The Power of Depth 2 Circuits over Algebras

Chandan Saha[1], **Ramprasad Saptharishi**[2*] **and Nitin Saxena**[3]

[1]Indian Institute of Technology, Kanpur 208016, India
csaha@iitk.ac.in

[2]Chennai Mathematical Institute, Chennai 603103, India
ramprasad@cmi.ac.in

[3]Hausdorff Center for Mathematics, Bonn 53115, Germany
ns@hcm.uni-bonn.de

ABSTRACT.
We study the problem of polynomial identity testing (PIT) for depth 2 arithmetic circuits over matrix algebra. We show that identity testing of depth 3 ($\Sigma\Pi\Sigma$) arithmetic circuits over a field $\mathbb{F}$ is polynomial time equivalent to identity testing of depth 2 ($\Pi\Sigma$) arithmetic circuits over $U_2(\mathbb{F})$, the algebra of upper-triangular 2×2 matrices with entries from $\mathbb{F}$. Such a connection is a bit surprising since we also show that, as computational models, $\Pi\Sigma$ circuits over $U_2(\mathbb{F})$ are strictly 'weaker' than $\Sigma\Pi\Sigma$ circuits over $\mathbb{F}$. The equivalence further implies that PIT of $\Sigma\Pi\Sigma$ circuits reduces to PIT of width-2 commutative *Algebraic Branching Programs*(ABP). Further, we give a deterministic polynomial time identity testing algorithm for a $\Pi\Sigma$ circuit of size s over commutative algebras of dimension $O(\log s / \log\log s)$ over $\mathbb{F}$. Over commutative algebras of dimension $\mathrm{poly}(s)$, we show that identity testing of $\Pi\Sigma$ circuits is at least as hard as that of $\Sigma\Pi\Sigma$ circuits over $\mathbb{F}$.

1 Introduction

Polynomial identity testing (PIT) is a fundamental problem in theoretical computer science. Over the last decade this problem has drawn significant attention from many leading researchers owing to its role in designing efficient algorithms and in proving circuit lower bounds. Identity testing is the following problem:

PROBLEM 1. *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 by C is identically zero.*

Besides being a natural problem in algebraic computation, identity testing appears in important complexity theory results such as, $\mathsf{IP} = \mathsf{PSPACE}$ [Sha90] and the PCP theorem [ALM+98]. It also plays a promising role in proving super-polynomial circuit lower bound for permanent [KI03, Agr05]. Moreover, algorithms for problems like primality testing [AKS04], graph matching [Lov79] and multivariate polynomial interpolation [CDGK91] also involve identity testing. Several efficient randomized algorithms [Sch80, Zip79, CK97, LV98, AB99, KS01] are known for identity testing. However, despite many attempts a deterministic polynomial time algorithm has remained elusive. Nevertheless, important progress

*Supported by MSR India PhD Fellowship

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.

Assume that a circuit C has alternate layers of addition and multiplication gates. A layer of addition gates is denoted by Σ and that of multiplication gates is denoted by Π. Kayal and Saxena [KS07] gave a deterministic polynomial time identity testing algorithm for depth 3 ($\Sigma\Pi\Sigma$) circuits with constant top fan-in. As such, no other general polynomial time result is known for depth 3 circuits. A justification behind the hardness of PIT even for small depth circuits was provided by Agrawal and Vinay [AV08]. They showed that a deterministic black box identity test for depth 4 ($\Sigma\Pi\Sigma\Pi$) circuits implies a quasi-polynomial time deterministic PIT algorithm for circuits computing polynomials of *low* degree[†].

Thus, the non-trivial case for identity testing starts with depth 3 circuits; whereas circuits of depth 4 are *almost* the general case. At this point, it is natural to ask as to what is the complexity of the PIT problem for depth 2 ($\Pi\Sigma$) circuits if we allow the *constants* of the circuit to come from an *algebra* [‡] $\mathcal{R}$ that has dimension over $\mathbb{F}$, $\dim_{\mathbb{F}}(\mathcal{R}) > 1$. Can we relate this problem to the classical PIT problem for depth 3 and depth 4 circuits? In this paper, we address and answer this question. We assume that the algebra $\mathcal{R}$ is given in *basis form* i.e. we know an $\mathbb{F}$-basis $\{e_1, \ldots, e_k\}$ of $\mathcal{R}$ and we also know how $e_i e_j$ can be expressed in terms of the basis elements, for all i and j. Since elements of a finite dimensional algebra, given in basis form, can be expressed as matrices over $\mathbb{F}$, the problem at hand is the following.

PROBLEM 2. *Given an expression $P = \prod_{i=1}^{d} \sum_{j=0}^{n} A_{ij} x_j$ with $x_0 = 1$ and $A_{ij} \in \mathsf{M}_k(\mathbb{F})$, the algebra of $k \times k$ matrices over $\mathbb{F}$, check if P is zero using $\mathsf{poly}(n \cdot k \cdot d)$ many $\mathbb{F}$-operations.*

How hard is this problem? It is quite easy to verify that if we allow randomness then it is solvable just like the usual PIT problem (using Schwartz-Zippel test [Sch80, Zip79]). So we are only interested in deterministic methods in this work.

Conventions - Whenever we say 'arithmetic circuit (or formula)' without an extra qualification, we mean a circuit (or formula) over a field. Otherwise, we explicitly mention 'arithmetic circuit (or formula) over *some* algebra' to mean that the constants of the circuit are taken from 'that' algebra. Also, by depth 3 and depth 2 circuits, we always mean $\Sigma\Pi\Sigma$ and $\Pi\Sigma$ circuits respectively. Further, we take $x_0 = 1$ throughout this paper.

1.1 The depth 2 model of computation

A depth 2 circuit C over matrices naturally defines a computational model. Assuming $\mathcal{R} = \mathsf{M}_k(\mathbb{F})$, 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 matrix P is f. Sometimes we say P *computes* f to mean the same. In the following discussion, we denote the algebra of upper-triangular $k \times k$ matrices by $\mathsf{U}_k(\mathbb{F})$. The algebra $\mathsf{U}_2(\mathbb{F})$ is the *smallest* noncommutative algebra with unity over $\mathbb{F}$, in the sense that $\dim_{\mathbb{F}} \mathsf{U}_2(\mathbb{F}) = 3$ and any algebra of smaller dimension is commutative. We show here that already $\mathsf{U}_2(\mathbb{F})$ captures an open case of identity testing.

[†] A polynomial is said to have low degree if its degree is less than the size of the circuit that computes it.

[‡] In this paper, an algebra is always a finite dimensional associative algebra with unity.

Ben-Or and Cleve [BC88] showed that a polynomial computed by an arithmetic formula E of depth d, and fan-in (of every gate) bounded by 2, can also be computed by a straight-line program of length at most 4^d using only 3 registers. The following fact can be readily derived from their result (see Appendix): From an arithmetic formula E of depth d and fan-in bounded by 2, we can efficiently compute the expression, $P = \prod_{i=1}^{m} \sum_{j=0}^{n} A_{ij} x_j$, where $m \leq 4^d$ and $A_{ij} \in \mathsf{M}_3(\mathbb{F})$ such that P *computes* the polynomial that E does. Thus solving Problem 2 in polynomial time even for 3×3 matrices yields a polynomial time algorithm for PIT of constant depth circuits, in particular depth 4 circuits. There is an alternative way of arguing that the choice of $\mathcal{R}$ as $\mathsf{M}_3(\mathbb{F})$ is *almost* the general case.

Given an arithmetic circuit of size s, computing a low degree polynomial, use the depth-reduction result by Allender, Jiao, Mahajan and Vinay [AJMV98] (see also [VSBR83]) to construct an equivalent bounded fan-in formula of size $s^{O(\log s)}$ and depth $O(\log^2 s)$. From this, obtain a depth 2 circuit over $\mathsf{M}_3(\mathbb{F})$ of size $4^{O(\log^2 s)} = s^{O(\log s)}$ (using Ben-Or and Cleve's result) that computes the same polynomial as the formula. Thus, derandomization of PIT for depth 2 circuits over 3×3 matrices yields a quasi-polynomial time PIT algorithm for any circuit computing a low degree polynomial. This means, in essence a depth 2 circuit over $\mathsf{M}_3(\mathbb{F})$ plays the role of a depth 4 circuit over $\mathbb{F}$ (in the spirit of Agrawal and Vinay's result).

It is natural to ask how the complexity of PIT for depth 2 circuits over $\mathsf{M}_2(\mathbb{F})$ relates to PIT for arithmetic circuits. In this paper, we provide an answer to this. We show a surprising connection between PIT of depth 2 circuits over $\mathsf{U}_2(\mathbb{F})$ and PIT of depth 3 circuits. The reason this is 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, $x_1x_2 + x_3x_4 + x_5x_6$!

Known related models

Identity testing and circuit lower bounds have been studied for different algebraic models. Nisan [Nis91] showed an exponential lower bound on the size of any arithmetic formula computing the determinant of a matrix in the non-commutative *free algebra* model. The result was generalized by Chien and Sinclair [CS04] to a large class of non-commutative algebras satisfying polynomial identities, called PI-algebras. Identity testing has also been studied for the non-commutative model by Raz and Shpilka [RS04], Bogdanov and Wee [BW05], and Arvind, Mukhopadhyay and Srinivasan [AMS08]. But unlike those models where the variables do not commute, in our setting the variables always commute but the *constant coefficients* are taken from an algebra $\mathcal{R}$. The motivation for studying this latter model (besides it being a natural generalization of circuits over fields) is that it provides a different perspective to the complexity of the classical PIT problem in terms of the dimension of the underlying algebra. It seems to 'pack' the combinatorial nature of the circuit into a larger base algebra and hence opens up the possibility of using algebra structure results. The simplest nontrivial circuit in this model is a $\Pi\Sigma$ circuit over the non-commutative algebra $\mathcal{R} = \mathsf{U}_2(\mathbb{F})$, and even this, as we show, represents the frontier of our understanding.

1.2 Our Results

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

Identity testing

We show the following result.

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

The above theorem 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 [RS04]. But no interesting result is known for identity testing of even width-2 commutative ABP's. The following result justifies this.

COROLLARY 4. *Identity testing of depth* 3 *circuits* $(\Sigma\Pi\Sigma)$ *reduces to that of width-*2 *ABPs.*

We mentioned before the prospect of using algebra structure results to solve PIT for depth 2 circuits over algebras. Our next result shows this idea at work for commutative algebras.

THEOREM 5. *Given an expression* $P = \prod_{i=1}^{d} \sum_{j=0}^{n} A_{ij}x_j$, *where* $A_{ij} \in \mathcal{R}$, *a commutative algebra of dimension* k *over* $\mathbb{F}$, *there is a deterministic algorithm to test if* P *is zero running in time* $\mathsf{poly}\left(k^k, n, d\right)$.

The above result gives a polynomial time algorithm for $k = O\left(\log s / \log\log s\right)$ where $s = O(nd)$. This result establishes that the power of depth 2 circuits over *small* algebras is primarily derived from the non-commutative nature of the algebra. However, we show that commutative algebras of polynomial dimension over $\mathbb{F}$ are much more powerful.

THEOREM 6. *Identity testing of a depth* 3 *circuit reduces to identity testing of a depth* 2 *circuit* C *over a commutative algebra of dimension polynomial in the size of* C.

Our argument for proving Theorem 3 is relatively simple in nature. Perhaps the reason why such a connection was overlooked before is that, unlike a depth 2 circuit over $\mathsf{M}_3(\mathbb{F})$, we do not 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 second part of our results.

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

We show that depth 2 circuits over $\mathsf{U}_2(\mathbb{F})$ are computationally weaker than depth 3 circuits.

THEOREM 7. *Let* $f \in \mathbb{F}[x_1, \ldots, x_n]$ *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 $(\Pi\Sigma)$ *circuit over* $\mathsf{U}_2(\mathbb{F})$.

Even a simple polynomial like $x_1x_2 + x_3x_4 + x_5x_6$ satisfies the condition stated in the above theorem, and so it is not computable by any depth 2 circuit over $\mathsf{U}_2(\mathbb{F})$, no matter how large! This contrast makes Theorem 3 surprising as it establishes an equivalence of identity testing in two models of different computational strengths. We further show that the computational power of depth 2 circuits over $\mathsf{M}_2(\mathbb{F})$ is also severely restrictive. Let P_ℓ denote the partial product $P_\ell = \prod_{i=\ell}^{d} \sum_{j=0}^{n} A_{ij}x_j$, where $A_{ij} \in \mathsf{M}_2(\mathbb{F})$ and $1 \leq \ell \leq d$.

DEFINITION 8. *A polynomial f is computed by a depth* 2 *circuit* $(\Pi\Sigma)$ under a degree restriction *of m if the degree of every partial product P_ℓ is bounded by m, for $1 \leq \ell \leq d$.*

THEOREM 9. *There exists a class of polynomials over $\mathbb{F}$ of degree n that cannot be computed by a depth* 2 $(\Pi\Sigma)$ *circuit over $\mathsf{M}_2(\mathbb{F})$, under a degree restriction of n.*

The motivation behind imposing a condition like *degree restriction* comes naturally from depth 2 circuits over $\mathsf{M}_3(\mathbb{F})$. 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 $\mathsf{M}_3(\mathbb{F})$ that literally forms these monomials and adds them up one by one. This computation is degree restricted, if we extend our definition of degree restriction to $\mathsf{M}_3(\mathbb{F})$. However, the above theorem shows that this simple scheme fails over $\mathsf{M}_2(\mathbb{F})$.

2 Identity testing over $\mathsf{M}_2(\mathbb{F})$

We show that PIT of depth 2 circuits over $\mathsf{M}_2(\mathbb{F})$ is at least as hard as PIT of depth 3 circuits. This implies that PIT of a width-2 commutative ABP is also 'harder' than the latter problem.

2.1 Equivalence with depth 3 identity testing

Given a depth 3 circuit, assume (without loss of generality) that the fan-in of the multiplication gates are the same. This multiplicative fan-in is referred to as the *degree* of the depth 3 circuit. For convenience, we call a matrix with linear functions as entries, a *linear* matrix.

LEMMA 10. *Let f be a polynomial over $\mathbb{F}$ computed by a depth* 3 *circuit C of degree d and top fan-in s. Given C, it is possible to efficiently construct a depth* 2 *circuit over $\mathsf{U}_2(\mathbb{F})$ of size $O(ds^2)$ that computes $L \cdot f$, 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 show that there exists such a sequence of length $O(ds^2)$ 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, $f = \sum_{i=1}^{s} P_i$, where each summand $P_i = \prod_j l_{ij}$ is a product of linear functions. Observe that a single P_i can be computed using the following product sequence of length d.

$$\begin{bmatrix} l_{i1} & \\ & 1 \end{bmatrix} \cdots \begin{bmatrix} l_{i(d-1)} & \\ & 1 \end{bmatrix} \begin{bmatrix} 1 & l_{id} \\ & 1 \end{bmatrix} = \begin{bmatrix} L' & P_i \\ & 1 \end{bmatrix}, \quad \text{where } L' = l_{i1} \cdots l_{i(d-1)}. \tag{1}$$

The proof proceeds through induction, where Equation 1 serves as the induction basis. A generic intermediate matrix looks 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 sum of the P_i's. Inductively double the number of summands in g as follows.

At the i-th iteration, suppose 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 want a sequence that computes a polynomial of the form $L \cdot (g + h)$. Consider the following sequence,

$$\begin{bmatrix} L_1 & L_2 g \\ & L_3 \end{bmatrix} \begin{bmatrix} A & \\ & B \end{bmatrix} \begin{bmatrix} M_1 & M_2 h \\ & M_3 \end{bmatrix} = \begin{bmatrix} AL_1M_1 & AL_1M_2h + BL_2M_3g \\ & BL_3M_3 \end{bmatrix}, \tag{2}$$

where A, B are products of linear functions. By setting $A = L_2M_3$ and $B = L_1M_2$ we have,

$$\begin{bmatrix} L_1 & L_2g \\ & L_3 \end{bmatrix} \begin{bmatrix} A & \\ & B \end{bmatrix} \begin{bmatrix} M_1 & M_2h \\ & M_3 \end{bmatrix} = \begin{bmatrix} L_1L_2M_1M_3 & L_1L_2M_2M_3(g+h) \\ & L_1L_3M_2M_3 \end{bmatrix}.$$

This way, we have doubled the number of summands in $g + h$. By induction, each L_i and M_i is a product of n_i linear functions. Therefore, the matrix $\begin{bmatrix} A & \\ & B \end{bmatrix}$ is a product of at most $2n_i$ diagonal linear matrices and the length of the sequence given in Equation 2 is bounded by $4n_i$. This process of doubling the summands needs to be repeated at most $\log s + 1$ times and so the length of the final product sequence is bounded by $d \cdot 4^{\log s} = ds^2$. ∎

PROOF. [Theorem 3] Given a depth 3 circuit computing f we can construct a depth 2 circuit D over $\mathsf{U}_2(\mathbb{F})$ that computes $L \cdot f$. The output of D can be projected appropriately so that we may assume that D outputs the matrix $\begin{bmatrix} 0 & L \cdot f \\ & 0 \end{bmatrix}$, which is zero if and only if f is zero.

To see the other direction of the equivalence, observe that the off-diagonal entry of the output of any depth 2 circuit D over $\mathsf{U}_2(\mathbb{F})$ is a sum of at most d' products of linear functions, where d' is the multiplicative fan-in of D. ∎

2.2 Width-2 algebraic branching programs

Algebraic Branching Program (ABP) is a model of computation introduced by Nisan [Nis91].

DEFINITION 11. *An ABP is a directed acyclic graph with a source and a sink. The vertices of this graph are partitioned into levels, where edges go from level i to level $i + 1$, with the source at the first level and the sink at the last level. Each edge is labelled with a homogeneous linear function of $x_1, \ldots, x_n$. The* width *of the ABP is the maximum number of vertices at any level. An ABP computes a function by summing over all paths from source to sink, the product of all linear functions by which the edges of the path are labelled.*

PROOF. [Corollary 4] In Theorem 3 we have constructed a depth 2 circuit D that computes $P = \prod_i \sum_j A_{ij}x_j$, where each $A_{ij} \in \mathsf{U}_2(\mathbb{F})$. We can make D homogeneous by introducing an extra variable z, such that $P = \prod_i (A_{i0}z + A_{i1}x_1 + \ldots + A_{in}x_n)$. By making the i^{th} linear matrix in the sequence act as the biadjacency matrix between level i and $i + 1$ of the ABP, we have a width-2 ABP computing the same polynomial. ∎

3 Identity testing over commutative algebras

The main idea behind the proof of Theorem 5 is a structure theorem for finite dimensional commutative algebras involving *local rings*.

DEFINITION 12. *A ring $\mathcal{R}$ is* local *if it has a unique maximal ideal.*

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, as we need an effective version of this theorem, we present an appropriate proof here.

THEOREM 13. *A finite dimensional commutative algebra $\mathcal{R}$ is isomorphic to a direct sum of local rings, i.e. $\mathcal{R} \cong \oplus_{i=1}^{\ell} \mathcal{R}_i$, where $\mathcal{R}_i$ is a local ring 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. Suppose, there is a non-nilpotent non-unit 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 will later show that using z it is possible to find an *idempotent* $v \notin \{0, 1\}$ (i.e. $v^2 = v$) in $\mathcal{R}$. But at first, let us see what happens if we already 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 for any $b \in \mathcal{R}v$ and $c \in \mathcal{R}(1 - v)$, $b \cdot c = 0$, we get $\mathcal{R} \cong \mathcal{R}v \oplus \mathcal{R}(1 - v)$ as a non-trivial decomposition of $\mathcal{R}$. By repeating the splitting process on the sub-algebras we can eventually prove the theorem.

Now we show how to find an idempotent from a zero-divisor z. An element $a \in \mathcal{R}$ can be equivalently expressed as a matrix in $\mathsf{M}_k(\mathbb{F})$, 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 can assume that z is already in Jordan normal form. (We will not 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. Then, $w = z^k = \begin{bmatrix} B & 0 \\ 0 & 0 \end{bmatrix}$, where $B = A^k$ 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 v that splits $\mathcal{R}$. First observe that the minimum polynomial of w 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 $w \cdot (aw - 1) = 0$. Hence $v = aw$ is the identity element of $\mathcal{R}w$ and is also an idempotent in $\mathcal{R}$. ∎

We are now ready to prove Theorem 5.

PROOF. [Theorem 5] Let $\{e_1, \ldots, e_k\}$ be a basis of $\mathcal{R}$ over $\mathbb{F}$. As argued before, any element in $\mathcal{R}$ can be equivalently expressed as a $k \times k$ matrix over $\mathbb{F}$. Hence, assume that $A_{ij} \in \mathsf{M}_k(\mathbb{F})$, for all i and j. Since $\mathcal{R}$ is given in basis form, the matrix representations of A_{ij}'s can be found efficiently. If every A_{ij} is non-singular, then surely $P \neq 0$. So, assume that $\exists A_{ij} = z$ such that z is a zero-divisor i.e. singular. From the proof of Theorem 13 it follows that the sub-algebra $\mathcal{R}w$, where $w = z^k$, contains an identity element v which is an idempotent. The idempotent 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 can be easily computed from the elements $e_1 w, \ldots, e_k w$. Express v as, $v = \sum_{j=1}^{k'} \nu_j b_j$, where $\nu_j \in \mathbb{F}$ are unknowns. Since v is an identity in $\mathcal{R}w$ it satisfies the relation, $\sum_{j=1}^{k'} \nu_j b_j \cdot b_i = b_i$, for $1 \leq i \leq k'$. Expressing each b_i in terms of $e_1, \ldots, e_k$, we get a system of linear equations in the ν_j's. Find v by solving this linear system.

Since $\mathcal{R} \cong \mathcal{R}v \oplus \mathcal{R}(1 - v)$, we can split the identity testing problem into two subproblems. That is, P is zero if and only if, $Pv \in \mathcal{R}v$ and $P(1 - v) \in \mathcal{R}(1 - v)$ are both zero. Now apply the above process, recursively, on Pv and $P(1 - v)$. By decomposing the algebra each time an A_{ij} is a non-nilpotent zero-divisor, we are finally left with the easier problem

of checking if, $P = \prod_{i=1}^{d} \left(\sum_{j=0}^{n} A_{ij} x_j \right)$ is zero, where the coefficients A_{ij}'s are either nilpotent or invertible matrices. It is not hard to see that such a P is zero, if and only if the product of all those terms for which all the coefficients are nilpotent matrices is zero. If the number of such terms is greater than k then P is automatically zero (this follows from the fact that commuting nilpotent matrices can be simultaneously triangularized).

Otherwise, treat each term $\sum_{j=0}^{n} A_{ij} x_j$ as a $k \times k$ linear matrix. Since, there are at most k such linear matrices in P, the total number of linear functions occurring as entries of these linear matrices is bounded by k^3. Using a basis of these linear functions we can reduce the number of effective variables in P to k^3. Now, checking if P is zero takes only $\mathsf{poly}(k^k)$ field operations and hence the overall time complexity is bounded by $\mathsf{poly}(k^k, n, d)$. ∎

Thus, PIT of depth 2 circuits over finite dimensional commutative algebras reduces in polynomial time to that over local rings. If dimensions of these local rings are small we have an efficient algorithm. But what happens for much larger dimensions?

THEOREM 6. *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. Consider a depth 3 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 $1 \le i \le s$. 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 $\mathcal{R}$. Notice that the polynomial, $P = \prod_{j=1}^{d} \sum_{i=1}^{s} l_{ij} y_i = f \cdot y_1^d$ is zero if and only if f is zero. Polynomial P can indeed be computed by a depth 2 circuit over $\mathcal{R}$. ∎

4 Weakness of the depth 2 model

In Lemma 10, we have constructed a depth 2 circuit over $\mathsf{U}_2(\mathbb{F})$ that 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.

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

The ideal of $\mathbb{F}[x_1, \ldots, x_n]$ generated by two linear functions l_1 and l_2 is denoted by (l_1, l_2). We say that l_1 is *independent* of l_2 if $1 \notin (l_1, l_2)$. Let f be a polynomial such that there are no two independent linear functions l_1 and l_2 which make $f \bmod (l_1, l_2)$ also a linear function.

PROOF. [Theorem 7] Assume on the contrary that f can be computed by a depth 2 circuit over $\mathsf{U}_2(\mathbb{F})$. That is, there is a product sequence $M_1 \cdots M_t$ of 2×2 upper-triangular linear matrices such that f is the top-right entry of the product matrix. Let $M_i = \begin{bmatrix} l_{i1} & l_{i2} \\ & l_{i3} \end{bmatrix}$, then

$$f = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} l_{11} & l_{12} \\ & l_{13} \end{bmatrix} \begin{bmatrix} l_{21} & l_{22} \\ & l_{23} \end{bmatrix} \cdots \begin{bmatrix} l_{t1} & l_{t2} \\ & l_{t3} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{3}$$

Case 1**:** Not all the l_{i1}'s are constants. Let k be the smallest such that l_{k1} is not a constant and $l_{i1} = c_i$ for all $i < k$. Let $[B\, L]^T = M_{k+1} \cdots M_t \cdot [0\, 1]^T$ and $[d_i\, D_i] = [1\, 0] \cdot M_1 \cdots M_{i-1}$. Observe that L is just a product of linear functions, and for all $1 \leq i < k$, we have the relations, $d_{i+1} = \prod_{j=1}^{i} c_j$ and $D_{i+1} = d_i l_{i2} + l_{i3} D_i$. Hence, Equation 3 simplifies as,

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

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

$$f = \alpha l_{k2} + \beta l_{k3} D_k \pmod{l_{k1}}. \tag{4}$$

By inducting on k, we argue that the above relation can not be true. 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):

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

The last equation can be rewritten in the form of Equation 4 with the term $\beta l_{k3} D_k$ 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, $f = d_t l_{t2} + l_{t3} D_t$. This relation is again of the form in Equation 4 (without the mod term) and so the same argument can be repeated. ∎

Some explicit examples of functions that cannot be computed are as follows (see Appendix).

COROLLARY 14. *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 functions like the determinant and permanent polynomials.*

4.2 Depth 2 model over $\mathsf{M}_2(\mathbb{F})$

The power of depth 2 circuits is very restrictive even if the underlying algebra is $\mathsf{M}_2(\mathbb{F})$.

DEFINITION 15. *A polynomial f is said to be r-*robust *if f does not belong to any ideal generated by r linear forms. (A homogeneous linear function is called a* linear form*.)*

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, denote the d^{th} homogeneous part of f by $[f]_d$. Recall the definition of *degree restriction* (Definition 8).

THEOREM 16. *A polynomial f of degree n, such that $[f]_n$ is* 5*-robust, cannot be computed by a depth* 2 $(\Pi\Sigma)$ *circuit over* $\mathsf{M}_2(\mathbb{F})$ *under a degree restriction of n.*

We prove this with the help of the following lemma.

LEMMA 17. *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 $[f_1\, f_2]^T = M \cdot [g_1\, g_2]^T$. 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 defer the proof of this lemma to the end of this section.

PROOF. [Theorem 16] Assume that there is such a sequence of matrices computing f. Without loss of generality, let the first matrix in the sequence be a row vector $\bar{v}$ and the last matrix be a column vector $\bar{w}$. Let $f = \bar{v} \cdot M_1 M_2 \cdots M_d \cdot \bar{w}$ be a sequence of minimum length computing f. Using Lemma 17, we repeatedly transform this sequence, replacing the term $M_i M_{i+1}$ by $(M_i A)(A^{-1} M_{i+1})$ for an appropriate invertible column transformation A. To begin, let $\bar{v} = [l_1\, l_2]$ for two linear functions l_1 and l_2, and $[f_1\, f_2]^T = M_1 \cdots M_d \bar{w}$. Then, $[f\, 0]^T = \begin{bmatrix} l_1 & l_2 \\ 0 & 0 \end{bmatrix} \cdot [f_1\, f_2]^T$. Applying Lemma 17, we can assume that $\bar{v} = [1\, h]$ and so $f = f_1 + h f_2$. Also, $h \neq 0$, by the minimality of the sequence. This forces $[f_1]_n = [f]_n$ 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 $[g_1]_n$ and $[g_2]_n$ respectively. We use induction and translate these properties to the vectors $M_i \cdots M_d \bar{w}$, for all $i \geq 2$. So, suppose that the relation, $[f_1\, f_2]^T = M_i \cdot [g_1\, g_2]^T$, holds in general for some i, where $[f_1]_n$ is 4-robust and $[f_2]_n = 0$.

Since $[f_1]_n$ is 4-robust, using Lemma 17 again, we can assume that

$$\begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} 1 & h_2 \\ c_3 & c_4 + h_4 \end{bmatrix} \begin{bmatrix} g_1 \\ g_2 \end{bmatrix} \tag{5}$$

by reusing the symbols g_1, g_2. 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 17) and that would contradict the minimality of the sequence. Therefore $h_4 \neq 0$.

We claim that, in Equation 5, $c_3 = 0$. As $h_4 \neq 0$, the degree restriction forces $[g_2]_n = 0$. And since $[f_2]_n = 0$, 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 4-robustness of $[f_1]_n$ as then $[f_1]_n = [g_1]_n + h_2 [g_2]_{n-1} \in (h_2, h_4)$.

From the relations, $[f_2]_n = 0$, $c_3 = 0$ and $h_4 \neq 0$, it follow that $[g_2]_{n-1} = 0$. Hence, $[g_1]_n = [f_1]_n$ is 4-robust. Thus, we have translated the properties to $[g_1\, g_2]^T$, showing that $[g_1]_n$ is 4-robust and $[g_2]_n = 0$. However, since the sequence is finite, there must come a point when degree of g_1 in $[g_1\, g_2]^T = M_i \cdots M_d \bar{w}$ drops below n for some $i \geq 2$. At this point we get a contradiction. ∎

PROOF. [Lemma 17] Suppose we have the equation,

$$\begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} h_1 + c_1 & h_2 + c_2 \\ h_3 + c_3 & h_4 + c_4 \end{bmatrix} \begin{bmatrix} g_1 \\ g_2 \end{bmatrix} \tag{6}$$

where $c_1, \ldots, c_4$ are constants and $h_1, \ldots, h_4$ are linear forms. On comparing degree $n+1$ terms, we have the relations, $h_1 [g_1]_n + h_2 [g_2]_n = 0$ and $h_3 [g_1]_n + h_4 [g_2]_n = 0$. If h_3 and h_4 (a similar reasoning holds for h_1 and h_2) are not proportional (i.e. not multiple of each other),

then $[g_1]_n, [g_2]_n \in (h_3, h_4)$. But this implies that, $[f_1]_n = h_1[g_1]_{n-1} + h_2[g_2]_{n-1} + c_1[g_1]_n + c_2[g_2]_n \in (h_1, h_2, h_3, h_4)$, contradicting the 4-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 6 simplifies to $\begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} c_1 & h_2 + c_2 \\ c_3 & h_4 + c_4 \end{bmatrix} \begin{bmatrix} g_1 \\ g_2 \end{bmatrix}$, reusing symbols g_1, g_2 and others. If $c_1 = [g_2]_n = 0$ then $[f_1]_n = h_2[g_2]_{n-1}$, contradicting robustness. Therefore, either $c_1 \neq 0$, in which case another column transformation gets the matrix to the form claimed, or $[g_2]_n \neq 0$ implying that $h_2 = h_4 = 0$. But then c_1 and c_2 both cannot be zero, $[f_1]_n$ being 4-robust, and hence a column transformation yields the desired form. ∎

5 Concluding remarks

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 $U_2(\mathbb{F})$. We also give a deterministic polynomial time identity testing algorithm for depth 2 circuits over commutative algebras of small dimension. 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? In fact, we have a specific non-commutative algebra in mind. The question is - Is it possible to use properties very specific to the ring of upper-triangular 2×2 matrices to solve PIT for depth 3 circuits?

Acknowledgement

This work was started when the first author visited Hausdorff Center for Mathematics, Bonn. We thank Marek Karpinski for the generous hospitality and several discussions. We also thank Manindra Agrawal for several insightful discussions on this work. And finally thanks to V Vinay for many useful comments on the first draft of this paper.

References

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

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

[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.

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

[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.

[AMS08] Vikraman Arvind, Partha Mukhopadhyay, and Srikanth Srinivasan. New results on noncommutative and commutative polynomial identity testing. In *IEEE Conference on Computational Complexity*, pages 268–279, 2008.

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

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

[BW05] Andrej Bogdanov and Hoeteck Wee. More on noncommutative polynomial identity testing. In *CCC*, pages 92–99, 2005.

[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.

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

[CS04] Steve Chien and Alistair Sinclair. Algebras with polynomial identities and computing the determinant. In *FOCS*, pages 352–361, 2004.

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

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

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

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

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

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

[RS04] Ran Raz and Amir Shpilka. Deterministic Polynomial Identity Testing in Non-Commutative Models. In *CCC*, pages 215–222, 2004.

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

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

[VSBR83] Leslie G. Valiant, Sven Skyum, S. Berkowitz, and Charles Rackoff. Fast parallel computation of polynomials using few processors. *SIAM J. Comput.*, 12(4):641–644, 1983.

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

A Appendix

Ben-Or and Cleve's result

For the sake of completeness, we provide a proof of the result by Ben-Or and Cleve [BC88].

THEOREM 18. *[BC88] Let E be an arithmetic formula of depth d with fan-in (of every gate) bounded by 2. Then, there exists a sequence of 3×3 matrices, whose entries are either variables or constants, of length at most 4^d such that one of the entries of their product is E.*

PROOF. The proof is by induction on the structure of E. The base case when $E = c \cdot x_i$ is computed as,

$$\begin{bmatrix} 1 & & \\ & 1 & \\ c \cdot x_i & & 1 \end{bmatrix}$$

Suppose $E = f_1 + f_2$ and that we have inductively constructed sequences computing f_1 and f_2. Then the following equation gives a sequence for E.

$$\begin{bmatrix} 1 & & \\ & 1 & \\ f_1 & & 1 \end{bmatrix} \begin{bmatrix} 1 & & \\ & 1 & \\ f_2 & & 1 \end{bmatrix} = \begin{bmatrix} 1 & & \\ & 1 & \\ f_1 + f_2 & & 1 \end{bmatrix}$$

If $E = f_1 \cdot f_2$, then the following sequence computes E

$$\begin{bmatrix} 1 & & \\ -f_2 & 1 & \\ & & 1 \end{bmatrix} \begin{bmatrix} 1 & & \\ & 1 & \\ & f_1 & 1 \end{bmatrix} \begin{bmatrix} 1 & & \\ f_2 & 1 & \\ & & 1 \end{bmatrix} \begin{bmatrix} 1 & & \\ & 1 & \\ & -f_1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & & \\ & 1 & \\ f_1 f_2 & & 1 \end{bmatrix}$$

Applying the above two equations inductively, it is clear that E can be computed by a sequence of length at most 4^d. ∎

Proof of Corollary 14

PROOF. [Corollary 14] It suffices to show that $f = x_1x_2 + x_3x_4 + x_5x_6$ satisfy the requirement in Theorem 7.

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 $f \bmod (l_1, l_2)$ is linear. We can evaluate $f \bmod (l_1, l_2)$ 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 $f \bmod (l_1, l_2)$ cannot be linear.

Otherwise, let $x_1 = l_1'$ and $x_2 = l_2'$ be the substitutions. If $f \bmod (l_1, l_2) = l_1'l_2' + x_3x_4 + x_5x_6$ 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. ∎