$\min_i \nu_{i,j} = \min_i \mathrm{val}_z\left(\frac{P_{i,j}}{Q_{i,j}}\right) = \nu_{k-j,j}.$$

Further, for all $i \in [k-j]$, $0 \le \nu_{i,j} < d_j$, because $\nu_{i,j} = d_j$ would imply that $T_{i,j} = 0$ over $R_j(x)$. We divide by $T_{k-j,j}$ which has the minimum valuation and then derive as before:

$$\begin{aligned}
\sum_{i\in[k-j]} T_{i,j} = f_j &\iff \sum_{i\in[k-j-1]} \frac{T_{i,j}}{T_{k-j,j}} + 1 = \frac{f_j}{T_{k-j,j}} \\
&\implies \sum_{i\in[k-j-1]} \partial_z\left(\frac{T_{i,j}}{T_{k-j,j}}\right) = \partial_z\left(\frac{f_j}{T_{k-j,j}}\right) \\
&\iff \sum_{i=1}^{k-j-1} \frac{T_{i,j}}{T_{k-j,j}} \cdot \mathrm{dlog}\left(\frac{T_{i,j}}{T_{k-j,j}}\right) = \partial_z\left(\frac{f_j}{T_{k-j,j}}\right) \qquad (6.3)
\end{aligned}$$

Define $d_{j+1} := d_j - \nu_{k-j,j} - 1$. For all $i \in [k-1]$, define $T_{i,j+1} \in \mathbb{F}(\boldsymbol{x}, z)$ such that

$$T_{i,j+1} = \left(\frac{T_{i,j}}{T_{k-j,j}}\right) \cdot \operatorname{dlog}\left(\frac{T_{i,j}}{T_{k-j,j}}\right) \text{ over } R_{j+1}(\boldsymbol{x}).$$

Definability. Division by minimum valuation implies that $\operatorname{val}(f_j) \geq \nu_{k-j,j}$, and thus $T_{i,j+1}$ and f_{j+1} are all well-defined over $R_{j+1}(\boldsymbol{x})$. When an identity holds over mod z^{d_j}, then it must hold over mod $z^{d_{j+1}}$ as well, since $d_{j+1} \leq d_j$. Therefore, Equation (6.3) holds over $R_{j+1}(\boldsymbol{x})$ and

$$\sum_{i=1}^{k-j-1} T_{i,j+1} = f_{j+1} \text{ over } R_{j+1}(\boldsymbol{x}).$$

Valuation. Since we divide by the min val, by definition it implies $\operatorname{val}_z(T_{i,j+1}) \geq 0$. Further, we claim that min val computation in DiDI is easy. Recall from the definition of valuation

$$\min_i \operatorname{val}_z\left(\frac{P_{i,j}}{Q_{i,j}}\right) = \min_i(\operatorname{val}_z(P_{i,j}) - \operatorname{val}_z(P_{i,j})).$$

Therefore, for min val we compute $\operatorname{val}_z(P_{i,j})$ and $\operatorname{val}_z(Q_{i,j})$ for all $i \in [k-j]$. Using Lemma 1.36 we known $\operatorname{coef}_{z^e}(P_{i,j})$ and $\operatorname{coef}_{z^e}(Q_{i,j})$ are in $\Sigma\wedge\Sigma\wedge$ over $F[\boldsymbol{x}]$. We keep track of degree of z and thus interpolate to find the minimum $e < d_j$ such that the computed coefficients are non-zero, giving the respective valuations.

Invertibility. To show that $f_{j+1} \in \mathsf{Gen}(k-j-1, \cdot)$, it remains to show that $\Pi\Sigma\wedge$ circuit in $T_{i,j+1}$ are invertible. Note that this was implicit in Lemma 6.8. Borrowing the notations from Definition 2.10, we simplify $T_{i,j+1}$ as follows:

$$\frac{T_{i,j}}{T_{k-j,j}} = \frac{U_{i,j} \cdot V_{k-j,j}}{V_{i,j} \cdot U_{k-j,j}} \cdot \frac{P_{i,j} \cdot Q_{k-j,j}}{Q_{i,j} \cdot P_{k-j,j}}. \tag{6.4}$$

Define $U_{i,j+1} := U_{i,j} \cdot V_{k-j,j}$ and $V_{i,j+1} := V_{i,j} \cdot U_{k-j,j}$. Then clearly

$$U_{i,j+1}(\boldsymbol{x}, z=0), V_{i,j+1}(\boldsymbol{x}, z=0) \in \mathbb{F}\backslash\{0\}.$$

The P's and the Q's will be analysed with the action of dlog on Equation (6.4) in the upcoming discussion on the size blow up. We will essentially bring together $\Sigma\wedge\Sigma\wedge/\Sigma\wedge\Sigma\wedge$ to define the final $P_{i,j+1}$ and $Q_{i,j+1}$.

Size Bound. The size analysis is different from Lemma 6.8 for two reasons. First, we don't use the DiDI homomorphism in the inductive reduction, and second, the analysis

of bloated model demands more care. We begin the analysis by considering the overall expression.

$$\begin{aligned} T_{i,j+1} &= \frac{T_{i,j}}{T_{k-j,j}} \cdot \operatorname{dlog}\left(\frac{T_{i,j}}{T_{k-j,j}}\right) \\ &= \underbrace{\boxed{\frac{U_{i,j+1}}{V_{i,j+1}}}}_{(1)} \cdot \underbrace{\boxed{\boxed{\frac{P_{i,j}\cdot Q_{k-j,j}}{Q_{i,j}\cdot P_{k-j,j}}}}}_{(2)} \cdot \underbrace{\boxed{\operatorname{dlog}\left(\frac{T_{i,j}}{T_{k-j,j}}\right)}}_{3} \Bigg]_{(4)} \\ &= \frac{U_{i,j+1}}{V_{i,j+1}} \cdot \frac{P_{i,j+1}}{Q_{i,j+1}} \end{aligned} \tag{6.5}$$

It is easy to note that in the equation above (1) is computable by $\Pi\Sigma\wedge/\Pi\Sigma\wedge$ with constant blow up in size. Similarly, (2) is computable by

$$\frac{\Pi^{[2]}(\Sigma\wedge\Sigma\wedge)}{\Pi^{[2]}(\Sigma\wedge\Sigma\wedge)} \subseteq \frac{\Sigma\wedge\Sigma\wedge}{\Sigma\wedge\Sigma\wedge}$$

of size $(D_j \cdot s_j^2)$, using the properties. And finally, because of linearisation, $\operatorname{dlog}(\cdot)$ is computable by:

$$\sum \operatorname{dlog}(\Sigma\wedge) \pm \sum^{[4]} \operatorname{dlog}(\Sigma\wedge\Sigma\wedge)$$

Using analysis similar to Equation (6.2) obtain a single $\Sigma\wedge\Sigma\wedge$ circuit of size $O(D_j^2 \cdot d_j \cdot s_j) \leq O(D_j^3 \cdot s_j)$ for the first summand. Since $\Sigma\wedge\Sigma\wedge$ is closed under derivation (Lemma 1.37), $\operatorname{dlog}(\Sigma\wedge\Sigma\wedge)$ is computable by $\Sigma\wedge\Sigma\wedge/\Sigma\wedge\Sigma\wedge$ circuit of size $O(D_j^2 \cdot s_j)$ and syntactic degree $O(D_j)$. Re-indexing the sum, and using additive closure, we obtain a single $\Sigma\wedge\Sigma\wedge/\Sigma\wedge\Sigma\wedge$ circuit of size $O(D_j^12 \cdot s_j^4)$. Lastly adding it to $\Sigma\wedge\Sigma\wedge$ gives the final $\Sigma\wedge\Sigma\wedge/\Sigma\wedge\Sigma\wedge$ for (3) in Equation (6.5) of size $O(D_j^{16} \cdot d \cdot s_j^5)$.

Adding (2) and (3), once again by additive closure, gives $\Sigma\wedge\Sigma\wedge/\Sigma\wedge\Sigma\wedge$ for (4) of size $s_{j+1} \coloneqq O(D_j^{O(1)} \cdot d \cdot s_j^7)$, and syntactic degree $D_{j+1} \coloneqq O(d \cdot D_j)$. □

We are now ready to bring the pieces together for the PIT algorithm and prove our theorem of this chapter.

Theorem 6.1 (Restated). *There is a deterministic, whitebox PIT algorithm for* $\Sigma^{[k]}\Pi\Sigma\wedge(s)$ *over* $\mathbb{F}[\mathbf{x}]$ *running in time* $s^{O(k\,7^k)}$.

Proof. We initiate by defining $f_0 \coloneqq f$ as before, such that $T_{i,0} \coloneqq T_i$ where $T_i \in \Pi\Sigma\wedge$ and $f_0 = \sum_i T_{i,0}$. Apply non-zeroness preserving DiDI map from Definition 6.7. We proceed with recursively reducing the identity testing from top fan-in k to $k-1$ using Lemma 6.8. Note: $k = 1$ is trivial. Followed by repeatedly using Lemma 6.9, $k-1$ times, gives $f_{k-1} = T_{1,k-1} \in \mathsf{Gen}(1, s_{k-1})$ where

$$s_{k-1} = O(d^{O(k-1)} \cdot s_{k-2}^7) \leq s^{O(k \cdot 7^k)}.$$

The degree of z in the numerator and denominator can be bounded by

$$D_{k-1} = O(d \cdot D_{k-1}) = d^{O(k)}.$$

As remarked earlier, even after reducing the identity testing to top fan-in 1, we cannot directly perform identity testing on constituent circuits of $\mathsf{Gen}(1, \cdot)$ because of rational ring $R_{k-1}(\boldsymbol{x})$. However, note that $\mathrm{val}(T_{1,k-1}) \geq 0$, then using Proposition 2.13 we know that $T_{1,k-1} \in \mathbb{F}(\boldsymbol{x})[\![z]\!]$. Therefore, it suffices to do identity testing on the first term of the power-series: $T_{1,k-1}(\boldsymbol{x}, z = 0)$, over $\mathbb{F}(\boldsymbol{x})$.

The *invertibility* of Lemma 6.9 guarantees that $\Pi\Sigma\wedge$ part of $\mathsf{Gen}(1, s_{k-1})$, when evaluated at $z = 0$, computing $T_{1,k-1}(\boldsymbol{x}, z = 0)$, is a non-zero field element. Therefore, it suffices to test identity of $\Sigma\wedge\Sigma\wedge/\Sigma\wedge\Sigma\wedge$, which perhaps could be undefined. Given that we keep track of degree of z in numerator and denominator, we extract the appropriate power of z from each to make $\Sigma\wedge\Sigma\wedge/\Sigma\wedge\Sigma\wedge$ well-defined. Using Lemma 1.36, find the minimum $\tilde{e} < d_{k-1}$, such that $\mathrm{coef}_{z^e}(P_{1,k-1})$ and $\mathrm{coef}_{z^e}(Q_{1,k-1})$ is non-zero (Recall the notations from Definition 2.10). Moreover, the coefficients are computable by $\Sigma\wedge\Sigma\wedge$ of size $s^{O(k \cdot 7^k)}$ over $F[\boldsymbol{x}]$. Cancelling the power of z from the numerator and the denominator will leave a well-defined expression for identity testing of the form $z^e \cdot (\Sigma\wedge\Sigma\wedge/\Sigma\wedge\Sigma\wedge)$, where $e < \tilde{e}$. Therefore, it suffices to test $z^e \cdot (\Sigma\wedge\Sigma\wedge)$ over $\mathbb{F}[\boldsymbol{x}]$. Use poly-time whitebox PIT algorithm from Lemma 6.6 to test $\Sigma\wedge\Sigma\wedge$, and the trivial PIT algorithm from Lemma 6.3 to test z^e using the degree bound.

The *non-zeroness* preserving reduction in Lemma 6.9 shows that:

$$f \neq 0, \text{ over } \mathbb{F}[\boldsymbol{x}] \iff f_{k-1} \neq 0, \text{ over } R_{k-1}(\boldsymbol{x}),$$
$$\text{or } \exists\ 0 \leq i \leq j-1 \text{ such that } \left(\frac{f_i}{T_{k-i,i}}\right)_{\boldsymbol{x}, z=0} \neq 0, \text{ over } \mathbb{F}(\boldsymbol{x})$$

Our algorithm constructively computes f_{k-1} iteratively, together with it's constituent terms. In the preceding paragraph we have described our approach of testing f_{k-1}. However, the the $z = 0$ evaluation above could *short-circuit* the algorithm much before it reaches the final $j = k-1$ in the top fan-in reduction. At a certain step $1 \leq j \leq k-1$, we know that:

$$\frac{f_j}{T_{k-j,j}} \in \left(\frac{\Sigma\wedge\Sigma\wedge}{\Sigma\wedge\Sigma\wedge}\right).$$

However, the $z = 0$ evaluation of it may not be well-defined. Once again, as before, we interpolate using the degree bound of z to make the identity testing work on $\Sigma\wedge\Sigma\wedge$ of size $s_j^{O(k-j)} \leq (sd)^{O(j(k-j)7^j)}$ using Lemma 6.6. To bound the final time complexity, we need to consider the j which maximizes the exponent. Since $\max_{j\in[k-1]} j\cdot(k-j)\cdot 7^j = (k-1)\cdot 7^{k-1}$ by differentiating and computing maxima, the final complexity of whitebox identity testing is $s^{O(k\cdot 7^k)}$. □

Remark 6.10.

1. *Bit Complexity.* It is routine to show that the bit-complexity is bounded. The blowup due to dlog is bounded polynomially. While using Lemma 1.35 (using Lemma 1.34), we may need to go to a field extension of at most $s^{O(k)}$. Also, Theorem 6.4 and Lemma 6.6 computations blowup bit-complexity polynomially.

2. The above method does *not* give whitebox PIT (in poly-time) for $\Sigma^{[[k]]}\Pi\Sigma\Pi^{[]}[\delta]$, as we donot know poly-time whitebox PIT for $\Sigma\wedge\Sigma\Pi^{[\delta]}$. However, the above methods do show that whitebox PIT for $\Sigma^{[[k]]}\Pi\Sigma\Pi^{[]}[\delta]$ polynomially *reduces* to whitebox PIT for $\Sigma\wedge\Sigma\Pi^{[\delta]}$.

3. The above proof works when the characteristic is greater than d. This is because the non-zeroness remains *preserved* after derivation with respect to z.

Chapter 7

Identity Testing of $\overline{\text{Depth-4}}$ Circuits

> Algebra is generous, she often gives more than is asked of her.
>
> *Jean-Pierre Serre*

Very few hitting sets are known for border classes, meanwhile most of the hitting sets for classical complexity classes do not directly work for their respective border classes. Mulmuley [Mul17] asked the question of constructing an efficient hitting set for $\overline{\mathsf{VP}}$. Forbes and Shpilka [FS18] gave a PSPACE algorithm over the field $\mathbb{C}$. In [GSS19], the authors extended this result to *any* field. Very few better hitting set constructions are known for the restricted border classes, for example poly-time hitting set for $\overline{\Pi\Sigma} = \Pi\Sigma$ [BT88, KS01], quasi-poly hitting set for $\overline{\Sigma\wedge\Sigma} \subseteq \overline{\mathsf{ARO}} \subseteq \overline{\text{ROABP}}$ [FS13b, AGKS15, GKS17] and poly-time hitting set for the border of a restricted sum of log-variate ROABPs [BS21].

We formally state the best known hitting set for ARO, which is also *robust* because of complete de-bordering of model Lemma 2.4.

Theorem 7.1 (Hitting Set for ARO)**.** *For a class of* n *variate and degree* d *polynomials computable by size* s *ARO, there exists an explicit hitting set of size* $\mathsf{s}^{\mathrm{O}(\log\log \mathsf{s})}$.

The following lemma is useful in combining the hitting sets. When the hitting sets of classes $\mathcal{C}_1$ and $\mathcal{C}_2$ is known, then the lemma combines the two to give a hitting set for $\mathcal{C}_1 \cdot \mathcal{C}_2$.

Lemma 7.2 (Product of Hitting Sets)**.** *Let $\mathcal{H}_1, \mathcal{H}_2 \in \mathbb{F}^n$ of size s_1 and s_2 respectively be the hitting set of a class of n variate and degree d polynomials C_1 and C_2 respectively. Then there is an explicit hitting set $\mathcal{H}$ of size $s_1 \cdot s_2 \cdot O(d)$ for the class of polynomials in $\mathcal{C}_1 \cdot \mathcal{C}_2$.*

Proof. Consider a polynomial $f = f_1 \cdot f_2 \in \mathcal{C}_1 \cdot \mathcal{C}_2$ such that $f_1 \in \mathcal{C}_1$ and $f_2 \in \mathcal{C}_2$. For all $\alpha \in \mathcal{H}_1$ and $\beta \in \mathcal{H}_2$, define a set H of points $c = \alpha + z \cdot \beta$, where z is a formal variable. Then there exists a point $c \in H$ such that $f(c) \neq 0$, because shifting and scaling preserves non-zeroness. Moreover, $\deg(f(c)) \leq O(d)$. Then using Lemma 6.3 we get the required hitting set $\mathcal{H}$ of size $s_1 \cdot s_2 \cdot O(d)$. □

7.1 Identity Testing $\overline{\Sigma\wedge\Sigma\Pi^{[\delta]}}$ Circuits

Forbes used *rank* based methods to give a quasipolynomial-time algorithm for blackbox PIT of $\Sigma\wedge\Sigma\Pi[\delta]$ circuits [For15]. Due to the algebraic nature of approximation, most of the rank based results extend and continue to hold over border classes [Gro15b]. In this section we will make observations to extend Forbes PIT algorithm to border PIT of $\overline{\Sigma\wedge\Sigma\Pi[\delta]}$ circuits. We encourage interested readers to refer [For15] for details. We need some definitions and properties to describe the rank approach to PIT.

Shifted Partial Derivative measure $x^{\leq \ell} \cdot \partial_{\leq m}$ is a linear operator first introduced in [Kay12, GKKS14] as:

$$x^{\leq \ell} \cdot \partial_{\leq m}(g) := \left(x^c \cdot \partial_{x^b}(g)\right)_{\deg x^c \leq \ell, \deg x^b \leq m}.$$

It was shown in [For15] that the rank of shifted partial derivatives of a polynomial computed by $\Sigma\wedge\Sigma\Pi^{[\delta]}$ is small. We state the result formally in the next lemma. Consider the fractional field $\mathcal{R} := \mathbb{F}(\varepsilon)$.

Lemma 7.3 (Measure upper bound)**.** *Let $g(\varepsilon, x) \in \mathcal{R}[x_1, \ldots, x_n]$ be computable by $\Sigma\wedge\Sigma\Pi^{[\delta]}$ circuit of size s. Then*

$$\operatorname{rank}\operatorname{span} x^{\leq \ell} \cdot \partial_{\leq m}(g) \leq s \cdot m \cdot \binom{n + (\delta-1)m + \ell}{(\delta-1)m + \ell}.$$

Under any *monomial ordering*, the *trailing monomial* of g denoted by $\mathrm{TM}(g)$ is the smallest monomial in the support set $\mathrm{supp}(g) := \{x^a : \mathrm{coef}_{x^a}(g) \neq 0\}$. It was observed in

[For15] that, the rank of shifted partials can be lower bounded using the trailing monomial (ref [CLO15, Section 2]).

Proposition 7.4 (Measure the trailing monomial)**.** *Consider* $g \in \mathcal{R}[\mathbf{x}]$*. For any* $\ell, m \geq 0$*,*

$$\operatorname{rank}\operatorname{span}\mathbf{x}^{\leq \ell} \cdot \partial_{\leq m}(g) \geq \operatorname{rank}\operatorname{span}\mathbf{x}^{\leq \ell} \cdot \partial_{\leq m}(\mathrm{TM}(g))\,.$$

For fields of characteristic zero, a lower bound on a monomial was obtained.

Lemma 7.5 (Monomial lower bound)**.** *Consider a monomial* $\mathbf{x}^{\mathbf{a}} \in \mathcal{R}[x_1, \ldots, x_n]$*. Then,*

$$\operatorname{rank}\operatorname{span}\left(\mathbf{x}^{\leq \ell} \cdot \partial_{\leq m}(\mathbf{x}^{\mathbf{a}})\right) \geq \binom{\eta}{m}\binom{\eta - m + \ell}{\ell}$$

where $\eta := |\operatorname{supp}(\mathbf{x}^{\mathbf{a}})|$*.*

The results above were combined to show that the trailing monomial of polynomials computed by $\Sigma\wedge\Sigma\Pi^{[\delta]}$ circuits have logarithmically small support size. Using the same idea we show that if such a polynomial approximates f (ref Definition 2.1), then the support of $\mathrm{TM}(f)$ is also small. We formalize this in the next lemma.

Lemma 7.6 (Trailing monomial support)**.** *Let* $g(\varepsilon, \mathbf{x}) \in \mathcal{R}[x_1, \ldots, x_n]$ *be computable by a* $\Sigma\wedge\Sigma\Pi^{[\delta]}$ *circuit of size* s *such that* $g = f + \varepsilon \cdot Q$ *where* $f \in \mathbb{F}[\mathbf{x}]$ *and* $Q \in \mathbb{F}[\varepsilon, \mathbf{x}]$*. Let* $\eta := |\operatorname{supp}(\mathrm{TM}(f))|$*. Then* $\eta = O(\delta \log s)$*.*

Proof. Let $\mathbf{x}^{\mathbf{a}} := \mathrm{TM}(f)$ and $S := \{i \mid a_i \neq 0\}$. Define a substitution map ρ such that $x_i \to y_i$ for $i \in S$ and $x_i \to 0$ for $i \notin S$. It is easy to observe that $\mathrm{TM}(\rho(f)) = \rho(\mathrm{TM}(f)) = \mathbf{y}^{\mathbf{a}}$. Using Lemma 7.3 we know:

$$\operatorname{rank}_{\mathcal{R}} \mathbf{y}^{\leq \ell}\partial_{\leq m}(\rho(g)) \;\leq\; s \cdot m \cdot \binom{\eta + (\delta - 1)m + \ell}{(\delta - 1)m + \ell} =: R\,.$$

To obtain the upper bound for $\rho(f)$ we use the following claim.

Claim 7.7. $\operatorname{rank}_{\mathbb{F}} \mathbf{y}^{\leq \ell}\partial_{\leq m}(\rho(f)) \leq R$*.*

Proof. Define the *coefficient matrix* $N(\rho(g))$ with respect to $\mathbf{y}^{\leq \ell}\partial_{\leq m}(\rho(g))$ as follows: the rows are indexed by the operators $\mathbf{y}^{=\ell_i}\,\partial_{\mathbf{y}^{=m_i}}$, while the columns are indexed by the terms present in $\rho(g)$; and the entries are the respective operator-action on the respective term in $\rho(g)$. Note that $\operatorname{rank}_{\mathbb{F}(\varepsilon)} N(\rho(g)) \leq R$. Similarly define $N(\rho(f))$ with respect to $\mathbf{y}^{\leq \ell}\partial_{\leq m}(\rho(f))$, then it suffices to show that $\operatorname{rank}_{\mathbb{F}} N(\rho(f)) \leq R$.

For any $r > R$, let $\mathcal{N}(\rho(g))$ be a $r \times r$ sub-matrix of $N(\rho(g))$. The rank bound ensures: $\det \mathcal{N}(\rho(g)) = 0$. This will remain true under the limit $\varepsilon = 0$; thus, $\det(\mathcal{N}(\rho(f))) = 0$. Since $r > R$ was arbitrary and linear dependence is preserved, we deduce that $\operatorname{rank}_{\mathbb{F}} N(\rho(f)) \leq R$. □

For lower bound, recall $\mathbf{y}^{\mathbf{a}} = \mathrm{TM}(\rho(f))$. Then, by Proposition 7.4 and Lemma 7.5, we get:

$$\operatorname{rank}_{\mathbb{F}} \mathbf{y}^{\leq \ell} \partial_{\leq m}(\rho(f)) \geq \binom{n}{m}\binom{n-m+\ell}{\ell}. \tag{7.1}$$

Comparing Claim 7.7 and Equation (7.1) we get:

$$s \geq \frac{1}{m} \cdot \binom{n}{m} \cdot \binom{n-m+\ell}{\ell} \Big/ \binom{n+(\delta-1)m+\ell}{(\delta-1)m+\ell}.$$

For $\ell := (\delta-1)(n+(\delta-1)m)$ and $m := \lfloor n/e^3\delta \rfloor$, [For15, Lem.A.6] showed $n \leq O(\delta \log s)$. □

The existence of a small support monomial in a polynomial which is being approximated, is a structural result which will help in constructing a hitting set for this larger class. The idea is to use a map that reduces the number of variables to the size of the support of the trailing monomial, and then invoke Lemma 6.3.

Theorem 7.8 (Hitting set for $\overline{\Sigma\wedge\Sigma\Pi^{[\delta]}}$)**.** *For the class of* n*-variate, degree* d *polynomials* approximated *by* $\Sigma\wedge\Sigma\Pi[\delta]$ *circuits of size* s*, there is an explicit hitting set* $\mathcal{H} \subseteq \mathbb{F}^n$ *of size* $s^{O(\delta \log s)}$.

Proof. Let $g(\varepsilon, \mathbf{x}) \in \mathcal{R}[x_1, \ldots, x_n]$ be computable by a $\Sigma\wedge\Sigma\Pi[\delta]$ circuit of size s such that $g =: f + \varepsilon \cdot Q$, where $f \in \mathbb{F}[\mathbf{x}]$ and $Q \in \mathbb{F}[\varepsilon, \mathbf{x}]$. Then Lemma 7.6 shows that there exists a monomial $\mathbf{x}^{\mathbf{a}}$ of f such that $n := |\operatorname{supp}(\mathbf{x}^{\mathbf{a}})| = O(\delta \log s)$.

Let $S \in \binom{[n]}{n}$. Define a substitution map ρ_S such that $x_i \to y_i$ for $i \in S$ and $x_i \to 0$ for $i \notin S$. Note that, under this substitution non-zeroness of f is preserved for some S; because monomials of support $S \supseteq \operatorname{supp}(\mathbf{x}^{\mathbf{a}})$ will survive for instance. Essentially $\rho_S(f)$ is an n-variate degree-d polynomial, for which Lemma 6.3 gives a trivial hitting set of size $O(d^n)$. Therefore, with respect to S we get a hitting set $\mathcal{H}_S$ of size $O(d^n)$. To finish, we

do this for all such S, to obtain the final hitting set $\mathcal{H}$ of size:

$$\binom{n}{\eta} \cdot O\left(d^{\eta}\right) \leq O((nd)^{\eta}).$$

□

We obtained this result without de-bordering the circuit. In the upcoming sections we will another example of such PIT result without de-bordering.

7.2 De-randomizing PIT using DiDIL

In this section we will see an amalgamation of DiDIL (Section 2.3) and DiDI (Section 6.2) to obtain efficient hitting sets for $\overline{\Sigma^{[k]}\Pi\Sigma\Pi^{[\delta]}}$ and $\overline{\Sigma^{[k]}\Pi\Sigma\wedge}$ circuits.

For brevity, we denote these two types of depth-4 circuits by $\Sigma^{[k]}\Pi\Sigma\Upsilon$ where Υ denotes a layer of operator fluid gates. After the analysis, Υ gates will be replaces by $\wedge$ or Π^{δ} gates for final hitting sets. To invoke DiDIL, for PIT on $\overline{\Sigma^{[k]}\Pi\Sigma\wedge}$ we can once again use $\mathsf{Gen}(k)$ from Definition 2.10. However, for $\overline{\Sigma^{[k]}\Pi\Sigma\Pi^{[\delta]}}$ we need a slightly tweaked bloated model. We redefine $\mathsf{Gen}(k)$ using the operator fluid Υ gates for this section .

Definition 7.9 (DiDIL model for PIT)**.** *Let* $R(\mathbf{x})$ *be the ring of rational functions. Denote* $\mathsf{Gen}(k, s)$ *as a class of circuits* $\mathcal{C}$ *over* R *which computes polynomial* $f(\mathbf{x}) \in R(x_1, \ldots, x_n)$ *of the form:* $f = \sum_{i=1}^{k} T_i$ *such that*

$$T_i = \left(\frac{U_i}{V_i}\right) \cdot \left(\frac{P_i}{Q_i}\right),$$

where $U_i, V_i \in \Pi\Sigma\Upsilon$ *and* $P_i, Q_i \in \Sigma\wedge\Sigma\Upsilon$ *are polynomials in* $R[x_1, \ldots, x_n]$.

The size of the circuit is defined naturally as $\operatorname{size}(\mathcal{C}) \coloneqq \sum_{i=1}^{k} \operatorname{size}(T_i) \leq s$ *where*

$$\operatorname{size}(T_i) = \operatorname{size}(U_i) + \operatorname{size}(V_i) + \operatorname{size}(P_i) + \operatorname{size}(Q_i).$$

The syntactic degree of the circuit is defined as the maximum syntactic degree of the numerator and denominator.

The DiDIL homomorphism from Definition 2.14 suffices for $\overline{\Sigma^{[k]}\Pi\Sigma\wedge}$, however for $\overline{\Sigma^{[k]}\Pi\Sigma\Pi^{[\delta]}}$ we obtain the shift $\mathbf{a} \in \mathbb{F}^n$ for the map using the hitting set of product of sparse

polynomials from Lemma 6.5. We start with *First Reduction* similar to Lemma 2.15 and Lemma 6.8.

Lemma 7.10 (First Reduction)**.** *Let* $\Phi(f_0) \in \overline{\Sigma^{[k]}\Pi\Sigma\Upsilon}(s_0)$ *over* $R_0 := \mathbb{F}[z]/\langle z^d \rangle$*, where* d *is the syntactic degree. Then*

$$f_1 := \partial_z\left(\frac{\Phi(f_0)}{t_{k,0}}\right) \in \overline{\mathsf{Gen}(k-1, s_1)}$$

over $R_1 := \mathbb{F}[z]/\langle z^{d_1} \rangle$*, where* $t_{k,0} \in R_0(\mathbf{x})$*,* $d_1 \leq d$*, and* $s_1 = O(d^3 \cdot s)$ *when* $\Upsilon = \wedge$*, while* $s_1 = O(3^\delta \cdot s)$ *when* $\Upsilon = \Pi^{[\delta]}$ *. Moreover,*

$$\Phi(f_0) \neq 0 \ \text{ over } \mathbb{F}[\mathbf{x}] \iff f_1 \neq 0 \ \text{ over } R_1(\mathbf{x}), \ \text{ or } \ \Phi(f_0)(\mathbf{x}, z = 0) \neq 0 \in \mathbb{F}\,.$$

Proof. We break the proof in two parts based on the choice of operator fluid gates $\Upsilon = \{\wedge, \Pi^{[\delta]}\}$.

When $\Upsilon = \wedge$. The proof of the first part is same as that of Lemma 2.15. Further, the proof of second part is similar to that of Lemma 6.8, where $\Phi(T_{k,0})$ is appropriately replaced by $t_{k,0}$.

When $\Upsilon = \Pi^{[\delta]}$. The proof of both the parts remains the same as earlier. However, the size analysis changes slightly. Refer the proof of Lemma 2.15 for notations. Observe that, $\mathrm{size}(A - z \cdot B) \in \Sigma\wedge^{[\delta]}$ is no longer $\mathrm{poly}(s_0)$ because of the shift. Let $\mathbf{x}^{\mathbf{a}}$ be a monomial of degree δ, such that $\sum_i a_i \leq \delta$. Then the number of monomials produced by Φ can be upper bounded by the AM-GM inequality:

$$\prod_i (a_i + 1) \leq \left(\frac{\sum_i a_i + n}{n}\right)^n \leq (1 + \delta/n)^n$$

As $\delta/n \to 0$, we have $(1 + \delta/n)^n \to e^\delta \leq 3^\delta$. Hence, appropriately replacing the bound in the proof of Lemma 2.15 gives required size-blow up. □

Next we prove that the we can continue to reduce top fan-in with repeated application of Division and Derivation, while preserving the non-zeroness.

Lemma 7.11 (Inductive Reduction)**.** *For a positive integer* $j < k$*, let* $f_j \in \overline{\mathsf{Gen}(k-j, s_j)}$ *of syntactic degree* D_j*, over* $R_j := \mathbb{F}[z]\langle z^{d_j} \rangle$*, where* $d_j \leq d$*.*

- *(Valuation) Suppose* $g_j = \sum_i T_{i,j}$ *approximates* f_j*. For all* $i \in [k-j]$ *assume that* $\nu_{i,j} := \mathrm{val}_z(T_{i,j}) \geq 0$*.*

- *(Invertibility) Further, assume that* $U_{i,j}(\mathbf{x}, z = 0, \varepsilon)$ *and* $V_{i,j}(\mathbf{x}, z = 0, \varepsilon) \in \mathbb{F}(\varepsilon)\backslash\{0\}$.

Then, $f_{j+1} := \partial_z(f_j/t_{k-j,j}) \in \overline{\mathsf{Gen}(k-j-1, s_{j+1})}$ *over* $R_{j+1} := \mathbb{F}[z]/\langle z^{d_{j+1}}\rangle$, *where* $t_{k-j,j} \in R_j(\mathbf{x})$, $d_{j+1} \leq d_j \leq d$, *and* $s_{j+1} = s_j^7 \cdot d^{O(j)}$. *Moreover, the valuation and invertibility properties above continue to hold with respect to* g_{j+1} *approximating* $f_{j+1} = \sum_i T_{i,j+1}$. *Moreover,*

$$f_j \neq 0 \text{ over } R(\mathbf{x}) \iff f_{j+1} \neq 0 \text{ over } R_{j+1}(\mathbf{x}), \text{ or, } 0 \neq \left(\frac{f_j}{t_{k-j,j}}\right)\Big|_{z=0} \in \mathbb{F}(\mathbf{x}) .$$

Proof Sketch. In either case of $\Upsilon = \{\wedge, \Pi^{[\delta]}\}$, the proof of first part comes from Lemma 2.16 and the second part from Lemma 6.9. □

In *de-bordering* we need Induction to finally reconstruct the circuit exactly computing the polynomial. Moreover, for PIT, DiDI reduces the problem to testing the bloated model of top fan-in one. When we combine the two ideas to use solve PIT for border classes, we need a way fuse the hitting sets obtained in each step due to the $z = 0$ evaluation test. The following claim will help us achieve it. Borrowing the notations from the proof of Lemma 2.16, we state the following claim.

Claim 7.12. *For* $\mathbf{b} \in \mathbb{F}^n$, *if* $f_{j+1}(\mathbf{x} = \mathbf{b}, z) \neq 0$, *over* R_{j+1}, *and* $\mathrm{val}_z(\tilde{T}_{k-j,j})(\mathbf{x} = \mathbf{b}, z)) = \nu_{k-j,j}$. *Then* $f_j(\mathbf{x} = \mathbf{b}) \neq 0$, *over* R_j.

Proof. Suppose the hypothesis holds, but $f_j(\mathbf{x} = \mathbf{b}) = 0$, over R_j. Then,

$$\mathrm{val}_z\left(\left(\frac{f_j}{\tilde{T}_{k-j,j}}\right)_{\mathbf{x}=\mathbf{b}}\right) \geq d_j - \nu_{k-j,j} \implies \mathrm{val}_z\left(\partial_z\left(\left(\frac{f_j}{\tilde{T}_{k-j,j}}\right)_{\mathbf{x}=\mathbf{b}}\right)\right) \geq d_{j+1}..$$

The last condition implies that $\partial_z(f_j/\tilde{T}_{k-j,j})|_{\mathbf{x}=\mathbf{b}} = 0$, over $R_{j+1}(\mathbf{x})$. Fixing $\varepsilon = 0$ we deduce $f_{j+1}|_{\mathbf{x}=\mathbf{b}} = 0$. This is a contradiction! □

We are no ready to bring everything together to prove the main theorem of this chapter.

Theorem 7.13 (Hitting set for bounded border depth-4)**.** *There exists an explicit* $s^{O(k\cdot 7^k\cdot \log\log s)}$*-time hitting set for* $\overline{\Sigma^{[k]}\Pi\Sigma\wedge}(s)$. *Further, there exists an explicit* $s^{O(\delta^2 k 7^k \log s)}$*-time hitting set for* $\overline{\Sigma^{[k]}\Pi\Sigma\Pi^{[\delta]}}(s)$.

Proof Sketch. We follow on the lines of the proof of Theorem 6.1, while discussing the required tweaking for it to work for border PIT. Invoke Lemma 7.10 and Lemma 7.11 repeatedly to obtain that:

$$f_{k-1} \in \mathsf{Gen}(1, s_{k-1}) \subseteq \frac{\Pi\Sigma\Upsilon}{\Pi\Sigma\Upsilon} \cdot \frac{\overline{\Sigma \wedge \Sigma\Upsilon}}{\overline{\Sigma \wedge \Sigma\Upsilon}},$$

using Lemma 2.3. As seen before, we also need to understand the evaluation at $z = 0$. By a similar argument, it will follow that

$$\left(\frac{f_j}{t_{k-j,j}}\right)_{z=0} \in \overline{\Sigma \wedge \Sigma\Upsilon}\,.$$

When $\Upsilon = \wedge$. We know from Lemma 2.6 that $\overline{\Sigma\wedge\Sigma\wedge} \subseteq \mathsf{ARO}$ and thus $\overline{\Sigma\wedge\Sigma\wedge}$ has a hitting set of size $s^{O(k\cdot 7^k\cdot \log\log s)}$. We also have a hitting set for $\Pi\Sigma\wedge$ from Lemma 6.5. Combining them using Lemma 7.2, we have the final hitting set of size $s^{O(k7^k \log\log s)}$. In the jth step, the hitting set of ARO suffices to test the $z = 0$ evaluation part Theorem 7.1. For all $j \in [k-2]$ let $\mathcal{H}_j$ be that required hitting set of size $s^{O(k7^k \log\log s)}$. We know that H_{k-1} hits both f_{k-1} and $t_{2,k-2}$, because they are computable by same bloated model of same size. We lift these hitting sets using Claim 7.12. Define, the final hitting set in $\mathbf{x}$ variables: $\mathcal{H} := \bigcup_{j\in[k-1]} \mathcal{H}_j$. We remark that we do not need extra hitting set for each $t_{k-j,j}$, because it is already covered by $\mathcal{H}_{k-1}$. We have also kept track of $\deg(z)$ which is bounded by $s^{O(k)}$. We use a trivial hitting set for z which does not change the size. Thus, we have successfully constructed a $s^{O(k7^k \log\log s)}$-time hitting set for $\overline{\Sigma^{[k]}\Pi\Sigma\wedge}$.

When $\Upsilon = \Pi^{[\delta]}$. We do not know the size complexity upper bound of $\overline{\Sigma\wedge\Sigma\Pi^{[\delta]}}$. However, as we still have a hitting set for $\overline{\Sigma\wedge\Sigma\Pi[\delta]}$, from Theorem 7.8. Following the same line of argument as before we obtain our desired hitting set of size $s^{O(\delta^2 k7^k \log s)}$ for $\overline{\Sigma^{[k]}\Pi\Sigma^{[\delta]}\Pi}$. □

Chapter 8

Conclusion

In this thesis, we have attempted to advance our understanding of polynomials and algebraic circuits. The thesis is divided into three parts: Explicitness, Circuit Factoring, and Identity Testing. In each part, we address a different problem in Algebraic Complexity Theory. The order of presenting our results is intentionally chosen to highlight that our de-bordering results help in Circuit Factoring and Identity Testing. In the future, we are hopeful for more such interesting applications and perhaps direct and general connections between de-bordering and other naturally compelling problems in Algebraic Complexity Theory. We now present some open directions for improving our results presented in the thesis.

8.1 Explicitness

In Chapter 2, we showed that the polynomials in $\overline{\Sigma^{[k]}\Pi\Sigma\wedge}$ are explicit in a strong sense. To prove our result, we relied on our novel technique DiDIL, which in turn uses the known de-bordering results of $\overline{\Sigma\wedge\Sigma\wedge}$ and $\overline{\mathsf{ARO}}$. Naturally, the next step is to ask if $\overline{\Sigma^{[k]}\Pi\Sigma\Pi^{[\delta]}}$ is explicit. To answer this using DiDIL, we need to first de-border $\overline{\Sigma\wedge\Sigma\Pi^{\delta}}$, which to the best of our knowledge is currently unknown. The multiplicative blow-up in the size in each step of DiDIL gives $\exp(k)$ in the exponent, which makes the result relevant only until the top fan-in is at most logarithmic in the size of the circuit. Therefore, another direction to look into is devising new and more robust de-bordering techniques, which give better upper bound results.

The non-constructive nature of approximative circuits was discussed in Chapter 3, where we proposed a more realistic definition called *presentability*. We also proved that over finite fields, the presentable border of VNP is explicit. Such a general de-bordering result was not known until our work. We leave it as an open problem to prove similar de-bordering results for other fields. Another closely related problem is using presentability to obtain constructive de-bordering results, as most known techniques are non-constructive.

8.2 Circuit Factoring

In the second section of this thesis we proved that VNP is closed under factorisation over finite fields (Chapter 4) and the low-degree factors of small circuits are explicit (Chapter 5). Both of our main results in Chapters 4 and 5 can be viewed as applications of the techniques used to prove the results in the previous section. This still does not rule out the possibility: Could the Permanent polynomial be a factor of a small circuit of exponential degree?

Currently, we do not know if the class VP is closed under factorisation over fields of small characteristic [Bü00, Problem 2.1]. One reason for it is that we do not know a neat coefficient-based criterion for VP like we do for VNP (see Proposition 1.5). We can pose similar questions for all known factor closure results for restricted classes. Apart from resolving Conjecture 1.12, a direction to explore is if there are deeper and direct connections between de-bordering classes and its factors.

8.3 Identity Testing

Polynomial Identity Testing is one of the most widely studied problems in Algebraic Complexity Theory. In Chapter 6, we gave the first polynomial time whitebox PIT algorithm for $\Sigma^{[k]}\Pi\Sigma\wedge$, and in Chapter 7 we designed efficient robust hitting sets for $\overline{\Sigma^{[k]}\Pi\Sigma\wedge}$ and $\overline{\Sigma^{[k]}\Pi\Sigma\Pi^{[\delta]}}$. The tools used for the PIT results were heavily inspired from the de-bordering technique DiDIL. Although we were unable to de-border $\overline{\Sigma^{[k]}\Pi\Sigma\Pi^{[\delta]}}$, our technique reduces the problem to a model for which hitting sets are known. In particular, to completely de-randomise PIT for $\overline{\Sigma^{[k]}\Pi\Sigma\Pi^{[\delta]}}$ we first need to obtain an explicit hitting set for $\overline{\Sigma\wedge\Sigma\Pi^{\delta}}$.

The $\exp(k)$ in the exponent of the time complexity of our algorithm is inevitable because of the multiplicative blow-up in our iterative technique. Currently, the algorithm is efficient as long as the top fan-in is logarithmic in the size. An improvement in the exponent will derandomize PIT for a bigger class of polynomials. Another interesting

class for PIT problem is $\Sigma^{[k]}\Pi\Sigma\mathsf{M}_2$ circuits, where $\Sigma\mathsf{M}_2$ denotes bivariate polynomials. Our current techniques do not give a (robust) hitting set this model.

Bibliography

[AB03] Manindra Agrawal and Somenath Biswas. Primality and identity testing via Chinese remaindering. *J. ACM*, 50(4):429–443, 2003.

[AB09] Sanjeev Arora and Boaz Barak. *Computational complexity*. Cambridge University Press, Cambridge, 2009.

[AGKS15] Manindra Agrawal, Rohit Gurjar, Arpita Korwar, and Nitin Saxena. Hitting-Sets for ROABP and Sum of Set-Multilinear Circuits. *SIAM J. Comput.*, 44(3):669–697, 2015.

[AGL+18] Zeyuan Allen-Zhu, Ankit Garg, Yuanzhi Li, Rafael Mendes de Oliveira, and Avi Wigderson. Operator scaling via geodesically convex optimization, invariant theory and polynomial identity testing. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 172–181, 2018.

[Agr05] Manindra Agrawal. Proving Lower Bounds Via Pseudo-random Generators. In *FSTTCS 2005: Foundations of Software Technology and Theoretical Computer Science, 25th International Conference, Hyderabad, India, December 15-18, 2005, Proceedings*, volume 3821 of *Lecture Notes in Computer Science*, pages 92–105. Springer, 2005.

[AGS19] Manindra Agrawal, Sumanta Ghosh, and Nitin Saxena. Bootstrapping variables in algebraic circuits. *Proc. Natl. Acad. Sci. USA*, 116(17):8107–8118, 2019.

[AKS04] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. *Annals of mathematics*, pages 781–793, 2004.

[ALM+98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof Verification and the Hardness of Approximation Problems. *J. ACM*, 45(3):501–555, 1998.

[And20] Robert Andrews. Algebraic Hardness Versus Randomness in Low Characteristic. In *35th Computational Complexity Conference, CCC 2020, July 28-31, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 169 of *LIPIcs*, pages 37:1–37:32. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[And22] Robert Andrews. On Matrix Multiplication and Polynomial Identity Testing. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 356–365. IEEE, 2022.

[AS98] Sanjeev Arora and Shmuel Safra. Probabilistic Checking of Proofs: A New Characterization of NP. *J. ACM*, 45(1):70–122, 1998.

[AS09] Manindra Agrawal and Ramprasad SaptharishiI. Classifying polynomials and identity testing. Indian Academy of Sciences, 2009.

[ASS13] Manindra Agrawal, Chandan Saha, and Nitin Saxena. Quasi-polynomial hitting-set for set-depth-Δ formulas. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA,USA, June 1-4, 2013*, pages 321–330. ACM, 2013.

[ASSS16] Manindra Agrawal, Chandan Saha, Ramprasad Saptharishi, and Nitin Saxena. Jacobian Hits Circuits: Hitting Sets, Lower Bounds for Depth-D Occur-k Formulas and Depth-3 Transcendence Degree-k Circuits. *SIAM J. Comput.*, 45(4):1533–1562, 2016.

[AV08] Manindra Agrawal and V. Vinay. Arithmetic Circuits: A Chasm at Depth Four. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 67–75. IEEE Computer Society, 2008.

[AW16] Eric Allender and Fengming Wang. On the power of algebraic branching programs of width two. *Computational Complexity*, 25(1):217–253, 2016.

[BC92] Michael Ben-Or and Richard Cleve. Computing Algebraic Formulas Using a Constant Number of Registers. *SIAM J. Comput.*, 21(1):54–58, 1992.

[BCRL79] Dario Bini, Milvio Capovani, Francesco Romani, and Grazia Lotti. $O(n^{2.7799})$ complexity for $n \times n$ approximate matrix multiplication. *Information Processing Letters*, 8(5):234–235, 1979.

[BCS97] Peter Bürgisser, Michael Clausen, and Mohammad Amin Shokrollahi. *Algebraic complexity theory*, volume 315 of *Grundlehren der mathematischen Wissenschaften*. Springer, 1997.

[BFG+19] Peter Bürgisser, Cole Franks, Ankit Garg, Rafael Mendes de Oliveira, Michael Walter, and Avi Wigderson. Towards a Theory of Non-Commutative Optimization: Geodesic 1st and 2nd Order Methods for Moment Maps and Polytopes. In *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 845–861, 2019.

[BGdO+18] Peter Bürgisser, Ankit Garg, Rafael Mendes de Oliveira, Michael Walter, and Avi Wigderson. Alternating Minimization, Scaling Algorithms, and the Null-Cone Problem from Invariant Theory. In *9th Innovations in Theoretical Computer Science Conference, ITCS*, pages 24:1–24:20, 2018.

[BHS08] Markus Bläser, Moritz Hardt, and David Steurer. Asymptotically Optimal Hitting Sets Against Polynomials. In *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part I: Tack A: Algorithms, Automata, Complexity, and Games*, volume 5125 of *Lecture Notes in Computer Science*, pages 345–356. Springer, 2008.

[BI13] Peter Bürgisser and Christian Ikenmeyer. Explicit lower bounds via geometric complexity theory. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 141–150. ACM, 2013.

[BIL+21] Markus Bläser, Christian Ikenmeyer, Vladimir Lysikov, Anurag Pandey, and Frank-Olaf Schreyer. On the Orbit Closure Containment Problem and Slice Rank of Tensors. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 2565–2584, 2021.

[BIM+20] Markus Bläser, Christian Ikenmeyer, Meena Mahajan, Anurag Pandey, and Nitin Saurabh. Algebraic Branching Programs, Border Complexity, and Tangent Spaces. In *DROPS-IDN/v2/Document/10.4230/LIPIcs.CCC.2020.21*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020.

[Bin80] D. Bini. Relations between exact and approximate bilinear algorithms. Applications. *Calcolo. A Quarterly on Numerical Analysis and Theory of Computation*, 17(1):87–97, 1980.

[BIZ18] Karl Bringmann, Christian Ikenmeyer, and Jeroen Zuiddam. On Algebraic Branching Programs of Small Width. *Journal of the ACM*, 65(5):32:1–32:29, August 2018.

[BLMW11] Peter Bürgisser, J. M. Landsberg, Laurent Manivel, and Jerzy Weyman. An Overview of Mathematical Issues Arising in the Geometric Complexity Theory Approach to VP $\neq$ VNP. *SIAM J. Comput.*, 40(4):1179–1209, 2011.

[BMS13] Malte Beecken, Johannes Mittmann, and Nitin Saxena. Algebraic independence and blackbox identity testing. *Inf. Comput.*, 222:2–19, 2013.

[BP20] Markus Bläser and Anurag Pandey. Polynomial Identity Testing for Low Degree Polynomials with Optimal Randomness. In Jaroslaw Byrka and Raghu Meka, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2020, August 17-19, 2020, Virtual Conference*, volume 176 of *LIPIcs*, pages 8:1–8:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[BS21] Pranav Bisht and Nitin Saxena. Blackbox identity testing for sum of special ROABPs and its border class. *Comput. Complex.*, 30(1):8, 2021.

[BSV20a] Vishwas Bhargava, Shubhangi Saraf, and Ilya Volkovich. Deterministic Factorization of Sparse Polynomials with Bounded Individual Degree. *J. ACM*, 67(2):8:1–8:28, May 2020.

[BSV20b] Vishwas Bhargava, Shubhangi Saraf, and Ilya Volkovich. Reconstruction of Depth-4 Multilinear Circuits. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 2144–2160. SIAM, 2020.

[BT88] Michael Ben-Or and Prasoon Tiwari. A Deterministic Algorithm for Sparse Multivariate Polynominal Interpolation (Extended Abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, STOC*, pages 301–309. ACM, 1988.

[Bür04] Peter Bürgisser. The Complexity of Factors of Multivariate Polynomials. *Found. Comput. Math.*, 4(4):369–396, 2004.

[Bü00] Peter Bürgisser. *Completeness and reduction in algebraic complexity theory*, volume 7 of *Algorithms and computation in mathematics*. Springer-Verlag, 2000.

[Bü20] Peter Bürgisser. Correction to: The complexity of factors of multivariate polynomials. *Foundations of Computational Mathematics*, 20(6):1667–1668, 2020.

[CCG12] Enrico Carlini, Maria Virginia Catalisano, and Anthony V. Geramita. The solution to the Waring problem for monomials and the sum of coprime monomials. *Journal of Algebra*, 370:5–14, 2012.

[CK00] Zhi-Zhong Chen and Ming-Yang Kao. Reducing Randomness via Irrational Numbers. *SIAM J. Comput.*, 29(4):1247–1256, 2000.

[CKR+20] Prerona Chatterjee, Mrinal Kumar, C. Ramya, Ramprasad Saptharishi, and Anamay Tengse. On the Existence of Algebraically Natural Proofs. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 870–880, 2020.

[CKS18] Chi-Ning Chou, Mrinal Kumar, and Noam Solomon. Hardness vs Randomness for Bounded Depth Arithmetic Circuits. In *33rd Computational Complexity Conference, CCC 2018, June 22-24,2018, San Diego, CA, USA*, volume 102 of *LIPIcs*, pages 13:1–13:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[CKS19a] Chi-Ning Chou, Mrinal Kumar, and Noam Solomon. Closure of VP under taking factors: a short and simple proof, 2019. Pre-print available at `arXiv:1903.02366`.

[CKS19b] Chi-Ning Chou, Mrinal Kumar, and Noam Solomon. Closure results for polynomial factorization. *Theory of Computing. An Open Access Journal*, 15:Paper No. 13, 34, 2019. Preliminary version in the *33rd Annual Computational Complexity Conference (CCC 2018)*.

[CKW11] Xi Chen, Neeraj Kayal, and Avi Wigderson. Partial Derivatives in Arithmetic Complexity and Beyond. *Found. Trends Theor. Comput. Sci.*, 6(1-2):1–138, 2011.

[CL24] Prasad Chaugule and Nutan Limaye. On The Closures of Monotone Algebraic Classes and Variants of the Determinant. *Algorithmica*, 2024.

[CLO15] David A. Cox, John Little, and Donal O'Shea. *Ideals, varieties, and algorithms - an introduction to computational algebraic geometry and commutative algebra*. Undergraduate texts in mathematics. Springer, 2015.

[CR88] Benny Chor and Ronald L. Rivest. A knapsack-type public key cryptosystem based on arithmetic in finite fields. *IEEE Trans. Inform. Theory*, 34(5, part 1):901–909, 1988.

[CW90] Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251–280, 1990.

[DDS21] Pranjal Dutta, Prateek Dwivedi, and Nitin Saxena. Deterministic Identity Testing Paradigms for Bounded Top-Fanin Depth-4 Circuits. In *36th Computational Complexity Conference, CCC 2021, July 20-23, 2021, Toronto, Ontario, Canada (Virtual Conference)*, volume 200 of *LIPIcs*, pages 11:1–11:27. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[DDS22] Pranjal Dutta, Prateek Dwivedi, and Nitin Saxena. Demystifying the Border of Depth-3 Algebraic Circuits. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 92–103, February 2022.

[DG24] Pranjal Dutta and Sumanta Gosh. SIGACT News Complexity Theory Column 121. *SIGACT News*, 55(2), jun 2024.

[DL78] Richard A. Demillo and Richard J. Lipton. A probabilistic remark on algebraic program testing. *Information Processing Letters*, 1978.

[DOS14] Zeev Dvir, Rafael Oliveira, and Amir Shpilka. Testing Equivalence of Polynomials under Shifts. In *Automata, Languages, and Programming - 41st International Colloquium,ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, PartI*, volume 8572 of *Lecture Notes in Computer Science*, pages 417–428. Springer, 2014.

[DSS22] Pranjal Dutta, Nitin Saxena, and Amit Sinhababu. Discovering the Roots: Uniform Closure Results for Algebraic Classes Under Factoring. *J. ACM*, 69(3):18:1–18:39, 2022.

[DSY10] Zeev Dvir, Amir Shpilka, and Amir Yehudayoff. Hardness-randomness tradeoffs for bounded depth arithmetic circuits. *SIAM J. Comput.*, 39(4):1279–1293, 2009/10.

[Dut22] Pranjal Dutta. *A Tale of Hardness, De-randomization and De-bordering in Complexity Theory.* PhD thesis, Chennai Mathematical Institute, India, 2022.

[FGT21] Stephen A. Fenner, Rohit Gurjar, and Thomas Thierauf. Bipartite Perfect Matching is in Quasi-NC. *SIAM J. Comput.*, 50(3), 2021.

[For14] Michael A. Forbes. *Polynomial identity testing of read-once oblivious algebraic branching programs.* PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2014.

[For15] Michael A. Forbes. Deterministic Divisibility Testing via Shifted Partial Derivatives. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 451–465, October 2015.

[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.

[FS13a] Michael A. Forbes and Amir Shpilka. Explicit Noether Normalization for Simultaneous Conjugation via Polynomial Identity Testing. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 16th International Workshop, APPROX, and 17th International Workshop, RANDOM*, pages 527–542, 2013.

[FS13b] Michael A. Forbes and Amir Shpilka. Quasipolynomial-Time Identity Testing of Non-commutative and Read-Once Oblivious Algebraic Branching Programs. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 243–252, 2013.

[FS15] Michael A. Forbes and Amir Shpilka. Complexity Theory Column 88: Challenges in Polynomial Factorization1. *ACM SIGACT News*, 46(4):32–49, December 2015.

[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, STOC*, pages 1180–1192, 2018.

[FSTW21] Michael A. Forbes, Amir Shpilka, Iddo Tzameret, and Avi Wigderson. Proof complexity lower bounds from algebraic circuit complexity. *Theory Comput.*, 17:Paper No. 10, 88, 2021.

[GGdOW16] Ankit Garg, Leonid Gurvits, Rafael Mendes de Oliveira, and Avi Wigderson. A Deterministic Polynomial Time Algorithm for Non-commutative Rational Identity Testing. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS*, pages 109–117, 2016.

[GKKS14] Ankit Gupta, Pritish Kamath, Neeraj Kayal, and Ramprasad Saptharishi. Approaching the Chasm at Depth Four. *Journal of the ACM*, 61(6):33:1–33:16, 2014.

[GKS17] Rohit Gurjar, Arpita Korwar, and Nitin Saxena. Identity Testing for Constant-Width, and Any-Order, Read-Once Oblivious Arithmetic Branching Programs. *Theory Comput.*, 13(1):1–21, 2017.

[GKSS17] Joshua A. Grochow, Mrinal Kumar, Michael E. Saks, and Shubhangi Saraf. Towards an algebraic natural proofs barrier via polynomial identity testing. *CoRR*, abs/1701.01717, 2017. Pre-print available at `arXiv:1701.01717`.

[GKSS22] Zeyu Guo, Mrinal Kumar, Ramprasad Saptharishi, and Noam Solomon. Derandomization from algebraic hardness. *SIAM J. Comput.*, 51(2):315–335, 2022.

[Gre16] Bruno Grenet. Bounded-degree factors of lacunary multivariate polynomials. *J. Symb. Comput.*, 75:171–192, 2016.

[Gro15a] Joshua A Grochow. Unifying known lower bounds via geometric complexity theory. *Computational Complexity*, 24(2):393–475, 2015.

[Gro15b] Joshua A. Grochow. Unifying Known Lower Bounds via Geometric Complexity Theory. *computational complexity*, 24(2):393–475, June 2015.

[GS99] Venkatesan Guruswami and Madhu Sudan. Improved decoding of Reed-Solomon and algebraic-geometry codes. *IEEE Trans. Inform. Theory*, 45(6):1757–1767, 1999.

[GS11] S. B. Gashkov and I. S. Sergeev. Complexity of computations in finite fields. *Fundamentalnaya i Prikladnaya Matematika*, 17(4):95–131, December 2011.

[GSS19] Zeyu Guo, Nitin Saxena, and Amit Sinhababu. Algebraic Dependencies and PSPACE Algorithms in Approximative Complexity over Any Field. *Theory Comput.*, 15:1–30, 2019.

[GT20] Rohit Gurjar and Thomas Thierauf. Linear Matroid Intersection is in Quasi-NC. *Comput. Complex.*, 29(2):9, 2020.

[Gur15] Rohit Gurjar. *Derandomizing PIT for ROABP and Isolation Lemma for Special Graphs*. PhD thesis, Indian Institute of Technology Kanpur, 2015.

[HS80] Joos Heintz and Claus-Peter Schnorr. Testing Polynomials which Are Easy to Compute (Extended Abstract). In *Proceedings of the 12th Annual ACM Symposium on Theory of Computing, April 28-30, 1980, Los Angeles, California, USA*, pages 262–272. ACM, 1980.

[IQS18] Gábor Ivanyos, Youming Qiao, and K. V. Subrahmanyam. Non-commutative Edmonds' problem and matrix semi-invariants. *Comput. Complex.*, 26(3):717–763, 2018.

[Kal85] Erich Kaltofen. Polynomial-time reductions from multivariate to bi- and univariate integral polynomial factorization. *SIAM J. Comput.*, 14(2):469–489, 1985.

[Kal86] Erich L. Kaltofen. Uniform Closure Properties of P-Computable Functions. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing (STOC 1986)*, pages 330–337. ACM, 1986.

[Kal87] Erich Kaltofen. Single-Factor Hensel Lifting and Its Application to the Straight-Line Complexity of Certain Polynomials. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing (STOC 1987)*, page 443–452. Association for Computing Machinery, 1987.

[Kal89] Erich Kaltofen. Factorization of Polynomials Given by Straight-Line Programs. *Adv. Comput. Res.*, 5:375–412, 1989.

[Kay12] Neeraj Kayal. An exponential lower bound for the sum of powers of bounded degree polynomials. *Electronic Colloquium on Computational Complexity (ECCC)*, 19:81, 2012.

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

[Koi96] Pascal Koiran. Hilbert's Nullstellensatz Is in the Polynomial Hierarchy. *Journal of Complexity*, 12(4):273–286, December 1996.

[Koi12] Pascal Koiran. Arithmetic circuits: The chasm at depth four gets wider. *Theor. Comput. Sci.*, 448:56–65, 2012.

[KP11] Pascal Koiran and Sylvain Perifel. Interpolation in Valiant's theory. *Comput. Complexity*, 20(1):1–20, 2011.

[KRS24] Mrinal Kumar, Varun Ramanathan, and Ramprasad Saptharishi. Deterministic Algorithms for Low Degree Factors of Constant Depth Circuits. In David P. Woodruff, editor, *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024, Alexandria, VA, USA, January 7-10, 2024*, pages 3901–3918. SIAM, 2024.

[KRST22] Mrinal Kumar, C. Ramya, Ramprasad Saptharishi, and Anamay Tengse. If VNP Is Hard, Then so Are Equations for It. In *39th International Symposium on Theoretical Aspects of Computer Science, STACS*, pages 44:1–44:13, 2022.

[KS01] Adam R. Klivans and Daniel A. Spielman. Randomness efficient identity testing of multivariate polynomials. In *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*, pages 216–223. ACM, 2001.

[KS06] Adam R. Klivans and Amir Shpilka. Learning Restricted Models of Arithmetic Circuits. *Theory Comput.*, 2(10):185–206, 2006.

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

[KS09] Zohar Shay Karnin and Amir Shpilka. Reconstruction of Generalized Depth-3 Arithmetic Circuits with Bounded Top Fan-in. In *Proceedings of the 24th Annual IEEE Conference on Computational Complexity, CCC 2009, Paris, France, 15-18 July 2009*, pages 274–285. IEEE Computer Society, 2009.

[KS18] Mrinal Kumar and Shubhangi Saraf. Arithmetic Circuits with Locally Low Algebraic Rank. *CoRR*, abs/1806.06097, 2018. Pre-print available at `arXiv:1806.06097`.

[KS19] Mrinal Kumar and Ramprasad Saptharishi. Hardness-Randomness Tradeoffs for Algebraic Computation. *Bull. EATCS*, 129, 2019.

[KSS15] Swastik Kopparty, Shubhangi Saraf, and Amir Shpilka. Equivalence of polynomial identity testing and polynomial factorization. *Computational Complexity*, 24(2):295–331, 2015.

[KST19a] Mrinal Kumar, Ramprasad Saptharishi, 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. SIAM, 2019.

[KST19b] Mrinal Kumar, Ramprasad Saptharishi, and Anamay Tengse. Near-optimal Bootstrapping of Hitting Sets for Algebraic Circuits. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 639–646. SIAM, 2019.

[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. *J. Symbolic Comput.*, 9(3):301–320, 1990.

[Kum20] Mrinal Kumar. On the Power of Border of Depth-3 Arithmetic Circuits. *ACM Transactions on Computation Theory*, 12(1):5:1–5:8, February 2020.

[Len99] H. W. Lenstra, Jr. Finding small degree factors of lacunary polynomials. In *Number theory in progress, Vol. 1 (Zakopane-Kościelisko, 1997)*, pages 267–276. de Gruyter, 1999.

[LO15a] Joseph M. Landsberg and Giorgio Ottaviani. New lower bounds for the border rank of matrix multiplication. *Theory of Computing. An Open Access Journal*, 11:285–298, 2015.

[LO15b] Joseph M. Landsberg and Giorgio Ottaviani. New Lower Bounds for the Border Rank of Matrix Multiplication. *Theory Comput.*, 11:285–298, 2015.

[Lov79] László Lovász. On determinants, matchings, and random algorithms. In Lothar Budach, editor, *Fundamentals of Computation Theory, FCT 1979, Proceedings of the Conference on Algebraic, Arthmetic, and Categorial Methods in Computation Theory, Berlin/Wendisch-Rietz, Germany, September 17-21, 1979*, pages 565–574. Akademie-Verlag, Berlin, 1979.

[LST21] Nutan Limaye, Srikanth Srinivasan, and Sébastien Tavenas. Superpolynomial Lower Bounds Against Low-Depth Algebraic Circuits. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 804–814. IEEE, 2021.

[LV98] Daniel Lewin and Salil P. Vadhan. Checking Polynomial Identities over any Field: Towards a Derandomization? In Jeffrey Scott Vitter, editor, *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, pages 438–447. ACM, 1998.

[Mah14] Meena Mahajan. *Algebraic Complexity Classes*, pages 51–75. Springer International Publishing, 2014.

[Mal03] Guillaume Malod. *Polynômes et coefficients. (Polynomials and coefficients).* PhD thesis, Claude Bernard University Lyon , France, 2003.

[Mit13] Johannes Mittmann. *Independence in Algebraic Complexity Theory.* PhD thesis, University of Bonn, Germany, 2013.

[MP08] Guillaume Malod and Natacha Portier. Characterizing Valiant's algebraic complexity classes. *J. Complex.*, 24(1):16–38, 2008.

[MR08] Meena Mahajan and B. V. Raghavendra Rao. Arithmetic Circuits, Syntactic Multilinearity, and the Limitations of Skew Formulae. In *Mathematical Foundations of Computer Science 2008, 33rd International Symposium, MFCS 2008, Torun, Poland, August 25-29, 2008, Proceedings*, volume 5162 of *Lecture Notes in Computer Science*, pages 455–466. Springer, 2008.

[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.

[MS21] Dori Medini and Amir Shpilka. Hitting Sets and Reconstruction for Dense Orbits in VP_{e} and ΣΠΣ Circuits. In *36th Computational Complexity Conference, CCC 2021, July 20-23, 2021, Toronto, Ontario, Canada (Virtual Conference)*, volume 200 of *LIPIcs*, pages 19:1–19:27. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[Muk16] Partha Mukhopadhyay. Depth-4 Identity Testing and Noether's Normalization Lemma. In *Computer Science - Theory and Applications - 11th International Computer Science Symposium in Russia, CSR 2016, St. Petersburg, Russia, June 9-13, 2016, Proceedings*, volume 9691 of *Lecture Notes in Computer Science*, pages 309–323. Springer, 2016.

[Mul07] Ketan Mulmuley. Geometric Complexity Theory VI: the flip via saturated and positive integer programming in representation theory and algebraic

geometry. *CoRR*, abs/0704.0229, 2007. Pre-print available at `arXiv:0704.0229`.

[Mul12a] Ketan Mulmuley. The GCT program toward the P vs. NP problem. *Commun. ACM*, 55(6):98–107, 2012.

[Mul12b] Ketan D. Mulmuley. The GCT Program toward the P vs. NP Problem. *Commun. ACM*, 55(6):98–107, 2012.

[Mul17] Ketan Mulmuley. Geometric Complexity Theory V: Efficient Algorithms for Noether Normalization. *Journal of the American Mathematical Society*, 30(1):225–309, January 2017.

[Mum76] David Mumford. *Algebraic geometry. I.* Grundlehren der Mathematischen Wissenschaften. Springer-Verlag, 1976. Complex projective varieties.

[MV97] Meena Mahajan and V. Vinay. Determinant: Combinatorics, Algorithms, and Complexity. *Chic. J. Theor. Comput. Sci.*, 1997, 1997.

[MVV87] Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. *Comb.*, 7(1):105–113, 1987.

[Nis91] Noam Nisan. Lower Bounds for Non-Commutative Computation. In *Proceedings of the Twenty-Third Annual ACM Symposium on Theory of Computing*, STOC '91, pages 410–418, New York, NY, USA, January 1991. Association for Computing Machinery.

[NW94] Noam Nisan and Avi Wigderson. Hardness vs Randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994.

[Oli16] Rafael Oliveira. Factors of low individual degree polynomials. *Comput. Complexity*, 25(2):507–561, 2016.

[Oli20] Rafael Oliveira. Conditional lower bounds on the spectrahedral representation of explicit hyperbolicity cones. In *Proceedings of the 2020 International Symposium on Symbolic and Algebraic Computation (ISSAC 2020)*, pages 396–401. ACM, 2020.

[Ore22] Øystein Ore. Über höhere kongruenzen. *Norsk Mat. Forenings Skrifter*, 1922.

[Pap94] Christos H. Papadimitriou. *Computational complexity.* Addison-Wesley Publishing Company, Reading, MA, 1994.

[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 *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 259–271. ACM, 2021.

[Pé04] Sylvain Périfel. *Polynômes donnés par des circuits algébriques et généralisation du modèle de Valiant*. École Normal Supérieure de Lyon, France, 2004. Master's Thesis.

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

[Sap13] Ramprasad Saptharishi. *Unified Approaches to Polynomial Identity Testing and Lower Bounds*. PhD thesis, Ph. D. thesis, Chennai Mathematical Institute, India, 2013.

[Sax08] Nitin Saxena. Diagonal Circuit Identity Testing and Lower Bounds. In *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, PartI: Tack A: Algorithms, Automata, Complexity, and Games*, volume 5125 of *Lecture Notes in Computer Science*, pages 60–71. Springer, 2008.

[Sax09a] Nitin Saxena. Progress on Polynomial Identity Testing. *Bull. EATCS*, 99:49–79, 2009.

[Sax09b] Nitin Saxena. Progress on Polynomial Identity Testing. *Bull. EATCS*, 99:49–79, 2009.

[Sax14a] Nitin Saxena. Progress on Polynomial Identity Testing - II. *Perspectives in Computational Complexity: The Somenath Biswas Anniversary*, 26:131–146, 2014.

[Sax14b] Nitin Saxena. Progress on polynomial identity testing-II. In *Perspectives in Computational Complexity*. Springer, 2014.

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

[Sha92] Adi Shamir. IP = PSPACE. *J. ACM*, 39(4):869–877, 1992.

[Sho09] Victor Shoup. *A computational introduction to number theory and algebra*. Cambridge University Press, second edition, 2009.

[Shp09] Amir Shpilka. Interpolation of Depth-3 Arithmetic Circuits with Two Multiplication Gates. *SIAM J. Comput.*, 38(6):2130–2161, 2009.

[Sin16] Gaurav Sinha. Reconstruction of Real Depth-3 Circuits with Top Fan-In 2. In *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, volume 50 of *LIPIcs*, pages 31:1–31:53. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

[Sin19] Amit Kumar Sinhababu. *Power series in complexity: Algebraic Dependence, Factor Conjecture and Hitting Set for Closure of VP*. PhD thesis, Indian Institute of Technology, Kanpur, India, 2019.

[SK12] Jayalal Sharma and Dinesh K. Advanced Complexity Theory. Lecture Notes, 2012.

[SS12] Nitin Saxena and C. Seshadhri. Blackbox Identity Testing for Bounded Top-Fanin Depth-3 Circuits:The Field Doesn't Matter. *SIAM J. Comput.*, 41(5):1285–1298, 2012.

[SSS13] Chandan Saha, Ramprasad Saptharishi, and Nitin Saxena. A Case of Depth-3 Identity Testing, Sparse Factorization and Duality. *Comput. Complex.*, 22(1):39–69, 2013.

[ST17] Ola Svensson and Jakub Tarnawski. The Matching Problem in General Graphs Is in Quasi-NC. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 696–707. IEEE Computer Society, 2017.

[ST21] Amit Sinhababu and Thomas Thierauf. Factorization of Polynomials Given by Arithmetic Branching Programs. *Comput. Complex.*, 30(2):15, 2021.

[Str74] Volker Strassen. Polynomials with rational coefficients which are hard to compute. *Siam Journal On Computing*, 3:128–149, 1974.

[Sud97] Madhu Sudan. Decoding of Reed Solomon codes beyond the error-correction bound. *J. Complexity*, 13(1):180–193, 1997.

[Sud98] Madhu Sudan. Algebra and Computation. Lecture Notes, 1998.

[SV10] Amir Shpilka and Ilya Volkovich. On the relation between polynomial identity testing and finding variable disjoint factors. In *Automata, languages and programming. Part I*, volume 6198 of *Lecture Notes in Comput. Sci.*, pages 408–419. Springer, 2010.

[SV18] Shubhangi Saraf and Ilya Volkovich. Black-Box Identity Testing of Depth-4 Multilinear Circuits. *Comb.*, 38(5):1205–1238, 2018.

[SY10] Amir Shpilka and Amir Yehudayoff. Arithmetic Circuits: A Survey of Recent Results and Open Questions. *Found. Trends Theor. Comput. Sci.*, 5(3–4):207–388, 2010.

[Tav15] Sébastien Tavenas. Improved bounds for reduction to depth 4 and depth 3. *Inf. Comput.*, 240:2–11, 2015.

[Val79] Leslie G. Valiant. Completeness Classes in Algebra. In *Proceedings of the 11th Annual ACM Symposium on Theory of Computing (STOC 1979)*, pages 249–261. ACM, 1979.

[Val82] L. G. Valiant. Reducibility by algebraic projections. In *Logic and algorithmic*, volume 30 of *Monogr. Enseign. Math.*, pages 365–380. Univ. Genève, Geneva, 1982.

[vzG84] Joachim von zur Gathen. Hensel and Newton methods in valuation rings. *Math. Comp.*, 42(166):637–661, 1984.

[vzGG13] Joachim von zur Gathen and Jürgen Gerhard. *Modern computer algebra*. Cambridge University Press, Cambridge, third edition, 2013.

[vzGK85] Joachim von zur Gathen and Erich L. Kaltofen. Factoring Sparse Multivariate Polynomials. *J. Comput. Syst. Sci.*, 31(2):265–287, 1985.

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