Electronic Colloquium on Computational Complexity, Report No. 111 (2018)

Beating Brute Force for Polynomial Identity Testing of General Depth-3 Circuits

V. Arvind* Abhranil Chatterjee† Rajit Datta‡
Partha Mukhopadhyay§

June 4, 2018

Abstract

Let C be a depth-3 $\Sigma\Pi\Sigma$ arithmetic circuit of size s, computing a polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ (where $\mathbb{F} = \mathbb{Q}$ or $\mathbb{C}$) with fan-in of product gates bounded by d. We give a deterministic time $2^d \operatorname{poly}(n, s)$ polynomial identity testing algorithm to check whether $f \equiv 0$ or not.

In the case of finite fields, for $\mathrm{Char}(\mathbb{F}) > d$ we obtain a deterministic algorithm of running time $2^{\gamma \cdot d} \operatorname{poly}(n, s)$, whereas for $\mathrm{Char}(\mathbb{F}) \leq d$, we obtain a deterministic algorithm of running time $2^{(\gamma+2)\cdot d \log d} \operatorname{poly}(n, s)$ where $\gamma \leq 5$.

1 Introduction

Polynomial Identity Testing (PIT) is the following well-studied algorithmic problem: Given an arithmetic circuit C computing a polynomial in $\mathbb{F}[x_1, \ldots, x_n]$, determine whether C computes an identically zero polynomial or not. The problem can be presented either in the *white-box* model or in the *black-box* model. In the white-box model, the arithmetic circuit is given explicitly as the input. In the black-box model, the arithmetic circuit is given black-box access. I.e., the circuit can be evaluated at any point in $\mathbb{F}^n$ (or in F^n, for a suitable extension field F). In the last three decades, PIT has played a pivotal role in many important results in complexity theory and algorithms: Primality Testing [AKS04], the PCP Theorem [ALM⁺98], IP = PSPACE [Sha90], graph matching algorithms [Lov79, MVV87]. The problem PIT has a randomized polynomial-time algorithm (more precisely, a co-RP algorithm) via the Schwartz-Zippel-Lipton-DeMillo Lemma [Sch80, Zip79, DL78], but an efficient deterministic algorithm is known only in some special cases. An important result of Impagliazzo and Kabanets [KI04] (also, see [HS80, Agr05]) shows a

*Institute of Mathematical Sciences (HBNI), Chennai, India, email: arvind@imsc.res.in
†Institute of Mathematical Sciences (HBNI), Chennai, India, email: abhranilc@imsc.res.in
‡Chennai Mathematical Institute, Chennai, India, email: rajit@cmi.ac.in
§Chennai Mathematical Institute, Chennai, India, email: partham@cmi.ac.in

ISSN 1433-8092

connection between the existence of a subexponential time PIT algorithm and arithmetic circuit lower bounds.

We refer the reader to the survey of Shpilka and Yehudayoff [SY10] for the exposition of important results in arithmetic circuit complexity, and the polynomial identity testing problem.

Agrawal and Vinay [AV08] have shown that polynomial size degree-d n-variate arithmetic circuits can be depth-reduced to $\Sigma\Pi\Sigma\Pi$ circuits of $n^{O(\sqrt{d})}$ size. Thus, a nontrivial deterministic PIT algorithm for depth-4 (i.e., $\Sigma\Pi\Sigma\Pi$) circuits would imply a nontrivial deterministic PIT algorithm for general arithmetic circuits. Indeed, for characteristic zero fields, derandomization of PIT even for depth-3 $\Sigma\Pi\Sigma$ circuits would have a similar implication [GKKS13].

Motivated by the results of [KI04, Agr05, AV08], a large body of research has focussed on PIT for restricted classes of depth-3 and depth-4 circuits. In particular, a well-studied subclass of depth-3 arithmetic circuits are $\Sigma\Pi\Sigma(k)$ circuits (where the fan-in of the top + gate is bounded by k). Dvir and Shpilka have shown a *white-box* quasi-polynomial time deterministic PIT algorithm for $\Sigma\Pi\Sigma(k)$ circuits [DS07]. Kayal and Saxena have given a deterministic $\mathrm{poly}(d^k, n, s)$ white-box algorithm for the same problem [KS07]. Following the result of [KS07](also see [AM10] for a different analysis), Karnin and Shpilka have given the first *black-box* quasi-polynomial time algorithm for $\Sigma\Pi\Sigma(k)$ circuits [KS11]. Later, Kayal and Saraf [KS09] have shown a polynomial-time deterministic black-box PIT algorithm for the same class of circuits over $\mathbb{Q}$ or $\mathbb{R}$. Finally, Saxena and Sheshadhri have settled the problem for $\Sigma\Pi\Sigma(k)$ completely by giving a deterministic polynomial-time *black-box* algorithm [SS12] over any field. We also note that Oliveira et al. have recently given a sub-exponential PIT -algorithm for depth-3 and depth-4 *multilinear* formulas [dOSlV16].

Summary of our results.

For general depth-3 $\Sigma\Pi\Sigma$ circuits with $\times$-gate fan-in bounded by d, to the best of our knowledge, no deterministic algorithm with running time better than $\min\{d^n, \binom{n+d}{d}\}\,\mathrm{poly}(n, d)$ is known. Our main results are the following.

Theorem 1. *Let C be a $\Sigma\Pi\Sigma$ circuit of size s, computing a polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ (where $\mathbb{F} = \mathbb{Q}$ or $\mathbb{C}$) and the fan-in of the product gates of C is bounded by d. We give a* white-box *deterministic polynomial time identity testing algorithm to check whether $f \equiv 0$ or not in time $2^d\,\mathrm{poly}(n, s)$.*

Over the fields of positive characteristic, we show the following result.

Theorem 2. *Let C be a $\Sigma\Pi\Sigma$ circuit of size s, computing a polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ and the fan-in of the product gates of C is bounded by d. For $\mathrm{Char}(\mathbb{F}) > d$, we give a* white-box *deterministic polynomial time identity testing algorithm to check whether $f \equiv 0$ or not in time $2^{\gamma \cdot d}\,\mathrm{poly}(n, s)$. The constant γ is at most 5.*

As an immediate corollary we get the following.

Corollary 1. *Let C be a depth-3 $\Sigma\Pi\Sigma$ circuit of size s, computing a polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ and the fan-in of the product gates of C is bounded by $c \log n$) for some constant c (where $\mathbb{F} = \mathbb{Q}$ or $\mathbb{C}$ or a finite field such that $\mathrm{Char}(\mathbb{F}) > c \log n$) . We give a deterministic $\mathrm{poly}(n, s)$ time identity testing algorithm to check whether $f \equiv 0$ or not.*

Over the fields of smaller characteristic, we have the following result.

Theorem 3. *Let C be a $\Sigma\Pi\Sigma$ circuit of size s, computing a polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ and the fan-in of the product gates of C is bounded by d. For $\mathrm{Char}(\mathbb{F}) \leq d$, we give a* white-box *deterministic polynomial time identity testing algorithm to check whether $f \equiv 0$ or not in time $2^{(\gamma+2)\cdot d \log d} \mathrm{poly}(n, s)$. The constant γ is at most 5.*

2 Organization

The paper is organized as follows. Section 3 covers the background materials. In Section 4, we prove Theorem 1 that shows a deterministic $2^d \cdot \mathrm{poly}(n)$ PIT for depth-3 circuits over $\mathbb{Q}$ and $\mathbb{C}$. The PIT algorithms for depth-3 circuits over finite fields are presented in Section 5, where we prove Theorems 2 and 3.

3 Preliminaries

For a monomial m and a polynomial f, let $[m]f$ denote the coefficient of the monomial m in f. We denote the field of rational numbers as $\mathbb{Q}$, and the field of complex numbers as $\mathbb{C}$. Depth-3 $\Sigma^{[s]}\Pi^{[d]}\Sigma$ circuits computing polynomials in $\mathbb{F}[x_1, x_2, \ldots, x_n]$ are of the following form:

$$C(x_1, \ldots, x_n) = \sum_{i=1}^{s} \prod_{j=1}^{d} L_{i,j}(x_1, \ldots, x_n),$$

where each $L_{i,j}$ is an affine linear form over $\mathbb{F}$.

We refer to them as $\Sigma\Pi\Sigma$ circuits for unspecified s and d.

We recall a well-known fact which states that for the purpose of solving PIT , it suffices to consider homogeneous circuits. We use the notation $\Sigma^{[s]}\Pi^{[d]}\Sigma$ to denote homogeneous depth-3 circuits of top sum gate fan-in s, product gates fan-in bounded by d.

Fact 1. *Let $C(x_1, \ldots, x_n)$ be a $\Sigma^{[s]}\Pi^{[d]}\Sigma$ circuit. Then $C \equiv 0$ if and only if $z^d C(x_1/z, \ldots, x_n/z) \equiv 0$ where z is a new variable.*

We say a monomial m is of *type* $\boldsymbol{e} = (e_1, e_2, \ldots, e_q)$ if $m = x_{i_1}^{e_1} x_{i_2}^{e_2} \ldots x_{i_q}^{e_q}$ for $e_1 \leq e_2 \leq \ldots \leq e_q$ and each i_j is distinct. For the monomial $m = x_{i_1}^{e_1} x_{i_2}^{e_2} \ldots x_{i_q}^{e_q}$ we use $m!$ to denote the product $e_1! \cdot e_2! \cdots e_k!$ as a convenient abuse of notation.

Connection to noncommutative computation

In this paper, we will also deal with the free noncommutative ring $\mathbb{F}\langle Y\rangle$, where Y is a set of noncommuting variables. In this ring, monomials are words in Y^* and polynomials in $\mathbb{F}\langle Y\rangle$ are $\mathbb{F}$-linear combinations of words. We define noncommutative arithmetic circuits essentially as their commutative counterparts. The only difference is that at each product gate in a noncommutative circuit there is a prescribed left to right ordering of its inputs.

Given a noncommutative monomial $m = y_{i_1} y_{i_2} \dots y_{i_d}$ of degree d and a permutation $\sigma \in S_d$, we use m^σ to denote the position-permuted monomial $y_{i_{\sigma(1)}} y_{i_{\sigma(2)}} \cdots y_{i_{\sigma(d)}}$.

For our PIT algorithms over finite fields given in Section 5, we will be applying the Raz-Shpilka PIT algorithm [RS05] for noncommutative algebraic branching programs. For this purpose, we prescribe a way of transforming a given commutative circuit C computing a polynomial in $\mathbb{F}[x_1, x_2, \dots, x_n]$ to a noncommutative version C^{nc}. The circuit C^{nc} is defined by fixing an ordering of the inputs to each product gate in C and replacing x_i by the noncommutative variable $y_i, 1 \le i \le n$. Thus, C^{nc} will compute a polynomial f_C^{nc} in the ring $\mathbb{F}\langle Y\rangle$, where $Y = \{y_1, y_2, \dots, y_n\}$ are n noncommuting variables.

Remark 1. *We stress that the above transformation of a commutative circuit C to a noncommutative circuit C^{nc} does not preserve polynomial identities. However, given a commutative $\Sigma\Pi\Sigma$ circuit C, we will suitably "symmetrize" it to obtain $\hat{C}$ ensuring that the noncommutative version $\hat{C}^{nc}$ is identically zero iff $C \equiv 0$.*

We recall the definition of Hadamard Product of two polynomials. The concept of Hadamard product is particularly useful in noncommutative computations [AJ09, AS18].

Definition 1. *Given two degree d polynomials $f, g \in \mathbb{F}[x_1, x_2, \dots, x_n]$, the Hadamard Product $f \circ g$ is defined as*

$$f \circ g = \sum_m ([m]f \cdot [m]g)\ m.$$

For the PIT purpose in the commutative setting, we adapt the notion of Hadamard Product suitably and define a scaled version of Hadamard Product of two polynomials.

Definition 2. *Given two degree d polynomials $f, g \in \mathbb{F}[x_1, x_2, \dots, x_n]$, the scaled version of the Hadamard Product $f \circ^s g$ is defined as*

$$f \circ^s g = \sum_m (m! \cdot [m]f \cdot [m]g)\ m,$$

where $m = x_{i_1}^{e_1} x_{i_2}^{e_2} \dots x_{i_r}^{e_r}$ for some $r \le d$ and $m! = e_1! \cdot e_2! \cdots e_r!$, as already defined.

For solving PIT over $\mathbb{Q}$, it suffices to compute $f \circ^s f(1, 1, \dots, 1)$. This is because all monomials in $f \circ^s f$ have nonnegative coefficients. Thus, $f \circ^s$

$f(1,1,\ldots,1) \neq 0$ if and only if $f \not\equiv 0$. In the case $\mathbb{F} = \mathbb{C}$, it suffices to compute $f \circ^s \bar{f}(1,1,\ldots,1)$ where $\bar{f}$ denotes the polynomial obtained by conjugating every coefficient of f.

We also recall a result of Ryser [Rys63] that gives a $\Sigma^{[2^n]}\Pi^{[n]}\Sigma$ circuit for the Permanent polynomial of $n \times n$ symbolic matrix.

Lemma 1 (Ryser [Rys63]). *For a matrix X with variables $x_{ij} : 1 \leq i,j \leq n$ as entries,*

$$\mathrm{Perm}(X) = (-1)^n \sum_{S \subseteq [n]} (-1)^{|S|} \prod_{i=1}^{n} \left(\sum_{j \in S} x_{ij} \right).$$

Remark 2. *We note here that Ryser's formula holds over all fields $\mathbb{F}$. Furthermore, if X is a matrix of* free noncommuting *variables $y_{ij} : 1 \leq i,j \leq n$ as entries, then too Ryser's formula holds. More precisely, we have*

$$\mathrm{Perm}(X) = (-1)^n \sum_{S \subseteq [n]} (-1)^{|S|} \prod_{i=1}^{n} \left(\sum_{j \in S} y_{ij} \right),$$

where the order of linear forms in each product gate is increasing order of index i.

The following simple lemma about the coefficient of a monomial in a product of homogeneous linear forms is important for the paper.

Lemma 2. *For a degree-d monomial $m = x_{i_1}x_{i_2}\cdots x_{i_d}$ (where the variables can have repeated occurrences) and a homogeneous $\Pi\Sigma$ circuit $C = \prod_{j=1}^{d} L_j$, the coefficient of monomial m in C is given by:*

$$[m]C = \frac{1}{m!} \sum_{\sigma \in S_d} \prod_{j=1}^{d} ([x_{i_j}]L_{\sigma(j)}).$$

Proof. We assume without loss of generality that the monomial $m = x_{i_1}x_{i_2}\cdots x_{i_d}$ is such that repeated variables are adjacent, where the first e_1 variables are x_{j_1}, and the next e_2 variables are x_{j_2} and so on until the last e_q variables are x_{j_q}, and the $x_{j_k}, 1 \leq k \leq q$ are distinct variables.

We notice that the monomial m can be generated C by first fixing an order $\sigma : [d] \mapsto [d]$ for multiplying the d linear forms as $L_{\sigma(1)}L_{\sigma(2)}\cdots L_{\sigma(d)}$, and then multiplying the coefficients of variable $x_{i_k}, 1 \leq k \leq d$ picked successively from linear forms $L_{\sigma(k)}, 1 \leq k \leq d$. However, these $d!$ orderings repeatedly count terms.

We claim that each distinct product of coefficients term is counted exactly $m!$ times. Let $E_k \subseteq [d]$ denote the interval $E_k = \{j \mid e_{k-1}+1 \leq j \leq e_k\}, 1 \leq k \leq q$, where we set $e_0 = 0$.

Now, to see the claim we only need to note that two permutations $\sigma, \tau \in S_d$ give rise to the same product of coefficients term iff $\sigma(E_k) = \tau(E_k), 1 \leq k \leq q$. Thus, the number of permutations τ that generate the same term as σ is $m!$.

Therefore the actual coefficient $[m]C$, which is the sum of distinct product of coefficients is given by $\frac{1}{m!}\sum_{\sigma \in S_d}\prod_{j=1}^{d}([x_{i_j}]L_{\sigma(j)})$, which completes the proof. □

Perfect Hash Functions

We recall the notion of perfect hash functions from [NSS95, AG10]. An (n,k)-family of perfect hash functions is a collection of functions $\mathcal{F}$ from $[n]$ to $[k]$ such that for every subset $S \subseteq [n]$ of size k, there exists at least one function $f \in \mathcal{F}$ such that f is one-one on S. Explicit deterministic construction of (n,k)-family of perfect hash function is well-known [NSS95, AG10]. For the best known construction, the size of the family is $e^k k^{O(\log k)} \log n$, and the running time of the construction is $O(e^k k^{O(\log k)} \log n)$.

4 PIT for $\Sigma\Pi\Sigma$ circuits over $\mathbb{Q}$ and $\mathbb{C}$

We first outline the main ideas of the PIT algorithm over $\mathbb{Q}$. For two polynomials f and g of degree d, consider their Hadamard product $f \circ g = \sum_m [m]f \cdot [m]g \cdot m$. Clearly, since $f \circ f$ has nonnegative coefficients, $f \equiv 0$ if and only if $f \circ f(1,\ldots,1) = 0$. Thus, given a circuit computing a polynomial f, if we can compute a circuit for $f \circ f$ then we can check if $f \equiv 0$. Actually, we will use a slightly different product which we call the *scaled* Hadamard product defined as

$$f \circ^s g = \sum_m m! \cdot [m]f \cdot [m]g \cdot m.$$

Notice that computing a circuit for $f \circ^s f$ also suffices to solve the PIT problem. Clearly, $f \equiv 0$ if and only if $f \circ^s f(1\ldots,1) = 0$.

As already observed, we can assume w.l.o.g. that the given circuit is homogeneous. Given a $\Sigma^{[s]}\Pi^{[d]}\Sigma$ circuit computing a homogeneous polynomial f, our aim is to compute a circuit for $f \circ^s f$ efficiently. Since the scaled Hadamard product distributes over addition, it suffices to compute the scaled Hadamard product of two $\Pi^{[d]}\Sigma$ circuits C_1 and C_2. We will obtain a $\Sigma^{[2^d]}\Pi^{[d]}\Sigma$ circuit of size $2^d \operatorname{poly}(s,n,d)$ for $C_1 \circ^s C_2$. Surprisingly, we can use Ryser's $2^d \operatorname{poly}(d)$ sized depth-3 formula for the permanent of a $d \times d$ matrix to obtain a depth-3 circuit for $C_1 \circ^s C_2$.

For the $\mathbb{F} = \mathbb{C}$ case a modification of the above method works. Given a circuit C computing $f \in \mathbb{C}[x_1, x_2, \ldots, x_n]$, we first construct a circuit $\bar{C}$ computing $\bar{f} \in \mathbb{C}[x_1, x_2, \ldots, x_n]$, obtained by conjugating coefficients of the linear forms in C. The coefficients $C \circ^s \bar{C}$ are squares of the absolute values of the coefficients of f. Hence, evaluating $C \circ^s \bar{C}$ at $(1,1,\ldots,1)$ yields the desired PIT .

Now we are ready to prove Theorem 1.

Proof of Theorem 1. We present the proof only for $\mathbb{F} = \mathbb{Q}$. For $\mathbb{C}$, we only need a minor modification as explained in Remark 3. Given the circuit C we compute $C \circ^s C$ and evaluate at $(1,1,\ldots,1)$ point. Notice that over rationals, $C \circ^s C$ has non-negative coefficients. This also implies that $C \equiv 0$ if and only if $C \circ^s C(1,1,\ldots,1) = 0$. So it is sufficient to show that $C \circ^s C(1,\ldots,1)$ can be computed deterministically in time $2^d \operatorname{poly}(s,n)$. Since the scaled Hadamard Product distributes over addition, we only need to show that the scaled Hadamard Product of two $\Pi\Sigma$ circuits can be computed efficiently.

Lemma 3. *Given two homogeneous* $\Pi^{[d]}\Sigma$ *circuits* $C_1 = \prod_{i=1}^d L_i$ *and* $C_2 = \prod_{i=1}^d L'_i$ *we have:*

$$C_1 \circ^s C_2 = \sum_{\sigma \in S_d} \prod_{i=1}^d (L_i \circ^s L'_{\sigma(i)}).$$

Proof. We prove the formula monomial by monomial. Let $m = x_{i_1} x_{i_2} \dots x_{i_d}$ be a monomial in C_1 (Note that $i_1, i_2, \dots, i_d$ need not be distinct).

Now let m be a monomial that appears in both C_1 and C_2. From Lemma 2 the coefficients are

$$[m]C_1 = \alpha_1 = \frac{1}{m!} \left(\sum_{\sigma \in S_d} \prod_{j=1}^d [x_{i_j}] L_{\sigma(j)} \right)$$

and

$$[m]C_2 = \alpha_2 = \frac{1}{m!} \left(\sum_{\pi \in S_d} \prod_{j=1}^d [x_{i_j}] L'_{\pi(j)} \right)$$

respectively.

From the definition 2 we have

$$[m](C_1 \circ^s C_2) = m! \cdot \alpha_1 \cdot \alpha_2.$$

Now let us consider the matrix T where $T_{ij} = L_i \circ^s L'_j : 1 \le i, j \le d$ and $\mathrm{Perm}(T) = \sum_{\sigma \in S_d} \prod_{i=1}^d L_i \circ^s L'_{\sigma(i)}$. The coefficient of m in $\mathrm{Perm}(T)$ is

$$[m]\,\mathrm{Perm}(T) = \sum_{\sigma \in S_d} [m] \left(\prod_{j=1}^d L_j \circ^s L'_{\sigma(j)} \right).$$

Similar to Lemma 2, we notice the following.

$$\begin{aligned}
[m]\,\mathrm{Perm}(T) &= \sum_{\sigma \in S_d} \frac{1}{m!} \sum_{\pi \in S_d} \prod_{j=1}^d [x_{i_j}] (L_{\pi(j)} \circ^s L'_{\sigma(\pi(j))}) \\
&= \frac{1}{m!} \sum_{\sigma \in S_d} \sum_{\pi \in S_d} \prod_{j=1}^d ([x_{i_j}] L_{\pi(j)}) \cdot ([x_{i_j}] L'_{\sigma(\pi(j))}) \\
&= \frac{1}{m!} \sum_{\sigma \in S_d} \sum_{\pi \in S_d} \prod_{j=1}^d ([x_{i_j}] L_{\pi(j)}) \cdot \prod_{j=1}^d ([x_{i_j}] L'_{\sigma(\pi(j))}) \\
&= \sum_{\pi \in S_d} \left(\prod_{j=1}^d ([x_{i_j}] L_{\pi(j)}) \cdot \frac{1}{m!} \sum_{\sigma \in S_d} \prod_{j=1}^d ([x_{i_j}] L'_{\sigma(\pi(j))}) \right) \\
&= m! \cdot \frac{1}{m!} \sum_{\pi \in S_d} \left(\prod_{j=1}^d ([x_{i_j}] L_{\pi(j)}) \cdot \frac{1}{m!} \sum_{\sigma \in S_d} \prod_{j=1}^d ([x_{i_j}] L'_{\sigma(\pi(j))}) \right).
\end{aligned}$$

Clearly, for any fixed $\pi \in S_d$, we have that $\sum_{\sigma \in S_d} \prod_{j=1}^d [x_{i_j}] L'_{\sigma(\pi(j))} = m!\alpha_2$. Hence, $[m]\,\mathrm{Perm}(T) = m! \cdot \alpha_1 \cdot \alpha_2$ and the lemma follows.

□

Lemma 4. *Given two $\Pi^{[d]}\Sigma$ circuits C_1 and C_2 we can compute a $\Sigma^{[2^d]}\Pi^{[d]}\Sigma$ for $C_1 \circ^s C_2$ in time $2^d \operatorname{poly}(n, d)$.*

Proof. From Lemma 3 we observe that $\operatorname{Perm}(T)$ gives a circuit for $C_1 \circ^s C_2$. A $\Sigma^{[2^d]}\Pi^{[d]}\Sigma$ circuit for $\operatorname{Perm}(T)$ can be computed in $2^d \operatorname{poly}(n, d)$ time using Lemma 1.

□

Now we show how to take the scaled Hadamard Product of two $\Sigma\Pi\Sigma$ circuits.

Lemma 5. *Given two $\Sigma\Pi^{[d]}\Sigma$ circuits $C = \sum_{i=1}^{s} P_i$ and $\widetilde{C} = \sum_{i=1}^{\tilde{s}} \widetilde{P}_i$ We can compute a $\Sigma^{[2^d s\tilde{s}]}\Pi^{[d]}\Sigma$ circuit for $C \circ^s \widetilde{C}$ in time $2^d \operatorname{poly}(s, \tilde{s}, d, n)$.*

Proof. We first note that by distributivity,

$$C \circ^s \widetilde{C} = \sum_{i=1}^{s} \sum_{j=1}^{\tilde{s}} P_i \circ^s \widetilde{P_j}.$$

Using Lemma 4 for each pair $P_i \circ^s \widetilde{P_j}$ we get a $\Sigma^{[2^d]}\Pi^{[d]}\Sigma$ circuit P_{ij}. Now the formula $\sum_{i=1}^{s} \sum_{j=1}^{\tilde{s}} P_{ij}$ is a $\Sigma^{[2^d s\tilde{s}]}\Pi^{[d]}\Sigma$ formula which can be computed in $2^d \operatorname{poly}(s, \tilde{s}, d, n)$ time.

□

Now given a $\Sigma^{[s]}\Pi^{[d]}\Sigma$ circuit C we can compute $C \circ^s C$ using Lemma 5 and finally evaluating $C \circ^s C(1, 1, \ldots, 1)$ completes the PIT algorithm. Clearly all the computation can be done in $2^d \operatorname{poly}(s, n)$ time. This completes the proof of Theorem 1.

Remark 3. *To adapt the algorithm over $\mathbb{C}$, we need to just compute $C \circ^s \bar{C}$ where $\bar{C}$ is the polynomial obtained from C by conjugating each coefficient. Note that a circuit computing $\bar{C}$ can be obtained from C by just conjugating the scalars that appear in the linear forms of C. This follows from the fact that the conjugation operation distributes over addition and multiplication. Now we have $[m](C \circ^s \bar{C}) = |[m]C|^2$, so the coefficients are all positive and thus evaluating $C \circ^s \bar{C}(1, 1, \ldots, 1)$ is sufficient for the PIT algorithm.*

5 PIT for $\Sigma\Pi\Sigma$ circuits over finite fields

In this section we present the PIT algorithms for $\Sigma\Pi\Sigma$ circuits over finite fields in two subsections: the $\operatorname{Char}(\mathbb{F}) > d$ case and the $\operatorname{Char}(\mathbb{F}) \leq d$ case respectively, where d is the formal degree of the given $\Sigma\Pi\Sigma$ circuit.

5.1 Over large characteristic

We first outline the algorithm for fields $\mathbb{F}$ such that $\operatorname{Char}(\mathbb{F}) > d$, where d is the formal degree of the given $\Sigma\Pi\Sigma$ circuit. Since $\operatorname{Char}(\mathbb{F}) = p > d$, it turns out that the notion of scaled Hadamard product is still useful for us, as $m! \neq 0 \pmod{p}$ in $\mathbb{F}$. However, we cannot simply evaluate the circuit at some specific point to perform the PIT since the final sum could be zero (for instance, a multiple of p).

At this point, we will apply ideas from noncommutative computation.

Suppose the PIT instance is a homogeneous degree-d polynomial $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ given by circuit C. As explained in Section 3, we can consider the corresponding noncommutative circuit C^{nc} which computes a noncommutative homogeneous degree-d polynomial $f' \in \mathbb{F}\langle y_1, y_2, \ldots, y_n\rangle$.

Every monomial m of f can appear as different noncommutative monomials m' in f'. We use the notation $m' \to m$ to denote that $m' \in Y^*$ will be transformed to m by substituting x_i for $y_i, 1 \le i \le n$. Then, we observe that

$$[m]f = \sum_{m' \to m} [m']f'. \tag{1}$$

Clearly, the noncommutative circuit C^{nc} is not directly useful for PIT, because C^{nc} may compute a nonzero polynomial even when $C \equiv 0$. However, we observe that the following symmetrization trick will preserve identity. We first explain how permutations $\sigma \in S_d$ act on the set of degree-d monomials Y^d (and hence, by linearity, act on homogeneous degree-d polynomials).

For each monomial $m' = y_{i_1} y_{i_2} \cdots y_{i_d}$, the permutation $\sigma \in S_d$ maps m' to the monomial m'^{σ} which is defined as $m'^{\sigma} = y_{i_{\sigma(1)}} y_{i_{\sigma(2)}} \cdots y_{i_{\sigma(d)}}$. Consequently, by linearity, $f' = \sum_{m' \in Y^d} [m']f' \cdot m'$ is mapped by σ to the polynomial $f'^{\sigma} = \sum_{m' \in Y^d} [m']f' \cdot m'^{\sigma}$.

The following proposition tells us a simple way of transforming PIT for commutative circuits to PIT for noncommutative circuits.

Proposition 1. *Suppose* $\mathrm{Char}(\mathbb{F}) > d$. *For a homogeneous degree d polynomial $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ given by circuit C, and the corresponding noncommutative circuit C^{nc} computing $f' \in \mathbb{F}\langle y_1, y_2, \ldots, y_n\rangle$ consider the "symmetrized" polynomial*

$$f^* = \sum_{\sigma \in S_d} f'^{\sigma}.$$

Then the commutative polynomial f is identically zero iff the noncommutative polynomial $f^* \in \mathbb{F}\langle y_1, y_2, \ldots, y_n\rangle$ is identically zero.

Proof. Let $f = \sum_m [m]f \cdot m$ and $f' = \sum_{m'} [m']f' \cdot m'$. Notice that

$$[m]f = \sum_{m' \to m} [m']f'.$$

Now, we write

$$f^* = \sum_{m''} [m'']f^* \cdot m''.$$

The group S_d acts on Y^d (degree d monomials in Y) by permuting the coordinates. Suppose $m = x_{i_1}^{e_1} \cdots x_{i_q}^{e_q}$ is a type $e = (e_1, \ldots, e_q)$ degree-d monomial over X and $m'' \to m$. Then, by the Orbit-Stabilizer lemma the orbit $O_{m''}$ of m'' has size $\frac{d!}{m!}$. It follows that $[m'']f^* = \sum_{m' \in O_{m''}} m! \cdot [m']f' = m! \cdot [m]f$. Thus, $[m'']f^* = 0$ if and only if $[m]f = 0$, which proves the proposition. ☐

Thus, in order to check if the polynomial f computed by a commutative circuit C is identically zero, we can instead check if the noncommutative polynomial $f^* \equiv 0$. Clearly, if we have a small algebraic branching program (ABP) for f^*, we can use the deterministic identity testing algorithm of Raz and Shpilka [RS05] to do PIT for f^* and hence for f. We manage to do precisely this in the next result. Now we are ready to prove Theorem 2.

Proof of Theorem 2. We can write $f = \sum_{i=1}^{s} \prod_{j=1}^{d} L_{ij}$, for homogeneous linear forms L_{ij}. Now, the corresponding noncommutative polynomial f' is defined by the natural order of the j indices.

We claim that the noncommutative polynomial f^* defined in Proposition 1 has a noncommutative $\Sigma^{[2^d \cdot s]}\Pi^{[d]}\Sigma$ formula. Once we prove the claim we are done, because we can apply the Raz-Shpilka deterministic PIT algorithm to this formula and obtain the desired PIT, as a consequence of Proposition 1.

Now, consider one of the $\Pi\Sigma$ subcircuits of C, say, $P_i = L_{i1}L_{i2}\cdots L_{id}$. Then $P_i' = L_{i1}'L_{i2}'\cdots L_{id}'$, where L_{ij}' is obtained from L_{ij} by replacing variables x_k with the noncommutative variable y_k for each k. Now, we claim the following.

Claim 1.

$$P_i^* = \sum_{\sigma \in S_d} L'_{i\sigma(1)} L'_{i\sigma(2)} \cdots L'_{i\sigma(d)}.$$

Proof. Let us proof the claim monomial by monomial. Fix a monomial m'' in P_i^* such that $m'' \to m$. Suppose $m'' = y_{k_1} y_{k_2} \ldots y_{k_d}$. Note that, $m = x_{k_1} x_{k_2} \ldots x_{k_d}$. Recall from Proposition 1, $[m'']P_i^* = m! \cdot [m]P_i$. Now, the coefficient of m'' in $\sum_{\sigma \in S_d} \prod_{j=1}^{d} L'_{i\sigma(j)}$ is

$$[m'']\left(\sum_{\sigma \in S_d} \prod_{j=1}^{d} L'_{i\sigma(j)}\right) = \sum_{\sigma \in S_d} \prod_{j=1}^{d} [y_{k_j}] L'_{i\sigma(j)}.$$

Let us notice that, $[y_{k_j}]L'_{i\sigma(j)} = [x_{k_j}]L_{i\sigma(j)}$. Hence,

$$[m'']\left(\sum_{\sigma \in S_d} \prod_{j=1}^{d} L'_{i\sigma(j)}\right) = \sum_{\sigma \in S_d} \prod_{j=1}^{d} [x_{k_j}] L_{i\sigma(j)}.$$

Now, the claim directly follows from Lemma 2 as $\sum_{\sigma \in S_d} \prod_{j=1}^{d} [x_{k_j}] L_{i\sigma(j)} = m! \cdot [m]P_i$. □

Now define the $d \times d$ matrix T_i such that each row of T_i is just the linear forms $L'_{i1}, L'_{i2}, \ldots, L'_{id}$ appearing in P_i. The (noncommutative) permanent of T_i is given by

$$\text{Perm}(T_i) = \sum_{\sigma \in S_d} \prod_{j=1}^{d} L'_{i\sigma(j)},$$

which is just P_i^*.

We now apply Ryser's formula given by Lemma 1 (noting the fact that it holds for the noncommutative permanent too), to express $\text{Perm}(T_i)$ as a depth-3 homogeneous noncommutative $\Sigma^{[2^d]}\Pi^{[d]}\Sigma$ formula. It follows that $f^* = \sum_{i=1}^{s} \mathrm{Perm}(T_i)$ has a $\Sigma^{[2^d \cdot s]}\Pi^{[d]}\Sigma$ noncommutative formula.

Now we apply the identity testing algorithm of Raz and Shpilka for noncommutative ABPs to this $\Sigma^{[2^d \cdot s]}\Pi^{[d]}\Sigma$ noncommutative formula to get the desired result [RS05]. The bound on γ comes from Theorem 4 of their paper [RS05]. This completes the proof of Theorem 2.

Notice that, the statement of Claim 1 does not hold for an arbitrary polynomial over finite fields $\mathbb{F}$ where $\mathrm{Char}(\mathbb{F}) = p \leq d$. To be more precise, for a given homogeneous degree d polynomial f over $\mathbb{F}_p$, if f has a monomial m of form $x_{i_1}^{e_1} x_{i_2}^{e_2} \ldots x_{i_q}^{e_q}$ where $e_i \geq p$ for some $i \in [q]$ then $m! = 0 \pmod p$ and for each m'' such that $m'' \to m$, $[m'']f^* = 0$. Hence, this strategy of applying the noncommutative identity testing algorithm of Raz and Shpilka [RS05] to conclude the identity of f fails in small characteristics.

Remark 4. *If the given $\Sigma\Pi^{[d]}\Sigma$ circuit computes a multilinear polynomial then $m! = 1$ for every monomial and Theorem 2 works for fields of small characteristic also.*

5.2 Over small characteristic

In this section we extend the PIT results over finite fields $\mathbb{F}$ of small characteristic such that $\mathrm{Char}(\mathbb{F}) \leq d$ where d is the formal degree of the given circuit.

Over finite fields $\mathbb{F}$ of small characteristic such that $\mathrm{Char}(\mathbb{F}) = p \leq d$ where d is the formal degree of the given $\Sigma\Pi\Sigma$ circuit, the previous strategy of applying the noncommutative identity testing algorithm of Raz and Shpilka [RS05] to conclude the identity of f fails in general.

Inspired by Remark 4 we reduce the problem of identity testing of general $\Sigma\Pi\Sigma$ circuit over $\mathbb{F}_p$ (which is given as an input) to many instances of PIT of multilinear $\Sigma\Pi\Sigma$ circuits and invoke the algorithm of Theorem 2 to solve the problem over the fields of small characteristic. To do this, we partition the monomials by their *types*. Let f be a polynomial and $\boldsymbol{e}$ be fixed type, we define $f_{\boldsymbol{e}}$ as the restriction of f on the monomials of that type. Clearly that reduces the PIT problem of general depth-3 circuits to identity testing of each $f_{\boldsymbol{e}}$. To do PIT on $f_{\boldsymbol{e}}$, we first construct a $\Sigma\Pi\Sigma\wedge$ circuit that computes $f_{\boldsymbol{e}}$ with some spurious terms. Then we encode the circuit to a $\Sigma\Pi\Sigma$ circuit computing a multilinear polynomial and use Hadamard product and *perfect hash families* to get multilinear circuits each covering some parts of $f_{\boldsymbol{e}}$. By the exhaustiveness property of perfect hash families, we ensure that if $f_{\boldsymbol{e}}$ has nonzero monomial one of the multilinear circuits detects it.

Before going into the details let us first introduce the notion of *type of a monomial.*

Definition 3. *Let $m = x_{i_1}^{e_1} x_{i_2}^{e_2} \ldots x_{i_q}^{e_q}$ be a monomial of total degree d over the variables $x_1, \ldots, x_n$ where $e_1 \leq e_2 \leq \ldots \leq e_q$ and each i_j is distinct. Then the type of m is the q-tuple $\boldsymbol{e} = (e_1, e_2, \ldots, e_q)$.*

The notion of types is helpful in the following sense. Let X_d be the set of all monomials of degree d over $\{x_1, \ldots, x_n\}$. Define $X_{d,\boldsymbol{e}}$ as the set of monomials of type $\boldsymbol{e}$ in X_d. For a homogeneous degree d polynomial f, $f_{\boldsymbol{e}}$ is defined as $f_{\boldsymbol{e}} = \sum_{m \in X_{d,\boldsymbol{e}}} [m]f \cdot m$. Moreover, if we define T as the set of all types for degree d monomials then

$$X_d = \overset{+}{\bigcup_{\boldsymbol{e} \in T}} X_{d,\boldsymbol{e}},$$

i.e. X_d is the disjoint union of each $X_{d,\boldsymbol{e}}$. Therefore, $f = \sum_{\boldsymbol{e} \in T} f_{\boldsymbol{e}}$. We make the following important observation.

Observation 1. *$f \equiv 0$ if and only if $f_{\boldsymbol{e}} \equiv 0$ for each $\boldsymbol{e} \in T$.*

To effectively use typed part of a polynomial for a specific type, the following notion of Hadamard Product is very useful. Given two linear forms $L_1 = \sum_{i=1}^n a_i x_i$ and $L_2 = \sum_{i=1}^n b_i x_i$, define

$$L_1 \circ^p L_2 = \sum_{i=1}^n a_i \cdot b_i \ x_i^2.$$

We can naturally extend the notion to define $L_1 \circ^p \ldots \circ^p L_d$.

Given a type $\boldsymbol{e} = (e_1, e_2, \ldots, e_q)$ and a product of linear forms $L_1 L_2 \cdots L_d$ where L_i may be same as L_j for distinct i, j, we define

$$L_{j,e_j} = L_{e_{[j-1]}+1} \circ^p L_{e_{[j-1]}+2} \circ^p \ldots \circ^p L_{e_{[j-1]}+e_j}$$

where $e_{[j-1]} = \sum_{t=1}^{j-1} e_t$. For any $\sigma \in S_d$ we define,

$$L_{j,e_j}^{\sigma} = L_{\sigma(e_{[j-1]}+1)} \circ^p L_{\sigma(e_{[j-1]}+2)} \circ^p \ldots \circ^p L_{\sigma(e_{[j-1]}+e_j)}.$$

For a fixed *type* $\boldsymbol{e} = (e_1, e_2, \ldots, e_q)$, from the proof of Lemma 2 we recall the definition of $E_k \subseteq [d]$ which denotes the interval $E_k = \{j \mid e_{k-1} + 1 \leq j \leq e_k\}, 1 \leq k \leq q$, where we set $e_0 = 0$. We say that $\sigma, \pi \in S_d$ are *identical* permutations with respect to the *type* $\boldsymbol{e}$ if $\sigma(E_k) = \pi(E_k)$ for $1 \leq k \leq q$.

Clearly the above relation is an equivalence relation on S_d which partitions the set of permutations. We construct the set A_e of *distinct* permutations by choosing one permutation from each equivalence class.

Lemma 6. *For any monomial $m = x_{i_1}^{e_1} x_{i_2}^{e_2} \ldots x_{i_q}^{e_q}$ of degree d and of type $\boldsymbol{e} = (e_1, e_2, \ldots, e_q)$ and a homogeneous $\Pi^{[d]}\Sigma$ circuit $P = \prod_{j=1}^d L_j$ we have:*

$$[m]P = \sum_{\sigma \in A_{\boldsymbol{e}}} \prod_{j=1}^{d} [x_{i_j}] L_{\sigma(j)} = \sum_{\sigma \in A_{\boldsymbol{e}}} \prod_{j=1}^{q} [x_{i_j}^{e_j}] L_{j,e_j}^{\sigma}.$$

Proof. The proof directly follows from Lemma 2. □

Now we apply a *diagonal* trick to carefully merge the linear forms in a $\Pi\Sigma$ circuit and obtain a $\Pi\Sigma\wedge$ circuit. For each product gate $P_i = \prod_{j=1}^d L_{ij}$, we define the polynomial

$$P_{i,\boldsymbol{e}} = \sum_{\sigma \in A_{\boldsymbol{e}}} \prod_{j=1}^{q} L_{ij,e_j}^{\sigma}.$$

Notice that, all the monomials of $P_{i,\boldsymbol{e}}$ are of form $x_{i_1}^{e_1} x_{i_2}^{e_2} \dots x_{i_q}^{e_q}$ where each i_j may not be distinct, but for those monomials m where each i_j is distinct, $[m]P_{i,\boldsymbol{e}} = [m]P$ from Lemma 6.

Now we give the proof of Theorem 3.

Proof of Theorem 3. Given the $\Sigma\Pi\Sigma$ circuit $C = \sum_{i=1}^{s} P_i$, we construct the following $\Sigma\Pi\Sigma\wedge$ circuit $C_{\boldsymbol{e}} = \sum_{i=1}^{s} P_{i,\boldsymbol{e}}$. Now we introduce a set of new variables $\{z_{i,e_j}\}_{i\in[n],j\in[q]}$ to make $C_{\boldsymbol{e}}$ multilinear. We replace $x_i^{e_j}$ with z_{i,e_j} at the bottom of the circuit $C_{\boldsymbol{e}}$ and get a multilinear $\Sigma\Pi\Sigma$ circuit, call it $C'_{\boldsymbol{e}}$. Now for a monomial $m_z = z_{i_1,e_{i_1}} z_{i_2,e_{i_2}} \dots z_{i_q,e_{i_q}}$, if $i_1, i_2, \dots, i_q$ are distinct then m_z is uniquely decoded into the monomial $m_x = x_{i_1}^{e_{i_1}} x_{i_2}^{e_{i_2}} \dots x_{i_q}^{e_{i_q}}$ and Lemma 6 tells us that

$$[m_z]C'_{\boldsymbol{e}} = [m_x]C_{\boldsymbol{e}} = [m_x]C.$$

Hence, we are only left with the following problem. Given a $\Sigma\Pi^{[q]}\Sigma$ circuit $C'_{\boldsymbol{e}}$ computing a multilinear homogeneous polynomial over $\{z_{i,e_j}\}_{i\in[n],j\in[q]}$, we want to get another $\Sigma\Pi^{[q]}\Sigma$ circuit $\hat{C}_{\boldsymbol{e}}$ keeping only the monomials of the form $z_{i_1,e_1} z_{i_2,e_2} \dots z_{i_q,e_q}$ with distinct i_j. We do not extract all these monomials at once, instead we use a (n, q)-perfect hash family $\mathcal{F}$ and extract those multilinear monomials that are *hashed* by a function $\zeta \in \mathcal{F}$. We achieve this by creating a $\Pi^{[q]}\Sigma$ circuit that contains monomials hashed by ζ and take *Hadamard Product* with $C'_{\boldsymbol{e}}$.

For a fixed type $\boldsymbol{e} = (e_1, e_2, \dots, e_q)$, define E as the set of distinct e_j's. For each type $\boldsymbol{e}$ and each function $\zeta \in \mathcal{F}$ we construct the following $\Pi\Sigma$ circuit:

$$P_{\zeta,\boldsymbol{e}} = \prod_{j=1}^{q} \left(\sum_{\hat{e}\in E} \sum_{i\in\zeta^{-1}(j)} z_{i,\hat{e}} \right).$$

Note that all monomials of $P_{\zeta,\boldsymbol{e}}$ have distinct first indices, and using Lemma 3 we construct

$$C'_{\zeta,\boldsymbol{e}} = C'_{\boldsymbol{e}} \circ^{s} P_{\zeta,\boldsymbol{e}}.$$

Now $C'_{\zeta,\boldsymbol{e}}$ is a $\Sigma\Pi^{[q]}\Sigma$ circuit computing a multilinear polynomial and from Remark 4 we know that we can apply Theorem 2 to do PIT . The correctness of the algorithm follows from the following claim.

Claim 2. *$C \equiv 0$ if and only if $C'_{\zeta,\boldsymbol{e}} \equiv 0$ for each $\boldsymbol{e} \in T$ and for each $\zeta \in \mathcal{F}$.*

Proof. From observation 1 we know that $C \equiv 0$ if and only if $f_{\boldsymbol{e}} \equiv 0$ for each $\boldsymbol{e} \in T$. Now each $C'_{\zeta,\boldsymbol{e}}$ contains encodings of monomials of $f_{\boldsymbol{e}}$ that are hashed by ζ, and by the property of the perfect hash family the collection $\{C'_{\zeta,\boldsymbol{e}}\}_{\zeta\in\mathcal{F}}$ covers every monomial of $f_{\boldsymbol{e}}$. Thus if C has a non-zero monomial m of type $\boldsymbol{e}$, its encoding m_z is also present in some $C'_{\zeta,\boldsymbol{e}}$ with $[m_z]C'_{\zeta,\boldsymbol{e}} = [m]C$. □

Our algorithm computes circuits $C'_{\zeta,\boldsymbol{e}}$ for each $\boldsymbol{e} \in T$ and $\zeta \in \mathcal{F}$ and runs the algorithm of Raz and Shpilka [RS05] on $C'_{\zeta,\boldsymbol{e}}$. If the size of C is s then the size of $C'_{\zeta,\boldsymbol{e}}$ is $2^{d\log d}s$, the algorithm of Raz and Shpilka [RS05] on each of these takes $2^{\gamma d\log d}\,\mathrm{poly}(n, d, s)$ time. We need to do PIT for each $C'_{\zeta,\boldsymbol{e}}$ and there are $|T| \cdot |\mathcal{F}| \leq 2^{2d\log d}$ many circuits. Thus the running time of the algorithm is $2^{(\gamma+2)d\log d}\,\mathrm{poly}(n, s, d)$. This completes the proof of Theorem 3.

References

[Agr05] Manindra Agrawal. Proving lower bounds via pseudo-random generators. In *FSTTCS 2005: Foundations of Software Technology and Theoretical Computer Science, 25th International Conference*, pages 92–105, 2005.

[AJ09] Vikraman Arvind and Pushkar S. Joglekar. Arithmetic circuits, monomial algebras and finite automata. In *Mathematical Foundations of Computer Science 2009, 34th International Symposium, MFCS 2009, Novy Smokovec, High Tatras, Slovakia, August 24-28, 2009. Proceedings*, pages 78–89, 2009.

[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. *J. ACM*, 45(3):501–555, 1998.

[AM10] Vikraman Arvind and Partha Mukhopadhyay. The ideal membership problem and polynomial identity testing. *Inf. Comput.*, 208(4):351–363, 2010.

[AS18] Vikraman Arvind and Srikanth Srinivasan. On the hardness of the noncommutative determinant. *Computational Complexity*, 27(1):1–29, 2018.

[AV08] Manindra Agrawal and V Vinay. Arithmetic circuits: A chasm at depth four. In *Proceedings-Annual Symposium on Foundations of Computer Science*, pages 67–75. IEEE, 2008.

[DL78] Richard A. DeMillo and Richard J. Lipton. A probabilistic remark on algebraic program testing. *Inf. Process. Lett.*, 7:193–195, 1978.

[dOSlV16] Rafael Mendes de Oliveira, Amir Shpilka, and Ben lee Volk. Subexponential size hitting sets for bounded depth multilinear formulas. *Computational Complexity*, 25(2):455–505, 2016.

[DS07] Zeev Dvir and Amir Shpilka. Locally decodable codes with two queries and polynomial identity testing for depth 3 circuits. *SIAM J. Comput.*, 36(5):1404–1434, 2007.

[GKKS13] Ankit Gupta, Pritish Kamath, Neeraj Kayal, and Ramprasad Saptharishi. Arithmetic circuits: A chasm at depth three. In *FOCS*, pages 578–587, 2013.

[HS80] Joos Heintz and Claus-Peter Schnorr. Testing polynomials which are easy to compute (extended abstract). In *Proceedings of the 12th Annual ACM Symposium on Theory of Computing, 1980*, pages 262–272, 1980.

[KI04] Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004.

[KS07] Neeraj Kayal and Nitin Saxena. Polynomial identity testing for depth 3 circuits. *Computational Complexity*, 16(2):115–138, 2007.

[KS09] Neeraj Kayal and Shubhangi Saraf. Blackbox polynomial identity testing for depth 3 circuits. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009*, pages 198–207, 2009.

[KS11] Zohar Shay Karnin and Amir Shpilka. Black box polynomial identity testing of generalized depth-3 arithmetic circuits with bounded top fan-in. *Combinatorica*, 31(3):333–364, 2011.

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

[MVV87] Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987.

[RS05] Ran Raz and Amir Shpilka. Deterministic polynomial identity testing in non-commutative models. *Computational Complexity*, 14(1):1–19, 2005.

[Rys63] H.J. Ryser. *Combinatorial Mathematics*. Carus mathematical monographs. Mathematical Association of America, 1963.

[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 *31st Annual Symposium on Foundations of Computer Science, St. Louis, Missouri, USA, October 22-24, 1990, Volume I*, pages 11–15, 1990.

[SS12] Nitin Saxena and C. Seshadhri. Blackbox identity testing for bounded top-fanin depth-3 circuits: The field doesn't matter. *SIAM J. Comput.*, 41(5):1285–1298, 2012.

[AG10] Noga Alon and Shai Gutner. Balanced families of perfect hash functions and their applications. *ACM Trans. Algorithms*, 6(3):54:1–54:12, 2010.

[NSS95] Moni Naor, Leonard J. Schulman, and Aravind Srinivasan. Splitters and near-optimal derandomization. In *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, 23-25 October 1995*, pages 182–191, 1995.

[SY10] Amir Shpilka and Amir Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5(3-4):207–388, 2010.

[Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. In *Symbolic and Algebraic Computation, EUROSAM '79, An International Symposiumon Symbolic and Algebraic Computation*, pages 216–226, 1979.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il