Algebraic Circuit Complexity: New Lower Bounds, Algorithms, and Applications

By

Rajit Datta

A thesis submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

to

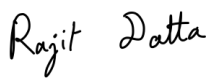
Chennai Mathematical Institute
July 2021

Plot-H1, SIPCOT IT Park,
Siruseri, Kelambakkam,
Tamilnadu - 603103
India

Declaration

I declare that this thesis titled, "Arithmetic Circuit Complexity: New Lower Bounds, Algorithms, and Applications" submitted by me for the degree of Doctor of Philosophy is the record of work carried out by me during the period from August 2016 to April 2021 under the guidance of Prof. Partha Mukhopadhyay. This work has not formed the basis for the award of any degree, diploma, associateship, fellowship, titles in this or any other university or other similar institution of higher learning.

Rajit Datta



July 2021
Chennai Mathematical Institute
H1, SIPCOT IT Park, Siruseri,
Chennai 603103, TN, India

Certificate

This is to certify that the Ph.D. thesis submitted by Rajit Datta to Chennai Mathematical Institute, titled "Arithmetic Circuit Complexity: New Lower Bounds, Algorithms, and Applications" is a record of bona fide research work done during the period from August 2016 till April 2021 under my guidance and supervision. The research work presented in this thesis has not formed the basis for the award of any degree, diploma, associateship, fellowship, titles in this institute or any other university or institution of higher learning. It is further certified that the thesis represents independent work by the candidate and collaboration when existed was necessitated by the nature and scope of problems dealt with.

Partha Mukhopadhyay
(Thesis Supervisor)

16/07/2021
July 2021
Chennai Mathematical Institute
H1, SIPCOT IT Park, Siruseri,
Chennai 603103, TN, India

Synopsis

Arithmetic Complexity Theory

Computational Complexity Theory aims to classify problems according to the resources needed to solve them. When computing boolean functions $f : \{0,1\}^n \mapsto \{0,1\}$ the notion of resources is captured by formal models for computing boolean functions, such as Turing machines [104] or boolean circuits [96]. A more specialized study is Algebraic Complexity Theory which aims to perform a similar classification for polynomials. An arithmetic circuit is similar to a boolean circuit in structure but the logical gates OR, AND are replaced by the arithmetic operations $+, \times$. The leaves of a boolean circuit are labelled by variables $x_1, x_2, \ldots, x_n$, which can take values in $\{0,1\}$. Whereas in the arithmetic world, these variables can take values from a field $\mathbb{F}$. The notion of efficient computation is captured by the size of the arithmetic circuit.

Analogous to the boolean class P, the set of efficiently computable polynomials form the set VP. An important and useful polynomial in VP is the determinant polynomial [65]. The class VNP is the arithmetic analog of NP. Informally speaking, polynomials in VNP are expressed as exponential sum over polynomials in VP. Valiant [107] showed that the permanent polynomial of a $n \times n$ matrix is complete for VNP in a precise sense and conjectured that permanent does not have polynomial size circuits. The conjecture $\text{VP} \neq \text{VNP}$ is the arithmetic analog of the $\text{P} \neq \text{NP}$ conjecture. In contrast the determinant polynomial of a $n \times n$ matrix has small arithmetic circuit [65].

Progress on this conjecture has been quite slim. The best known result is that of Mignon and Ressyare [69] who show a $\Omega(n^2)$ lower bound on the *determinantal complexity* of permanent. A series of influential results on depth reduction [4, 59, 102] show that the general arithmetic circuits of size s computing a polynomial of degree d can be compressed to a depth four circuit of size at most $s^{O(\sqrt{d})}$ computing the same polynomial. This has initiated effort towards proving strong lower bounds against depth four circuits, in the hope that the special structure can be exploited. Besides restriction on the depth, several other structural restrictions such as *multilinearity*, *monotonicity*, *formulas*, *arithmetic branching programs* have been studied and consequently lower bounds have been proved. Further details can be found in the excellent surveys [28, 97].

Another fundamental problem in arithmetic circuit complexity is to check whether a given input circuit computes the identically zero polynomial or not. This problem is known as polynomial identity Testing (PIT). The PIT problem has a randomized polynomial-time algorithm due to Schwarz-Zippel-DeMillo-Lipton lemma [34, 92, 116]. A major open problem is to find a deterministic subexponential-time algorithm for PIT. As shown in [53], derandomization of PIT is connected to proving circuit lower bounds. Over the years, PIT

has found several important applications in complexity theory and algorithm design. For example, the parallel randomized algorithm for detecting bipartite perfect matching [70] uses an instance of PIT. More recently, applications of PIT have given important results in the field of parameterized algorithm design. For example, important graph theoretic problems such as the detection of simple paths of length k and approximate counting of subgraphs use PIT in interesting ways. The reader is referred to the surveys [42, 111] for more details.

Contributions of the thesis

This thesis provides contributions on both the lower bounds as well as the algorithmic aspects of Arithmetic Complexity. On the lower bound side, we study *monotone* arithmetic circuits and prove new lower bound results. Most notably, using techniques from communication complexity, we exhibit a polynomial computable by (non-monotone) depth-3 circuit which requires monotone arithmetic circuits of exponential size.
The contribution on the algorithmic side is three fold. Our first algorithmic result makes progress in an important problem of algebraic automata theory which asks to design efficient deterministic algorithm for testing equivalence of multi-tape weighted automata [40]. We give the first deterministic quasi-polynomial time algorithm for the equivalence testing of k-tape weighted automata for constant k. Our algorithm is obtained by designing a PIT algorithm for arithmetic branching programs over certain partially commutative domains.
The next chapter of the thesis deals with PIT and irreducibility testing over the non-commutative and non-associative model. Hrubeš, Wigderson, and Yehudayoff [49] have studied the commutative non-associative model and proved completeness and lower bound results. Over non-commutative and non-associative domain, we complement their result by designing efficient deterministic identity testing and irreducibility testing algorithms.
In the final chapter of the thesis, we study the ideal membership problem from the angle of parameterized complexity and design new algorithms. One of the main results is a new algorithm for computing the permanent of low rank matrices.

Lower Bounds for Monotone Arithmetic Circuits

A polynomial over reals is said to be monotone if all its coefficients are positive. An arithmetic circuit is said to be monotone if every gate of the circuit computes a monotone polynomial. The study of monotone arithmetic circuits was initiated by Schnorr [91]. Then, Valiant [107] demonstrated that negations can exponentially reduce the resources needed to compute certain polynomials. More precisely, Valiant showed that the perfect matching polynomial of a triangular grid graph (which is *planar*) on n vertices requires monotone circuits of size at least $2^{\Omega(\sqrt{n})}$. In contrast, it has a non-monotone arithmetic branching program of polynomial size. Such a branching program can be obtained from the celebrated FKT algorithm [56, 103]. Thereafter, Jerrum and Snir [52] also exhibited many polynomial families which require monotone circuits of exponential size. Notable amongst them is the spanning tree polynomial which also has polynomial size non-monotone arithmetic branching program due to the classical matrix tree theorem [69].

We construct a family of polynomials P_n in n variables, each of its monomials has positive coefficient, such that P_n can be computed by a polynomial-size *depth-three formula* but every monotone circuit computing it has size $2^{\Omega(n^{1/4}/\log(n))}$. The polynomial P_n embeds the SINK $\circ$ XOR function devised recently by Chattopadhyay, Mande and Sherif [27] to refute the Log-Approximate-Rank Conjecture in communication complexity. To prove our lower bound for P_n, we develop a general connection between corruption of combinatorial rectangles by any function $f \circ \text{XOR}$ and corruption of product polynomials by a certain polynomial P^f that is an arithmetic embedding of f.

Recently Hrubeš [48] advocated a new approach for proving lower bounds against general arithmetic circuits. He showed that if $f \in \mathbb{F}[x_1, \ldots, x_n]$ is of degree d and has arithmetic circuits of size s then for some $\epsilon > 0$, the polynomial $(\sum_{i=1}^{n} x_i + 1)^d + \epsilon \cdot f$ has monotone arithmetic circuits of size at most $O(sd^2 + n \log n)$. Thus to show VP $\neq$ VNP, one approach is to exhibit a p-family $f \in$ VNP such that for every $\epsilon > 0$ the polynomial $(\sum_{i=1}^{n} x_i + 1)^d + \epsilon \cdot f$ requires monotone arithmetic circuits of super-polynomial size. We show progress in this approach by exhibiting a p-family $f \in$ VNP for which requires super-polynomial monotone circuits as long as $\epsilon > 2^{-\Omega(n)}$. Our proof relies on ideas from communication complexity and utilizes the discrepancy bound for the boolean function MOD3 $\circ$ MOD2.

Equivalence of k-tape Weighted Automata

Testing equivalence of multi-tape finite automata is a fundamental algorithmic problem. For a k-tape automaton, we denote by $\Sigma_1, \ldots, \Sigma_k$ the mutually disjoint alphabets for the k tapes. The automaton accepts a subset of the product monoid $\Sigma_1^* \times \cdots \times \Sigma_k^*$. Two multi-tape automata are equivalent if they both accept the same subset.

Equivalence testing of multi-tape *non-deterministic* automata is undecidable [44]. The problem was shown to be decidable for 2-tape *deterministic* automata [21,106], and then an exponential upper bound was shown for it [18]. Eventually, a polynomial-time algorithm was obtained [40] and the authors conjectured that equivalence testing of deterministic k-tape automata for any constant k is in polynomial time.

Motivated by equivalence testing of k-tape weighted automata, we study the *equivalence* testing of automata in the more general setting of partially commutative monoids (in short, pc monoids) and show efficient algorithms in special cases, exploiting the structure of the underlying non-commutation graph of the monoid.

Specifically, if the edge clique cover number of the non-commutation graph of the pc monoid is a constant, we obtain a deterministic quasi-polynomial time algorithm for equivalence testing. As corollary, we obtain the first deterministic quasi-polynomial time algorithms for equivalence testing of k-tape weighted automata and for equivalence testing of deterministic k-tape automata for constant k. Prior to this, the best complexity upper bound for these k-tape automata problems was randomized polynomial-time, shown by Worrell [114]. Finding polynomial-time deterministic algorithms for these problems has been open for several years [40], and our results make significant progress.

We also consider pc monoids for which the non-commutation graphs have an edge cover consisting of at most k cliques and star graphs for any constant k. We obtain randomized polynomial-time algorithm for equivalence testing of weighted automata over such monoids. Our results are obtained by designing efficient zero testing algorithms for weighted

automata over such pc monoids.

PIT and Irreducibility Testing in the Relationless Model

Non-commutative computation, introduced in complexity theory by Hyafil [51] and Nisan [71], is an important subfield of algebraic complexity theory. The main algebraic structure of interest is the free non-commutative ring $\mathbb{F}\langle X\rangle$ over a field $\mathbb{F}$, where $X = \{x_1, x_2, \cdots, x_n\}$ is a set of free noncommuting variables. When the multiplication operation of an arithmetic circuit is both non-commutative and non-associative, it is called a *relationless* circuit and it computes a polynomial in the free non-associative non-commutative ring $\mathbb{F}\{X\}$. Previously, the *relationless* arithmetic model of computation was considered by Hrubeš, Wigderson, and Yehudayoff [49]. They showed completeness and explicit lower bound results for this model. We study PIT in this model and show that given an arithmetic circuit of size s computing a polynomial $f(x_1, x_2, \ldots, x_n) \in \mathbb{F}\{X\}$ of degree d, we can test $f \equiv 0$ in deterministic $\text{poly}(s, n, d)$.
Next, we consider the problem of irreducibility testing in the ring $\mathbb{F}\{X\}$. Irreducibility testing is an important problem which has been studied over various domains. For example, over integers, irreducibility testing corresponds to the classical primality testing problem, which has a deterministic polynomial time algorithm [3]. Over the polynomial ring $\mathbb{F}[X]$ Kaltofen [54] gave a randomized polynomial time algorithm which can even factorize an input polynomial f given by an arithmetic circuit. The noncommutative polynomial ring $\mathbb{F}\langle X\rangle$ is not a *unique factorization domain* [73] and this poses difficulties towards algorithm design. However, unique factorization holds for homogeneous polynomials in $\mathbb{F}\langle X\rangle$, and it is shown in [12] that for homogeneous polynomials given by noncommutative circuits, the unique factorization into irreducible factors can be computed in randomized polynomial time (essentially, by reduction to the noncommutative PIT problem). In this chapter, we give a deterministic polynomial time algorithm for irreducibility testing for polynomials in $\mathbb{F}\{X\}$. Further, our algorithm finds a non-trivial factorization of the input polynomial in case it is reducible.

Univariate Ideal Membership problem

Let $X = \{x_1, x_2, \ldots, x_n\}$ be a set of n commuting variables and $\mathbb{F}$ be a field which is either the field $\mathbb{Q}$ of rationals or a finite field. Let $R = \mathbb{F}[X]$ be the ring of multivariate polynomials over the variables in X with coefficients from the field $\mathbb{F}$. A subring $I \subseteq R$ is an *ideal* if I absorbs multiplications by elements of R. That is, $I \cdot R \subseteq I$.
Computationally, an ideal $I \subset R$ is given by a set of generator polynomials : $I = \langle f_1, f_2, \ldots, f_\ell\rangle$. In other words, I is the smallest ideal containing the polynomials $f_i, 1 \leq i \leq \ell$. Given $f \in R$ and $I = \langle f_1, \ldots, f_\ell\rangle$, the *Ideal Membership problem* is to decide whether $f \in I$ or not. In general, the problem is notoriously intractable. It is EXPSPACE-complete even if f and the generators $f_i, i \in [\ell]$ are given explicitly as sums of monomials [67]. Nevertheless, special cases of ideal membership problem have played important roles in several results in arithmetic complexity. Motivated by Alon's Combinatorial Nullstellensatz we study the special case when the ideal is of the form $I = \langle p_1(x_1), \ldots, p_n(x_n)\rangle$, where p_i are univariates. We give algorithms for various cases of univariate ideal membership and highlight combinatorial applications. First we obtain a deterministic $d^{O(r)} \cdot \text{poly}(n)$

time division algorithm to evaluate the (unique) remainder polynomial $f(\ell_1, \ell_2, \ldots, \ell_r) \pmod I$ at any point $\vec{a} \in \mathbb{F}^n$, where $f \in \mathbb{F}[z_1, \ldots, z_r]$ is a polynomial of degree d given by a $n^{O(1)}$ size arithmetic circuit. Here $\ell_1, \ldots, \ell_r$ are linear forms over $x_1, \ldots, x_n$ and $I = \langle p_1(x_1), p_2(x_2), \ldots, p_n(x_n) \rangle$ is an univariate ideal. As an application we give an alternate algorithm to evaluate the permanent of $n \times n$ matrix of rank r. An algorithm of similar run time for low rank permanent is due to Barvinok [17] via a different technique. Our ideal membership algorithm also yields a randomized $n^{O(r)}$ algorithm for minimum vertex cover on graphs with rank-r adjacency matrices.
When the ideal is of the form $I = \langle x_1^{e_1}, \ldots, x_n^{e_n} \rangle$, we can test membership $f \in I$ in randomized $O((2e)^d n^{O(1)})$ time, where f is a degree d polynomial given by an arithmetic circuit of polynomial size. On the other hand, if each p_i has all distinct rational roots we can check if $f \in I$ in randomized $O(n^{d/2}(nd)^{O(1)})$ time, improving on the brute-force $\binom{n+d}{d}$-time search. Finally we show that when the roots of p_i are distinct, univariate ideal membership is in coNP. This complements a result of Alon and Tarsi [6] who proved coNP-hardness for this special case.

Publications in this Thesis:

1. Arkadev Chattopadhyay, Rajit Datta and Partha Mukhopadhyay,
   Lower bounds for monotone arithmetic circuits via communication complexity,
   STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing.

2. Vikraman Arvind, Abhranil Chatterjee, Rajit Datta and Partha Mukhopadhyay,
   Multiplicity Equivalence Testing of Automata over Partially Commutative Monoids,
   to appear MFCS '21: 46th International Symposium on Mathematical Foundations of Computer Science.

3. Vikraman Arvind, Abhranil Chatterjee, Rajit Datta and Partha Mukhopadhyay,
   Univariate Ideal Membership Parameterized by Rank, Degree, and Number of Generators,
   FSTTCS '18: 38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science.

4. Vikraman Arvind, Rajit Datta, Partha Mukhopadhyay and S. Raja,
   Lower bounds for monotone arithmetic circuits via communication complexity,
   MFCS '17: 42nd International Symposium on Mathematical Foundations of Computer Science.

Acknowledgements

I consider myself lucky to have been under the guidance of Prof. Partha Mukhopadhyay. He has a very keen understanding of people's abilities and always directed me to topics that were within my reach. His constant guidance saved me from being overwhelmed and made this long process much easier. His friendly and down to earth attitude made discussions fluid and enjoyable.
I am immensely grateful to Prof. V. Arvind for numerous discussions and research ideas. His vast knowledge and experience led discussions to fruitful ends. Along side being a good researcher he was a good teacher as well. Starting from my undergraduate days I have had the opportunity to attend his classes and was inspired to study complexity theory. I have learnt a great deal from him over this long period of association and I am thankful to have met him.
Ph.D. could have been dull and lonely had it not been for Abhranil Chatterjee - my "Guru Bhai". Having a friend to discuss at all times gave the courage to take on problems. Our discussions were interleaved with "poor jokes" which alleviated the mood and made research an enjoyable process. I am glad to have been a part of this small family.
I would like to express my gratitude towards Prof. Arkadev Chattopadhyay for exciting discussions and for teaching me communication complexity. His enthusiasm drove discussions even when we were unable to physically meet.
I would like to thank Prof. Piyush Srivastava for many insightful discussions and for introducing me to Markov chains. I am grateful to Sayantan Chakraborty and Rishab Batra for many discussions which helped me gradually learn the subject. I had the opportunity to collaborate with S. Raja during the first year of my Ph.D. I thank him for his friendly discussions which helped me to transition from a masters degree to research. I thank Prof. K. V. Subrahmanyam and Prof. Samir Datta for being on my doctoral committee. I would like to thank Tata Consultancy Services for funding my research via their research scholarship program.
At the brink of completion of this long process I realise that many aspects made this possible. I would like to thank my mother Mausumi Das, for her support and for always having faith in my choices. I am grateful to Moti Lal Sen and my grandmother Nilima Das for their support and blessings. In my childhood my mother introduced me two good teachers, Mr. Goutam Bhattacharyya and Mr. Gobindo Bhattacharyya. These people sowed the seed of scientific thinking and encouraged me to pursue topics outside the school syllabus. I am grateful to the teachers at the Ramanujan School of Mathematics (Kolkata) for developing my problem solving skills.
I would like to thank CMI providing an excellent learning atmosphere. I have come in contact with many wonderful personalities who have greatly influenced my life in both personal and academic aspects. I would like to thank Abhishek De, Soumyadip Sahu,

Anish Mukherjee, Niranka Banerjee, Debangshu Mukherjee, Pranav Ashok, Prantar Ghosh, Utsab Ghoshal, Abhishek Oswal, Richick Sinha, Prateek Roy, Adwitee Roy, Himalaya Senapati for making CMI an exciting place. Special thanks to Soumyadip Sahu and Himalaya Senapati for several discussions which enhanced my mathematical and problem solving abilities. I would like to thank Sreerupa Bhattacharjee for being with me. The time I spent with you was very peaceful. I am glad I met you.
The lockdown could have been a very difficult time for me. I am greatly indebted to CMI for the hospitality extend to me during that period which enabled me to continue research seamlessly. I thank the security guards and mess staff for their hospitality and friendly behaviour. Thanks to Alka Yadav for all the motivation which made the lockdown less dull. Many thanks to Anubhab Chatterjee for helping me out in times of need. Life would have been bland without the daily cooking experiences shared with Meghasan Senapati. I thank him for being a well wisher and for helping me during times of need. I would like to thank CMI administrative staff S. Sripathy, Rajeshwari Nair, V. Vijayalaksmi, Ranjini Girish and Nisha John for their help. I greatly appreciate how efficiently they do their work which saves us a lot of time and effort. They seamlessly managed my travel funds and other official documents and I could peacefully carry out my research.

Contents

Chapter 1

Introduction

Computational Complexity Theory aims to classify problems according to the resources needed to solve them. When computing Boolean functions $f : \{0,1\}^n \mapsto \{0,1\}$ the notion of resources is captured by formal models for computing Boolean functions, such as Turing machines [104] or Boolean circuits [96]. A more specialized study is Algebraic Complexity Theory which aims to perform a similar classification for polynomials. An arithmetic circuit is similar to a Boolean circuit in structure but the logical gates OR, AND are replaced by the arithmetic operations $+, \times$. The leaves of a Boolean circuit are labelled by variables $x_1, x_2, \ldots, x_n$, which could take values in $\{0,1\}$, whereas in the arithmetic world these variables can take values from a field $\mathbb{F}$.

Definition 1.0.1. *An* arithmetic circuit *C over a field $\mathbb{F}$ and indeterminates $X = \{x_1, x_2, \cdots, x_n\}$ is a directed acyclic graph (DAG) with each node of indegree zero labelled by a variable or a scalar constant from $\mathbb{F}$: the indegree 0 nodes are the input nodes of the circuit. Each internal node of the DAG is labeled by either a $+$ or a $\times$ (indicating that it is a plus gate or multiply gate, respectively). A gate of C is designated as* output*. Each internal gate computes a polynomial (by adding or multiplying its input polynomials), where the polynomial computed at an input node is just its label (variable or scalar). The* polynomial computed *by the circuit is the polynomial computed at its output gate.*

A polynomial of degree at most d over n variables has at most $\binom{n+d}{d}$ monomials and one can represent it by writing down all its coefficients as a long vector of length $\binom{n+d}{d}$. This representation is 'expensive'. On the other hand Arithmetic Circuits provide a small representation for certain polynomials. For example the circuit in Figure 1.1 is of size $2k+5$ but computes a polynomial with 2^{k-1} monomials. At this point we note that the study is not about individual polynomials but about polynomial families. A p-family is a family of polynomials $\{f_n\}_{n\in\mathbb{N}}$ where $f_n \in \mathbb{F}[x_1, \ldots, x_n]$. Some interesting p-families are

Example 1.0.1 (Elementary Symmetric Polynomial)**.**

$$S_{n,d}(x_1, \ldots, x_n) = \sum_{\substack{S\subseteq[n] \\ |S|=d}} \prod_{i\in S} x_i$$

Keeping d fixed we get the p-family $\{S_{n,d}\}_{n\geq d}$. A classical result of Ben-Or shows that $S_{n,d}$ has a 'small' circuit of size $O(n^2)$ whereas the naive representation would require

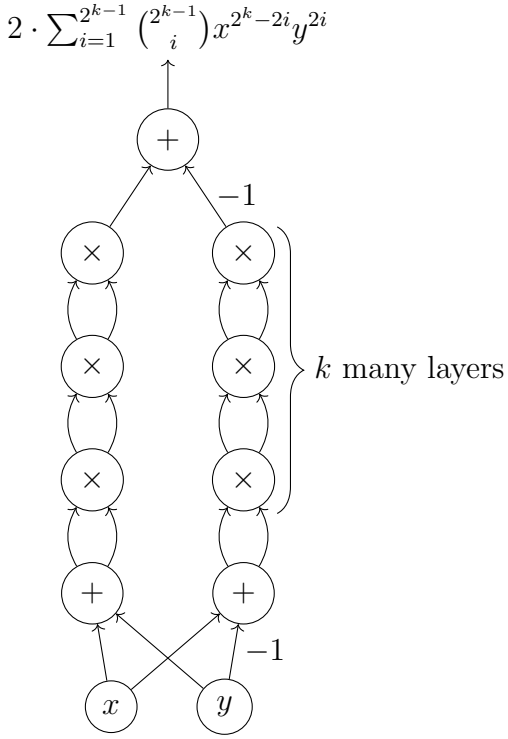


Figure 1.1: An Arithmetic Circuit

$\binom{n+d}{d}$ space to write. Another extremely important polynomial is the Determinant of a matrix which is defined over n^2 variables:

Example 1.0.2 (Determinant)**.** *For a matrix of variables $X = \{x_{i,j}\}$ we have*

$$\det_n(X) = \sum_{\sigma \in S_n} (-1)^{\mathrm{sgn}(\sigma)} \prod_{i=1}^{n} x_{i,\sigma(i)}$$

where S_n is the group of permutations on n letters and $\mathrm{sgn}(\sigma)$ is the sign of the permutation σ.

The determinant polynomial is ubiquitous in mathematics and has been an object of study since 1800s. Interestingly, though the polynomial is defined as an exponential sum, this too has polynomial size arithmetic circuits [65]. In the arithmetic setting, the class of p-families that have polynomial size arithmetic circuits represent efficient computation.

Definition 1.0.2. *For a computable function $s(n)$ we have*

$$\mathrm{ACSize}(s(n)) = \{\{p_n\}_{n\in\mathbb{N}} \mid \exists \text{ arithmetic circuit } C_n \text{ of size at most } s(n) \text{ computing } p_n.\}$$

The arithmetic analogue of P is

Definition 1.0.3.

$$\mathrm{VP} = \cup_{c=1}^{\infty} \mathrm{ACSize}(n^c)$$

Though 'very few' polynomials are in VP, we shall later see that many 'interesting' polynomials are in VP.
Now we turn to polynomials that are not expected to have small circuits. The permanent polynomial is strikingly similar to the determinant, yet it exhibits vastly different properties.

Definition 1.0.4 (Permanent)**.** *For a matrix of variables $X = \{x_{i,j}\}$ we have*

$$\mathrm{Per}_n(X) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} x_{i,\sigma(i)}$$

Unlike the determinant we do not know of polynomial size (or even sub-exponential size) arithmetic circuits for the permanent. The only known arithmetic circuit of size $O(2^n)$ for permanent is obtained from Ryser's Formula [84].

Lemma 1.0.1 (Ryser's Formula)**.** *First we define an auxiliary polynomial*

$$g(X, y_1, \ldots, y_n) = \prod_{i=1}^{n}(1 - 2y_j) \cdot \prod_{i=1}^{n}(\sum_{j=1}^{n} y_j X_{i,j})$$

Now the permanent can be written as an exponential sum over the auxiliary polynomial

$$\mathrm{Per}_n(X) = (-1)^n \sum_{\vec{\alpha} \in \{0,1\}^n} g(X, \vec{\alpha}_1, \ldots, \vec{\alpha}_n)$$

This motivates the definition of a larger class of polynomials known as VNP

Definition 1.0.5. *A p-family $\{f_n\}_{n\in\mathbb{N}} \in VNP$ if there exists another p-family $\{g_n\}_{n\in\mathbb{N}} \in VP$ and a polynomial $p(n)$ such that*

$$f_n(x_1, \ldots, x_n) = \sum_{\boldsymbol{y} \in \{0,1\}^{p(n)}} g_{n+p(n)}(x_1, \ldots, x_n, y_1, \ldots, y_{p(n)})$$

Valiant [107] studied the computational aspects of the permanent. Valiant defined a notion of reductions in arithmetic circuit classes and showed that the permanent is 'complete' for VNP under these reductions.

Definition 1.0.6. *A polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ is said to be a projection of $g \in \mathbb{F}[y_1, \ldots, y_m]$ if there exist affine linear forms $\ell_1, \ldots, \ell_m$ over the variables $x_1, \ldots, x_n$ such that $g(\ell_1, \ldots, \ell_m) = f(x_1, \ldots, x_n)$*

The two classes VP and VNP serve as the arithmetic analogues of P and NP and Valiant conjectured that these two classes are different.

Conjecture 1.0.1 (Valiant's Conjecture)**.**

$$\mathrm{VP} \subsetneq \mathrm{VNP}$$

Progress on this conjecture has been quite slim. The best known result is that of Mignotte and Resyare [69] who show a $\Omega(n^2)$ lower bound on the *determinantal complexity* of permanent. An unexpected line of research on depth reduction [4, 45, 59, 102] showed that general arithmetic circuits of size s computing a polynomial of degree d can be compressed to a depth 4 circuit of size at most $s^{O(\sqrt{d}\log d)}$. For more details the reader is referred to the rolling survey [86], in the hope that the special structure becomes useful. Besides restriction on the depth, several other structural restrictions such as *multilinearity*, *monotonicity*, *formulas*, *arithmetic branching program* have been studied and consequently lower bounds have been proved. For further details the reader is encouraged to read the excellent surveys [28, 97].

1.1 Monotone Arithmetic Circuits

A polynomial is said to be *monotone* if all its coefficients are positive. An arithmetic circuit is said to be *monotone* if every gate of the circuit computes a *monotone* polynomial. For example the circuit in Figure 1.1 is not a monotone circuit. The study of monotone arithmetic circuits was initiated by Schnorr [91] and then Valiant [108] demonstrated that negations can exponentially reduce the resources need to compute certain polynomials. More precisely, he showed that the perfect matching polynomial of a triangular grid graph on n vertices requires monotone circuits of size at least $2^{\Omega(\sqrt{n})}$ but has non-monotone circuits of polynomial size[1]. Soon after Jerrum and Snir [52] exhibited many polynomial families which require monotone circuits of exponential size. Notable amongst them is the spanning tree polynomial which also has polynomial size non-monotone arithmetic circuits due to the classical matrix tree theorem [110]. Thereafter several papers [41, 52, 79, 94, 95, 108] showed similar lower bounds for various polynomials. Yehudayoff noticed that all these lower bounds also end up showing that the corresponding polynomials are not in *monotone* VNP[2]. The monotone analog of Valiant's conjecture was proved in [115] where Yehudayoff exhibited a p-family which is contained in *monotone* VNP but requires *monotone* circuits of size at least $2^{\Omega(\sqrt{n})}$. A year later this was strengthened to a $2^{\Omega(n)}$ lower bound by Srinivasan [98] by employing expanders.

First, we construct a family of polynomials P_n in n variables, each of its monomials has positive coefficient, such that P_n can be computed by a polynomial-size *set-multilinear depth-three formula* but every monotone circuit computing it has size $2^{\Omega(n^{1/4}/\log(n))}$. To the best of our knowledge this also shows the first separation between *multilinear* and *monotone* computation[3]. The polynomial P_n embeds the SINK $\circ$ XOR function devised recently by Chattopadhyay, Mande and Sherif [27] to refute the Log-Approximate-Rank Conjecture in communication complexity. To prove our lower bound for P_n, we develop a general connection between corruption of combinatorial rectangles by any function $f \circ \text{XOR}$ and corruption of product polynomials by a certain polynomial P^f that is an

[1] This is due to the fact that the grid graph is planar and the celebrated FKT algorithm [56, 103] gives a circuit for the perfect matching polynmial over planar graphs.

[2] *Monotone* VNP contains those p-families which can be written as an exponential sum of a p-family in *monotone* VP.

[3] Prior to this thesis there was no example of a monotone polynomial which is computable by *multilinear* arithmetic circuits of small size but requires exponential size when computed by *montone* arithmetic circuits.

arithmetic embedding of f.
Recently Hrubes [48] advocated a new approach for proving lower bounds against general arithmetic circuits. He showed that if $f \in \mathbb{F}[x_1, \ldots, x_n]$ is of degree d and has arithmetic circuits of size s then for some $\epsilon > 0$, the polynomial $(\sum_{i=1}^{n} x_i + 1)^d + \epsilon \cdot f$ has monotone arithmetic circuits of size at most $O(sd^2 + n \log n)$. Thus to show VP $\neq$ VNP, one approach is to exhibit a p-family $f \in$ VNP such that for every $\epsilon > 0$ the polynomial $(\sum_{i=1}^{n} x_i + 1)^d + \epsilon \cdot f$ requires monotone arithmetic circuits of super-polynomial size. We show progress in this approach by exhibiting a p-family $f \in$ VNP for which requires super-polynomial monotone circuits as long as $\epsilon > 2^{-\Omega(n)}$. Our proof relies on ideas from communication complexity and utilizes the discrepancy bound for the Boolean function MOD3 $\circ$ MOD2.

1.2 Algorithms and Arithmetic Circuit Complexity

Arithmetic Circuit Complexity poses many interesting algorithmic problems which have also found application towards solving problems of a combinatorial nature. An important algorithmic problem is polynomial identity testing which is defined as follows:

Problem 1.2.1 (Polynomial Identity Testing (PIT))**.** *Given an arithmetic circuit C computing a polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$. Is $f \equiv 0$?*

The Schwarz-Zippel-DeMillo-Lipton lemma [34, 92, 116] yields a randomized polynomial time algorithm for PITand it is a long standing open problem to find a deterministic polynomial time algorithm for PIT. For progress on PIT the reader is referred to the surveys [88, 89]. In [70] the authors utilize PIT to give an efficient parallel algorithm for detecting perfect matchings on bipartite graphs. In fact the celebrated deterministic algorithm for primality testing [3] was obtained by derandomizing a special case of PIT. Over the years many combinatorial algorithms have been derived by the clever usage of polynomials and appropriate tools from the theory of arithmetic circuits. The reader is referred to the surveys and papers [22, 23, 62, 111–113] for more details. This thesis provides three distinct contributions. First, we extend a PIT algorithm of Forbes and Shpilka [39] to certain partially commutative domains. As an application we make progress towards a conjecture of Valiant [106] in Automata Theory regarding equivalence testing of multi-tape automata. Second, extend the study of Hrubes, Wigderson and Yehudayoff [49] on 'relationless' models of computation who first proved lower bounds in this model. We give deterministic polynomial time algorithms for PIT and irreducibility testing in this model. Third, we study special cases of ideal membership and show applications by computing permanents of low rank matrices and designing algorithms for graph theoretic problems.

Chapter 2

Lower Bounds against Monotone Arithmetic Circuits

Valiant [108] showed that general arithmetic circuits with negation can be exponentially more powerful than monotone ones. We give the first qualitative improvement to this classical result: we construct a family of polynomials P_n in n variables, each of its monomials has a non-negative coefficient, such that P_n can be computed by a polynomial-size *depth-three formula* but every monotone circuit computing it has size $2^{\Omega(n^{1/4}/\log(n))}$. The polynomial P_n embeds the SINK $\circ$ XOR function devised recently by Chattopadhyay, Mande and Sherif [27] to refute the Log-Approximate-Rank Conjecture in communication complexity. To prove our lower bound for P_n, we develop a general connection between corruption of combinatorial rectangles by any function $f \circ \text{XOR}$ and corruption of product polynomials by a certain polynomial P^f that is an arithmetic embedding of f. This connection should be of independent interest.

Using further ideas from communication complexity, we construct another family of set-multilinear polynomials $f_{n,m}$ such that both $F_{n,m} - \epsilon \cdot f_{n,m}$ and $F_{n,m} + \epsilon \cdot f_{n,m}$ have monotone circuit complexity $2^{\Omega(n/\log(n))}$ if $\epsilon \geq 2^{-\Omega(m)}$ and $F_{n,m} \coloneqq \prod_{i=1}^{n} (x_{i,1} + \cdots + x_{i,m})$, with $m = O(n/\log n)$. The polynomials $f_{n,m}$ have $0/1$ coefficients and are in VNP. Proving such lower bounds for monotone circuits has been advocated recently by Hrubeš [48] as a first step towards proving lower bounds against *general circuits* via his new approach.

2.1 Introduction

The arithmetic analog of Cook's P vs. NP question is Valiant's VP vs. VNP question. Despite the latter being seemingly an easier question, progress on it has been frustratingly slow. One class of natural but restricted circuits, in both the Boolean and arithmetic world, is that of monotone circuits where a lot of progress has taken place. Exponential lower bounds on these circuits have been known for long, first in the arithmetic world due to the work of Shamir and Snir [94] and then the breakthrough work of Razborov [81] in Boolean complexity. A long line of work has been carried out both in the arithmetic [41,52,79,94,95, 98,108,115] and the Boolean world (see for example [29,43,47,55,74,76,78,80,81,83,101]) to further strengthen these bounds and make progress. Despite these efforts, several problems remain open even for monotone complexity. In this chapter, we focus on two such problems.

The first problem concerns the understanding of the relative powers of monotone and non-monotone computation. How much advantage do non-monotone circuits have, exploiting cancellation, to compute a target function or polynomial that itself is monotone? This was first answered in the arithmetic world by Valiant [108], forty years ago. Valiant showed that a certain monotone polynomial can be computed efficiently by general circuits, but monotone circuits need exponential size to compute it. In the Boolean world, Razborov [80] gave a super-polynomial separation between monotone and non-monotone computations by showing that the bipartite matching problem in P needs $n^{\Omega(\log n)}$-size monotone circuits. Later, Tardos [101] proved an exponential separation using a different function. Raz and Wigderson [78] showed that matching needs exponential size monotone formulas.

Several researchers have investigated the following natural question that arises from these separation results: what is the weakest non-monotone model that can compute a monotone function which is hard even for the most powerful monotone model? It is known that matching can not only be computed in P, but it can also be computed efficiently, i.e. using polynomially many processors, in fast parallel time or depth of $O(\log^2 n)$ by randomized algorithms again using cancellations. Yet, Razborov's lower bound puts it outside monotone P. This was the best known separation for long. Recently, Göös et. al. [43] improved this by finding another function that has such an efficient and parallel deterministic algorithm and showed that it even requires exponential size monotone circuits. Can one find hard functions for monotone circuits in even shallower depths of general circuits? An important result of Ajtai and Gurevich [5] showed that there are monotone functions that can be computed efficiently by constant-depth, non-monotone circuits, and yet need super-polynomial size to be computed by constant-depth monotone circuits. The recent work of Chen et.al. [29] significantly strengthens this by exhibiting a monotone function having efficient constant-depth circuits with negations but that still requires exponential size to be computed by monotone circuits of constant depth even though they are allowed to use gates computing a powerful monotone function like Majority. However, these developments still leave tantalizingly open the possibility of finding a monotone function computable by small size constant-depth circuits with negation that require *exponential size* to be computed by *unrestricted-depth* monotone circuits.

The arithmetic analog of this possibility remained unresolved as well. However, for *non-commutative* circuits, it was answered in the positive by Hrubeš and Yehudayoff [50] by

exhibiting an exponential separation of constant-depth arithmetic formulas and monotone circuits. We provide the first such separation in the more natural and commonly studied setting of *commutative* arithmetic circuits that we describe next.

We consider set-multilinear monomials over sets of variables $X_1, X_2, \ldots, X_n$ where $X_i := \{x_{i,1}, x_{i,2}, \ldots, x_{i,m}\}$, i.e. multilinear monomials that depend precisely on just one of the m variables from each of the n blocks[1]. In order to generate hard polynomials for monotone arithmetic circuits, we will use an idea of embedding a Boolean function f that is known to be hard in the 2-party setting of communication complexity. The general idea to do so, is to associate with each multilinear monomial κ a unique m-bit Boolean string x. The coefficient of κ in the polynomial would be just $f(x) \in \{0,1\}$. Applying this general framework, for each $f : \{0,1\}^m \to \{0,1\}$ one derives a unique monotone polynomial P^f with 0/1 coefficients.

However, we want our target polynomial to be also easy to be computed by constant-depth arithmetic formulas, using the power of cancellations. This makes the design of associating a Boolean string to a monomial and the choice of f delicate. In particular, we will have to perturb a Boolean f at each point by a slight amount δ to create a real-valued f' that uniformly approximates f. We will then be able to argue that $P^{f'}$ retains the monotone hardness of P^f but just becomes easy to be computed with cancellations. Creating hard polynomials for arithmetic circuits in this way, as far as we know, was not done in any of the prior works.

More precisely, the set of such multilinear monomials can be identified with the set of all mappings from $[n]$ to $[m]$, the latter denoted by $\mathcal{F}_{n,m}$. For a mapping $\sigma \in \mathcal{F}_{n,m}$ (or a set-multilinear monomial κ) we define a parity vector $\vec{\oplus}(\sigma) \in \{0,1\}^m$ (or $\vec{\oplus}(\kappa)$) where the jth entry is $|\sigma^{-1}(j)| \pmod 2$. The parity vector $\vec{\oplus}(\kappa)$ of a monomial κ will be the unique m-bit Boolean string we associate with it. Now we describe the hard Boolean function f that we will be using: this will be the same function called SINK that was recently used in the refutation of the Log-Approximate-Rank Conjecture (LARC) in [27]. Set $m = \binom{k}{2}$, so that each bit of a string x in $\{0,1\}^m$ is viewed as the assignment to an edge of a complete graph on k vertices $\{1, \ldots, k\}$. If a bit of x, corresponding to an edge (u, v) in K_k is set to 0, then the edge orients $u \leftarrow v$, otherwise the edge orients $u \to v$, when $u < v$. An input string x, can thus be interpreted as a tournament by orienting the edges of the complete graph K_k. A vertex in a tournament is called a sink vertex if its out-degree is 0. SINK evaluates to 1 on x if the tournament specified by x has a sink vertex. Otherwise, SINK evaluates to 0 on x.

Thus, a monomial κ is called a sink monomial if $\text{SINK}(\vec{\oplus}(\kappa)) = 1$. Let $P^{\text{Non-Sink}}_{n,m}$ ($P^{\text{Sink}}_{n,m}$) be the polynomial that is supported completely on non-sink (sink) monomials. It can be shown that each of $P^{\text{Non-Sink}}_{n,m}$ and $P^{\text{Sink}}_{n,m}$ is hard for monotone circuits, but we don't know whether any one of them is easy for general formulas. We thus look at a slight perturbation of $P^{\text{Non-Sink}}_{n,m}$: A polynomial $P_{n,m}$ is called a δ-non-sink polynomial if the coefficient of every monomial κ in $P_{n,m}$ lies in the interval $[0, \delta]$ if κ is a sink, otherwise (i.e. κ is not a sink) it lies in the interval $[1 - \delta, 1]$. We are ready to state our result now:

Theorem 2.1.1.

1. *For every constant $0 < \delta < 1$, there exists a δ-non-sink polynomial $P_{\delta,n,m}$ that has a depth-three arithmetic formula of size $O_\delta(nm^4)$.*

[1]Many commonly studied polynomials like the Permanent and Determinant are set-multilinear.

2. *There exists a sufficiently small constant $\delta < 1$, such that every δ-non-sink polynomial $Q_{\delta,n,m}$ needs monotone circuits (with no depth restrictions) of size $2^{\Omega(\sqrt{m})}$ to be computed, when $4m \ln m \leq n$.*

Remark 2.1.1. *Let us recall that Valiant [108], Jerrum and Snir [52] gave exponential separations between VP and monotone VP. Applying modern depth reduction techniques [4, 45, 59, 102] to either of their polynomials would result in them being computed by depth-3/depth-4 formulas of exponential size of $2^{O(\sqrt{d}\log N)}$, where the polynomials are N-variate and degree d. This would result in just a super-polynomial separation between depth-3/depth-4 circuits and monotone arithmetic circuits. Our theorem gives an exponential separation.*

More generally, the use of parity vectors of monomials as described above, gives us a conceptually simple but novel way of constructing set-multilinear polynomials by embedding hard Boolean functions of the form $f \circ \mathrm{XOR}$, where $f : \{0,1\}^m \to \{0,1\}$. These functions are called XOR functions and we make more use of them, via parity vectors of monomials, to deal with a different problem.

We consider the problem of making progress towards proving lower bounds for general arithmetic circuits. Efforts in this direction have remained stuck for a long time. Very recently, Hrubeš [48] formulated a new interesting approach that reduces the task of proving lower bounds on the size of general arithmetic circuits to that of proving lower bounds on the monotone complexity of special classes of polynomials. A crucial aspect of the approach is that one, sort of, *plants* a conjectured hard polynomial scaled by a vanishingly small number ϵ inside an otherwise super-easy polynomial. One needs to prove that the new polynomial remains hard for monotone circuits to conclude that the conjectured hardness of the original polynomial against general circuits is true. There are no known monotone lower bounds against such polynomials and Hrubeš argues that new techniques in monotone complexity need to be developed to take on this challenge. Using our communication complexity based approach, we take a first step in this direction.

More precisely, Hrubeš showed that if a polynomial f_n is computed efficiently by a general circuit of size s, then there exists an $\epsilon_0 > 0$, such that for every $\epsilon \leq \epsilon_0$, the function $F_n + \epsilon \cdot f$ has efficient monotone circuits, where $F_n := (1 + \sum_i x_i)^d$ is the polynomial that contains all monomials of degree at most d. The difficulty in proving monotone lower bounds for such polynomials is the following: for $\epsilon = 0$, they become easy for monotone circuits and yet we need to show that for a tiny non-zero ϵ, the function is hard. Most monotone lower bounds in the literature are based on covering arguments, i.e. they are valid against all polynomials which are supported on the same hard set of monomials, paying no regard to the specific set of coefficients used in the target polynomial. Such arguments cannot obviously work against the kind of polynomials suggested by Hrubeš. Recently, Yehudayoff [115] and Srinivasan [98] gave new arguments to prove monotone lower bounds that take into consideration the distribution of coefficients. In fact, covering arguments would not be enough to even prove our Theorem 2.1.1 as the support set of a δ-non-sink polynomial can be full and our argument does make essential use of the *distribution* of the coefficients of monomials. Still, it is not clear how to use any of these arguments in the context of Hrubeš' question.

We give a new argument based on the discrepancy method from communication complexity to make progress. To discuss this approach, let us consider the Boolean function $\mathrm{MOD}_3 \circ$

XOR, where $\mathrm{MOD}_3(x) = 0$ if and only if $\sum_{i=1}^{n} x_i \equiv 0 \bmod 3$. It can be shown that this problem has small discrepancy w.r.t combinatorial rectangles and is therefore hard for Alice and Bob to solve in Yao's 2-party communication model. Our main insight is that embedding such a problem in a set-multilinear polynomial via the parity vectors of the monomials results in a monotone polynomial that is very hard for monotone circuits in the sense of Hrubeš' Theorem. More precisely, we define

$$P_{n,m}^{\mathrm{MOD}_3} := \sum_{\substack{\sigma:[n]\mapsto[m] \\ \mathrm{MOD}_3(\vec{\oplus}(\sigma))=0}} \prod_{i=1}^{n} x_{i,\sigma(i)}.$$

Clearly this polynomial is in VNP. Let the full set-multilinear polynomial be defined as

$$F_{n,m} := \prod_{i=1}^{n} (x_{i,1} + \cdots + x_{i,m}).$$

Our main result is the following:

Theorem 2.1.2. *There exists a constant $\gamma > 0$ such that both the polynomials $F_{n,m} - \epsilon \cdot P_{n,m}^{MOD_3}$ and $F_{n,m} + \epsilon \cdot P_{n,m}^{MOD_3}$ have monotone complexity $2^{\Omega(m)}$, provided $4m \ln m \leq n$ and $\epsilon \geq 2^{-\gamma m}$.*

In summary, both Theorems 2.1.1 and 2.1.2 obtain monotone lower bounds of the kinds that were not obtained before our work. They use two powerful techniques from communication complexity: the corruption and the discrepancy method. That communication complexity methods and arithmetic complexity lower bound techniques should be related is not a complete surprise. A low cost communication protocol induces a decomposition of the communication matrix of the target function into a non-negative sum of few rectangles. A small monotone circuit results in the decomposition of the computed polynomial into a sum of non-negative product polynomials. In fact, several researchers have used this similarity to draw intuition from communication complexity to prove monotone lower bounds. However, there is an important difference (among others) that, we feel, has prevented direct usage of measures like corruption bounds in past work. In the arithmetic decomposition, the partition of the input variables used varies across the product polynomials used. In standard version of communication complexity, this does not happen. One chooses the most convenient possible partition to prove lower bounds. Raz and Yehudayoff [79] used sophisticated exponential sum estimates of [24] to overcome this difficulty. We, on the other hand, use a simple but novel trick of using parity vectors to embed a hard XOR function in our target polynomial. Our general Corruption Transfer Lemma 2.4.1 and the proof of Theorem 2.1.2 are evidences of the broad applicability of this idea. We feel that this would also be further useful in proving lower bounds for circuits that are less restricted than monotone ones.

2.1.1 Proof Ideas and Techniques

Separating General Depth-3 From Monotone Circuits

We are looking for polynomials (with positive coefficients) that are very hard for monotone circuits, yet are not only easy with cancellations but even remain easy in constant-depth.

There are two important computations in which cancellations are known to help. First, is the computation of the determinant. This is not a monotone polynomial but one can effectively embed it into a such a polynomial. For example, Jerrum and Snir [52] used spanning tree polynomial to separate VP from monotone VP and it can be expressed as a determinant of linear forms [110]. But determinant is unlikely to be easy for small depth-computations. Ben-Or [97] showed that depth three is capable of interpolating which does make crucial use of cancellations. This yields small-size depth-3 circuits for computing elementary symmetric polynomials, making them a possible target for separating the power of constant-depth general formulas and monotone unrestricted-depth circuits. However, it is well known that elementary symmetric polynomials are actually easy for even monotone arithmetic branching programs (ABPs).

This forces us to seek alternative powers of cancellations: depth-2 circuits cannot be more powerful than their monotone counterparts. How do cancellations in random depth-three set-multilinear circuits take place? Consider the following $\Sigma\Pi\Sigma$ circuit:

$$\frac{1}{N}\sum_{i=1}^{N}\prod_{j=1}^{n}\left(b_1^i x_{j,1} + b_2^i x_{j,2} + \cdots + b_m^i x_{j,m}\right) \tag{2.1}$$

Here, $b^1, \ldots, b^N$ are randomly sampled points from $\{1,-1\}^m$. To talk about cancellations, let us consider the coefficient of a monomial $\kappa \coloneqq \prod_{j=1}^{n} x_{j,\sigma(j)}$, where $\sigma : [n] \to [m]$ is the map defining κ. The coefficient of κ in the polynomial computed by the random circuit is $(1/N)\cdot\sum_{i=1}^{N}\prod_{j=1}^{n} b^i_{\sigma(j)}$ which can be re-written as

$$\frac{1}{N}\sum_{i=1}^{N}\prod_{\ell=1}^{m}(b_\ell^i)^{\left|\sigma^{-1}(\ell)\right| \bmod 2} \tag{2.2}$$

Thus, the parity vector $\vec{\oplus}(\kappa) \coloneqq (|\sigma^{-1}(1)| \bmod 2, \ldots, |\sigma^{-1}(m)| \bmod 2)$ of monomial κ determines its coefficient. All monomials that are even, i.e. their parity vectors are all zeroes, will have coefficient exactly 1, i.e. there was no cancellation for them. For any odd monomial κ, the expected value of the coefficient is 0, i.e. we expect a lot of cancellations associated with κ taking place. To translate this phenomenon from random circuits to a fixed deterministic circuit, we choose the points $b^1, \ldots, b^N$ to form an ϵ-biased space in $\{0,1\}^m$. Armed with this insight, we want to craft a polynomial where the deterministic circuit is able to suppress the magnitude of the coefficients of a select group of monomials while keeping the rest of the magnitudes high. The group to be selected should be such that the polynomial becomes hard for monotone circuits.

The basic weakness of a monotone circuit of size s computing a multilinear polynomial P is well known i.e. such a P can be expressed as a sum of few *balanced product* polynomials. Each such product polynomial has a structure resembling that of a *combinatorial rectangle*, an object that appears commonly in the study of communication complexity. This similarity has been the source of intuition in past works in arithmetic complexity in general and monotone complexity in particular. But a direct correspondence had not been established, as far as we know, until now. We do so in this chapter and crucially use this correspondence to prove our monotone lower bounds.

Before we describe further details, we point out an interesting connection to the famous Log-Rank Conjecture in communication complexity and our problem. Consider any set

multilinear polynomial P^f and a partition $\{\mathcal{X}_1, \mathcal{X}_2\}$ of the sets of its input variables. A natural matrix w.r.t the partition is one where each row corresponds to a set-multilinear monomial in variables from $\mathcal{X}_1$ and every column to one in variables from $\mathcal{X}_2$. Each entry of this matrix is the coefficient in the target polynomial of the monomial formed by the product of the corresponding row and column monomials. Observe that if P^f is computed by a syntactic depth-3 set-multilinear circuit of top fain-in s, then the matrix has rank at most s, as every product gate computes polynomial whose corresponding matrix has rank 1. Let our polynomial P^f be the embedding of the Boolean function f by the parity vector scheme. This translates into saying that the communication matrix of $f \circ \mathrm{XOR}$ has rank at most s. To prove our lower bound on the monotone circuit complexity of P^f, current techniques end up proving lower bounds on the number of product polynomials needed in any decomposition of P^f. Each product polynomial appearing in the sum comes with its own partition. But even if all these partitions were $\{\mathcal{X}_1, \mathcal{X}_2\}$, this would imply proving a lower bound on the number of combinatorial rectangles needed to non-negatively sum up to the communication matrix of $f \circ \mathrm{XOR}$. A strong lower bound on the monotone complexity of P^f when s, the upper bound on the top fan-in of a depth-3 circuit computing P^f, is just polynomial in n, m would thus result in the refutation of the Log-Rank Conjecture (LRC) via the function $f \circ \mathrm{XOR}$. No refutation of the LRC is known. Under these circumstances, we do the next best possible thing: we take recourse to the recent refutation of the approximate/randomized version of the LRC [27], by embedding an *approximate* version of the f in our polynomial, where f is the same function SINK used in the refutation.

The set of points at which SINK outputs 1 is a small union of mutually disjoint sub-cubes. Roughly speaking, we observe that parity vectors in each sub-cube can be expressed by a single $\Sigma\Pi\Sigma$ circuit of the form in (2.1) by appropriately 'shifting' the parity vector. Thus, we express S by writing the polynomial as a sum of few (as many as the number of sub-cubes) such depth-3 circuits and then collapse the whole thing naturally to a single depth-3 circuit. The resulting polynomial has negative coefficients. We turn it monotone by subtracting it from a slightly scaled-up full product polynomial.

This polynomial is a δ-non-sink polynomial. As it has nearly full support, one needs to find an argument that uses the distribution of its coefficients to prove its hardness against balanced product polynomials. Our simple but key insight is to regard this polynomial's coefficients as the acceptance probabilities of the parity vectors of the respective monomials. In other words, just as $\mathrm{SINK} \circ \mathrm{XOR}$ was shown by [27] to be hard to be pointwise δ-approximated by a small non-negative sum of combinatorial rectangles, we should in principle be able to say that a small sum of non-negative product polynomials cannot compute any δ-non-sink polynomial. Realizing this idea requires care. More interestingly in doing so, we develop a general transfer theorem that relates *rectangular corruption*, a very useful measure in communication complexity, under natural probability distributions on the input space of Boolean XOR functions to that of an analogous corruption-like measure on the set-multilinear monomial space. This simple but powerful correspondence is developed in Section 2.4. While our immediate use of this correspondence is to establish the lower bound for δ-non-sink polynomials to prove Theorem 2.1.1, we believe this to be of independent interest in monotone complexity.

ϵ-Sensitive Monotone Lower Bounds

Now we briefly explain the main ideas for proving Theorem 2.1.2. The crucial insight comes from the fact that the boolean function $\mathrm{MOD}_3 \circ \mathrm{XOR}$ has small *discrepancy* w.r.t the combinatorial rectangles. The notion of discrepancy was defined by Babai, Nisan and Szegedy [16]. To exploit this, we define two measures W_0 and W_1 on the space of set-multilinear monomials as follows: W_0 puts uniform weights to the set of monomials κ such that the Hamming weight (wt) of $\vec{\oplus}(k)$ is a multiple of 3. The measure W_1 acts similarly on the set of monomials κ such that $\mathrm{wt}(\vec{\oplus}(\kappa)) \equiv 1 \pmod 3$. Combining W_0 and W_1, we define the main measure W on any polynomial P as $W(P) = W_1(P) - W_0(P)$.

Using a (nearly)-equidistribution property of parity vectors, we show that the number of monomials κ such that $\mathrm{wt}(\vec{\oplus}\kappa) \equiv b \pmod 3$ are roughly same for $b \in \{1, 2, 3\}$. This immediately shows that the contribution of the measure W for the polynomial $P = F_{n,m} - \epsilon \cdot P_{n,m}^{\mathrm{MOD}_3}$ is approximately proportional to the contribution from $P_{n,m}^{\mathrm{MOD}_3}$. In particular, this shows that $W(P) \geq O(\epsilon)$.

Further, the equidistribution property and a simple exponential sum estimate help us in proving that the measure $W(a \cdot b)$ is exponentially small for any balanced product polynomial. Conceptually, this step is a transfer of small discrepancy of $\mathrm{MOD}_3 \circ \mathrm{XOR}$ function w.r.t the combinatorial rectangles to the product polynomials. Since the measure of $W(a \cdot b)$ is exponentially small, the sub-additive property of W shows that the number of product polynomials needed to account for $W(P)$ must be large (for a suitable range of values for ϵ). Finally, the lower bound follows from the structure theorem of monotone circuits which says that if P is computable by a polynomial size monotone circuit, then P can be written as a small sum of balanced product polynomials.

2.2 Preliminaries

Notation

Let $[n] = \{1, 2, \ldots, n\}$. For a polynomial $p \in R[X]$ and a monomial κ, let $p[\kappa]$ be the coefficient of κ in p. For polynomials $p, q \in R[X]$, we write $p \leq q$ if for each κ, we have that $p[\kappa] \leq q[\kappa]$. For a polynomial p, let $\mathrm{var}(p)$ denote the set of variables in p. For a vector $u \in \{0, 1\}^n$, the notation $\mathrm{wt}(u)$ is used for the Hamming weight of u.

Set-Multilinear Polynomials

Let $X = \cup_{i=1}^{n} X_i$ be a set of variables where $X_i = \{x_{i,1}, x_{i,2}, \ldots, x_{i,m}\}$. A polynomial $p \in R[X]$ is set-multilinear if each monomial in p respects the partition given by the set of variables $X_1, X_2, \ldots, X_n$. In other words, each monomial κ in p is of the form $x_{1,j_1} x_{2,j_2} \cdots x_{n,j_n}$. One can naturally associate a function (mapping) $\sigma : [n] \to [m]$ such that $\sigma(i) = j_i$. This association forms a bijection between the space of all functions from $[n]$ to $[m]$, denoted by $\mathcal{F}_{n,m}$ and the space of all such set-multilinear monomials. The cardinality of each set is easily seen to be m^n. Often we shall abuse notation to identify the monomial κ with the function it represents.

Parity Vectors of Set-Multilinear Monomials

For a set-multilinear monomial κ with the associated function σ we define the parity vector $\vec{\oplus}(\kappa) \in \{0,1\}^m$ where the j^{th} entry is $\vec{\oplus}(\kappa)_j = |\sigma^{-1}(j)| \pmod 2$. For a set of parity vectors $S \subseteq \{0,1\}^m$ we shall denote

$$\mathcal{K}(S) = \{\kappa \mid \vec{\oplus}(\kappa) \in S\}.$$

Ordered Polynomial

For a monomial of the form $\kappa = x_{i_1,j_1}x_{i_2,j_2}\cdots x_{i_n,j_n}$ we define the set $I(\kappa) = \{i_1, i_2, \ldots, i_n\}$. If a polynomial p has the same set $I(\kappa)$ for every monomial occurring in it with a non-zero coefficient, then we say that the polynomial is ordered and we write $I(p) = I(\kappa)$ for each κ. Clearly, the set-multilinear polynomials are ordered polynomials with $I(p) = \{1, 2, \ldots, n\}$.

Structure of Monotone Circuits

The main structural result for monotone circuits that we use throughout, is the following theorem.

Theorem 2.2.1. *[115, Lemma 1] Let $n > 2$ and $p \in \mathbb{R}[X]$ be an ordered monotone polynomial with $I(p) = [n]$. Let C be a monotone circuit of size s that computes p. Then, we can write*

$$p = \sum_{t=1}^{s} a_t \cdot b_t$$

where a_t and b_t are monotone ordered polynomials with $\frac{n}{3} \leq |I(a_t)| \leq \frac{2n}{3}$ and $I(b_t) = I(a_t) \setminus [n]$. Moreover, $a_t \cdot b_t \leq p$ for each $1 \leq t \leq s$.

Such ordered product polynomials $a \cdot b$ with $\frac{n}{3} \leq |I(a)| \leq \frac{2n}{3}$ and $I(b) = [n] \setminus I(a)$ will be called balanced product polynomials.

Rectangular Corruption

We recall here the concept of corruption measure from communication complexity that we make use of in this chapter. To do so, let us very briefly first recall the basic notions in the 2-party communication model of Yao. The joint input space of Alice and Bob is $\{0,1\}^m \times \{0,1\}^m$ with each player receiving an m-bit Boolean string, and they want to evaluate a Boolean function $F : \{0,1\}^m \times \{0,1\}^m \to \{0,1\}$. One defines a combinatorial rectangle R as a product set $A \times B$, for some $A, B \subseteq \{0,1\}^m$. Put another way, R is just a submatrix of the $2^m \times 2^m$ communication matrix M_F of the function F, that Alice and Bob want to compute. The rows of this matrix are indexed by possible inputs of Alice and the columns by the ones of Bob and $M_F(x,y) = F(x,y)$. To define it, consider a probability distribution λ on $\{0,1\}^m \times \{0,1\}^m$ that is almost balanced. Then, the intuition is that λ renders F hard, if rectangles (sub-matrices) of F (M_F) cannot even be *approximately monochromatic* unless they are small as measured by λ. We will use the following notion of corruption to measure approximate monochromaticity. This was implicitly defined by Razborov [82].

Let $z \in \{0,1\}$. Then, the ϵ, z-*corruption* of F w.r.t. λ is denoted by $\mathrm{Corr}^z_{\lambda,\epsilon}(F)$, which is defined as follows:

$$\mathrm{Corr}^z_{\lambda,\epsilon}(F) := \min_{R:\lambda(R\cap F^{-1}(z))\leq\epsilon\lambda(R)} \log\Big(\frac{1}{\lambda(R)}\Big).$$

The importance of the measure above lies by the fact that several lower bounds on the randomized communication complexity of functions, beginning with the famous one for Set-Disjointness, are proved by the simple relation: $R_\epsilon(F) \geq \Omega(\mathrm{Corr}^z_{\lambda,\epsilon}(F))$, for each $z \in \{0,1\}$, where $R_\epsilon(F)$ is the ϵ-error randomized communication complexity of F.

Small Bias Spaces

We recall the well-known notion of ϵ-biased spaces.

Definition 2.2.1 (ϵ-biased space)**.** *A multi-set $\mathcal{B} \subseteq \{-1,+1\}^m$ of size N is called an ϵ-biased space if for every subset $S \subseteq [m]$ we have*

$$|\frac{1}{N}\sum_{b\in\mathcal{B}}\prod_{i\in S} b_i| < \epsilon.$$

Recently, a breakthrough work of Ta-Shma [100] gives near optimal size explicit construction of ϵ-biased spaces.

Theorem 2.2.2 (Explicit construction of ϵ-bias spaces [100])**.** *There is a deterministic algorithm that for every $\epsilon > 0$ constructs an ϵ-biased space $\mathcal{B}_{m,\epsilon}$ of size $O(\frac{m}{\epsilon^{2+o(1)}})$. The algorithm runs in time* $\mathrm{poly}(m, \frac{1}{\epsilon})$.

2.3 Equidistribution of Parity Vectors

In this section we record a few combinatorial results which will be used throughout the chapter. We establish an approximate equidistribution property of $\mathcal{F}_{n,m}$ which is the set of functions from $[n]$ to $[m]$. For a function $\sigma \in \mathcal{F}_{n,m}$ we define the parity vector $\vec{\oplus}(\sigma) \in \{0,1\}^m$ where the j^{th} entry is $\vec{\oplus}(\sigma)_j = |\sigma^{-1}(j)| \pmod 2$. For a vector $v \in \{0,1\}^m$ we define

$$\mathcal{F}^v_{n,m} = \{\sigma \mid \sigma \in \mathcal{F}_{n,m} \text{ such that } \vec{\oplus}(\sigma) = v\ \}.$$

We will show that $\mathcal{F}_{n,m}$ is partitioned into *approximately* equal size classes $\mathcal{F}^v_{n,m}$ (where $v \in \{0,1\}^m$) in the sense of Corollary 2.3.1. The following fact is easy to verify using a symmetry argument.

Fact 2.3.1. $|\mathcal{F}^v_{n,m}| = |\mathcal{F}^u_{n,m}|$ *if* $\mathrm{wt}(u) = \mathrm{wt}(v)$.

Proof. Let $\mathrm{ones}(u)$ denote the set of indices where u is 1. Let $\psi : [m] \mapsto [m]$ be a bijection that maps $\mathrm{ones}(v)$ to $\mathrm{ones}(u)$. For a function $\sigma \in \mathcal{F}^v_{n,m}$ we have $\psi \circ \sigma \in \mathcal{F}^u_{n,m}$. This gives a bijection $\psi : \mathcal{F}^v_{n,m} \mapsto \mathcal{F}^u_{n,m}$ which completes the proof of the fact. ∎

Now we write a recurrence for $|\mathcal{F}^v_{n,m}|$. Due to Fact 2.3.1, we may assume that the vector is of the form $v = (0,0,0,\ldots,1,1,1)$. Then we have,

$$|\mathcal{F}^v_{n,m}| = \sum_{\substack{k=0 \\ k \text{ is even}}}^{n-\mathrm{wt}(v)} \binom{n}{k} |\mathcal{F}^{v_{-1}}_{n-k,m-1}| \tag{2.3}$$

where v_{-1} is the vector obtained from v by deleting the first coordinate. If $v = (1,1,1,\ldots,1,1)$ then the recurrence is

$$|\mathcal{F}^v_{n,m}| = \sum_{\substack{k=1 \\ k \text{ is odd}}}^{n-\mathrm{wt}(v)} \binom{n}{k} |\mathcal{F}^{v_{-1}}_{n-k,m-1}|.$$

Claim 2.3.1. *$|\mathcal{F}^v_{n,m}| \geq |\mathcal{F}^u_{n,m}|$ when $wt(v) = wt(u) - 2$.*

Proof. The proof is by induction on m with base case at $m = 2$. Notice that for $m = 2$, $v = (0,0)$ and $u = (1,1)$. Hence

$$|\mathcal{F}^v_{n,m}| = \sum_{\substack{k=0 \\ k \text{ is even}}}^{n-\mathrm{wt}(v)} \binom{n}{k} = 2^{n-1}.$$

Similarly,

$$|\mathcal{F}^u_{n,m}| = 2^{n-1}.$$

Writing the recurrence for $|\mathcal{F}^v_{n,m}|$ we have

$$|\mathcal{F}^v_{n,m}| = \sum_{\substack{k=0 \\ k \text{ is even}}}^{n-\mathrm{wt}(v)} \binom{n}{k} |\mathcal{F}^{v_{-1}}_{n-k,m-1}|.$$

Applying the induction hypothesis for $m-1$ on the vectors v_{-1} and u_{-1} we conclude that $|\mathcal{F}^{v_{-1}}_{n-k,m-1}| \geq |\mathcal{F}^{u_{-1}}_{n-k,m-1}|$ which shows that,

$$\begin{aligned}
|\mathcal{F}^v_{n,m}| &= \sum_{\substack{k=0 \\ k \text{ is even}}}^{n-\mathrm{wt}(v)} \binom{n}{k} |\mathcal{F}^{v_{-1}}_{n-k,m-1}| \\
&\geq \sum_{\substack{k=0 \\ k \text{ is even}}}^{n-\mathrm{wt}(u)} \binom{n}{k} |\mathcal{F}^{u_{-1}}_{n-k,m-1}| + \binom{n}{n-\mathrm{wt}(v)} |\mathcal{F}^{v_{-1}}_{\mathrm{wt}(v),m-1}| \\
&\geq \sum_{\substack{k=0 \\ k \text{ is even}}}^{n-\mathrm{wt}(u)} \binom{n}{k} |\mathcal{F}^{u_{-1}}_{n-k,m-1}| \\
&= |\mathcal{F}^u_{n,m}|.
\end{aligned}$$

This completes the induction. ∎

When n is even we shall denote $\mathcal{F}^{(0,0,\ldots,0,0)}_{n,m}$ as $\mathcal{EF}_{n,m}$ and call it the set of even functions. Next we derive an upper bound on the number of even functions.

Lemma 2.3.1.

$$|\mathcal{EF}_{n,m}| \leq \frac{1}{2^{m-1}} \left(\sum_{i=0}^{\frac{m-2}{2}} \binom{m}{i} (m-2i)^n \right) \quad \textit{when } m \textit{ is even}$$

$$\leq \frac{1}{2^{m-1}} \left(\sum_{i=0}^{\frac{m-1}{2}} \binom{m}{i} (m-2i)^n \right) \quad \textit{when } m \textit{ is odd.}$$

Proof. The proof is by induction on m and the base case is $m = 2$. We have

$$\mathcal{EF}_{n,2} = \sum_{\substack{k=0 \\ k \text{ is even}}}^{n} \binom{n}{k} = 2^{n-1}.$$

and hence the base case is established. We do the induction step when $m+1$ is odd (the even case when $m+1$ is even is similar). We write the recurrence and apply the induction hypothesis.

$$\begin{aligned}
|\mathcal{EF}_{n,m+1}| &= \sum_{\substack{k=0 \\ k \text{ is even}}}^{n} \binom{n}{k} |\mathcal{EF}_{n-k,m}| \\
&\leq \sum_{\substack{k=0 \\ k \text{ is even}}}^{n} \binom{n}{k} \frac{1}{2^{m-1}} \left(\sum_{i=0}^{\frac{m-2}{2}} \binom{m}{i} (m-2i)^{n-k} \right) \\
&\leq \frac{1}{2^{m-1}} \sum_{i=0}^{\frac{m-2}{2}} \binom{m}{i} \left(\sum_{\substack{k=0 \\ k \text{ is even}}}^{n} \binom{n}{k} (m-2i)^{n-k} \right) \\
&\leq \frac{1}{2^m} \left(\sum_{i=0}^{\frac{m-2}{2}} \binom{m}{i} \left((m+1-2i)^n + (m-1-2i)^n \right) \right) \\
&\leq \frac{1}{2^m} \left(\sum_{i=0}^{\frac{m-2}{2}} \binom{m}{i} \left((m+1-2i)^n + (m+1-2(i+1))^n \right) \right) \\
&\leq \frac{1}{2^m} \left(\binom{m+1}{0} (m+1)^n + \sum_{i=1}^{\frac{m}{2}} \left[\binom{m}{i-1} + \binom{m}{i} \right] \left((m+1-2i)^n \right) \right) \\
&= \frac{1}{2^m} \left(\binom{m+1}{0} (m+1)^n + \sum_{i=1}^{\frac{m}{2}} \binom{m+1}{i} \left((m+1-2i)^n \right) \right).
\end{aligned}$$

This completes the induction.

■

From the above bounds we derive the main equidistribution property which will be used repeatedly.

Corollary 2.3.1 (Key Equidistribution Property)**.** *Let n be even. If $m \ln m \leq n$, then for every $v \in \{0,1\}^m$ we have,*

$$|\mathcal{F}^v_{n,m}| \leq \frac{4m^n}{2^m}.$$

Proof. We shall prove the bound for $|\mathcal{EF}_{n,m}|$ and then by lemma 2.3.1 the bound will hold for $|\mathcal{F}^v_{n,m}|$ for all $v \in \{0,1\}^m$. From Lemma 2.3.1 we know that

$$\begin{aligned}
|\mathcal{EF}_{n,m}| &\leq \frac{1}{2^{m-1}} \left(\sum_{i=0}^{\frac{m-2}{2}} \binom{m}{i} (m-2i)^n \right) \\
&\leq \frac{m^n}{2^{m-1}} \left(\sum_{i=0}^{\frac{m-2}{2}} \binom{m}{i} \left(\frac{m-2i}{m} \right)^n \right) \\
&\leq \frac{m^n}{2^{m-1}} \left(\sum_{i=0}^{\frac{m-2}{2}} m^i \left(1 - \frac{2i}{m} \right)^n \right) \\
&\leq \frac{m^n}{2^{m-1}} \left(\sum_{i=0}^{\frac{m-2}{2}} m^i e^{\frac{-2in}{m}} \right) \\
&\leq \frac{m^n}{2^{m-1}} \left(\sum_{i=0}^{\frac{m-2}{2}} \left(\frac{m}{e^{\frac{2n}{m}}} \right)^i \right).
\end{aligned}$$

Now by the choice of n and m we have $\frac{m}{e^{\frac{2n}{m}}} \leq \frac{1}{m}$ and hence we may bound

$$\sum_{i=0}^{\frac{m-2}{2}} \left(\frac{m}{e^{\frac{2n}{m}}} \right)^i \leq \sum_{i=0}^{\frac{m-2}{2}} \left(\frac{1}{m} \right)^i \leq \frac{m}{m-1} \leq 2.$$

Thus we conclude that when $m \ln m \leq n$ we have $|\mathcal{EF}_{n,m}| \leq \frac{4m^n}{2^m}$.

■

Henceforth, in the rest of this chapter, without loss of generality, we assume that n is even.

2.4 Corruption Transfer from Rectangles to Product Polynomials

Let $f : \{0,1\}^m \mapsto \{0,1\}$ be a boolean function and let $F = f \circ \text{XOR}$ be the boolean function defined on $\{0,1\}^{2m}$ as $F(x,y) = f(x \oplus y)$. Suppose F has high *corruption* with respect to rectangles i.e. $\text{Corr}^z_{\lambda,\nu}(F) \geq \log\left(\frac{1}{T}\right)$, or in other words:

$$\begin{aligned}
\text{if} \quad & \lambda(R \cap F^{-1}(z)) \leq \nu\lambda(R) \\
\text{then} \quad & \lambda(R) \leq T
\end{aligned} \tag{2.4}$$

where λ is the distribution on $\{0,1\}^{2m}$ defined as $\lambda(x,y) = \frac{1}{2^m}\mu(x \oplus y)$ using a distribution μ on $\{0,1\}^m$, and $0 \leq \nu \leq 1$ is some constant.
Using the distribution μ we define a measure on ordered polynomials. We first define W for an ordered monomial κ and then we extend it linearly to all ordered polynomials:

$$W(\kappa) = \mu(\vec{\oplus}(\kappa)) \cdot \frac{2^m}{m^n}.$$

Analogous to the boolean setting we define a concept of corruption for product polynomials. Formally we define

$$\mathrm{MCorr}^z_{W,\gamma}(f) \coloneqq \min_{\substack{\alpha\cdot\beta \,:\, \text{balanced} \\ \|\alpha\|_\infty, \|\beta\|_\infty \leq 1 \\ W(\alpha\cdot\beta \cap \mathcal{K}(f^{-1}(z))) \leq \gamma W(\alpha\cdot\beta)}} \log\Big(\frac{1}{W(\alpha\cdot\beta)}\Big).$$

where $\alpha \cdot \beta \cap \mathcal{K}(f^{-1}(z))$ is the polynomial obtained from $\alpha \cdot \beta$ by retaining the monomials in $\mathcal{K}(f^{-1}(z))$ along with their coefficients. Here for a polynomial α we denote $\|\alpha\|_\infty$ as the maximum absolute value of any coefficient of α. Now we shall prove that the corruption of product polynomials is high.

Lemma 2.4.1 (Corruption Transfer)**.**

$$MCorr^z_{W,\frac{\nu}{3}}(f) \geq \min\{Corr^z_{\lambda,\nu}(f \circ \mathrm{XOR}), m\} - \log_2 48.$$

In other words if a balanced product polynomial $H = \alpha \cdot \beta$ with $\|\alpha\|_\infty, \|\beta\|_\infty \leq 1$ satisfies

$$W(H \cap \mathcal{K}(f^{-1}(z))) \leq \frac{\nu}{3} W(H) \tag{2.5}$$

then we have

$$W(H) \leq 48 \cdot 2^{-\min\{Corr^z_{\lambda,\nu}(f\circ\mathrm{XOR}), m\}}.$$

Proof.
Let $H = \alpha \cdot \beta$ be a product polynomial whose coefficients are at most 1. We define $\widetilde{\alpha}, \widetilde{\beta} \in \mathbb{R}^{2^m}$ as well by assigning for each $u \in \{0,1\}^m$

$$\widetilde{\alpha}[u] = \sum_{\kappa \;|\; \vec{\oplus}(\kappa) = u} \alpha[\kappa]$$

and

$$\widetilde{\beta}[u] = \sum_{\kappa \;|\; \vec{\oplus}(\kappa) = u} \beta[\kappa].$$

By definition of product polynomials we have

$$\begin{aligned}
\alpha \cdot \beta &= \left(\sum_{u \in \{0,1\}^m} \sum_{\kappa \;|\; \vec{\oplus}(\kappa) = u} \alpha[\kappa] \cdot \kappa \right) \cdot \left(\sum_{v \in \{0,1\}^m} \sum_{\kappa' \;|\; \vec{\oplus}(\kappa') = v} \beta[\kappa'] \cdot \kappa' \right) \\
&= \sum_{x \in \{0,1\}^m} \left(\sum_{u \in \{0,1\}^m} \left(\sum_{\kappa \,:\, \vec{\oplus}(\kappa) = u} \alpha[\kappa] \cdot \kappa \right) \cdot \left(\sum_{\kappa' \,:\, \vec{\oplus}(\kappa') = u \oplus x} \beta[\kappa'] \cdot \kappa' \right) \right).
\end{aligned}$$

Now applying W on both sides using linearity we have

$$W(\alpha \cdot \beta) = \sum_{x \in \{0,1\}^m} W(x) \cdot \left(\sum_{u \in \{0,1\}^m} \left(\sum_{\kappa \,:\, \vec{\oplus}(\kappa) = u} \alpha[\kappa] \right) \cdot \left(\sum_{\kappa' \,:\, \vec{\oplus}(\kappa') = u \oplus x} \beta[\kappa'] \right) \right)$$
$$= \sum_{x \in \{0,1\}^m} W(x) \cdot \left(\sum_{u \in \{0,1\}^m} \widetilde{\alpha}[u] \cdot \widetilde{\beta}[u \oplus x] \right),$$

where by abuse of notation we denote $W(x) = W(\kappa)$ where κ is some monomial with $\vec{\oplus}(\kappa) = x$. For ease of writing we denote,

$$\widetilde{W}(\widetilde{\alpha}, \widetilde{\beta}) \coloneqq \sum_{x \in \{0,1\}^m} W(x) \cdot \left(\sum_{u \in \{0,1\}^m} \widetilde{\alpha}[u] \cdot \widetilde{\beta}[u \oplus x] \right) = W(\alpha \cdot \beta).$$

Similarly we have,

$$\widetilde{W}_z(\widetilde{\alpha}, \widetilde{\beta}) \coloneqq \sum_{\substack{x \in \{0,1\}^m \\ f(x) = z}} W(x) \cdot \left(\sum_{u \in \{0,1\}^m} \widetilde{\alpha}[u] \cdot \widetilde{\beta}[u \oplus x] \right) = W(\alpha \cdot \beta \cap \mathcal{K}(f^{-1}(z))).$$

Now we construct an optimization problem with $\gamma = \frac{\nu}{3}$

Program A

Variables:	$\widetilde{\alpha}[v], \widetilde{\beta}[v] \;:\; v \in \{0,1\}^m$
Objective:	$\max \widetilde{W}(\widetilde{\alpha}, \widetilde{\beta})$
Constraints:	$0 \le \widetilde{\alpha}[v] \le \lvert\mathcal{F}^v_{I(\alpha),m}\rvert$
	$0 \le \widetilde{\beta}[v] \le \lvert\mathcal{F}^v_{I(\beta),m}\rvert$
	$\widetilde{W}_z(\widetilde{\alpha}, \widetilde{\beta}) \le \gamma \cdot \widetilde{W}(\widetilde{\alpha}, \widetilde{\beta})$

Let OPT1 be the optimum value of the optimization program A. Clearly, the Lemma will be proved by establishing the required upper bound on OPT1. This is the aim of the remaining part of the argument. First, we define a relaxation of the above optimization problem.

Program B

Variables:	$\widetilde{\alpha}[v], \widetilde{\beta}[v] \;:\; v \in \{0,1\}^m$
Objective:	$\max \widetilde{W}(\widetilde{\alpha}, \widetilde{\beta})$
Constraints:	$0 \le \widetilde{\alpha}[v] \le \dfrac{4m^{\lvert I(\alpha)\rvert}}{2^m}$
	$0 \le \widetilde{\beta}[v] \le \dfrac{4m^{\lvert I(\beta)\rvert}}{2^m}$
	$\widetilde{W}_z(\widetilde{\alpha}, \widetilde{\beta}) \le \gamma \cdot \widetilde{W}(\widetilde{\alpha}, \widetilde{\beta})$

Let OPT2 be the optimum value of the optimization program B. Since we have $m \ln m \leq \frac{n}{4}$, Corollary 2.3.1 tell us that the second optimization problem is indeed a relaxation of the first[2], and hence OPT1 $\leq$ OPT2. The goal in the next few steps is to extract a combinatorial rectangle R from $\widetilde{\alpha}$ and $\widetilde{\beta}$ such that OPT2 is upper bounded by $O(\lambda(R))$. Then, applying the corruption bound on R we will get our desired bound on OPT2. To do this, it will be convenient to understand a simple structure of an optimal solution to Program B.

Let $(\widehat{\alpha}, \widehat{\beta})$ be an optimum solution to the optimization Program B. We obtain a linear program on the variables $\tilde{\alpha}$ from Program B by fixing the values $\widetilde{\beta} = \widehat{\beta}$. The constraints of this LP define a polytope and let θ be a corner point. Then, θ has the property that for at most one coordinate $\widetilde{u}$, that we call exceptional, we have $\theta[\widetilde{u}] \notin \{0, \frac{4m^{|I(\widetilde{\alpha})|}}{2^m}\}$. For every other coordinate $u \neq \widetilde{u}$ we have $\theta[u] \in \{0, \frac{4m^{|I(\alpha)|}}{2^m}\}$. Hence, there exists an optimum solution at one of the corner points, denoted by α^*. Clearly, $W(\alpha^*, \widehat{\beta}) = W(\widehat{\alpha}, \widehat{\beta})$. Again fixing $\widetilde{\alpha} = \alpha^*$ in Program B we get a linear program on the variables $\tilde{\beta}$. We can get an optimum solution β^* at a corner point of the polytope defined by the constraints on the β variables. This gives us a corner point solution (α^*, β^*) (with exceptional coordinates u^*, v^*) which achieves the optimum.

Now for such a product polynomial we define a rectangle $R = A \times B$ where

$$A = \{u \mid \alpha^*[u] \neq 0\} \setminus \{u^*\}.$$

$$B = \{v \mid \beta^*[v] \neq 0\} \setminus \{v^*\}.$$

From the definition we have

$$\widetilde{W}(\alpha^*, \beta^*) = \sum_{x \in \{0,1\}^m} W(x) \cdot \left(\sum_{u \in \{0,1\}^m} \alpha^*[u] \cdot \beta^*[u \oplus x] \right).$$

Now interchanging the order of summation we have,

$$\widetilde{W}(\alpha^*, \beta^*) = \sum_{u \in A \cup \{u^*\}} \alpha^*[u] \sum_{\substack{x \in \{0,1\}^m \\ u \oplus x \in B \cup \{v^*\}}} \beta^*[u \oplus x] \cdot W(x).$$

Renaming $u \oplus x$ as v,

$$\begin{aligned} \widetilde{W}(\alpha^*, \beta^*) \leq & \sum_{u \in A, v \in B} \alpha^*[u] \cdot \beta^*[v] \cdot W(u \oplus v) + \sum_{v \in \{0,1\}^m} \alpha^*[u^*] \cdot \beta^*[v] \cdot W(u^* \oplus v) \\ & + \sum_{u \in \{0,1\}^m} \alpha^*[u] \cdot \beta^*[v^*] \cdot W(u \oplus v^*). \end{aligned}$$

For ease of notation we define $\widehat{W}(\alpha^*, \beta^*) = \sum_{u \in A, v \in B} \alpha^*[u] \cdot \beta^*[v] \cdot W(u \oplus v)$ and $W'(\alpha^*, \beta^*) = \widetilde{W}(\alpha^*, \beta^*) - \widehat{W}(\alpha^*, \beta^*)$.

We establish an upper bound on $\widetilde{W}(\alpha^*, \beta^*)$ in terms of $\lambda(R)$ via parts 1 and 2 of the following claims which gives upper bounds on $\widehat{W}(\alpha^*, \beta^*)$ and $W'(\alpha^*, \beta^*)$.

[2]This is the only place that we make use of the fact that H is a balanced product polynomial, i.e. $\frac{n}{3} \leq |I(\alpha)|, |I(\beta)| \leq \frac{2n}{3}$.

Claim 2.4.1.

1. $\widehat{W}(\alpha^*, \beta^*) = 16\lambda(R)$.
2. $W'(\alpha^*, \beta^*) \leq \frac{32}{2^m}$.
3. $\widetilde{W}_z(\alpha^*, \beta^*) \geq 16\lambda(R \cap F^{-1}(z))$.

First we complete the proof of Lemma 2.4.1 assuming the above claim. Combining parts 1 and 2 of Claim 2.4.1 we conclude that

$$\widetilde{W}(\alpha^*, \beta^*) \leq 16\lambda(R) + 32 \cdot 2^{-m}.$$

If $\lambda(R) \leq 2^{-m}$ then $\widetilde{W}(\alpha^*, \beta^*) \leq 48 \cdot 2^{-m}$ and then we are done. So we may assume $\lambda(R) > 2^{-m}$ and hence we obtain the upper bound

$$\widetilde{W}(\alpha^*, \beta^*) \leq 16\lambda(R) + 32 \cdot 2^{-m} \leq 16\lambda(R) + 32\lambda(R) = 48\lambda(R). \tag{2.6}$$

Finally we have,

$$16\lambda(R \cap F^{-1}(z)) \underset{\text{Claim 2.4.1Part 3}}{\leq} \widetilde{W}_z(\alpha^* \cdot \beta^*) \underset{\text{constraint}}{\leq} \gamma \cdot \widetilde{W}(\alpha^* \cdot \beta^*) \underset{\text{Equation 2.6}}{\leq} 48\gamma \cdot \lambda(R).$$

Since $\gamma \leq \frac{\nu}{3}$, we note that the rectangle R satisfies

$$\lambda(R \cap F^{-1}(z)) \leq \nu\lambda(R).$$

Therefore we have,

$$\text{OPT1} \leq \text{OPT2} \underset{\text{Equation 2.6}}{\leq} 48\lambda(R) \underset{\text{Equation 2.4}}{\leq} 48 \cdot 2^{-\min\{\text{Corr}^z_{\lambda,\nu}(F), m\}}.$$

This yields the bound for all balanced product polynomials satisfying Equation 2.5 and completes the proof of Lemma 2.4.1.
All that remains is to prove Claim 2.4.1 which we do next.
Proof. [Proof of Claim 2.4.1] For the first part we have,

$$\begin{aligned}
\widehat{W}(\alpha^*, \beta^*) &= \sum_{u \in A, v \in B} \alpha^*[u] \cdot \beta^*[v] \cdot W(u \oplus v) \\
&= \sum_{u \in A, v \in B} \frac{4m^{|I(\alpha)|}}{2^m} \cdot \frac{4m^{|I(\beta)|}}{2^m} \cdot W(u \oplus v) \\
&= \sum_{u \in A, v \in B} \frac{16m^n}{2^{2m}} \cdot \mu(u \oplus v) \cdot \frac{2^m}{m^n} \\
&= \sum_{u \in A, v \in B} \frac{16}{2^m} \cdot \mu(u \oplus v) \\
&= \sum_{u \in A, v \in B} 16\lambda(u, v) \\
&= 16\lambda(R).
\end{aligned} \tag{2.7}$$

For the second part we have,

$$\begin{aligned}
W'(\alpha^*,\beta^*) &\leq \sum_{v\in\{0,1\}^m} \alpha^*[u^*]\cdot W(u^*\oplus v)\beta^*[v]\cdot + \sum_{u\in\{0,1\}^m} \alpha^*[u]\cdot\beta^*[v^*]\cdot W(u\oplus v^*)\\
&\leq \sum_{v\in\{0,1\}^m} \frac{4m^{|I(\alpha)|}}{2^m}\cdot\frac{4m^{|I(\beta)|}}{2^m}\cdot W(u^*\oplus v) + \sum_{u\in\{0,1\}^m} \frac{4m^{|I(\alpha)|}}{2^m}\cdot\frac{4m^{|I(\beta)|}}{2^m}\cdot W(u\oplus v^*)\\
&= \sum_{v\in\{0,1\}^m} 16\frac{1}{2^m}\mu(u^*\oplus v) + \sum_{u\in\{0,1\}^m} 16\frac{1}{2^m}\mu(u\oplus v^*)\\
&= \frac{32}{2^m}.
\end{aligned}$$

In the last equality we have used the fact that μ is a probability measure on $\{0,1\}^m$. For the third part we have,

$$\begin{aligned}
\widetilde{W_z}(\alpha^*,\beta^*) &\geq \sum_{\substack{u\in A, v\in B\\ F(u\oplus v)=z}} \alpha^*[u]\cdot\beta^*[v]\cdot W(u\oplus v)\\
&= \sum_{\substack{u\in A, v\in B\\ F(u\oplus v)=z}} 16\lambda(u,v)\\
&= 16\lambda(R\cap F^{-1}(z)).
\end{aligned}$$

■

■

We give a simple reformulation of our corruption bound[3] for product polynomials which will be useful later.

Corollary 2.4.1. *For every balanced product polynomial $H=\alpha\cdot\beta$ with $\|\alpha\|_\infty, \|\beta\|_\infty \leq 1$ we have for every $z\in\{0,1\}$*

$$W(\alpha\cdot\beta\cap\mathcal{K}(f^{-1}(z))) \geq \frac{\nu}{3}W(\alpha\cdot\beta) - 48\cdot 2^{-\min\{Corr^z_{\lambda,\nu}(f\circ XOR),m\}}. \tag{2.8}$$

Proof. From Lemma 2.4.1 we know that if the product polynomial satisfies

$$W(\alpha\cdot\beta\cap\mathcal{K}(f^{-1}(z)) \leq \frac{\nu}{3}W(\alpha\cdot\beta).$$

then $W(\alpha\cdot\beta)\leq 48\cdot 2^{-\min\{\mathrm{Corr}^z_{\lambda,\nu}(F),m\}}$, where $F=f\circ\mathrm{XOR}$ and thus the right hand side of Equation 2.8 would be negative and hence (2.8) would be true. Otherwise the product polynomial satisfies

$$W(\alpha\cdot\beta\cap\mathcal{K}(f^{-1}(z))) > \frac{\nu}{3}W(\alpha\cdot\beta).$$

and hence (2.8) is again true. ■

[3]This form of the corruption bound appears in Razborov's [82] argument for Set-Disjointness.

2.5 Exponential Separation Between Depth-3 Formulas and Monotone VP

In this section, we prove Theorem 2.1.1. We show the construction of a polynomial-size depth three circuit for a δ-non-sink polynomial which embeds an approximation of the boolean function SINK $\circ$ XOR. Next, we use the corruption transfer result from Section 2.4 to show that δ-non-sink polynomials are hard for monotone circuits.

2.5.1 The construction of depth-3 formula.

For the sake of the reader, we recall some concepts from the introduction. For a function $\sigma \in \mathcal{F}_{n,m}$ when $m = \binom{k}{2}$, the parity vector $\vec{\oplus}(\sigma)$ can be interpreted as a tournament by orienting the edges of K_k according to $\vec{\oplus}(\sigma)$. More precisely, we fix a bijection $\phi : [\binom{k}{2}] \mapsto E(K_k)$ and interpret $\vec{\oplus}(\sigma)_j$ giving an orientation to the edge $\phi(j) = (x, u)$ where $x < u$. If $\vec{\oplus}(\sigma)_j = 1$ then we give the orientation $x \to u$ otherwise we give the orientation $u \to x$. Let $T(\vec{\oplus}(\sigma))$ be the tournament on k vertices obtained using the above process. A function is said to have a sink if the tournament $T(\vec{\oplus}(\sigma))$ has a sink. If $T(\vec{\oplus}(\sigma))$ has a sink u then we have

$$\vec{\oplus}(\sigma) = (*, *, *, \underbrace{1, 1, \ldots, 1}_{\substack{\phi(j)=(x,u) \\ \text{with } x<u}}, \underbrace{0, 0, \ldots, 0}_{\substack{\phi(j)=(u,x) \\ \text{with } x>u}}, *, *, *)$$

where the coordinates marked by $*$ could be either 0 or 1. For a vertex u, we define $\text{sink}(u) \coloneqq \{\sigma \mid \vec{\oplus}(\sigma) \text{ has a sink at } u\}$.

Proof. [Proof of Theorem 2.1.1 Part (1):] Let $X_i = \{x_{i,j}\}_{j=1}^{m}$ be a set of variables and let $X = \cup_{i=1}^{n} X_i$. For a vertex $u \in K_k$, consider the vector

$$\boldsymbol{u} = (*, *, *, \underbrace{1, 1, \ldots, 1}_{\substack{\phi(j)=(x,u) \\ \text{with } x<u}}, \underbrace{0, 0, \ldots, 0}_{\substack{\phi(j)=(u,x) \\ \text{with } x>u}}, *, *, *).$$

For a vertex $u \in K_k$ and a vector $b^{(r)} \in \{+1, -1\}^m$, we define a polynomial

$$Q_{u,r} \coloneqq \prod_{j|\vec{u}_j=1} b_j^{(r)} \cdot \prod_{i=1}^{n} \Big(\sum_{j|\vec{u}_j \neq *} b_j^{(r)} x_{i,j} + \sum_{j|\vec{u}_j=*} x_{i,j} \Big).$$

For $0 < \epsilon < 1$, let $\mathcal{B}_{m,\frac{\epsilon}{k}}$ be the ϵ/k-biased space obtained from Theorem 2.2.2 of size $N = O(\frac{m}{(\epsilon/k)^{2+o(1)}})$.

Using the $\frac{\epsilon}{k}$-biased space $\mathcal{B}_{m,\frac{\epsilon}{k}} = \{b^{(1)}, b^{(2)}, \ldots, b^{(N)}\}$, we define the following polynomial Q_u

$$Q_u \coloneqq \frac{1}{N} \cdot \sum_{r=1}^{N} Q_{u,r}.$$

Any monomial in $Q_{u,r}$ has the form $\kappa = \prod_{i=1}^{n} x_{i,\sigma(i)}$ for some function $\sigma : [n] \mapsto [m]$ and further,

$$Q_{u,r}[\kappa] = \prod_{j|\boldsymbol{u}_j=0} (b_j^{(r)})^{|\sigma^{-1}(j)|} \cdot \prod_{j|\boldsymbol{u}_j=1} (b_j^{(r)})^{|\sigma^{-1}(j)|+1}.$$

Let us note that when $\vec{\oplus}(\kappa) \in \text{sink}(u)$, we have $Q_{u,r}[\kappa] = 1$ and for any other monomial its coefficient depends on at least one coordinate of $b^{(r)}$. Since $b^{(1)}, b^{(2)}, \ldots, b^{(N)}$ form an $\frac{\epsilon}{k}$-bias space, we have $|Q_u[\kappa]| < \frac{\epsilon}{k}$ for any κ with $\vec{\oplus}(\kappa) \notin \text{sink}(u)$. Now we define the polynomial

$$Q_{\epsilon,n,m} \coloneqq \sum_{u\in[k]} Q_u.$$

Let us note that $Q_{\epsilon,n,m}[\kappa] \in (1-\epsilon, 1+\epsilon)$ if $\vec{\oplus}(\kappa)$ has a sink and $|Q_{\epsilon,n,m}[\kappa]| < \epsilon$ otherwise. We define another polynomial

$$H_{\epsilon,n,m} \coloneqq \frac{1}{1+2\epsilon}\left[(1+\epsilon)Q_{\text{All}} - Q_{\epsilon,n,m}\right],$$

where $Q_{\text{All}} = \prod_{i=1}^{n}(\sum_{j=1}^{m} x_{i,j})$. We observe that $H_{\epsilon,n,m}[\kappa] \in (0, \frac{2\epsilon}{1+2\epsilon})$ if $\vec{\oplus}(\kappa)$ has a sink and $H_{\epsilon,n,m}[\kappa] \in (\frac{1}{1+2\epsilon}, 1)$ otherwise.
By definition, the polynomial $H_{\epsilon,n,m}$ has a depth-three formula of size $O(kNnm)$ which is $O(nm^{3.5+o(1)}/\epsilon^{2+o(1)})$. Finally, we define the following polynomial

$$P_{\delta,n,m} \coloneqq H_{\frac{\delta}{2(1-\delta)},n,m}.$$

Notice that $P_{\delta,n,m}[\kappa] \in (0,\delta)$ if $\vec{\oplus}(\kappa)$ has a sink, and $P_{\delta,n,m}[\kappa] \in (1-\delta, 1)$ otherwise. Note that $P_{\delta,n,m}$ is a δ-non-sink Polynomial. ∎

2.5.2 The Lower Bound.

In this section, we establish the following theorem which is a restatement of part 2 of Theorem 2.1.1.

Theorem 2.5.1 (Restatement of Theorem 2.1.1, part 2)**.** *There exists a sufficiently small constant $\delta < 1$ such that for every δ-non-sink polynomial $Q_{\delta,n,m}$, the monotone circuit complexity of $Q_{\delta,n,m}$ is $2^{\Omega(\sqrt{m})}$, when $4m \ln m \le n$.*

Since in the previous section, we have given an example of a δ-non-sink polynomial that can be computed by depth-3 general formula of small size, we get our required separation as claimed by Theorem 2.1.1.
In order to prove our theorem, we will make use of a rectangular corruption bound established by Chattopadhyay, Mande and Sherif [27] for the Boolean function SINK∘XOR. More precisely, consider the SINK function associated with the complete graph K_k, with $m = \binom{k}{2}$. Define the following distribution μ on the space of inputs $\{0,1\}^m$ to SINK: toss a fair coin b. If $b = 1$, sample a vertex $i \in [k]$ at random, and then sample at random $x \in \{0,1\}^m$ from all inputs that make i a sink. If $b = 0$, sample x at random from $\{0,1\}^m$. Let $\lambda : \{0,1\}^m \times \{0,1\}^m \to [0,1]$ be the probability distribution given by $\lambda(x,y) = \frac{1}{2^m} \cdot \mu(x \oplus y)$. We collect some simple facts together:

Fact 2.5.1.

1. $\forall z \in SINK^{-1}(0) : \mu(z) = \frac{1}{2^{m+1}}$.

2. $\left|SINK^{-1}(1)\right| = k2^{m-k+1} = O(\sqrt{m}2^{m-\sqrt{m}})$.

Proof. The first part follows by noting that the distribution μ puts non-zero mass on $z \in \text{SINK}^{-1}(0)$ only when $b = 0$ (which happens with probability $1/2$), thereafter μ picks $z \in \{0,1\}^m$.
For the second part we note that $\left|\text{SINK}^{-1}(1)\right|$ is a disjoint union of k many affine subspaces. For $i \in [k]$ the ith affine subspace corresponds to the bit-vectors which have vertex i as a SINK. Since the SINK vertex is unique (if it exists), we conclude that the affine subspaces are disjoint. Finally e observe that each affine subspace is of dimension $n - k + 1$, hence completing the proof. ∎

Now we state the main corruption bound that we use is as follows:

Theorem 2.5.2 (Lemma 6.2 in[4] [27])**.** *Let $0 \leq \nu \leq 1/2$ be any constant. Then,*

$$Corr^1_{\lambda,\nu}\Big(SINK \circ XOR\Big) = \Omega(\sqrt{m}).$$

Now we prove Theorem 2.5.1 using the corruption bound from [27].
Proof. Let Q by any δ-non-sink polynomial. Let C be a monotone circuit of size s computing Q. Then by the structure Theorem 2.2.1, we may write Q as a sum of s many balanced product polynomials

$$Q = \sum_{t=1}^{s} \alpha_t \cdot \beta_t.$$

The general idea of our argument is as follows: given the hard distribution μ on SINK considered by [27], we define a measure W on monomials as prescribed by the Transfer Lemma 2.4.1 established in the previous section. Combining Theorem 2.5.2 with the Corruption Transfer Lemma, we immediately obtain that every product polynomial $\alpha_t \cdot \beta_t$ either measures very little w.r.t W or its contribution to the sink monomials, measured w.r.t W, is a significant fraction of the total measure $W(\alpha_t \cdot \beta_t)$. However, we show that the sink monomials of Q, weighted by their coefficients, measure up to a tiny fraction of the total measure $W(Q)$. These two opposing facts can be reconciled only if the number of product polynomials, s, in the decomposition of Q is exponentially large[5].
Forthwith the details: set measure W on monomials according to our prescription, i.e. for any monomial κ, $W(\kappa) \coloneqq \mu(\vec{\oplus}(\kappa)) \cdot \frac{2^m}{m^n}$, where μ is the hard probability distribution on the inputs of SINK.

[4]Please note that [27] use the symbol m to represent the number of vertices in the complete graph whose edges represent the variables to SINK. Their m corresponds to our k. Further, their ν corresponds to our λ, and their value of $4 \cdot \epsilon$ corresponds to our ν.

[5]While this is the well-known idea behind the formulation of the notion of corruption in communication complexity, we are not aware of its prior use in arithmetic complexity.

Transferring the rectangular corruption bound of Theorem 2.5.2 to W via Lemma 2.4.1 and then using the bound of Corollary 2.4.1, we observe that

$$\begin{aligned} W(Q \cap \mathcal{K}(\mathrm{SINK}^{-1}(1))) &= \sum_{t=1}^{s} W(\alpha_t \cdot \beta_t \cap \mathcal{K}(\mathrm{SINK}^{-1}(1))) \\ &\geq \sum_{t=1}^{s} \frac{\nu}{3} W(\alpha_t \cdot \beta_t) - 48 \cdot s \cdot 2^{-\mathrm{Corr}^1_{\lambda,\nu}(\mathrm{SINK} \circ \mathrm{XOR})}. \end{aligned}$$

Hence,

$$48 \cdot s \cdot 2^{-\mathrm{Corr}^1_{\lambda,\nu}(\mathrm{SINK} \circ \mathrm{XOR})} \geq \frac{\nu}{3} W(Q) - W\Big(Q \cap \mathcal{K}(\mathrm{SINK}^{-1}(1))\Big). \tag{2.9}$$

Now we would like to prove a lower bound on the right hand side. In order to do so we first prove two claims.

Claim 2.5.1.

$$W\Big(Q \cap \mathcal{K}(SINK^{-1}(1))\Big) \leq 4\delta.$$

Proof.

$$\begin{aligned} W\Big(Q \cap \mathcal{K}(\mathrm{SINK}^{-1}(1))\Big) &\leq \delta \cdot \sum_{\substack{\kappa \\ \mathrm{SINK}(\vec{\oplus}(\kappa))=1}} W(\kappa) \\ &\leq \delta \cdot \sum_{\substack{\kappa \\ \mathrm{SINK}(\vec{\oplus}(\kappa))=1}} \mu(\vec{\oplus}(\kappa)) \cdot \frac{2^m}{m^n} \\ &= \delta \cdot \sum_{\substack{x \in \{0,1\}^m \\ \mathrm{SINK}(x)=1}} \sum_{\substack{\kappa \\ \vec{\oplus}(\kappa)=x}} \mu(\vec{\oplus}(\kappa)) \cdot \frac{2^m}{m^n} \\ &\leq \delta \cdot \sum_{\substack{x \in \{0,1\}^m \\ \mathrm{SINK}(x)=1}} \frac{4m^n}{2^m} \cdot \mu(x) \cdot \frac{2^m}{m^n} \\ &\leq 4\delta \cdot \sum_{\substack{x \in \{0,1\}^m \\ \mathrm{SINK}(x)=1}} \mu(x) \\ &\leq 4\delta. \end{aligned}$$

where the last inequality follows because μ is a probability measure on $\{0,1\}^m$. ∎

Claim 2.5.2.

$$W(Q) \geq \frac{1-\delta}{3}$$

for large m.

Proof. Using the fact that the coefficient of every non-sink monomial in Q is in the interval $[1-\delta, 1]$, and substituting f for SINK we get,

$$
\begin{aligned}
W(Q) &\geq (1-\delta) W\Big(Q \cap \mathcal{K}(\mathrm{SINK}^{-1}(0))\Big) \\
&= (1-\delta) \cdot \sum_{\substack{\kappa \\ f(\vec{\oplus}(\kappa))=0}} W(\kappa) \\
&= (1-\delta) \cdot \sum_{\substack{x \in \{0,1\}^m \\ f(x)=0}} \sum_{\substack{\kappa \\ \vec{\oplus}(\kappa)=x}} \mu(\vec{\oplus}(\kappa)) \cdot \frac{2^m}{m^n} \\
&= (1-\delta) \cdot \sum_{\substack{x \in \{0,1\}^m \\ f(x)=0}} \sum_{\substack{\kappa \\ \vec{\oplus}(\kappa)=x}} \frac{1}{2^{m+1}} \cdot \frac{2^m}{m^n} \\
&= (1-\delta) \cdot \frac{1}{2} \frac{|\mathcal{K}(f^{-1}(0))|}{m^n} \\
&= (1-\delta) \cdot \frac{1}{2} \left(1 - \frac{|\mathcal{K}(f^{-1}(1))|}{m^n}\right).
\end{aligned}
$$

Now we may bound the total number of sink monomials, using Fact 2.5.1 and Corollary 2.3.1, as

$$
|\mathcal{K}(f^{-1}(1))| \leq c \cdot \sqrt{m} 2^{m-\sqrt{m}} \cdot \frac{4m^n}{2^m} \leq c\sqrt{m} \cdot \frac{4m^n}{2^{\sqrt{m}}}.
$$

for some constant c. Thus we have

$$
\left(1 - \frac{|\mathcal{K}(f^{-1}(1))|}{m^n}\right) \geq (1 - \frac{4c\sqrt{m}}{2^{\sqrt{m}}}) \geq \frac{2}{3},
$$

for large m. Finally we conclude that

$$
W(Q) \geq (1-\delta) \cdot \frac{1}{2} \left(1 - \frac{|\mathcal{K}(f^{-1}(1))|}{m^n}\right) \geq \frac{1-\delta}{3}.
$$

for large m. ∎

Continuing from Equation 2.9 and applying Claims 2.5.1, 2.5.2 we have

$$
48 \cdot s \cdot 2^{-\mathrm{Corr}^1_{\lambda,\nu}\left(\mathrm{SINK} \circ \mathrm{XOR}\right)} \geq \frac{\nu}{3} W(Q) - W\Big(Q \cap \mathcal{K}(\mathrm{SINK}^{-1}(1))\Big) \geq \frac{\nu}{3} \frac{1-\delta}{3} - 4\delta.
$$

Now since δ is small enough we may write $\frac{\nu}{3}\frac{1-\delta}{3} - 4\delta \geq \delta$ and hence

$$
s \geq \frac{\delta}{48} \cdot 2^{\mathrm{Corr}^1_{\lambda,\nu}\left(\mathrm{SINK} \circ \mathrm{XOR}\right)} \geq 2^{\Omega(\sqrt{m})}.
$$

∎

2.6 ϵ-Sensitive Monotone Lower Bound for MOD3 $\circ$ MOD2 Polynomial

In this section, we prove Theorem 2.1.2. Consider the boolean function defined on $\{0,1\}^m$ as $f(x) = 0$ if $\mathrm{wt}(x) \equiv 0 \pmod 3$ and $f(x) = 1$ otherwise. This is often called the MOD_3 Boolean function. We define a polynomial $P_{n,m}$ that remarkably remains hard even when it is added to the full set-multilinear polynomial after being multiplied by a tiny number ϵ. The question of proving such lower bounds against monotone circuits was raised in the recent work of Hrubeš [48] where he gives an alternative approach for attacking general (non-monotone) circuits. More precisely, Hrubeš shows that if one could prove strong lower bounds for arbitrary small, but non-zero ϵ, then they imply comparable bounds for general set-multilinear circuits[6]. Our lower bound works as long as $\epsilon \geq 2^{-\gamma n/\log n}$ for some constant γ.

Candidate polynomial.

First, define the following polynomial.

$$P_{n,m}^{\mathrm{MOD}_3} := \sum_{\substack{\sigma:[n]\mapsto[m] \\ \mathrm{MOD}_3(\vec{\oplus}(\sigma))=0}} \prod_{i=1}^{n} x_{i,\sigma(i)}.$$

It is trivial to see that there is a polynomial time algorithm that given a monomial decides whether the coefficient of the monomial is zero or one in $P_{n,m}^{\mathrm{MOD}_3}$. Hence, by Valiant's criterion [107, Proposition 4], the polynomial $P_{n,m}^{\mathrm{MOD}_3}$ is in VNP. We show a monotone circuit lower bound for

$$P = F_{n,m} - \epsilon \cdot P_{n,m}^{\mathrm{MOD}_3}.$$

for some $\epsilon > 0$, where we define the full polynomial as $F_{n,m} = \prod_{i=1}^{n}(\sum_{j=1}^{m} x_{i,j})$. The lower bound proof for

$$P = F_{n,m} + \epsilon \cdot P_{n,m}^{\mathrm{MOD}_3}.$$

is analogous. Since the polynomial $F_{n,m}$ can be computed by a polynomial-size monotone circuit, the lower bound result for $P = F_{n,m} + \epsilon \cdot P_{n,m}^{\mathrm{MOD}_3}$ also shows that the polynomial $P_{n,m}^{\mathrm{MOD}_3}$ needs exponential-size monotone circuit.
Now we are ready to prove the main result under the condition $4m \ln m \leq n$.

Proof of Theorem 2.1.2.
We define two measures W_0 and W_1. For a monomial κ of the form $\prod_{i=1}^{n} x_{i,\sigma(i)}$ we define

$$\begin{aligned} W_0(\kappa) &= \frac{1}{m^n} && \text{if } \mathrm{wt}(\vec{\oplus}(\kappa)) \equiv 0 \pmod 3 \\ &= 0 && \text{otherwise.} \end{aligned}$$

[6]In fact the method of Hrubeš works for general arithmetic circuits. However since we choose the full polynomial to be set-multilinear, improving the monotone lower bound against our candidate polynomial to arbitrary ϵ would imply lower bounds against general set-multilinear arithmetic circuits.

and similarly,

$$\begin{aligned} W_1(\kappa) &= \frac{1}{m^n} \quad \text{if } \mathrm{wt}(\vec{\oplus}(\kappa)) \equiv 1 \pmod 3 \\ &= 0 \qquad \text{otherwise.} \end{aligned}$$

We linearly extend these measures to all polynomials and define our main measure

$$W(P) = W_1(P) - W_0(P).$$

Our main result will follow immediately from the following two claims: the main technical result of this section will show that the measure W is exponentially small on product polynomials.

Lemma 2.6.1 (Measure is small for balanced product polynomials)**.** *Let $a \cdot b$ be a balanced product polynomial whose coefficients are at most 1. Then*

$$|W(a \cdot b)| \leq \frac{64}{3}\left(\sqrt{\frac{3}{4}}\right)^m,$$

when $4m \ln m \leq n$.

However, the following claim shows that $W(P)$ is large.

Claim 2.6.1. *There exists a constant $\gamma_0 < 1$, such that if $\epsilon \geq 2^{-\gamma_0 \cdot m}$, then*

$$W(P) \geq \frac{\epsilon}{5},$$

when $4m \ln m \leq n$.

Given these two results, the structure theorem readily yields the main result as shown by the following short argument. Suppose that P has a monotone circuit of size s. Then by the structure Theorem 2.2.1, we can write

$$P = \sum_{t=1}^{s} a_t \cdot b_t$$

where a_t, b_t are balanced product polynomials.
Thus,

$$\frac{\epsilon}{5} \underset{\text{Claim 2.6.1}}{\leq} W(P) = \sum_{t=1}^{s} W(a_t \cdot b_t) \underset{\text{Lemma 2.6.1}}{\leq} s \cdot \frac{64}{3}\left(\sqrt{\frac{3}{4}}\right)^m.$$

Hence, $s \geq \frac{3\epsilon}{320}\left(\sqrt{\frac{4}{3}}\right)^m$. To get the required lower bound for s, we must have $\epsilon \geq 2^{-\gamma_1 m}$ for some $\gamma_1 > 0$. Claim 2.6.1 needs $\epsilon \geq 2^{-\gamma_0 m}$. Hence, choose $\epsilon \geq 2^{-\min\{\gamma_0, \gamma_1\} m}$. The result now immediately follows.
All that remains is to establish Lemma 2.6.1 and Claim 2.6.1. The latter follows by a short calculation from the former and so we first prove it below.

Proof. [Proof of Claim 2.6.1] Recall the following notation

$$N_b = |\{\kappa \mid \mathrm{wt}(\vec{\oplus}(\kappa)) \equiv b \pmod 3\}| \qquad \forall\, b \in \{0,1,2\}$$

We explicitly compute the measure for our target polynomial using linearity.

$$W(P) = W(F_{n,m}) - \epsilon \cdot W(P_{n,m}).$$

For the full polynomial one can easily compute $W(F_{n,m}) = \frac{N_1}{m^n} - \frac{N_0}{m^n}$ and clearly $W(P_{n,m}) = -W_0(P_{n,m}) = -\frac{N_0}{m^n}$. Thus, we have

$$W(P) = \frac{N_1}{m^n} - \frac{N_0}{m^n} + \epsilon \cdot \frac{N_0}{m^n}.$$

and hence $|W(P)| \geq \epsilon \cdot |\frac{N_0}{m^n}| - |\frac{N_0}{m^n} - \frac{N_1}{m^n}|$. Since $F_{n,m}$ is a product polynomial, using Lemma 2.6.1 we conclude that $W(F_{n,m}) = |\frac{N_1}{m^n} - \frac{N_0}{m^n}| \leq \frac{64}{3} \cdot \left(\sqrt{\frac{3}{4}}\right)^m$.

By a similar calculation one can show that $|\frac{N_0}{m^n} - \frac{N_2}{m^n}|, |\frac{N_1}{m^n} - \frac{N_2}{m^n}| \leq 64 \cdot \left(\sqrt{\frac{3}{4}}\right)^m$ and since $\frac{N_0}{m^n} + \frac{N_1}{m^n} + \frac{N_2}{m^n} = 1$ it must be the case that $|\frac{N_0}{m^n} - \frac{1}{3}| \leq 2^{-\Omega(m)}$ and hence we may write $\frac{N_0}{m^n} \geq \frac{1}{4}$. Finally we have

$$|W(P)| \geq \epsilon \cdot |\frac{N_0}{m^n}| - |\frac{N_0}{m^n} - \frac{N_1}{m^n}| \geq \frac{\epsilon}{5}$$

which is true if the parameter ϵ satisfies the following condition

$$\frac{\epsilon}{20} \geq \frac{64}{3}\left(\sqrt{\frac{3}{4}}\right)^m.$$

Thus we can choose the parameter γ_0 appropriately such that $\epsilon \geq 2^{-\gamma_0 n/\log n}$. ∎

All that remains to finish this section is to argue for the correctness of Lemma 2.6.1. We do so by an argument that is inspired by discrepancy estimation techniques used in communication complexity, especially of the kind that appeared in Ada et.al. [1].

Proof. [Proof of Lemma 2.6.1] Given any product polynomial $a \cdot b$, we define vectors $A, B \in \mathbb{R}^{2^m}$ below, where $v \in \{0,1\}^m$ is an arbitrary parity vector.

$$A[v] \coloneqq \frac{|\{\kappa | \vec{\oplus}(\kappa) = v \text{ and } a[\kappa] \neq 0\}|}{m^{|I(a)|}}.$$

$$B[v] \coloneqq \frac{|\{\kappa | \vec{\oplus}(\kappa) = v \text{ and } b[\kappa] \neq 0\}|}{m^{|I(b)|}}.$$

We also define as

$$\vec{\oplus}(a) \coloneqq \{\vec{\oplus}(\kappa) \mid a[\kappa] \neq 0\},$$
$$\vec{\oplus}(b) \coloneqq \{\vec{\oplus}(\kappa) \mid b[\kappa] \neq 0\}.$$

Further define vectors $\alpha, \beta \in \mathbb{R}^{2^m}$ where $\alpha[v] \coloneqq 2^m \cdot A[v]$ (similarly $\beta[v] \coloneqq 2^m \cdot B[v]$). Assuming that n is even and using Corollary 2.3.1, we conclude that $\alpha[v], \beta[v] \leq 4$.

We write the measures for a product polynomial $a \cdot b$ as

$$W_0(a \cdot b) = \sum_{u,v \in \{0,1\}^m} A[u] \cdot B[v] \cdot \frac{1}{3}(1 + \omega^{|u \oplus v|} + \omega^{2|u \oplus v|}),$$

and

$$W_1(a \cdot b) = \sum_{u,v \in \{0,1\}^m} A[u] \cdot B[v] \cdot \frac{1}{3}(1 + \omega^{|u \oplus v|+2} + \omega^{2|u \oplus v|+1}),$$

where $|u \oplus v| = (\sum_i u_i + \sum_i v_i - 2\sum_i u_i v_i)$, and ω is the complex third root of unity. Note that we have made use of the fact that $(1 + \omega + \omega^2) = 0$.
We write

$$\begin{aligned}
3 \cdot |W(a \cdot b)| = |&\sum_{u,v \in \{0,1\}^m} A[u] \cdot B[v] \cdot \omega^{|u \oplus v|} + \sum_{u,v \in \{0,1\}^m} A[u] \cdot B[v] \cdot \omega^{2|u \oplus v|} \\
&- \sum_{u,v \in \{0,1\}^m} A[u] \cdot B[v] \cdot \omega^{|u \oplus v|+2} - \sum_{u,v \in \{0,1\}^m} A[u] \cdot B[v] \cdot \omega^{2|u \oplus v|+1}| \\
&\leq |\sum_{u,v \in \{0,1\}^m} A[u] \cdot B[v] \cdot \omega^{|u \oplus v|}| + |\sum_{u,v \in \{0,1\}^m} A[u] \cdot B[v] \cdot \omega^{2|u \oplus v|}| \\
&+ |\sum_{u,v \in \{0,1\}^m} A[u] \cdot B[v] \cdot \omega^{|u \oplus v|}| + |\sum_{u,v \in \{0,1\}^m} A[u] \cdot B[v] \cdot \omega^{2|u \oplus v|}|.
\end{aligned}$$

Now we proceed to bound each of the four terms separately.

$$\begin{aligned}
|\sum_{u,v \in \{0,1\}^m} A[u] \cdot B[v] \cdot \omega^{|u \oplus v|}|^2 &= |\sum_{u,v \in \{0,1\}^m} \frac{\alpha[u]}{2^m} \cdot \frac{\beta[v]}{2^m} \cdot \omega^{|u \oplus v|}|^2 \\
&= |\mathop{\mathrm{E}}_{u,v \in \{0,1\}^m} \alpha[u] \cdot \beta[v] \cdot \omega^{|u \oplus v|}|^2.
\end{aligned}$$

Claim 2.6.2.

$$|\mathop{\mathrm{E}}_{u,v \in \{0,1\}^m} \alpha[u] \cdot \beta[v] \cdot \omega^{|u \oplus v|}|^2 \leq 256 \cdot \left(\frac{3}{4}\right)^m.$$

Proof. We apply triangle inequality and then we remove $\alpha[u]$ using the fact that $\alpha[u] \leq 4$. Then we apply the Cauchy-Schwarz inequality,

$$\begin{aligned}
|\mathop{\mathrm{E}}_{u \in \{0,1\}^m} \alpha[u] \cdot \left(\mathop{\mathrm{E}}_{v \in \{0,1\}^m} \beta[v] \cdot \omega^{|u \oplus v|}\right)|^2 &\leq \left(\mathop{\mathrm{E}}_{u \in \{0,1\}^m} \alpha[u] \cdot |\mathop{\mathrm{E}}_{v \in \{0,1\}^m} \beta[v] \cdot \omega^{|u \oplus v|}|\right)^2 \\
&\leq 16 \cdot \left(\mathop{\mathrm{E}}_{u \in \{0,1\}^m} |\mathop{\mathrm{E}}_{v \in \{0,1\}^m} \beta[v] \cdot \omega^{|u \oplus v|}|\right)^2 \\
&\leq 16 \cdot \mathop{\mathrm{E}}_{u \in \{0,1\}^m} |\mathop{\mathrm{E}}_{v \in \{0,1\}^m} \beta[v] \cdot \omega^{|u \oplus v|}|^2.
\end{aligned}$$

Next, we write $|z|^2 = z \cdot \bar{z}$, rearrange terms and then apply triangle inequality,

$$\begin{aligned}
\mathop{\mathrm{E}}_{u \in \{0,1\}^m} |\mathop{\mathrm{E}}_{v \in \{0,1\}^m} \beta[v] \cdot \omega^{|u \oplus v|}|^2 &= \mathop{\mathrm{E}}_{u \in \{0,1\}^m} \left(\mathop{\mathrm{E}}_{v \in \{0,1\}^m} \beta[v] \cdot \omega^{|u \oplus v|}\right) \cdot \left(\mathop{\mathrm{E}}_{\tilde{v} \in \{0,1\}^m} \beta[\tilde{v}] \cdot \omega^{-|u \oplus \tilde{v}|}\right) \\
&= \mathop{\mathrm{E}}_{v \in \{0,1\}^m} \mathop{\mathrm{E}}_{\tilde{v} \in \{0,1\}^m} \beta[v]\beta[\tilde{v}] \cdot \left(\mathop{\mathrm{E}}_{u \in \{0,1\}^m} \omega^{|u \oplus v| - |u \oplus \tilde{v}|}\right) \\
&\leq \mathop{\mathrm{E}}_{v \in \{0,1\}^m} \mathop{\mathrm{E}}_{\tilde{v} \in \{0,1\}^m} \beta[v]\beta[\tilde{v}] \cdot |\mathop{\mathrm{E}}_{u \in \{0,1\}^m} \omega^{|u \oplus v| - |u \oplus \tilde{v}|}|.
\end{aligned}$$

Again we use $\beta[v] \leq 4$,

$$\begin{aligned}
\mathop{\mathrm{E}}_{v\in\{0,1\}^m}\mathop{\mathrm{E}}_{\tilde{v}\in\{0,1\}^m}\beta[v]\beta[\tilde{v}]\cdot\Big|\mathop{\mathrm{E}}_{u\in\{0,1\}^m}\omega^{|u\oplus v|-|u\oplus\tilde{v}|}\Big| &\leq 16\cdot\mathop{\mathrm{E}}_{v\in\{0,1\}^m}\mathop{\mathrm{E}}_{\tilde{v}\in\{0,1\}^m}\Big|\mathop{\mathrm{E}}_{u\in\{0,1\}^m}\omega^{|u\oplus v|-|u\oplus\tilde{v}|}\Big| \\
&= 16\cdot\mathop{\mathrm{E}}_{v,\tilde{v}\in\{0,1\}^m}\Big|\mathop{\mathrm{E}}_{u\in\{0,1\}^m}\omega^{|v|-|\tilde{v}|+2\langle u,\tilde{v}-v\rangle}\Big| \\
&= 16\cdot\mathop{\mathrm{E}}_{v,\tilde{v}\in\{0,1\}^m}\Big|\mathop{\mathrm{E}}_{u\in\{0,1\}^m}\omega^{2\langle u,\tilde{v}-v\rangle}\Big| \\
&= 16\cdot\sum_{k=0}^{m}\sum_{\substack{\hat{v}\in\{-1,0,1\}^m\\ \text{number of zeros in }\hat{v}=k}}\frac{2^k}{2^{2m}}\cdot\prod_i\Big|\mathop{\mathrm{E}}_{u_i\in\{0,1\}}\omega^{2u_i\cdot\hat{v}_i}\Big|.
\end{aligned}$$

A simple calculation shows that $\big|\mathop{\mathrm{E}}_{u_i\in\{0,1\}}\omega^{2u_i\cdot\hat{v}_i}\big| = \frac{1}{2}$, whenever $\hat{v}_i \in \{-1,1\}$. Plugging this back into the previous equation we get

$$\begin{aligned}
\sum_{k=0}^{m}\sum_{\substack{\hat{v}\in\{-1,0,1\}^m\\ \text{number of zeros in }\hat{v}=k}}\frac{2^k}{2^{2m}}\cdot\prod_i\Big|\mathop{\mathrm{E}}_{u_i\in\{0,1\}}\omega^{2u_i\cdot\hat{v}_i}\Big| &= \sum_{k=0}^{m}\binom{m}{k}\cdot 2^{m-k}\cdot\frac{2^k}{2^{2m}}\cdot\frac{1}{2^{m-k}} \\
&= \frac{1}{4^m}\sum_{k=0}^{m}\binom{m}{k}\cdot 2^k \\
&= \left(\frac{3}{4}\right)^m.
\end{aligned}$$

■

One can use a similar analysis to bound the other terms as well. Finally we conclude that $|W(a\cdot b)| \leq \frac{64}{3}\cdot\left(\sqrt{\frac{3}{4}}\right)^m$. ■

2.7 Conclusion

The main interesting feature of this chapter is to provide a framework for constructing polynomials by embedding boolean functions which *transfers* boolean hardness to the arithmetic hardness in the monotone setting. More precisely, we find a way to transfer the corruption measure and demonstrate a separation between the strongest model of monotone computation and the weakest model of non-monotone computation. However our lower bound is not strongly exponential in the number of variables. It would be interesting to find another candidate polynomial for which we can show a strongly exponential separation between monotone circuits and depth-3 circuits. Currently, we know that the family of spanning tree polynomials over constant degree expander graphs show strongly exponential lower bound for monotone circuits [26]. However the best known upper bound for spanning tree polynomial is polynomial-size arithmetic branching program.
The results in section 2.6 shows some initial progress in proving lower bounds in the ϵ-sensitive framework introduced by Hrubeš [48]. Our result can handle ϵ exponentially small in the number of variables. It is implicit in the work of Hrubeš that handling doubly exponentially small ϵ will result in the separation VP $\neq$ VNP. So any progress in this direction will be very interesting.

Chapter 3

Equivalence Testing of Weighted Automata over Partially Commutative Monoids

Motivated by equivalence testing of k-tape automata, we study the *equivalence* testing of weighted automata in the more general setting of partially commutative monoids (in short, pc monoids), and show efficient algorithms in some special cases, exploiting the structure of the underlying non-commutation graph of the monoid.
Specifically, if the edge clique cover number of the non-commutation graph of the pc monoid is a constant, we obtain a deterministic quasi-polynomial time algorithm for equivalence testing. As a corollary, we obtain the first deterministic quasi-polynomial time algorithms for equivalence testing of k-tape weighted automata and for equivalence testing of deterministic k-tape automata for constant k. Prior to this, the best complexity upper bound for these k-tape automata problems were randomized polynomial-time, shown by Worrell [114]. Finding a polynomial-time deterministic algorithm for equivalence testing of deterministic k-tape automata for constant k has been open for several years [40] and our results make progress.
We also consider pc monoids for which the non-commutation graphs have an edge cover consisting of at most k cliques and star graphs for any constant k. We obtain a randomized polynomial-time algorithm for equivalence testing of weighted automata over such monoids. Our results are obtained by designing efficient zero-testing algorithms for weighted automata over such pc monoids.

3.1 Introduction

Testing the equivalence of two multi-tape finite automata is a fundamental problem in automata theory. For a k-tape automaton, we denote the mutually disjoint alphabets for the k tapes by $\Sigma_1, \ldots, \Sigma_k$. The automaton accepts a subset of the product monoid $\Sigma_1^* \times \cdots \times \Sigma_k^*$. Two multi-tape automata are *equivalent* if they accept the same subset. Equivalence testing of multi-tape *non-deterministic* automata is undecidable [44]. The problem was shown to be decidable for 2-tape *deterministic* automata independently by Bird [21] and Valiant [106]. Subsequently, an exponential upper bound was established [18]. Eventually, a polynomial-time algorithm was obtained by Friedman and Greibach [40] and the authors conjectured that equivalence testing of deterministic k-tape automata for any constant k is in polynomial time.

A closely related problem is testing the *multiplicity equivalence* of non-deterministic multi-tape automata. The multiplicity equivalence testing problem is to decide whether for each tuple in the product monoid $\Sigma_1^* \times \cdots \times \Sigma_k^*$, the number of accepting paths in the two input automata is the same. Since a deterministic automaton has at most one accepting path for each input word, the equivalence of deterministic k-tape automata coincides with multiplicity equivalence. More generally, equivalence testing for *weighted automata* (over the underlying field or ring of coefficients) is to decide if the coefficient of each word (i.e. the total sum of weights of each accepting path) is the same for the two given automata. For the weighted case, equivalence testing is in deterministic polynomial time for one-tape automata [92, 105]. Equivalence testing of k-tape weighted automata was shown *decidable* by Harju and Karhumäki [46] using the theory of free groups [1]. An improved complexity-theoretic upper bound remained elusive for k-tape multiplicity equivalence testing, until recently Worrell [114] obtained a *randomized* polynomial-time algorithm for testing the equivalence of k-tape weighted automata (and equivalence testing of deterministic k-tape automata) for any constant k. Worrell takes a different approach via Polynomial Identity Testing (PIT). In [114], Worrell asked if the equivalence testing problem for k-tape weighted automata can be solved in *deterministic* polynomial time, for constant k.

Building on Worrell's results [114] and exploiting further the connections between weighted automata equivalence and polynomial identity testing, we show that equivalence testing of two k-tape weighted automata is in *deterministic* quasi-polynomial time. This immediately yields the first deterministic quasi-polynomial time algorithm for equivalence testing of deterministic k-tape automata.

Our approach solves a more general problem in the setting of *partially commutative monoids*. To motivate this, let us consider k-tape weighted automata in this setting. The *product monoid* $M = \Sigma_1^* \times \cdots \times \Sigma_k^*$ associated with k-tape automata is a *partially commutative monoid* (henceforth, *pc monoid*), in the sense that any two variables $x \in \Sigma_i, y \in \Sigma_j, i \neq j$ commute with each other [2]. Variables in the same tape alphabet Σ_i are mutually non-commuting. We associate a *non-commutation graph* G_M with M to describe the non-commutation relations: (x, y) is an edge if and only if x and y do not commute. If there is no edge (x, y) in G_M, the words xy and yx are equivalent as the variables x and y commute. The words over any pc monoid are defined with respect to the equivalence

[1] This also shows the decidability of equivalence problem for deterministic multi-tape automata.

[2] These are sometimes also called as free partially commutative monoids.

relation induced by the non-commutation graph of the pc monoid. The notion of words and their equivalence over a pc monoid is formally explained in Section 3.3. For the k-tape case, the non-commutation graph G_M is a union of k disjoint cliques: its vertex set is $\Sigma_1 \cup \ldots \cup \Sigma_k$ and G_M is the union of k disjoint cliques, induced by each Σ_i.
More generally, we obtain an equivalence testing algorithm for weighted automata over any pc monoid whose non-commutation graph has a constant-size edge clique cover (*not necessarily* disjoint) with a constant number of isolated vertices. Recall that the edge clique cover of a graph is a collection of subgraphs where each subgraph is a clique and each edge of the graph is contained in at least one of the subgraphs. The size of the edge clique cover is the number of cliques in it.
The isolated vertices can be thought of as a part of the edge clique cover by adding a new vertex (variable) for each isolated vertex and introducing a matching edge between them. Henceforth, we will not worry about the isolated vertices separately and consider them as part of the edge clique cover. We call such monoids as *k-clique monoids* where the edge clique cover size is bounded by k.

Remark 3.1.1. *Since two weighted automata, A and B are equivalent if and only if their difference $C = A - B$ is a weighted automaton equivalent to zero (formally explained in Section 3.2), we can describe the results in terms of zero-testing of a weighted automaton.*[3]

In this chapter, the field $\mathbb{F}$ from which the weights of automata are taken is an infinite field. For computational implementation, we assume that the field arithmetic can be performed efficiently (for example, $\mathbb{F}$ could be the field of rational numbers).

Theorem 3.1.1. *Let A be an input $\mathbb{F}$-weighted automaton of size s over a pc monoid M such that its non-commutation graph G_M has an edge clique cover of size k. Then, the zero-testing of A has a deterministic $(nks)^{O(k^2 \log ns)}$ time algorithm. Here n is the size of the alphabet of M, and the edge clique cover is given as part of the input.*

As an immediate corollary, the above theorem yields a deterministic quasi-polynomial time algorithm for equivalence testing of k-tape weighted automata (also for equivalence testing of deterministic k-tape automata). Notice that, for the k-tape case, the edge clique cover of size k is also part of the input since for each $1 \leq i \leq k$, the i^{th} tape alphabet Σ_i is explicitly given and it induces a clique.

Corollary 3.1.1. *The equivalence testing problem for k-tape weighted automata and deterministic k-tape automata can be solved in deterministic quasi-polynomial time for constant k.*

Next, we consider equivalence testing over more general pc monoids M.
Given a graph $G = (X, E)$, a collection of k graphs $\{G_i = (X_i, E_i)\}_{i=1}^{k}$ such that $X = \cup_{i=1}^{k} X_i$ and $E = \cup_{i=1}^{k} E_i$ is called a *k-covering* of G. It seems natural to investigate whether there are covers other than just edge clique cover for which one can obtain efficient equivalence test.
We say M is a *k-monoid* if its non-commutation graph G_M has a 2-covering $\{G_1, G_2\}$ such that, for some $k' \leq k$, G_1 has an edge clique cover of size at most k' and G_2 has

[3]The difference C of two weighted automata A and B means the weight of each word w in C is the difference between the weights of w in A and B.

a vertex cover of size at most $k - k'$ (hence the edges of G_2 can be covered by $k - k'$ many star graphs). We show that equivalence testing over k-monoids has a randomized polynomial-time algorithm for constant k. This result can be seen as a generalization of Worrell's result [114].

Theorem 3.1.2. *Let A be an input $\mathbb{F}$-weighted automaton of size s over a k-monoid M. Then the zero-testing of A can be decided in randomized $(ns)^{O(k)}$ time. Here n is the size of the alphabet of M.*

Remark 3.1.2. *What is the complexity of equivalence testing for weighted automata over an arbitrary pc monoid? The non-commutation graph G_M of any pc monoid M over the alphabet X trivially has an edge clique cover of size bounded by $\binom{|X|}{2}$. Hence, the above results would only give an exponential-time algorithm. Note that if G_M has an* induced matching [4] *of size more than k then M is not a k-monoid. Call M a* matching monoid *if G_M is a perfect matching. It follows from Lemma 3.3.2, shown in Section 3.3, that equivalence testing over arbitrary pc monoids is deterministic polynomial-time reducible to equivalence testing over matching monoids. Thus, the complexity of zero-testing of $\mathbb{F}$-weighted automata over matching monoids is essentially the most general case.*

Various automata-theoretic problems have been studied in the setting of pc monoids. For example, pc monoids have found applications in modeling the behavior of concurrent systems [68]. Droste and Gastin [37] have studied the relation between recognizability and rationality over pc monoids.

Proof Ideas and Techniques

Our proof is inspired by Worrell's key insight [114] that the k-tape automata equivalence problem can be reduced to a suitable instance of polynomial identity testing problem over partially commuting variables. Worrell's algorithm is randomized. In contrast, since we are considering automata over general pc monoids and we aim to design efficient deterministic algorithms, it requires additional ideas. First, we suitably apply a classical algebraic framework to transfer the zero-testing problem over general pc monoids to pc monoids whose non-commutation graphs are a disjoint union of cliques [30, 35]. This allows us to generalize a zero-testing criteria for weighted automata over standard non-commutative setting [38, Cor. 8.3] to the setting of *general pc monoids*. The generalization says that any nonzero weighted automata of size s over any pc monoid must have a non-zero word within the length $\text{poly}(s, n)$ where n is the size of the alphabet. Furthermore, this allows us to reduce the zero-testing of weighted automata to an instance of polynomial identity testing over pc monoids. Moreover, the polynomials can be computed by small arithmetic branching programs (ABPs) over pc monoids. Over non-commutative variables, ABPs are well-studied in arithmetic circuit complexity [72]. It turns out that we can solve the identity testing problem for ABPs over k-clique monoids in deterministic quasi-polynomial time by suitably adapting a black-box polynomial identity test for non-commutative arithmetic branching programs based on a quasi-polynomial size hitting set construction [39]. Our algorithm recursively builds on this construction, ensuring that the resulting hitting set remains of quasi-polynomial size.

[4] An induced matching is a matching that includes every edge connecting any two vertices in the subset as an induced subgraph.

The proof of Theorem 3.1.2 is along similar lines. First, we obtain a randomized polynomial-time identity testing algorithm over pc monoids whose non-commutation graph has a k-vertex cover for constant k. This algorithm itself uses ideas from automata theory. Then a composition lemma yields an identity testing algorithm over k-monoids.

3.2 Preliminaries

We recall some basic definitions and results, mainly from automata theory and arithmetic circuit complexity, and define some notations used in the subsequent sections.

Notation : Let $\mathbb{F}$ be an infinite field. $\mathrm{M}_t(\mathbb{F})$ denotes the ring of $t \times t$ matrices over $\mathbb{F}$. For matrices A and B of sizes $m \times n$ and $p \times q$ respectively, their tensor (Kronecker) product $A \otimes B$ is defined as the block matrix $(a_{ij}B)_{1 \le i \le m, 1 \le j \le n}$, and the dimension of $A \otimes B$ is $pm \times qn$. Given bases $\{v_i\}$ and $\{w_j\}$ for the vector spaces V and W, the vector space $V \otimes W$ is the tensor product space with a basis $\{v_i \otimes w_j\}$.
For a series (resp. polynomial) S and a word (resp. monomial) w, let $[w]S$ denote the coefficient of w in the series S (resp. polynomial). In this chapter, we consider weighted automata over a field $\mathbb{F}$ and alphabet (or variables) $X = \{x_1, \ldots, x_n\}$.

Arithmetic Circuit Complexity : An *algebraic branching program* (ABP) is a layered directed acyclic graph with one in-degree-0 vertex called *source*, and one out-degree-0 vertex called *sink*. Its vertex set is partitioned into layers $0, 1, \ldots, d$, with directed edges only between adjacent layers (i to $i+1$). The source and the sink are in layers zero and d, respectively. Each edge is labeled by an affine linear form over $\mathbb{F}$ in variables $X = \{x_1, \ldots, x_n\}$. The polynomial computed by the ABP is the sum over all source-to-sink directed paths of the product of linear forms that label the edges of the path. The maximum number of nodes in any layer is called the width of the algebraic branching program. The size of the branching program is taken to be the total number of nodes.
Equivalently, the computation of an arithmetic branching program can be defined via the iterated matrix product $\lambda^T M_1 M_2 \cdots M_d \mu$, where λ, μ are vectors in $\mathbb{F}^w$ and each M_i is a $w \times w$ matrix whose entries are affine linear forms over X. Here w corresponds to the ABP width and $d+1$ corresponds to the number of layers in the ABP.
If X is a set of non-commuting variables then the ABP is a non-commutative arithmetic branching program (e.g., see [72]).
Let $S \subset \mathbb{F}\langle X \rangle$ be a subset of polynomials in the non-commutative polynomial ring $\mathbb{F}\langle X \rangle$. A mapping $\boldsymbol{v} : X \to \mathrm{M}_t(\mathbb{F})$ from variables to $t \times t$ matrices, it defines an *evaluation map* defined for any polynomial $f \in \mathbb{F}\langle X \rangle$ as $\boldsymbol{v}(f) = f(v(x_1), \ldots, v(x_n))$. A collection H of such evaluation maps is a *hitting set* for S, if for every nonzero f in S, there is an evaluation $\boldsymbol{v} \in H$ such that $\boldsymbol{v}(f) \neq 0$.
Let $S_{n,d,s}$ denote the set of non-commutative polynomials in $\mathbb{F}\langle X \rangle$ (where $n = |X|$) that have arithmetic branching programs of size s and layers $0, 1, \ldots, d$. Forbes and Shpilka [39] have given a quasi-polynomial size hitting set $H_{n,d,s}$ for $S_{n,d,s}$ that can be constructed in quasi-polynomial time. Moreover, the matrix tuples in $H_{n,d,s}$ are $d+1$ dimensional.

Theorem 3.2.1. *[39, Theorem I.8] For all $s, d, n \in \mathbb{N}$ if $|\mathbb{F}| \ge \mathrm{poly}(d, n, s)$, then there is a set $H_{n,d,s}$ which is a hitting set for $S_{n,d,s}$. Further $|H_{n,d,s}| \le (sdn)^{O(\log d)}$ and there is a deterministic algorithm to output the set $H_{n,d,s}$ in time $(sdn)^{O(\log d)}$.*

Automata Theory :

We recall some basic facts of the algebraic automata theory. More details can be found in the Berstel-Reutenauer book [20].

Let $\mathbb{F}$ be a field[5] and X be an alphabet [6]. A $\mathbb{F}$-weighted automaton[7] A over X has a finite set of states Q. Let $E : Q \times X \times Q \to \mathbb{F}$ is the weight function which assigns weight to each transition. The number of states, $|Q|$ is the *size* of the automaton. A path is a sequence of edges : $(q_0, x_1, q_1)(q_1, x_2, q_2) \cdots (q_{t-1}, x_t, q_t)$. The weight of the path is the product of the weights of the edges. For each word $w = x_1 x_2 \cdots x_t \in X^*$, the coefficient of w, $[w]S$ is the total contribution of all the paths between a start and accepting state for the word w, which is an element of $\mathbb{F}$. This defines a formal series $S = \sum_{w \in X^*} [w]S \cdot w$ which is an element of the *formal power series ring* $\mathbb{F}\langle\langle X \rangle\rangle$. We say that S is the formal series *recognized* by the (weighted) automaton A.

Multi-tape automata.

Next, we briefly explain weighted multi-tape automata defined in terms of pc monoids. Let M be the pc monoid over variables $X = X_1 \cup \cdots \cup X_k$ defined as follows: the variables in each X_i are non-commuting, but for all $i \neq j$ and any $x \in X_i, y \in X_j$ we have $xy = yx$. As defined already, the transition function E is a mapping $Q \times X \times Q \to \mathbb{F}$. A path is a sequence of edges : $(q_0, x_1, q_1)(q_1, x_2, q_2) \cdots (q_{t-1}, x_t, q_t)$ where each $x_i \in X_j$ for some j. The label of the run is $m = x_1 x_2 \cdots x_t$ in the pc monoid M, and $[m]A$ is the total contribution of all the runs between start and accepting states having the label equivalent to m.

An automaton is *deterministic* if the set of states can be partitioned as $Q = Q^{(1)} \sqcup \ldots \sqcup Q^{(k)}$, where states in $Q^{(i)}$ read input only from the i^{th} tape alphabet X_i, and each state has a single transition for every input variable. Thus, a deterministic automaton has at most one accepting path for each input $m \in M$.

Now we explain how equivalence testing of weighted automata is polynomial-time reducible to zero testing of weighted automaton. Let A and B be $\mathbb{F}$-weighted automata over the alphabet X. The transition matrices N_A and N_B are defined as follows: $N_A[i, j] = \sum_{x \in X} E_A(q_i, x, q_j) \cdot x$. ($N_B$ is defined similarly). Let the series computed by A and B be $\lambda_A^T \cdot \sum_{i \geq 0} N_A^i \cdot \mu_A$ and $\lambda_B^T \cdot \sum_{i \geq 0} N_B^i \cdot \mu_B$, respectively. Here $\lambda_A, \mu_A, \lambda_B, \mu_B$ are column scalar vectors. Define the weighted automaton C with transition matrix N_C and the scalar vectors λ_C, μ_C as follows:

$$\lambda_C = \begin{bmatrix} \lambda_A \\ \lambda_B \end{bmatrix}, \qquad N_C = \begin{bmatrix} N_A & 0 \\ 0 & N_B \end{bmatrix}, \qquad \mu_C = \begin{bmatrix} \mu_A \\ -\mu_B \end{bmatrix}.$$

Following is an easy fact which is also used in [114].

Fact 3.2.1. *A and B are equivalent if and only if C is a zero automaton.*

[5] In general $\mathbb{F}$ can be a semiring, but for our purpose it suffices to consider fields.

[6] To unify the notation with polynomial variables, we use the notation X for alphabets over the usual notation Σ.

[7] Sometimes these are also called nondeterministic weighted automata in the literature.

3.3 A Zero Testing Criteria Over Partially Commutative Monoids

A basic result in algebraic automata theory, says that an $\mathbb{F}$-weighted automaton A of size s represents a nonzero series in $\mathbb{F}\langle\langle X\rangle\rangle$ if and only if there is a word $w \in X^*$ of length at most $s-1$, such that $[w]S$ is nonzero. It has a simple linear algebraic proof [38, Corollary 8.3, Page 145][8]. In this section, we prove a theorem similar in spirit over general pc monoids.

pc monoids and partitioned pc monoids : Let X be a finite alphabet (equivalently, variable set). Formally, a pc monoid M over X is a pair $M = (X^*, I)$ where $I \subseteq X \times X$ be such that $(x_1, x_2) \in I$ if and only if $x_1x_2 = x_2x_1$. I is reflexive and symmetric. Let $\tilde{I}$ be the *congruence* generated from I by transitive closure. The monoid elements are defined as the congruence classes $\tilde{m}$ for $m \in X^*$. In other words, M is a factor monoid of X^* generated by $\tilde{I}$. The *non-commutation graph* $G_M = (X, E)$ of M is a simple undirected graph such that $(x_1, x_2) \in E$ if and only if $(x_1, x_2) \notin I$.
A pc monoid M over alphabet (i.e. variable set) X is a *k-partitioned pc monoid* if its non-commutation graph G_M has a k-covering $\{G_i\}_{i=1}^k$ such that the subgraphs G_i are pairwise vertex disjoint. Given any pc monoid M with a k-covering, we can associate a k-partitioned pc monoid M' with it, such that M is isomorphic to a submonoid of M', as follows.
Suppose $G_M = (X, E)$ has a k-covering $\{G_i\}_{i=1}^k$, where $G_i = (X_i, E_i)$. Let $\hat{X} = \{x_{ti} \mid 1 \leq t \leq n, 1 \leq i \leq k\}$ be kn new variables. Let $G'_i = (X'_i, E'_i)$ be a copy of G_i obtained by replacing the vertex $x_t \in X_i$ by its i^{th} copy x_{ti}, such that (x_{ti}, x_{si}) is an edge in G'_i if and only if (x_t, x_s) is an edge in G_i. Let G' denote the disjoint union graph $G' = G'_1 \sqcup G'_2 \sqcup \cdots \sqcup G'_k$, and M' be the pc monoid whose non-commutation graph is $G' = (X', E')$. Clearly, M' is a k-partitioned pc monoid, defined by M and its given k-covering.
As an $\mathbb{F}$-algebra, we note that $\mathbb{F}\langle M'\rangle$ is isomorphic to the tensor product of the $\mathbb{F}$-algebras $\mathbb{F}\langle M'_1\rangle \otimes \cdots \otimes \mathbb{F}\langle M'_k\rangle$.
The following simple observation, which shows that the pc monoid M is isomorphic to a submonoid of M', is well-known [30, 35, 36].

Lemma 3.3.1. *Let $\psi : \mathbb{F}\langle M\rangle \to \mathbb{F}\langle M'\rangle$ be the map such that $\psi(m) = m_1 \otimes m_2 \otimes \cdots \otimes m_k$ for any monomial m in M and extend by linearity. Here, for $1 \leq i \leq k$, the monomial m_i is obtained from m (by dropping the letters of m not in X_i) and replacing each occurrence $x_t \in X_i$ by the variable x_{ti}, $1 \leq t \leq n$. Then, ψ is an injective homomorphism.*

Proof. It is straightforward to check that ψ is a ring homomorphism. To show the injectivity, it is enough to show that $\psi(m) = \psi(m')$ implies $m = m'$ in M for any words $m, m' \in M$. We prove the claim by induction on the length of words in M. Suppose that for words $m \in M$ of length at most ℓ, if m' is not $\tilde{I}$-equivalent to m then $\psi(m) \neq \psi(m')$. The base case, for $\ell = 0$ clearly holds.
Now, suppose $m = x \cdot m_1 \in X^{\ell+1}$ for $x \in X$ and $\psi(m) = \psi(m')$.

Claim 3.3.1. *For some $m_2 \in M$, $m' = x \cdot m_2$ in M.*

[8] It is folklore that the result is attributed to Schützenberger.

Proof. Assume, to the contrary, that there is no $m_2 \in M$ such that $m' = x \cdot m_2$. Let $J = \{j \in [k] \mid x \in X_j\}$. If the variable x does not occur in m' then $m|_{X_j} \neq m'|_{X_j}$ for each $j \in J$. This implies that $\psi(m) \neq \psi(m')$ which is a contradiction.
On other hand, suppose x occurs in m' and it cannot be moved to the leftmost position in m' applying the commutation relations in I. Then we must have $m' = ayxb$ for some $y \in X_j$ and $j \in J$, where $a, b \in X^*$, for the leftmost occurrence of x in m'. Hence $m|_{X_j} \neq m'|_{X_j}$, because x is the first variable in $m|_{X_j}$ and x comes after y in $m'|_{X_j}$. Therefore, $\psi(m) \neq \psi(m')$ which is a contradiction. ∎

Now, $\psi(x \cdot m_1) = \psi(x \cdot m_2)$ implies that $\psi(m_1) = \psi(m_2)$. Both m_1 and m_2 are of length ℓ. By induction hypothesis it follows that $m_1 = m_2$, and hence $m = m'$. ∎

By Lemma 3.3.1, we can show that the zero testing for weighted automata over pc monoids reduces to zero-testing of weighted automata over partitioned pc monoids in deterministic polynomial time. More formally, we show the following result.

Lemma 3.3.2. *Let A be the given $\mathbb{F}$-weighted automaton of size s over a pc monoid M for which the non-commutation graph G_M has k-covering $\{G_i = (X_i, E_i)\}_{i=1}^k$. Then the zero testing of A is reducible to the zero testing of another $\mathbb{F}$-weighted automaton B over the associated k-partitioned pc monoid M' in deterministic polynomial time. Moreover the size of the automaton B is $O(ns^2k)$.*

Proof. The automaton B is simply obtained by applying the map ψ on the variables in M. For a variable x_t, let $J_t \subseteq \{1, 2, \ldots, k\}$ be the set of indices such that, $i \in J_t$ if and only if $x_t \in X_i$. Then $\psi(x_t) = \eta_{i_1} \otimes \cdots \otimes \eta_{i_{|J_t|}}$ where $i_1 < i_2 < \cdots < i_{|J_t|}$ and for each j, $i_j \in J_t$, $\eta_{i_j} = x_{ti_j}$. Now for each $q_0, q_k \in Q$ such that $(q_0, x_t, q_k) \in E$ [9] and $wt(q_0, x_t, q_k) = \alpha \in \mathbb{F}$, we introduce new states $q_1, \ldots, q_{|J_t|-1}$ and for each $j \leq |J_t| - 1$, add the edge $e_j = (q_{j-1}, \eta_{i_j}, q_j)$ in E and $wt(e_1) = \alpha$ and for other newly added edges the weight is 1. Since the number of edges in A is $O(ns^2)$, it is easy to see the number of nodes in B is $O(ns^2k)$. The fact that A computes the zero series if and only if B computes the zero series, follows from Lemma 3.3.1 and in particular from the fact that ψ is injective on the set of monomials. ∎

Worrell has already proved that the zero-testing of weighted automata over partitioned monoids whose non-commutation graphs are the union of disjoint cliques, can be reduced to the identity testing of non-commutative ABPs [114]. We restate the following proposition from Worrell's paper in a form that fits with our framework.

Proposition 3.3.1 (Adaptation of Proposition 5 of [114])**.** *Let A be a given $\mathbb{F}$-weighted automaton of size s over a partitioned pc monoid M computing a series S. Moreover the non-commutation graph G_M is the disjoint union of k cliques. Let N be the transition matrix of A. Then S is a zero series if and only if the ABPs $\lambda^T N^\ell \mu = 0$ for each $0 \leq \ell \leq s-1$, where λ, μ are vectors in $\mathbb{F}^s$.*

Combining Lemma 3.3.2 and Proposition 3.3.1, we obtain the following generalization of [38, Corollary 8.3] over arbitrary pc monoids which can be of independent interest.

[9] Here for simplicity of notation, we have used q_0, q_1 to represent an arbitrary pair such that there is a transition between them, and q_0 is not necessarily the initial state.

Theorem 3.3.1. *Let A be a given $\mathbb{F}$-weighted automaton of size s over any pc monoid M representing a series S. Then S is a nonzero series if and only if there exists a word $w \in X^*$ such that $[w]S$ is nonzero where the length of w is bounded by $O(n^3s^2)$.*

Proof. Observe that the non-commutation graph G_M has a trivial edge clique cover of size $\leq n^2$ where n is the size of the alphabet. Then we apply Lemma 3.3.2 to conclude that S is a zero series if and only if the series S' computed by the $\mathbb{F}$-weighted automaton B over the associated partitioned pc monoid (whose non-commutation graph is a disjoint union of cliques) is zero. The size s' of B is bounded by $O(n^3s^2)$. Now we use Proposition 3.3.1 to see that S' is identically zero if and only if the ABPs $\lambda^T N^\ell \mu = 0$ for each $0 \leq \ell \leq s'-1$ are identically zero where N is the transition matrix of B. Now notice that under the image of ψ map, the length of any word can only increase. In other words, for any word $w : |\psi(w)| \geq |w|$. It follows that $(S' = \psi(S))^{\leq s'-1}$ is a nonzero polynomial where S' is the part of $\psi(S)$ of degree at most $s'-1$. Since ψ is injective, it must be the case that $S^{\leq s'-1}$ is also a nonzero polynomial, and the proof of the theorem follows. ∎

3.4 Deterministic Zero Testing of Weighted Automata Over k-Clique Monoids

In this section, we show that zero testing for weighted automata over k-clique monoids for constant k is in deterministic quasi-polynomial time. In fact, by Lemma 3.3.2 and Proposition 3.3.1, the zero testing problem reduces to the polynomial identity testing of ABPs over partitioned pc monoids whose non-commutation graph is a disjoint union of k cliques. Thus, in order to prove Theorem 3.1.1 it suffices to design an efficient identity testing algorithm for ABPs computing polynomials in $\mathbb{F}\langle X_1\rangle \otimes \cdots \otimes \mathbb{F}\langle X_k\rangle$, where k is a constant and the variable sets $X_j = \{x_{ij}\}_{1\leq i\leq n}$ are of size n each and pairwise disjoint.

Evaluation over algebras : For a polynomial $f \in \mathbb{F}\langle X_1\rangle \otimes \cdots \otimes \mathbb{F}\langle X_k\rangle$ and a k-tuple of $\mathbb{F}$-algebras $\mathbf{A} = (A_1, \ldots, A_k)$, an *evaluation* of f in $\mathbf{A}$ is given by a k-tuple of maps $\boldsymbol{v} = (v_1, v_2, \ldots, v_k)$, where $v_i : X_i \to A_i$. We can extend it to the map $\boldsymbol{v} : \mathbb{F}\langle X_1\rangle \otimes \cdots \otimes \mathbb{F}\langle X_k\rangle \to A_1 \otimes \cdots \otimes A_k$ as follows: For any monomial $m = m_1 \otimes \cdots \otimes m_k$ where $m_i \in X_i^*$, let $\boldsymbol{v}(m) = v_1(m_1) \otimes \cdots \otimes v_k(m_k)$. In particular, for each $x \in X_j$ let $\boldsymbol{v}(x) = 1_1 \otimes \cdots \otimes v_j(x) \otimes \cdots \otimes 1_k$ where 1_j is the multiplicative identity of A_j. We can now extend $\boldsymbol{v}$ by linearity to all polynomials in the domain $\mathbb{F}\langle X_1\rangle \otimes \cdots \otimes \mathbb{F}\langle X_k\rangle$.
Next, we define a *partial evaluation* of $f \in \mathbb{F}\langle X_1\rangle \otimes \cdots \otimes \mathbb{F}\langle X_k\rangle$ in $\mathbf{A}$. Let $k' < k$ and $\hat{\boldsymbol{A}} = (A_1, \ldots, A_{k'})$ be a k'-tuple of $\mathbb{F}$-algebras. A partial evaluation of $\mathbb{F}\langle X_1\rangle \otimes \cdots \otimes \mathbb{F}\langle X_k\rangle$ in $\hat{\boldsymbol{A}}$ is given by a k'-tuple of maps $\hat{\boldsymbol{v}} = (v_1, \ldots, v_{k'})$, where $v_i : X_i \to A_i$. Now, we can define $\hat{\boldsymbol{v}} : \mathbb{F}\langle X_1\rangle \otimes \cdots \otimes \mathbb{F}\langle X_k\rangle \to A_1 \otimes \cdots \otimes A_{k'} \otimes \mathbb{F}\langle X_{k+1}\rangle \otimes \cdots \otimes \mathbb{F}\langle X_k\rangle$ as follows. For a monomial $m = (m_1 \otimes \cdots \otimes m_k)$, $m_i \in X_i^*$, we let $\hat{\boldsymbol{v}}(m) = v_1(m_1) \otimes \cdots \otimes v_{k'}(m_{k'}) \otimes m_{k'+1} \otimes \cdots \otimes m_k$. By linearity, the partial evaluation $\hat{\boldsymbol{v}}$ is defined for any $f \in \mathbb{F}\langle X_1\rangle \otimes \cdots \otimes \mathbb{F}\langle X_k\rangle$ where $\hat{\boldsymbol{v}}$ takes values in $A_1 \otimes \cdots \otimes A_{k'} \otimes \mathbb{F}\langle X_{k'+1}\rangle \otimes \cdots \mathbb{F}\langle X_k\rangle$.
Although it is implicit, we formally recall that when we consider ABPs over $\mathbb{F}\langle X_1\rangle \otimes \cdots \otimes \mathbb{F}\langle X_k\rangle$ the linear forms are defined over tensors of the form $1 \otimes \cdots \otimes x_{ij} \otimes \cdots \otimes 1$. These tensors play the role of an individual variable in the tensor product structure.

Some more notation : Let $S_{k,n,d,s}$ denote the set of all polynomials in $\mathbb{F}\langle X_1\rangle\otimes\cdots\otimes\mathbb{F}\langle X_k\rangle$ computed by ABPs of size s and layers $0,1,\ldots,d$, and $n = |X_i|$ for each $1 \le i \le k$. Following the notation in Theorem 3.2.1, we will denote by $\mathcal{H}_{k,n,d,s}$ the hitting set that we will construct for $S_{k,n,d,s}$. That is, $\mathcal{H}_{k,n,d,s}$ is a collection of evaluations in the ring of square matrices $\boldsymbol{v} = (v_1,\ldots,v_k)$, such that for any nonzero polynomial $f \in S_{k,n,d,s}$ there is an evaluation $\boldsymbol{v} = (v_1,\ldots,v_k) \in \mathcal{H}_{k,n,d,s}$ such that $\boldsymbol{v}(f)$ is a nonzero matrix. Recall from Theorem 3.2.1 that a quasi-polynomial size hitting set $\mathcal{H}_{1,n,d,s}$ for $S_{1,n,d,s}$ can be explicitly constructed. In the next lemma we describe an efficient bootstrapped construction of the hitting set $\mathcal{H}_{k,n,d,s}$ for the set $S_{k,n,d,s}$ of polynomials, from the hitting set $\mathcal{H}_{1,n,d,s}$.

Lemma 3.4.1. *There is a set of evaluation maps $\mathcal{H}_{k,n,d,s} = \{(v_1,\ldots,v_k) : v_i \in \mathcal{H}_{1,n,d,s_k}\}$ where $s_k = s(d+1)^{(k-1)}$ such that, for $i \in [k]$, we have $v_i : X_i \to \mathcal{M}_{d+1}(\mathbb{F})$, and $\mathcal{H}_{k,n,d,s}$ is a hitting set for the class of polynomials $S_{k,n,d,s}$. Moreover, the size of the set is at most $(nskd)^{O(k^2\log d)}$, and it can be constructed in deterministic $(nskd)^{O(k^2\log d)}$ time.*

The above lemma yields the identity test: we only need to evaluate the input polynomial on each point of the hitting set and check whether it is nonzero.
Before presenting the proof, we discuss two important ingredients. A polynomial f in $\mathbb{F}\langle X_1\rangle \otimes \cdots \otimes \mathbb{F}\langle X_k\rangle$ can be written as $f = \sum_{m\in X_k^*} f_m \otimes m$ where each m is a monomial over variables X_k and $f_m \in \mathbb{F}\langle X_1\rangle \otimes \cdots \otimes \mathbb{F}\langle X_{k-1}\rangle$. Given that f has a small ABP, we first show that each polynomial f_m also has a small ABP.

Lemma 3.4.2. *For each $f \in S_{k,n,d,s}$ and $m \in X_k^*$, the polynomial $f_m \in \mathbb{F}\langle X_1\rangle \otimes \cdots \otimes \mathbb{F}\langle X_{k-1}\rangle$ has an ABP of size $s(d+1)$ and $d+1$ layers.*

Proof. Suppose $f \in \mathbb{F}\langle X_1\rangle \otimes \cdots \otimes \mathbb{F}\langle X_k\rangle$ has an ABP B of size s and d layers, and the monomial $m = x_{i_1k}x_{i_2k}\cdots x_{i_\ell k}$ where some of the indices could be repeated. We will construct an ABP of size $s(d+1)$ for the polynomial f_m. First, we identify each variable $1\otimes\cdots\otimes x_{ij}\otimes\cdots\otimes 1$ as x_{ij} and construct the following ABP B' from B:
For every node u in the ABP B, we have nodes $(u,i), 0 \le i \le \ell$ in the ABP B'. We now describe the edges of B' and the edge labels. In the ABP B, let (u,v) be an edge, where u is in layer j and v is in layer $j+1$, for some $j \le d-1$. We can write the linear form labeling (u,v) as a sum $L_1 + L_2$, where L_1 is an affine linear form in variables from $X \setminus X_k$, and L_2 is a homogeneous linear form in variables from X_k.
For $0 \le r \le \ell - 1$: we put an edge from (u,r) to (v,r) with label L_1. For $0 \le r \le \ell-1$: we put an edge from (u,r) to $(v,r+1)$ with edge label $\alpha \cdot x_{i_{r+1}k}$ if the coefficient of $x_{i_{r+1}k}$ in L_2 is $\alpha \neq 0$. If s and t are the source and sink nodes of the ABP B, we designate $(s,0)$ and (t,ℓ) as the source and sink nodes of the ABP B'.
It is evident from the construction that the ABP B' has at most $s(d+1)$ many nodes. Furthermore, the only nonzero monomials in the polynomial computed by B' are of the form $m' \otimes m$, where m' is a monomial over the letters $X \setminus X_k$, and the coefficient of $m' \otimes m$ is the same as its coefficient in polynomial f. It follows, that B' computes the polynomial $f_m \otimes m$, and we can obtain an ABP for f_m by setting to 1 all the variables occurring in m. This completes the proof. ∎

For a polynomial f in $\mathbb{F}\langle X_1\rangle\otimes\cdots\otimes\mathbb{F}\langle X_k\rangle$, consider a partial evaluation $\boldsymbol{v} = (v_1,\ldots,v_{k-1})$ such that each $v_i : X_i \to \mathcal{M}_{t_i}(\mathbb{F})$. The evaluation $\boldsymbol{v}(f)$ is a $T \times T$ matrix with entries from $\mathbb{F}\langle X_k\rangle$, where $T = t_1t_2\cdots t_{k-1}$.

Lemma 3.4.3. *For each $p, q \in [T]$, the $(p,q)^{th}$ entry of $\boldsymbol{v}(f)$ can be computed by an ABP of size sT and $d+1$ layers.*

Proof.
In effect the edges of the input branching program B are now labelled by matrices of dimension T with entries are linear forms over the variables X'_k. To show that each entry of the final $T \times T$ matrix can be computed by an ABP of size sT, let us fix some (i,j) such that $1 \leq i, j \leq T$ and construct an ABP B'_{ij} computing the polynomial in the $(i,j)^{th}$ entry.
The construction of B'_{ij} is as follows. We make T copies of each node p (except the source and sink node) of B and label it as (p,k) for each $k \in [T]$. Let us fix two nodes p and q from B such that there is a $T \times T$ matrix M_{pq} labelling the edge (p,q) after the substitution. Then, for each $j_1, j_2 \in [T]$, add an edge between (p,j_1) and (q,j_2) in B'_{ij} and label it by the $(j_1,j_2)^{th}$ entry of M_{pq}. When p is the source node, for each $j_2 \in T$, add an edge between the source node and (q,j_2) in B'_{ij} and label it by the $(i,j_2)^{th}$ entry of M_{pq}. Similarly, when q is the sink node, for each $j_1 \in T$, add an edge between (p,j_1) and the sink node in B'_{ij} and label it by the $(j_1,j)^{th}$ entry of M_{pq}.
We just need to argue that the intermediate edge connections simulate matrix multiplications correctly. This is simple to observe, since for each path $\mathcal{P} = \{(s,p_1),(p_1,p_2),\ldots,(p_{\ell-1},t)\}$ in B (where s,t are the source and sink nodes respectively) and each $(j_1,\ldots,j_{\ell-1})$ such that $1 \leq j_1,\ldots,j_{\ell-1} \leq T$, there is a path $(s,(p_1,j_1)),((p_1,j_1),(p_2,j_2)),\ldots,((p_{\ell-1},j_{\ell-1}),t)$ in B'_{ij} that computes $M_{(s,p_1)}[i,j_1]M_{(p_1,p_2)}[j_1,j_2]\cdots M_{p_{\ell-1},t}[j_{\ell-1},j]$ where $M_{(p,q)}$ is the $T \times T$ matrix labelling the edge (p,q) in B. The size of B'_{ij} is sT, and the number of layers is $d+1$. ∎

Now we are ready to prove Lemma 3.4.1.
Proof of Lemma 3.4.1. The proof is by induction on k. For the base case $k=1$ the hitting set $\mathcal{H}_{1,n,d,s}$ of Theorem 3.2.1 suffices. We can write each nonzero $f \in S_{k,n,d,s}$ as $f = \sum_{m \in X_k^*} f_m \otimes m$, where $m \in X_k^*$ and $f_m \in \mathbb{F}\langle X_1\rangle \otimes \cdots \otimes \mathbb{F}\langle X_{k-1}\rangle$. Since $f \not\equiv 0$ we have $f_m \not\equiv 0$ for some $m \in X_k^*$. By Lemma 3.4.2, for each $m \in X_k{}^*$ the polynomial $f_m \in \mathbb{F}\langle X_1\rangle \otimes \cdots \otimes \mathbb{F}\langle X_{k-1}\rangle$ has an ABP of size $s(d+1)$. Let $s' = s(d+1)$.
By induction hypothesis, f_m is nonzero on some point in the set : $\mathcal{H}_{k-1,n,d,s'} = \{(v_1,v_2,\ldots,v_{k-1}) | v_i \in \mathcal{H}_{1,n,d,s'_{k-1}}\}$ where $s'_{k-1} = s'(d+1)^{k-2} = s(d+1)^{k-1}$. Hence, there is an evaluation $\boldsymbol{v'} \in \mathcal{H}_{k-1,n,d,s'}$ such that $\boldsymbol{v'}(f_m)$ is a nonzero matrix of dimension $(d+1)^{k-1}$. Interpreting $\boldsymbol{v'}$ as a *partial evaluation* for f, we observe that $\boldsymbol{v'}(f)$ is a $(d+1)^{k-1} \times (d+1)^{k-1}$ matrix with entries from $\mathbb{F}\langle X_k\rangle$. Since $\boldsymbol{v'}(f_m) \neq 0$, it follows that some $(p,q)^{th}$ entry of $\boldsymbol{v'}(f)$ is a nonzero polynomial $g \in \mathbb{F}\langle X_k\rangle$. By Lemma 3.4.3, each entry of $\boldsymbol{v'}(f)$ has an ABP of size $s(d+1)^{k-1}$. In particular, $g \in S_{1,n,d,s(d+1)^{k-1}}$ and it follows from Theorem 3.2.1 that there is a an evaluation v'' in $\mathcal{H}_{1,n,d,s(d+1)^{k-1}}$ such that $v''(g)$ is a nonzero matrix of dimension $(d+1) \times (d+1)$.
Thus, for the combined evaluation map $\boldsymbol{v} = (\boldsymbol{v'}, v'')$, $\boldsymbol{v}(f)$ is a nonzero matrix of dimension $(d+1)^k \times (d+1)^k$. Define $\mathcal{H}_{k,n,d,s} = \{(v_1,\ldots,v_k) : v_i \in \mathcal{H}_{1,n,d,s_k}\}$, where $s_k = s(d+1)^{k-1}$. However, by induction hypothesis, we have $\boldsymbol{v'} = (v_1,\ldots,v_{k-1}) \in \mathcal{H}_{k-1,n,d,s(d+1)}$ where each $v_i \in \mathcal{H}_{1,n,d,s(d+1)^{k-1}}$. Therefore, $\boldsymbol{v} = (\boldsymbol{v'}, v'') \in \mathcal{H}_{k,n,d,s}$ and $\mathcal{H}_{k,n,d,s}$ is a hitting set for the class of polynomials $S_{k,n,d,s}$.
Finally, note that $|\mathcal{H}_{k,n,d,s}| = |\mathcal{H}_{1,n,d,s_k}|^k$. Since $|\mathcal{H}_{1,n,d,s_k}| \leq (nds_k)^{O(\log d)}$, it follows that $|\mathcal{H}_{k,n,d,s}| \leq (nskd)^{O(k^2 \log d)}$, and the hitting set $\mathcal{H}_{k,n,d,s}$ can be constructed in the claimed

running time[10]. For zero testing, we need to evaluate the input ABP on the matrices of the hitting set, and this can be done in time polynomial in the input size and the size of the hitting set by matrix additions and multiplications.

3.5 Randomized Zero Testing of Weighted Automata Over k-Monoids

We now consider pc monoids more general than k-clique monoids. A *k-monoid* is a pc monoid M whose non-commutation graph G_M has a 2-covering $\{G_1, G_2\}$ such that G_1 has a edge clique cover of size k' and G_2 has a vertex cover of size $k - k'$, for some k'. It follows that G_M has a k-covering of cliques and star graphs. We assume that this k-covering of G_M is given as part of the input. Let $\mathbb{F}\langle M\rangle$ denote the $\mathbb{F}$-algebra generated by the monoid M.

Lemma 3.5.1. *Let $\{M_i\}_{i=1}^k$ be pc monoids defined over disjoint variable sets $\{X_i\}_{i=1}^k$, respectively. For each i, suppose $\mathcal{A}_i$ is a randomized procedure that outputs an evaluation $v_i : \mathbb{F}\langle M_i\rangle \to \mathcal{M}_{t_i(d)}(\mathbb{F})$ such that for any polynomial g_i in $\mathbb{F}\langle M_i\rangle$ of degree at most d, g_i is nonzero if and only if $v_i(g_i)$ is a nonzero matrix with probability at least $1 - \frac{1}{2k}$.*
Then, for the evaluation $\boldsymbol{v} : \mathbb{F}\langle M_1\rangle \otimes \cdots \otimes \mathbb{F}\langle M_k\rangle \to \mathcal{M}_{t_1(d)}(\mathbb{F}) \otimes \cdots \otimes \mathcal{M}_{t_k(d)}(\mathbb{F})$ such that $\boldsymbol{v} = (v_1, \ldots, v_k)$ and any nonzero polynomial $f \in \mathbb{F}\langle M_1\rangle \otimes \cdots \otimes \mathbb{F}\langle M_k\rangle$ of degree at most d, the matrix $\boldsymbol{v}(f)$ is nonzero with probability at least $1/2$.

Proof. The proof is by induction on k. For the base case $k = 1$, it is trivial. Let us fix an $f \in \mathbb{F}\langle M_1\rangle \otimes \cdots \otimes \mathbb{F}\langle M_k\rangle$ of degree at most d such that $f \not\equiv 0$. The polynomial f can be written as $f = \sum_{m \in M_k} f_m \otimes m$ where m are the words over the pc monoid M_k and $f_m \in \mathbb{F}\langle M_1\rangle \otimes \cdots \otimes \mathbb{F}\langle M_{k-1}\rangle$. Since $f \not\equiv 0$ we must have $f_m \not\equiv 0$ for some $m \in M_k$.
Now, inductively we have the evaluation $\boldsymbol{v'} = (v_1, \ldots, v_{k-1})$ for the class of polynomials in $\mathbb{F}\langle M_1\rangle \otimes \cdots \otimes \mathbb{F}\langle M_{k-1}\rangle$ of degree at most d. Since $f_m \not\equiv 0$, with high probability $\boldsymbol{v'}(f_m)$ is a nonzero matrix of dimension $\prod_{i=1}^{k-1} t_i(d)$. By induction the failure probability is bounded by $\frac{k-1}{2k}$.
As $\boldsymbol{v'}$ is a *partial evaluation* for f, we observe that $\boldsymbol{v'}(f)$ is a matrix of dimension $\prod_{i=1}^{k-1} t_i(d)$ whose entries are polynomials in $\mathbb{F}\langle M_k\rangle$. Since $\boldsymbol{v'}(f_m) \neq 0$ we conclude that some $(p, q)^{th}$ entry of $\boldsymbol{v'}(f)$ contains a nonzero polynomial $g \in \mathbb{F}\langle M_k\rangle$ of degree at most d. Choose the evaluation $v_k \in S_k$ which is the output of the randomized procedure $\mathcal{A}_k$, such that $v_k(g)$ is a nonzero matrix of dimension $t_k(d)$. Hence, for the combined evaluation $\boldsymbol{v} = (\boldsymbol{v'}, v_k)$, $\boldsymbol{v}(f)$ is a nonzero matrix of dimension $\prod_{i=1}^{k} t_i(d)$. A union bound shows that the failure probability is at most $1/2$. ∎

For the proof of Theorem 3.1.2, we first give a randomized polynomial-time identity testing algorithm for polynomials over pc monoids whose non-commutation graph is a star graph.

Lemma 3.5.2. *Let $M = ((X \cup y)^*, I)$ be a monoid whose non-commutation graph G_M is a star graph with center y. Then for any constant k, there is a randomized procedure that outputs an evaluation $\boldsymbol{v} : X \cup \{y\} \to \mathrm{M}_{t(d)}(\mathbb{F})$ where $t(d)$ is at most d, such that for any*

[10]Independent to the context of the current chapter, bootstrapping hitting sets has found other interesting applications in arithmetic circuit complexity [2].

polynomial $f \in \mathbb{F}\langle M \rangle$ of degree at most d, the polynomial f is nonzero if and only if $\boldsymbol{v}(f)$ is a nonzero matrix. The success probability of the algorithm is at least $1 - \frac{1}{2k}$.

Proof. If f is nonzero, then there exists a monomial m in M with nonzero coefficient. The idea is to isolate all monomials in $\{X \cup y\}^*$ that are equivalent to m in M. Let the degree of y in monomial m be $\ell \leq d$. Then m can be written as $m = m_1 y m_2 \cdots m_\ell y m_{\ell+1}$ where each m_i is a word in X^*. As X is a commuting set of variables, any permutation of m_i produces a monomial equivalent to m in M. Now consider the automaton in Figure 4.4.

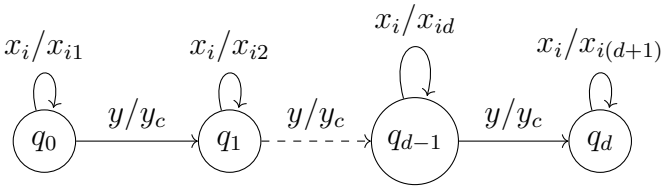


Figure 3.1: The transition diagram of the automaton

Let m as $m = m_1 y m_2 \cdots m_\ell y m_{\ell+1}$, where each m_i is a maximal substring of m in X^*. We refer to the m_i as blocks. The above automaton keeps count of blocks as it scans the monomial m. As it scans m, if the automaton is in the j^{th} block, it substitutes each variable $x_i \in X$ read by a corresponding commuting variable x_{ij} where the index j encodes the block number. The y variable is renamed by a commutative variable y_c. The transition matrices N_{x_i} and N_y of dimension $d+1$. The transition matrices are explicitly given below.

$$N_{x_i} = \begin{bmatrix} x_{i1} & 0 & 0 & \dots & 0 \\ 0 & x_{i2} & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & x_{id} & 0 \\ 0 & 0 & \dots & 0 & x_{i(d+1)} \end{bmatrix}, \qquad N_y = \begin{bmatrix} 0 & y_c & 0 & \dots & 0 \\ 0 & 0 & y_c & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & y_c \\ 0 & 0 & \dots & 0 & 0 \end{bmatrix}.$$

Now we explain this matrix substitution. Let $f = \sum_m \alpha_m m$, where $\alpha_m \in \mathbb{F}$. We write $f = \sum_{\ell=1}^{d} f_\ell$, where $f_\ell = \sum_{m:\deg_y(m)=\ell} \alpha_m m$. That is, f_ℓ is the part of f consisting of monomials m with y-degree $\deg_y(m) = \ell$.
From the description of the automaton, we can see that for each $\ell \in [d]$, the $(0,\ell)^{th}$ entry of the output matrix is the commutative polynomial $f_\ell^c \in \mathbb{F}[\{x_{i,j}\}_{1\leq i\leq n, 1\leq j\leq d+1}, y_c]$. The construction ensures the following: For each $0 \leq \ell \leq d$, $f_\ell = 0$ if and only if $f_\ell^c = 0$.
The randomized identity test is by substituting random scalar values for the commuting variables x_{ij} and y_c from a set $S \subseteq \mathbb{F}$ of size at least $2kd$, such that the output matrix becomes nonzero. The bound on the success probability follows from Polynomial Identity Lemma [34, 93, 116]. ∎

Now we are ready to prove Theorem 3.1.2.
Proof. Let M' be a pc monoid whose non-commutation graph $G_{M'}$ is a clique. Let $g \in \mathbb{F}\langle M' \rangle$ be a nonzero polynomial of degree at most d. By the Amitsur-Levitzki Theorem [8], if we substitute variables $x_i \in M'$ by generic matrix of size d over the variables

$\{x^{(i)}_{u,v}\}_{1\leq u,v\leq d}$, the output matrix is nonzero [11]. Moreover, the entries of the output matrix are commutative polynomials of degree at most d in the variables $\{x^{(i)}_{u,v}\}_{1\leq i\leq n,1\leq u,v\leq d}$. It suffices to randomly substitute for each $x^{(i)}_{u,v}$ variable from a set $S \subseteq \mathbb{F}$ of size at least $2kd$. This defines the evaluation map $v : \mathbb{F}\langle M'\rangle \to \mathbb{M}_d(\mathbb{F})$. The resulting identity test succeeds with probability at least $1 - \frac{1}{2k}$. For the star graphs, the evaluation map is already defined in Lemma 3.5.2.
Given a $\mathbb{F}$-weighted automaton A of size s over a k-monoid $M = (X^*, I)$, by Theorem 3.3.1, the zero testing of A reduces to identity testing of a collection of ABPs of the form : $f = \lambda^T N^d \mu$ over $\mathbb{F}\langle M\rangle$, where N is the transition matrix of A and d is bounded by $O(ns^2k)$. Now, to test identity of f where M is a k-monoid, it suffices to test identity of $\psi(f)$ where ψ is the injective homomorphism from Lemma 3.3.1. Now $\psi(f)$ in $\mathbb{F}\langle M'_1\rangle \otimes \cdots \otimes \mathbb{F}\langle M'_k\rangle$, where for each $i \in [k]$ the non-commutation graph of M'_i is either a clique or a star.
By Lemma 3.5.1, we can construct the evaluation map $\boldsymbol{v} = v_1 \otimes v_2 \otimes \cdots \otimes v_k$ where for each $i \in [k]$, v_i is an evaluation map for either a clique or a star graph depending on M'_i. The range of $\boldsymbol{v}$ is matrices of dimension at most d^k, which is bounded by $(sn)^{O(k)}$ as d is bounded by $O(ns^2k)$. This completes the proof of Theorem 3.1.2. ∎

3.6 Conclusion

The bootstrapped construction presented in Section 3.4 designs a hitting set of quasi-polynomial size over a suitable matrix algebra for the k-clique monoid problem. However, in the hope of designing a polynomial-time algorithm, one may try to exploit finer structures in the problem than evaluating it over a matrix algebra. It seems natural to get inspiration from the white-box polynomial-time identity testing algorithm for non-commutative arithmetic branching programs [77]. However, it is unclear whether such a technique can be adapted in the present setting of partially commutative monoids to obtain a deterministic polynomial-time algorithm. This is the main open problem left for future research.

[11] In fact the Amitsur-Levitzki theorem guarantees that generic matrices of size $\lceil \frac{d}{2} \rceil + 1$ suffice [8].

Chapter 4

PIT and Irreducibility Testing in the Non-Associative Non-Commutative Model

In this chapter we study arithmetic computations in the non-associative, and non-commutative free polynomial ring $\mathbb{F}\{x_1, x_2, \ldots, x_n\}$. Prior to this work, non-associative arithmetic computation was considered by Hrubeš, Wigderson, and Yehudayoff [49] who showed lower bounds and proved completeness results. We consider the algorithmic aspects and give deterministic polynomial time algorithms for Polynomial Identity Testing (PIT) and irreducibility testing over $\mathbb{F}\{x_1, x_2, \ldots, x_n\}$. Our PIT result is obtained by a suitable adaptation of the PIT algorithm of Raz-Shpilka [77] for non-commutative ABPs. The irreducibility testing algorithm utilizes ideas from a factorization algorithm for homogeneous non-commutative ABPs [12].

4.1 Introduction

Non-commutative computation, introduced in complexity theory by Hyafil [51] and Nisan [71], is an important subfield of algebraic complexity theory. The main algebraic structure of interest is the free non-commutative ring $\mathbb{F}\langle X\rangle$ over a field $\mathbb{F}$, where $X = \{x_1, x_2, \cdots, x_n\}$ is a set of free noncommuting variables. A central problem is Polynomial Identity Testing (PIT) which may be stated as follows:

Problem 4.1.1. *Let $f \in \mathbb{F}\langle X\rangle$ be a polynomial represented by a non-commutative arithmetic circuit C. The circuit C can either be given by a black box (using which we can evaluate C on matrices with entries from $\mathbb{F}$ or an extension field), or the circuit may be explicitly given. The algorithmic problem is to check if the polynomial computed by C is identically zero.*

We recall the formal definition of a non-commutative arithmetic circuit.
When the multiplication operation of the circuit is *non-commutative*, it is called a *non-commutative arithmetic circuit* and it computes a polynomial in the free non-commutative ring $\mathbb{F}\langle X\rangle$. Since cancellation of terms is restricted by noncommutativity, intuitively it appears non-commutative polynomial identity testing would be easier than polynomial

identity testing in the commutative case. This intuition is supported by the fact that there is a deterministic polynomial-time white-box PIT algorithm for non-commutative ABP [77]. In the commutative setting a deterministic polynomial-time PIT for ABPs would be a major breakthrough.[1] However, there is little progress towards obtaining an efficient deterministic PIT for general non-commutative arithmetic circuits. For example, the problem is open even for non-commutative *skew* circuits.

If *associativity* is also dropped then it turns out that PIT becomes easy, as we show in this work. More precisely for a set of variables $X = \{x_1, x_2, \ldots, x_n\}$, we consider the free non-commutative and non-associative ring of polynomials $\mathbb{F}\{X\}$, where a polynomial is an $\mathbb{F}$-linear combination of monomials, and each monomial comes with a bracketing order of multiplication. For example, in the non-associative ring $\mathbb{F}\{X\}$ the monomial $(x_1(x_2x_1))$ is different from monomial $((x_1x_2)x_1)$, although in the associative ring $\mathbb{F}\langle X\rangle$ they clearly coincide.

When the multiplication operation is both non-commutative and non-associative, it is called a *non-associative non-commutative circuit* and it computes a polynomial in the free non-associative non-commutative ring $\mathbb{F}\{X\}$. Previously, the non-associative arithmetic model of computation was considered by Hrubeš, Wigderson, and Yehudayoff [49]. They showed completeness and explicit lower bound results for this model. We study the PIT problem and give a deterministic polynomial time algorithm which $C \equiv 0$ for an input arithmetic circuit over $\mathbb{F}\{X\}$.

Remark 4.1.1. *We note that our algorithm in the above result does not depend on the choice of the field $\mathbb{F}$. A recent result of Lagarde et al. [63] shows an exponential lower bound, and a deterministic polynomial-time* PIT *algorithm over $\mathbb{F}\langle X\rangle$ for non-commutative circuits where all parse trees in the circuit are isomorphic. We also note that in [15] an exponential lower bound is shown for set-multilinear arithmetic circuits with the additional semantic constraint that each monomial has a unique parse tree in the circuit (but different monomials can have different parse trees).*

Next, we consider the problem of irreducibility testing in the ring $\mathbb{F}\{X\}$. Irreducibility testing is an important problem which has been studied over various domains. For example, over integers, irreducibility testing corresponds to the classical primality testing problem, which has a deterministic polynomial time algorithm [3]. Over the polynomial ring $\mathbb{F}[X]$ Kaltofen [54] gave a randomized polynomial time algorithm which can even factorize an input polynomial f given by an arithmetic circuit. The non-commutative polynomial ring $\mathbb{F}\langle X\rangle$ is not a *unique factorization domain* [73] and this poses difficulties towards algorithm design. However, unique factorization holds for homogeneous polynomials in $\mathbb{F}\langle X\rangle$, and it is shown in [12] that for homogeneous polynomials given by non-commutative circuits, the unique factorization into irreducible factors can be computed in randomized polynomial time (essentially, by reduction to the non-commutative PIT problem). In this chapter, we give a deterministic polynomial time algorithm for irreducibility testing for polynomials in $\mathbb{F}\{X\}$. Further, our algorithm finds a non-trivial factorization of the input polynomial in case it is reducible.

[1] The situation is similar even in the lower bound case where Nisan proved that non-commutative determinant or permanent polynomial would require exponential-size algebraic branching program [71].

4.1.1 Proof Ideas and Techniques

Identity Testing Result: The main ideas for our algorithm are based on the white-box Raz-Shpilka PIT algorithm for non-commutative ABPs [77]. As in the Raz-Shpilka algorithm [77], if the circuit computes a nonzero polynomial $f \in \mathbb{F}\{X\}$, then our algorithm outputs a *certificate monomial* m such that coefficient of m in f is nonzero.

We first sketch the main steps of the Raz-Shpilka algorithm. The Raz-Shpilka algorithm processes the input ABP (assumed homogeneous) layer by layer. Suppose layer i of the ABP has w nodes. The algorithm maintains a spanning set $\mathbb{B}_i$ of at most w many linearly independent w-dimensional vectors of monomial coefficients. More precisely, each vector $v_m \in \mathbb{B}_i$ is the vector of coefficients of monomial m computed at each of the w nodes in layer i. Furthermore, the coefficient vector at layer i of any monomial is in the span of $\mathbb{B}_i$. The construction of $\mathbb{B}_{i+1}$ from $\mathbb{B}_i$ can be done efficiently. Clearly the identity testing problem can be solved by checking if there is a nonzero vector in $\mathbb{B}_d$, where d is the total number of layers.

Now we sketch our PIT algorithm for polynomials over $\mathbb{F}\{X\}$ given by circuits. Let f be the input polynomial given by the circuit C.

We encode monomials in the free non-associative non-commutative ring $\mathbb{F}\{X\}$ as monomials in the free non-commutative ring $\mathbb{F}\langle X, (,)\rangle$, such that the encoding preserves the multiplication structure of $\mathbb{F}\{X\}$ (Observation 4.2.1). For $1 \leq j \leq d$, we can efficiently find from C a homogeneous circuit C_j that computes the degree j homogeneous part of C. Thus, it suffices to test if $C_j \equiv 0$ for each j. Hence, it suffices to consider the case when $f \in \mathbb{F}\{X\}$ is homogeneous and C is a homogeneous circuit computing f.

For $j \leq d$ let G_j denote the set of degree j gates of C. The algorithm maintains a set $\mathbb{B}_j$ of $|G_j|$-dimensional linearly independent vectors of monomial coefficients such that any degree j monomial's coefficient vector is in the linear span of $\mathbb{B}_j$. Clearly, $|\mathbb{B}_j| \leq |G_j|$. We compute $\mathbb{B}_{j+1}$ from the sets $\{\mathbb{B}_i : 1 \leq i \leq j\}$. For each vector in $\mathbb{B}_j$ we also keep the corresponding monomial. In the non-associative model a degree d monomial $m = (m_1 m_2)$ is generated in a *unique way*. To check if the coefficient vector of m is in the span of $\mathbb{B}_d$ it suffices to consider vectors in the spans of $\mathbb{B}_{d_1}$ and $\mathbb{B}_{d_2}$, where $d_1 = \deg(m_1)$ and $d_2 = \deg(m_2)$. This is a crucial difference from a general non-commutative circuit and using this property we can compute $\mathbb{B}_{j+1}$.

Irreducibility testing in $\mathbb{F}\{X\}$

Our Irreducibility testing algorithm uses the PIT algorithm described earlier as a subroutine and also repeatedly uses a result of [12]. In [12] the authors show that given a monomial m and a homogeneous non-commutative circuit C, in deterministic polynomial time circuits for the formal left and right derivatives of C with respect to m can be efficiently computed.

We sketch the easy case, when the given polynomial f of degree d has no constant term. Applying our PIT algorithm to the homogeneous circuit C_d (computing the dth degree homogenous component f_d) we find a nonzero monomial $m = (m_1\ m_2)$ of degree d in f_d along with its coefficient $c_m(f)$. Notice that for any nontrivial factorization $f = gh$, m_1 is a nonzero monomial in g and m_2 is a nonzero monomial in h. Suppose $|m_1| = d_1$ and $|m_2| = d_2$. Then the left derivative of C_d with respect to m_1 gives $c_{m_1}(g)\ h_{d_2}$ and the right derivative of C_d with respect to m_2 gives $c_{m_2}(h)\ g_{d_1}$. We now use the circuits for these derivatives and the non-associative structure, to find circuits for different homogeneous parts of g and h. The details, including the general case when f has a nonzero constant

term, is in Section 4.4.

4.2 Preliminaries

For an arithmetic circuit C, a *parse tree* for a monomial m is a multiplicative sub-circuit of C rooted at the output gate defined by the following process starting from the output gate:

- At each + gate retain exactly one of its input gates.
- At each $\times$ gate retain both its input gates.
- Retain all inputs that are reached by this process.
- The resulting subcircuit is multiplicative and computes a monomial m (with some coefficient).

For arithmetic circuits C computing polynomials in the free non-associative non-commutative ring $\mathbb{F}\{X\}$, the same definition for the parse tree of a monomial applies. As explained in the introduction, in this case each parse tree (generating some monomial) comes with a bracketed structure for the multiplication. It is convenient to consider a polynomial in $\mathbb{F}\{x_1, \ldots, x_n\}$ as an element in the non-commutative ring $\mathbb{F}\langle x_1, \ldots, x_n, (,)\rangle$ where we introduce two auxiliary variables "(" and ")" (for left and right bracketing) to encode the parse tree structure of any monomial. We illustrate the encoding by the following example.
Consider the monomial (which is essentially a binary tree with leaves labeled by variables) in the non-associative ring $\mathbb{F}\{x, y\}$ shown in Figure 4.1. Its encoding as a bracketed string in the free non-commutative ring $\mathbb{F}\langle x, y, (,)\rangle$ is ((x y) y) and its parse tree shown in Figure 4.2.
Consider an arithmetic circuit C computing a polynomial $f \in \mathbb{F}\{X\}$. The circuit C can be efficiently transformed to a circuit $\tilde{C}$ that computes the corresponding polynomial $\tilde{f} \in \mathbb{F}\langle X, (,)\rangle$ by simply introducing the bracketing structure for each multiplication gate of C in a bottom-up manner as indicated in the following example figures. Consider the circuits described in Figures 4.1 and 4.2 where f_i, g_i, h_i's are polynomials computed by subcircuits. Clearly the bracket variables preserve the parse tree structure. The following fact is immediate.

Observation 4.2.1. *A non-associative non-commutative circuit C computes a nonzero polynomial $f \in \mathbb{F}\{X\}$ if and only if the corresponding non-commutative circuit $\tilde{C}$ computes a nonzero polynomial $\tilde{f} \in \mathbb{F}\langle X, (,)\rangle$.*

Given a non-commutative circuit C computing a homogeneous polynomial in $\mathbb{F}\langle X\rangle$ and a monomial m over X, one can talk of the left and right derivatives of C w.r.t m [12]. Let $f = \sum_{m'} c_{m'}(f)m'$ for some $f \in \mathbb{F}\langle X\rangle$ and A be the subset of monomials m' of f that have m as prefix. Then the left derivative of f w.r.t. m is

$$\frac{\partial^{\ell} f}{\partial m} = \sum_{m' \in A} c_{m'}(f)m'',$$

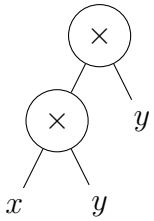


Figure 4.1: A non-associative and non-commutative monomial xyy

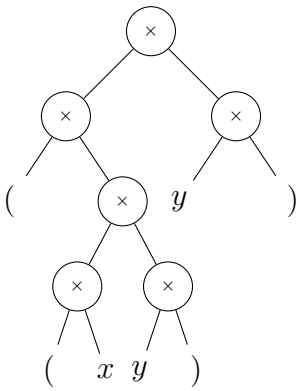


Figure 4.2: Corresponding monomial $((xy)\ y) \in \mathbb{F}\langle X\rangle$.

Figure 4.5: non-associative circuit and its corresponding non-commutative bracketed circuit

where $m' = m \cdot m''$ for $m' \in A$. Similarly we can define the right derivative[2] $\frac{\partial^r f}{\partial m}$. As shown in [12], if f is given by a circuit C then in deterministic polynomial time we can compute circuits for $\frac{\partial^\ell f}{\partial m}$ and $\frac{\partial^r f}{\partial m}$. We briefly discuss this in the following lemma.

Lemma 4.2.1. [12] *Given a non-commutative circuit C of size s computing a homogeneous polynomial f of degree d in $\mathbb{F}\langle X\rangle$ and monomial m, there is a deterministic* $\mathrm{poly}(n, d, s)$ *time algorithm that computes circuits $C_{m,\ell}$ and $C_{m,r}$ for the left and right derivatives $\frac{\partial^\ell C}{\partial m}$ and $\frac{\partial^r C}{\partial m}$, respectively.*

Proof. We explain only the left partial derivative case. Let m be a degree d' monomial and $f \in \mathbb{F}\langle X\rangle$ be a homogeneous degree d polynomial f computed by circuit C. The proof is similar to [12] but we provide it here for completeness. A small substitution deterministic finite automaton A with $d'+1$ states is constructed that recognizes all length d strings with prefix m and substitutes 1 for prefix m. Let $m = x_{i_1}x_{i_2}\cdots x_{i_{d'}}$ then the automaton is The transition of this automaton can be represented by the $(d'+1)\times(d'+1)$ matrices M_i defined as

$$\begin{aligned} M_i[k,\ell] &= 1[x_{i_k} = x_i] && \text{when } \ell = k+1 \text{ and } k \le d' \\ &= x_i && \text{when } \ell = k = d'+1 \\ &= 0 && \text{otherwise} \end{aligned} \tag{4.1}$$

[2]The superscripts ℓ, r denote left and right respectively.

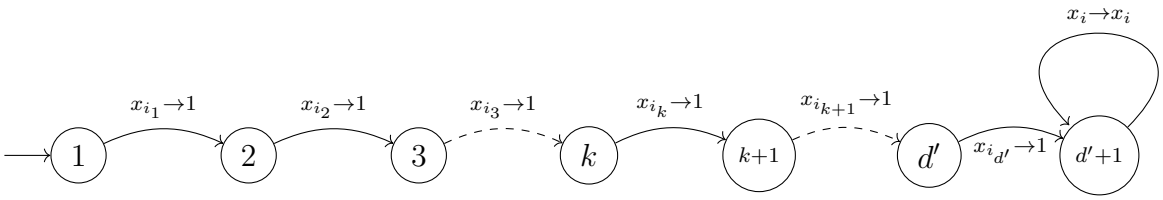


Figure 4.6: The transition diagram of the automaton.

where $1[x_{i_k} = x_i]$ is the indicator that the variable in the kth position of m is the same as x_i or not. Now for any monomial $m' = m \cdot \hat{m}$ where $\hat{m}$ is of degree $d - d'$ we see that $m'(M_1, M_2, \ldots, M_n)[1, d'+1] = \hat{m}$ and if m' does not have m as a prefix then we have $m'(M_1, M_2, \ldots, M_n)[1, d'+1] = 0$. By linearity of evaluation we conclude that $f(M_1, M_2, \ldots, M_n)[1, d'+1]$ contains the sum of precisely those monomials (along with their respective coefficients) in f which have m as a prefix. In order to obtain a circuit for $\frac{\partial^\ell f}{\partial m}$ we replace the variables x_i in C by the matrices M_i and we 'blow up' each gate in C into $(d'+1)^2$ many gates which simulates the matrix multiplication. This gives a circuit for $\frac{\partial^\ell f}{\partial m}$ which is of size $\text{poly}(n, d, s)$. ∎

The left and right partial derivatives of inhomogeneous polynomials are similarly defined. The same matrix substitution works for non-homogeneous polynomials as well [12]. As discussed above, given a non-associative arithmetic circuit C computing a polynomial $f \in \mathbb{F}\{X\}$, we can transform C into a non-commutative circuit $\tilde{C}$ that computes a polynomial $\tilde{f} \in \mathbb{F}\langle X, (,)\rangle$. Suppose we want to compute the left partial derivative of f w.r.t. a monomial $m \in \mathbb{F}\{X\}$. Using the tree structure of m we transform it into a monomial $\tilde{m} \in \mathbb{F}\langle X, (,)\rangle$ and then we can apply Lemma 4.2.1 to $\tilde{C}$ and $\tilde{m}$ to compute the required left partial derivative. We can similarly compute the right partial derivative. We use this in Section 4.4.
We also note the following simple fact that the homogeneous parts of a polynomial $f \in \mathbb{F}\{X\}$ given by a circuit C can be computed efficiently. We can apply the above transformation to obtain circuit $\tilde{C}$ and use a standard lemma (see e.g., [97]) to compute the homogeneous parts of $\tilde{C}$.

Lemma 4.2.2. *Given a non-commutative circuit C of size s computing a non-commutative polynomial f of degree d in $\mathbb{F}\langle X, (,)\rangle$, one can compute homogeneous circuits C_j (where each gate computes a homogeneous polynomial) for j^{th} homogeneous part f_j of f, where $0 \leq j \leq d$, deterministically in time* $\text{poly}(n, d, s)$.

4.3 Identity Testing in $\mathbb{F}\{X\}$

In this section we describe our identity testing algorithm.

Theorem 4.3.1. *Let $f(x_1, x_2, \ldots, x_n) \in \mathbb{F}\{X\}$ be a degree d polynomial given by an arithmetic circuit of size s. Then in deterministic* $\text{poly}(s, n, d)$ *time we can test if f is an identically zero polynomial in $\mathbb{F}\{X\}$.*

Proof. By Lemma 4.2.2 we can assume that the input is a homogeneous non-associative circuit C computing some homogeneous degree d polynomial in $\mathbb{F}\{X\}$ (i.e. every gate in C computes a homogeneous polynomial). Also, all the $\times$ gates in C have fanin 2 and $+$ gates have unbounded fanin. We can assume the output gate is a $+$ gate. We can also assume w.l.o.g. that the $+$ and $\times$ gates alternate in each path from input gate to output gate (otherwise we introduce sum gates with fan-in 1).
The j^{th}-layer of circuit C to be the set of all $+$ gates in computing degree j homogeneous polynomials. Let s^+ be the total number of $+$ gates in C. To each monomial m we can associate a vector $v_m \in \mathbb{F}^{s^+}$ of coefficients, where v_m is indexed by the $+$ gates in C, and $v_m[g]$ is the coefficient of monomial m in the polynomial computed at the $+$ gate g. We can also write

$$v_m[g] = c_m(p_g),$$

where p_g is the polynomial computed at the sum gate g.
For the j^{th} layer of $+$ gates, we will maintain a maximal linearly independent set $\mathbb{B}_j$ of vectors v_m of monomials. These vectors correspond to degree j monomials. Although $v_m \in \mathbb{F}^{s^+}$, notice that $v_m[g] = 0$ at all $+$ gates that do not compute a degree j polynomial. Thus, $|\mathbb{B}_j|$ is bounded by the number of $+$ gates in the j^{th} layer. Hence, $|\mathbb{B}_j| \leq s$.
The sets $\mathbb{B}_j$ are computed inductively for increasing values of j. For the base case, the set $\mathbb{B}_1$ can be easily constructed by direct computation. Inductively, suppose the sets $\mathbb{B}_i : 1 \leq i \leq j-1$ are already constructed. We describe the construction of $\mathbb{B}_j$. Computing $\mathbb{B}_d$ and checking if there is a nonzero vector in it yields the identity testing algorithm.
We now describe the construction for the j^{th} layer assuming we have basis $\mathbb{B}_{j'}$ for every $j' < j$. Consider a $\times$ gate with its children computing homogeneous polynomials of degree d_1 and d_2 respectively. Notice that $j = d_1 + d_2$ and $0 < d_1, d_2 < j$. Consider the monomial[3] set

$$M = \{m_1 m_2 \mid v_{m_1} \in \mathbb{B}_{d_1} \text{ and } v_{m_2} \in \mathbb{B}_{d_2}\}.$$

We construct vectors $\{v_m \mid m \in M\}$ as follows.

$$v_{m_1 m_2}[g] = \sum_{(g_{d_1}, g_{d_2})} v_{m_1}[g_{d_1}] v_{m_2}[g_{d_2}],$$

where g is a $+$ gate in the j^{th} layer, g_{d_1} is a $+$ gate in the d_1^{th} layer, g_{d_2} is a $+$ gate in the d_2^{th} layer, and there is a $\times$ gate which is input to g and computes the product of g_{d_1} and g_{d_2}.
Let $\mathbb{B}_{d_1,d_2}$ denote a maximal linearly independent subset of $\{v_m \mid m \in M\}$. Then we let $\mathbb{B}_d$ be a maximal linearly independent subset of

$$\bigcup_{d_1+d_2=d} \mathbb{B}_{d_1,d_2}.$$

Claim 4.3.1. *For every monomial m of degree j, v_m is in the span of $\mathbb{B}_j$.*

[3]We note that the non-associative monomial $m_1 m_2$ is a binary tree with the root having two children: the left child is the root of the binary tree for m_1 and the right child is the root of the binary tree for m_2.

Proof of Claim. Let $m = m_1 m_2$ and the degree of m_1 is d_1 and the degree of m_2 is d_2 [4]. By *Induction Hypothesis* vectors v_{m_1} and v_{m_2} are in the span of $\mathbb{B}_{d_1}$ and $\mathbb{B}_{d_2}$ respectively. Hence, we can write

$$v_{m_1} = \sum_{i=1}^{D_1} \alpha_i v_{m_i} \quad v_{m_i} \in \mathbb{B}_{d_1} \quad \text{and} \quad v_{m_2} = \sum_{j=1}^{D_2} \beta_j v_{m'_j} \quad v_{m'_j} \in \mathbb{B}_{d_2},$$

where $|\mathbb{B}_{d_j}| = D_j$. Now, for a gate g in the j^{th} layer, By Induction Hypothesis and by construction we have

$$\begin{aligned} v_m[g] &= \sum_{(g_{d_1}, g_{d_2})} v_{m_1}[g_{d_1}] v_{m_2}[g_{d_2}] = \sum_{g_{d_1}, g_{d_2}} (\sum_{i=1}^{D_1} \alpha_i v_{m_i}[g_{d_1}])(\sum_{j=1}^{D_2} \beta_j v_{m'_j}[g_{d_2}]) \\ &= \sum_{i=1}^{D_1} \sum_{j=1}^{D_2} \alpha_i \beta_j \sum_{g_{d_1}, g_{d_2}} v_{m_i}[g_{d_1}] v_{m'_j}[g_{d_2}] = \sum_{i=1}^{D_1} \sum_{j=1}^{D_2} \alpha_i \beta_j v_{m_i m'_j}[g]. \end{aligned}$$

Thus v_m is in the span of $\mathbb{B}_{d_1,d_2}$ and hence in the span of $\mathbb{B}_j$. This proves the claim. The PIT algorithm only has to check if $\mathbb{B}_d$ has a nonzero vector. This proves the claim. Suppose the input non-associative circuit C computing some degree d polynomial $f \in \mathbb{F}\{X\}$ is inhomogeneous. Then, using Lemma 4.2.2 we can first compute in polynomial time homogeneous circuits $C_j \ : \ 0 \leq j \leq d$, where C_j computes the degree-j homogeneous part f_j. Then we run the above algorithm on each C_j to check whether f is identically zero. This completes the proof of the theorem. ∎

4.4 Irreducibility Testing in $\mathbb{F}\{X\}$

In this section we describe our polynomial-time white-box irreducibility testing algorithm for polynomials in $\mathbb{F}\{X\}$. More precisely, given as input a non-associative circuit C computing a polynomial $f \in \mathbb{F}\{X\}$, the algorithm detects whether f is irreducible or not. In case it is not irreducible, it outputs a non-trivial factorization of f. The algorithm uses as subroutine the PIT algorithm for polynomials in $\mathbb{F}\{X\}$ described in Section 4.3.

Theorem 4.4.1. *Let $f \in \mathbb{F}\{X\}$ be a degree d polynomial given by a circuit of size s. If $\mathbb{F} = \mathbb{Q}$, we give a deterministic* $\text{poly}(s, n, d)$ *time algorithm that computes a nontrivial factorization of f or reports f is irreducible. If $\mathbb{F}$ is a finite field such that $char(\mathbb{F}) = p$, we obtain a deterministic* $\text{poly}(s, n, d, p)$ *time algorithm that computes a nontrivial factorization of f or reports that f is irreducible.*

Proof. We build the algorithm through some special cases. We start with the case when the input polynomial f has no constant term. When f has a constant term we recover the factors without their constant terms and then we separately compute the constant terms.

[4] Here a crucial point is that for a non-associative monomial of degree d, such a choice for d_1 and d_2 is *unique*. This is a place where a general non-commutative circuit behaves very differently.

Lemma 4.4.1. *Let $f \in \mathbb{F}\{X\}$ be a degree d polynomial given by a circuit C of size s such that the constant term in f is zero. Furthermore, suppose there is a factorization $f = g \cdot h$ such that the constant terms in g and h are also zero. Then in deterministic* $\mathrm{poly}(n, d, s)$ *time we can compute the circuits for polynomials g and h.*

Proof. We first consider the even more restricted case when C computes a homogeneous degree d polynomial $f \in \mathbb{F}\{X\}$. For the purpose of computing partial derivatives, it is convenient to transform C into the non-commutative circuit $\tilde{C}$, as explained in Section 5.2, which computes the fully bracketed polynomial $\tilde{f} \in \mathbb{F}\langle X, (,)\rangle$. Using Theorem 4.3.1 we compute a monomial $m = (m_1 m_2)$ where m_1 and m_2 are also fully bracketed. We can transform $\tilde{C}$ to drop the outermost opening and closing brackets. Now, using Lemma 4.2.1, we compute the resulting circuits left partial derivative w.r.t. m_1 and right partial derivative w.r.t. m_2. Call these $\tilde{f}_1$ and $\tilde{f}_2$. We can check if $\tilde{f} = (\tilde{f}_1 \tilde{f}_2)$: we first recover the corresponding non-associative circuits for f_1 and f_2 from the circuits for $\tilde{f}_1$ and $\tilde{f}_2$. Then we can apply the PIT algorithm of Theorem 4.3.1 to check if $f = f_1 f_2$. Clearly, f is irreducible iff $f \neq f_1 f_2$.
Now we prove the actual statement. Applying Lemma 4.2.2, we compute homogeneous circuits $C_j : 1 \leq j \leq d$ for the homogeneous degree j component f_j of the polynomial f. Clearly $f_d = g_{d_1} h_{d_2}$. We run the PIT algorithm of Theorem 4.3.1 on the circuit C_d to extract a monomial m of degree d along with its coefficient $c_m(f_d)$ in f_d. Notice that the monomial m is of the form $m = (m_1\ m_2)$. If g and h are nontrivial factors of f then m_1 and m_2 are monomials in g and h respectively. Compute the circuits for the left and right derivatives with respect to m_1 and m_2.

$$\frac{\partial^{\ell} C_d}{\partial m_1} = c_{m_1}(g_{d_1}) \cdot h_{d_2} \quad \text{and} \quad \frac{\partial^{r} C_d}{\partial m_2} = c_{m_2}(h_{d_2}) \cdot g_{d_1}.$$

In general the $(i + d_2)^{th} : i \leq d - d_2$ homogeneous part of f can be expressed as

$$f_{i+d_2} = g_i h_{d_2} + \sum_{t=i+1}^{i+d_2-1} g_t h_{d_2\ -\ (t-i)}.$$

We depict the circuit C_{i+d_2} for the polynomial f_{i+d_2} in Figure 4.7. The top gate of the circuit is a + gate. From C_{i+d_2}, we construct another circuit C'_{i+d_2} keeping only those $\times$ gates as children whose left degree is i and right degree is d_2. The resulting circuit is shown in Figure 4.8. The circuit C'_{i+d_2} must compute $g_i h_{d_2}$. By taking the right partial of C'_{i+d_2} with respect to m_2, we obtain the circuit for $c_{m_2}(h_{d_2})\ g_i$.
We repeat the above construction for each $i \in [d_1]$ to obtain circuits for $c_{m_2}(h_{d_2}) g_i$ for $1 \leq i \leq d_1$. Similarly we can get the circuits for $c_{m_1}(g_{d_1}) h_i$ for each $i \in [d_2]$ using the left derivatives with respect to the monomial m_1.
By adding the above circuits we get the circuits C_g and C_h for $c_{m_2}(h_{d_2}) g$ and $c_{m_1}(g_{d_1}) h$ respectively. We set $C_g = \frac{c_{m_2}(h_{d_2})}{c_m(f)} g$ so that $C_g C_h = f$. Using PIT algorithm one can easily check whether g and h are nontrivial factors. ∎

Now we consider the general case when f and its factors g, h have arbitrary constant terms. In the subsequent proofs we assume, for convenience, that $\deg(g) \geq \deg(h)$. The case when $\deg(g) < \deg(h)$ can be handled analogously. We first consider the case $\deg(g) = \deg(h)$.

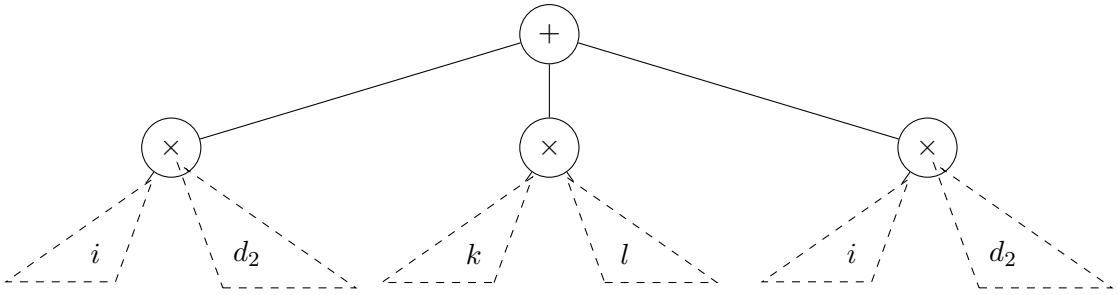


Figure 4.7: Circuit C_{i+d_2} for f_{i+d_2}

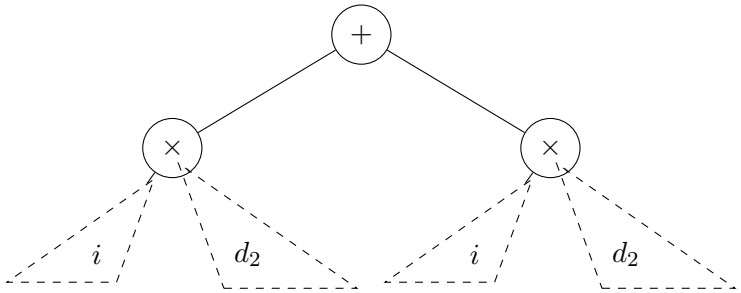


Figure 4.8: C'_{i+d_2} keeps only degree (i, d_2) type $\times$ gates.

Lemma 4.4.2. *For a degree d polynomial $f \in \mathbb{F}\{X\}$ given by a circuit C suppose $f = (g+\alpha)(h+\beta)$, where $g, h \in \mathbb{F}\{X\}$ such that $\deg(g) = \deg(h)$, and $\alpha, \beta \in \mathbb{F}$. Suppose $m = (m_1 m_2)$ is a nonzero degree d monomial. Then, in deterministic polynomial time we can compute circuits for the polynomials $c_{m_1}(g) \cdot h$ and $c_{m_2}(h) \cdot g$, where $c_{m_1}(g)$ and $c_{m_2}(h)$ are coefficient of m_1 and m_2 in g and h respectively.*

Proof. We can write $f = (g+\alpha)(h+\beta) = g \cdot h + \beta \cdot g + \alpha \cdot h + \alpha \cdot \beta$. Applying the PIT algorithm of Theorem 4.3.1 on f, we compute a maximum degree monomial $m = (m_1 m_2)$. Computing the left derivative of circuit C w.r.t. monomial m_1, after removing the outermost brackets, we obtain a circuit computing $c_{m_1}(g)h + \beta c_{m_1}(g) + \alpha c_{m_1}(h)$. Dropping the constant term, we obtain a circuit computing polynomial $c_{m_1}(g)h$. Similarly, computing the right derivative w.r.t m_2 yields a circuit for $c_{m_2}(h)g + \beta c_{m_2}(g) + \alpha c_{m_2}(h)$. Removing the constant term we get a circuit for $c_{m_2}(h)g$. ∎

When $\deg(g) > \deg(h)$ we can recover $h + \beta$ entirely (upto a scalar factor) and we need to obtain the homogeneous parts of g separately.

Lemma 4.4.3. *Let $f = (g + \alpha) \cdot (h + \beta)$ be a polynomial of degree d in $\mathbb{F}\{X\}$ given by a circuit C. Suppose $\deg(g) > \deg(h)$. Then, in deterministic polynomial time we can compute the circuit C' for $c_{m_1}(g)(h + \beta)$.*

Proof. Again, applying the PIT algorithm to f we obtain a nonzero degree d monomial $m = (m_1\ m_2)$ of f. If $f = (g+\alpha)(h+\beta)$ then $f = g \cdot h + \alpha h + \beta g + \alpha\beta$. As $\deg(g) > \deg(h)$, the left partial derivative of C with respect to m_1 yields a circuit C' for $c_{m_1}(g)\ (h + \beta)$. ∎

Extracting the homogeneous components from the circuit C' given by Lemma 4.4.3, yields

circuits for $\{c_{m_1}(g)h_i \ : \ i \in [d_2]\}$. We also get the constant term $c_{m_1}(g)\beta$. Now we obtain the homogeneous components of g as follows.

Lemma 4.4.4. *Suppose circuit C computes f, where $f = (g + \alpha)\ (h + \beta)$ of degree d, $\alpha, \beta \in \mathbb{F}$, $\deg(g) = d_1$ and $\deg(h) = d_2$ such that $d_1 > d_2$.*

- *Let m be a nonzero degree d monomial of f such that $m = (m_1\ m_2)$. Then circuits for $\{c_{m_2}(h)g_i : i \in [d_1 - d_2 + 1, d_1]\}$ can be computed in deterministic polynomial time.*
- *The $(d_2 + i)^{th}$ homogeneous part of f is given by $f_{d_2+i} = \sum_{j=0}^{d_2-1} g_{d_2+i-j}\ h_j + g_i\ h_{d_2}$ for $1 \leq i \leq d_1 - d_2$. From the circuit C_{d_2+i} of f_{d_2+i}, we can efficiently compute circuits for $\{c_{m_2}(h_{d_2})g_i : 1 \leq i \leq d_1 - d_2\}$.*

Proof. For the first part, fix any $i \in [d_1 - d_2 + 1, d_1]$, and compute the homogeneous $(i + d_2)^{th}$ part f_{i+d_2} of f by a circuit C_{i+d_2}. Similar to Lemma 4.4.1, we focus on the sub-circuits of C_{i+d_2} formed by $\times$ gate of the degree type (i, d_2). Since i is at least $d_1 - d_2 + 1$, such gates can compute the multiplication of a degree i polynomial with a degree d_2 polynomial. Then, by taking the right partial derivative with respect to m_2 we recover the circuits for $c_{m_2}(h_{d_2})\ g_i$ for any $i \in [d_1 - d_2 + 1, d_1]$.
Next, the goal is to recover the circuits for g_i (upto a scalar multiple), where $1 \leq i \leq d_1 - d_2$, and also recover the constant terms α and β. When $i \leq d_1 - d_2$ a product gate of type (i, d_2) can entirely come from g which requires a different handling.
We explain only the case when $i = d_1 - d_2$ (the others are similar). For $i = d_1 - d_2$, we have $f_{d_1} = \beta g_{d_1} + \sum_{j=1}^{d_2-1} g_{d_1-j}\ h_j + g_{d_1-d_2}\ h_{d_2}$. By Lemma 4.4.3, we can compute a circuit C' for $c_{m_1}(g)(h + \beta)$. Extracting the constant term yields $c_{m_1}(g)\beta$. From Lemma 4.4.4 we have a circuit C'' for $c_{m_2}(h)g_{d_1}$. Multiplying these circuits, we obtain a circuit C^* for $c_{m_2}(h)c_{m_1}(g)\beta g_{d_1}$. Since $c_{m_2}(h)c_{m_1}(g) = c_m(f)$, dividing C^* by $c_m(f)$ yields a circuit for βg_{d_1}. Note that, by the first part of this lemma, we already have circuits for every term g_{d_1-j} appearing in the above sum. Subtracting $\beta g_{d_1} + \sum_{j=1}^{d_2-1} g_{d_1-j}\ h_j$ from the circuit C_{d_1} for f_{d_1}, yields a circuit for polynomial $g_{d_1-d_2}h_{d_2}$. Computing the right derivative of the resulting circuit w.r.t m_2 (Lemma 4.2.1) yields a circuit for $c_{m_2}(h)g_{d_1-d_2}$.
For general $i \leq d_1 - d_2$, when we need to compute g_i, again we will have already computed circuits for all $g_j, j > i$. A suitable right derivative computation will yield a circuit for $c_{m_2}(h)g_i$. ∎

Lemmas 4.4.1, 4.4.2, 4.4.3, and 4.4.4 yield an efficient algorithm for computing circuits for the two factors $c_{m_2}(h)(\sum_{i=1}^{d_1} g_i)$ and $c_{m_1}(g)(\sum_{i=1}^{d_2} h_i)$ when $\deg(g) \geq \deg(h)$. The case when $\deg(g) < \deg(h)$ is similarly handled using left partial derivatives in the above lemmas.
Now we explain how to compute the constant terms of the individual factors. We discuss the case when $\alpha \neq 0$. The other case is similar.
First we recall that given a monomial m and a non-commutative circuit C, the coefficient of m in C can be computed in deterministic polynomial time [14]. We know that $f_0 = \alpha \cdot \beta$. We compute the coefficient of the monomial m_1 in the circuits for polynomials $c_{m_2}(h)c_{m_1}(g)gh$, $c_{m_2}(h)g$, and $c_{m_1}(g)h$. Let these coefficients be a, b and c, respectively. Moreover, we know that $c_{m_2}(h)c_{m_1}(g)$ is the coefficient of monomial $m = (m_1\ m_2)$ in f. Let the coefficient of m_1 in f be γ. Let $\gamma_1 = c_{m_1}(g)$ and $\gamma_2 = c_{m_2}(h)$ and $\delta = c_{m_1}(g)c_{m_2}(h)$.

Now equating the coefficient of m_1 from both side of the equation $f = (g+\alpha)(h+\beta)$ and substituting $\beta = \frac{f_0}{\alpha}$, we get

$$\gamma = \frac{a}{\gamma_1\gamma_2} + \frac{\alpha c}{\gamma_1} + \frac{f_0 b}{\alpha\gamma_2} = \frac{a}{\delta} + \frac{\alpha c}{\gamma_1} + \frac{f_0 b}{\alpha\gamma_2}.$$

Letting $\xi = \alpha\gamma_2$, this gives a quadratic equation in the unknown ξ.

$$c\xi^2 + (a - \gamma\delta)\xi + f_0 b\delta = 0.$$

By solving the above quadratic equation we get two solutions A_1 and A_2 for $\xi = \alpha\gamma_2$. Notice that $\beta\gamma_1 = \frac{\delta f_0}{\xi}$. As we have circuits for $c_{m_2}(h)g = \gamma_2 g$ and for $c_{m_1}(g)h = \gamma_1 h$, we obtain circuits for $\gamma_2(g+\alpha)$ and $\gamma_1(h+\beta)$ (two solutions, corresponding to A_1 and A_2). To pick the right solution, we can run the PIT algorithm to check if $\gamma_1\gamma_2 f$ equals the product of these two circuits that purportedly compute $\gamma_2(g+\alpha)$ and $\gamma_1(h+\beta)$.
Over $\mathbb{Q}$ we can just solve the quadratic equation in deterministic polynomial time using standard method. If $\mathbb{F} = \mathbb{F}_q$ for $q = p^r$, we can factorize the quadratic equation in deterministic time $\text{poly}(p, r)$ [109]. Using randomness, one can solve this problem in time $\text{poly}(\log p, r)$ using Berlekamp's factoring algorithm [19]. ∎

4.5 Conclusion

Motivated by the non-associative circuit lower bound result shown in [49], we study PIT and irreducibility testing in the free non-associative non-commutative ring $\mathbb{F}\{X\}$ and obtain efficient white-box algorithms for the problems.
Hrubeš, Wigderson, and Yehudayoff [49] have also shown exponential circuit-size lower bounds for non-associative, commutative circuits. It would be interesting to obtain an efficient polynomial identity testing algorithm for that circuit model too. Even a randomized polynomial-time algorithm is not known.
Obtaining an efficient *black-box* PIT in the ring $\mathbb{F}\{X\}$ is also an interesting problem. Of course, for such an algorithm the black-box can be evaluated on a suitable non-associative algebra. To the best of our knowledge, there seems to be no algorithmically useful analogue of the Amitsur-Levitzki theorem [8].

Chapter 5

Complexity of Univariate Ideal Membership with Applications

Let $\mathbb{F}[\mathrm{X}]$ be the polynomial ring over the variables $\mathrm{X} = \{x_1, x_2, \ldots, x_n\}$. An ideal $I = \langle p_1(x_1), \ldots, p_n(x_n)\rangle$ generated by univariate polynomials $\{p_i(x_i)\}_{i=1}^n$ is a *univariate ideal*. In this chapter we study various cases of the univariate ideal membership problem. First we address the case when $f(\mathrm{X}) \in \mathbb{F}[\ell_1, \ldots, \ell_r]$ is a (low rank) polynomial given by an arithmetic circuit, where $\ell_i : 1 \leq i \leq r$ are linear forms. We show that the (unique) remainder $f(\mathrm{X}) \pmod I$ can be evaluated at any given point $\vec{\alpha}$ in deterministic time $d^{O(r)} \cdot \mathrm{poly}(n)$, where $d = \max\{\deg(f), \deg(p_1) \ldots, \deg(p_n)\}$. This yields a randomized $n^{O(r)}$ algorithm for minimum vertex cover in graphs with rank-r adjacency matrices and it also yields an $n^{O(r)}$ algorithm for evaluating the permanent of a $n \times n$ matrix of rank r, over any field $\mathbb{F}$. Over $\mathbb{Q}$, an algorithm of similar run time for low rank permanent is due to Barvinok [17] via a different technique.
Next we investigate the univariate ideal membership problem from the perspective of parameterized complexity. Suppose the generators $p_i(x_i)$ of the univariate ideal I have distinct roots over $\mathbb{Q}$. Given a polynomial $f(X) \in \mathbb{C}[X]$ of degree k and I as input, we give algorithm to decide whether $f \in I$ or not in randomized $O^*(n^{k/2})$ time. On the other hand when $I = \langle x_1^{e_1}, \ldots, x_n^{e_n}\rangle$, we obtain a randomized $O^*(4.08^k)$ time and $\mathrm{poly}(n, k)$ space algorithm to test the membership $f \in I$. When $I = \langle p_1(x_1), p_2(x_2), \ldots, p_k(x_k)\rangle$ and number of generators k is the fixed parameter we show that checking $f \in I$ is W[2]-hard. Finally we show a complexity theoretic upper bound that complements a NP-hardness result of Alon and Tarsi [7]. Let $f \in \mathbb{Q}[\mathrm{X}]$ be a polynomial of degree at most d given by a black-box. Let $I = \langle p_1(x_1), \ldots, p_n(x_n)\rangle$ be an ideal given explicitly by a set of univariate polynomials $p_1, p_2, \ldots, p_n$ as generators of maximum degree bounded by d. Let L be the bit-size upper bound for any coefficient in $f, p_1, p_2, \ldots, p_n$. Moreover, assume that p_is have distinct roots over $\mathbb{C}$. Then there is a non-deterministic algorithm running in time $\mathrm{poly}(n, d, L)$ that decides the non-membership of f in the ideal I.

5.1 Introduction

Let $R = \mathbb{F}[x_1, x_2, \ldots, x_n]$ [1] be the ring of polynomials over the variables $\mathrm{X} = \{x_1, x_2, \ldots, x_n\}$. A subring $I \subseteq R$ is an ideal if $IR \subseteq I$. Computationally, an ideal I is often given by generators : $I = \langle f_1, f_2, \ldots, f_\ell \rangle$. Given $f \in R$ and $I = \langle f_1, \ldots, f_\ell \rangle$, the *Ideal Membership problem* is to decide whether $f \in I$ or not. In general, this is computationally highly intractable. In fact, it is EXPSPACE-complete even if f and the generators $f_i, i \in [\ell]$ are given explicitly by sum of monomials [67]. Nevertheless, special cases of ideal membership problem have played important roles in several results in arithmetic complexity. For example, the polynomial identity testing algorithm for depth three $\Sigma\Pi\Sigma$ circuits with bounded top fan-in; the structure theorem for $\Sigma\Pi\Sigma(k, d)$ identities use ideal membership very crucially [13, 57, 90].
In this chapter, our study of ideal membership is motivated by a basic algebraic result : the Combinatorial Nullstellensatz of Alon [6], and we recall a basic result in that paper.

Theorem 5.1.1. *Let $\mathbb{F}$ be any field, and $f(\mathrm{X}) \in \mathbb{F}[\mathrm{X}]$. Define polynomials $g_i(x_i) = \prod_{s \in S_i}(x_i - s)$ for non-empty subsets $S_i, 1 \leq i \leq n$ of $\mathbb{F}$. If f vanishes on all the common zeros of $g_1, \ldots, g_n$, then there are polynomials $h_1, \ldots, h_n$ satisfying $\deg(h_i) \leq \deg(f) - \deg(g_i)$ such that $f = \sum_{i=1}^{n} h_i g_i$.*

The theorem can be restated in terms of ideal membership: Let $f(\mathrm{X}) \in \mathbb{F}[\mathrm{X}]$ be a given polynomial, and $I = \langle g_1(x_1), g_2(x_2), \ldots, g_n(x_n) \rangle$ be an ideal generated by univariate polynomials g_i *without repeated roots*. Let $Z(g_i)$ denote the zero set of $g_i, 1 \leq i \leq n$. By Theorem 5.1.1, if $f \notin I$ then there is a $\vec{\alpha} = (\alpha_1, \ldots, \alpha_n) \in Z(g_1) \times \cdots \times Z(g_n)$ such that $f(\vec{\alpha}) \neq 0$. Of course, if $f \in I$ then $f|_{Z(g_1) \times \cdots \times Z(g_n)} = 0$.
Ideals I generated by univariate polynomials are called *univariate ideals*. For any univariate ideal I and any polynomial f, by repeated application of the division algorithm, we can write $f(\mathrm{X}) = \sum_{i=1}^{n} h_i(\mathrm{X}) g_i(x_i) + R(\mathrm{X})$ where R is unique and for each $i \in [n] : \deg_{x_i}(R) < \deg(g_i(x_i))$. Since the remainder is unique, it is convenient to write $R = f \pmod{I}$. By Alon's theorem, if $f \notin I$ then there is a $\vec{\alpha} \in Z(g_1) \times \cdots \times Z(g_n)$ such that $R(\vec{\alpha}) \neq 0$.
As an application of the theorem, Alon and Tarsi showed that checking k-colorability of a graph G is polynomial-time equivalent to testing whether the graph polynomial f_G is in the ideal $\langle x_1^k - 1, \ldots, x_n^k - 1 \rangle$ [6]. It follows that univariate ideal membership problem coNP-hard.
Univariate ideal membership is further motivated by its connection with two well-studied problems. Computing the permanent of a $n \times n$ matrix over any field $\mathbb{F}$ can be cast in terms of univariate ideal membership. Given a matrix $A = (a_{i,j})_{1 \leq i,j \leq n} \in \mathbb{F}^{n \times n}$, consider the product of linear forms $P_A(\mathrm{X}) = \prod_{i=1}^{n} (\sum_{j=1}^{n} a_{ij} x_j)$. The following observation is well known.

Fact 5.1.1. *The permanent of the matrix A is given by the coefficient of the monomial $x_1 x_2 \cdots x_n$ in P_A.*

It follows immediately that $P_A(\mathrm{X}) \pmod{\langle x_1^2, \ldots, x_n^2 \rangle} = \mathrm{Perm}(A)\ x_1 x_2 \cdots x_n$. Thus, the remainder $P_A \pmod{\langle x_1^2, \ldots, x_n^2 \rangle}$ evaluates to $\mathrm{Perm}(A)$ at the point $\vec{1} \in \mathbb{F}^n$.

[1] We often use the shorthand notation $\mathbb{F}[\mathrm{X}]$.

Next, we briefly mention the connection of univariate ideal membership with the multilinear monomial detection problem, a benchmark problem that is useful in designing fast parameterized algorithms for a host of problems [60–62].
Notice that, given an arithmetic circuit C computing a polynomial $f \in \mathbb{F}[\mathrm{X}]$ of degree k, checking if f has a non-zero multilinear monomial of degree k is equivalent to checking if $f \pmod{\langle x_1^2, \ldots, x_n^2 \rangle}$ is non-zero. Moreover, the constrained multilinear detection problem studied in [23, 61] can also be viewed as a problem of deciding membership in a univariate ideal.

Our Results : A contribution of this chapter is to consider several parameterized problems in arithmetic complexity as instances of univariate ideal membership. One parameter of interest is the rank of a multivariate polynomial: We say $f \in \mathbb{F}[\mathrm{X}]$ is a *rank* r polynomial if $f \in \mathbb{F}[\ell_1, \ell_2, \ldots, \ell_r]$ for linear forms $\ell_j : 1 \leq j \leq r$. This concept has found application in algorithms for depth-3 polynomial identity testing [90]. Given a univariate ideal I, a point $\vec{\alpha} \in \mathbb{F}^n$, and an arithmetic circuit computing a polynomial f of rank r, we obtain an efficient algorithm to compute $f \pmod{I}$ at $\vec{\alpha}$.

Theorem 5.1.2. *Let $\mathbb{F}$ be an arbitrary field where the field arithmetic can be done efficiently, and C be a polynomial-size arithmetic circuit computing a polynomial f in $\mathbb{F}[\ell_1, \ell_2, \ldots, \ell_r]$, where $\ell_1, \ell_2, \ldots, \ell_r$ are given linear forms in $\{x_1, x_2, \ldots, x_n\}$. Let $I = \langle p_1, \ldots, p_n \rangle$ be a univariate ideal generated by $p_i(x_i) \in \mathbb{F}[x_i], 1 \leq i \leq n$. Given $\vec{\alpha} \in \mathbb{F}^n$, we can evaluate the remainder $f \pmod{I}$ at the point $\vec{\alpha}$ in time $d^{O(r)}\mathrm{poly}(n)$, where $d = \max\{\deg(f), \deg(p_i) : 1 \leq i \leq n\}$.*

This also allows us to check whether $f \in I$ by picking a point $\vec{\alpha}$ at random and checking whether $f \pmod{I}$ evaluated at $\vec{\alpha}$ is zero or not. The intuitive idea behind the proof of Theorem 5.1.2 is as follows. Given a polynomial $f(\mathrm{X}) \in \mathbb{F}[\ell_1, \ldots, \ell_r]$, a univariate ideal $I = \langle p_1(x_1), \ldots, p_n(x_n) \rangle$, and a point $\vec{\alpha} \in \mathbb{F}^n$, we first find an invertible linear transformation T such that the polynomial $T(f)$ becomes a polynomial over at most $2r$ variables. Additionally T has the property that T fixes the variables $x_1, \ldots, x_r$. Then we recover the polynomial (call it $\tilde{f}$) over at most $2r$ variables explicitly and perform division algorithm with respect to the ideal $I_{[r]} = \langle p_1(x_1), \ldots, p_r(x_r) \rangle$. For notational convenience, call $\tilde{f}$ be the polynomial obtained over at most $2r$ variables. It turns out $T^{-1}(\tilde{f})$ is the *true remainder* $f \pmod{I_{[r]}}$. Since the variables $x_1, \ldots, x_r$ do not play role in the subsequent stages of division, we can eliminate them by substituting $x_i \leftarrow \alpha_i$ for each $1 \leq i \leq r$. Then we apply the division algorithm on $T^{-1}(\tilde{f})|_{x_i \leftarrow \alpha_i : 1 \leq i \leq r}$ recursively with respect to the ideal $I_{[n]\setminus[r]}$ to compute the final remainder at the point $\vec{\alpha}$.
Our next result is an efficient algorithm to detect vertex cover in low rank graphs. A graph G is said to be of rank r if the rank of the adjacency matrix A_G is of rank r. Graphs of low rank were studied by Lovasz and Kotlov [9, 10] in the context of graph coloring. Our idea is to construct a low rank polynomial from the graph and check its membership in an appropriate univariate ideal.

Theorem 5.1.3. *Given a graph $G = (V, E)$ on n vertices such that the rank of the adjacency matrix A_G is at most r, and a parameter k, there is a randomized $n^{O(r)}$ algorithm to decide if the graph G has vertex cover of size k or not.*

Theorem 5.1.2 also yields an $n^{O(r)}$ algorithm to compute the permanent of rank-r matrices over any field. Barvinok had given [17] an algorithm of same running time for the permanent

of low rank matrices (over $\mathbb{Q}$) using apolar bilinear forms. By Fact 5.1.1, if matrix A is rank r then P_A is a rank-r polynomial, and for the univariate ideal $I = \langle x_1^2, \ldots, x_n^2 \rangle$ computing $P_A \pmod I$ at the point $\vec{1}$ yields the permanent. Theorem 5.1.2 works more generally for all univariate ideals. In particular, the ideal in the proof of Theorem 5.1.3 is generated by polynomials that are not powers of variables. Thus, Theorem 5.1.2 can potentially have more algorithmic consequences than the technique in [17].
Next we consider the degree of the input polynomial to be the parameter. When the generators of the univariate ideal have distinct roots we obtain an algorithm that improves over the brute-force running time.

Theorem 5.1.4. *Let $I = \langle p_1(x_1), \ldots, p_n(x_n) \rangle$ be a univariate ideal given explicitly by a set of univariate polynomials $p_1, \ldots, p_n$ such that for each $i \in [n]$, $p_i(x_i)$ has distinct roots over $\mathbb{Q}$. Given a polynomial $f(X) \in \mathbb{C}[X]$ of degree k and I as input, we can decide whether $f \in I$ or not in randomized $O^*(n^{k/2})$ time.*

If the univariate ideal is generated by the powers of variables, we have a randomized FPT algorithm for the ideal membership problem.

Theorem 5.1.5. *Given an arithmetic circuit C computing a polynomial $f(\mathrm{X}) \in \mathbb{Z}[\mathrm{X}]$ of degree k and integers $e_1, e_2, \ldots, e_n$, there is a randomized algorithm to decide whether $f \in \langle x_1^{e_1}, x_2^{e_2}, \ldots, x_n^{e_n} \rangle$ in $4.08^k \cdot \mathrm{poly}(nsd(\sum_{i=1}^n e_i))$ time.*

Note that this generalizes the well-known problem of *multilinear monomial detection* for which the ideal of interest would be $I = \langle x_1^2, x_2^2, \ldots, x_n^2 \rangle$. Surprisingly, the run time of the algorithm in Theorem 5.1.5 is independent of the e_i. Brand et al. have given the first FPT algorithm for multilinear monomial detection in the case of general circuit with run time randomized $O^*(4.32^k)$ [25]. Recently, this problem has also been studied using the Hadamard product technique [11] and also by another technique using apolar bilinear forms [75]). When the number of generators in the ideal is treated as the fixed parameter, the problem is W[2]-hard.

Theorem 5.1.6. *Given a polynomial $f(\mathrm{X}) \in \mathbb{F}[\mathrm{X}]$ by an arithmetic circuit C and univariate polynomials $p_1(x_1), p_2(x_2), \ldots, p_k(x_k)$, checking if $f \notin \langle p_1(x_1), p_2(x_2), \ldots, p_k(x_k) \rangle$ is W[2]-hard with k as the parameter.*

Theorem 5.1.6 is shown by a suitable reduction from k-dominating set problem to ideal membership. To find a dominating set of size k, the reduction produces an ideal with k univariates and the polynomial created from the graph has k variables.
As already mentioned, the result of Alon and Tarsi [6] shows that the membership of f_G in $\langle x_1^k - 1, \ldots, x_n^k - 1 \rangle$ is coNP-hard and the proof crucially uses the fact that the roots of the generator polynomials are all distinct. This naturally raises the question if univariate ideal membership is in coNP when each generator polynomial has distinct roots. We affirmatively answer this question.

Theorem 5.1.7. *Let $f \in \mathbb{Q}[\mathrm{X}]$ be a polynomial of degree at most d given by a black-box. Let $I = \langle p_1(x_1), \ldots, p_n(x_n) \rangle$ be an ideal given explicitly by a set of univariate polynomials $p_1, p_2, \ldots, p_n$ as generators of maximum degree bounded by d. Let L be the bit-size upper bound for any coefficient in $f, p_1, p_2, \ldots, p_n$. Moreover, assume that p_is have distinct roots over $\mathbb{C}$. Then there is a non-deterministic algorithm running in time $\mathrm{poly}(n, d, L)$ that decides the non-membership of f in the ideal I.*

Remark 5.1.1. *The distinct roots case discussed in Theorem 5.1.7 is in stark contrast to the complexity of testing membership of $P_A(\mathrm{X})$ in the ideal $\langle x_1^2, \ldots, x_n^2 \rangle$. That problem is equivalent to checking if Perm(A) is nonzero for a rational matrix A, which is hard for the exact counting class $C_=P$. Hence it cannot be in* coNP *unless the polynomial-time hierarchy collapses.*

Recall from Alon's Nullstellensatz that if $f \notin I$, then there is always a point $\vec{\alpha} \in Z(p_1) \times \ldots \times Z(p_n)$ such that $f(\vec{\alpha}) \neq 0$. Notice that in general the roots $\alpha_i \in \mathbb{C}$ and in the standard *Turing Machine* model the NP machine can not guess the roots directly with only finite precision. But we are able to prove that the NP machine can guess the tuple of roots $\vec{\alpha} \in \mathbb{Q}^n$ using only polynomial bits of precision and still can decide the non-membership. The main technical idea is to compute efficiently a parameter M only from the input parameters such that $|f(\vec{\alpha})| \leq M$ if $f \in I$, and $|f(\vec{\alpha})| \geq 2M$ if $f \notin I$. The NP machine decides the non-membership according to the final value of $|f(\vec{\alpha})|$. We remark that Koiran has considered the weak version of Hilbert Nullstellensatz (HN) problem [58]. The input is a set of multivariate polynomials $f_1, f_2, \ldots, f_m \in \mathbb{Z}[\mathrm{X}]$ and the problem is to decide whether $1 \in \langle f_1, \ldots, f_m \rangle$. The result of Koiran shows that $\overline{\mathrm{HN}} \in \mathrm{AM}$ (under GRH), and it is an outstanding open problem problem to decide whether $\overline{\mathrm{HN}} \in \mathrm{NP}$.

5.2 Preliminaries

Basics of Ideal Membership

Let $\mathbb{F}[\mathrm{X}]$ be the ring of polynomials $\mathbb{F}[x_1, x_2, \ldots, x_n]$. Let $I \subseteq \mathbb{F}[\mathrm{X}]$ be an ideal given by a set of generators $I = \langle g_1, \ldots, g_\ell \rangle$. Then for any polynomial $f \in \mathbb{F}[\mathrm{X}]$, it is a member of the ideal if and only if $f = \sum_{i=1}^{\ell} h_i g_i$ where $\forall i : h_i \in \mathbb{F}[\mathrm{X}]$. Dividing f by the g_i by applying the standard division algorithm does not work in general to check if $f \in I$. Indeed, the remainder is not even uniquely defined. However, if the leading monomials(LM) of the generators are already pairwise relatively prime, then we can apply the division algorithm to compute the unique remainder.

Theorem 5.2.1 (See [32], Theorem 3, proposition 4, pp.101)**.** *Let I be a polynomial ideal given by a basis $G = \{g_1, g_2, \cdots, g_s\}$ such that all pairs $i \neq j$ $LM(g_i)$ and $LM(g_j)$ are relatively prime. Then G is a Gröbner basis for I.*

In particular, if the ideal I is a univariate ideal given by $I = \langle p_1(x_1), \ldots, p_n(x_n) \rangle$, we can apply the division algorithm to compute the unique remainder $f \pmod I$. To bound the run time of this procedure we note the following: Let $\bar{p}$ denote the ordered list $\{p_1, p_2, \ldots, p_n\}$. Let Divide$(f; \bar{p})$ be the procedure that divides f by p_1 to obtain remainder f_1, then divides f_1 by p_2 to obtain remainder f_2, and so on to obtain the final remainder f_n after dividing by p_n. We note the following time bound for Divide$(f; \bar{p})$.

Fact 5.2.1 (See [99], Section 6, pp.5-12)**.** *Let $f \in \mathbb{F}[\mathrm{X}]$ be given by a size s arithmetic circuit and $p_i(x_i) \in \mathbb{F}[x_i]$ be given univariate polynomials. The running time of Divide$(f; \bar{p})$ is bounded by $O(s \cdot \prod_{i=1}^{n} (d_i + 1)^{O(1)})$, where $d_i = \max\{\deg_{x_i}(f), \deg(p_i(x_i))\}$.*

On Roots of Univariate Polynomials

The following lemma shows that the absolute value of any root of a univariate polynomial can be bounded in terms of the degree and the coefficients. The result is folklore.

Lemma 5.2.1. *Let $f(x) = \sum_{i=0}^{d} a_i x^i \in \mathbb{Q}[x]$ be a univariate polynomial and α be a root of f. Then, either $\frac{|a_0|}{\sum_{i=1}^{d} |a_i|} \leq |\alpha| < 1$ or $1 \leq |\alpha| \leq d \cdot \frac{\max_i |a_i|}{|a_d|}$.*

Proof. Since α is a root of f, we have that, $0 = f(\alpha) = \sum_{i=0}^{d} a_i \alpha^i = 0$, and $\sum_{i=1}^{d} a_i \alpha^i = -a_0$. Then by an application of triangle inequality, we get that $\sum_{i=1}^{d} |a_i||\alpha|^i \geq |a_0|$. Now we analyse two different cases. In the first case assume that $|\alpha| < 1$. Observe that $|\alpha| \cdot (\sum_{i=1}^{d} |a_i|) \geq |a_0|$, and hence $|\alpha| \geq \frac{|a_0|}{\sum_{i=1}^{d} |a_i|}$. In the second case $|\alpha| \geq 1$. Observe that $-a_d \alpha^d = \sum_{i=0}^{d-1} a_i \alpha^i$. Then use triangle inequality to get that $|a_d||\alpha|^d \leq |\alpha|^{d-1} \cdot (\sum_{i=0}^{d-1} |a_i|)$. Now we get the following, $|\alpha| \leq \frac{\sum_{i=0}^{d-1} |a_i|}{|a_d|} \leq d \cdot \frac{\max_i |a_i|}{|a_d|}$. The lemma follows by combining the two cases. ∎

The next lemma shows that the separation between two distinct roots of any univariate polynomial can be lower bounded in terms of degree and the size of the coefficients. This was shown by Mahler [66].

Lemma 5.2.2. *Let $g(x) = \sum_{i=0}^{d} a_i x^i \in \mathbb{Q}[x]$ and $2^{-L} \leq |a_i| \leq 2^L$ (if $a_i \neq 0$). Let α, β are two distinct roots of g. Then $|\alpha - \beta| \geq \frac{1}{2^{O(dL)}}$.*

The following lemma states that any univariate polynomial can not get a very small value (in absolute sense) on any point which is far from every root.

Lemma 5.2.3. *Let $f = \sum_{i=1}^{d} a_i x^i$ be a univariate polynomial with $2^{-L} \leq |a_i| \leq 2^L$ (if $a_i \neq 0$). Let $\tilde{\alpha}$ be a point such that $|\tilde{\alpha} - \beta_i| \geq \delta$ for every root β_i of f then $|f(\tilde{\alpha})| \geq 2^{-L}\delta^d$.*

Proof. We observe that, $f(\tilde{\alpha}) = c \prod_{i=1}^{d} (\tilde{\alpha} - \beta_i)$. Since $|\tilde{\alpha} - \beta_i| \geq \delta$ we get, $|f(\tilde{\alpha})| = |c| \prod_{i=1}^{d} |\tilde{\alpha} - \beta_i| \geq 2^{-L}\delta^d$. This completes the proof. ∎

Hadamard Product

We recall the definition of Hadamard product of two polynomials.

Definition 5.2.1. *Given two polynomials $f, g \in \mathbb{F}[\mathrm{X}]$, the Hadamard product $f \circ g$ is defined as $f \circ g = \sum_m [m]f \cdot [m]g \cdot m$.*

In this chapter we adapt the notion of Hadamard product suitably and define a scaled version of Hadamard Product of two polynomials.

Definition 5.2.2. *Given two polynomials $f, g \in \mathbb{F}[X]$, their* scaled Hadamard Product *$f \circ^s g$, is defined as $f \circ^s g = \sum_m m! \cdot [m]f \cdot [m]g \cdot m$, where $m = x_{i_1}^{e_1} x_{i_2}^{e_2} \dots x_{i_r}^{e_r}$ and $m! = e_1! \cdot e_2! \cdots e_r!$ abusing the notation.*

Remark 5.2.1. *Given two polynomials $f \in \mathbb{F}[X]$ and $g \in \mathbb{F}[X]$, if one of these two is a multilinear polynomial then scaled Hadamard product $f \circ^s g$ is same as Hadamard product $f \circ g$.*

Symmetric Polynomial and Weakly Equivalence of Polynomials

The symmetric polynomial of degree k over n variables $\{x_1, x_2, \dots, x_n\}$, denoted by $S_{n,k}$, is defined as follows:

$$S_{n,k}(x_1, x_2, \dots, x_n) = \sum_{T \subseteq [n], |T|=k} \prod_{i \in T} x_i.$$

Notice that, $S_{n,k}$ contains all the degree k multillinear terms. A recent result of Lee gives the following homogeneous diagonal circuit for $S_{n,k}$ [64].

Lemma 5.2.4. *The symmetric polynomial $S_{n,k}$ can be computed by a homogenous $\Sigma^{[s]}\wedge^{[k]}\Sigma$ circuit where $s \leq \sum_{i=0}^{k/2} \binom{n}{i}$.*

A polynomial $f \in \mathbb{F}[X]$ is said to be *weakly equivalent* to a polynomial $g \in \mathbb{F}[X]$, if the following is true. For each monomial m, $[m]f = 0$ if and only if $[m]g = 0$. In this chapter, we will use polynomials which are weakly equivalent to $S_{n,k}$.

Parameterized Complexity Classes

We recall some standard definitions in parameterized Complexity [33, ch.1,pp. 7-14]. We only state them informally. For a parameterized input problem (x, k) with k be the parameter of interest, we say that the problem is in FPT if it has an algorithm with run time $f(k)|(x,k)|^{O(1)}$ for some computable function f. A parameterized reduction [33, def. 13.1] between two problems should be computable in time $f(k)|(x,k)|^{O(1)}$, and if the reduction outputs (x', k') then $k' \leq f(k)$.

5.3 Ideal Membership for Low Rank Polynomials

In this section we prove Theorem 5.1.2. Given a r-rank polynomial f by an arithmetic circuit, a univariate ideal I, and a point $\vec{\alpha} \in \mathbb{F}^n$, we give an $d^{O(r)}$ time algorithm to evaluate the remainder polynomial $f \pmod I$ at $\vec{\alpha}$ where d is the degree of the polynomial f. As mentioned in Section 5.1, an application of our result yields an $n^{O(r)}$ time algorithm for computing the permanent of rank-r matrices over any field. Barvinok [17], via a different method, had obtained an $n^{O(r)}$ time algorithm for this problem over $\mathbb{Q}$. We also obtain a randomized $n^{O(r)}$ time algorithm for minimum vertex cover of low rank graphs. We first define the notion *rank* of a polynomial in $\mathbb{F}$[X].

Definition 5.3.1. *A polynomial $f(\mathrm{X}) \in \mathbb{F}[\mathrm{X}]$ is a* rank-r *polynomial if there are linear forms $\ell_1, \ell_2, \ldots, \ell_r$ such that $f(\mathrm{X})$ is in the sub-algebra $\mathbb{F}[\ell_1, \ldots, \ell_r]$.*

For an unspecified fixed parameter r, we refer to rank-r polynomials as *low rank polynomials*.

Given $\vec{\alpha} \in \mathbb{F}^n$, a univariate ideal $I = \langle p_1(x_1), \ldots, p_n(x_n)\rangle$, and a rank r polynomial $f(\ell_1, \ldots, \ell_r)$ we show how to compute $f(\ell_1, \ldots, \ell_r) \pmod I$ at $\vec{\alpha}$ using a recursive procedure $\mathrm{REM}(f(\ell_1, \ldots, \ell_r), I, \vec{\alpha})$ efficiently. We introduce the following notation. For $S \subseteq [n]$, the ideal $I_S = \langle p_i(x_i) : i \in [S]\rangle$.

We first observe the following lemma which shows how to remove the redundant variables from a low rank polynomial.

Lemma 5.3.1. *Given a polynomial $f(\ell_1, \ldots, \ell_r)$ where $\ell_1, \ldots, \ell_r$ are linear forms in $\mathbb{F}$[X], there is an invertible linear transform $T : \mathbb{F}^n \mapsto \mathbb{F}^n$ that fixes $x_1, \ldots, x_r$ and the transformed polynomial $T(f)$ is over at most $2r$ variables.*

Proof. Write each linear form ℓ_i in two parts: $\ell_i = \ell_{i,1} + \ell_{i,2}$, where $\ell_{i,1}$ is the part over variables $x_1, \ldots, x_r$ and $\ell_{i,2}$ is over variables $x_{r+1}, \ldots, x_n$. W.l.o.g, assume that $\{\ell_{i,2}\}_{i=1}^{r'}$ is a maximum linearly independent subset of linear forms in $\{\ell_{i,2}\}_{i=1}^{r}$. Let $T : \mathbb{F}^n \to \mathbb{F}^n$ be the invertible linear map that fixes $x_1, \ldots, x_r$, maps the independent linear forms $\{\ell_{i,2}\}_{i=1}^{r'}$ to variables $x_{r+1}, \ldots, x_{r+r'}$, and suitably extends T to an invertible map. This completes the proof. ∎

The following lemma shows that the univariate division and evaluating the remainder at the end can be achieved by division and evaluation partially.

Lemma 5.3.2. *Let $f(\mathrm{X}) \in \mathbb{F}[\mathrm{X}]$ and $I = \langle p_1(x_1), \ldots, p_n(x_n)\rangle$ be a univariate ideal. Let $R(\mathrm{X})$ be the unique remainder $f \pmod I$. Let $\vec{\alpha} \in \mathbb{F}^r, r \leq n$ and $R_r(\mathrm{X}) = f \pmod{I_{[r]}}$. Then $R(\alpha_1, \ldots, \alpha_r, x_{r+1}, \ldots, x_n) = R_r(\alpha_1, \ldots, \alpha_r, x_{r+1}, \ldots, x_n) \pmod{I_{[n]\setminus[r]}}$.*

Proof. From the uniqueness of the remainder for the univariate ideals, we get that $R(\mathrm{X}) = R_r(\mathrm{X}) \pmod{I_{[n]\setminus[r]}}$. Now we write explicitly the polynomial $R_r(\mathrm{X})$ as $R_r = \sum_{\bar{u}} r_{\bar{u}} \cdot x_{r+1}^{u_1} \ldots, x_n^{u_{n-r}}$ where $r_u \in \mathbb{F}[\mathrm{X}_{[r]}]$. So we get that,

$$R_r \pmod{I_{[n]\setminus[r]}} = \sum_{\bar{u}} r_{\bar{u}} \prod_{j=1}^{n-r} q(x_{r+j})$$

where $q(x_{r+j}) = x_{r+j}^{u_j} \pmod{p(x_{r+j})}$. Then the lemma follows by substituting $x_1 = \alpha_1, \ldots, x_r = \alpha_r$ in the relation $R = R_r \pmod{I_{[n]\setminus[r]}}$. ∎

We require the following lemma in the proof of the main result of this section.

Lemma 5.3.3. *Let $f \in \mathbb{F}[\mathrm{X}]$, and $T : \mathbb{F}^n \to \mathbb{F}^n$ be an invertible linear transformation fixing $x_1, \ldots, x_r$ and mapping $x_{r+1}, \ldots, x_n$ to linearly independent linear forms over $x_{r+1}, \ldots, x_n$. Write $R = f \pmod{I_{[r]}}$ and $R' = T(f) \pmod{I_{[r]}}$. Then $R' = T(R)$.*

Proof. Let $f = \sum_{i=1}^{r} h_i(\mathrm{X}) \cdot p_i(x_i) + R(\mathrm{X})$ and $T(f) = \sum_{i=1}^{r} h'_i(\mathrm{X}) \cdot p_i(x_i) + R'(\mathrm{X})$. Note that $\deg_{x_i} R, \deg_{x_i} R' < \deg(p_i(x_i))$ for $1 \leq i \leq r$. Since T is invertible and also fixes $x_1, \ldots, x_r$, we can write $f = \sum_{i=1}^{r} T^{-1}(h'_i(\mathrm{X})) \cdot p_i(x_i) + T^{-1}(R'(\mathrm{X}))$. By the property of T it is clear that $\deg_{x_i}(T^{-1}(R'(\mathrm{X}))) < \deg(p_i(x_i))$ for $1 \leq i \leq r$. Combining two expression for f, we immediately conclude that $(R - T^{-1}(R')) = 0 \pmod{I_{[r]}}$ which forces that $R = T^{-1}(R')$. ∎

5.3.1 Proof of Theorem 5.1.2

Proof. We now describe a recursive procedure REM to solve the problem. The initial call to it is $\mathrm{REM}(f(\ell_1, \ldots, \ell_r), I_{[n]}, \vec{\alpha})$. We apply the invertible linear transformation obtained in Lemma 5.3.1 to get the polynomial $T(f)$ over the variables $x_1, \ldots, x_r, x_{r+1}, \ldots, x_{r+r'}$ where $r' \leq r$.[2] The polynomial $T(f)$ can be explicitly computed in time $\mathrm{poly}(L, s, n, d^{O(r)})$. Then we compute the remainder polynomial $f'(x_1, \ldots, x_{r+r'}) = T(f) \pmod{I_{[r]}}$ by applying the division algorithm which runs in time $\mathrm{poly}(L, s, n, d^{O(r)})$. Next we compute the polynomial $g = f'(\alpha_1, \ldots, \alpha_r, x_{r+1}, \ldots, x_{r+r'})$. Notice from Lemma 5.3.1 that $T^{-1}(x_{r+i}) = \ell_{i,2}$ for $1 \leq i \leq r'$, thus we are interested in the polynomial $g(\ell_{1,2}, \ldots, \ell_{r',2})$. Now we recursively compute $\mathrm{REM}(g(\ell_{1,2}, \ldots, \ell_{r',2}), I_{[n]\setminus[r]}, \vec{\alpha}')$ where $\vec{\alpha}' = (\alpha_{r+1}, \ldots, \alpha_n)$.

Correctness of the algorithm : Let $R(\mathrm{X}) = f \pmod{I_{[n]}}$ be the unique remainder polynomial. Let $R_r(\mathrm{X}) = f \pmod{I_{[r]}}$ and we know that $R_r \pmod{I_{[n]\setminus[r]}} = R$. So by Lemma 5.3.2, to show the correctness of the algorithm, it is enough to show that $g(\ell_{1,2}, \ldots, \ell_{r',2}) = R_r(\alpha_1, \ldots, \alpha_r, x_{r+1}, \ldots, x_n)$.

[2] We use f to denote $f(\ell_1, \ldots, \ell_r)$.

Following Lemma 5.3.3, write $R' = f'(x_1, \ldots, x_r, x_{r+1}, \ldots, x_n) = T(f) \pmod{I_{[r]}}$. Then, by Lemma 5.3.3 we conclude that $R' = T(R_r)$. It immediately follows that $R_r = T^{-1}(R') = f'(x_1, \ldots, x_r, T^{-1}(x_{r+1}), \ldots, T^{-1}(x_n))$. Now by definition the polynomial $g(\ell_{1,2}, \ldots, \ell_{r',2})$ is $f'(\alpha_1, \ldots, \alpha_r, T^{-1}(x_{r+1}), \ldots, T^{-1}(x_{r+r'}))$ which is simply $R_r(\alpha_1, \ldots, \alpha_r, x_{r+1}, \ldots, x_n)$.

Time complexity : First, suppose that the field arithmetic over $\mathbb{F}$ can be implemented using polynomial bits and L be the bit-size upper bound for any coefficient in $f, p_1, \ldots, p_n$. This covers all the finite fields where the field is given by an explicit irreducible polynomial. Also, over any such field the polynomial $T(f)$ can be explicitly computed from the input arithmetic circuit deterministically in time $\text{poly}(L, s, n, d^{O(r)})$.
Notice that in each recursive application the number of generators in the ideal is reduced by at least one. Furthermore, in each recursive step we need time $\text{poly}(L, s, n, d^{O(r)})$ to run the division algorithm. This gives us a recurrence of $t(n) \leq t(n-1) + \text{poly}(L, s, n, d^{O(r)})$ which solves to $t(n) \leq \text{poly}(L, s, n, d^{O(r)})$.

Bit-size growth over $\mathbb{Q}$: Over $\mathbb{Q}$, we only need to argue that the intermediate bit-size complexity growth is only polynomial in the input size.Let $\tilde{L}$ be the maximum bit size of any coefficient appearing in $f(z_1, \ldots, z_r)$, and let L be an upper bound on the bit sizes of the other inputs, i.e. bit sizes of coefficients of $\ell_1, \ldots, \ell_r, p_1, \ldots, p_n$ and $\alpha_1, \ldots, \alpha_n$. We will show that the circuit that we use in the next recursive step has coefficients of bit size at most $\tilde{L} + \text{poly}(n, d, L)$.
Let $|c(h)|$ denote the maximum coefficient (in absolute value) appearing in any polynomial h. Then by direct expansion we can see that $|c(f(\ell_1, \ldots, \ell_r))| \leq 2^{\tilde{L}+\text{poly}(n,d,L)}$. Also the linear transformation from lemma 5.3.1 can be implemented using poly-bit size entries. Together, we get that that $c(T(f(\ell_1, \ldots, \ell_r)) \leq 2^{\tilde{L}+\text{poly}(n,d,L)}$. At this point, we expand the circuit and obtain $T(f)$ explicitly as a sum of $d^{O(r)}$ monomials. Then divide $T(f)$ by $p_1(x_1), \ldots, p_r(x_r)$ one-by-one, and substitute $x_1 = \alpha_1, \ldots, x_r = \alpha_r$ giving us the remainder $g(x_{r+1}, \ldots, x_{r+r'})$. We note that $|c(g)| \leq 2^{\tilde{L}+\text{poly}(n,d,L)}$ [3]. Now the algorithm passes the $d^{O(r)}$ size $\Sigma\Pi\Sigma$ circuit $g(\ell_{1,2}, \ldots, \ell_{r',2})$ (We note that $T^{-1}(x_{r+1}) = \ell_{1,2}, \ldots, T^{-1}(x_{r+r'}) = \ell_{r',2}$), univariates $p_{r+1}(x_{r+1}), \ldots, p_n(x_n)$ and the point $(\alpha_{r+1}, \ldots, \alpha_n)$ for the next recursive call. We note that the bit-size upper bound L does not change for the input linear forms, and the coefficient bit-size of f grows from $\tilde{L}$ to $\tilde{L} + \text{poly}(n, d, L)$ in one step of the recursion. This gives us the recurrence $S(n) \leq S(n-1) + \text{poly}(n, d, L)$ with $S(1) = \tilde{L}$, which solves to $S(n) = O(\tilde{L} + \text{poly}(n, d, L))$. ∎

Remark 5.3.1. *Given a rank r polynomial $f(\ell_1, \ldots, \ell_r)$ and a univariate ideal $I = \langle p_1(x_1), \ldots, p_n(x_n) \rangle$, we can decide the membership of f in I by testing identity of $f \pmod{I}$ i.e. by evaluating $f \pmod{I}$ at some $\alpha \in \mathbb{F}^n$ chosen randomly [34, 93, 116]. Hence, the membership can be decided in randomized $d^{O(r)} \cdot \text{poly}(n)$ time where $d = \max\{\deg(f), \deg(p_i) : 1 \leq i \leq n\}$ using Theorem 5.1.2.*

5.3.2 Small Circuit for the Remainder Polynomial

The first algorithm is based on repeated division and partial evaluation. As such, it does not directly yield a small circuit for $f \pmod{I}$.

[3] We tackle a similar situation in Section 5.6, and Lemma 5.6.1 gives further explanation on the bit-complexity growth when we divide by univariate polynomials.

We now show that $f \pmod I$ has an arithmetic circuit of size $O^*(d^{O(r)})$, where $d = \deg(f)$. The circuit has a nice form: it is a $d^{O(r)}$-sum of products of univariate polynomials, each of degree at most d. Moreover, this circuit can be constructed in time $O^*(d^{O(r)})$ from the input f and I. This also yields another proof of Theorem 5.1.2, since evaluation of the circuit obtained at a given scalar point can be done in $O^*(d^{O(r)})$ time.
Some notation for the sequel: For $q \in \mathbb{F}[t_1, t_2, \ldots, t_r, X]$, let $[t_1^{d_1} t_2^{d_2} \cdots t_r^{d_r}](q)$ denote the coefficient of $t_1^{d_1} t_2^{d_2} \cdots t_r^{d_r}$ in q, noting that $[t_1^{d_1} t_2^{d_2} \cdots t_r^{d_r}](f) \in \mathbb{F}[X]$.
Now, we can write $f = g(\ell_1, \ell_2, \ldots, \ell_r)$ as a sum of $d^{O(r)}$ d-products of the r linear forms. Thus, it suffices to give a small circuit, of the above form, for a remainder $\ell_1^{d_1} \ell_2^{d_2} \cdots \ell_r^{d_r} \pmod I$, where $I = \langle p_1(x_1), p_2(x_2), \ldots, p_n(x_n) \rangle$. A +-gate summing up all these remainder circuits would be a circuit of the claimed form for $f \pmod I$.
We first consider a single power $\ell^d \pmod I$, where $\ell = \sum_{i=1}^n a_i x_i$ is a homogeneous linear form in $\mathbb{F}[X]$. By the multinomial theorem

$$\left(\sum_{i=1}^n a_i x_i t \right)^d = \sum_{j_1 + j_2 + \cdots + j_n = d} \binom{d}{j_1, j_2, \ldots, j_n} \prod_{i=1}^n (a_i x_i t)^{j_i}.$$

For fields $\mathbb{F}$ of characteristic zero, we can write:

$$\left(\sum_{i=1}^n a_i x_i \right)^d = d! [t^d] \left(\prod_{i=1}^n \left(\sum_{j=0}^d \frac{1}{j!} (a_i x_i t)^j \right) \right). \tag{5.1}$$

Equation 5.1 is combinatorially verified by noting that the term $\prod_{i=1}^n (a_i x_i t)^{j_i}$, for $j_1 + j_2 + \ldots j_n = d$ occurs precisely $\binom{d}{j_1, j_2, \ldots, j_n}$ times on the right side, matching the multinomial expansion of the left side. This identity was first used in arithmetic circuit complexity by Saxena [87][4] and has found many applications.

Remark 5.3.2. *Observe that, the right hand side expression of Equation 5.1 can be viewed as a univariate polynomial in t of degree nd. Therefore, by interpolation, we can find $\alpha_1, \ldots, \alpha_{nd+1} \in \mathbb{F}$ (or a suitable extension field of $\mathbb{F}$) and $\beta_1, \ldots, \beta_{nd+1} \in \mathbb{F}$ such that,*

$$\left(\sum_{i=1}^n a_i x_i \right)^d = \sum_{\ell=1}^{nd+1} \beta_\ell \left(\prod_{i=1}^n \left(\sum_{j=0}^d \frac{1}{j!} (a_i x_i \alpha_\ell)^j \right) \right). \tag{5.2}$$

Therefore, a power of a linear form can be expressed as a small sum of product of univariates.

This has been generalized to the finite fields setting [39]. We give a self-contained description of this, as it is required for the circuit construction for $f \pmod I$. First, for $\mathrm{Char}(\mathbb{F}) = p$, Equation 5.1 only holds for $d < p$, as each $k!$ occurring in it is invertible in $\mathbb{F}_p$ precisely if $d < p$. To obtain a suitable form of the equation for $d \geq p$, we first write $d = \sum_{k=0}^s e_k p^k$, for $s \leq \log_p(d) - 1$ and each $e_k < p$. Since $\mathrm{Char}(\mathbb{F}) = p$ for each $k \leq s$, using $a_i^p = a_i$ we have:

$$\left(\sum_{i=1}^n a_i x_i t \right)^{e_k p^k} = \left(\sum_{i=1}^n a_i x_i^{p^k} t^{p^k} \right)^{e_k}.$$

[4]The idea of using the analytic identity $e^{\sum_i y_i} = \prod_i e^{y_i}$, and then applying taylor series expansion for each e^{y_i}.

Combined with Equation 5.1 we get for $0 \leq k \leq s$:

$$\ell^{e_k p^k} = [t^{e_k p^k}] \left(\sum_{i=1}^{n} a_i x_i^{p^k} t^{p^k} \right)^{e_k} = (e_k)! [t^{e_k p^k}] \left(\prod_{i=1}^{n} (\sum_{j=0}^{e_k} \frac{1}{j!} (a_i x_i^{p^k} t^{p^k})^j) \right).$$

As $d = \sum_{k=0}^{s} e_k p^k$, multiplying over all k gives

$$\begin{aligned} \ell^d &= \prod_{k=0}^{s} [t^{e_k p^k}] \left(\sum_{i=1}^{n} a_i x_i^{p^k} t^{p^k} \right)^{e_k} \\ &= \prod_{k=0}^{s} (e_k)! [t^{e_k p^k}] \left(\prod_{i=1}^{n} (\sum_{j=0}^{e_k} \frac{1}{j!} (a_i x_i^{p^k} t^{p^k})^j) \right) \end{aligned}$$

Let $t_0, t_1, \ldots, t_s$ be new variables. Replacing t^{p^k} by t_k for each $0 \leq k \leq s$ in the above equations we get:

$$\ell^d = [t_0^{e_0} t_1^{e_1} \ldots t_s^{e_s}] \prod_{k=0}^{s} (e_k)! \left(\prod_{i=1}^{n} (\sum_{j=0}^{e_k} \frac{1}{j!} (a_i x_i^{p^k} t_k)^j) \right). \tag{5.3}$$

Thus, $\ell^d = [t_0^{e_0} t_1^{e_1} \ldots t_s^{e_s}] Q_{\ell,d}$, where $Q_{\ell,d}$ is a product of the sn many polynomials as above (each of which is a bivariate polynomial in $x_i, t_k, i \in [n], k \in [s]$). This equation generalizes to express the product $\ell_1^{d_1} \cdot \ell_2^{d_2} \cdots \ell_r^{d_r}$ in the following form:

$$\ell_1^{d_1} \cdot \ell_2^{d_2} \cdots \ell_r^{d_r} = [t_1^{\nu_1} t_2^{\nu_2} \ldots t_D^{\nu_D}] \prod_{k=1}^{D} \prod_{i=1}^{n} q_{k,i}, \tag{5.4}$$

where $D = (s+1)r$, and $\nu_k < p$ for each $k \in [D]$ such that $d_j = \sum_{k=(s+1)(j-1)+1} (s+1) j \nu_k p^{k-(s+1)(j-1)-1}, j \in [r]$. It is obtained simply by applying Equation 5.3 to each $\ell_j^{d_j}$ with a different set of $s+1$ many variables t_i and multiplying these equations for $1 \leq j \leq r$. We note that each $q_{k,i} \in \mathbb{F}[x_i, t_k]$ is a polynomial of individual variable degree at most $d = \sum_{j=1}^{r} d_j$, as is clear from Equation 5.3. The next claim will complete the proof of Theorem 5.1.2.

Claim 5.3.1. *$\ell_1^{d_1} \cdot \ell_2^{d_2} \cdots \ell_r^{d_r} \pmod{I}$ has an arithmetic circuit which is a $d^{O(r)}$-sum of products of univariate polynomials, where each univariate polynomial in x_i involved in a product has degree at most $\deg(p_i(x_i)) - 1$.*

For the proof, we first consider the following subexpression in Equation 5.4

$$[t_1^{\nu_1} t_2^{\nu_2} \ldots t_D^{\nu_D}] \prod_{k=1}^{D} q_{k,i},$$

which we will evaluate modulo $p_i(x_i)$. Note that the number of monomials of the form $\prod_{k=1}^{D} t_k^{\mu_k}, \mu_k \leq \nu_k < p$ is bounded by $p^D = (p^{s+1})^r = d^{O(r)}$. Thus, in $O^*(d^{O(r)})$ time we can expand the product $\prod_{k=1}^{D} q_{k,i}$ by multiplying out the polynomials, one by one, from left to right. After each multiplication, we replace x_i^a by its remainder $x_i^a \pmod{p}_i$ and

drop any term with a factor $t_k^p, k \in [D]$. This will result in a polynomial expression of the form

$$Q_i = \sum_{\bar{\mu}} r_{\bar{\mu}}(x_i) \prod_{k=1}^{D} t_k^{\mu_k},$$

where the sum runs over the $d^{O(r)}$ many tuples $\bar{\mu} = (\mu_1, \mu_2, \ldots, \mu_k)$ such that $\mu_k \leq \nu_k$ for each k. Thus, each $r_{\bar{\mu}}(x_i)$ is a univariate in x_i of degree at most $\deg(p_i) - 1$. We can now evaluate the product $Q_1 Q_2 \cdots Q_n$ modulo the ideal $\langle t_1^p, t_2^p, \ldots, t_D^p \rangle$ by multiplying out adjacent pairs and dropping any terms with a factor $t_k^p, k \in [D]$. This will given an expression for $Q_1 Q_2 \cdots Q_n$ modulo $\langle t_1^p, t_2^p, \ldots, t_D^p \rangle$ of the form $\sum_{\bar{\mu}} R_{\bar{\mu}} \prod_{k=1}^{D} t_k^{\mu_k}$, where each $R_{\bar{\mu}}$ is a $d^{O(r)}$-sum of products of n univariate polynomials (and in each product the i^{th} is a polynomial in x_i of degree $\deg(p_i) - 1$). Finally, we note that $R_{\bar{\nu}}$ is the desired polynomial expression for $\prod_{j=1}^{r} \ell_j^{d_j} \pmod{I}$, completing the proof of the claim.

5.3.3 Vertex Cover Detection in Low Rank Graphs

In the Vertex Cover problem, we are given a graph $G = (V, E)$ on n vertices and an integer k and the question is to decide whether there is a vertex cover of size k in G. This is a classical NP-complete problem. In this section we show an efficient algorithm to detect vertex cover in a graph whose adjacency matrix is of low rank.

Proof of Theorem 5.1.3. We present a reduction from Vertex Cover problem to Univariate Ideal Membership problem that produces a polynomial whose rank is almost same as the rank of A_G. Consider the ideal $I = \langle x_1^2 - x_1, x_2^2 - x_2, \ldots, x_n^2 - x_n \rangle$ and the polynomial

$$f = \prod_{s=1}^{\binom{n}{2}} (\vec{x} A_G \vec{x}^T - s) \cdot \prod_{t=0}^{n-k-1} \left(\sum_{i=1}^{n} x_i - t \right),$$

where A_G is the adjacency matrix of the graph G and $\vec{x} = (x_1, x_2, \ldots, x_n)$ is row-vector.

Lemma 5.3.4. *The rank of the polynomial f is at most $r + 1$.*

Proof. We note that A_G is symmetric since it encodes an undirected graph. Let Q be an invertible $n \times n$ matrix that diagonalizes A_G. So we have $Q A_G Q^T = D$ where D is a diagonal matrix with only the first r diagonal elements being non-zero. Let $\vec{y} = (y_1, y_2, \ldots, y_n)$ be another row-vector of variables. Now, we show the effect of the transform $\vec{x} \mapsto \vec{y} Q$ on the polynomial $\vec{x} A_G \vec{x}^T$. Clearly, $\vec{y} Q A_G Q^T \vec{y}^T = \vec{y} D \vec{y}^T$ and since there are only r non-zero entries on the diagonal, the polynomial $\vec{y} D \vec{y}^T$ is over the variables $y_1, y_2, \ldots, y_r$. Thus $g = \prod_{s=1}^{\binom{n}{2}} (\vec{x} A_G \vec{x}^T - s)$ is a rank r polynomial. Also $h = \prod_{t=0}^{n-k-1} (\sum_{i=1}^{n} x_i - t)$ is a rank 1 polynomial as there is only one linear form $\sum_{i=1}^{n} x_i$. Since $f = gh$, we conclude that f is a rank $r + 1$ polynomial. ∎

Now the proof of Theorem 5.1.3 follows from the next claim.

Claim 5.3.2. *The graph G has a Vertex Cover of size k if and only if $f \notin I$.*

Proof. First, observe that the set of common zeroes of the generators of the ideal I is the set $\{0,1\}^n$. Let S be a vertex cover in G such that $|S| \leq k$. We will exhibit a point $\vec{\alpha} \in \{0,1\}^n$ such that $f(\vec{\alpha}) \neq 0$. This will imply that $f \notin I$. Identify the vertices of G with $\{1,2,\ldots,n\}$. Define $\vec{\alpha}(i) = 0$ if and only if $i \in S$. Since $\vec{x}A_G\vec{x}^T = \sum_{(i,j)\in E_G} x_i x_j$ and S is a vertex cover for G, it is clear that $\vec{x}A_G\vec{x}^T(\vec{\alpha}) = 0$. Also $(\sum_{i=1}^n x_i)(\vec{\alpha}) \geq n-k$. Then clearly $f(\vec{\alpha}) \neq 0$.
For the other direction, suppose that $f \notin I$. Then by Theorem 5.1.1, there exists $\vec{\alpha} \in \{0,1\}^n$ such that $f(\vec{\alpha}) \neq 0$. Define the set $S \subseteq [n]$ as follows. Include $i \in S$ if and only if $\vec{\alpha}(i) = 0$. Since $f(\vec{\alpha}) \neq 0$, and the range of values that $\vec{x}A_G\vec{x}^T$ can take is $\{0,1,\ldots,|E|\}$, it must be the case that $\vec{x}A_G\vec{x}^T(\vec{\alpha}) = 0$. It implies that the set S is a vertex cover for G. Moreover, $\prod_{t=0}^{n-k-1}(\sum_{i=1}^n x_i - t)(\vec{\alpha}) \neq 0$ implies that $|S| \leq k$. ∎

The degree of the polynomial f is bounded by n^2+n and from Claim 5.3.2 we know that $f \pmod I$ is a non-zero polynomial if and only if G has a vertex cover of size k. By Schwartz-Zippel-Demillo-Lipton [34, 93, 116] lemma $(f \pmod I)(\vec{\beta})$ is non-zero with high probability when $\vec{\beta}$ is chosen randomly from a small domain. Now, we need to just compute $(f \pmod I)(\vec{\beta})$ where f is a rank $r+1$ polynomial with $\ell_i = (\vec{x}Q^{-1})_i$ for each $1 \leq i \leq r$ and $\ell_{r+1} = \sum_{i=1}^n x_i$ which can be performed in $(n,k)^{O(r)}$ time using Theorem 5.1.2.

5.4 Parameterized Complexity of Univariate Ideals

We have already mentioned in Fact 5.1.1, that checking if the integer permanent is zero is reducible to testing membership of a polynomial $f(\mathrm{X})$ in the ideal $\langle x_1^2,\ldots,x_n^2\rangle$. So univariate ideal membership is hard for the complexity class $\mathrm{C_{=}P}$ even when the ideal is generated by powers of variables [85]. In this section we study the univariate ideal membership with the lens of parametrized complexity. The parameters we consider are the degree of the input polynomial and the generators for the ideal.

5.4.1 Parameterized by the Degree of the Polynomial

We consider the following: Let I be a univariate ideal given by generators and $f \in \mathbb{F}[\mathrm{X}]$ a degree k polynomial. Is checking whether f is in I fixed parameter tractable (with k as the fixed parameter)? We consider two different cases, one with completely repeated roots and one with distinct roots.

Repeated Roots

We show that given a polynomial (by an arithmetic circuit) f of degree k there is an FPT in k algorithm to check $f \in I$ when $I = \langle x_1^{e_1}, x_2^{e_2},\ldots,x_n^{e_n}\rangle$. The algorithm works over either $\mathbb{Z}$ or any finite field of large characteristic.

Proof of Theorem 5.1.5

Proof. The proof consists of following three lemmas. Firstly, given an input instance a degree-k $f(\mathrm{X})$ and ideal $I = \langle x_1^{e_1}, x_2^{e_2},\ldots,x_n^{e_n}\rangle$ of ideal membership, we reduce it to

computing the (scaled) Hadamard product of $f(\mathrm{X})$ and a polynomial $g(\mathrm{X})$, where $g(\mathrm{X})$ is a weighted sum of all degree k monomials that are not in I. Then we show that we can evaluate Hadamard product (defined in Section 5.2) of any two polynomials at a point in time roughly linear in the product of the size of the circuits when one of the polynomials is given by a diagonal circuit as input. Finally the last part of the proof is a randomized construction of a homogeneous degree k diagonal circuit of top fain-in roughly $O^*(4.08^k)$ that computes a polynomial weakly equivalent to the polynomial g with constant probability. Recall that, two polynomials f and g are said to be *weakly equivalent* if they share same set of monomials.
To define the polynomial $g(\mathrm{X})$, let $S_{m,k}$ be the elementary symmetric polynomial of degree k over m variables. Set $m = \sum_{i=1}^{n}(e_i - 1)$. Let $S_{m,k}$ is defined over the variable set $\{z_{1,1}, \ldots, z_{1,e_1-1}, \ldots, z_{n,1}, \ldots, z_{n,e_n-1}\}$. We define $g(\mathrm{X})$ as the polynomial obtained from $S_{m,k}$ replacing each $z_{i,j}$ by x_i.

Lemma 5.4.1. *Given integers $e_1, e_2, \ldots, e_n$, and a polynomial $f(\mathrm{X})$ of degree k, $f \in \langle x_1^{e_1}, x_2^{e_2}, \ldots, x_n^{e_n}\rangle$ if and only if $f \circ^s g \equiv 0$.*

Proof. Suppose, $f \notin \langle x_1^{e_1}, x_2^{e_2}, \ldots, x_n^{e_n}\rangle$, then f must contain a degree k monomial $M = x_1^{f_1}x_2^{f_2}\ldots x_n^{f_n}$ such that $f_i < e_i$ for each $1 \leq i \leq n$. From the construction, it is clear that $g(\mathrm{X})$ contains M. Therefore, the polynomial $f \circ^s g$ is not identically zero. The converse is also true for the similar reason. ∎

Lemma 5.4.2. *Given a circuit C of size s computing a polynomial $g \in \mathbb{F}[\mathrm{X}]$ and a homogeneous degree k diagonal circuit $\Sigma^{[s']}\wedge^{[k]}\Sigma$ circuit D of top fan-in s' computing $f \in \mathbb{Q}[\mathrm{X}]$ and $\vec{a} \in \mathbb{Q}^n$, we can evaluate $(f \circ^s g)(\vec{a})$ in deterministic $ss' \cdot \mathrm{poly}(n,k)$ time using $\mathrm{poly}(n,k)$ space.*

Proof. Let M be a degree d monomial over X in f and $M = x_1^{e_1}\cdots x_n^{e_n}$, it follows from the definition that

$$\left(M \circ^s (b_1x_1 + \ldots + b_nx_n)^d\right)(\vec{a}) = \left(M! \cdot \frac{d!}{M!} \cdot b_1^{e_1}\cdots b_n^{e_n} \cdot M\right)(\vec{a}) = d! \cdot M(a_1b_1, \ldots, a_nb_n).$$

Recall that, $M!$ is used for $e_1!\cdots e_n!$. As $\circ^s$ distributes over addition, we can write

$$\left(f \circ^s \sum_{i=1}^{s'}(b_{i1}x_1 + \ldots + b_{in}x_n)^d\right)(\vec{a}) = d! \cdot \sum_{i=1}^{s'} f(a_1b_{i1}, \ldots, a_nb_{in}).$$

The computation can be done in deterministic $ss' \cdot \mathrm{poly}(n,k)$ time using $\mathrm{poly}(n,k)$ space. ∎

Lemma 5.4.3. *There is an efficient randomized algorithm that constructs with constant probability a homogeneous degree k diagonal circuit D of top fan-in $O^*(4.08^k)$ which computes a polynomial weakly equivalent to the polynomial g (defined before Lemma 5.4.1).*

Proof. To construct such a diagonal circuit D, we use the idea of [75]. We pick a collection of colourings $\{\zeta : [m] \to [1.5 \cdot k]\}$ of size roughly $O^*((\frac{e}{\sqrt{3}})^k)$ uniformly at random. For each such colouring ζ_i, we define a $\Pi^{[1.5\cdot k]}\Sigma$ formula $P_i = \prod_{j=1}^{1.5k}(L_j + 1)$, where $L_j = \sum_{\ell:\zeta_i(\ell)=j} x_\ell$.

We say that a monomial is *covered* by a coloring ζ_i if the monomial is in P_i. It is easy to see that, given any multilinear monomial of degree k, the probability that a random coloring will cover the monomial is roughly $(\frac{\sqrt{3}}{e})^k$. Hence, going over such a collection of colorings of size $O^*((\frac{e}{\sqrt{3}})^k)$ chosen uniformly at random, with a constant probability all the multilinear terms of degree k will be covered. To take the Hadamard product with a polynomial of degree k, we need to extract out the degree k homogeneous part (say P_i') from each P_i. Notice that, using elementary symmetric polynomial over $1.5k$ many variables $S_{1.5k,k}$, we can write $P_i' = S_{1.5k,k}(L_1, \ldots, L_{1.5k})$. Now we use Lemma 5.2.4 to get a diagonal $\Sigma\wedge^{[k]}\Sigma$ circuit of top fan-in roughly $\binom{1.5k}{0.5k}$ for each P_i'. Define $D = \sum_{i=1}^{O^*((\frac{e}{\sqrt{3}})^k)} P_i'$. By a direct calculation, one can obtain a diagonal circuit D of top fan-in $O^*(4.08^k)$ which is weakly equivalent to the polynomial $S_{m,k}$. The construction of the polynomial $g(\mathrm{X})$ from $S_{m,k}$ is already explained before Lemma 5.4.1. ∎

Now, given a circuit C computing $f \in \mathbb{Z}[\mathrm{X}]$ and integers $e_1, \ldots, e_n$, to decide the membership of f in the ideal $I = \langle x_1^{e_1}, \ldots, x_n^{e_n}\rangle$, we construct a diagonal circuit D from Lemma 5.4.3. Following Lemma 5.4.1, we can decide the membership of f in the ideal checking the polynomial $C \circ^s D$ is identically zero or not which can be performed by randomly picking $\vec{a} \in \mathbb{Z}^n$ using Schwartz-Zippel-Demillo-Lipton Lemma [34, 93, 116] and evaluating[5] $(C \circ^s D)(\vec{a})$ using Lemma 5.4.2. The running time is dominated by the size of D and the time required to compute $(C \circ^s D)(\vec{a})$, which are bounded by $4.08^k \cdot \mathrm{poly}(nsd(\sum_{i=1}^n e_i))$. ∎

Distinct Roots

For degree-k, n-variate polynomials f, we do not have an algorithm with running time better than $O^*(\binom{n+k}{k})$ for univariate ideal membership in general. However, if each generator polynomial p_i has distinct roots we obtain a faster algorithm.

Theorem 5.4.1. *Let $I = \langle p_1(x_1), \ldots, p_n(x_n)\rangle$ be a univariate ideal given explicitly by a set of univariate polynomials $p_1, \ldots, p_n$ such that for each $i \in [n]$, $p_i(x_i)$ has distinct roots over $\mathbb{Q}$. Given a polynomial $f(X) \in \mathbb{C}[X]$ of degree k and I as input, we can decide whether $f \in I$ or not in randomized $O^*(n^{k/2})$ time.*

Proof. W.l.o.g. we can assume the degree of each p_i is at most k. Otherwise, we can drop p_i from I. For $i \in [n]$, let $S_i \subset \mathbb{Q}$ be the set of all roots of p_i. By Alon's Combinatorial Nullstellensatz (Theorem 5.1.1), Theorem 5.4.1 can be restated as the following.

Claim 5.4.1. *Given a polynomial $f(X) \in \mathbb{C}[X]$ of degree k and $S_1, \ldots, S_n$ such that for each $i \in [n]$, $S_i \subset \mathbb{C}$ as inputs, we can decide whether $S_1 \times \cdots \times S_n$ contains a nonzero of f in in randomized $O^*(n^{k/2})$ time.*

For a degree-k polynomial $f \in \mathbb{F}[X]$ let

$$\tilde{f} = x_{n+1}^k \cdot f\left(\frac{x_1}{x_{n+1}}, \frac{x_2}{x_{n+1}}, \ldots, \frac{x_n}{x_{n+1}}\right),$$

[5]Over $\mathbb{Z}$ the given circuit can compute numbers as large as $2^{2^{n^{O(1)}}}$. To handle this, a standard idea is to evaluate the circuit modulo a random polynomial bit prime.

be its homogenization. Thus, $\tilde{f}$ is homogeneous of degree k and $\tilde{f}(x_1, x_2, \ldots, x_n, 1) = f(x_1, \ldots, x_n)$. Clearly, f is nonzero on the n-dimensional grid $S_1 \times \cdots \times S_n$ if and only if $\tilde{f}$ is nonzero on the $n+1$ -dimensional grid $S_1 \times \cdots \times S_n \times \{1\}$. Hence, without loss of generality we can assume f is homogeneous degree k.

Observation 5.4.1. *For a homogeneous polynomial f of degree k,*

$$f \circ^s (a_1x_1 + \ldots + a_nx_n)^k \mid_{\vec{1}} = k! \cdot f(a_1, \ldots, a_n).$$

We need to decide whether there exists a point $\vec{a} \in S_1 \times \cdots \times S_n$ such that $f(\vec{a}) \neq 0$. For each $(a_1, \ldots, a_n) \in S_1 \times \ldots \times S_n$, by Equation 5.2 we can write,

$$\frac{1}{k!} \cdot (a_1x_1 + \ldots + a_nx_n)^k = \sum_{\ell=1}^{nk+1} \beta_\ell \cdot \prod_{i=1}^{n} p_i(a_i\alpha_\ell x_i).$$

where $\alpha_1, \ldots, \alpha_n \in \mathbb{Q}$ are some distinct points, $\beta_\ell \in \mathbb{Q}$, and each p_i is univariate. Now, we define the "grid" polynomial

$$g = \sum_{\ell=1}^{nk+1} \beta_\ell \cdot \prod_{i=1}^{n} \left(\sum_{a_i \in S_i} \xi_{i,a_i} p_i(a_i\alpha_\ell x_i) \right) \tag{5.5}$$

$$= \sum_{(a_1,\ldots,a_n) \in S_1 \times \ldots \times S_n} \prod_{i=1}^{n} \xi_{i,a_i} \left(\sum_{\ell=1}^{nk+1} \beta_\ell \cdot \prod_{i=1}^{n} p_i(a_i\alpha_\ell x_i) \right), \tag{5.6}$$

where $\xi_{i,a_i}, i \in [n], a_i \in S_i$ are new variables. Hence,

$$f \circ^s g \mid_{\vec{1}} = \sum_{(a_1,\ldots,a_n) \in S_1 \times \ldots \times S_n} \prod_{i=1}^{n} \xi_{i,a_i} f \circ^s \left(\sum_{\ell=1}^{nk+1} \beta_\ell \cdot \prod_{i=1}^{n} p_i(a_i\alpha_\ell x_i) \right) \mid_{\vec{1}} \tag{5.7}$$

$$= \sum_{(a_1,\ldots,a_n) \in S_1 \times \ldots \times S_n} \prod_{i=1}^{n} \xi_{i,a_i} f(a_1, a_2, \ldots, a_n) \tag{5.8}$$

Thus, $f \circ^s g \mid_{\vec{1}}$ is a nonzero polynomial (in the ξ_{i,a_i} variables) of degree n iff $f \circ^s (a_1x_1 + \cdots a_nx_n)^k$ is nonzero for some $(a_1, \ldots, a_n) \in S_1 \times \ldots S_n$. By the Polynomial Identity Lemma [34, 93, 116], we can independently randomly assign values for the ξ_{i,a_i} variables from $[n^2]$, and the evaluation is nonzero with probability at least $1 - 1/n$ iff f nonzero on a grid point in $S_1 \times \cdots \times S_n$. Furthermore, from Equation 5.7 we note that we can clear the denominators of all the β_ℓ and the polynomials $p_i(a_i\alpha_i x_i)$ and the polynomial f (given by input circuit) and take out a common factor $\frac{1}{D}$ (where D is a polynomially many bits long integer) to write Equation 5.7 as

$$f \circ^s g \mid_{\vec{1}} = \frac{1}{D} \sum_{(a_1,\ldots,a_n) \in S_1 \times \ldots \times S_n} \prod_{i=1}^{n} \xi_{i,a_i} \hat{f} \circ^s \left(\sum_{\ell=1}^{nk+1} \gamma_\ell \cdot \prod_{i=1}^{n} \hat{p}_i(a_i\alpha_\ell x_i) \right) \mid_{\vec{1}},$$

where $\hat{f}$ and $\hat{p}_i(a_i\alpha_\ell x_i)$ have integer coefficients. Thus, when $f \circ^s g \mid_{\vec{1}}$ is nonzero at a choice of the ξ_{i,a_i} then it is of absolute value at least $1/D$.
Therefore, after randomly choosing $\xi_{i,a_i} \in_R [n^2]$, it is clear from Equation 5.5 that the problem reduces to efficiently computing the scaled Hadamard product $f \circ^s h \mid_{\vec{1}}$ evaluated at $\vec{1}$, where $h = \prod_{i=1}^{n} q_i(x_i)$ and each q_i is of degree k. We now show that $f \circ^s h \mid_{\vec{1}}$ can be computed in $O^*(n^{k/2})$ time which suffices to detect if $f \circ^s g \mid_{\vec{1}}$ is nonzero in $O^*(n^{k/2})$ time.

Claim 5.4.2. *$f \circ^s \prod_{i=1}^{n} q_i(x_i) \mid_{\vec{1}}$ can be computed in $O^*(n^{k/2})$ time.*

Notice that the above claim completes the proof, because the summation over ℓ has $nk+1$ terms. Let $\beta = \max_\ell\{|\beta_\ell|\}$. Then the overall error in $f \circ^s g \mid_{\vec{1}}$ is bounded by the precision error of the claim multiplied by $(nk+1)\beta$ which can be made smaller than $1/D$ by choosing the precision error of the claim.
We now prove the claim. We need approximations because we will need to approximately compute the roots of the univariate polynomials q_i. Let R_i denote the nonzero roots of q_i. Then we can write

$$\prod_i q_i = \prod_{i=1}^{n} x_i^{\mu_i} \prod_{i=1}^{n} \prod_{-\beta \in R_i} (x_i + \beta)^{\nu_{i,\beta}},$$

where $\nu_{i,\beta}$ is the multiplicity of root $-\beta$ in q_i. If $\sum_i \mu_i > k$ then clearly $f \circ^s \prod_i q_i = 0$. Otherwise, let $\sum_i \mu_i = s$ and let $r = k - s$. Let $\prod_i \prod_{\beta \in R_i} \beta^{\nu_{i,\beta}} = \Gamma$. Write $\prod_{i=1}^{n} \prod_{-\beta \in R_i} (x_i+\beta)^{\nu_{i,\beta}}$ as $\Gamma \prod_{i=1}^{n} \prod_{-\beta \in R_i} (x_i/\beta + 1)^{\nu_{i,\beta}}$. Let $m = \sum_i \deg(q_i) - s$ and consider the elementary symmetric polynomial $S_{m,r}$ in variables $y_1, y_2, \ldots, y_m$. By Lee's result [64], $S_{m,r}$ can be expressed as $O^*(m^{r/2})$ sum of powers of linear forms. In the polynomial $S_{m,r}$ we replace the m variables $y_1, y_2, \ldots, y_m$ by the m nonzero roots (of the form x_i/β, as explained above) of $\prod_j q_j$. Let the product of the resulting polynomial (which is still a $O^*(m^{r/2})$-sum of r-power of linear forms) with $\Gamma \cdot \prod_{i=1}^{n} x_i^{\mu_i}$ be denoted by Q. Clearly, $f \circ^s \prod_i q_i = f \circ^s Q$. Since Q is a sum of power of linear forms using Observation 5.4.1, we can evaluate $f \circ^s Q \mid_{\vec{1}}$ with $O^*(n^{k/2})$ arithmetic operations.
Now, replacing each root β by a rational approximation β' such that $|\beta - \beta'| \leq 1/2^L$ for a suitably chosen polynomial bit number L, the overall error in the approximation to $f \circ^s Q \mid_{\vec{1}}$ will be bounded. It can be made smaller than ε by choosing L suitably large. We can use any efficient root approximation algorithm for univariate polynomials to find all such root approximations β'.
This completes the proof of the claim and the theorem. ∎

5.5 Univariate Ideal Membership Parameterized by Number of Generators

In this section, we consider the univariate ideal membership parameterized on the number of generators of the univariate ideal. More precisely, we consider univariate ideal membership for input $f(\mathrm{X})$ by a circuit of size s and univariate ideal $I = \langle p_1(x_1), \ldots, p_k(x_k)\rangle$ (with k as fixed parameter).
We show that the *nonmembership* problem is W[2]-hard by giving an efficient reduction from the k-dominating set problem which is W[2]-complete [33].
Proof of Theorem 5.1.6. Let (G, k) be an instance of the k-dominating set problem, where $G = (V, E)$ is an n-vertex graph and the fixed parameter k is the size of the dominating set. Let $V(G) = \{1, 2, \ldots, n\}$. For $1 \leq i \leq k$, we define polynomials

$$p_i(x_i) = \prod_{j=1}^{n} (x_i - j).$$

The W[2]-hardness proof is an application of Alon's Combinatorial Nullstellensatz (Theorem 5.1.1): By definition, for each p_i its zero set is $Z(p_i) = [n]$. Therefore, a polynomial $g \in \mathbb{Q}[x_1, x_2, \ldots, x_k]$ is in the ideal $\langle p_1, p_2, \ldots, p_k \rangle$ if and only if g is zero on every point in the k-dimensional grid $[n] \times [n] \times \cdots \times [n]$.
For each $u \in V$, let $N_u = \{u\} \cup \{v \in V \mid uv \in E\}$ denote its closed neighborhood in G. Define polynomials $q_u, u \in V$

$$q_u = \sum_{i=1}^{k} \prod_{v \in \overline{N_u}} (x_i - v)^2.$$

Notice that q_u is nonzero at a grid point $x_i = v_i, 1 \leq i \leq k$ if and only if there is a $v_i \in N_u$. That is, q_u is nonzero at $(v_1, v_2, \ldots, v_k)$ if and only if some v_i dominates u. Now, letting

$$q_G(x_1, x_2, \ldots, x_k) = \prod_{u=1}^{n} q_u,$$

it follows that q_G is nonzero at a grid point $x_i = v_i, 1 \leq i \leq k$ if and only if $\{v_1, v_2, \ldots, v_k\}$ is a dominating set for G.
Hence, by Theorem 5.1.1 we have the following claim which completes the proof.

Claim 5.5.1. *The polynomial q_G is* not *in the univariate ideal $\langle p_1, p_2, \ldots, p_n \rangle$ if and only if the graph G has a dominating set of size k.*

5.6 Non-deterministic Algorithm for Univariate Ideal Membership

In this section we prove Theorem 5.1.7. Given a polynomial $f(\mathrm{X}) \in \mathbb{Q}[\mathrm{X}]$ and a univariate ideal $I = \langle p_1(x_1), \ldots, p_n(x_n) \rangle$ where the generators are $p_1, \ldots, p_n$, we show a non-deterministic algorithm to decide the (non)-membership of f in I. By Theorem 5.1.1, it suffices to show that there is a common zero $\vec{\alpha}$ of the generators $p_1, p_2, \ldots, p_n$ such that $f(\alpha) \neq 0$. Since in general $\vec{\alpha} \in \mathbb{C}^n$, it is not immediately clear how to guess such a common zero by a NP machine. However, we are able to show that for the NP machine it suffices to guess such an $\vec{\alpha}$ upto polynomially many bits of approximation.
We begin by proving a few technical facts which are useful for the main proof. Write $f(\mathrm{X}) = \sum_{i=1}^{n} h_i(\mathrm{X})\ p_i(x_i) + R(\mathrm{X})$ where for all $i \in [n]$, $\deg_{x_i}(R) < \deg(p_i)$. For any polynomial g, let $|c(g)|$ be the maximum coefficient (in absolute value) appearing in g. The following lemma gives an estimate for the coefficients of the polynomials $h_1, \ldots, h_n, R$.

Lemma 5.6.1. *Let $2^{-L} \leq |c(f)|, |c(p_i)| \leq 2^L$. Then there is $L' = \mathrm{poly}(L, d, n)$ such that $2^{-L'} \leq |c(h_i)|, |c(R)| \leq 2^{L'}$ where d is the degree upper bound for f, and $\{p_i : 1 \leq i \leq n\}$.*

Proof. The estimate on L' follows implicitly from the known results [31]. It can be also seen by direct computation. Write $f(\mathrm{X}) = \sum_i f_i(x_2, \ldots, x_n)\ x_1^i$ and then divide $x_1^i \pmod{p_1(x_1)}$ for each i. The modulo computation can be done by writing $x_1^i = q_1(x_1)p(x_1) + r_1(x_1)$ with the coefficients of q_1 and r_1 are unknown. We can then solve it using standard linear

algebra. In particular, one can use the Cramer's rule for system of linear equation solution. The growth of the bit-size is only $\text{poly}(L, d)$. More precisely, if $c_{\max}$ is the maximum among $|c(f)|, |c(p_1)|$, any final coefficient is at most $c_{\max} \cdot 2^{\text{poly}(L,d)}$. We repeat the procedure for the other univariate polynomials one by one. The final growth on the coefficients size is at most $\text{poly}(n, L, d)$. ∎

Let $\vec{\alpha} = (\alpha_1, \ldots, \alpha_n) \in \mathbb{C}^n$ be such that $p_i(\alpha_i) = 0$, $1 \leq i \leq n$. From Lemma 5.2.1, we get that $\frac{1}{2^{\hat{L}}} \leq |\alpha_i| \leq 2^{\hat{L}}$ where $\hat{L} = \text{poly}(L, d)$. Let $\tilde{\alpha}_i \in \mathbb{Q}[i]$ be an ϵ-approximation of α_i, e.g. $|\alpha_i - \tilde{\alpha}_i| \leq \epsilon$. Then we show that the absolute value of $p_i(\tilde{\alpha}_i)$ is not too far from zero.

Observation 5.6.1. *For $1 \leq i \leq n$ we have that $|p_i(\tilde{\alpha}_i)| \leq \epsilon \cdot 2^{(dL)^{O(1)}}$.*

Proof. Let $p_i(x_i) = c \cdot \prod_{j=1}^{d}(x_i - \beta_{i,j})$ and w.l.o.g assume that $\tilde{\alpha}_i$ is the approximation of the root $\beta_{i,1}$. Then $|p_i(\tilde{\alpha}_i)| \leq \epsilon \cdot |c| \cdot \prod_{j=2}^{d} |\tilde{\alpha}_i - \beta_{i,j}| \leq \epsilon \cdot |c| \cdot \prod_{j=2}^{d}(|\beta_{i,1} - \beta_{i,j}| + \epsilon) \leq \epsilon \cdot 2^{\text{poly}(d,L)}$. The final bound follows from the bound on the roots given in Lemma 5.2.1. ∎

Since we have an upper bound on the coefficients of the polynomials $\{h_i : 1 \leq i \leq n\}$ from Lemma 5.6.1, it follows that for $1 \leq i \leq n$ we have that $|h_i(\tilde{\alpha})| \leq 2^{(ndL)^{O(1)}}$. Here we use the fact that the approximate root α_i can be trivially bounded by $2^{\hat{L}+1}$.

5.6.1 Proof of Theorem 5.1.7

Proof. If f is not in the ideal I, by Alon's Nullstellensatz, we know that there exists a tuple $\vec{\alpha} = (\alpha_1, \ldots, \alpha_n) \in Z(p_1) \times \ldots \times Z(p_n)$ such that $R(\vec{\alpha}) \neq 0$. Suppose that the NP Machine guess the tuple $\vec{\tilde{\alpha}} = (\tilde{\alpha}_1, \ldots, \tilde{\alpha}_n)$ which is the ϵ-approximation of the tuple $\vec{\alpha} = (\alpha_1, \ldots, \alpha_n)$ [6]. Using the black-box for f, obtain the value for $f(\vec{\tilde{\alpha}})$. Next, we show that the value $|f(\vec{\tilde{\alpha}})|$ distinguishes between the cases $f \in I$ and $f \notin I$.

Case 1 : $f \in I$
$|f(\vec{\tilde{\alpha}})| = |\sum_{i=1}^{n} h_i(\vec{\tilde{\alpha}}) p_i(\tilde{\alpha_i})| \leq (\sum_{i=1}^{n} |h_i(\vec{\tilde{\alpha}})|) \cdot \epsilon \cdot 2^{(dL)^{c_1}} \leq \epsilon \cdot 2^{(ndL)^{c_2}}$. where the constant c_2 is fixed by Observation 5.6.1 and the bounds on $|h_i(\vec{\tilde{\alpha}})|$.

Case 2 : $f \notin I$
Recall the inequality for complex numbers : $|Z_1 + Z_2| \geq |Z_2| - |Z_1|$. Using this write $|f(\vec{\tilde{\alpha}})| \geq |R(\vec{\tilde{\alpha}})| - \sum_{i=1}^{n} |h_i(\vec{\tilde{\alpha}})| \; |p_i(\vec{\tilde{\alpha}})|$. Notice that $|R(\vec{\tilde{\alpha}})| \geq |R(\vec{\alpha})| - |R(\vec{\tilde{\alpha}}) - R(\vec{\alpha})|$. Combining we get the following : $|f(\vec{\tilde{\alpha}}) \geq |R(\vec{\alpha})| - |R(\vec{\tilde{\alpha}}) - R(\vec{\alpha})| - \epsilon \cdot 2^{(ndL)^{c_2}}$.
Now to complete the proof, we show a lower bound on $|R(\vec{\alpha})|$ and an upper bound for $|R(\vec{\tilde{\alpha}}) - R(\vec{\alpha})|$.

Claim 5.6.1. $|R(\vec{\alpha})| \geq \frac{1}{2^{(ndL)^{c_3}}}$ *for some constant c_3.*

Proof. Define the polynomial $\hat{R}(x_n) = R(\alpha_1, \ldots, \alpha_{n-1}, x_n) = c \cdot \prod_{j=1}^{d'}(x_n - \beta_j)$ where c is some constant and $d' \leq d$. Note that α_n is not a zero for $\hat{R}(x_n)$. Consider the polynomial $Q(x_n) = p_n(x_n)\hat{R}(x_n)$. The set $\{\alpha_n, \beta_1, \ldots, \beta_{d'}\} \subseteq Z(Q)$ and $\alpha_n \neq \beta_j : 1 \leq j \leq d'$. Using

[6] Later we fix ϵ suitably and use Lemma 5.2.3 to verify in polynomial time that $\vec{\tilde{\alpha}}$ is indeed ϵ-approximation of $\vec{\alpha}$.

the root separation bound for $|\alpha_n - \beta_j|$ obtained in Lemma 5.2.2, we can easily lower bound that $|\hat{R}(\alpha_n)| \geq \frac{1}{2^{(ndL)^{c_3}}}$. ∎

Claim 5.6.2. *$|R(\vec{\tilde{\alpha}}) - R(\vec{\alpha})| \leq 2^{(ndL)^{c_4}}$ for some constant c_4.*

Proof. Define $R^0(\vec{\tilde{\alpha}}) = R(\vec{\alpha})$ and $R^i(\vec{\tilde{\alpha}}) = R(\tilde{\alpha}_1, \ldots, \tilde{\alpha}_i, \alpha_{i+1}, \ldots, \alpha_n)$. Then we use triangle inequality to notice that $|R(\vec{\alpha}) - R(\vec{\tilde{\alpha}})| \leq \sum_{i=1}^{n} |R^{i-1}(\vec{\tilde{\alpha}}) - R^i(\vec{\tilde{\alpha}})|$. Write explicitly $R^{i-1}(\vec{\tilde{\alpha}}) - R^i(\vec{\tilde{\alpha}}) = \sum_{\vec{e}} c_{\vec{e}} \tilde{\alpha}_1^{e_1} \ldots \tilde{\alpha}_{i-1}^{e_{i-1}} (\alpha_i^{e_i} - \tilde{\alpha}_i^{e_i}) \alpha_i^{e_{i+1}} \ldots \alpha_n^{e_n}$. Notice the upper bounds on $|\alpha_i| \leq 2^{(ndL)^{O(1)}}$, and $|\alpha_i - \tilde{\alpha}_i| \leq \epsilon$. We apply these bounds and use triangle inequality to get that $|R(\vec{\tilde{\alpha}}) - R(\vec{\alpha})| \leq \epsilon \cdot 2^{(ndL)^{c_4}}$. ∎

Combining Claim 5.6.1, and Claim 5.6.2, we get the lower bound $|f(\vec{\tilde{\alpha}})| \geq \frac{1}{2^{(ndL)^{c_3}}} - \epsilon \cdot (2^{(ndL)^{c_4}} + 2^{(ndL)^{c_2}})$. To make the calculation precise, let $3M = \frac{1}{2^{(ndL)^{c_3}}}$ and choose ϵ such that $\epsilon \cdot (2^{(ndL)^{c_4}} + 2^{(ndL)^{c_2}}) \leq M$.
The final implication will be $|f(\vec{\tilde{\alpha}})| \leq M$ when $f \in I$ and $|f(\vec{\tilde{\alpha}})| \geq 2M$ when $f \notin I$. It is important to note that the parameter M can be pre-computed from the input parameters efficiently.
Now we show how to verify that the guessed point $\vec{\tilde{\alpha}}$ is a good approximation of the roots for the univariate polynomials. We need to only verify that for each i, $\tilde{\alpha}_i$ is a good approximation for *some root* of the univariate polynomial $p_i(x_i)$. The fact that it is also a good approximation for the non-zero of R is already verified above. The NP machine, given $p_1, \ldots, p_n$ guesses $\tilde{\alpha}_i$ using b bits and verifies that $|p_i(\tilde{\alpha}_i)| < 2^{-L}\epsilon^d$ which, by lemma 5.2.3, shows that the guessed $\tilde{\alpha}_i$ is ϵ-close to some root of p_i.
We note that such a guess always exists. Indeed, invoking Observation 5.6.1 with $|\alpha_i - \tilde{\alpha}_i| \leq \delta$ we can conclude that $|p_i(\tilde{\alpha}_i)| \leq \delta \cdot 2^{(dL)^{O(1)}}$. Now, the NP machine can guess b bits such that $|\alpha_i - \tilde{\alpha}_i| \leq 2^{-b}$. We require $2^{-b} \cdot 2^{(dL)^{O(1)}} < 2^{-L}\epsilon^d$, simplifying we get, $2^{-b} < 2^{-(dL)^{O(1)}} \cdot \epsilon^d$. Hence $b > (dL)^{O(1)} \log \frac{1}{\epsilon}$. Thus using $\text{poly}(d, L, \log \frac{1}{\epsilon})$ bits there is always a guess $\tilde{\alpha}_i$ for which $|p_i(\tilde{\alpha}_i)| < 2^{-L}\epsilon^d$. ∎

5.7 Conclusion

This chapter studies special cases of univariate ideal membership problem and obtains exact algorithms in the paradigm of parameterized complexity. Most notably, we give new algorithm for computing the permanent of low rank matrices and an algorithm for detecting vertex covers in graphs whose adjacency matrices have low rank.
We give a co-non-deterministic algorithm for univariate ideal membership when the univariates have distinct roots. This complements the result of Alon and Tarsi [7] who showed coNP-hardness for this special case. This establishes the coNP-completeness and settles the complexity of distinct roots univariate ideal membership.

Bibliography

[1] Anil Ada, Arkadev Chattopadhyay, Omar Fawzi, and Phuong Nguyen. The NOF multiparty communication complexity of composed functions. *Comput. Complex.*, 24(3):645–694, 2015. doi:10.1007/s00037-013-0078-4.

[2] 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. doi:10.1137/140975103.

[3] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. Primes is in p. *Annals of Mathematics*, 160, 09 2002. doi:10.4007/annals.2004.160.781.

[4] 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. doi:10.1109/FOCS.2008.32.

[5] Miklós Ajtai and Yuri Gurevich. Monotone versus positive. *J. ACM*, 34(4):1004–1015, 1987.

[6] Noga Alon. Combinatorial nullstellensatz. *Comb. Probab. Comput.*, 8(1-2):7–29, January 1999. URL: http://dl.acm.org/citation.cfm?id=971651.971653.

[7] Noga Alon and Michael Tarsi. A note on graph colorings and graph polynomials. *Journal of Combinatorial Theory, Series B*, 70(1):197–201, 1997. URL: https://www.sciencedirect.com/science/article/pii/S0095895697917536, doi:https://doi.org/10.1006/jctb.1997.1753.

[8] A. S. Amitsur and J. Levitzki. Minimal identities for algebras. *Proceedings of the American Mathematical Society*, 1(4):449–463, 1950. URL: http://www.jstor.org/stable/2032312.

[9] Kotlov Andrei. Rank and chromatic number of a graph. *J. Graph Theory*, 26(1):1–8, September 1997.

[10] Kotlov Andrew and Lovasz Laszlo. The rank and size of graphs. *Journal of Graph Theory*, 23(2):185–189.

[11] Vikraman Arvind, Abhranil Chatterjee, Rajit Datta, and Partha Mukhopadhyay. Fast exact algorithms using hadamard product of polynomials. *CoRR*, abs/1807.04496, 2018. URL: http://arxiv.org/abs/1807.04496, arXiv:1807.04496.

[12] Vikraman Arvind, Pushkar S. Joglekar, and Gaurav Rattan. On the complexity of noncommutative polynomial factorization. In *Mathematical Foundations of Computer Science 2015 - 40th International Symposium, MFCS 2015, Milan, Italy, August 24-28, 2015, Proceedings, Part II*, pages 38–49, 2015. URL: http://dx.doi.org/10.1007/978-3-662-48054-0_4, doi:10.1007/978-3-662-48054-0_4.

[13] Vikraman Arvind and Partha Mukhopadhyay. The ideal membership problem and polynomial identity testing. *Inf. Comput.*, 208(4):351–363, 2010. doi:10.1016/j.ic.2009.06.003.

[14] Vikraman Arvind, Partha Mukhopadhyay, and Srikanth Srinivasan. New results on noncommutative and commutative polynomial identity testing. *Computational Complexity*, 19(4):521–558, 2010. URL: http://dx.doi.org/10.1007/s00037-010-0299-8, doi:10.1007/s00037-010-0299-8.

[15] Vikraman Arvind and S. Raja. Some lower bound results for set-multilinear arithmetic computations. *Chicago J. Theor. Comput. Sci.*, 2016 (6), 2016.

[16] László Babai, Noam Nisan, and Mario Szegedy. Multiparty protocols, pseudorandom generators for logspace, and time-space trade-offs. *J. Comput. Syst. Sci.*, 45(2):204–232, 1992. Preliminary version appeared in STOC 1989. doi:10.1016/0022-0000(92)90047-M.

[17] Alexander I. Barvinok. Two algorithmic results for the traveling salesman problem. *Math. Oper. Res.*, 21(1):65–84, February 1996. URL: http://dx.doi.org/10.1287/moor.21.1.65, doi:10.1287/moor.21.1.65.

[18] C. Beeri. An improvement on valiant's decision procedure for equivalence of deterministic finite turn pushdown machines. *Theoretical Computer Science*, 3(3):305 – 320, 1976. URL: http://www.sciencedirect.com/science/article/pii/0304397576900499, doi:https://doi.org/10.1016/0304-3975(76)90049-9.

[19] E. R. Berlekamp. Factoring polynomials over large finite fields*. In *Proceedings of the Second ACM Symposium on Symbolic and Algebraic Manipulation*, SYMSAC '71, pages 223–, New York, NY, USA, 1971. ACM. URL: http://doi.acm.org/10.1145/800204.806290, doi:10.1145/800204.806290.

[20] J. Berstel and C. Reutenauer. *Noncommutative Rational Series with Applications*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2011. URL: https://books.google.co.in/books?id=LL8Nhn72I_8C.

[21] Malcolm Bird. The equivalence problem for deterministic two-tape automata. *J. Comput. Syst. Sci.*, 7(2):218–236, 1973. doi:10.1016/S0022-0000(73)80045-5.

[22] Andreas Björklund. Determinant sums for undirected hamiltonicity. *SIAM J. Comput.*, 43(1):280–299, 2014. doi:10.1137/110839229.

[23] Andreas Björklund, Petteri Kaski, and Lukasz Kowalik. Constrained multilinear detection and generalized graph motifs. *Algorithmica*, 74(2):947–967, 2016. doi:10.1007/s00453-015-9981-1.

[24] J. Bourgain, A. A. Glibichuk, and S. V. Konyagin. Estimates for the number of sums and products and for exponential sums in fields of prime order. *Journal of London Mathematical Society*, 2:380–398, 2006.

[25] Cornelius Brand, Holger Dell, and Thore Husfeldt. Extensor-coding. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 151–164, 2018. URL: http://doi.acm.org/10.1145/3188745.3188902, doi:10.1145/3188745.3188902.

[26] Arkadev Chattopadhyay, Rajit Datta, and Partha Mukhopadhyay. Negations provide strongly exponential savings. *Electron. Colloquium Comput. Complex.*, 27:191, 2020. URL: https://eccc.weizmann.ac.il/report/2020/191.

[27] Arkadev Chattopadhyay, Nikhil Mande, and Suhail Sherif. The log-approximate-rank conjecture is false. *J. ACM*, 67(4):23:1–23:28, 2020.

[28] 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. doi:10.1561/0400000043.

[29] Xi Chen, Igor Carboni Oliviera, and Rocco A. Servedio. Addition is exponentially harder than counting for shallow monotone circuits. In *STOC*, pages 665–677, 2017.

[30] Mireille Clerbout, Michel Latteux, and Yves Roos. Decomposition of partial commutations. In Mike Paterson, editor, *Automata, Languages and Programming, 17th International Colloquium, ICALP90, Warwick University, England, UK, July 16-20, 1990, Proceedings*, volume 443 of *Lecture Notes in Computer Science*, pages 501–511. Springer, 1990. doi:10.1007/BFb0032054.

[31] George E. Collins. Subresultants and reduced polynomial remainder sequences. *J. ACM*, 14(1):128–142, 1967. URL: http://doi.acm.org/10.1145/321371.321381, doi:10.1145/321371.321381.

[32] David A. Cox, John Little, and Donal O'Shea. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra, 3/e (Undergraduate Texts in Mathematics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[33] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. doi:10.1007/978-3-319-21275-3.

[34] Richard A. Demillo and Richard J. Lipton. A probabilistic remark on algebraic program testing. *Information Processing Letters*, 7(4):193 – 195, 1978. URL: http://www.sciencedirect.com/science/article/pii/0020019078900674, doi:https://doi.org/10.1016/0020-0190(78)90067-4.

[35] Volker Diekert. *Combinatorics on Traces*, volume 454 of *Lecture Notes in Computer Science*. Springer, 1990. doi:10.1007/3-540-53031-2.

[36] Volker Diekert, Markus Lohrey, and Alexander Miller. Partially commutative inverse monoids. In Rastislav Kralovic and Pawel Urzyczyn, editors, *Mathematical Foundations of Computer Science 2006, 31st International Symposium, MFCS 2006, Stará Lesná, Slovakia, August 28-September 1, 2006, Proceedings*, volume 4162 of *Lecture Notes in Computer Science*, pages 292–304. Springer, 2006. doi:10.1007/11821069_26.

[37] Manfred Droste and Paul Gastin. On recognizable and rational formal power series in partially commuting variables. In Pierpaolo Degano, Roberto Gorrieri, and Alberto Marchetti-Spaccamela, editors, *Automata, Languages and Programming, 24th International Colloquium, ICALP'97, Bologna, Italy, 7-11 July 1997, Proceedings*, volume 1256 of *Lecture Notes in Computer Science*, pages 682–692. Springer, 1997. doi:10.1007/3-540-63165-8_222.

[38] Samuel Eilenberg. *Automata, Languages, and Machines (Vol A)*. Pure and Applied Mathematics. Academic Press, 1974.

[39] 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 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 243–252, 2013. URL: http://dx.doi.org/10.1109/FOCS.2013.34, doi:10.1109/FOCS.2013.34.

[40] Emily P. Friedman and Sheila A. Greibach. A polynomial time algorithm for deciding the equivalence problem for 2-tape deterministic finite state acceptors. *SIAM J. Comput.*, 11:166–183, 1982.

[41] S. B. Gashkov and I. S. Sergeev. A method for deriving lower bounds for the complexity of monotone arithmetic circuits computing real polynomials. *Sbornik. Mathematics*, 203(10), 2012.

[42] James Geelen. An algebraic approach to matching problems. 04 2001. URL: https://www.math.uwaterloo.ca/~jfgeelen/Publications/survey.pdf.

[43] Mika Göös, Pritish Kamath, Robert Robere, and Dmitry Sokolov. Adventures in monotone complexity and TFNP. In *ITCS*, pages 38:1–38:19, 2019.

[44] T. V. Griffiths. The unsolvability of the equivalence problem for nondeterministic generalized machines. *J. ACM*, 15(3):409–413, July 1968. URL: http://doi.acm.org/10.1145/321466.321473, doi:10.1145/321466.321473.

[45] Ankit Gupta, Pritish Kamath, Neeraj Kayal, and Ramprasad Saptharishi. Arithmetic circuits: A chasm at depth 3. *SIAM J. Comput.*, 45(3):1064–1079, 2016. doi:10.1137/140957123.

[46] Tero Harju and Juhani Karhumäki. The equivalence problem of multitape finite automata. *Theor. Comput. Sci.*, 78(2):347–355, 1991. doi:10.1016/0304-3975(91)90356-7.

[47] Danny Harnik and Ran Raz. Higher lower bounds on monotone size. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA*, pages 378–387. ACM, 2000. doi:10.1145/335305.335349.

[48] Pavel Hrubes. On ϵ-sensitive monotone computations. *Computational Complexity*, 29(2):6, 2020. doi:10.1007/s00037-020-00196-6.

[49] Pavel Hrubes, Avi Wigderson, and Amir Yehudayoff. Relationless completeness and separations. In *Proceedings of the 25th Annual IEEE Conference on Computational Complexity, CCC 2010, Cambridge, Massachusetts, June 9-12, 2010*, pages 280–290, 2010. URL: http://doi.ieeecomputersociety.org/10.1109/CCC.2010.34, doi:10.1109/CCC.2010.34.

[50] Pavel Hrubes and Amir Yehudayoff. Formulas are exponentially stronger than monotone circuits in non-commutative setting. In *Computational Complexity Conference*, pages 10–14, 2013.

[51] Laurent Hyafil. The power of commutativity. In *18th Annual Symposium on Foundations of Computer Science (FOCS), Providence, Rhode Island, USA, 31 October - 1 November 1977*, pages 171–174, 1977. URL: http://dx.doi.org/10.1109/SFCS.1977.31, doi:10.1109/SFCS.1977.31.

[52] Mark Jerrum and Marc Snir. Some exact complexity results for straight-line computations over semirings. *J. ACM*, 29(3):874–897, 1982. doi:10.1145/322326.322341.

[53] Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Comput. Complex.*, 13(1-2):1–46, 2004. doi:10.1007/s00037-004-0182-6.

[54] Erich Kaltofen. Factorization of polynomials given by straight-line programs. *Adv. Comput. Res.*, 5:375–412, 1989.

[55] Mauricio Karchmer and Avi Wigderson. Monotone circuits for connectivity require superlogarithmic depth. *SIAM J. Discrete Math*, 3:255–265, 1990.

[56] P. W. Kasteleyn. *The Statistics of Dimers on a Lattice*, pages 281–298. Birkhäuser Boston, Boston, MA, 1987. doi:10.1007/978-0-8176-4842-8_20.

[57] Neeraj Kayal and Nitin Saxena. Polynomial identity testing for depth 3 circuits. *Computational Complexity*, 16(2):115–138, 2007. doi:10.1007/s00037-007-0226-9.

[58] Pascal Koiran. Hilbert's nullstellensatz is in the polynomial hierarchy. *J. Complexity*, 12(4):273–286, 1996. doi:10.1006/jcom.1996.0019.

[59] Pascal Koiran. Arithmetic circuits: The chasm at depth four gets wider. *Theor. Comput. Sci.*, 448:56–65, 2012. doi:10.1016/j.tcs.2012.03.041.

[60] Ioannis Koutis. Faster algebraic algorithms for path and packing problems. 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*, pages 575–586, 2008. doi:10.1007/978-3-540-70575-8_47.

[61] Ioannis Koutis. Constrained multilinear detection for faster functional motif discovery. *Inf. Process. Lett.*, 112(22):889–892, 2012. doi:10.1016/j.ipl.2012.08.008.

[62] Ioannis Koutis and Ryan Williams. LIMITS and applications of group algebras for parameterized problems. *ACM Trans. Algorithms*, 12(3):31:1–31:18, 2016. URL: http://doi.acm.org/10.1145/2885499, doi:10.1145/2885499.

[63] Guillaume Lagarde, Guillaume Malod, and Sylvain Perifel. Non-commutative computations: lower bounds and polynomial identity testing. *Electronic Colloquium on Computational Complexity (ECCC)*, 23:94, 2016. URL: http://eccc.hpi-web.de/report/2016/094.

[64] Hwangrae Lee. Power sum decompositions of elementary symmetric polynomials. 492, 08 2015.

[65] Meena Mahajan and V. Vinay. A combinatorial algorithm for the determinant. In Michael E. Saks, editor, *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, 5-7 January 1997, New Orleans, Louisiana, USA*, pages 730–738. ACM/SIAM, 1997. URL: http://dl.acm.org/citation.cfm?id=314161.314429.

[66] K. Mahler. An inequality for the discriminant of a polynomial. *Michigan Math. J.*, 11(3):257–262, 09 1964. doi:10.1307/mmj/1028999140.

[67] E. Mayr and A. Meyer. The complexity of word problem for commutative semigroups and polynomial ideals. *Adv. Math*, 46:305–329, 1982.

[68] Antoni W. Mazurkiewicz. Trace theory. In *Petri Nets: Central Models and Their Properties, Advances in Petri Nets 1986, Part II, Proceedings of an Advanced Course, Bad Honnef, Germany, 8-19 September 1986*, pages 279–324, 1986. doi:10.1007/3-540-17906-2_30.

[69] Thierry Mignon and Nicolas Ressayre. A quadratic bound for the determinant and permanent problem. *International Mathematics Research Notices*, 2004(79):4241–4253, 01 2004. arXiv:https://academic.oup.com/imrn/article-pdf/2004/79/4241/2375754/2004-79-4241.pdf, doi:10.1155/S1073792804142566.

[70] Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. *Comb.*, 7(1):105–113, 1987. doi:10.1007/BF02579206.

[71] Noam Nisan. Lower bounds for non-commutative computation (extended abstract). In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 410–418, 1991. URL: http://doi.acm.org/10.1145/103418.103462, doi:10.1145/103418.103462.

[72] Noam Nisan. Lower bounds for non-commutative computation (extended abstract). In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 410–418, 1991. URL: http://doi.acm.org/10.1145/103418.103462, doi:10.1145/103418.103462.

[73] E. M. Patterson. Free rings and their relations. *Bulletin of the London Mathematical Society*, 6(1):104–105, 1974. URL: https://londmathsoc.onlinelibrary.wiley.com/doi/abs/10.1112/blms/6.1.104, arXiv:https://londmathsoc.onlinelibrary.wiley.com/doi/pdf/10.1112/blms/6.1.104, doi:https://doi.org/10.1112/blms/6.1.104.

[74] Toniann Pitassi and Robert Robere. Strongly exponential lower bounds for monotone computation. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 1246–1255. ACM, 2017. doi:10.1145/3055399.3055478.

[75] Kevin Pratt. Faster algorithms via waring decompositions. *CoRR*, abs/1807.06194, 2018. URL: http://arxiv.org/abs/1807.06194, arXiv:1807.06194.

[76] Ran Raz and Pierre McKenzie. Separation of the monotone NC hierarchy. *Combinatorica*, 19(3):403–435, 1999. Preliminary version in FOCS 1997.

[77] Ran Raz and Amir Shpilka. Deterministic polynomial identity testing in non-commutative models. *Computational Complexity*, 14(1):1–19, 2005. URL: http://dx.doi.org/10.1007/s00037-005-0188-8, doi:10.1007/s00037-005-0188-8.

[78] Ran Raz and Avi Wigderson. Monotone circuits for matching require linear depth. *J. ACM*, 39(3):736–744, 1992.

[79] Ran Raz and Amir Yehudayoff. Multilinear formulas, maximal-partition discrepancy and mixed-sources extractors. *J. Comput. Syst. Sci.*, 77(1):167–190, 2011. doi:10.1016/j.jcss.2010.06.013.

[80] Alexander A. Razborov. A lower bound on the monotone network complexity of the logical permanent. *Mathematicheskie Garnetki*, 37(6):887–900, 1985. Mathematical notes of the Academy of Sciences of the USSR, 37:6, 485–493.

[81] Alexander A. Razborov. Lower bounds for the monotone complexity of some boolean functions. *Dokl. Ak. Nauk. SSSR*, 281:354–357, 1985. (in Russian) English translation in *Sov. Math. Dokl.*

[82] Alexander A. Razborov. On the distributional complexity of disjointness. *Theor. Comput. Sci.*, 106(2):385–390, 1992. Preliminary version in ICALP 1990. doi:10.1016/0304-3975(92)90260-M.

[83] Benjamin Rossman. Correlation bounds against monotone nc^1. In *30th Conference on Computational Complexity, CCC 2015, June 17-19, 2015, Portland, Oregon, USA*, volume 33 of *LIPIcs*, pages 392–411. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015.

[84] H.J. Ryser. *Combinatorial mathematics*. Carus mathematical monographs. Mathematical Association of America; distributed by Wiley [New York, 1963. URL: https://books.google.co.in/books?id=wOruAAAAMAAJ.

[85] Sanjeev Saluja. A note on the permanent value problem. *Information Processing Letters*, 43(1):1 – 5, 1992. URL: http://www.sciencedirect.com/science/article/pii/002001909290021M, doi:https://doi.org/10.1016/0020-0190(92)90021-M.

[86] Ramprasad Saptharishi. A survey of lower bounds in arithmetic circuit complexity. *Github Survey*. URL: https://github.com/dasarpmar/lowerbounds-survey/releases/, arXiv:https://github.com/dasarpmar/lowerbounds-survey/releases/.

[87] 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, Part I: Tack A: Algorithms, Automata, Complexity, and Games*, pages 60–71, 2008. doi:10.1007/978-3-540-70575-8_6.

[88] Nitin Saxena. Progress on polynomial identity testing. *Bull. EATCS*, 99:49–79, 2009.

[89] Nitin Saxena. Progress on polynomial identity testing - II. *Electron. Colloquium Comput. Complex.*, 20:186, 2013. URL: http://eccc.hpi-web.de/report/2013/186.

[90] Nitin Saxena and C. Seshadhri. From sylvester-gallai configurations to rank bounds: Improved blackbox identity test for depth-3 circuits. *J. ACM*, 60(5):33:1–33:33, 2013. URL: http://doi.acm.org/10.1145/2528403, doi:10.1145/2528403.

[91] C.P. Schnorr. A lower bound on the number of additions in monotone computations. *Theoretical Computer Science*, 2(3):305 – 315, 1976. URL: http://www.sciencedirect.com/science/article/pii/0304397576900839, doi:https://doi.org/10.1016/0304-3975(76)90083-9.

[92] M.P. Schützenberger. On the definition of a family of automata. *Information and Control*, 4(2):245 – 270, 1961. URL: http://www.sciencedirect.com/science/article/pii/S001999586180020X, doi:https://doi.org/10.1016/S0019-9958(61)80020-X.

[93] Jacob T. Schwartz. Fast probabilistic algorithm for verification of polynomial identities. *J. ACM.*, 27(4):701–717, 1980.

[94] Eli Shamir and Marc Snir. Lower bounds on the number of multiplications and additions in monotone computations. Technical report, IBM, 1977.

[95] Eli Shamir and Marc Snir. On the depth complexity of formulas. *Theory of Computing Systems*, 13(1):301–322, 1980.

[96] C. E. Shannon. The synthesis of two-terminal switching circuits. *The Bell System Technical Journal*, 28(1):59–98, 1949. doi:10.1002/j.1538-7305.1949.tb03624.x.

[97] 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. doi:10.1561/0400000039.

[98] Srikanth Srinivasan. Strongly exponential separation between monotone VP and monotone VNP. *Electron. Colloquium Comput. Complex.*, 26:32, 2019. URL: https://eccc.weizmann.ac.il/report/2019/032.

[99] Madhu Sudan. Lectures on algebra and computation : Lecture notes 6,12,13,14. 1998.

[100] Amnon Ta-Shma. Explicit, almost optimal, epsilon-balanced codes. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 238–251. ACM, 2017. doi:10.1145/3055399.3055408.

[101] Eva Tardos. The gap between monotone and non-monotone circuit complexity is exponential. *Combinatorica*, 8:141–142, 1988.

[102] Sébastien Tavenas. Improved bounds for reduction to depth 4 and depth 3. *Inf. Comput.*, 240:2–11, 2015. doi:10.1016/j.ic.2014.09.004.

[103] H. N. V. Temperley and Michael E. Fisher. Dimer problem in statistical mechanics-an exact result. *The Philosophical Magazine: A Journal of Theoretical Experimental and Applied Physics*, 6(68):1061–1063, 1961. arXiv:https://doi.org/10.1080/14786436108243366, doi:10.1080/14786436108243366.

[104] A. M. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, s2-42(1):230–265, 1937. URL: https://londmathsoc.onlinelibrary.wiley.com/doi/abs/10.1112/plms/s2-42.1.230, arXiv:https://londmathsoc.onlinelibrary.wiley.com/doi/pdf/10.1112/plms/s2-42.1.230, doi:https://doi.org/10.1112/plms/s2-42.1.230.

[105] W. Tzeng. A polynomial-time algorithm for the equivalence of probabilistic automata. *SIAM Journal on Computing*, 21(2):216–227, 1992.

[106] Leslie G. Valiant. The equivalence problem for deterministic finite-turn pushdown automata. *Information and Control*, 25(2):123 – 133, 1974. URL: http://www.sciencedirect.com/science/article/pii/S0019995874908390, doi:https://doi.org/10.1016/S0019-9958(74)90839-0.

[107] Leslie G. Valiant. Completeness classes in algebra. In Michael J. Fischer, Richard A. DeMillo, Nancy A. Lynch, Walter A. Burkhard, and Alfred V. Aho,

editors, *Proceedings of the 11h Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1979, Atlanta, Georgia, USA*, pages 249–261. ACM, 1979. doi:10.1145/800135.804419.

[108] Leslie G. Valiant. Negation can be exponentially powerful. *Theor. Comput. Sci.*, 12:303–314, 1980. Preliminary version in STOC 1979.

[109] Joachim von zur Gathen and Victor Shoup. Computing frobenius maps and factoring polynomials. *Computational Complexity*, 2:187–224, 1992.

[110] Moon J W. Counting labelled trees. *Canadian Mathematical Congress, Montreal*, 1970.

[111] Richard Ryan Williams. The polynomial method in circuit complexity applied to algorithm design (invited talk). In Venkatesh Raman and S. P. Suresh, editors, *34th International Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2014, December 15-17, 2014, New Delhi, India*, volume 29 of *LIPIcs*, pages 47–60. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2014. doi:10.4230/LIPIcs.FSTTCS.2014.47.

[112] Ryan Williams. Finding paths of length k in $O^*(2^k)$ time. *Inf. Process. Lett.*, 109(6):315–318, 2009. doi:10.1016/j.ipl.2008.11.004.

[113] Ryan Williams. Algorithms for circuits and circuits for algorithms. In *IEEE 29th Conference on Computational Complexity, CCC 2014, Vancouver, BC, Canada, June 11-13, 2014*, pages 248–261, 2014. doi:10.1109/CCC.2014.33.

[114] James Worrell. Revisiting the equivalence problem for finite multitape automata. In *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part II*, pages 422–433, 2013. doi:10.1007/978-3-642-39212-2_38.

[115] Amir Yehudayoff. Separating monotone VP and VNP. In Moses Charikar and Edith Cohen, editors, *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 425–429. ACM, 2019. doi:10.1145/3313276.3316311.

[116] R. Zippel. Probabilistic algorithms for sparse polynomials. In *Proc. of the Int. Sym. on Symbolic and Algebraic Computation*, pages 216–226, 1979.