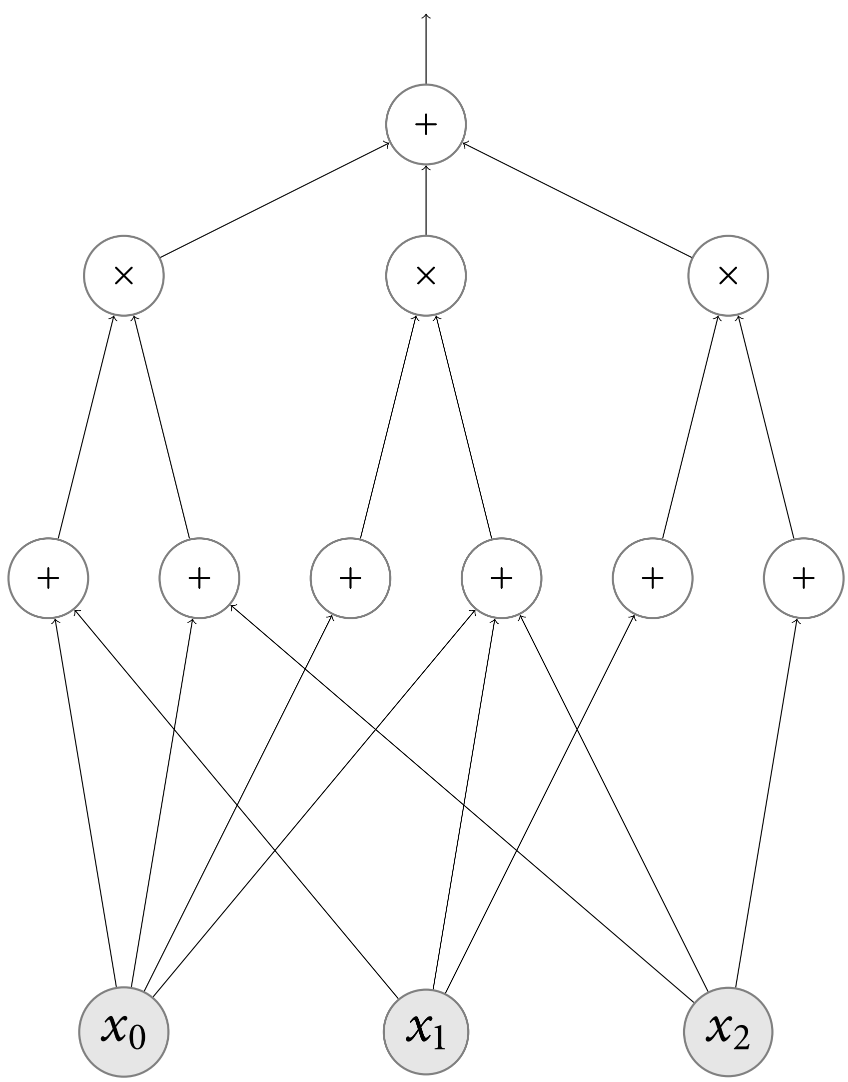
Arithmetic Circuits and Identity Testing



Ramprasad Saptharishi

ARITHMETIC CIRCUITS

AND

IDENTITY TESTING

Thesis submitted in
Partial Fulfilment of the
Master of Science (M.Sc.)
in Computer Science

by

Ramprasad Saptharishi

cmi

Chennai Mathematical Institute
Plot H1 SIPCOT IT Park
Padur PO, Siruseri 603 103

May 2009

Abstract

We study the problem of *polynomial identity testing* (PIT) in arithmetic circuits. This is a fundamental problem in computational algebra and has been very well studied in the past few decades. Despite many efforts, a deterministic polynomial time algorithm is known only for restricted circuits of depth 3. A recent result of Agrawal and Vinay show that PIT for depth 4 circuit is almost as hard as the general case, and hence explains why there is no progress beyond depth 3. The main contribution of this thesis is a new approach to designing a polynomial time algorithm for depth 3 circuits.

We first provide the background and related results to motivate the problem. We discuss the connections of PIT with arithmetic circuit lower bounds, and also briefly the results in depth reduction. We then look at the deterministic algorithms for PIT on restricted circuits.

We then proceed to the main contribution of the thesis which studies the power of arithmetic circuits over higher dimensional algebras. We consider the model of $\Pi\Sigma$ circuits over the algebra of upper-triangular matrices and show that PIT in this model is equivalent to identity testing of depth three circuits over fields. Further we also show that $\Pi\Sigma$ circuits over upper-triangular matrices is computationally very weak.

In the case when the underlying algebra is a constant dimensional commutative algebra, we present a polynomial time algorithm. PITs on arbitrary dimensional commutative algebras, however, are as hard as PIT on depth 3 circuits.

Thus $\Sigma\Pi$ circuits over upper-triangular matrices, the smallest non-commutative algebra, captures PIT on depth 3 circuits. We hope that ideas used in the commutative case would be useful to design a polynomial time identity test for depth 3 circuits.

Acknowledgements

I'd like to thank my advisor Manindra Agrawal for giving me absolute freedom regarding the contents of this thesis and otherwise. This independence helped me mull over the subject more and hence understand it better.

Most of the work over the last two years were done at IIT Kanpur. Manindra has been extremely helpful in arranging accommodation, despite severe shortage in hostel space. The faculty and students at IITK have been very friendly. I greatly benefited from various discussions with faculty like Piyush Kurur, Somenath Biswas, Sumit Ganguly, Shashank Mehta, and all SIGTACS members. Chandan Saha, in particular, has been subjected to numerous hours of discussion, besides other torture, by me over the last two years. I'm very grateful to him, and Nitin Saxena, for having me a part of their work.

I'm deeply indebted to Chennai Mathematical Institute and the Institute of Mathematical Sciences for a truly wonderful undergraduate course. My sincere thanks Madhavan Mukund, Narayan Kumar and V Vinay, who were instrumental in my decision to join CMI. My interaction with Samir Dutta, V Arvind and Meena Mahajan were also a great inspiration to make me work in complexity theory and computational algebra. S P Suresh (LKG) introduced me to logic. Thanks to him for enlightening discussions on research(!), also for helping me get these lovely fonts (Minion Pro and Myriad Pro (for sans-serif)) for LaTeX.

Besides excellent professors, CMI is endowed with a very efficient and affable office staff as well. Sripathy, Vijayalakshi and Rajeshwari (a.k.a Goach) do a spectacular job at administration. I'm very grateful to them for a completely hassle free stay at CMI.

Thanks to V Vinay for — well everything. He has been my primary motivation to do mathematics and theoretical computer science. My salutations to him, and to my aunt Subhashini, for countless pieces of advice over all these years.

Salutations to my father, L C Saptharishi for giving me complete liberty in whatever I do. He is, by far, the one who is proudest of what I am. My humble namaskarams to him, and everyone at home — amma, pati, thatha; my sister Kamakshi gets a not-an-elder version of a namaskaram.

Since it would be very difficult to make the list exhaustive, let me just thank everyone that need be thanked.

Contents

1 Introduction

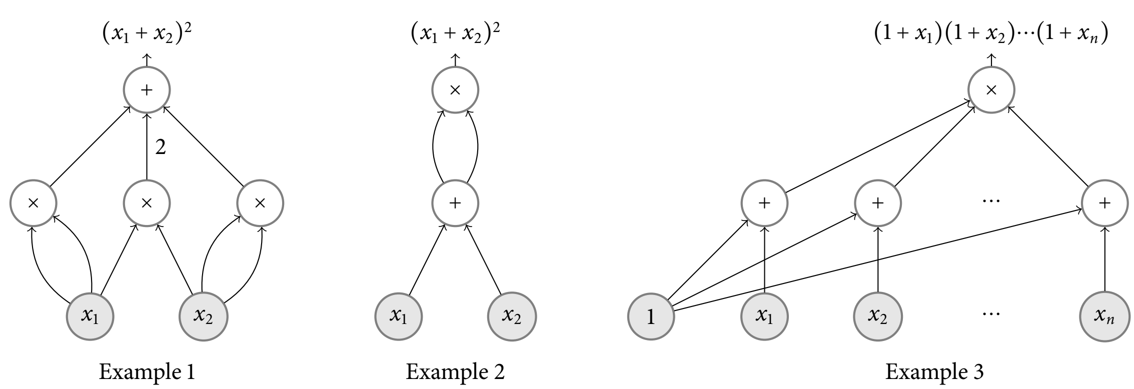
The interplay between mathematics and computer science demands algorithmic approaches to various algebraic constructions. The area of computational algebra addresses precisely this. The most fundamental objects in algebra are polynomials and it is natural to desire a classification based on their "simplicity". The algorithmic approach suggests that the simple polynomials are those that can be computed easily. The ease of computation is measured in terms of the number of arithmetic operations required to compute the polynomial. This yields a very robust definition of simple (and hard) polynomials that can be studied analytically.

It is worth remarking that the number of terms in a polynomial is not a good measure of its simplicity. For example, consider the polynomials

$$\begin{aligned} f_1(x_1,\cdots,x_n) &= (1+x_1)(1+x_2)\cdots(1+x_n) \\ f_2(x_1,\cdots,x_n) &= (1+x_1)(1+x_2)\cdots(1+x_n) - x_1 - x_2 - \cdots - x_n. \end{aligned}$$

The former has n more terms than the later, however, the former is easier to describe as well as compute.

We use *arithmetic circuits* (formally defined in the Section 1.1) to represent the computation of a polynomial. This also allows us to count the number of operations required in computation. A few examples are the following:



The *complexity* of a given polynomial is defined as the *size* (the number of operations) of the smallest arithmetic circuit that computes the polynomial.

With this definition, how do we classify "simple" polynomials? For this, we need to define the intuitive notion of "polynomials that can be computed easily". Following standard ideas from computer science, we call any polynomial that can be computed using at most $n^{O(1)}$ operations an easy polynomial. Strictly speaking, this definition applied to an infinite family of polynomials over n variables, one for each n, as otherwise every polynomial can be computed using $O(1)$ operations rendering the whole exercise meaningless. However, often we will omit to explicitly mention the infinite family to which a polynomial belongs when talking about its complexity; the family would be obvious.

The class VP is the class of *low degree*[1] polynomial families that are easy in the above sense. The polynomials in VP are essentially represented by the *determinant* polynomial: the determinant of an $n \times n$ matrix whose entries are affine linear combinations of variables. It is known that determinant polynomial belongs to the class VP [Sam42, Ber84, Chi85, MV97] and any polynomial in VP over n variables can be written as a determinant polynomial of a $m \times m$ matrix with $m = n^{O(\log n)}$ [Tod91].

This provides a excellent classification of easy polynomials. Hence, any polynomial that cannot be written as a determinant of a small sized matrix is not easy. A simple counting argument shows that there exist many such polynomials. However, proving an explicitly given polynomial to be hard has turned out to be a challenging problem which has not been solved yet. In particular, the *permanent* polynomial, the permanent of a matrix with affine linear entries, is believed to be very hard to compute — requiring $2^{\Omega(n)}$-size circuits for an $n \times n$ matrix in general. However, there is no proof yet of this. It is not even known if it requires $\omega(n^2)$ operations!

A general way of classifying a given polynomial is to design an algorithm that, given a polynomial as an arithmetic circuit as input, outputs the smallest size arithmetic circuit computing the polynomial. Such an algorithm is easy to design: given an arithmetic circuit C computing a polynomial, run through all the smaller circuits D and check if any computes the same polynomial (this check can be performed easily as we discuss in the next paragraph). However, the algorithm is not efficient: it will take exponential time (in the size of the given circuit) to find the classification of a polynomial. No efficient algorithm for this is known; further, it is believed that no efficient algorithm exists.

[1] A polynomial is said to have low degree if its degree is less than the size of the circuit

A closely related problem that occurs above is to check if given two arithmetic circuits C and D compute the same polynomial. The problem is equivalent to asking if the circuit $C - D$ is the zero polynomial or not. This problem of checking if a circuit computes the zero polynomial is called *polynomial identity testing* (PIT). It turns out that this problem is easy to solve algorithmically. We give later several randomized polynomial time algorithms for solving it. Moreover, in a surprising connection, it has been found that if there is a deterministic polynomial time algorithm for solving PIT, then certain explicit polynomials are hard to compute [KI03, Agr05]! Therefore, the solution to PIT problem has a key role in our attempt to computationally classify polynomials. In this article, we will focus on this connection between PIT and polynomial classification.

We now formally define arithmetic circuits and the identity testing problem.

1.1 Problem definition

We shall fix an underlying field $\mathbb{F}$.

***Definition* 1.1** (Arithmetic Circuits and formulas)**.** *An* arithmetic circuit *is a directed acyclic graph with one sink (which is called the output gate). Each of the source vertices (which are called input nodes) are either labelled by a variable x_i or an element from an underlying field $\mathbb{F}$. Each of the internal nodes are labelled either by $+$ or $\times$ to indicate if it is an addition or multiplication gate respectively. Sometimes edges may carry weights that are elements from the field.*

Such a circuit naturally computes a multivariate polynomial at every node. The circuit is said to compute a polynomial $f \in \mathbb{F}[x_1, \cdots, x_n]$ if the output node computes f. Sometimes edges may carry weights that are elements from the field.

If the underlying field $\mathbb{F}$ has characteristic zero, then a circuit is said to be monotone *if none of the constants are negative.*

An arithmetic circuit is a formula *if every internal node has out-degree* 1.

Without loss of generality, the circuit is assumed to be layered, with edges only between successive layers. Further, it is assumed it consists of alternating layers of addition and multiplication gates. A layer of addition gates is denoted by Σ and that of multiplication gates by Π.

Some important parameters of an arithmetic circuit are the following:

- *Size*: the number of gates in the circuit
- *Depth*: the longest path from a leaf gate to the output gate
- *Degree*: the syntactic degree of the polynomial computed at the output gate. This is computed recursively at every gate in the most natural way (max of the degrees of children at an addition gate, and the sum of the degrees at a multiplication gate).

  This needn't be the degree of the polynomial computed at the output gate (owing to cancellations) but this is certainly an upper bound.

A circuit evaluating a polynomial provides a succinct representation of the polynomial. For instance, in Example 3, though the polynomial has 2^n terms, we have a circuit size $O(n)$ computing the polynomial. The PIT problem is deciding if a given succinct representation is zero or not.

Also, a circuit of size s can potential compute a polynomial of exponential degree. But usually in identity testing, it is assumed that the degree of the polynomial is $O(n)$ where n is the number of variables. Most interesting polynomials, like the determinant or permanent, satisfy this property.

***Problem* 1.2** (Polynomial Identity Testing)**.** *Given an arithmetic circuit C with input variables $x_1, \ldots, x_n$ and constants taken from a field $\mathbb{F}$, check if the polynomial computed is identically zero.*

The goal is to design a deterministic algorithm for PIT that runs in time polynomial in n, size of C and $|\mathbb{F}|$. A much stronger algorithm is one that doesn't look into the structure of the circuit at all, but just evaluates it at chosen input points. Such an algorithm that just uses the circuit as a "black box" is hence called a *black-box algorithm*.

1.2 Current Status

A likely candidate of a *hard* polynomial is the *permanent* polynomial. It is widely believed that it requires circuits of exponential size, but this is still open. However, progress has been made in restricted settings. Raz and Yehudayoff [RY08] showed that monotone circuits for permanent require exponential size. Nisan and Wigderson [NW95] showed that "homogeneous" depth 3 circuits for the $2d$-th symmetric polynomial requires $\left(\frac{n}{4d}\right)^{\Omega(d)}$ size. Shpilka and Wigderson [SW99] showed that depth 3 circuits for determinant or permanent over $\mathbb{Q}$ require quadratic size. Over finite fields, Grigoriev and Karpinsky [GK98] showed that determinant or permanent required exponential sized depth 3 circuit.

As for PIT, the problem has drawn significant attention due to its role in various fields of theoretical computer science. Besides being a natural problem in algebraic computation, identity testing has found applications in various fundamental results like Shamir's IP = PSPACE [Sha90], the PCP Theorem [ALM+98] etc. Many other important results such as the AKS Primality Test [AKS04], check if some special polynomials are identically zero or not. Algorithms for graph matchings [Lov79] and multivariate polynomial interpolation [CDGK91] also involve identity testing. Another promising role of PIT is its connection to the question of "hardness of polynomials". It is known that strong algorithms for PIT can be used to construct polynomials that are *very hard* [KI03, Agr05].

There is a score of randomized algorithms proposed for PIT. The first randomized polynomial time algorithm for identity testing was given by Schwartz and Zippel [Sch80, Zip79]. Several other *randomness-efficient* algorithms [CK97, LV98, AB99, KS01] came up subsequently, resulting in a significant improvement in the number of random bits used. However, despite numerous attempts a deterministic polynomial time algorithm has remained unknown. Nevertheless, important progress has been made both in the designing of deterministic algorithms for special circuits, and in the understanding of why a general deterministic solution could be hard to get.

Kayal and Saxena [KS07] gave a deterministic polynomial time identity testing algorithm for depth 3 ($\Sigma\Pi\Sigma$) circuits with constant top fan-in (the top addition gate has only constantly many children). When the underlying field if $\mathbb{Q}$, this was further improved to a black-box algorithm by Kayal and Saraf [KS09]. Saxena [Sax08] gave a polynomial time algorithm for a restricted form of depth 3 circuits called "diagonal circuits". As such, no polynomial time PIT algorithm is known for general depth 3 circuits.

Most of the progress made appears to stop at around depth 3. A "justification" behind the hardness of PIT even for small depth circuits was provided recently by Agrawal and Vinay [AV08]. They showed that a deterministic polynomial time black-box identity test for depth 4 ($\Sigma\Pi\Sigma\Pi$) circuits would imply a quasi-polynomial ($n^{O(\log n)}$) time deterministic PIT algorithm for any circuit computing a polynomial of low degree. Thus, PIT for depth 4 circuits over a field is *almost* the general case.

Thus we see that the non-trivial case for identity testing starts with depth 3 circuits; whereas circuits of depth 4 are *almost* the general case. Thus, the first step to attack PIT is general $\Sigma\Pi\Sigma$ circuits.

It is natural to ask what is the complexity of PIT, if the constants of the circuit are taken from a finite dimensional algebra over $\mathbb{F}$. We shall assume that the algebra is given in its basis form.

1.3 Circuits over algebras

The PIT problem for depth two circuits over an algebra is the following:

***Problem* 1.3.** *Given an expression,*

$$P = \prod_{i=1}^{d} (A_{i0} + A_{i1}x_1 + \ldots + A_{in}x_n)$$

where $A_{ij} \in \mathcal{R}$, an algebra over $\mathbb{F}$ given in basis form, check if P is zero.

Since elements of a finite dimensional algebra, given in basis form, can be expressed as matrices over $\mathbb{F}$ we can equivalently write the above problem as,

***Problem* 1.4.** *Given an expression,*

$$P = \prod_{i=1}^{d} (A_{i0} + A_{i1}x_1 + \ldots + A_{in}x_n) \tag{1.1}$$

where $A_{ij} \in \mathcal{M}_k(\mathbb{F})$, the algebra of $k \times k$ matrices over $\mathbb{F}$, check if P is zero using $\mathsf{poly}(k \cdot n \cdot d)$ number of $\mathbb{F}$-operations.

A result by Ben-Or and Cleve [BC88] shows that any polynomial computed by an arithmetic formula of size s can be computed by an expression as in Equation 1.1 consisting of 3×3 matrices. Therefore, solving Problem 1.4 for $k = 3$ is *almost* the general case. Therefore, it is natural to ask how the complexity of PIT for depth 2 circuits over $\mathcal{M}_2(\mathbb{F})$ relates to PIT for arithmetic circuits.

In this thesis, we provide an answer to this. We show a connection between PIT of depth 2 circuits over $\mathsf{U}_2(\mathbb{F})$, the algebra of upper-triangular 2×2 matrices, and PIT of depth 3 circuits over fields. The reason this is a bit surprising is because we also show that, a depth 2 circuit over $\mathsf{U}_2(\mathbb{F})$ is not even powerful enough to compute a simple polynomial like, $x_1x_2 + x_3x_4 + x_5x_6$!

1.4 Contributions of the thesis

A depth 2 circuit C over matrices, as in Equation 1.1, naturally defines a computational model. Assuming $\mathcal{R} = \mathcal{M}_k(\mathbb{F})$ for some k, a polynomial $P \in \mathcal{R}[x_1, \ldots, x_n]$ outputted by C can be viewed as a $k \times k$ matrix of polynomials in $\mathbb{F}[x_1, \ldots, x_n]$. We say that a polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ is *computed* by C if one of the k^2 polynomials in P is f. Sometimes we would abuse terminology a bit and say that P *computes* f to mean the same.

Our main results are of two types. Some are related to identity testing and the rest are related to the weakness of the depth 2 computational model over $\mathsf{U}_2(\mathbb{F})$ and $\mathcal{M}_2(\mathbb{F})$.

1.4.1 Identity testing

We fill in the missing information about the complexity of identity testing for depth 2 circuits over 2×2 matrices by showing the following result.

***Theorem* 1.5.** *Identity testing for depth* 2 *(*$\Pi\Sigma$*) circuits over* $\mathsf{U}_2(\mathbb{F})$ *is polynomial time equivalent to identity testing for depth* 3 *(*$\Sigma\Pi\Sigma$*) circuits.*

The above result has an interesting consequence on identity testing for Algebraic Branching Program (ABP) [Nis91]. It is known that identity testing for non-commutative ABP can be done in deterministic polynomial time (a result due to Raz and Shpilka [RS04]). But no result is known for identity testing of even width-2 commutative ABP's. The following result explains why this is the case.

***Corollary* 1.6.** *Identity testing of depth* 3 *circuits is equivalent to identity testing of width-*2 *ABPs with polynomially many paths from source to sink.*

Further, we give a deterministic polynomial time identity testing algorithm for depth 2 circuits over any constant dimensional commutative algebra given in basis form.

***Theorem* 1.7.** *Given an expression,*

$$P = \prod_{i=1}^{d} (A_{i0} + A_{i1}x_1 + \ldots + A_{in}x_n)$$

where $A_{ij} \in \mathcal{R}$, a commutative algebra of constant dimension over $\mathbb{F}$ that is given in basis form, there is a deterministic polynomial time algorithm to test if P is zero.

In a way, this result establishes the fact that the power of depth 2 circuits is primarily derived from the non-commutative structure of the underlying algebra.

It would be apparent from the proof of Theorem 1.5 that our argument is simple in nature. Perhaps the reason why such a connection was overlooked before is that, unlike a depth 2 circuit over $\mathcal{M}_3(\mathbb{F})$, we do not always have the privilege of *exactly* computing a polynomial over $\mathbb{F}$ using a depth 2 circuit over $\mathsf{U}_2(\mathbb{F})$. Showing this weakness of the latter computational model constitutes the other part of our results.

1.4.2 Weakness of the depth 2 model over $\mathsf{U}_2(\mathbb{F})$ and $\mathcal{M}_2(\mathbb{F})$

Although Theorem 1.5 shows an equivalence of depth 3 circuits and depth 2 circuits over $\mathsf{U}_2(\mathbb{F})$ with respect to PIT, the computational powers of these two models are very different. The following result shows that a depth 2 circuit over $\mathsf{U}_2(\mathbb{F})$ is computationally strictly weaker than depth 3 circuits.

***Theorem* 1.8.** *Let $f \in F[x_1, \ldots, x_n]$ be a polynomial such that there are no two linear functions l_1 and l_2 (with $1 \notin (l_1, l_2)$, the ideal generated by l_1 and l_2) which make $f \bmod (l_1, l_2)$ also a linear function. Then f is not computable by a depth 2 circuit over $\mathsf{U}_2(\mathbb{F})$.*

It can be shown that even a simple polynomial like $x_1x_2 + x_3x_4 + x_5x_6$ satisfies the condition stated in the above theorem, and hence it is not computable by any depth 2 circuit over $\mathsf{U}_2(\mathbb{F})$, no matter how large! This contrast makes Theorem 1.5 surprising as it establishes an equivalence of identity testing in two models of different computational strengths.

At this point, it is natural to investigate the computational power of depth 2 circuits if we graduate from $\mathsf{U}_2(\mathbb{F})$ to $\mathcal{M}_2(\mathbb{F})$. The following result hints that even such a model is severely restrictive in nature.

A depth 2 circuit over $\mathcal{M}_2(\mathbb{F})$ computes $P = \prod_{i=1}^{d}(A_{i0} + A_{i1}x_1 + \ldots + A_{in}x_n)$ with $A_{ij} \in \mathcal{M}_2(\mathbb{F})$. Let $P_\ell = \prod_{i=\ell}^{d}(A_{i0} + A_{i1}x_1 + \ldots + A_{in}x_n)$, where $\ell \leq d$, denote the partial product.

***Definition* 1.9.** *A polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ is computed by a depth 2 circuit over $\mathcal{M}_2(\mathbb{F})$* under a degree restriction *of m if the degree of each of the partial products P_ℓ is bounded by m.*

***Theorem* 1.10.** *There exists a class of polynomials of degree n that cannot be computed by a depth 2 circuit over $\mathcal{M}_2(\mathbb{F})$, under a degree restriction of n.*

The motivation for imposing a condition like degree restriction comes very naturally from depth 2 circuits over $\mathcal{M}_3(\mathbb{F})$. Given a polynomial $f = \sum_i m_i$, where m_i's are the monomials of f, it is easy to construct a depth 2 circuit over $\mathcal{M}_3(\mathbb{F})$ that literally forms these monomials and adds then one by one. This computation is degree restricted, if we extend our definition of degree restriction to $\mathcal{M}_3(\mathbb{F})$. However, the above theorem suggests that no such scheme to compute f would succeed over $\mathcal{M}_2(\mathbb{F})$.

Remark- By transferring the complexity of an arithmetic circuit from its depth to the dimension of the underlying algebras while fixing the depth to 2, our results provide some evidence that identity testing for depth 3 circuits appears to be *mathematically* more tractable than depth 4 circuits. Besides, it might be possible to exploit the properties of these underlying algebras to say something useful about identity testing. A glimpse of this indeed appears in our identity testing algorithm over commutative algebras.

1.5 Organization of the thesis

In chapter 2, we look at some randomized algorithms for identity testing. We then proceed to some deterministic algorithms for restricted circuits in chapter 3. Chapter 4 covers the result by Agrawal and Vinay[AV08] to help understand why depth 4 circuits are "as hard as it can get". We then move to study circuits over algebras, the main contribution of the thesis, in chapter 5, and then conclude in chapter 6.

Randomized Algorithms for PIT

2

Though deterministic algorithms for PIT have remained elusive, a number of randomized solutions are available. Quite an extensive study has been made on reducing the number of random bits through various techniques. In this chapter, we shall inspect a few of them. We start with the simplest and the most natural test.

2.1 The Schwarz-Zippel test

The Schwarz-Zippel test is the oldest algorithm for PIT. The idea of the test is that, if the polynomial computed is non-zero then the value of the polynomial at a random point would probably be non-zero too. This intuition is indeed true.

***Lemma* 2.1.** *[Sch80, Zip79]Let $p(x_1, \cdots, x_n)$ be a polynomial over $\mathbb{F}$ of total degree d. Let S be any subset of $\mathbb{F}$ and let $a_1, \cdots, a_n$ be randomly and independently chosen from S. Then,*

$$\Pr_{a_1,\cdots,a_n}[p(a_1, a_2, \cdots, a_n) = 0] \leq \frac{d}{|S|}$$

Proof. The proof proceeds by induction on n. For the base case when $n = 1$, we have a univariate polynomial of degree d and hence has at most d roots. Therefore, the probability that $p(a) = 0$ at a randomly chosen a is at most $\frac{d}{|S|}$.

For $n > 1$, rewrite $p(x_1, \cdots, x_n) = p_0 + p_1 x_n + p_2 x_n^2 \cdots p_k x_n^k$ where each p_i is a polynomial over the variables $x_1, \cdots, x_{n-1}$ and k is the degree of x_n in p. Then,

$$\begin{aligned}
\Pr[p(a_1, \cdots, a_n) = 0] &\leq \Pr[p(a_1, \cdots, a_{n-1}, x_n) = 0] \\
&\quad + \Pr[p(a_1, \cdots, a_n) = 0 | p(a_1, \cdots, a_{n-1}, x_n) \neq 0] \\
&\leq \Pr[p_k(a_1, \cdots, a_{n-1}) = 0] \\
&\quad + \Pr[p(a_1, \cdots, a_n) = 0 | p(a_1, \cdots, a_{n-1}, x_n) \neq 0] \\
&\leq \frac{d-k}{|S|} + \frac{k}{|S|} = \frac{d}{|S|}
\end{aligned}$$

□

If we wish to get the error less than ε, then we need a set S that is as large as $\frac{d}{\varepsilon}$. Hence, the total number of random bits required would be $n \cdot \log \frac{d}{\varepsilon}$.

2.2 Chen-Kao: Evaluating at irrationals

Let is consider the case where the circuit computes an integer polynomial. We wish to derandomize the Schwarz-Zippel test by finding out a small number of special points on which we can evaluate and test if the polynomial is non-zero. Suppose the polynomial was univariate, can we find out a single point on which we can evaluate to test if it is non-zero? Indeed, if we evaluate it at some transcendental number like π; $p(\pi) = 0$ for an integer polynomial if and only if $p = 0$. More generally, if we can find suitable irrationals such that there exists no degree d multivariate relation between them, we can use those points to evaluate and test if p is zero or not. This is the basic idea in Chen-Kao's paper [CK97].

However, it is infeasible to actually evaluate at irrational points since they have infinitely many bits to represent them. Chen and Kao worked with approximations of the numbers, and introduced randomness to make their algorithm work with high probability.

2.2.1 Algebraically d-independent numbers

The goal is to design an identity test for all n-variate polynomials whose degree in each variable is less than d. The following definition is precisely what we want for the identity test.

***Definition* 2.2** (Algebraically d-independence). *A set of number $\{\pi_1, \cdots, \pi_n\}$ is said to be algebraically d-independent over $\mathbb{F}$ if there exists no polynomial relation $p(\pi_1, \cdots, \pi_n) = 0$ over $\mathbb{F}$ with the degree of $p(x_1, \cdots, x_n)$ in each variable bounded by d.*

It is clear that if we can find such a set of numbers then this is a single point that would be non-zero at all non-zero polynomials with degree bounded by d. The following lemma gives an explicit construction of such a point.

***Lemma* 2.3.** *Set $k = \log(d+1)$ and $K = nk$. Let $p_{11}, p_{12}, \cdots, p_{1k}, p_{2k} \cdots, p_{nk}$ be first K distinct primes and let $\pi_i = \sum_{j=1}^{k} \sqrt{p_{ij}}$. Then $\{\pi_1, \cdots, \pi_n\}$ is algebraically d-independent.*

Proof. Let $A_0 = B_0 = \mathbb{Q}$ and inductively define $A_i = A_{i-1}(\pi_i)$ and $B_i = B_{i-1}(\sqrt{p_{i1}}, \cdots, \sqrt{p_{ik}})$. We shall prove by induction that $A_n = B_n$. Of course it is clear when $i = 0$. Assuming that

$A_{i-1} = B_{i-1}$, we now want to show that $A_i = B_i$. It is of clear that $A_i \subseteq B_i$. Since none of the square roots $\left\{\sqrt{p_{ij}}\right\}_j$ lie in B_{i-1}, we have that B_i is a degree 2^k extension over B_{i-1}. And if $\pi_i' = \sum_j \alpha_j \sqrt{p_{ij}}$ where each $\alpha_j = \pm 1$, we have an automorphism of A_i that fixes A_{i-1} and sends π to π_i'. Since there are 2^k distinct automorphisms, we have that A_i is indeed equal to B_i and is a 2^k degree extension over A_{i-1}.

Hence there is no univariate polynomial $p_i(x)$ over A_{i-1} with degree bounded by d such that $p_i(\pi_i) = 0$. Unfolding the induction, there is no polynomial $p(x_1, \cdots, x_n)$ over $\mathbb{Q}$ with degree in each variable bounded d such that $p(\pi_1, \cdots, \pi_n) = 0$. □

2.2.2 Introducing randomness

As remarked earlier, it is not possible to actually evaluate the polynomial at these irrational points. Instead we consider approximations of these irrational points and evaluate them. However, we can no longer have the guarantee that all non-zero polynomials will be non-zero at this truncated value. Chen and Kao solved this problem by introducing randomness in the construction of the π_i's.

It is easy to observe that Lemma 2.3 would work even if each $\pi_i = \sum_{j=1}^{k} \alpha_{ij} \sqrt{p_{ij}}$ where each $\alpha_{ij} = \pm 1$. Randomness is introduced by setting each α_{ij} to ± 1 independently and uniformly at random, and then we evaluate the polynomial at the $\pi_i's$ truncated to ℓ decimal places.

Chen and Kao showed that if we want the error to be less than ε, we would have to choose $\ell \geq d^{O(1)} \log n$. Randomness used in this algorithm is for choosing the α_{ij}'s and hence $n \log d$ random bits are used[1]; this is independent of ε! Therefore, to get better accuracy, we don't need to use a single additional bit of randomness but just need to look at better approximations of π_i's.

2.2.3 Chen-Kao over finite fields

Though the algorithm that is described seems specific to polynomials over $\mathbb{Q}$, they can be extended to finite fields as well. Lewin and Vadhan [LV98] showed how use the same idea over finite fields. Just like of using square root of prime numbers in $\mathbb{Q}$, they use square roots of irreducible polynomials. The infinite decimal expansion is paralleled by the infinite

[1]In fact, if the degree in x_i is bounded by d_i, then $\sum_i \log(d_i + 1)$ random bits would be sufficient

power series expansion of the square roots, with the approximation as going modulo x^ℓ.

Lewin and Vadhan achieve more or less the exact same parameters as in Chen-Kao and involves far less error analysis as they are working over a discrete finite field. They also present another algorithm that works over integers by considering approximations over p-adic numbers, i.e. solutions modulo p^ℓ. This again has the advantage that there is little error analysis.

2.3 Agrawal-Biswas: Chinese Remaindering

Agrawal and Biswas [AB99] presented a new approach to identity testing via Chinese remaindering. This algorithm works in randomized polynomial time in the size of the circuit and also achieves the time-error tradeoff as in the algorithm by Chen and Kao. It works over all fields but we present the case when it is a finite field $\mathbb{F}_q$.

The algorithm proceeds in two steps. The first is a deterministic conversion to a univariate polynomial of exponential degree. The second part is a novel construction of a sample space consisting of "almost coprime" polynomials that is used for chinese remaindering.

2.3.1 Univariate substitution

Let $f(x_1, \cdots, x_n)$ be the polynomial given as a circuit of size s. Let the degree of f in each variable be less than d. The first step is a conversion to a univariate polynomial of exponential degree that maps distinct monomials to distinct monomials. The following substitution achieves this

$$x_i = y^{d^i}$$

***Claim* 2.4.** *Under this substitution, distinct monomials go to distinct monomials.*

Proof. Each monomial of f is of the form $x_1^{d_1} \cdots x_n^{d_n}$ where each $d_i < d$. Hence each monomial can be indexed by d-bit number represented by $\langle d_1, \cdots, d_n \rangle$ and this is precisely the exponent it is mapped to. □

Let the univariate polynomial thus produced by $P(x)$ and let the degree be D. We now wish to test that this univariate polynomial is non-zero. This is achieved by picking a polynomial $g(x)$ from a suitable sample space and doing all computations in the circuit modulo $g(x)$ and return zero if the polynomial is zero modulo $g(x)$.

Suppose these $g(x)$'s came from a set such that the lcm of any ε fraction of them has degree at least D, then the probability of success would be $1-\varepsilon$. One way of achieving this is to choose a very large set of mutually coprime polynomials say $\{(z-\alpha) : \alpha \in \mathbb{F}_q\}$. But if every epsilon fraction of them must have an lcm of degree D, then the size of the sample space must be at least $\frac{D}{\varepsilon}$. This might force us to go to an extension field of $\mathbb{F}_q$ and thus require additional random bits. Instead, Agrawal and Biswas construct a sample space of polynomials that share very few common factors between them which satisfies the above property.

2.3.2 Polynomials sharing few factors

We are working with the field $\mathbb{F}_q$ of characteristic p. For a prime number r, let $Q_r(x) = 1 + x + \cdots + x^{r-1}$. Let $\ell \geq 0$ be a parameter that would be fixed soon. For a sequence of bits $b_0, \cdots, b_{\ell-1} \in \{0,1\}$ and an integer $t \geq \ell$, define

$$\begin{aligned} A_{b,t}(x) &= x^t + \sum_{i=0}^{\ell-1} b_i x^i \\ T_{r,b,t}(x) &= Q_r(A_{b,t}(x)) \end{aligned}$$

The space of polynomials consists of $T_{r,b,t}$ for all values of b, having suitably fixed r and t.

***Lemma* 2.5.** *Let $r \neq p$ be a prime such that r does not divide any of $q-1, q^2-1, \cdots, q^{\ell-1}-1$. Then, $T_{r,b,t}(x)$ does not have any factor of degree less than ℓ, for any value of b and t.*

Proof. Let $U(x)$ be an irreducible factor of $T_{r,b,t}(x)$ of degree δ and let α be a root of $U(x)$. Then, $F_q(\alpha)$ is an extension field of degree δ. Therefore, $Q_r(A_{b,t}(\alpha)) = 0$ as $U(x)$ divides $T_{r,b,t}(x)$. If $\beta = A_{b,t}(\alpha)$, then $Q_r(\beta) = 0$. Since $r \neq p$, we have $\beta \neq 1$. And $\beta^r = 1$ in $\mathbb{F}_{q^\delta}$ therefore $r \mid q^\delta - 1$ which forces $\delta \geq \ell$. □

***Lemma* 2.6.** *Let r be chosen as above. Then for any fixed t, a polynomial $U(x)$ can divide $T_{r,b,t}(x)$ for at most $r-1$ many values of b.*

Proof. Let $U(x)$ be an irreducible polynomial that divides $T_{r,b_1,t}(x), \cdots, T_{r,b_k,t}(x)$ and let the degree of $U(x)$ be δ. As earlier, consider the field extension $\mathbb{F}_{q^\delta} = \mathbb{F}_q(\alpha)$ where α is a root of $U(x)$. Then, $A_{b_i,t}(\alpha)$ is a root of $Q_r(x)$ for every $1 \leq i \leq k$.

Suppose $A_{b_i,t}(\alpha) = A_{b'_i,t}(\alpha)$ for $b_i \neq b'_i$, then α is a root of $A_{b_i,t}(x) - A_{b'_i,t}(x)$ which whose degree is less than ℓ. This isn't possible as $U(x)$ is the minimum polynomial of α and by Lemma 2.5 must have degree at least ℓ. Therefore, $\{A_{b_i,t}(\alpha)\}_{1\leq i\leq k}$ are infact distinct roots of $Q_r(x)$. Hence clearly $k \leq r-1$. □

***Lemma* 2.7.** *Let r be chosen as above and let t be fixed. Then, the lcm of any K polynomials from the set has degree at least $K \cdot t$.*

Proof. The degree of the product of any K of them is $K \cdot t \cdot (r-1)$. And by Lemma 2.6, any $U(x)$ can be a factor of at most $r-1$ of them and hence the claim follows. □

The algorithm is now straightforward:

1. Set parameters $\ell = \log D$ and $t = \max\left\{\ell, \frac{1}{\varepsilon}\right\}$.
2. Let r is chosen as the smallest prime that doesn't divide any of p, $q-1$, q^2-1, $\cdots$, $q^{\ell-1}-1$.
3. Let $b_0, b_1, \cdots, b_{\ell-1}$ be randomly and uniformly chosen from $\{0,1\}$.
4. Compute $P(x)$ modulo $T_{r,b,t}(x)$ and accept if and only if $P(x) = 0 \bmod T_{r,b,t}(x)$.

Since $P(x)$ was obtained from a circuit of size S, we have $D \leq 2^s$. It is easy to see that the algorithm runs in time poly $\left(s, \frac{1}{\varepsilon}, q\right)$, uses $\log D$ random bits and is correct with probability at least $1-\varepsilon$.

Remark: Saha [Sah08] observed that there is a deterministic algorithm to find an irreducible polynomial $g(x)$ over $\mathbb{F}_q$ of degree roughly d in poly$(d, \log q)$ time. Therefore, by going to a suitable field extension, we may even use a sample space of coprime polynomials of the form $x^t + \alpha$ and choose $t = \frac{1}{\varepsilon}$ to bound the error probability by ε. This also uses only $\log D$ random bits and the achieves a slightly better time complexity.

2.4 Klivans-Spielman: Random univariate substitution

All the previous randomized discussed uses $\Omega(n)$ random bits. It is easy to see that identity testing for n-variate polynomials of degree bounded by total d need $\Omega(d \log n)$ random bits. For polynomials with m monomials, one can prove a lower bound of $\Omega(\log m)$.

Klivans and Spielman [KS01] present a randomized identity test that uses $O(\log(mnd))$ random bits which works better than the earlier algorithms if m was subexponential.

The idea is to reduce the given multivariate polynomial $f(x_1, \cdots, x_n)$ to a univariate polynomial whose degree is not too large. This reduction will be randomized and the resulting univariate polynomial would be non-zero with probability $1 - \varepsilon$ if the polynomial was non-zero to begin with.

One possible approach is to just substitute $x_i = y^{r_i}$ for each i where r_i's are randomly chosen in a suitable range. This indeed works due to the following lemma. The original version of the *Isolation lemma* was by Mulmuley, Vazirani and Vazirani which isolates a single element of a family of sets. Their lemma was extended by Chari, Rohatgi and Srinivasan [CRS95] where they show that the number of random bits could be reduced if the size of the family was small. A multi-set version has been proposed by many people through a proof that closely follows the original proof of Mulmuley, Mulmuley and Vazirani. Here we present the version in the paper by Klivans and Spielman.

***Lemma* 2.8** (Isolation Lemma)**.** *Let $\mathcal{F}$ be a family of distinct linear forms $\{\sum_{i=1}^n c_i x_i\}$ where each c_i is an integer less than C. If each x_i is randomly set to a value in $\{1, \cdots, Cn/\varepsilon\}$, then with probability at least $1 - \varepsilon$ there is a unique linear form of minimum value.*

Proof. For any fixed assignment, an index i is said to be *singular* if there are two linear forms in $\mathcal{F}$ with different coefficients of x_i that attain minimum value under this assignment. It is clear that if there are no singular indices, then there is a unique linear form of minimum value.

Consider an assignment for all variables other than x_i. This makes the value of each linear form a linear function of x_i. Partition $\mathcal{F}$ into sets $S_0, \cdots, S_C$ depending on what the coefficient of x_i is. In each set, the minimum value would be attained by polynomials with the smallest constant term. For each class S_t, choose one such representative and let its weight be $a_t x_i + b_t$.

Now randomly assign x_i to a value in $\{1, \cdots, Cn/\varepsilon\}$.The index i would be critical if and only if there are two representatives of different classes that give the same value after assignment of x_i. This means that x_i must be set to one of the critical points of the following piece-wise linear function:

$$w(x_i) = \min_{0 \le t \le C} \{a_t x_i + b_t\}$$

Clearly, there can be at most C many such critical points. Therefore, there are at most C values of x_i that makes an index singular and hence the probability that an assignment makes i singular is at most $\frac{C}{Cn/\varepsilon} = \frac{\varepsilon}{n}$. A union bound over indices completes the proof. □

The isolation lemma gives a simple randomized algorithm for identity testing of polynomials whose degree in each variable is bounded by d: make the substitution $x_i = y^{r_i}$ for $r_i \in \{1, \cdots, dn/\varepsilon\}$. Each monomial $x_1^{d_1} \cdots x_n^{d_n}$ is now mapped to $y^{\sum d_i r_i}$. Since r_i's are chosen at random, there is a unique monomial which has least degree and hence is never cancelled.

However, the number of random bits required is $n \log\left(\frac{dn}{\varepsilon}\right)$. Klivans and Spielman use a different reduction to univariate polynomials which uses $O\left(\log\left(\frac{mnd}{\varepsilon}\right)\right)$ random bits where m is the number of monomials. This technique closely parallels ideas in the work by Chari, Rohatgi and Srinivasan. Just as Chari et al wish to isolate a single element from a family of sets, Klivans and Spielman try to isolate a single monomial. The ideas are very similar.

2.4.1 Reduction to univariate polynomials

Let t be a parameter that shall be fixed later. Pick a prime p larger than t and d. The reduction picks a k at random from $\{0, \cdots, p-1\}$ and makes the following substitution:

$$x_i = y^{a_i} \quad \text{where} \quad a_i = k^i \bmod p$$

***Lemma* 2.9.** *Let $f(x_1, \cdots, x_n)$ be a non-zero polynomial whose degree is bounded by d. Then, each monomial of f is mapped to different monomials under the above substitution, with probability at least $\left(1 - \frac{m^2 n}{t}\right)$.*

Proof. Suppose not. Let $x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ and $x_1^{\beta_1} \cdots x_n^{\beta_n}$ be two fixed monomials. If these two are mapped to the same value, then $\sum \alpha_i k^i = \sum \beta_i k^i \bmod p$. And two different polynomials of degree n, which they are as $p > d$, can agree at at most n points. Therefore the probability of choosing such a k is hence at most $\frac{n}{t}$. A union bound over all pairs of monomials completes the proof. □

If we want the error to be less than ε, then choose $t \geq \frac{m^2 n}{\varepsilon}$. This would make the final degree of the polynomial bounded by $\frac{m^2 nd}{\varepsilon}$ on which we can use a Schwarz-Zippel test by going to a large enough extension. Klivans and Spielman deal with this large degree (since it depends on m) by using the Isolation Lemma. We now sketch the idea.

2.4.2 Degree reduction (a sketch)

The previous algorithm described how a multivariate polynomial can be converted to a univariate polynomial while still keeping each monomial separated. Now we look at a small modification of that construction that uses the Isolation lemma to isolate a single non-zero monomial, if present.

The earlier algorithm made the substitution $x_i = y^{a_i}$ for some suitable choice of a_i. Let us assume that each a_i is a q bit number and let $\ell = O(\log(dn))$. Represent each a_i in base 2^ℓ as

$$a_i = b_{i0} + b_{i1}2^\ell + \cdots + b_{i(\frac{q}{\ell}-1)}2^{(\frac{q}{\ell}-1)\ell}$$

The modified substitution is the following:

1. Pick k at random from $\{0, \cdots, p-1\}$ and let $a_i = k^i \bmod p$.
2. Represent a_i in base 2^ℓ as above.
3. Pick $r_0, \cdots, r_{\frac{q}{\ell}-1}$ values independently and uniformly at random from a small range $\{1, \cdots, R\}$.
4. Make the substitution $x_i = y^{c_i}$ where $c_i = b_{i0}r_0 + b_{i1}r_1 + \cdots + b_{i(\frac{q}{\ell}-1)}r_{\frac{q}{\ell}-1}$.

After this substitution, each monomial in the polynomial is mapped to a power of y where the power is a linear function over r_i's.

***Claim* 2.10.** *Assume that the choice of k is a positive candidate in Lemma 2.9. Then, under the modified substitution, different monomials are mapped to exponents that are different linear functions of the r_i's.*

Proof. A particular assignment of the r_i's, namely $r_i = 2^{i\ell}$ maps is precisely the mapping in Lemma 2.9 and by assumption produces different values. Hence, the linear functions have to be distinct. □

Therefore, each exponent of y in the resulting polynomial is a distinct linear function of the r_i's. It is a simple calculation to check that the coefficients involved are $\mathsf{poly}(n, d)$ and we can choose our range $\{1, \cdots, R\}$ appropriately to make sure that the Isolation Lemma guarantees a unique minimum value linear form. This means that the exponent of least degree will be contributed by a unique monomial and hence the resulting polynomial is non-zero. The degree of the resulting polynomial is $\mathsf{poly}\left(n, d, \frac{1}{\varepsilon}\right)$ and the entire reduction uses only $O\left(\log\left(\frac{mnd}{\varepsilon}\right)\right)$ random bits.

We now proceed to study some deterministic algorithms for PIT.

Deterministic Algorithms for PIT

3

In this chapter we look at deterministic algorithms for PIT for certain restricted circuits. Progress has been only for restricted version of depth 3 circuits.

Easy cases

Depth 2 circuits can only compute "sparse" polynomials, i.e. polynomials with few (polynomially many) monomials in them.

In fact PIT of any circuit, not just depth 2, that computes a sparse polynomial can be solved efficiently. The following observation can be directly translated to a polynomial time algorithm.

***Observation* 3.1.** *Let $p(x_1, \cdots, x_n)$ be a non-zero polynomial whose degree in each variable is less than d and the number of monomials is m. Then there exists an $r \leq (mn \log d)^2$ such that*

$$p\left(1, y, y^d, \cdots, y^{d^{n-1}}\right) \neq 0 \bmod y^r - 1$$

Proof. We know that $q(y) = p\left(1, y, y^d, \cdots, y^{d^{n-1}}\right)$ is a non-zero polynomial and let y^a be a non-zero monomial in p. If $p = 0 \bmod y^r - 1$, then there has to be another monomials $y^b = y^a \bmod y^r - 1$ and this is possible if and only if $r \mid b - a$. This would not happen if r was chosen such that

$$r \nmid \prod_{y^b \in q(y)} (b - a) \leq d^{nm} = R.$$

Since R has at most $\log R = mn \log d$ prime factors, and since we'd encounter at least $\log R + 1$ in the range $1 \leq r \leq (mn \log d)^2$ it is clear that at least one such r we'd get $q(y) \neq 0 \bmod y^r - 1$. $\square$

Several black-box tests have also been devised for depth 2 circuits but considerable progress has been made for restricted depth 3 circuits as well.

The case when root is a multiplication gate is easy to solve. This is because the polynomial computed by a $\Pi\Sigma\Pi$ circuit is zero if and only if one of the addition gates compute the zero polynomial. Therefore, it reduces to depth 2 circuits. Thus, the non-trivial case is $\Sigma\Pi\Sigma$ circuits. PIT for general $\Sigma\Pi\Sigma$ circuits is still open but polynomial time algorithms are known for restricted versions.

3.1 The Kayal-Saxena test

Let C be a $\Sigma\Pi\Sigma$ circuit over n variables and degree d such that the top addition gate has k children. For sake of brevity, we shall refer to such circuits as $\Sigma\Pi\Sigma(n, k, d)$ circuits. Kayal and Saxena [KS07] presented a $\mathsf{poly}(n, d^k, |\mathbb{F}|)$ algorithm for PIT. Hence, for the case when the top fan-in is bounded, this algorithm runs in polynomial time.

3.1.1 The idea

Let C be the given circuit, with top fan-in k, that computes a polynomial f. Therefore, $f = T_1 + \cdots + T_k$ where each $T_i = \prod_{j=1}^{d} L_{ij}$ is a product of linear forms. Fix an ordering of the variables to induce a total order $\preceq$ on the monomials. For any polynomial g, let $\mathrm{LM}(g)$ denote the leading monomial of g. We can assume without loss of generality that

$$\mathrm{LM}(T_1) \succeq \mathrm{LM}(T_2) \succeq \cdots \succeq \mathrm{LM}(T_k)$$

If f is zero then the coefficient of the $\mathrm{LM}(f)$ must be zero, and this can be checked easily. Further, if we are able to show that $f \equiv 0 \bmod T_1$, then $f = 0$. And this would be done by induction on k.

Let us assume that T_1 consists of distinct linear forms. Therefore by Chinese Remainder Theorem, $f \equiv 0 \bmod T_1$ if and only if $f \equiv 0 \bmod L_{1i}$ for each i. To check if $f \equiv 0 \bmod L$ for some linear form L, we replace L by variable x_1 and transform the rest to make it an invertible transformation. Thus the equation reduces to the form $f \bmod x_1 \in \frac{\mathbb{F}[x_1,\cdots,x_n]}{x_1} = \mathbb{F}[x_2, \cdots, x_n]$. The polynomial $f \bmod x_1$ is now a $\Sigma\Pi\Sigma$ circuit with top fan-in $k-1$ (as $T_1 \equiv 0 \bmod x_1$), and using induction, can be checked if it is zero. Repeating this for every L_{1i}, we can check if f is identically zero or not.

This method fails if T_1 happens to have repeated factors. For e.g., if $T_1 = x_1^5 x_2^3$, we should instead be checking if $f \bmod x_1^5$ and $f \bmod x_2^3$ are zero or not. Here $f \bmod x_1^5 \in \frac{\mathbb{F}[x_1,\cdots,x_n]}{x_1^5} =$

$\left(\frac{\mathbb{F}[x_1]}{x_1^5}\right)[x_2, \cdots, x_n]$ is a polynomial over a *local ring*. Thus, in the recursive calls, we would be over a local ring rather than over the field. Therefore, we need to make sure that Chinese Remaindering works over local rings; Kayal and Saxena showed that it indeed does.

3.1.2 Local Rings

We shall now look at a few properties that local rings inherit from fields.

***Definition* 3.2.** *(Local ring) A ring $\mathcal{R}$ is said to be* local *if it contains a unique maximal ideal. This unique maximal ideal $\mathcal{M}$ is the ideal of all nilpotent elements.*

***Fact* 3.3.** *Every element $a \in \mathcal{R}$ can be written uniquely as $\alpha + m$ where $\alpha \in \mathbb{F}$ and $m \in \mathcal{M}$.*

This therefore induces a natural homomorphism g$\varphi : \mathcal{R} \longrightarrow \mathbb{F}$ such that $\varphi(a) = \alpha$, which can also be lifted as $\varphi : \mathcal{R}[x_1, \cdots, x_n] \longrightarrow \mathbb{F}[x_1, \cdots, x_n]$. The following lemma essentially states that Chinese Remaindering works over local rings as well.

***Lemma* 3.4.** *[KS07] Let $\mathcal{R}$ be a local ring over $\mathbb{F}$ and let $p, f, g \in \mathcal{R}[z_1, \cdots, z_n]$ be multivariate polynomials such that $\varphi(f)$ and $\varphi(g)$ are coprime.*

$$\begin{aligned} \textit{If } p &\equiv 0 \bmod f \\ \textit{and } p &\equiv 0 \bmod g \\ \textit{then } p &\equiv 0 \bmod fg \end{aligned}$$

Proof. Let the total degree of f and g be d_f and d_g. Without loss of generality, using a suitable invertible transformation on the variables, we can assume that the coefficients of $x_n^{d_f}$ and $x_n^{d_g}$ in f and g respectively are units.

Thinking of f and g as univariate polynomials $\mathcal{R}(x_1, \cdots, x_{n-1})[x_n]$ (the ring of fractions), we have $\varphi(f)$ and $\varphi(g)$ coprime over $\mathbb{F}(x_1, \cdots, x_{n-1})$ as well. Hence, there exists $a, b \in \mathbb{F}(x_1, \cdots, x_{n-1})$ such that $a\varphi(f) + b\varphi(g) = 1$. This can be re-written as

$$af + bg = 1 \bmod \mathcal{R}(x_1, \cdots, x_{n-1})[x_n]/\mathcal{M}$$

where $\mathcal{M} = \{r : r \in \mathcal{R}(x_1, \cdots, x_{n-1})[x_n] \text{ is nilpotent}\}$. Since $\mathcal{M}$ is nilpotent, let t be such that $\mathcal{M}^t = 0$. Using repeated applications of Hensel lifting, the above equation becomes

$$\begin{aligned} a^* f^* + b^* g^* &= 1 \bmod \mathcal{R}(x_1, \cdots, x_{n-1})[x_n]/\mathcal{M}^t \\ &= 1 \bmod \mathcal{R}(x_1, \cdots, x_{n-1})[x_n] \end{aligned}$$

And there exists a $m \in \mathcal{M}$ such that $a^* f + b^* g = 1 + m$, which is invertible. Setting $a' = a^*(1+m)^{-1}$ and $b' = b^*(1+m)^{-1}$, we get

$$a' f + b' g = 1 \text{ in } \mathcal{R}(x_1, \cdots, x_{n-1})[x_n]$$

$$\begin{aligned}
p &= 0 \bmod f \\
\Longrightarrow p &= fq \text{ for } q \in \mathcal{R}(x_1, \cdots, x_n)[x_n] \\
\Longrightarrow fq &= 0 \bmod g \\
\Longrightarrow q &= 0 \bmod g \\
\Longrightarrow q &= gh \text{ for } h \in \mathcal{R}(x_1, \cdots, x_n)[x_n] \\
\Longrightarrow p &= fgh \text{ in } \mathcal{R}(x_1, \cdots, x_{n-1})[x_n]
\end{aligned}$$

Since p, f, g are polynomials in $\mathcal{R}[x_1, \cdots, x_n]$ and f, g monic, by Gauss's lemma we have $p = fgh$ in $\mathcal{R}[x_1, \cdots, x_n]$ itself. □

We are now set to look at the identity test.

3.1.3 The identity test

Let C be a $\Sigma\Pi\Sigma$ arithmetic circuit, with top fan-in k and degree d computing a polynomial f. The algorithm is recursive where each recursive call decreases k but increases the dimension of the base ring (which is $\mathbb{F}$ to begin with).

Input

The algorithm takes three inputs:

- A local ring $\mathcal{R}$ over a field $\mathbb{F}$ with the maximal ideal $\mathcal{M}$ presented in its basis form. The initial setting is $\mathcal{R} = \mathbb{F}$ and $\mathcal{M} = \langle 0 \rangle$.
- A set of k coefficients $\langle \beta_1, \cdots, \beta_k \rangle$, with $\beta_i \in \mathcal{R}$ for all i.
- A set of k terms $\langle T_1, \cdots, T_k \rangle$. Each T_i is a product of d linear functions in n variables over $\mathcal{R}$. That is, $T_i = \prod_{j=1}^{d} L_{ij}$.

Output

Let $p(x_1, \cdots, x_n) = \beta_1 T_1 + \cdots, \beta_k T_k$. The output, $\mathrm{ID}(\mathcal{R}, \langle \beta_1, \cdots, \beta_k \rangle, \langle T_1, \cdots, T_k \rangle)$ is YES if and only if $p(x_1, \cdots, x_n) = 0$ in $\mathcal{R}$.

Algorithm

Step 1**:** (Rearranging T_i's) If necessary, rearrange the T_i's to make sure that

$$\mathrm{LM}(T_1) \geq \cdots \geq \mathrm{LM}(T_k)$$

Also make sure that coefficient of $\mathrm{LM}(T_1)$ is a unit.

Step 2**:** (Single multiplication gate) If $k = 1$, we need to test if $\beta_1 T_1 = 0$ in $\mathcal{R}$. Since the leading coefficient of T_1 is a unit, it suffices to check if $\beta_1 = 0$.

Step 3**:** (Checking if $p = 0 \bmod T_1$) Write T_1 as a product of coprime factors, where each factor is of the form

$$S = (l + m_1)(l + m_2) \cdots (l + m_t)$$

where $l \in \mathbb{F}[x_1, \cdots, x_n]$ and $m_i \in \mathcal{M}$ for all i.

For each such factor S, do the following:

Step 3.1**:** (Change of variables) Apply an invertible linear transformation σ on the variables to make convert l to x_1. Thus, S divides p if and only if $\sigma(S)$ divides $\sigma(p)$.

Step 3.2**:** (Recursive calls) The new ring $\mathcal{R}' = R[x_1]/(\sigma(S))$ which is a local ring as well. For $2 \leq i \leq k$, the transformation σ might convert some of the factors of T_i to an element of $\mathcal{R}'$. Collect all such ring elements of $\sigma(T_i)$ as $\gamma_i \in \mathcal{R}'$ and write $\sigma(T_i) = \gamma_i T_i'$.

Recursively call $\mathrm{ID}(\mathcal{R}', \langle \beta_2 \gamma_2, \cdots, \beta_k \gamma_k \rangle, \langle T_2', \cdots, T_k' \rangle)$. If the call returns NO, exit and output NO.

Step 4**:** Output YES.

***Theorem* 3.5.** *The algorithm runs in time* $\mathsf{poly}(\dim \mathcal{R}, n, d^k)$ *outputs YES if and only if* $p(x_1, \cdots, x_n) = 0$ *in* $\mathcal{R}$.

Proof. The proof is an induction on k. The base case when $k = 1$ is handled in step 2. For $k \geq 2$, let $T_1 = S_1 S_2 \cdots S_m$. By induction we verify in step 3 if $p(x_1, \cdots, x_n) = 0 \bmod S_i$ for each

S_i. Therefore, by Lemma 3.4 we have $p(x_1, \cdots, x_n) = 0 \bmod T_1$. Since $p = \sum T_i$, the leading monomial in T_1 and p are the same. But since we also checked that the coefficient of the leading monomial is zero, p has to be zero.

For the running time analysis, let r be the dimension of the ring R. In every recursive call, k decreases by 1 and the dimension of the ring $\mathcal{R}$ grows by a factor of at most R'. If $T(r,k)$ is the time taken for $\mathrm{ID}(\mathcal{R}, \langle \beta_1, \cdots, \beta_k \rangle, \langle T_1, \cdots, T_k \rangle)$, we have the recurrence

$$T(r,k) \leq d \cdot T(dr, k-1) + \mathsf{poly}(n, d^k, r)$$

which is $\mathsf{poly}(n, d^k, r)$. □

This completes the Kayal-Saxena identity test for $\Sigma\Pi\Sigma$ circuits with bounded top fan-in.

3.2 Rank bounds and $\Sigma\Pi\Sigma(n,k,d)$ circuits

The *rank approach* asks the following question: if C is a $\Sigma\Pi\Sigma$ circuit that indeed computes the zero polynomial, then how many variables does it *really* depend on? To give a reasonable answer, we need to assume that the given circuit is not "redundant" in some ways.

***Definition* 3.6** (Minimal and simple circuits)**.** *A $\Sigma\Pi\Sigma$ circuit $C = P_1 + \cdots + P_k$ is said to be* minimal *if no proper subset of $\{P_i\}_{1 \leq i \leq k}$ sums to zero.*

The circuit is said to be simple *there is no non-trivial common factor between all the P_i's.*

***Definition* 3.7** (Rank of a circuit)**.** *For a given circuit $\Sigma\Pi\Sigma$ circuit, the* rank *of the circuit is the maximum number of independent linear functions that appear as a factor of any product gate.*

Suppose we can get an upper-bound R on the rank of any minimal and simple $\Sigma\Pi\Sigma(n,k,d)$ circuit computing the zero polynomial. Then we have a partial approach towards identity testing.

1. If k is a constant, it can be checked recursively if C is simple and minimal.

2. Compute the rank r of the circuit C.

3. If the $r < R$ is small, then the circuit is essentially a circuit on just R variables. We can check in d^R time if C is zero or not.

4. If the rank is larger than the upper-bound then the circuit computes a non-zero polynomial.

This was infact the idea in Dvir and Shpilka's $n^{O(\log n)}$ algorithm [DS05] for $\Sigma\Pi\Sigma$ circuits of bounded top fan-in (before the algorithm by Kayal and Saxena [KS07]). It was conjectured by Dvir and Shpilka that $R(k,d)$ is a polynomial function of k alone. However, Kayal and Saxena [KS07] provided a counter-example over finite fields. Karnin and Shpilka showed how rank bounds can be turned into black-box identity tests.

Theorem 3.8. *[KS08] Fix a field* $\mathbb{F}$. *Let* $R(k,d)$ *be an integer such that every minimal, simple* $\Sigma\Pi\Sigma(n,k,d)$ *circuit computing the zero polynomial has rank at most* $R(k,d)$. *Then, there is a black-box algorithm to test if a given* $\Sigma\Pi\Sigma(n,k,d)$ *circuit is zero or not, in deterministic time* $\mathsf{poly}(d^{R(k,d)}, n)$.

We'll see the proof of this theorem soon but we first look at some consequences of the above theorem with some recent developments in rank bounds. Saxena and Seshadri recently showed rank upper bounds that are almost tight.

Theorem 3.9. *[SS09] Let* C *be a minimal, simple* $\Sigma\Pi\Sigma(n,k,d)$ *circuit that is identically zero. Then, rank*$(C) = O(k^3 \log d)$. *And there exist identities with rank* $\Omega(k \log d)$. □

Using the bound by Saxena and Seshadri, this gave a $n^{O(\log n)}$ black-box test for depth 3 circuits with bounded top fan-in.

Though the conjecture of Dvir and Shpilka was disproved over finite fields, the question remained if $R(k,d)$ is a function of k alone over $\mathbb{Q}$ or $\mathbb{R}$. This was answered in the affirmative by Kayal and Saraf [KS09] very recently.

Theorem 3.10. *[KS09] Every minimal, simple* $\Sigma\Pi\Sigma(n,k,d)$ *circuit with coefficients from* $\mathcal{R}$ *that computes the zero polynomial has rank bounded by* $3^k((k+1)!) = 2^{O(k\log k)}$. □

This, using with Theorem 3.8, gives a black-box algorithm for $\Sigma\Pi\Sigma$ circuits with bounded top fan-in.

Theorem 3.11. *[KS09] There is a deterministic black-box algorithm for* $\Sigma\Pi\Sigma(n,k,d)$ *circuits over* $\mathbb{Q}$, *running in time* $\mathsf{poly}(d^{2^{O(k\log k)}}, n)$. □

We now see how rank bounds can be converted to black-box PITs

3.2.1 Rank bounds to black-box PITs

The proof of Theorem 3.8 crucially uses the following Lemma by Gabizon and Raz [GR05].

***Lemma* 3.12.** *[GR05] For integers $n \geq t > 0$ and an element $\alpha \in \mathbb{F}$, define the linear transformation*

$$\varphi_{\alpha,t,n}(x_1, \cdots, x_n) = \begin{bmatrix} 1 & \alpha & \alpha^2 & \cdots & \alpha^{n-1} \\ 1 & \alpha^2 & \alpha^4 & \cdots & \alpha^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^t & \alpha^{2t} & \cdots & \alpha^{t(n-1)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Fix any number of subspaces $W_1, \cdots, W_s \subseteq \mathbb{F}^n$ of dimension at most t. Then, there are at most $s \cdot (n-1) \cdot \binom{t+1}{2}$ elements $\alpha \in \mathbb{F}$ such for some W_i with $\dim(\varphi_{\alpha,t,n}(W_i)) < \dim(W_i)$. □

Using this lemma, we shall come up with a small set of linear transformations such that for each non-zero $\Sigma\Pi\Sigma(n,k,d)$ circuit, at least one of the linear transformations continues to keep it non-zero. We'll assume that all linear functions that appear in the circuit are linear forms i.e. they don't have a constant term. This is because we can assume that a linear function $a_0 + a_1x_1 + \cdots a_nx_n$ is actually $a_0x_0 + \cdots a_nx_n$. We shall assume that $R(k,d)$ is a rank bound for minimal, simple $\Sigma\Pi\Sigma(n,k,d)$ circuits that evaluate to zero.

***Theorem* 3.13.** *[KS08] Let C be a $\Sigma\Pi\Sigma(n,k,d)$ circuit that is non-zero. Let $S \subseteq \mathbb{F}$ such that*

$$|S| \geq \left(\binom{dk}{2} + 2^k\right) \cdot n \cdot \binom{R(k,d)+2}{2} + 1$$

Then there is some $\alpha \in S$ such that $V_\alpha(C) = \varphi_{\alpha,n,R(k,d)+1}(C)$ is non-zero.

Proof. Let $C = T_1 + \cdots + T_k$. We shall define $\mathrm{sim}(C)$ as follows:

$$\mathrm{sim}(C) = \frac{C}{\gcd(T_1, \cdots, T_k)}$$

The proof proceeds by picking up a few subspaces of linear functions of small dimension such that, if all their dimensions were preserved, then the circuit has to be non-zero after the transformation. The subspaces are as follows:

1. For every pair of linear forms ℓ_i, ℓ_j that appear in the circuit, let $W_{\ell_i,\ell_j} = (\ell_i, \ell_j)$.

2. For every subset $\emptyset \neq A \subseteq [k]$, define $C_A = \sum_{i \in A} T_i$ and let

$$r_A = \min\{R(k,d)+1, \mathrm{rank}(\mathrm{sim}(C_A))\}$$

   Let W_A be the space spanned by r_A linearly independent vectors that appear in $\mathrm{sim}(C_A)$.

The claim is that, if each of the W_{ℓ_i,ℓ_j}'s and W_A's are preserved, then the circuit will continue to be non-zero. Suppose not.

The first observation is that the transformation cannot map two linear functions that appear in the circuit to the same linear function since it preserves W_{ℓ_i,ℓ_j}'s. Hence, we have $V_\alpha(\mathrm{sim}(C_A)) = \mathrm{sim}(V_\alpha(C_A))$. We shall just argue on the simple part of the circuit so we shall assume that C (and hence $V_\alpha(C)$ as well) is simple.

If $V_\alpha(C) = 0$, then the rank bound says that either $V_\alpha(C)$ is not minimal or we have $\mathrm{rank}(V_\alpha(C)) \leq R(k,d)$. If the rank was small, since we are preserving W_A's, and in particular $W_{[k]}$, the circuit C has to be preserved.

Suppose the $V_\alpha(C)$ is not minimal. Let A be a minimal subset such that $V_\alpha(C_A) = 0$ with $C_A \neq 0$. Hence, $V_\alpha(\mathrm{sim}(C_A))$ is a simple, minimal circuit that evaluates to zero and therefore has rank at most $R(k,d)$. But the rank of W_A is preserved, $C_A \neq 0$ forces $V_\alpha(C_A) \neq 0$ which is a contradiction. $\square$

Once we have obtained such a rank-preserving transformation, the circuit is now essentially over $R(k,d)+1$ variables. An application of a brute-force Schwarz-Zippel would complete the proof of Theorem 3.8.

3.3 Saxena's test for diagonal circuits

In this section we shall look at yet another restricted version of depth 3 circuits.

***Definition* 3.14.** *A $\Sigma\Pi\Sigma$ circuit C is said to be* diagonal *if it is of the form*

$$C(x_1,\cdots,x_n) = \sum_{i=1}^{k} \ell_i^{e_i} \quad \text{where } \ell_i \text{ is a linear function over the variables}$$

The idea is to reduce this problem to a PIT problem of a formula over non-commuting variables. In the setting of formulas over non-commuting variables, Raz and Shpilka [RS04] showed that it can be tested in deterministic polynomial time if it zero.

The reduction to a non-commutative formula is by a conversion to express a multiplication gate $(a_0 + a_1x_1 + \cdots a_nx_n)^d$ to a *dual* form:

$$(a_0 + a_1x_1 + \cdots a_nx_n)^d = \sum_j f_{j1}(x_1)f_{j2}(x_2)\cdots f_{jn}(x_n)$$

The advantage of using the expression on the RHS is that the variables can be assumed to be non-commuting. Therefore if the above conversion can achieved in polynomial time, then we have a polynomial algorithm for identity testing of diagonal circuits by just making this transformation and using the algorithm by Raz and Shpilka. Saxena provides a simple way to convert a multiplication gate to its dual. We present the case when $\mathbb{F}$ is a field of characteristic zero though it may be achieved over any field.

***Lemma* 3.15.** *[Sax08] Let $a_0, \cdots, a_n$ be elements of a field $\mathbb{F}$ of characteristic zero. Then, in* $\mathsf{poly}(n, d)$ *many field operations, we can compute univariate polynomials $f_{i,j}$'s such that*

$$(a_0 + a_1x_1 + \cdots a_nx_n)^d = \sum_{i=1}^{nd+d+1} f_{i1}(x_1)f_{i2}(x_2)\cdots f_{in}(x_n)$$

Proof. Let $E(x) = \exp(x) = 1 + x + \frac{x^2}{2!} + \cdots$ and let $E_d(x) = E(x) \bmod x^d$ be the truncated version.

$$\begin{aligned} \frac{(a_0 + a_1x_1 + \cdots a_nx_n)^d}{d!} &= \text{coefficient of } z^d \text{ in } \exp((a_0 + a_1x_1 + \cdots a_nx_n)z) \\ &= \text{coefficient of } z^d \text{ in } \exp(a_0z)\exp(a_1x_1z)\cdots\exp(a_nx_nz) \\ &= \text{coefficient of } z^d \text{ in } E_d(a_0z)E_d(a_1x_1z)\cdots E_d(a_nx_nz) \end{aligned}$$

Hence, viewing $E_d(a_0z)E_d(a_1x_1z)\cdots E_d(a_nx_nz)$ as a univariate polynomial in z of degree $d' = nd + d$, we just need to find the coefficient of the z^d. This can be done by evaluating the polynomial at distinct points and interpolating. Hence, for distinct points $\alpha_1, \cdots, \alpha_{d'} \in \mathbb{F}$, we can compute $\beta_1, \cdots, \beta_{d'}$ in polynomial time such that

$$(a_0 + a_1x_1 + \cdots a_nx_n)^d = \sum_{i=1}^{d'} \beta_i E_d(a_0\alpha_i)E_d(a_1x_1\alpha_i)\cdots E_d(a_nx_n\alpha_i)$$

which is precisely what we wanted. $\square$

Chasm at Depth 4

4

In this chapter we look at a result by Agrawal and Vinay [AV08] on depth reduction. Informally, the result states that exponential sized circuits do not gain anything if the depth is beyond 4. Formally, the main result can be stated as follows:

***Theorem* 4.1** ([AV08])**.** *If a polynomial $P(x_1, \cdots, x_n)$ of degree $d = O(n)$ can be computed by an arithmetic circuit of size $2^{o(d + d\log\frac{n}{d})}$, it can also be computed by a depth 4 circuit of size $2^{o(d+d\log\frac{n}{d})}$.*

It is a simple observation that any polynomial $p(x_1, \cdots, x_n)$ has at most $\binom{n+d}{d}$ monomials and hence can be trivially computed by a $\Sigma\Pi$ circuit of size $\binom{n+d}{d} = 2^{O(d+d\log\frac{n}{d})}$. Hence, the above theorem implies that if we have subexponential lower bounds for depth 4 circuits, we have subexponential lower bounds for any depth!

***Corollary* 4.2.** *Let $p(x_1, \cdots, x_n)$ be a multivariate polynomial. Suppose there are no $2^{o(n)}$ sized depth 4 arithmetic circuits that can compute p. Then there is no $2^{o(n)}$ sized arithmetic circuit (of arbitrary depth) that can compute p.*

The depth reduction is proceeded in two stages. The first stage reduces the depth to $O(\log d)$ by the construction of Allender, Jiao, Mahajan and Vinay [AJMV98]. Using a careful analysis of this reduction, the circuit is further reduced to a depth 4 circuit.

4.1 Reduction to depth $O(\log d)$

Given as input is a circuit C computing a polynomial $p(x_1, \cdots, x_n)$ of degree $d = O(n)$. Without loss of generality, we can assume that the circuit is layered with alternative layers of addition and multiplication gates. Further, we shall assume that each multiplication gate has exactly two children.

4.1.1 Computing degrees

Though the polynomial computed by the circuit is of degree less than d, it could be possible that the intermediate gates compute larger degree polynomials which are somehow cancelled later. However, we can make sure that each gate computes a polynomial of degree at most d. Further, we can label each gate by the formal degree of the polynomial computed there.

Each gate g_i of the circuit is now replaced by $d+1$ gates $g_{i0}, g_{i1}, \cdots, g_{id}$. The gate g_{is} would compute the degree s homogeneous part of the polynomial computed at g_i.

If g_0 was an addition gate with $g_0 = h_1 + h_2 + \cdots + h_k$, then we set $g_{0i} = h_{0i} + \cdots + h_{ki}$ for each i. If g_0 was a multiplication gate with two children h_1 and h_2, we set $g_{0i} = \sum_{j=0}^{i} h_{1j} h_{2(i-j)}$.

Thus, every gate is naturally labelled by its degree. As a convention, we shall assume that the degree of the left child of any multiplication gate is smaller than or equal to the degree of the right child.

4.1.2 Evaluation through proof trees

A *proof tree* rooted at a gate g is a sub-circuit of C that is obtained as follows:

- start with the sub-circuit in C that has gate g at the top and computes the polynomial associated with gate g,
- for every addition gate in this sub-circuit, retain only one of the inputs to this gate and delete the other input lines,
- for any multiplication gate, retain both the inputs.

A simple observation is that a single proof tree computes one monomial of the formal expression computed at g. And the polynomial computed at g is just the sum of the polynomial computed at every proof tree rooted at g. We shall denote the polynomial computed by a proof tree T as $p(T)$.

For every gate g, define $[g]$ to be the polynomial computed at gate g. Also, for every pair of gates g and h, define $[g, h] = \sum_T p(T, h)$ where T runs over all proof trees rooted at g with h occurring on its rightmost path and $p(T, h)$ is the polynomial computed by the proof tree T when the last occurance of h is replaced by the constant 1. If h does not occur on the right most path, then $[g, h]$ is zero. The gates of the new circuits are $[g]$, $[g, h]$ and $[x_i]$ for gates $g, h \in C$ and variables x_i. We shall now describe the connections between the gates.

Firstly, $[g] = \sum_i [g, x_i][x_i]$. Also, if g is an addition gate with children $g_1, \cdots, g_k$, then $[g, h] = \sum_i [g_i, h]$. If g is a multiplication gate, it is a little tricker. If the rightmost path from g to h consists of just addition gates, then $[g, h] = [g_L]$, the left child of g. Otherwise, for any fixed rightmost path, there must be at least a unique intermediate multiplication gate p on this path such that

$$\deg(p_R) \leq \frac{1}{2}(\deg g + \deg h) \leq \deg p$$

Since there could be rightmost paths between g and h, we just run over all gates p that satisfy the above equation. Then, $[g, h] = \sum_p [g, p][p_L][p_R, h]$. We want to ensure that the degree of each child of $[g, h]$ is at most $(\deg(g) - \deg(h))/2$.

- $\deg([g, p]) = \deg(g) - \deg(p) \leq \frac{1}{2}(\deg g - \deg h)$
- $\deg([p_R, h]) = \deg(p_R) - \deg(h) \leq \frac{1}{2}(\deg(g) - \deg(h))$
- $\deg(p_L) \leq \deg(p) \leq \frac{1}{2}\deg(g)$

  Also, $\deg(p_L) \leq \deg(p_L) + \deg(p_R) - \deg(h) \leq \deg(g) - \deg(h)$

Thus the problem is with p_L as the degree hasn't dropped by a factor of 2. However, we know that $\deg(p_L) \leq \deg(g)/2$. By unfolding this gate further to reduce the degree. Note that $[p_L] = \sum_i [p_L, x_i][x_i]$ and p_L is an addition gate. Let $p_L = \sum_j p_{L,j}$ where each $p_{L,j}$ is a multiplication gate. Therefore $[p_L, x_i] = \sum_j [p_{L,j}, x_i]$. Repeating the same analysis for this gate, we have $[p_L, x_i] = \sum_q [p_L, q][q_L][q_R, x_i]$ for states q satisfying the degree constraint. Now, $\deg(q_L) \leq \frac{1}{2}\deg(p_L) \leq \frac{1}{2}\deg([g, h])$ as required. We hence have

$$[g, h] = \sum_p \sum_i \sum_j \sum_q [g, p][p_{L,j}, q][q_L][q_R, x_i][p_R, h]$$

where p and q satisfy the appropriate degree constraints. This completes the description of the new circuit.

It is clear that the depth of the circuit is $O(\log d)$ and the fan-in of multiplication gates is 6. The size of the new circuit is polynomial bounded by size of C.

4.2 Reduction to depth 4

We now construct an equivalent depth 4 circuit from the reduced circuit. Choose an $\ell \le \frac{d + d\log\frac{n}{d}}{\log S}$ where S is the size of the circuit. And let $t = \frac{1}{2}\log_6 \ell$. Cut the circuit into two two parts: the top has exactly t layers of multiplication gates and the rest of the layers belonging to the bottom. Let $g_1, \cdots, g_k$ (where $k \le S$) be the output gates at the bottom layer. Thus, we can think of the top half as computing a polynomial P_{top} in new variables $y_1, \cdots, y_k$ and each of the g_i computing a polynomial P_i over the input variables. The polynomial computed by the circuit equals

$$P_{\text{top}}(P_1(x_1, \cdots, x_n), P_2(x_1, \cdots, x_n), \cdots, P_k(x_1, \cdots, x_n))$$

Since the top half consists of t levels of multiplication gates, and each multiplication gate has at most 6 children, $\deg(P_{\text{top}})$ is bounded by 6^t. And since the degree drops by a factor of two across multiplication gates, we also have $\deg(P_i) \le \frac{d}{2^t}$. Expressing each of these as a sum of product, we have a depth 4 circuit computing the same polynomial. The size of this circuit is

$$\binom{S + 6^t}{6^t} + S\binom{n + \frac{d}{2^t}}{\frac{d}{2^t}}$$

It is now just a calculation to see that the choice of t makes this $2^{o(d + d\log\frac{n}{d})}$. And that completes the proof of Theorem 4.1.

4.3 Identity testing for depth 4 circuits

In this section we briefly describe how the depth reduction implies that PIT for depth 4 circuits is almost the general case.

***Proposition* 4.3.** *If there is a PIT algorithm for depth 4 circuit running in deterministic polynomial time, then there is a PIT algorithm for any general circuit computing a low degree polynomial running in deterministic $2^{o(n)}$ time.*

Proof. Given any circuit computing a low degree polynomial, we can convert it to a depth 4 circuit of size $2^{o(n)}$. Further, this conversion can be done in time $2^{o(n)}$ as well. Therefore, a polynomial time PIT algorithm for depth 4 would yield a $2^{o(n)}$ algorithm for general circuits. □

Agrawal and Vinay further showed that if there was a stronger derandomization for identity testing on depth 4 circuits, then we get stronger results for general circuits.

As remarked earlier, a black-box algorithm is one that does not look into the circuit but just evaluations of the circuit at chosen points. This can be equivalently presented as *pseudorandom generators* for arithmetic circuits.

***Definition* 4.4** (Pseudorandom generators for arithmetic circuits). *Let $\mathbb{F}$ be a field and $\mathcal{C}$ be a class of low degree arithmetic circuits over $\mathbb{F}$. A function $f : \mathbb{N} \longrightarrow (\mathbb{F}[y])^*$ is a $s(n)$-pseudorandom generator against $\mathcal{C}$ if*

- $f(n) = (p_1(y), p_2(y), \cdots, p_n(y))$ *where each $p_i(y)$ is a univariate polynomial over $\mathbb{F}$ whose degree is bounded by $s(n)$ and computable in time polynomial in $s(n)$*
- *For any arithmetic circuit $C \in \mathcal{C}$ of size n,*

$$C(x_1, \cdots, x_n) = 0 \text{ if and only if } C(p_1(y), p_2(y), \cdots, p_n(y)) = 0$$

It is clear that given a $s(n)$-pseudorandom generator f against $\mathcal{C}$, we can solve the PIT problem for circuits in $\mathcal{C}$ in time $(s(n))^{O(1)}$ by just evaluating the univariate polynomial. A polynomial time derandomization is obtained if $s(n)$ is $n^{O(1)}$ and such generators are called *optimal pseudorandom generators*.

4.3.1 Hardness vs randomness in arithmetic circuits

Just like in the boolean setting, there is a "Hardness vs randomness" tradeoff in arithmetic circuits as well. The following lemma shows that existence of optimal PRGs leads to lower bounds.

***Lemma* 4.5.** *[Agr05] Let $f : \mathbb{N} \longrightarrow (\mathbb{F}(y))^*$ be a $s(n)$-pseudorandom generator against a class $\mathcal{C}$ of arithmetic circuit computing a polynomials of degree at most n. If $n \cdot s(n) \leq 2^n$, then there is a multilinear polynomial computed in $2^{O(n)}$ time that cannot be computed by $\mathcal{C}$.*

Proof. Let $q(x_1, \cdots, x_n)$ be a generic multivariate polynomial, that is:

$$q(x_1, \cdots, x_n) = \sum_{S \subseteq [n]} c_S \prod_{i \in S} x_i$$

We shall choose coefficients to satisfy the following condition

$$\sum_{S\subseteq[n]} c_S \prod_{i\in S} p_i(y) = 0$$

where $f(n) = (p_1(y),\cdots,p_n(y))$. Such a polynomial certainly exists because the coefficients can be obtained by solving a set of linear equations in the c_S's. The number of variables is 2^n and the number of constraints is at most $n\cdot s(n)$ (since each $p_i(y)$ has degree at most $s(n)$). As long as $n\cdot s(n)\le 2^n$ we have an under-determined set of linear equations and there exists a non-trivial solution, which can certainly be computed in time $2^{O(n)}$. □

Thus, in particular, optimal pseudorandom generators would yield lower bounds. It is also known that explicit lower bounds of this kind would yield to black-box algorithms for PIT as well.

Lemma 4.6. *[KI03] Let $\{q_m\}_{m\ge1}$ be a multilinear polynomial over $\mathbb{F}$ computable in exponential time and that cannot be computed by subexponential sized arithmetic circuits. Then identity testing of low degree polynomial can be solved in time $n^{O(\log n)}$.*

Proof. Pick subsets $S_1,\cdots,S_n$ be subsets of $[1,c\log n]$ such that $|S_i| = d\log n$ (for suitable constants c and d) and $|S_i\cap S_j|\le\log n$ for $i\ne j$. Such a family of sets is called a *Nisan-Wigderson Design* [NW94]. For a tuple of variables $(x_1,\cdots,x_n)$, let $(x_1,\cdots,x_n)_S$ denote the tuple that retains only those x_i where $i\in S$. Let $p_i = q_{d\log n}(x_1,\cdots,x_n)_{S_i}$. We claim that the function $f : n\mapsto(p_1,\cdots,p_n)$ is a pseudorandom generator for the class of size n arithmetic circuits computing low degree polynomials.

Suppose not, then there exists a circuit C of size n computing a polynomial of degree at most n such that $C(z_1,\cdots,z_n)\ne0$ but $C(p_1,\cdots,p_n)=0$. By an hybrid argument, there must exist an index j such that $C(p_1,\cdots,p_{j-1},z_j,z_{j+1}\cdots,z_n)\ne0$ but $C(p_1,\cdots,p_{j-1},p_j,z_{j+1},\cdots,z_n) = 0$. Fix values for the variables $z_{j+1},\cdots,z_n$ and also for the x_i's not occurring in p_j to still ensure that the resulting polynomial in $C(p_1,\cdots,p_{j-1},z_j,z_{j+1}\cdots,z_n)\ne0$. For each p_i for $i\le j$, all but $\log n$ of the variables have been fixed and the degree of p_i is bounded by n. Replace each such p_i by $\Sigma\Pi$ circuit of size at most n. After all the fixing and replacement, we have a circuit of size at most n^2 over variables $(x_1,\cdots,x_{c\log n})_{S_j}$ and z_j. This circuit computes a non-zero polynomial but becomes zero when z is substituted by p_j. Hence, z_j-p_j divides the polynomial computed by this circuit. We now use can a multivariate polynomial factorization algorithm to compute this factor, and hence compute p_j. The circuit computing the factor has size n^e for some constant e independent of d and this gives a $n^e + n^2$ sized circuit that computes p_j contradicting the hardness of $q_{d \log n}$ for suitable choice of d.

Therefore, C would continue to be non-zero even after the substitution. After the substitution, we have a polynomial over $d \log n$ variables of degree $O(n \log n)$ and it has at most $n^{O(\log n)}$ terms. Therefore, this gives a $n^{O(\log n)}$ algorithm to check if C is zero. □

***Theorem* 4.7.** *[AV08] If there is a deterministic* black-box *PIT algorithm for depth* 4 *circuit running in polynomial time, then there is a deterministic* $n^{O(logn)}$ *algorithm for PIT on general circuits computing a low degree polynomial.*

Proof. Suppose there does indeed exist an optimal pseudorandom generator against depth 4 circuits. By Lemma 4.5 we know that we have a subexponential lower bound in depth 4 circuits for a family of multilinear polynomials $\{q_m\}$. By Corollary 4.2 we know that this implies a subexponential lower bound for $\{q_m\}$ in arithmetic circuits of any depth. To finish, Lemma 4.6 implies such a family $\{q_m\}$ can be used to give a $n^{O(\log n)}$ algorithm for PIT. □

It would be interesting to see if the above result can be tightened to get a polynomial time PIT algorithm for general circuits.

Depth 2 Circuits Over Algebras 5

... or "Trading Depth for Algebra"

We now come to the main contribution of the thesis – a new approach to identity testing for depth 3 circuits. The contents this chapter is joint work with Saha and Saxena [SSS09].

As mentioned earlier, we study a generalization of arithmetic circuits where the constants come from an algebra instead of a field. The variables still commute with each other and with elements of the algebra. In particular, we are interested in depth 2 circuits over the algebra. These involve expressions like

$$P = \prod_{i=1}^{d} \left(A_{i0} + A_{i1}x_1 + \ldots + A_{in}x_n \right)$$

where $A_{ij} \in \mathcal{R}$, an algebra over $\mathbb{F}$ given in basis form.

Without loss of generality, every finite dimensionally associative algebra can be thought of as a matrix algebra. Therefore, we are interested in circuits over matrix algebras. Thus the above equation reduces to a product of matrices, each of whose entries is a linear function of the variables. For convenience, let us call such a matrix a *linear matrix*.

Ben-Or and Cleve showed that any polynomial computed by a small sized arithmetic formula can be computed by a depth 2 circuits over the algebra of 3×3 matrices. Therefore PIT over depth 2 circuits over 3×3 matrices almost capture PIT over general circuits. A natural question to study the setting over 2×2 matrices. We show that this question is very closely related to PIT of depth 3 circuits.

5.1 Equivalence of PIT with $\Sigma\Pi\Sigma$ circuits

We will now prove Theorem 1.5.

***Theorem* 1.5. (restated)** *Identity testing for depth 2 ($\Pi\Sigma$) circuits over $\mathsf{U}_2(\mathbb{F})$ is polynomial time equivalent to identity testing for depth 3 ($\Sigma\Pi\Sigma$) circuits.*

The usual trick would be to construct a $\Pi\Sigma$ circuit over matrices that computes the same function f as the $\Sigma\Pi\Sigma$ circuit. However, this isn't possible in the setting of upper-triangular 2×2 matrices because they are computationally very weak. We shall see soon that they cannot even compute simple polynomials like $x_1x_2 + x_3x_4 + x_5x_6$.

To circumvent this issue, we shall compute a multiple of f instead of f itself. The following lemma shows how this can be achieved.

***Lemma* 5.1.** *Let $f \in \mathbb{F}[x_1, \ldots, x_n]$ be a polynomial computed by a $\Sigma\Pi\Sigma(n, s, d)$ circuit. Given circuit C, it is possible to construct in polynomial time a depth 2 circuit over $\mathsf{U}_2(\mathbb{F})$ of size $O((d+n)s^2)$ that computes a polynomial $p = L \cdot f$, where L is a product of non-zero linear functions.*

Proof. A depth 2 circuit over $\mathsf{U}_2(\mathbb{F})$ is simply a product sequence of 2×2 upper-triangular linear matrices. We now show that there exists such a sequence of length $O((d+n)s^2)$ such that the product 2×2 matrix has $L \cdot f$ as one of its entries.

Since f is computed by a depth 3 circuit, we can write $f = \sum_{i=1}^{s} P_i$, where each summand $P_i = \prod_j l_{ij}$ is a product of linear functions. Observe that we can compute a single P_i using a product sequence of length d as:

$$\begin{bmatrix} l_{i1} & \\ & 1 \end{bmatrix}\begin{bmatrix} l_{i2} & \\ & 1 \end{bmatrix}\cdots\begin{bmatrix} l_{i(d-1)} & \\ & 1 \end{bmatrix}\begin{bmatrix} 1 & l_{id} \\ & 1 \end{bmatrix} = \begin{bmatrix} L' & P_i \\ & 1 \end{bmatrix} \tag{5.1}$$

where $L' = l_{i1} \cdots l_{i(d-1)}$.

The proof will proceed by induction where Equation 5.1 serves as the induction basis. A generic intermediate matrix would look like $\begin{bmatrix} L_1 & L_2 g \\ & L_3 \end{bmatrix}$ where each L_i is a product of non-zero linear functions and g is a partial summand of P_i's. We shall inductively double the number of summands in g at each step.

At the i-th iteration, assume that we have the matrices $\begin{bmatrix} L_1 & L_2 g \\ & L_3 \end{bmatrix}$ and $\begin{bmatrix} M_1 & M_2 h \\ & M_3 \end{bmatrix}$, each computed by a sequence of n_i linear matrices. We now want a sequence that computes a polynomial of the form $L \cdot (g + h)$. Consider the following sequence,

$$\begin{bmatrix} L_1 & L_2 g \\ & L_3 \end{bmatrix}\begin{bmatrix} A & \\ & B \end{bmatrix}\begin{bmatrix} M_1 & M_2 h \\ & M_3 \end{bmatrix} = \begin{bmatrix} AL_1M_1 & AL_1M_2h + BL_2M_3g \\ & BL_3M_3 \end{bmatrix} \tag{5.2}$$

where A, B are products of linear functions. By setting $A = L_2M_3$ and $B = L_1M_2$ we get the desired sequence,

$$\begin{bmatrix} L_1 & L_2 g \\ & L_3 \end{bmatrix}\begin{bmatrix} A & \\ & B \end{bmatrix}\begin{bmatrix} M_1 & M_2 h \\ & M_3 \end{bmatrix} = \begin{bmatrix} L_1L_2M_1M_3 & L_1L_2M_2M_3(g+h) \\ & L_1L_3M_2M_3 \end{bmatrix}$$

This way, we have doubled the number of summands in $g + h$. The length of the sequence computing $L_2 g$ and $M_2 h$ is n_i, hence each L_i and M_i is a product of n_i many linear functions. Therefore, both A and B are products of at most $2n_i$ linear functions and the matrix $\begin{bmatrix} A & \\ & B \end{bmatrix}$ can be written as a product of at most $2n_i$ diagonal linear matrices. The total length of the sequence given in Equation 5.2 is hence bounded by $4n_i$.

The number of summands in f is s and the above process needs to be repeated at most $\log s + 1$ times. The final sequence length is hence bounded by $(d+n)\cdot 4^{\log s} = (d+n)s^2$. □

Proof of Theorem 1.5. It follows from Lemma 5.1 that, given a depth 3 circuit C computing f we can efficiently construct a depth 2 circuit over $\mathsf{U}_2(\mathbb{F})$ that outputs a matrix, $\begin{bmatrix} L_1 & L\cdot f \\ & L_2 \end{bmatrix}$, where L is a product of non-zero linear functions. Multiplying this matrix by $\begin{bmatrix} 1 & 0 \\ & 0 \end{bmatrix}$ to the left and $\begin{bmatrix} 0 & 0 \\ & 1 \end{bmatrix}$ to the right yields another depth 2 circuit D that outputs $\begin{bmatrix} 0 & L\cdot f \\ & 0 \end{bmatrix}$. Thus D computes an identically zero polynomial over $\mathsf{U}_2(\mathbb{F})$ if and only if C computes an identically zero polynomial. This shows that PIT for depth 3 circuits reduces to PIT of depth 2 circuits over $\mathsf{U}_2(\mathbb{F})$.

The other direction, that is PIT for depth 2 circuits over $\mathsf{U}_2(\mathbb{F})$ reduces to PIT for depth 3 circuits, is trivial to observe. The diagonal entries of the output 2×2 matrix is just a product of linear functions whereas the off-diagonal entry is a sum of at most d' many products of linear functions, where d' is the multiplicative fan-in of the depth 2 circuit over $\mathsf{U}_2(\mathbb{F})$. □

5.1.1 Width-2 algebraic branching programs

The main theorem has an interesting consequence in terms of *algebraic branching programs*. Algebraic Branching Programs (ABPs) is a model of computation defined by Nisan [Nis91].

Formally, an ABP is defined as follows.

***Definition* 5.2.** *(Nisan [Nis91]) An* algebraic branching program (ABP) *is a directed acyclic graph with one source and one sink. The vertices of this graph are partitioned into levels labelled* 0 *to d, where edges may go from level i to level $i+1$. The parameter d is called the degree of the ABP. The source is the only vertex at level* 0 *and the sink is the only vertex at level d. Each edge is labelled with a homogeneous linear function of $x_1, \dots, x_n$ (i.e. a function of the form $\sum_i c_i x_i$). The* width *of the ABP is the maximum number of vertices in any level, and the* size *is the total number of vertices.*

An ABP computes a function in the obvious way; sum over all paths from source to sink, the product of all linear functions by which the edges of the path are labelled.

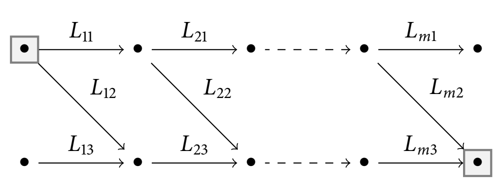
The following argument shows how Corollary 1.6 follows easily from Theorem 1.5.

***Corollary* 1.6. (restated)** Identity testing of depth 3 circuits is equivalent to identity testing of width-2 ABPs with polynomially many paths from source to sink.

Proof. It is firstly trivial to see that if the number of paths from source to sink is small, then we one can easily construct a depth 3 circuit that computes the same polynomial. Thus, we only need to show that PIT on depth 3 circuits reduces to that on such restricted ABP's. Theorem 1.5 constructs a depth 2 circuit D that computes a product of the form

$$P = \begin{bmatrix} L_{11} & L_{12} \\ & L_{13} \end{bmatrix} \begin{bmatrix} L_{21} & L_{22} \\ & L_{23} \end{bmatrix} \cdots \begin{bmatrix} L_{m1} & L_{m2} \\ & L_{m3} \end{bmatrix}$$

where each L_{ij} is a linear function over the variables. We can make sure that all linear functions are homogeneous by introducing an extra variable x_0, such that $a_0 + a_1x_1 + \cdots a_nx_n$ is transformed to $a_0x_0 + a_1x_1 + \ldots + a_nx_n$. It is now straightforward to construct a width-2 ABP by making the j^{th} linear matrix in the sequence act as the adjacency matrix between level j and $j+1$ of the ABP.



It is clear that the branching program has only polynomially many paths from source to sink. $\square$

As a matter of fact, the above argument actually shows that PIT of depth 2 circuits over $\mathcal{M}_k(\mathbb{F})$ is equivalent to PIT of width-k ABPs.

5.2 Identity testing over commutative algebras

We would now prove Theorem 1.7. The main idea behind this proof is a structure theorem for finite dimensional commutative algebras over a field. To state the theorem we need the following definition.

***Definition* 5.3.** *A ring $\mathcal{R}$ is* local *if it has a unique maximal ideal.*

An element u in a ring $\mathcal{R}$ is said to be a *unit* if there exist an element u' such that $uu' = 1$, where 1 is the identity element of $\mathcal{R}$. An element $m \in \mathcal{R}$ is *nilpotent* if there exist a positive integer n with $m^n = 0$. In a local ring the unique maximal ideal consists of all non-units in $\mathcal{R}$.

The following theorem shows how a commutative algebra decomposes into local sub-algebras. The theorem is quite well known in the theory of commutative algebras. But since we need an effective version of this theorem, we present the proof here for the sake of completion and clarity.

***Theorem* 5.4.** *A finite dimensional commutative algebra $\mathcal{R}$ over $\mathbb{F}$ is isomorphic to a direct sum of local rings i.e.*

$$\mathcal{R} \cong \mathcal{R}_1 \oplus \ldots \oplus \mathcal{R}_\ell$$

where each $\mathcal{R}_i$ is a local ring contained in $\mathcal{R}$ and any non-unit in $\mathcal{R}_i$ is nilpotent.

Proof. If all non-units in $\mathcal{R}$ are nilpotents then $\mathcal{R}$ is a local ring and the set of nilpotents forms the unique maximal ideal. Therefore, suppose that there is a non-nilpotent zero-divisor z in $\mathcal{R}$. (Any non-unit z in a finite dimensional algebra is a zero-divisor i.e. $\exists y \in \mathcal{R}$ and $y \neq 0$ such that $yz = 0$.) We would argue that using z we can find an *idempotent* $v \notin \{0,1\}$ in $\mathcal{R}$ i.e. $v^2 = v$.

Assume that we do have a non-trivial idempotent $v \in \mathcal{R}$. Let $\mathcal{R}v$ be the sub-algebra of $\mathcal{R}$ generated by multiplying elements of $\mathcal{R}$ with v. Since any $a = av + a(1-v)$ and $\mathcal{R}v \cap \mathcal{R}(1-v) = \{0\}$, we get $\mathcal{R} \cong \mathcal{R}v \oplus \mathcal{R}(1-v)$ as a non-trivial decomposition of $\mathcal{R}$. (Note that $\mathcal{R}$ is a direct sum of the two sub-algebras because for any $a \in \mathcal{R}v$ and $b \in \mathcal{R}(1-v)$, $a \cdot b = 0$. This is the place where we use commutativity of $\mathcal{R}$.) By repeating the splitting process on the sub-algebras we can eventually prove the theorem. We now show how to find an idempotent from the zero-divisor z.

An element $a \in \mathcal{R}$ can be expressed equivalently as a matrix in $\mathcal{M}_k(\mathbb{F})$, where $k = \dim_{\mathbb{F}}(\mathcal{R})$, by treating a as the linear transformation on $\mathcal{R}$ that takes $b \in \mathcal{R}$ to $a \cdot b$. Therefore, z is a zero-divisor if and only if z as a matrix is singular. Consider the Jordan normal form of z. Since it is merely a change of basis we would assume, without loss of generality, that z is already in Jordan normal form. (We won't compute the Jordan normal form in our algorithm, it is used only for the sake of argument.) Let,

$$z = \begin{bmatrix} A & 0 \\ 0 & N \end{bmatrix}$$

where A, N are block diagonal matrices and A is non-singular and N is nilpotent. Therefore there exits a positive integer $t < k$ such that,

$$w = z^t = \begin{bmatrix} B & 0 \\ 0 & 0 \end{bmatrix}$$

where $B = A^t$ is non-singular. The claim is, there is an identity element in the sub-algebra $\mathcal{R}w$ which can be taken to be the idempotent that splits $\mathcal{R}$. To see this first observe that the minimum polynomial of w over $\mathbb{F}$ is $m(x) = x \cdot m'(x)$, where $m'(x)$ is the minimum polynomial of B. Also if $m(x) = \sum_{i=1}^{k} \alpha_i x^i$ then $\alpha_1 \neq 0$ as it is the constant term of $m'(x)$ and B is non-singular. Therefore, there exists an $a \in \mathcal{R}$ such that $w \cdot (aw - 1) = 0$. We can take $v = aw$ as the identity element in the sub-algebra $\mathcal{R}w$. This $v \notin \{0, 1\}$ is the required idempotent in $\mathcal{R}$. □

We are now ready to prove Theorem 1.7.

***Theorem* 1.7** (restated.) *Given an expression,*

$$P = \prod_{i=1}^{d} (A_{i0} + A_{i1}x_1 + \ldots + A_{in}x_n)$$

where $A_{ij} \in \mathcal{R}$, a commutative algebra of constant dimension over $\mathbb{F}$ that is given in basis form, there is a deterministic polynomial time algorithm to test if P is zero.

Proof. Suppose, the elements $e_1, \ldots, e_k$ form a basis of $\mathcal{R}$ over $\mathbb{F}$. Since any element in $\mathcal{R}$ can be equivalently expressed as a $k \times k$ matrix over $\mathbb{F}$ (by treating it as a linear transformation), we will assume that $A_{ij} \in \mathcal{M}_k(\mathbb{F})$, for all i and j. Further, since $\mathcal{R}$ is given in basis form, we can find these matrix representations of A_{ij}'s efficiently.

If every A_{ij} is non-singular, then surely $P \neq 0$. (This can be argued by fixing an ordering $x_1 > x_2 > \ldots > x_n$ among the variables. The coefficient of the leading monomial of P, with respect to this ordering, is a product of invertible matrices and hence $P \neq 0$.) Therefore, assume that $\exists A_{ij} = z$ such that z is a zero-divisor i.e. singular. From the proof of Theorem 5.4 it follows that there exists a $t < k$ such that the sub-algebra $\mathcal{R}w$, where $w = z^t$, contains an identity element v which is an idempotent. To find the right w we can simply go through all $1 \leq t < k$. We now argue that for the correct choice of w, v can be found by solving a system of linear equations over $\mathbb{F}$. Let $b_1, \ldots, b_{k'}$ be a basis of $\mathcal{R}w$, which we can find easily from the elements $e_1 w, \ldots, e_k w$. In order to solve for v write it as,

$$v = v_1 b_1 + \ldots + v_{k'} b_{k'}$$

where $v_j \in \mathbb{F}$ are unknowns. Since v is an identity in $\mathcal{R}w$ we have the following equations,

$$(v_1 b_1 + \ldots + v_{k'} b_{k'}) \cdot b_i = b_i \quad \text{for } 1 \leq i \leq k'.$$

Expressing each b_i in terms of $e_1, \ldots, e_k$, we get a set of linear equations in v_j's. Thus for the right choice of w (i.e. for the right choice of t) there is a solution for v. On the other hand, a solution for v for any w gives us an idempotent, which is all that we need.

Since $\mathcal{R} \cong \mathcal{R}v \oplus \mathcal{R}(1-v)$ we can now split the identity testing problem into two similar problems, i.e. P is zero if and only if,

$$\begin{aligned} Pv &= \prod_{i=1}^{d} (A_{i0}v + A_{i1}v \cdot x_1 + \ldots + A_{in}v \cdot x_n) \quad \text{and} \\ P(1-v) &= \prod_{i=1}^{d} (A_{i0}(1-v) + A_{i1}(1-v) \cdot x_1 + \ldots + A_{in}(1-v) \cdot x_n) \end{aligned}$$

are both zero. What we just did with $P \in \mathcal{R}$ we can repeat for $Pv \in \mathcal{R}v$ and $P(1-v) \in \mathcal{R}(1-v)$. By decomposing the algebra each time an A_{ij} is a non-nilpotent zero-divisor, we have reduced the problem to the following easier problem of checking if

$$P = \prod_{i=1}^{d} (A_{i0} + A_{i1}x_1 + \ldots + A_{in}x_n)$$

is zero, where the coefficients A_{ij}'s are either nilpotent or invertible matrices.

Let $T_i = (A_{i0} + A_{i1}x_1 + \ldots + A_{in}x_n)$ be a term such that the coefficient of x_j in T_i, i.e. A_{ij} is invertible. And suppose Q be the product of all terms other than T_i. Then $P = T_i \cdot Q$ (since $\mathcal{R}$ is commutative). Fix an ordering among the variables so that x_j gets the highest priority. The leading coefficient of P, under this ordering, is A_{ij} times the leading coefficient of Q. Since A_{ij} is invertible this implies that $P = 0$ if and only if $Q = 0$. (If A_{i0} is invertible, we can arrive at the same conclusion by arguing with the coefficients of the least monomials of P and Q under some ordering.) In other words, $P = 0$ if and only if the product of all those terms for which all the coefficients are nilpotent matrices is zero. But this is easy to check since the dimension of the algebra, k is a constant. (In fact, this is the only step where we use that k is a constant.) If number of such terms is greater than k then P is automatically zero (this follows easily from the fact that the commuting nilpotent matrices can be simultaneously triangularized with zeroes in the diagonal). Otherwise, simply multiply those terms and check if it is zero. This takes $O(n^k)$ operations over $\mathbb{F}$. □

It is clear from the above discussion that identity testing of depth 2 $(\Pi\Sigma)$ circuits over commutative algebras reduces in polynomial time to that over local rings. As long as the dimensions of these local rings are constant we are through. But what happens for non-constant dimensions? The following result justifies the hardness of this problem.

***Theorem* 5.5.** *Given a depth* 3 $(\Sigma\Pi\Sigma)$ *circuit C of degree d and top level fan-in s, it is possible to construct in polynomial time a depth* 2 $(\Pi\Sigma)$ *circuit $\tilde{C}$ over a local ring of dimension* $s(d-1)+2$ *over* $\mathbb{F}$ *such that $\tilde{C}$ computes a zero polynomial if and only if C does so.*

Proof. The proof is relatively straightforward. Consider a depth 3 $(\Sigma\Pi\Sigma)$ circuit computing a polynomial $f = \sum_{i=1}^{s} \prod_{j=1}^{d} l_{ij}$, where l_{ij}'s are linear functions. Consider the ring $\mathcal{R} = \mathbb{F}[y_1, \ldots, y_s]/\mathcal{I}$, where $\mathcal{I}$ is an ideal generated by the elements $\{y_i y_j\}_{1\le i<j\le s}$ and $\{y_1^d - y_i^d\}_{1<i\le s}$. Observe that $\mathcal{R}$ is a local ring, as $y_i^{d+1} = 0$ for all $1 \le i \le s$. Also the elements $\{1, y_1, \ldots, y_1^d, y_2, \ldots, y_2^{d-1}, \ldots, y_s, \ldots, y_s^{d-1}\}$ form an $\mathbb{F}$-basis of $\mathcal{R}$. Now notice that the polynomial,

$$\begin{aligned} P &= \prod_{j=1}^{d} \left(l_{j1}y_1 + \ldots + l_{js}y_s \right) \\ &= f \cdot y_1^d \end{aligned}$$

is zero if and only if f is zero. Polynomial P can indeed be computed by a depth 2 ($\Pi\Sigma$) circuit over $\mathcal{R}$. $\square$

5.3 Weakness of the depth 2 model

In Lemma 5.1, we saw that the depth 2 circuit over $\mathsf{U}_2(\mathbb{F})$ computes $L \cdot f$ instead of f. Is it possible to drop the factor L and simply compute f? In this section, we show that in *many* cases it is impossible to find a depth 2 circuit over $\mathsf{U}_2(\mathbb{F})$ that computes f.

5.3.1 Depth 2 model over $\mathsf{U}_2(\mathbb{F})$

We will now prove Theorem 1.8. In the following discussion we use the notation (l_1, l_2) to mean the ideal generated by two linear functions l_1 and l_2. Further, we say that l_1 is *independent* of l_2 if $1 \notin (l_1, l_2)$.

***Theorem* 1.8** (restated.) *Let $f \in F[x_1, \ldots, x_n]$ be a polynomial such that there are no two linear functions l_1 and l_2 (with $1 \notin (l_1, l_2)$) which make $f \bmod (l_1, l_2)$ also a linear function. Then f is not computable by a depth 2 circuit over $\mathsf{U}_2(\mathbb{F})$.*

Proof. Assume on the contrary that f can be computed by a depth 2 circuit over $\mathsf{U}_2(\mathbb{F})$. In other words, there is a product sequence $M_1 \cdots M_t$ of 2×2 upper-triangular linear matrices such that f appears as the top-right entry of the final product. Let $M_i = \begin{bmatrix} l_{i1} & l_{i2} \\ & l_{i3} \end{bmatrix}$, then

$$f = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} l_{11} & l_{12} \\ & l_{13} \end{bmatrix} \begin{bmatrix} l_{21} & l_{22} \\ & l_{23} \end{bmatrix} \cdots \begin{bmatrix} l_{t1} & l_{t2} \\ & l_{t3} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{5.3}$$

Case 1: Not all the l_{i1}'s are constants.

Let k be the least index such that l_{k1} is not a constant and $l_{i1} = c_i$ for all $i < k$. To simplify Equation 5.3, let

$$\begin{aligned} \begin{bmatrix} B \\ L \end{bmatrix} &= M_{k+1} \cdots M_t \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\ \begin{bmatrix} d_i & D_i \end{bmatrix} &= \begin{bmatrix} 1 & 0 \end{bmatrix} \cdot M_1 \cdots M_{i-1} \end{aligned}$$

Observe that L is just a product of linear functions, and for all $1 \le i < k$, we have the following relations.

$$\begin{aligned} d_{i+1} &= \prod_{j=1}^{i} c_j \\ D_{i+1} &= d_i l_{i2} + l_{i3} D_i \end{aligned}$$

Hence, Equation 5.3 simplifies as

$$\begin{aligned} f &= \begin{bmatrix} d_k & D_k \end{bmatrix} \begin{bmatrix} l_{k1} & l_{k2} \\ & l_{k3} \end{bmatrix} \begin{bmatrix} B \\ L \end{bmatrix} \\ &= d_k l_{k1} B + (d_k l_{k2} + l_{k3} D_k) L \end{aligned}$$

Suppose there is some factor l of L with $1 \notin (l_{k1}, l)$. Then $f = 0 \bmod (l_{k1}, l)$, which is not possible. Hence, L must be a constant modulo l_{k1}. For appropriate constants α, β, we have

$$f = \alpha l_{k2} + \beta l_{k3} D_k \pmod{l_{k1}} \tag{5.4}$$

We argue that the above equation cannot be true by inducting on k. If l_{k3} was independent of l_{k1}, then $f = \alpha l_{k2} \bmod (l_{k1}, l_{k3})$ which is not possible. Therefore, l_{k3} must be a constant modulo l_{k1}. We then have the following (reusing α and β to denote appropriate constants):

$$\begin{aligned} f &= \alpha l_{k2} + \beta D_k \pmod{l_{k1}} \\ &= \alpha l_{k2} + \beta \left(d_{k-1} l_{(k-1)2} + l_{(k-1)3} D_{k-1} \right) \pmod{l_{k1}} \\ \Longrightarrow f &= \left(\alpha l_{k2} + \beta d_{k-1} l_{(k-1)2} \right) + \beta l_{(k-1)3} D_{k-1} \pmod{l_{k1}} \end{aligned}$$

The last equation can be rewritten in the form of Equation 5.4 with $\beta l_{k3} D_k$ replaced by $\beta l_{(k-1)3} D_{k-1}$. Notice that the expression $\left(\alpha l_{k2} + \beta d_{k-1} l_{(k-1)2} \right)$ is linear just like αl_{k2}. Hence by using the argument iteratively we eventually get a contradiction at D_1.

Case 2: All the l_{i1}'s are constants.

In this case, Equation 5.3 can be rewritten as

$$\begin{aligned} f &= \begin{bmatrix} d_t & D_t \end{bmatrix} \begin{bmatrix} c_t & l_{t2} \\ & l_{t3} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\ &= d_t l_{t2} + l_{t3} D_t \end{aligned}$$

The last equation is again of the form in Equation 5.4 (without the mod term) and hence the same argument can be repeated here as well to give the desired contradiction. □

The following corollary provides some explicit examples of functions that cannot be computed.

***Corollary* 5.6.** *A depth 2 circuit over* $\mathsf{U}_2(\mathbb{F})$ *cannot compute the polynomial* $x_1x_2 + x_3x_4 + x_5x_6$. *Other examples include well known functions like* det_n *and* perm_n, *the determinant and permanent polynomials, for* $n \geq 3$.

Proof. It suffices to show that $f = x_1x_2 + x_3x_4 + x_5x_6$ satisfy the requirement in Theorem 1.8.

To obtain a contradiction, let us assume that there does exist two linear functions l_1 and l_2 (with $1 \notin (l_1, l_2)$) such that f mod (l_1, l_2) is linear. We can evaluate f mod (l_1, l_2) by substituting a pair of the variables in f by linear functions in the rest of the variables (as dictated by the equations $l_1 = l_2 = 0$). By the symmetry of f, we can assume that the pair is either $\{x_1, x_2\}$ or $\{x_1, x_3\}$.

If $x_1 = l_1'$ and $x_3 = l_2'$ are the substitutions, then $l_1'x_2 + l_2'x_4$ can never contribute a term to cancel off x_5x_6 and hence f mod (l_1, l_2) cannot be linear.

Otherwise, let $x_1 = l_1'$ and $x_2 = l_2'$ be the substitutions. If f mod $(l_1, l_2) = l_1'l_2' + x_3x_4 + x_5x_6$ is linear, there cannot be a common x_i with non-zero coefficient in both l_1' and l_2'. Without loss of generality, assume that l_1' involves x_3 and x_5 and l_2' involves x_4 and x_6. But then the product $l_1'l_2'$ would involve terms like x_3x_6 that cannot be cancelled, contradicting linearity again. □

5.3.2 Depth 2 model over $\mathcal{M}_2(\mathbb{F})$

In this section we show that the power of depth 2 circuits is very restrictive even if we take the underlying algebra to be $\mathcal{M}_2(\mathbb{F})$ instead of $\mathsf{U}_2(\mathbb{F})$. In the following discussion, we will refer to a homogeneous linear function as a *linear form*.

***Definition* 5.7.** *A polynomial f of degree n is said to be r-*robust *if f does not belong to any ideal generated by r linear forms.*

For instance, it can be checked that det_n and perm_n, the symbolic determinant and permanent of an $n \times n$ matrix, are $(n-1)$-robust polynomials. For any polynomial f, we will denote the d^{th} homogeneous part of f by $[f]_d$. And let $(h_1, \cdots, h_k)$ denote the ideal generated by $h_1, \cdots, h_k$. For the following theorem recall the definition of *degree restriction* (Definition 1.9) given in the introduction.

***Theorem* 5.8.** *A polynomial f of degree n, such that $[f]_n$ is 5-robust, cannot be computed by a depth 2 circuit over $\mathcal{M}_2(\mathbb{F})$ under a degree restriction of n.*

We prove this with the help of the following lemma, which basically applies Gaussian column operations to simplify matrices.

***Lemma* 5.9.** *Let f_1 be a polynomial of degree n such that $[f_1]_n$ is 4-robust. Suppose there is a linear matrix M and polynomials f_2, g_1, g_2 of degree at most n satisfying*

$$\begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = M \begin{bmatrix} g_1 \\ g_2 \end{bmatrix}$$

Then, there is an appropriate invertible column operation A such that

$$M \cdot A = \begin{bmatrix} 1 & h_2 \\ c_3 & h_4 + c_4 \end{bmatrix}$$

where c_3, c_4 are constants and h_2, h_4 are linear forms.

We will defer the proof of this lemma to the end of this section, and shall use it to prove Theorem 5.8.

Proof of Theorem 5.8. Assume, on the contrary, that we do have such a sequence of matrices computing f. Since only one entry is of interest to us, we shall assume that the first matrix is a row vector and the last matrix is a column vector. Let the sequence of minimum length computing f be the following:

$$f = \bar{v} \cdot M_1 M_2 \cdots M_d \cdot \bar{w}$$

Using Lemma 5.9 we shall repeatedly transform the above sequence by replacing $M_i M_{i+1}$ by $(M_i A)(A^{-1} M_{i+1})$ for an appropriate invertible column transformation A. Since A would consist of just constant entries, $M_i A$ and $A^{-1} M_{i+1}$ continue to be linear matrices.

To begin, let $\bar{v} = [l_1, l_2]$ for two linear functions l_1 and l_2. And let $[f_1, f_2]^T = M_1 \cdots M_d \bar{w}$. Then we have,

$$\begin{bmatrix} f \\ 0 \end{bmatrix} = \begin{bmatrix} l_1 & l_2 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}$$

Hence, by Lemma 5.9, we can assume $\bar{v} = [1, h]$ and hence $f = f_1 + hf_2$. By the minimality of the sequence, $h \neq 0$. This forces f_1 to be 4-robust and the degree restriction makes $[f_2]_n = 0$.

Let $[g_1, g_2]^T = M_2 \cdots M_d \bar{w}$. The goal is to translate the properties that $[f_1]_n$ is 4-robust and $[f_2]_n = 0$ to the polynomials g_1 and g_2. Translating these properties would show each M_i is of the form described in Lemma 5.9. Thus, inducting on the length of the sequence, we would arrive at the required contradiction. In general, we have an equation of the form

$$\begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = M_i \begin{bmatrix} g_1 \\ g_2 \end{bmatrix}$$

Since $[f_1]_n$ is 4-robust, using Lemma 5.9 again, we can assume that

$$\begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} 1 & h_2 \\ c_3 & c_4 + h_4 \end{bmatrix} \begin{bmatrix} g_1 \\ g_2 \end{bmatrix} \tag{5.5}$$

by reusing the variables g_1, g_2 and others. Observe that in the above equation if $h_4 = 0$ then $M_{i-1}M_i$ still continues to be a linear matrix (since, by induction, M_{i-1} is of the form as dictated by Lemma 5.9) and that would contradict the minimality of the sequence. Therefore $h_4 \neq 0$.

Claim: $c_3 = 0$ (by comparing the n^{th} homogeneous parts of f_1 and g_1, as explained below).
Proof: As $h_4 \neq 0$, the degree restriction forces $\deg g_2 < n$. And since $\deg f_2 < n$, we have the relation $c_3[g_1]_n = -h_4[g_2]_{n-1}$. If $c_3 \neq 0$, we have $[g_1]_n \in (h_4)$, contradicting robustness of $[f_1]_n$ as then $[f_1]_n = [g_1]_n + h_2[g_2]_{n-1} \in (h_2, h_4)$. □

Therefore Equation 5.5 gives,

$$\begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} 1 & h_2 \\ 0 & c_4 + h_4 \end{bmatrix} \begin{bmatrix} g_1 \\ g_2 \end{bmatrix}$$

with $h_4 \neq 0$. Also, since $[f_2]_{n+1} = [f_2]_n = 0$ this implies that $[g_2]_n = [g_2]_{n-1} = 0$. Hence, $[g_1]_n = [f_1]_n$ is 4-robust. This argument can be extended now to g_1 and g_2. Notice that the degree of g_1 remains n. However, since there are only finitely many matrices in the sequence, there must come a point when this degree drops below n. At this point we get a contradiction as $[g_1]_n = 0$ (reusing symbol) which contradicts robustness. □

We only need to finish the proof of Lemma 5.9.

Proof of Lemma 5.9. Suppose we have an equation of the form

$$\begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} h_1 + c_1 & h_2 + c_2 \\ h_3 + c_3 & h_4 + c_4 \end{bmatrix} \begin{bmatrix} g_1 \\ g_2 \end{bmatrix} \tag{5.6}$$

On comparing degree $n+1$ terms, we have

$$\begin{aligned} h_1[g_1]_n + h_2[g_2]_n &= 0 \\ h_3[g_1]_n + h_4[g_2]_n &= 0 \end{aligned}$$

If h_3 and h_4 (a similar reasoning holds for h_1 and h_2) were not proportional (i.e. not multiple of each other), then the above equation would imply $[g_1]_n, [g_2]_n \in (h_3, h_4)$. Then,

$$[f_1]_n = h_1[g_1]_{n-1} + h_2[g_2]_{n-1} + c_1[g_1]_n + c_2[g_2]_n \in (h_1, h_2, h_3, h_4)$$

contradicting the robustness of $[f_1]_n$. Thus, h_3 and h_4 (as well as h_1 and h_2) are proportional, in the same ratio as $[-g_2]_n$ and $[g_1]_n$. Using an appropriate column operation, Equation 5.6 simplifies to

$$\begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} c_1 & h_2 + c_2 \\ c_3 & h_4 + c_4 \end{bmatrix} \begin{bmatrix} g_1 \\ g_2 \end{bmatrix}$$

If $c_1 = 0$, then together with $[g_2]_n = 0$ we get $[f_1]_n = h_2[g_2]_{n-1}$ contradicting robustness. Therefore $c_1 \neq 0$ and another column transformation would get it to the form claimed. $\square$

6 Conclusion

A deterministic algorithm for PIT continues to evade various attempts by researchers. Numerous ideas have been employed for randomized algorithms and for deterministic algorithms in restricted settings. Partial evidence for the problem's hardness has also be provided. In this thesis, we shed some more light on the problem and a possible attack on general $\Sigma\Pi\Sigma$ circuits.

We give a new perspective to identity testing of depth 3 arithmetic circuits by showing an equivalence to identity testing of depth 2 circuits over $\mathsf{U}_2(\mathbb{F})$. The reduction implies that identity testing of a width-2 algebraic branching program is at least as hard as identity testing of depth 3 circuits.

The characterization in terms of depth 2 circuits over $\mathsf{U}_2(\mathbb{F})$ seem more vulnerable than general depth 3 circuits. Can we obtain new (perhaps easier) proofs of known results using the characterization in terms of linear matrices, or width 2-ABPs?

We also give a deterministic polynomial time identity testing algorithm for depth 2 circuits over any constant dimensional commutative algebra. Our algorithm crucially exploits an interesting structural result involving local rings. This naturally poses the following question — Can we use more algebraic insight on non-commutative algebras to solve the general problem? The solution for the commutative case does not seem to give any interesting insight into the non-commutative case. But we have a very specific non-commutative case at hand. The question is - Is it possible to use properties very specific to the ring of 2×2 matrices to solve identity testing for depth 3 circuits?

We also show that PIT over depth 2 circuits over higher dimensional commutative algebras capture $\Sigma\Pi\Sigma$ circuits completely. Does this case falter to mathematics as well? Can this approach be used to get a deterministic polynomial time PIT for depth 3 circuits.

Hopefully, the new ideas outlined in this thesis renders useful to an algorithm for depth 3 circuits.

Bibliography

[AB99] Manindra Agrawal and Somenath Biswas. Primality and Identity Testing via Chinese Remaindering. In *FOCS*, pages 202–209, 1999. [5, 14]

[Agr05] Manindra Agrawal. Proving Lower Bounds Via Pseudo-random Generators. In *FSTTCS*, pages 92–105, 2005. [3, 5, 35]

[AJMV98] Eric Allender, Jia Jiao, Meena Mahajan, and V. Vinay. Non-Commutative Arithmetic Circuits: Depth Reduction and Size Lower Bounds. *Theor. Comput. Sci.*, 209(1-2):47–86, 1998. [31]

[AKS04] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. *Ann. of Math*, 160(2):781–793, 2004. [5]

[ALM+98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof Verification and the Hardness of Approximation Problems. *Journal of the ACM*, 45(3):501–555, 1998. [5]

[AV08] Manindra Agrawal and V Vinay. Arithmetic circuits: A chasm at depth four. In *FOCS*, pages 67–75, 2008. [5, 9, 31, 37]

[BC88] Michael Ben-Or and Richard Cleve. Computing Algebraic Formulas Using a Constant Number of Registers. In *STOC*, pages 254–257, 1988. [6]

[Ber84] Stuart J. Berkowitz. On computing the determinant in small parallel time using a small number of processors. *Inf. Process. Lett.*, 18(3):147–150, 1984. [2]

[CDGK91] Michael Clausen, Andreas W. M. Dress, Johannes Grabmeier, and Marek Karpinski. On Zero-Testing and Interpolation of k-Sparse Multivariate Polynomials Over Finite Fields. *Theor. Comput. Sci.*, 84(2):151–164, 1991. [5]

[Chi85] Alexander L. Chistov. Fast parallel calculation of the rank of matrices over a field of arbitrary characteristic. In *FCT '85: Fundamentals of Computation Theory*, pages 63–69, London, UK, 1985. Springer-Verlag. [2]

[CK97] Zhi-Zhong Chen and Ming-Yang Kao. Reducing Randomness via Irrational Numbers. In *STOC*, pages 200–209, 1997. [5, 12]

[CRS95] Suresh Chari, Pankaj Rohatgi, and Aravind Srinivasan. Randomness-optimal unique element isolation with applications to perfect matching and related problems. *SIAM J. Comput.*, 24(5):1036–1050, 1995. [17]

[DS05] Zeev Dvir and Amir Shpilka. Locally decodable codes with 2 queries and polynomial identity testing for depth 3 circuits. In *STOC '05: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 592–601, New York, NY, USA, 2005. ACM. [27]

[GK98] Dima Grigoriev and Marek Karpinski. An exponential lower bound for depth 3 arithmetic circuits. In *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 577–582, New York, NY, USA, 1998. ACM. [5]

[GR05] Ariel Gabizon and Ran Raz. Deterministic extractors for affine sources over large fields. In *FOCS '05: Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 407–418, Washington, DC, USA, 2005. IEEE Computer Society. [28]

[KI03] Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. In *STOC*, pages 355–364, 2003. [3, 5, 36]

[KS01] Adam Klivans and Daniel A. Spielman. Randomness efficient identity testing of multivariate polynomials. In *STOC*, pages 216–223, 2001. [5, 17]

[KS07] Neeraj Kayal and Nitin Saxena. Polynomial Identity Testing for Depth 3 Circuits. *Computational Complexity*, 16(2), 2007. [5, 22, 23, 27]

[KS08] Zohar S. Karnin and Amir Shpilka. Black box polynomial identity testing of generalized depth-3 arithmetic circuits with bounded top fan-in. In *CCC '08: Proceedings of the 2008 IEEE 23rd Annual Conference on Computational Complexity*, pages 280–291, Washington, DC, USA, 2008. IEEE Computer Society. [27, 28]

[KS09] Neeraj Kayal and Shubhangi Saraf. Blackbox polynomial identity testing for depth 3 circuits. *Electronic Colloquium on Computational Complexity (ECCC)*, 2009. [5, 27]

[Lov79] László Lovász. On determinants, matchings, and random algorithms. In *FCT*, pages 565–574, 1979. [5]

[LV98] Daniel Lewin and Salil P. Vadhan. Checking Polynomial Identities over any Field: Towards a Derandomization? In *STOC*, pages 438–447, 1998. [5, 13]

[MV97] Meena Mahajan and V. Vinay. Determinant: Combinatorics, algorithms, and complexity. *Chicago J. Theor. Comput. Sci.*, 1997, 1997. [2]

[MVV87] Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. In *STOC '87: Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 345–354, New York, NY, USA, 1987. ACM. []

[Nis91] Noam Nisan. Lower bounds for non-commutative computation. In *STOC*, pages 410–418, 1991. [7, 41, 42]

[NW94] Noam Nisan and Avi Wigderson. Hardness vs. randomness. *Jounal of Computer Systems and Sciences*, 49:149–167, 2 1994. [36]

[NW95] N. Nisan and A. Wigderson. Lower bounds on arithmetic circuits via partial derivatives. In *FOCS '95: Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, page 16, Washington, DC, USA, 1995. IEEE Computer Society. [5]

[RS04] Ran Raz and Amir Shpilka. Deterministic Polynomial Identity Testing in Non-Commutative Models. In *IEEE Conference on Computational Complexity*, pages 215–222, 2004. [7, 29]

[RY08] Ran Raz and Amir Yehudayoff. Multilinear formulas, maximal-partition discrepancy and mixed-sources extractors. In *FOCS '08: Proceedings of the 2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 273–282, Washington, DC, USA, 2008. IEEE Computer Society. [4]

[Sah08] Chandan Saha. A note on irreducible polynomials and identity testing. (Manuscript) http://www.cse.iitk.ac.in/users/csaha/PID_CR.pdf, 2008. [16]

[Sam42] P. A. Samuelson. A method of determining explicitly the coefficients of the characteristic polynomial. *Ann. Math. Stat.*, 13(2):424–429, 1942. [2]

[Sax08] Nitin Saxena. Diagonal circuit identity testing and lower bounds. In *ICALP '08: Proceedings of the 35th international colloquium on Automata, Languages and Programming, Part I*, pages 60–71, Berlin, Heidelberg, 2008. Springer-Verlag. [5, 30]

[Sch80] Jacob T. Schwartz. Fast Probabilistic Algorithms for Verification of Polynomial Identities. *J. ACM*, 27(4):701–717, 1980. [5, 11]

[Sha90] Adi Shamir. IP=PSPACE. In *FOCS*, pages 11–15, 1990. [5]

[SS09] Nitin Saxena and C. Seshadhri. An almost optimal rank bound for depth 3 identities. In *Conference on Computational Complexity*, 2009. [27]

[SSS09] Chandan Saha, Ramprasad Saptharishi, and Nitin Saxena. The power of depth 2 circuits over algebras. (Manuscript) http://arxiv.org/abs/0904.2058, 2009. [39]

[SW99] Amir Shpilka and Avi Wigderson. Depth-3 arithmetic formulae over fields of characteristic zero. In *COCO '99: Proceedings of the Fourteenth Annual IEEE Conference on Computational Complexity*, page 87, Washington, DC, USA, 1999. IEEE Computer Society. [5]

[Tod91] S. Toda. Counting problems computationally equivalent to determinant. (Manuscript), 1991. [2]

[Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. *EUROSAM*, pages 216–226, 1979. [5, 11]