



Randomized Black-Box PIT for Small Depth $+$ -Regular Non-Commutative Circuits

G. V. Sumukha Bharadwaj ✉ 

Department of Computer Science & Engineering, Indian Institute of Technology Tirupati, India

S. Raja ✉ 

Department of Computer Science & Engineering, Indian Institute of Technology Tirupati, India

Abstract

In this paper, we address the black-box polynomial identity testing (PIT) problem for non-commutative polynomials computed by $+$ -regular circuits, a class of homogeneous circuits introduced by Arvind, Joglekar, Mukhopadhyay, and Raja (STOC 2017, Theory of Computing 2019). These circuits can compute polynomials with a number of monomials that are doubly exponential in the circuit size. They gave an efficient randomized PIT algorithm for $+$ -regular circuits of depth 3 and posed the problem of developing an efficient black-box PIT for higher depths as an open problem. Our work makes progress on this open problem by resolving it for constant-depth $+$ -regular circuits.

We present a randomized black-box polynomial-time algorithm for $+$ -regular circuits of any constant depth. Specifically, our algorithm runs in $s^{O(d^2)}$ time, where s and d represent the size and the depth of the $+$ -regular circuit, respectively. Our approach combines several key techniques in a novel way. We employ a nondeterministic substitution automaton that transforms the polynomial into a structured form and utilizes polynomial sparsification along with commutative transformations to maintain non-zeros. Additionally, we introduce matrix composition, coefficient modification via the automaton, and multi-entry outputs – methods that have not previously been applied in the context of black-box PIT. Together, these techniques enable us to effectively handle exponential degrees and doubly exponential sparsity in non-commutative settings, enabling polynomial identity testing for higher-depth circuits. In particular, we show that if f is a non-zero non-commutative polynomial in n variables over the field \mathbb{F} , computed by a depth- d $+$ -regular circuit of size s , then f cannot be a polynomial identity for the matrix algebra $\mathbb{M}_N(\mathbb{F})$, where $N = s^{O(d^2)}$ and the size of the field \mathbb{F} depends on the degree of f . Interestingly, the size of the matrices does not depend on the degree of f . Our result can be interpreted as an Amitsur-Levitzki-type result [2] for polynomials computed by small-depth $+$ -regular circuits.

2012 ACM Subject Classification Theory of computation \rightarrow Algebraic complexity theory

Keywords and phrases Polynomial Identity Testing, Non-commutative Circuits, Algebraic Circuits, $+$ -Regular Circuits, Black-Box

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2025.51

Related Version *Full Version:* <https://arxiv.org/abs/2411.06569> [5]

Acknowledgements We would like to extend our sincere gratitude to Prof. Arvind (IMSc and CMI) for his valuable discussions. SR also expresses his sincere gratitude to Prof. Arvind and Prof. Meena Mahajan (IMSc) for facilitating a visit to IMSc, where part of this research was conducted. We also acknowledge the assistance of ChatGPT in rephrasing sections of this paper to improve clarity and articulation. However, we affirm that no technical ideas or proofs presented in this paper were generated by ChatGPT.

1 Introduction

The non-commutative polynomial ring, denoted by $\mathbb{F}\langle X \rangle$, over a field \mathbb{F} in non-commuting variables X , consists of non-commuting polynomials in X . These are just \mathbb{F} -linear combinations of words (we call them monomials) over the alphabet $X = \{x_1, \dots, x_n\}$. Hyafil [8]



© G. V. Sumukha Bharadwaj and S. Raja;

licensed under Creative Commons License CC-BY 4.0

45th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2025).

Editors: C. Aiswarya, Ruta Mehta, and Subhajit Roy; Article No. 51; pp. 51:1–51:16



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

and Nisan [10], studied the complexity of *non-commutative* arithmetic computations, in particular the complexity of computing the determinant polynomial with non-commutative computations.

Non-commutative arithmetic circuit families compute non-commutative polynomial families in a non-commutative polynomial ring $\mathbb{F}\langle X \rangle$, where multiplication is non-commutative (i.e., for distinct $x, y \in X$, $xy \neq yx$). Designing an efficient deterministic algorithm for non-commutative polynomial identity testing is a major open problem. Let $f \in \mathbb{F}\langle X \rangle$ be a polynomial represented by a non-commutative arithmetic circuit C . In this work, we assume that the polynomial f is given by a black-box access to C , meaning we can evaluate the polynomial f on matrices with entries from \mathbb{F} or an extension field. Note that the degree of an n -variate polynomial computed by the circuit C of size s can be as large as 2^s and the sparsity, i.e., the number of non-zero monomials, can be as large as n^{2^s} . For example, the non-commutative polynomial $(x + y)^{2^s}$ has degree 2^s , doubly exponential sparsity 2^{2^s} , and has a circuit of size $O(s)$.

Bogdanov and Wee [6] have given an efficient randomized PIT algorithm for non-commutative circuits computing polynomials of degree $d = \text{poly}(s, n)$. Their algorithm is based on the result of Amitsur-Levitzki [2], which states the existence of matrix substitutions $M = (M_1, M_2, \dots, M_n)$ such that the matrix $f(M_1, M_2, \dots, M_n)$ is not the zero matrix, where the dimension of the matrices in M depends linearly on the degree d of the polynomial f . [2] shows that a non-zero non-commutative polynomial f of degree $2d - 1$ does not vanish on the matrix algebra $\mathcal{M}_d(\mathbb{F})$. Since the degree of the polynomial computed by circuit C can be exponentially large in the size of the circuit, their approach will not work directly. Finding an efficient randomized PIT algorithm for general non-commutative circuits is a well-known open problem. It was highlighted at the Workshop on Algebraic Complexity Theory (WACT 2016) as one of the key problems to work on.

Recently, [3, 4] gave an efficient randomized algorithm for the PIT problem when the circuits are allowed to compute polynomials of exponential degree, but the sparsity (i.e., the number of non-zero monomials) could be exponential in the size of the circuit. In particular, it has been shown that if the sparsity of the polynomial is k , then identity testing can be performed using matrices of dimension $O(\log k)$. To handle doubly-exponential sparsity, they studied a class of homogeneous non-commutative circuits, that they call $+$ -regular circuits, and gave an efficient deterministic white-box PIT algorithm. These circuits can compute non-commutative polynomials with the number of monomials that is doubly exponential in the circuit size. For the black-box setting, they obtain an efficient randomized PIT algorithm only for depth-3 $+$ -regular circuits. In particular, they show that if a non-zero non-commutative polynomial $f \in \mathbb{F}\langle X \rangle$ is computed by a depth-3 $+$ -regular circuit of size s , then f cannot be a polynomial identity for the matrix algebra $\mathbb{M}_s(\mathbb{F})$ for a sufficiently large field \mathbb{F} . That is, the polynomial f cannot be identically zero under all substitutions from the matrix algebra $\mathbb{M}_s(\mathbb{F})$ for its variables. In other words, there exists at least one matrix substitution for the variables in f such that the resulting evaluated matrix is non-zero.

Finding an efficient randomized PIT algorithm for higher depth $+$ -regular circuits is listed as an interesting open problem [3, 4]. We resolve this problem for constant depth $+$ -regular circuits. In particular, we show that if $f \in \mathbb{F}\langle X \rangle$ is a non-zero non-commutative polynomial computed by a depth- d $+$ -regular circuit of size s , then f cannot be a polynomial identity for the matrix algebra $\mathbb{M}_N(\mathbb{F})$, with $N = s^{O(d^2)}$ and the size of the field \mathbb{F} depends on the degree of polynomial f . We note that we get a *black-box* randomized polynomial time PIT algorithm for constant depth $+$ -regular circuits.

1.1 Our results

The main result of the paper is the following theorem¹.

► **Theorem 1.** *Let f be a non-commutative polynomial of degree D over $X = \{x_1, \dots, x_n\}$ computed by a $+$ -regular circuit of depth d and size s . Then $f \not\equiv 0$ if and only if f is not identically zero on the matrix algebra $\mathbb{M}_N(\mathbb{F})$, with $N = s^{O(d^2)}$ and \mathbb{F} is sufficiently large.*

For degree D non-zero non-commutative polynomial f , the classical Amitsur-Levitzki [2] theorem guarantees that f does not vanish on the matrix algebra $\mathcal{M}_{\frac{D}{2}+1}(\mathbb{F})$. If $D = 2^{\Omega(s)}$, this gives us an exponential time randomized PIT algorithm [6], where s is the size of the circuit computing f . In contrast, in our result, the dimension of the matrices is independent of the degree of the polynomial. If the sparsity of the polynomial, i.e., the number of non-zero monomials, is doubly exponential, then the main result of [4] gives only an exponential time randomized PIT algorithm as their matrix dimension depends on the logarithm of the sparsity.

This above theorem demonstrates that if the polynomial f is computed by a $+$ -regular circuit of size s and depth $o(\sqrt{s}/\log s)$, we can determine if f is identically zero or not using a $2^{o(s)}$ time randomized PIT algorithm, which is exponentially faster than the existing methods. In particular, if depth is $O(1)$ then our algorithm runs in polynomial time. It is important to note that the number of product gates (within each Π^* layers) in any input-to-output path can be arbitrary and is only bounded by the circuit size s .

We note that [4] presented a white-box deterministic polynomial-time PIT for arbitrary depth $+$ -regular circuits. For the small-degree case, [11] provided a white-box deterministic polynomial-time PIT for non-commutative ABPs, while [7, 1] have shown a deterministic quasi-polynomial-time black-box PIT algorithm for non-commutative ABPs. In the commutative setting, a randomized polynomial-time Polynomial Identity Test (PIT) was given by [9] using modular arithmetic evaluations.

1.2 Proof Sketch

We provide a brief sketch of our randomized PIT algorithm for constant-depth $+$ -regular circuits. Our approach proceeds through a sequence of transformations:

1. A *structured transformation* using substitution automata, which converts the non-commutative polynomial into a more structured form while introducing spurious monomials.
2. A *product sparsification* step that reduces the number of non-commutative polynomial factors in each product to a small subset, preserving non-zoneness.
3. A *commutative transformation* step that reduces the PIT problem to the commutative setting.

Each step relies on small substitution automata and guarantees preservation of non-zoneness. These steps are carefully composed to preserve non-zoneness while keeping the dimension of the substitution matrices polynomial in s . We also introduce a novel coefficient modification strategy to prevent cancellation between structured and spurious terms.

In the rest of the paper, we will use “n.c.” as an abbreviation for “non-commutative”.

¹ For all omitted proofs, definitions, figures, and additional details, please refer to the full version of this paper [5]

1.3 Organization of the Paper

Section 2 presents our randomized black-box PIT algorithm for depth-5 +-regular circuits, highlighting the main steps: structure transformation, product sparsification, commutative transformation, and coefficient modification. Section 3 extends this result to arbitrary constant-depth +-regular circuits.

2 Black-Box PIT for $\Sigma\Pi^*\Sigma\Pi^*\Sigma$ Circuits

In this section, we show that if $f \in \mathbb{F}\langle X \rangle$ is a n.c. polynomial of degree D computed by a depth-5 +-regular circuit of size s , then there exists an efficient randomized algorithm for identity testing the polynomial f . The main result of this section is the following theorem.

► **Theorem 2.** *Let f be a n.c. polynomial of degree D over $X = \{x_1, \dots, x_n\}$ computed by a $\Sigma\Pi^*\Sigma\Pi^*\Sigma$ circuit of size s . Then $f \neq 0$ if and only if f is not identically zero on the matrix algebra $\mathbb{M}_{s^6}(\mathbb{F})$.*

The polynomial f can be written as follows: $f = \sum_{i \in [s]} \prod_{j \in [D_2]} Q_{ij}$. Here, the degree of each Q_{ij} is D_1 , where $i \in [s], j \in [D_2]$, and it can be computed by a $\Sigma\Pi^*\Sigma$ circuit of size at most s . We establish Theorem 2 in three steps. First, we transform each polynomial Q_{ij} into a more structured n.c. polynomial, as defined below (see Definition 3). This structured polynomial has the property that we can simply consider it as a commutative polynomial preserving non-zerosness (Claim 5). As noted above, this is not true for n.c. polynomials, in general.

► **Definition 3.** *Let $s \in \mathbb{N} \cup \{0\}$. We call a n.c. polynomial g over $\xi = \{\xi_1, \xi_2, \dots, \xi_s\}$ as an s -ordered polynomial if it is of the form*

$$g = \sum_{i_1 \geq 0, \dots, i_s \geq 0} \alpha_i \cdot \xi_1^{i_1} \xi_2^{i_2} \dots \xi_s^{i_s}, \text{ where } \alpha_i \in \mathbb{F}.$$

► **Remark 4.** In the above definition, it is important to note that $i_j \geq 0$ for each $j \in [s]$. However, in this paper, we will focus on a special case of s -ordered polynomials, where $i_j > 0$, for each $j \in [s-1]$ and $i_s \geq 0$.

We have the following observation about the s -ordered polynomial.

► **Claim 5.** Suppose $g \in \mathbb{F}\langle \xi \rangle$ is an s -ordered polynomial. Let $g^{(c)}$ be the polynomial obtained by treating variables in ξ as commutative. Then, $g \neq 0$ if and only if $g^{(c)} \neq 0$.

► **Remark 6.** We will later allow coefficients α_i to be commutative polynomials. The proof of this generalized statement follows the same reasoning as the proof of Claim 5.

Let us define the following concept, which will be relevant in subsequent sections:

► **Definition 7** (ξ -Pattern / Ordered Monomial). *A ξ -pattern is a monomial of the form $\xi_1^{\ell_1} \xi_2^{\ell_2} \dots \xi_k^{\ell_k}$, where $\ell_1, \ell_2, \dots, \ell_k \geq 0$. In other words, it is a monomial in the variables $\xi_1, \xi_2, \dots, \xi_k$ where the variables appear in the fixed order $\xi_1, \xi_2, \dots, \xi_k$, but the exponents ℓ_i can vary. Such a monomial is also called an ordered monomial.*

We now explain the three steps of our method to establish Theorem 2.

2.1 Step 1: Transforming the Polynomial for Improved Structure

The initial step of our method involves transforming the polynomial to introduce more structure. During this process, we obtain a structured polynomial but also introduce some additional spurious monomials. This is one of the main differences between this work and [4]. We show that these spurious monomials have a distinguishing property that can be used to differentiate them from the structured part. We discuss the process of transforming each polynomial Q_{ij} into an s -ordered polynomial. To do this, we introduce a *new* set of commutative and n.c. variables as follows.

Let $Z = \{z_1, \dots, z_n\}$ and let $Y = \{y_{ij} \mid i \in [n] \text{ and } j \in [s-1]\}$. The variables in Y and Z are commutative. Let $\xi = \{\xi_1, \xi_2, \dots, \xi_s\}$ be the set of n.c. variables. We do this transformation using a small substitution automaton.

We consider the output of this substitution automaton on the n.c. polynomial f . The corresponding $s \times s$ substitution matrix $\mathbf{M}_{\mathbf{x}_i}$ for each variable x_i is defined from this substitution automaton.

Output of the Automaton

Let $\mathbf{M} = f(\mathbf{M}_{\mathbf{x}_1}, \mathbf{M}_{\mathbf{x}_2}, \dots, \mathbf{M}_{\mathbf{x}_n})$. Then we consider the *output of the automaton* as:

$$f' = \mathbf{M}[q_0, q_{s-1}] \quad (1)$$

which is a polynomial in the variables $\xi \sqcup Y \sqcup Z$.

Suppose a monomial m is computed by a $+$ -regular circuit C . The monomial m has non-zero coefficient in $\prod_{j \in [D_2]} Q_{ij}$ for some $i \in [s]$. This monomial can be written as $m = m_1 \cdot m_2 \cdot \dots \cdot m_{D_2}$, where each sub-monomial $m_j \in X^{D_1}$ has non-zero coefficient in Q_{ij} .

Next, we consider the output of the substitution automaton \mathcal{A} on m . The automaton knows how to replace/substitute any variable x_j at any state q . Suppose $m = x_{i_1} \cdot x_{i_2} \cdot x_{i_3} \cdot \dots \cdot x_{i_D}$, the output of the substitution automaton \mathcal{A} on the monomial m is given by

$$\mathbf{M}_{\mathbf{m}}[q_0, q_{s-1}]$$

where $\mathbf{M}_{\mathbf{m}} = \mathbf{M}_{\mathbf{x}_{i_1}} \cdot \mathbf{M}_{\mathbf{x}_{i_2}} \cdot \dots \cdot \mathbf{M}_{\mathbf{x}_{i_D}}$. Each variable $x_i, i \in [n]$, is substituted by a degree two monomial over $\xi \sqcup Y \sqcup Z$ (one n.c. variable and one commutative variable). Consequently, the automaton transforms the monomial m into a degree $2D$ polynomial over $\xi \sqcup Y \sqcup Z$. Importantly, the *new* n.c. degree (i.e., sum of exponents of ξ variables) equals to the original degree D .

Computation by the substitution automaton

The automaton has exponentially many paths (with states allowed to repeat) from q_0 to q_{s-1} , all labeled by the same monomial m . Each computation path transforms the monomial m , originally over the variables X , into a new monomial over $\xi \sqcup Y \sqcup Z$.

For any path ρ from q_0 to q_{s-1} , we denote the transformed monomial as m_ρ . The polynomial computed by $\mathbf{M}_{\mathbf{m}}[q_0, q_{s-1}]$ is given by

$$\sum_{\rho: q_0 \xrightarrow{m} q_{s-1}} m_\rho,$$

which is a polynomial in $\mathbb{F}[Y \sqcup Z][\xi]$. Recall that n.c. polynomials are \mathbb{F} -linear combinations of words/strings (called monomials). For a n.c. monomial m , we can identify the variable at position e in m , where $1 \leq e \leq |m|$.

Recall that m can be written as $m = m_1 \cdot m_2 \cdots m_{D_2}$. Each computation path ρ substitutes each n.c. variable in m according to the automaton's transition rules, resulting in a monomial m_ρ over new variables $\xi \sqcup Y \sqcup Z$. We group all commutative variables appearing in m_ρ and denote it by c_m , which is a commutative monomial over $Y \sqcup Z$. The resulting monomial m_ρ has the following form.

► **Proposition 8.** *Let ρ be a path from q_0 to q_{s-1} labeled by the monomial m . The transformed monomial m_ρ can be expressed in the form: $m_\rho = c_m \cdot m'_1 \cdot m'_2 \cdots m'_N$, where $N \geq 1$, c_m is a monomial over $Y \sqcup Z$, and each m'_ℓ (for $\ell \in [N]$) is given by $m'_\ell = \xi_1^{\ell_1} \cdot \xi_2^{\ell_2} \cdots \xi_s^{\ell_s}$, where $\ell_k > 0$ for $k \in [s-1]$, and $\ell_s \geq 0$.*

For $i \neq j$, the exponents of the ξ variables in the sub-monomials m'_i and m'_j can vary. In particular, it is generally possible that $(i_1, i_2, \dots, i_s) \neq (j_1, j_2, \dots, j_s)$.

Types of sub-monomials: Two cases

It is important to note that the number of new sub-monomials m'_i , denoted as N , may not be equal to D_2 . This is because N depends on how many times the path ρ returns to the initial state q_0 . Also, the sum of exponents of ξ variables in each sub-monomial m'_i in m_ρ can vary. This leads us to consider two possible cases for each computation path ρ that starts at q_0 and ends at q_{s-1} : (recall $m = m_1 \cdot m_2 \cdots m_{D_2}$).

- **Case 1:** For each $j < D_2$, the boundary between m_j and m_{j+1} in m is respected by the path ρ . In this computation path ρ , the state of \mathcal{A} is at q_0 precisely when it begins processing each sub-monomial $m_j \in X^{D_1}$ for $j \in [D_2]$. This means that when \mathcal{A} reads the last variable of the sub-monomial m_{j-1} (for $j > 1$), it transitions back to state q_0 . As a result, \mathcal{A} is in state q_0 exactly when it reads the first variable of the sub-monomial m_j . This holds true for all sub-monomials m_j where $j \in [N]$. By Proposition 8, the transformed monomial can be expressed as: $m_\rho = c_m \cdot m'_1 \cdot m'_2 \cdots m'_N$ where c_m is a monomial over $Y \sqcup Z$ and each m'_ℓ is of form $m'_\ell = \xi_1^{\ell_1} \cdots \xi_s^{\ell_s}$. In this case, we observe that $N = D_2$ since there are exactly D_2 sub-monomials in m . or
- **Case 2:** For some $j < D_2$, the boundary between m_j and m_{j+1} in m is not respected by the path ρ . In this case, there exists a sub-monomial m_j , where $j \in [D_2]$, such that either (1) the computation path ρ visits the state q_0 while processing the variable located at position c , where $1 < c \leq D_1$. This means ρ returns to q_0 in the middle of processing m_j . or (2) the path ρ is in a state $q_j, j \neq 0$ (i.e., other than the initial state q_0) while processing the variable that appears at the first position of the sub-monomial m_j . By Proposition 8, the transformed monomial can be expressed as: $m_\rho = c_m \cdot m'_1 \cdot m'_2 \cdots m'_N$ where c_m is a monomial over $Y \sqcup Z$ and each m'_ℓ is of form $m'_\ell = \xi_1^{\ell_1} \cdots \xi_s^{\ell_s}$. In this case, we cannot definitively say whether N is equal to D_2 or not.

► **Remark 9.** Any path ρ from q_0 to q_{s-1} labeled by a monomial $m \in X^D$ will satisfy either Case 1 or Case 2, but not both.

In Case 1, we can make the following important observation about the obtained monomial m_ρ . Recall that D_1 is the degree of Q_{ij} polynomial for all $i \in [s]$ and $j \in [D_2]$.

▷ **Claim 10.** Let ρ be a path from q_0 to q_{s-1} labeled by the monomial m that satisfies Case 1. In this case, for each sub-monomial m'_ℓ , where $\ell \in [D_2]$, of the monomial m_ρ , the sum of the exponents of its n.c. variables is D_1 . That is, $\sum_{j \in [s]} \ell_j = D_1$.

For all paths ρ that satisfy Case 2, this is not true. We note this down as the following claim.

▷ **Claim 11.** Let ρ be a path from q_0 to q_{s-1} labeled by the monomial m that satisfies Case 2. In this case, there exists a sub-monomial m'_ℓ , where $\ell \in [N]$, in the obtained monomial m_ρ such that the sum of the exponents of its n.c. variables is *not* equal to D_1 . That is, $\sum_{j \in [s]} \ell_j \neq D_1$.

We crucially utilize Claims 10 and 11 later to ensure the non-zerosness of the transformed commutative polynomial.

The *structured* part and the *spurious* part

For a monomial $m = m_1 \cdot m_2 \cdots m_{D_2}$, we define the polynomial \hat{f}_m as the sum of all monomials that are obtained from computation paths ρ labeled by m from Case 1 above. Similarly, the polynomial F_m is defined as the sum of all monomials obtained from computation paths ρ labeled by m from Case 2 above. We consider the output of the substitution automaton \mathcal{A} on the given n.c. polynomial $f \in \mathbb{F}\langle X \rangle$. The output of the automaton is the sum of all monomials produced by computation paths ρ starting from q_0 and leading to q_{s-1} , with these paths labeled by monomials generated by a depth-5 +-regular circuit.

Let $Mon(f)$ be the set of all monomials computed/generated by the given depth-5 +-regular circuit computing f . That is, suppose m is computed by $\prod_{j \in [D_2]} Q_{i,j}$ for some $i \in [s]$, with coefficient $\alpha_{m,i}$ then $\alpha_{m,i} \cdot m \in Mon(f)$. Let

$$\hat{f} = \sum_{\alpha_{m,i} \cdot m \in Mon(f)} \hat{f}_{\alpha_{m,i} \cdot m} \quad (2)$$

$$F = \sum_{\alpha_{m,i} \cdot m \in Mon(f)} F_{\alpha_{m,i} \cdot m}. \quad (3)$$

We refer to F as the sum of spurious monomials obtained from the automaton, which can be viewed as noise resulting from our method.

We assume that the linear forms in the $Q_{i,j}$ polynomials are numbered from 1 to D_1 . For $I \subseteq [D_1]$ with size at most $s-1$, define $Q_{i,j,I}$ as the polynomial obtained from $Q_{i,j}$ by treating linear forms indexed by I as non-commuting and the rest of the linear forms as commuting. We also substitute the n.c. variables that appear in linear forms indexed by I with double-indexed commutative variables Y , as shown in the substitution automaton.

We have the following claim regarding the polynomial \hat{f} .

▷ **Claim 12.** The polynomial \hat{f} can be expressed as $\hat{f} = \sum_{i \in [s]} \prod_{j \in [D_2]} \sum_{I \subseteq [D_1], |I|=s-1} Q_{i,j,I} \times \xi_I$ where $\xi_I = \xi_1^{\ell_1} \cdot \xi_2^{\ell_2 - \ell_1} \cdots \xi_s^{D - \ell_{s-1}}$ for $I = \{\ell_1, \ell_2, \dots, \ell_{s-1}\}$ such that $\ell_1 < \ell_2 < \dots < \ell_{s-1}$, and $Q_{i,j,I}$ is the polynomial obtained from $Q_{i,j}$ by treating the linear forms indexed by I as non-commuting and the rest of the linear forms as commuting.

Let

$$\hat{Q}_{ij} = \sum_{I \subseteq [D_1], |I|=s-1} Q_{i,j,I} \times \xi_I. \quad (4)$$

Then we can express \hat{f} as:

$$\hat{f} = \sum_{i \in [s]} \left(\prod_{j \in [D_2]} \hat{Q}_{ij} \right). \quad (5)$$

The output of the substitution automaton \mathcal{A} on the polynomial f is given by:

$$f' = \hat{f} + F.$$

This is stated in the following claim.

▷ **Claim 13.** Let f be a homogeneous n.c. polynomial computed by a $\Sigma\Pi^*\Sigma\Pi^*\Sigma$ circuit of size s . Then, the output $f' \in \mathbb{F}[Y \sqcup Z]\langle \xi \rangle$ of the substitution automaton \mathcal{A} on the polynomial f can be expressed as $f' = \hat{f} + F$, where \hat{f} is the structured part as defined in Equation 5 and F is the spurious part as defined in Equation 3.

2.1.1 Non-zeroness of \hat{f}

We establish that f' is non-zero by first proving that \hat{f} is not zero. This is shown in Lemma 15, which builds on the result of PIT for $\Sigma\Pi^*\Sigma$ circuits (see Section 6.2 in [4]). We briefly discuss this result.

Let $Z = \{z_1, \dots, z_n\}$ be the set of *new* commuting variables. Let $g \in \mathbb{F}\langle X \rangle$ be a polynomial of degree D computed by a $\Sigma\Pi^*\Sigma$ circuit of size s . Then g can be expressed as $g = \sum_{i \in [s]} \prod_{j \in [D]} L_{ij}$, where L_{ij} are homogeneous linear forms. Let $P_i = \prod_{j \in [D]} L_{ij}$, $i \in [s]$. We have $g = \sum_{i \in [s]} P_i$. For $I \subseteq [D]$ with size at most $s-1$, define $P_{i,I}$ as the polynomial obtained from P_i by treating linear forms indexed by I as non-commuting and the rest of the linear forms as commuting. We replace each n.c. variable x_i appearing in $[D] \setminus I$ by a *new* commuting variable z_i .

The number of n.c. linear forms appearing in $P_{i,I} \in \mathbb{F}[Z]\langle X \rangle$ is bounded by $|I| < s$. This is because the linear forms that appear with indices other than those in I are treated as commutative. Consequently, the number of non-commutative linear forms in $P_{i,I} \in \mathbb{F}[Z]\langle X \rangle$ is bounded by $|I| < s$. We refer to this as the n.c. degree of the polynomial $P_{i,I}$. Since this degree is small, $P_{i,I}$ can be converted into a commutative polynomial while preserving its non-zeroness. Let $P_{i,I}^{(c)}$ denote the commutative polynomial obtained from $P_{i,I}$ and define $g_I = \sum_{i \in [s]} P_{i,I}^{(c)}$. To keep all guesses of the set I distinct, additional commutative variables $\xi = \{\xi_1, \xi_2, \dots, \xi_{k+1}\}$ are introduced in [4]. The transformed commutative polynomial obtained in [4] is given by:

$$g^* = \sum_{I \subseteq [D_1], |I|=k} g_I \times \xi'_I \quad (6)$$

where $\xi'_I = \xi_1^{\ell_1-1} \cdot \xi_2^{\ell_2-\ell_1-1} \dots \xi_{k+1}^{D-\ell_k}$ with $I = \{\ell_1, \ell_2, \dots, \ell_k\}$. The degree of the monomial ξ'_I is $D - |I|$.

By Lemma 6.2 in [4], there exists a set of indices $I \subseteq [D]$, $|I| < s$, such that $g_I \neq 0$ implying $g^* \neq 0$. Replacing ξ'_I with $\xi_I = \xi_1^{\ell_1} \cdot \xi_2^{\ell_2-\ell_1} \dots \xi_{k+1}^{D-\ell_k}$ in g^* retains the non-zeroness of g^* while the degree of ξ_I becomes D .

► **Remark 14.**

1. Without loss of generality, we assume that the automaton nondeterministically guesses exactly $(s-1)$ indices, i.e., $|I| = s-1$ and the rest as commutative. If $|I| < s-1$, adding more indices still preserves non-zeroness.
2. If the degree of the polynomial g is smaller than $(s-1)$, we will handle this small-degree case separately. For now, we assume $D_1 \geq s-1$ in Lemma 15.

We have the following lemma that shows $\hat{f} \neq 0$.

► **Lemma 15.** *Let $f = \sum_{i \in [s]} \prod_{j \in [D_2]} Q_{ij}$ be a n.c. polynomial over $X = \{x_1, \dots, x_n\}$, computed by a $\Sigma\Pi^*\Sigma\Pi^*\Sigma$ circuit of size s . Assume that each polynomial Q_{ij} (for $i \in [s], j \in [D_2]$) has degree $D_1 \geq s - 1$. Define $\hat{f} \in \mathbb{F}[Y \sqcup Z]\langle \xi \rangle$ as $\hat{f} = \sum_{i \in [s]} \prod_{j \in [D_2]} \hat{Q}_{ij}$, where $\hat{Q}_{ij} = \sum_{I \subseteq [D_1], |I|=s-1} Q_{i,j,I} \times \xi_I$ and $Q_{i,j,I}$ denotes the polynomial obtained from $Q_{i,j}$ by treating linear forms indexed by I as non-commuting and the remaining linear forms as commuting.*

We will need a generalization of Lemma 15 for polynomials computed by larger depth $+$ -regular circuits.

The resulting n.c. polynomial \hat{f} (from Lemma 15) still has an exponential degree in ξ variables, but each \hat{Q}_{ij} is structured as s -ordered polynomials. Importantly, \hat{f} does not contain any monomials from the spurious polynomial $F = \sum_{m \in \text{Mon}(f)} F_m$.

► **Remark 16.** If the degree D_1 of each polynomial Q_{ij} is less than $s - 1$ (we call it as small degree case), this case requires a separate treatment.

We now show that each Q_{ij} polynomial can be converted into a structured polynomial \hat{Q}_{ij} . The key property of this transformation is that there is a bijection between the monomials of Q_{ij} and those of \hat{Q}_{ij} , which ensures that $\hat{Q}_{ij} \equiv 0$ if and only if $Q_{ij} \equiv 0$.

► **Proposition 17.** *For each $i \in [s], j \in [D_2]$, let $Q_{ij} \in \mathbb{F}\langle X \rangle$ be a n.c. polynomial computed by $\Sigma\Pi^*\Sigma$ circuit. There exists an explicit substitution automaton of size $O(s)$ that transforms Q_{ij} into a polynomial $\hat{Q}_{ij} \in \mathbb{F}[Y \sqcup Z]\langle \xi \rangle$ such that $\hat{Q}_{ij} \equiv 0$ if and only if $Q_{ij} \equiv 0$.*

It is easy to observe the following because the first index of each variable $z_{\ell k}$ indicates the position of the variable x_k within each Q_{ij} .

► **Observation 18.** *Suppose $\hat{Q}_{ij} \neq 0$. If we treat the variables $z_{\ell k}, \ell \in [c], k \in [n]$, appearing in \hat{Q}_{ij} as commuting, the resulting commutative polynomial $\hat{Q}_{ij}^{(c)}$ remains non-zero.*

This guarantees that for the small degree case, we can transform the polynomial f similarly to Lemma 15, ensuring that each Q_{ij} is transformed into \hat{Q}_{ij} which can be regarded as a commutative polynomial without making it zero. While \hat{Q}_{ij} remains a n.c. polynomial over Z , we acknowledge that our model is black-box and we do not know the value of D_1 . However, for the purpose of analyzing the existence of matrices of small dimensions for identity testing, we can assume D_1 is known.

Thus, we can successfully transform the given polynomial in both scenarios – whether $D_1 \geq s - 1$ or $D_1 < s - 1$ – ensuring that the resulting \hat{Q}_{ij} can be considered as a commutative polynomial without making it a zero polynomial.

However, it is important to note that this transformation alone will not provide a black-box PIT, as we cannot guarantee the non-zerosness of the sum of products of these \hat{Q}_{ij} polynomials. This is because if we simply treat all \hat{Q}_{ij} as commutative, the variables across different \hat{Q}_{ij} polynomials could mix, which may lead to cancellations. At this stage, the variables in Z are still considered n.c. in the polynomial \hat{f} .

2.1.2 Non-zerosness of f'

By Lemma 15, we established that $\hat{f} \neq 0$. Next, we show that the polynomial $f' = \hat{f} + F \neq 0$. In \hat{f} , for every monomial $m = m_1 m_2 \dots m_{D_2}$, and for all $\ell \in [D_2]$ each m_ℓ takes the form $\xi_1^{\ell_1} \xi_2^{\ell_2} \dots \xi_s^{\ell_s}$ where $\sum_{k \in [s]} \ell_k = D_1$ (see Claim 10). However, this property does not hold for monomials appearing in F (see Claim 11). Specifically, for any monomial $m' = m'_1 m'_2 \dots m'_N$ in F , there exists a sub-monomial $m'_a = \xi_1^{a_1} \xi_2^{a_2} \dots \xi_s^{a_s}$ such that $\sum_{h \in [s]} a_h \neq D_1$. This

distinction ensures that the monomials of \hat{f} do not cancel with those of F . Thus, we conclude that $f' = \hat{f} + F \neq 0$. It's important to note that if $f \equiv 0$ then clearly $f' \equiv 0$ as well (converse statement). We note these observations in the following claim.

▷ **Claim 19.** Let f be a homogeneous n.c. polynomial computed by a depth-5 +-regular circuit of size s . Then, $f \neq 0$ if and only if $f' = \hat{f} + F \neq 0$.

Next, we can simplify the polynomial f' using the Polynomial Identity Lemma for commutative polynomials. We replace the commuting variables $Y \sqcup Z$ with scalar substitutions from \mathbb{F} or an extension field, yielding a non-zero polynomial. Let us denote this resulting non-zero polynomial as \tilde{f} . After this substitution, the only remaining variables in \tilde{f} will be n.c. variables ξ .

Let us denote new polynomials obtained after replacing the commuting variables by scalars in \hat{f} and F by \hat{f}_1 and F_1 respectively. That is, $\tilde{f} = \hat{f}_1 + F_1$.

One of the goals of this transformation is to ensure that if ξ_s is followed by ξ_1 in the transformed monomials (for all such occurrences of ξ_s followed by ξ_1), there must be a transition from $Q_{i,j}$ to $Q_{i,j+1}$ for some $j \in [D_2]$. As noted, we cannot be sure of this. However, all those monomials where this transition occurs are captured in the structured part \hat{f}_1 . Since there is no such structure in F_1 , we cannot conclude anything about the monomials appearing in the spurious part F_1 .

► **Remark 20.** It is important to note that each monomial of \tilde{f} is a product of ξ -patterns (see Definition 7), and the boundaries of each ξ -pattern can be easily identified by an automaton which is crucially used by the remaining steps.

2.2 Step 2: Product Sparsification

In the second step of our transformation, we prove a *general lemma* that states if we have a sum of a small number of products of ordered polynomials, we can sparsify the product while preserving its non-zero property. Specifically, in each product term of the sum, we can treat only a small number of the ordered polynomials as non-commutative while treating the rest as commutative without affecting the non-zero nature of the polynomial. In particular, this step does not depend on the number of terms in each product.

We focus on the sparsification of the n.c. polynomial $\tilde{f} \in \mathbb{F}\langle\xi\rangle$, which was the output of Step (1). This transformation affects both the good part \hat{f}_1 and the spurious part F_1 of the polynomial $\tilde{f} \in \mathbb{F}\langle\xi\rangle$. We begin by analyzing the transformation of \hat{f}_1 , which is defined as:

$$\hat{f}_1 = \sum_{i \in [s]} \left(\prod_{j \in [D_2]} \hat{Q}_{ij} \right),$$

where each \hat{Q}_{ij} is an s -ordered polynomial in the n.c. variables $\xi = \{\xi_1, \dots, \xi_s\}$. Note that each \hat{Q}_{ij} is a homogeneous and degree D_1 n.c. polynomial.

The key observation is that we can preserve the non-zoneness of \hat{f}_1 by retaining at most $s-1$ of the s -ordered polynomials \hat{Q}_{ij} in each product $\prod_{j \in [D_2]} \hat{Q}_{ij}$ as non-commutative while treating the remaining ones as commutative. This is stated in the following lemma, which we refer to as the *product sparsification lemma*. This lemma generalizes Lemma 6.2 from [4]. However, unlike in [4], we are working with the product of non-commutative polynomials where the degree of individual factors can be greater than 1. If we simply treat them as commutative, as in [4], they may become zero. The proof of this lemma crucially relies on Claim 5. Unlike [4], one of the key distinctions in our setting is that the \hat{Q}_{ij} polynomial can be non-homogeneous in general.

The product sparsification step impacts both \hat{f}_1 and the spurious part F_1 of the polynomial \tilde{f} obtained after Step 1. We will address the effects on both components in a later step (see 2.4).

► **Lemma 21** (Product Sparsification Lemma). *Let $\hat{f}_1 = \sum_{i \in [s]} \prod_{j \in [D_2]} \hat{Q}_{ij}$, where each \hat{Q}_{ij} is an s -ordered polynomial of degree D_1 over $\xi = \{\xi_1, \xi_2, \dots, \xi_s\}$. Then, there exists a subset $I \subseteq [D_2]$ with size at most $s - 1$ such that if we treat the polynomials \hat{Q}_{ij} for $j \in I$, as non-commutative and the others ($j \notin I$) as commutative, then the polynomial \hat{f}_1 remains non-zero. Furthermore, each \hat{Q}_{ij} polynomial may be non-homogeneous in general. Moreover, there is a small substitution automaton of size $O(s)$ that performs this transformation.*

In other words, for each n.c. variable ξ_i (where $i \in [s]$) in \hat{f}_1 , there exists an $O(s)$ -dimensional matrix – acting as a transition matrix of a substitution automaton of size $O(s)$. By evaluating \hat{f}_1 on these matrices, the polynomial is transformed into a product-sparsified polynomial while maintaining its non-zero property.

► **Remark 22.** We remark that the proof of Lemma 21 relies solely on the fact that each \hat{Q}_{ij} polynomial in \hat{f}_1 is an ordered polynomial. In particular, the proof does not depend on the fact that each \hat{Q}_{ij} is obtained from a $\Sigma\Pi^*\Sigma$ circuit. Instead, the proof relies on the following facts:

1. the number of summands is small,
2. the boundary of each ordered polynomial can be efficiently identified using a small automaton

This makes it irrelevant where the \hat{Q}_{ij} polynomials originate from. As a result, we can apply this result whenever the given polynomial is represented as a sum of a small number of products of ordered polynomials (i.e., the number of summands is small). We will use this observation when working with higher-depth $+$ -regular circuits.

Since we do not know the index set I , the substitution automaton guesses the index set I . Since the index set $I \subseteq [D_2]$ is unknown, the automaton non-deterministically selects which \hat{Q}_{ij} polynomials will be treated as non-commutative. Given the structured nature of the polynomial \hat{f} , we can identify the *boundary* of each \hat{Q}_{ij} , ensuring that no additional spurious monomials are generated.

In the high-degree case ($D_1 \geq s - 1$), either ξ_s or ξ_{s-1} followed by ξ_1 indicates the end of each \hat{Q}_{ij} , which can be easily recognized by the automaton. In the low-degree case ($D_1 < s - 1$), the smaller degree allows us to identify the ends of each \hat{Q}_{ij} with a small automaton of size at most $s - 2$.

The substitution automaton selects at most $(s - 1)$ of the \hat{Q}_{ij} polynomials to be treated as n.c. while treating the remaining ones as commutative. The ξ variables in the chosen commutative polynomials \hat{Q}_{ij} are substituted with fresh commutative variables $\zeta = \{\zeta_1, \dots, \zeta_s\}$. In the selected commutative polynomials \hat{Q}_{ij} , each n.c. variable $\xi_k, k \in [s]$ is replaced by the corresponding commuting variable ζ_k . Additionally, to distinguish between different guesses made by the substitution automaton, we use fresh commutative block variables $\chi = \{\chi_1, \dots, \chi_s\}$.

Let $J = \{j_1, j_2, \dots, j_{s-1}\} \subseteq [D_2]$ with $j_1 < j_2 < \dots < j_{s-1}$. We define $\chi_J = \chi_1^{j_1-1} \cdot \chi_2^{j_2-j_1-1} \dots \chi_s^{D_2-j_{s-1}-1}$. If the automaton guesses the \hat{Q}_{ij} polynomials corresponding to the positions in the index set J as n.c., the output g_J of the substitution automaton for this specific guess J will be

$$g_J = \sum_{i \in [s]} \left(\prod_{j \in \bar{J}} \hat{Q}_{ij} \right) \left(\prod_{j \in J} \hat{Q}_{ij} \right) \times \chi_J. \quad (7)$$

Note that $\left(\prod_{j \in \bar{J}} \hat{Q}_{ij}\right)$ is a commutative polynomial over $\zeta = \{\zeta_1, \dots, \zeta_s\}$. We have the following proposition about the output of the substitution automaton, whose proof we omit as it follows by a straightforward argument.

► **Proposition 23.** *Let $\hat{f}_1 \in \mathbb{F}\langle \xi \rangle$ be the structured part of the polynomial obtained after Step 1. Let \hat{f}_1' be the output of the substitution automaton on the structured polynomial \hat{f}_1 and it can be expressed as*

$$\hat{f}_1' = \sum_{J \subseteq [D_2], |J|=s-1} g_J.$$

Moreover, $\hat{f}_1 \neq 0$ if and only if $\hat{f}_1' \neq 0$.

It's evident that for distinct guesses J and J' where $J \neq J'$, the monomials of g_J and $g_{J'}$ will not mix, since the sub-monomials χ_J and $\chi_{J'}$ are distinct (see Equation 7). By Lemma 21, there exists index set $J \subseteq [D_2]$ with size at most $s-1$, such that $g_J \neq 0$ implying $\hat{f}_1' \neq 0$.

Next, we can simplify \hat{f}_1' by using the Polynomial Identity Lemma for commutative polynomials to eliminate the commuting variables $\zeta \cup \chi$ by substituting scalars. As a result, the remaining variables in the polynomial will be solely the n.c. variables ξ .

Let us denote the new polynomial obtained after replacing the commuting variables by scalars in \hat{f}_1' by \hat{f}_2 .

This product sparsification step affects both the good part \hat{f}_1 and the spurious part F_1 of the polynomial \hat{f} obtained after Step 1. We will denote the new polynomial derived from the spurious part F_1 by F_2 . In Step 2, we apply product sparsification to both \hat{f}_1 and F_1 , which yields the n.c. polynomials \hat{f}_2 and F_2 respectively (with all commuting variables replaced by scalars).

2.3 Step 3: Commutative Transformation of \hat{f}_2

In this final step, we prove a general commutative transformation lemma, which states that if we have a non-commutative polynomial represented as a sum of products of a small number of ordered polynomials (i.e., the number of terms in each product is small), we can convert it into a commutative polynomial while preserving its non-zerosness property. In particular, this step does not depend on the number of summands. The key idea is to introduce a small number of new commutative variables to perform this transformation.

We now describe how to transform \hat{f}_2 into a commutative polynomial while preserving its non-zerosness. Note that \hat{f}_2 is a polynomial over $\mathbb{F}\langle \xi \rangle$. If we treat \hat{f}_2 as commutative by considering the n.c. variables ξ as commutative, the exponents of the variable ξ_i (for $i \in [s]$) from different n.c. \hat{Q}_{ij} polynomials will be summed (or mixed). This mixing makes it impossible to guarantee that the resulting polynomial remains non-zero.

However, we can carefully convert \hat{f}_2 into a commutative polynomial while preserving its non-zerosness. This is stated in the following lemma. In particular, there is a substitution automaton of size $O(s^2)$ that carries out this commutative transformation.

► **Lemma 24** (Commutative Transformation Lemma). *Let $g = \sum_{i \in [s]} \beta_i \left(\prod_{j \in [s]} \hat{Q}_{ij} \right)$, where $\beta_i \in \mathbb{F}$ and each \hat{Q}_{ij} is an s -ordered polynomial over $\xi = \{\xi_1, \xi_2, \dots, \xi_s\}$ of degree D . This can be expressed as $g = \sum_m \alpha_m m$, where each monomial m has the form $m = \prod_{j \in [s]} \xi_1^{i_{j1}} \xi_2^{i_{j2}} \dots \xi_s^{i_{js}}$. Then there exists a substitution automaton of size $O(s^2)$ that transforms the non-commutative polynomial g into a commutative polynomial $g^{(c)}$ while preserving non-zerosness. In particular, $g^{(c)} \equiv 0 \iff g \equiv 0$.*

In other words, for each n.c. variable ξ_i (where $i \in [s]$), there exists an $O(s^2)$ -dimensional matrix – acting as a transition matrix of a substitution automaton. By evaluating g on these matrices, the polynomial is transformed into a commutative polynomial $g^{(c)}$, while maintaining its non-zero property.

► **Remark 25.** We remark that Lemma 24 is more general. The proof depends only on the fact that the given non-commutative polynomial can be represented as a sum of products of a small number of ordered polynomials (i.e., the number of terms in each product is small). Crucially, the proof of this commutative transformation does not depend on the fact that the polynomials are derived from $+$ -regular circuits or whether they are homogeneous. Instead, the proof relies on the following two facts:

1. The number of terms in each product is small, and
2. Each term in the product is an ordered polynomial so that boundaries can be identified efficiently using an automaton.

This makes the result applicable whenever the given non-commutative polynomial is represented as a sum of products of a small number of ordered polynomials. We will use this observation when working with higher-depth $+$ -regular circuits.

By applying Lemma 24, we can transform the polynomial \hat{f}_2 , the structured part obtained after Step 2, into a commutative polynomial while preserving its non-zerosness. Let $\hat{f}_3^{(c)}$ denote the resulting commutative polynomial derived from \hat{f}_2 . Consequently, we establish that $\hat{f}_3^{(c)} \neq 0$ as a result of this lemma.

Next, given $\tilde{f} = \hat{f}_1 + F_1$, where \tilde{f} was obtained after Step 1, we can likewise transform \tilde{f} into a commutative polynomial. Let $F_3^{(c)}$ represent the commutative polynomial obtained from F after applying steps (2) and (3). If $\hat{f}_3^{(c)} + F_3^{(c)} \neq 0$, we have successfully converted a n.c. polynomial f , computed by a depth-5 $+$ -regular circuit, into a commutative polynomial that preserves non-zerosness. We can now check the non-zerosness of this commutative polynomial using the Polynomial Identity Lemma for commutative polynomials.

Assume $\hat{f}_3^{(c)} + F_3^{(c)} = 0$. We will now detail how to modify the coefficients of certain monomials in \tilde{f} , which was obtained in Step 1, before executing Steps (2) and (3). We establish that this coefficient modification maintains non-zerosness and remains non-zero even after the application of Steps (2) and (3).

2.4 Coefficient Modification by Modulo Counting Automaton

Assuming $\hat{f}_3^{(c)} + F_3^{(c)} = 0$, and given that $\hat{f}_3^{(c)} \neq 0$, it follows that $F_3^{(c)} \neq 0$ and $\hat{f}_3^{(c)} = -F_3^{(c)}$. To address this cancellation, we carefully modify certain monomial coefficients in the non-commutative polynomial $\tilde{f} = \hat{f}_1 + F_1$ prior to applying product sparsification (Lemma 21) and commutative transformation (Lemma 24). We show that these modifications ensure that the resulting polynomial remains non-zero after Steps (2) and (3).

2.5 Black-box Randomized PIT for $\Sigma\Pi^*\Sigma\Pi^*\Sigma$ Circuits

Each of these three steps, along with the coefficient modification step, results in its own set of matrices for evaluation. In particular, the matrices obtained in each step evaluate a n.c. polynomial derived from the previous step.

Given that our model operates as a black box, we cannot evaluate the polynomial in this manner. Instead, we require a single matrix substitution for each n.c. variable. To address this, we apply the matrix composition lemma to combine the substitution matrices from all

four steps into a single matrix for each n.c. variable. This approach allows us to establish an efficient randomized polynomial identity testing (PIT) algorithm for depth-5 +-regular circuits, as demonstrated in the following theorem.

► **Theorem 26.** *Let f be a non-commutative polynomial of degree D over $X = \{x_1, \dots, x_n\}$, computed by a $\Sigma\Pi^*\Sigma\Pi^*\Sigma$ circuit of size s . Then $f \neq 0$ if and only if it does not evaluate to zero on the matrix algebra $\mathbb{M}_{s^6}(\mathbb{F})$.*

The main idea of the proof is to compose the substitution matrices derived from the automata in Steps (1)–(3) using the matrix composition lemma.

► **Remark 27.** We observe that the commutative polynomial $f^{(c)} \in \mathbb{F}[W]$ is an s^2 -ordered polynomial over W , in the sense that we can arrange the variables in each monomial of $f^{(c)}$ in increasing order according to the first index of the W variables, allowing for some exponents to be zero as specified in Definition 3.

This is summarized in the following theorem.

► **Theorem 28.** *Let f be a non-zero non-commutative polynomial of degree D over $X = \{x_1, \dots, x_n\}$ computed by a $\Sigma\Pi^*\Sigma\Pi^*\Sigma$ circuit of size s . Then, f can be transformed into an s^2 -ordered polynomial while preserving its non-zerosness. In particular, there exists a small substitution automaton of size $O(s^6)$ that performs this transformation.*

In other words, for each n.c. input variable x_i (where $i \in [n]$) in f , there exists an $O(s^6)$ -dimensional matrix – acting as a transition matrix of a substitution automaton of size $O(s^6)$. By evaluating f on these matrices, the polynomial f is transformed into an s^2 -ordered polynomial while preserving its non-zerosness (in one particular entry of the resulting matrix).

2.5.1 An Automaton for Theorem 28

We can envision a substitution automaton \mathcal{A} for Theorem 28 as follows. By applying the matrix composition Lemma, we can combine the substitution matrices obtained from Steps (1) through (3), along with the modifications to coefficients, into a single substitution matrix $\mathbf{M} = (\mathbf{M}_{\mathbf{x}_1}, \mathbf{M}_{\mathbf{x}_2}, \dots, \mathbf{M}_{\mathbf{x}_n})$ of dimension $O(s^6)$. We then evaluate the polynomial as $\mathbf{O} = f(\mathbf{M}_{\mathbf{x}_1}, \mathbf{M}_{\mathbf{x}_2}, \dots, \mathbf{M}_{\mathbf{x}_n})$.

The output of the substitution automaton is defined as the sum of several entries of the matrix \mathbf{O} (refer to the polynomial $f^{(c)}$ defined in the proof of Theorem 26).

It is crucial to note that each entry of the matrices in $\mathbf{M} = (\mathbf{M}_{\mathbf{x}_1}, \mathbf{M}_{\mathbf{x}_2}, \dots, \mathbf{M}_{\mathbf{x}_n})$ is a monomial over $Y \sqcup Z \sqcup \zeta \sqcup \chi \sqcup W$, where $Y \sqcup Z$ are commutative variables from Step (1), and $\zeta \sqcup \chi$ are commutative variables from Step (2), while W contains commutative variables from Step (3).

As noted in Steps (1) and (2), we can replace the commutative variables in $Y \sqcup Z \sqcup \zeta \sqcup \chi$ with scalars without losing the non-zerosness of the output polynomial. Since these commutative variables are disjoint, for simplicity in the analysis, we substitute them with scalars. That is, there exist scalar substitutions for the variables in $Y \sqcup Z \sqcup \zeta \sqcup \chi$ such that non-zerosness is preserved. To avoid carrying these variables throughout all derivations, we replace them with scalars, and by the DeMillo–Lipton–Schwartz–Zippel lemma, such substitutions are guaranteed to exist. After these replacements, each entry of the matrices in $\mathbf{M} = (\mathbf{M}_{\mathbf{x}_1}, \mathbf{M}_{\mathbf{x}_2}, \dots, \mathbf{M}_{\mathbf{x}_n})$ transforms into scalar multiples of variables over W . We denote the resulting matrices as $\mathbf{M}' = (\mathbf{M}'_{\mathbf{x}_1}, \mathbf{M}'_{\mathbf{x}_2}, \dots, \mathbf{M}'_{\mathbf{x}_n})$.

We can construct a substitution automaton \mathcal{A} such that the substitution matrix for the variable x_i is given by the matrix \mathbf{M}'_{x_i} , where the entries are scalar multiples of variables in W . These entries correspond to transitions that substitute a n.c. variable with a scalar multiple of a variable in W . This automaton \mathcal{A} effectively transforms f into an s^2 -ordered polynomial $f^{(c)}$ while preserving its non-zoneness.

We can view the resulting s^2 -ordered polynomial $f^{(c)}$ as a n.c. polynomial over W . This idea is crucial for developing black-box randomized polynomial identity testing (PIT) for circuits of larger depths using induction.

► **Remark 29.** Note that we only *view* $f^{(c)}$ as a n.c. polynomial and no explicit conversion or transformation is done.

It is important to note that the monomials of $f^{(c)}$ do not correspond to a single entry of the output matrix $\mathbf{O} = f(\mathbf{M}'_{x_1}, \mathbf{M}'_{x_2}, \dots, \mathbf{M}'_{x_n})$. Instead, they represent the sum of several entries, as indicated in the polynomial $f^{(c)}$ defined in the proof of Theorem 26. Effectively, the column numbers of these entries form the set of accepting states for the new automaton \mathcal{A} , with row 1 serving as the starting state of this automaton.

3 Black-Box Randomized PIT for Small Depth \pm -Regular Circuits

In this section, we present an efficient randomized black-box polynomial identity testing (PIT) algorithm for polynomials computed by small-depth \pm -regular circuits.

► **Theorem 30.** *Let f be a non-commutative polynomial of degree D over $X = \{x_1, \dots, x_n\}$, computed by a \pm -regular circuit of size s and depth d . Then, $f \not\equiv 0$ if and only if f is not identically zero on $\mathbb{M}_N(\mathbb{F})$, where $N = s^{O(d^2)}$ and $|\mathbb{F}|$ is sufficiently large.*

3.1 Transforming f into an ordered polynomial

Similar to depth-5 case, the polynomial f computed by a size s depth d \pm -regular circuit can be converted into an ordered polynomial using a substitution automaton of size at most $s^{O(d^2)}$. We state the result in the following theorem.

► **Theorem 31.** *Let f be a non-zero n.c. polynomial of degree D over variables $X = \{x_1, \dots, x_n\}$, computed by a \pm -regular circuit of size s and depth d . Let d^+ denote the number of Σ layers in the circuit. Then there exists a substitution automaton \mathcal{A} of size at most $s^{O(d^2)}$ such that the polynomial $\hat{f} := f(\mathcal{A})$ is an $s^{(d^+-1)}$ -ordered polynomial. Moreover, this transformation preserves non-zoneness: $\hat{f} \equiv 0 \iff f \equiv 0$.*

3.2 Randomized Identity Test for Small Depth \pm -Regular Circuits

We are now ready to state and prove the main theorem.

► **Theorem 32.** *Let f be a non-commutative polynomial of degree D over $X = \{x_1, \dots, x_n\}$, computed by a \pm -regular circuit of size s and depth d . We denote the number of addition (i.e., Σ) layers in the circuit by d^+ . Then, $f \not\equiv 0$ if and only if f is not identically zero on $\mathbb{M}_N(\mathbb{F})$, where $N = s^{O(d^2)}$ and $|\mathbb{F}|$ is sufficiently large.*

Proof. We use Theorem 31 to convert the n.c. polynomial f into an $s^{(d^+-1)}$ -ordered polynomial f_{ops} , while preserving its non-zoneness. As discussed above, there is a substitution automaton of size bounded by $s^{O(d^2)}$, which results in substitution matrices of dimension $s^{O(d^2)}$. By Claim 5, f_{ops} can be treated as a commutative polynomial while preserving its

non-zeroneess. Using the DeMillo-Lipton-Schwartz-Zippel lemma, we can have a randomized PIT for depth d $+$ -regular circuit of size s using matrices of dimension at most $s^{O(d^2)}$. This completes the proof of the theorem. \blacktriangleleft

References

- 1 Manindra Agrawal, Rohit Gurjar, Arpita Korwar, and Nitin Saxena. Hitting-sets for ROABP and sum of set-multilinear circuits. *SIAM J. Comput.*, 44(3):669–697, 2015. doi:10.1137/140975103.
- 2 Avraham Shimshon Amitsur and Jacob Levitzki. Minimal identities for algebras. *Proceedings of the American Mathematical Society*, 1(4):449–463, 1950.
- 3 Vikraman Arvind, Pushkar S. Joglekar, Partha Mukhopadhyay, and S. Raja. Randomized polynomial time identity testing for noncommutative circuits. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 831–841. ACM, 2017. doi:10.1145/3055399.3055442.
- 4 Vikraman Arvind, Pushkar S. Joglekar, Partha Mukhopadhyay, and S. Raja. Randomized polynomial-time identity testing for noncommutative circuits. *Theory of Computing*, 15:1–36, 2019. doi:10.4086/toc.2019.v015a007.
- 5 G. V. Sumukha Bharadwaj and S. Raja. Randomized black-box PIT for small depth $+$ -regular non-commutative circuits, 2025. doi:10.48550/arXiv.2411.06569.
- 6 Andrej Bogdanov and Hoeteck Wee. More on noncommutative polynomial identity testing. In *20th Annual IEEE Conference on Computational Complexity (CCC'05)*, pages 92–99. IEEE, 2005. doi:10.1109/CCC.2005.13.
- 7 Michael A. Forbes and Amir Shpilka. Quasipolynomial-time identity testing of non-commutative and read-once oblivious algebraic branching programs. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 243–252. IEEE, 2013. doi:10.1109/FOCS.2013.34.
- 8 Laurent Hyafil. The power of commutativity. In *18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977*, pages 171–174. IEEE Computer Society, 1977. doi:10.1109/SFCS.1977.31.
- 9 Oscar H. Ibarra and Shlomo Moran. Probabilistic algorithms for deciding equivalence of straight-line programs. *Journal of the ACM (JACM)*, 30(1):217–228, 1983. doi:10.1145/322358.322373.
- 10 Noam Nisan. Lower bounds for non-commutative computation. In *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pages 410–418, 1991.
- 11 Ran Raz and Amir Shpilka. Deterministic polynomial identity testing in non-commutative models. *Computational Complexity*, 14(1):1–19, 2005. doi:10.1007/s00037-005-0188-8.