Lower Bounds and PIT for Non-Commutative Arithmetic Circuits with Restricted Parse Trees[*]

Guillaume Lagarde[1], Nutan Limaye[2], and Srikanth Srinivasan[3]

1 **Univ Paris Diderot, Sorbonne Paris Cité, IRIF, UMR 7089 CNRS, Paris, France**
guillaume.lagarde@irif.fr

2 **Department of Computer Science and Engineering, IIT Bombay, Mumbai, India**
nutan@cse.iitb.ac.in

3 **Department of Mathematics, IIT Bombay, Mumbai, India**
srikanth@math.iitb.ac.in

Abstract

We investigate the power of *Non-commutative Arithmetic Circuits*, which compute polynomials over the free non-commutative polynomial ring $\mathbb{F}\langle x_1, \ldots, x_N \rangle$, where variables do not commute. We consider circuits that are restricted in the ways in which they can compute monomials: this can be seen as restricting the families of *parse trees* that appear in the circuit. Such restrictions capture essentially all non-commutative circuit models for which lower bounds are known. We prove several results about such circuits.

1. We show explicit exponential lower bounds for circuits with up to an exponential number of parse trees, strengthening the work of Lagarde, Malod, and Perifel (ECCC 2016), who prove such a result for *Unique Parse Tree* (UPT) circuits which have a single parse tree.
2. We show explicit exponential lower bounds for circuits whose parse trees are rotations of a single tree. This simultaneously generalizes recent lower bounds of Limaye, Malod, and Srinivasan (Theory of Computing 2016) and the above lower bounds of Lagarde et al., which are known to be incomparable.
3. We make progress on a question of Nisan (STOC 1991) regarding separating the power of Algebraic Branching Programs (ABPs) and Formulas in the non-commutative setting by showing a tight lower bound of $n^{\Omega(\log d)}$ for any UPT formula computing the product of d $n \times n$ matrices.

   When $d \leq \log n$, we can also prove superpolynomial lower bounds for formulas with up to $2^{o(d)}$ many parse trees (for computing the same polynomial). Improving this bound to allow for $2^{O(d)}$ trees would yield an unconditional separation between ABPs and Formulas.
4. We give deterministic white-box PIT algorithms for UPT circuits over any field (strengthening a result of Lagarde et al. (2016)) and also for sums of a constant number of UPT circuits with different parse trees.

1998 ACM Subject Classification F.1.2 Modes of Computation

Keywords and phrases Non-commutative Arithemetic circuits, Partial derivatives, restrictions

Digital Object Identifier 10.4230/LIPIcs.MFCS.2017.41

[*] A full version of the paper is available at https://eccc.weizmann.ac.il/report/2017/077/.

1 Introduction

In this paper, we study questions related to Arithmetic Circuits, which are computational devices that use arithmetic operations (such as $+$ and $\times$) to compute multivariate polynomials over a field $\mathbb{F}$. While the more standard work in this area deals with the commutative polynomial ring $\mathbb{F}[x_1, \ldots, x_N]$, there is also a line of research, initiated by Hyafil [12] and Nisan [21], that studies the complexity of computing polynomials from the *non-commutative* polynomial ring $\mathbb{F}\langle x_1, \ldots, x_N\rangle$, where monomials are simply strings over the alphabet $X = \{x_1, \ldots, x_N\}$. The motivation for this is twofold: firstly, the study of polynomial computations over non-commutative algebras (e.g. the algebra of matrices over $\mathbb{F}$) naturally leads to such questions [7, 6], and secondly, computing, say, the Permanent non-commutatively[1] is at least as hard as computing it in the commutative setting, and thus the lower bound question should be easier to tackle in the non-commutative setting.

In an influential result, Nisan [21] justified this by proving exponential lower bounds for non-commutative *formulas*, and more generally *Algebraic Branching Programs* (ABPs), computing the Determinant and Permanent (and also other polynomials). The method used by Nisan to prove this lower bound can also be seen as a precursor to the method of Partial derivatives in Arithmetic circuit complexity (introduced by Nisan and Wigderson [22]), variants of which have been used to prove a large body of lower bound results in the area [22, 25, 9, 14, 16].

While lower bounds for general non-commutative circuits remain elusive, we do have other lower bounds that strengthen Nisan's result. Recently, Malod, along with two of the authors of this paper showed [19] that Nisan's method can be extended to prove lower bounds for *skew circuits*, which are circuits where every $\times$-gate has at most one non-variable input. Also, the first author, Malod and Perifel [18] proved lower bounds for another variant of non-commutative circuits that they defined to be *unambiguous* circuits (which we describe below). While these two results both strengthen Nisan's result, they are incomparable to each other, as shown in [18].

In this paper, we build on the above work to prove lower bounds that generalize these results significantly and also make progress on other problems related to non-commutative circuits. The circuits we consider are restricted in the ways they are allowed to compute monomials. We do this by restricting the "parse trees" that are allowed to appear in the circuits. Informally, the polynomial computed by any arithmetic circuit C can be written down as an exponentially-large sum of subcircuits, each of which contains only multiplication gates and hence computes a single monomial[2]; each such subcircuit gives rise to a tree, which we call a *parse tree of* C (see [18] and references therein for the background of parse trees), that tells us how the monomial was computed. For example, for the circuit C in Figure 1, the monomial $x_1x_2x_3x_4$ may be computed in C as $(x_1 \cdot x_2) \cdot (x_3 \cdot x_4)$ or as $(x_1 \cdot (x_2 \cdot x_3) \cdot x_4)$, each of which comes from a parse tree of C.

All the non-commutative circuit classes for which we know lower bounds can be defined by restrictions on the parse trees that appear in them. ABPs are circuits where all parse trees are left combs (i.e. a tree where every internal node has two children and the left child is always a leaf); skew circuits are equivalent in power to circuits where the parse trees are *twisted* combs (i.e. a tree where every internal node has two children and at least one of the

[1] We can define the Permanent in the non-commutative polynomial ring by ordering the variables in each monomial in the commutative permanent, say, in increasing order of the rows in which they appear.

[2] different subcircuits could compute the same monomial

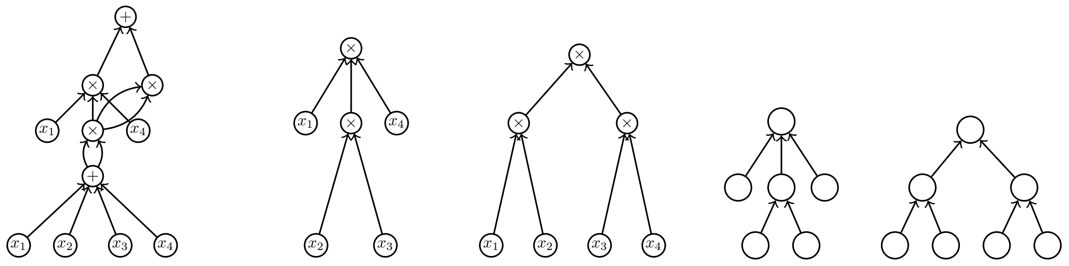

Figure 1 From left to right: a non-commutative arithmetic circuit C; two ways in which the monomial $x_1x_2x_3x_4$ is computed in the circuit; the corresponding parse trees.

two children is always a leaf); and unambiguous circuits (that we will call *Unique Parse Tree* (UPT) circuits below) are defined to be circuits that have only one parse tree. It is thus natural to consider other restrictions on the structure of the parse trees that appear in a circuit. We prove the following results about such circuits.

- We prove lower bounds for circuits that only contain a few different parse trees. The motivation for this is the lower bound of [18] for the case of circuits with a single parse tree and a construction in [19] that shows that poly(N, d)-sized circuits with $\exp(\Omega(d))$ many parse trees[3] evade all currently known techniques for proving lower bounds for non-commutative circuits. We show explicit exponential lower bound for circuits with up to $\exp(d^{\Omega(1)})$ parse trees (Theorem 13).
- We also consider structural restrictions on the collections of parse trees that appear in our circuits. As mentioned above, skew circuits are circuits where all parse trees are twisted combs, which can be seen as trees obtained by starting with a left comb (which defines an ABP) and successively applying rotations to the internal nodes that swap the children. We say that a circuit C is *rotation Unique Parse tree* (rotUPT) if there is a single tree T such that all the parse trees of C can be obtained as rotations of T.[4]
  We show an explicit exponential lower bound for rotUPT circuits (Theorem 17). Note that this result simultaneously generalizes the skew circuit lower bound of Limaye et al. [19] as well as the UPT circuit lower bound of Lagarde et al. [18].
- We consider the problem of separating ABPs from formulas, which was posed by Nisan [21] and is the non-commutative arithmetic analogue of separating NL from NC^1. Equivalently, this is the question of whether (an entry of) the product of d $n \times n$ matrices, all of whose entries are distinct variables, can be computed by a poly(n, d)-sized non-commutative formula. The standard divide-and-conquer approach yields, for every even Δ, a non-commutative formula of depth Δ and size $n^{O(\Delta d^{2/\Delta})}$ computing this polynomial and a size $n^{O(\log d)}$ formula in general. Further, these formulas can be seen to have a *unique* parse tree (i.e. they are UPT).
  We show that this upper bound is nearly tight for UPT formulas and every choice of Δ by showing a lower bound of $n^{\Omega(\Delta d^{1/\lfloor \Delta/2 \rfloor})}$.[5] (Theorem 18). In particular, our result implies that any UPT formula for this polynomial must have size $n^{\Omega(\log d)}$. We can extend this (Theorem 19) to prove a superpolynomial lower bound even in the case that the formula has at most k parse trees, where $k = 2^{o(d)}$ (however, for this result, we need the assumption that $d \leq \log n$).

[3] A close look at the circuits in [19] indicates that just about all parse trees of fan-in 2 appear in these circuits.
[4] There can be $\exp(\Omega(d))$ of these, as in the case of skew circuits.
[5] Our bounds are actually better stated in terms of the $\times$-*depth* of the formula.

- Finally, we consider the Polynomial Identity Testing (PIT) problem for non-commutative circuits with restricted parse trees. Lagarde et al. [18] show that deterministic PIT algorithms for UPT circuits can be obtained by adapting a PIT algorithm for ABPs due to Arvind, Joglekar and Srinivasan [3]. However, this technique only works over fields of characteristic zero. Here, we give a straightforward adaptation of an older PIT algorithm of Raz and Shpilka [24] (also for non-commutative ABPs) to show that PIT for UPT circuits can be solved in deterministic polynomial time over all fields (Theorem 20). We also consider circuits that are sums of UPT circuits (with possibly different parse trees). By using ideas from the work of Gurjar, Korwar, Saxena and Thierauf [10], we show that PIT for a sum of constant number of UPT circuits can be solved in deterministic polynomial time over any field (Theorem 21).

For lack of space, many of the proofs of the above statements have been omitted from this extended abstract. We refer the reader to the full version [17] for detailed proofs.

Related work. Hrubeš, Wigderson and Yehudayoff [11] initiated a study of the asymptotics of the classical *sum-of-squares* problem in mathematics and showed that a suitable result in this direction would yield strong lower bounds against general non-commutative circuits. While this line of work is currently the only feasible attack on the problem of general circuit lower bounds, we do not yet have any lower bounds using this technique.

Nisan and Wigderson [22] prove results that imply[6] some lower bounds for UPT formulas computing iterated matrix product. For depth-3 formulas, they prove an optimal n^d bound on computing the product of d $n \times n$ matrices. For depths $\Delta > 3$ though, the lower bound is only $\exp(\Theta(d^{1/\Delta}))$ and thus does not yield anything non-trivial when Δ approaches $\log d$. Indeed, the proof method of this result in [22] is not sensitive to the value of n and holds for any $n \geq 2$. Such a method cannot yield non-trivial lower bounds for general formulas since we do have poly(d)-sized formulas in the setting when $n = O(1)$.

The results of Kayal, Saha, and Saptharishi [15] and Fournier, Limaye, Malod, and Srinivasan [8] together also prove a superpolynomial lower bound on the size of *regular* formulas (defined by [15]) computing the product of d $n \times n$ matrices in the commutative setting. While these formulas (in the non-commutative setting) are definitely UPT, the converse is not true.

Arvind, Mukhopadhyay and Raja [4] and Arvind, Joglekar, Mukhopadhyay and Raja [2] have some recent work on PIT algorithms for general non-commutative circuits that run in time *polylogarithmic* in the degree of the circuit and polynomial in the size of the circuit. Our results are incomparable with theirs, since our algorithms run in time polynomial in *both* degree and size but are deterministic, whereas the algorithms of [4, 2] are faster (especially in terms of degree) but *randomized*.

An earlier manuscript of Arvind and Raja [5] contains a claim that the PIT problem for non-commutative skew circuits has a deterministic polynomial time algorithm, but the proof is unfortunately flawed.[7]

Techniques. The techniques used to prove the lower bounds in this paper are generalizations of the techniques of Hyafil [12] and Nisan [21]. Given a homogeneous polynomial $f \in \mathbb{F}\langle X\rangle$ of degree d, we associate with it an $N^{d/2} \times N^{d/2}$ matrix whose rows and columns are labelled

[6] The results of [22] in fact hold in the stronger commutative *set-multilinear* setting.
[7] Private communication with the authors.

by monomials (i.e. strings over X) m of degree $d/2$ each. Nisan [21] considers the matrix $M[f]$ where the (m_1, m_2)th entry is the coefficient of the monomial m_1m_2 in f. In [19, 18], along with our co-authors, we considered the more general family of matrices $M_Y[f]$ where $Y \subseteq [d]$ is of size $d/2$ and the (m_1, m_2)th entry of $M_Y[f]$ is the coefficient of the monomial m such that the projection of m to the locations in Y gives m_1 and the locations outside Y give m_2.

This is the general technique we use in this paper as well, though choosing the right Y requires some work. In the lower bound for circuits with few parse trees, it is chosen at random (in a similar spirit to a multilinear lower bound of Raz [23]). In the lower bound for rotUPT circuits, it is chosen in a way that depends on the structure of the parse trees in the circuit (combining the approaches of [19, 18]). In the separation of UPT formulas from ABPs, it is applied (after a suitable restriction) in a way that keeps the iterated matrix product polynomial high rank but reduces the rank of the UPT formula.

For the PIT algorithm for sums of UPT circuits, we use an observation of Gurjar et al. [10] (also see [21, 24]) that any polynomial P that has a small ABP has a small set of *characterizing identitites* such that $Q = P$ iff Q satisfies these identities. We are able to show (using a suitable decomposition lemma of [18]) that a similar fact is also true more generally in the case that P has a small UPT circuit. If Q also has a small UPT circuit, then checking these identities for Q reduces to a PIT circuit for a single UPT circuit, for which we already have algorithms. In this way, given two UPT circuits (with different parse trees) computing P, Q, we can check if $P - Q = 0$. Extending this idea exactly as in [10], we can efficiently check if the sum of any small number of UPT circuits is 0.

2 Preliminaries

We refer the reader to the survey [26] for standard definitions regarding arithmetic circuits.

2.1 Non-commutative polynomials

Throughout, we use $X = \{x_1, \ldots, x_N\}$ to denote the set of variables. We work over the *non-commutative* ring of polynomials $\mathbb{F}\langle X\rangle$ where monomials are *strings* over the alphabet X: for example, x_1x_2 and x_2x_1 are distinct monomials in this ring. For $d \in \mathbb{N}$, we use $\mathcal{M}_d(X)$ to denote the set of monomials (i.e. strings) over the variables in X of degree exactly d.

For $i, j \in \mathbb{N}$, we define $[i, j]$ to be the set $\{i, i+1, \ldots, j\}$ (the set is empty if $i > j$). We also use the standard notation $[i]$ to denote the set $[1, i]$.

Given homogeneous polynomials $g, h \in \mathbb{F}\langle X\rangle$ of degrees d_g and d_h respectively and an integer $j \in [0, d_h]$, we define the *j-product of g and h* – denoted $g \times_j h$ – as follows:

- When g and h are monomials, then we can factor h uniquely as a product of two monomials h_1h_2 such that $\deg(h_1) = j$ and $\deg(h_2) = d_h - j$. In this case, we define $g \times_j h$ to be $h_1 \cdot g \cdot h_2$.
- The map is extended bilinearly to general homogeneous polynomials g, h. Formally, let g, h be general homogeneous polynomials, where $g = \sum_\ell g_\ell$, $h = \sum_i h_i$ and g_ℓ, h_i are monomials of g, h respectively. For $j \in [0, d_h]$, each h_i can be factored uniquely into h_{i_1}, h_{i_2} such that $\deg(h_{i_1}) = j$ and $\deg(h_{i_2}) = d_h - j$. And $g \times_j h$ is defined to be $\sum_i \sum_\ell h_{i_1} g_\ell h_{i_2}$.

Note that $g \times_0 h$ and $g \times_{d_h} h$ are just the products $g \cdot h$ and $h \cdot g$ respectively.

2.2 The partial derivative matrix

Here we recall some definitions from [21] and [19]. Let Π denote a partition of $[d]$ given by an ordered pair (Y, Z), where $Y \subseteq [d]$ and $Z = [d] \setminus Y$. In what follows we only use ordered partitions of sets into two parts. We say that such a Π is *balanced* if $|Y| = |Z| = d/2$.

Given a monomial m of degree d and a set $W \subseteq [d]$, we use m_W to denote the monomial of degree $|W|$ obtained by keeping exactly the variables in the locations indexed by W.

▶ **Definition 1** (Partial Derivative matrix). Let $f \in \mathbb{F}\langle X\rangle$ be a homogeneous polynomial of degree d. Given a partition $\Pi = (Y, Z)$ of $[d]$, we define an $N^{|Y|} \times N^{|Z|}$ matrix $M[f, \Pi]$ with entries from $\mathbb{F}$ as follows: the rows of $M[f, \Pi]$ are labelled by monomials from $\mathcal{M}_{|Y|}(X)$ and the columns by elements of $\mathcal{M}_{|Z|}(X)$. Let $m' \in \mathcal{M}_{|Y|}(X)$ and $m'' \in \mathcal{M}_{|Z|}(X)$; the (m', m'')th entry of $M[f, \Pi]$ is the coefficient in the polynomial f of the unique monomial m such that $m_Y = m'$ and $m_Z = m''$.

We will use the rank of the matrix $M[f, \Pi]$ – denoted $\mathrm{rank}(f, \Pi)$ – as a measure of the complexity of f . Note that since the rank of the matrix is at most the number of rows, we have for any $f \in \mathbb{F}\langle X\rangle$ $\mathrm{rank}(f, \Pi) \le N^{|Y|}$.

▶ **Definition 2** (Relative Rank). Let $f \in \mathbb{F}\langle X\rangle$ be a homogeneous polynomial of degree d. For any $Y \subseteq [d]$, we define the *relative rank of f w.r.t.* $\Pi = (Y, Z)$ – denoted rel-rank(f, Π) – to be

$$\text{rel-rank}(f, \Pi) := \frac{\mathrm{rank}(M[f, \Pi])}{N^{|Y|}}.$$

Fix a partition $\Pi = (Y, Z)$ of $[d]$ and two homogeneous polynomials g, h of degrees d_g and d_h respectively. Let $f = g \times_j h$ for some $j \in [0, d_h]$. This induces naturally defined partitions Π_g of $[d_g]$ and Π_h of $[d_h]$ respectively in the following way. Let $I_g = [j + 1, j + d_g]$ and $I_h = [d] \setminus I_g$. We define $\Pi_g = (Y_g, Z_g)$ such that $Y_g = \{j \in [d_g] \mid Y \text{ contains the } j\text{th smallest element of } I_g\}$; $\Pi_h = (Y_h, Z_h)$ is defined similarly with respect to I_h. Let $|Y_g|, |Z_g|, |Y_h|, |Z_h|$ be denoted d'_g, d''_g, d'_h, d''_h respectively.

In the above setting, we have a simple description of the matrix $M[f, \Pi]$ in terms of $M[g, \Pi_g]$ and $M[h, \Pi_h]$. We use the observation that monomials of degree $|Y| = d'_g + d'_h$ are in one-to-one correspondence with pairs (m'_g, m'_h) of degrees d'_g and d'_h respectively (and similarly for monomials of degree $|Z|$). The following appears in [19].

▶ **Lemma 3** (Tensor Lemma). *Say $f = g \times_j h$ as above. Then, $M[f, \Pi] = M[g, \Pi_g] \otimes M[h, \Pi_h]$.*

▶ **Corollary 4.** *Say $f = g \times_j h$ as above. We have* $\mathrm{rank}(f, \Pi) = \mathrm{rank}(g, \Pi_g) \cdot \mathrm{rank}(h, \Pi_h)$. *In the special case that one of Y_g, Z_g, Y_h, or Z_h is empty, the tensor product is an outer product of two vectors and hence* $\mathrm{rank}(f, \Pi) \le 1$.

We associate any partition $\Pi = (Y, Z)$ with the string in $\{-1, 1\}^d$ that contains a -1 in exactly the locations indexed by Y. Given partitions $\Pi_1, \Pi_2 \in \{-1, 1\}^d$, we now define $\Delta(\Pi_1, \Pi_2)$ to be the Hamming distance between the two strings or equivalently as $|Y_1 \Delta Y_2|$ where $\Pi_1 = (Y_1, Z_1)$ and $\Pi_2 = (Y_2, Z_2)$.

▶ **Proposition 5.** *Let $f \in \mathbb{F}\langle X\rangle$ be homogeneous of degree d and say $\Pi \in \{-1, 1\}^d$. Then,* $\mathrm{rank}(f, \Pi) = \mathrm{rank}(f, -\Pi)$.

Proof. Follows from the fact that $M[f, -\Pi]$ is the transpose of $M[f, \Pi]$. ◀

▶ **Lemma 6** (Distance lemma). *Let $f \in \mathbb{F}\langle X\rangle$ be homogeneous of degree d and say $\Pi_1, \Pi_2 \in \{-1, 1\}^d$. Then,* $\mathrm{rank}(f, \Pi_2) \le \mathrm{rank}(f, \Pi_1) \cdot N^{\Delta(\Pi_1, \Pi_2)}$.

Proof. See the full version [17]. ◀

2.3 Standard definitions related to non-commutative circuits

We consider noncommutative arithmetic circuits that compute polynomials over the ring $\mathbb{F}\langle X\rangle$. These are arithmetic circuits where the children of each $\times$ gate are ordered and the polynomial computed by a $\times$ gate is the product of the polynomials computed by its children, where the product is computed in the given order. Further, unless mentioned otherwise, we allow both $+$ and $\times$ gates to have unbounded fan-in and the $+$ gates to compute arbitrary linear combinations of its inputs (the input wires to the $+$ gate are labelled by the coefficients of the linear combination). A noncommutative formula is a circuit where the underlying directed acyclic graph is a rooted tree. The size of an arithmetic circuit or formula is the number of edges or wires in the circuit (which can be assumed to be at least the number of gates in the circuit).

We always assume that the output gate of the circuit is a $+$ gate (possibly of fan-in 1) and that input gates feed into $+$ gates. We also assume that $+$ and $\times$ gates alternate on any path from the output gate to an input gate (some of these gates can have fan-in 1). Any circuit can be converted to one of this form with at most a constant blow-up in size.

Throughout, our circuits and formulas will be *homogeneous* in the following sense. Define the formal degree of a gate as follows: the formal degree of an input gate is 1, the formal degree of a $+$ gate is the maximum of the formal degrees of its children, and that of a $\times$ gate is the sum of the formal degrees of its children. We say that a circuit is homogeneous if each gate computes a homogeneous polynomial and any gate computing a non-zero polynomial computes one of degree equal to the formal degree of the gate. Note, in particular, that every input node is labelled by a variable only (and not by constants from $\mathbb{F}$).

Homogeneity is *not* a strong assumption on the circuit: it is a standard fact that any homogeneous polynomial of degree d computed by a non-commutative circuit of size s can be computed by a homogeneous circuit of size $O(sd^2)$ [11].

We also consider homogeneous Algebraic Branching Programs (ABPs), defined by Nisan [21] in the non-commutative context. We give here a slightly different definition that is equivalent up to polynomial factors.

Assume that $N = n^2 \cdot d$ for positive $n, d \in \mathbb{N}$ and let $\mathrm{IMM}_{n,d}(X)$ denote the following polynomial in N variables (see, e.g. [22]). Assume X is partitioned into d sets of variables $X_1, \ldots, X_d$ of size n^2 each and let $M_1, \ldots, M_d$ be $n \times n$ matrices such that the entries of M_i ($i \in [d]$) are distinct variables in X_i. Let $M = M_1 \cdot M_2 \cdots M_d$; each entry of M is a homogeneous polynomial of degree d from $\mathbb{F}\langle X\rangle$. We define the polynomial $\mathrm{IMM}_{n,d}$ to be the sum of the diagonal entries of M.

A homogeneous ABP for a homogenous polynomial $f \in \mathbb{F}\langle X\rangle$ of degree d is a pair (n_1, ρ) where $n_1 \in \mathbb{N}$ and ρ is a map from $X' = \{x'_1, \ldots, x'_{n_1^2 d}\}$ to homogeneous linear functions from $\mathbb{F}\langle X\rangle$ such that f can be obtained by substituting $\rho(x'_i)$ for each x'_i in the polynomial $\mathrm{IMM}_{n_1,d}(X')$. The parameter n_1 is called the *width* of the ABP.

2.4 Non-commutative circuits with restricted parse trees

In this paper, we study restricted forms of non-commutative arithmetic circuits. The restrictions are defined by the way the circuits are allowed to multiply variables to compute a monomial. To make this precise we need the notion of a parse tree of a circuit, which has been considered in many previous works [13, 1, 20, 18].

Fix a homogeneous non-commutative circuit C. A *parse formula* of C is a *formula* C' obtained by making copies of gates in C as follows:

- Corresponding to the output $+$ gate of C, we add an output $+$ gate to C',

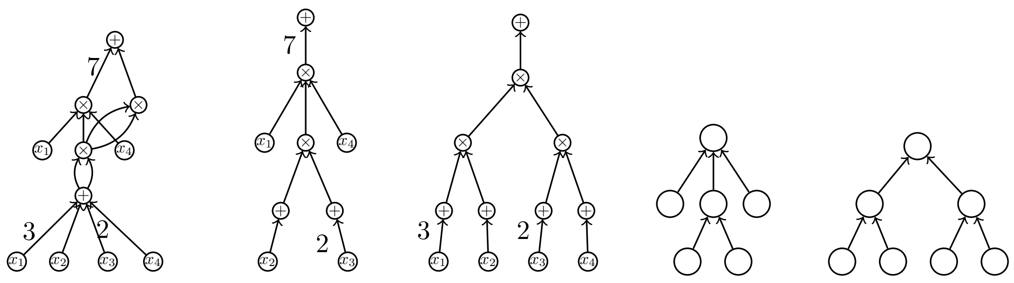

Figure 2 From left to right: a non-commutative arithmetic circuit; two parse formulas in the circuit; the corresponding parse trees. (To simplify the picture, we have not depicted the edges that carry the constant 1. Also we have not introduced + gates between the two layers of × gates; the reader should assume that the edges between the two layers carry + gates of fan-in 1.)

- For every + gate Φ' added to C' corresponding to a + gate Φ in C, we choose exactly one child Ψ of Φ in C and add a copy Ψ' to C' as a child of Φ'. The constant along the wire from Ψ' to Φ' remains the same as in C.
- For every × gate Φ' added to C' corresponding to a × gate Φ in C and *every wire* from a child Ψ to Φ in C, we make a copy of Ψ' to C' and make it a child of Φ.

Any such parse formula C' computes a *monomial* (with a suitable coefficient) and the polynomial computed by C is the sum of all monomials computed by parse formulas C' of C. We define $\mathit{val}(C')$ to be the monomial computed by C'.

A parse tree of C is a rooted, ordered tree obtained by taking a parse formula C' of C, "short circuiting" the + nodes (i.e. we remove the + nodes and connect the edges that were connected to it directly), and deleting all labels of the nodes and the edges of the tree. See Figure 2 for an example. Note that in a homogeneous circuit C, each such tree has exactly d leaves. We say that the tree T is the *shape* of the parse formula C'.

The process that converts the parse formula C' to T associates each internal node of T with a multiplication gate of C' and each leaf of T with an input gate of C'.

Let T be a parse tree of a homogeneous circuit C with d leaves. Given a node $v \in V(T)$, we define the $\deg(v)$ to be the number of leaves in the subtree rooted at v and $\mathrm{pos}(v) :=$ $(1+$the number of leaves preceding v in an in-order traversal of $T)$. The type of v is defined to be $\mathrm{type}(v) := (\deg(v), \mathrm{pos}(v))$. (The reason for this definition is that in any parse formula C' of shape T, the monomial computed by the multiplication gate or input gate corresponding to v in C' computes a monomial of degree $\deg(v)$ which sits at position $\mathrm{pos}(v)$ w.r.t. the monomial computed by the circuit C'.) We also use $\mathcal{I}(T)$ to denote the set of internal nodes of T and $\mathcal{L}(T)$ to denote the set of leaves of T.

We use $\mathcal{T}(C)$ to denote the set of parse trees that can be obtained from parse formulas of C. We say that a homogeneous non-commutative arithmetic circuit is a *Unique Parse Tree circuit* (or *UPT circuit*) if $|\mathcal{T}(C)| = 1$. More generally if $|\mathcal{T}(C)| \leq k$, we say that C is *k-PT*. Finally, if $\mathcal{T}(C) \subseteq \mathcal{T}$ for some family $\mathcal{T}$ of trees, we say that C is *$\mathcal{T}$-PT*. Similarly, we also define UPT formulas, k-PT formulas and $\mathcal{T}$-PT formulas. If C be a UPT circuit with $\mathcal{T}(C) = \{T\}$, we say that T is the *shape* of the circuit C.

We say that a UPT circuit C is in *normal form* if we can associate with each gate Φ of the circuit a node $v(\Phi) \in V(T)$ such that the following holds: if Φ is an input gate, then $v(\Phi)$ is a leaf; if Φ is a × gate with children $\Psi_1, \ldots, \Psi_t$ (in that order), then the nodes $v(\Psi_1), \ldots, v(\Psi_t)$ are the children of $v(\Phi)$ (in that order); and finally, if Φ is a + gate with children $\Psi_1, \ldots, \Psi_t$ (which are all × or input gates since we assume that + and × gates are alternating along each input to output path), then $v(\Phi) = v(\Psi_1) = \cdots = v(\Psi_t)$. (Intuitively, what this means is that in any unravelling of a parse formula containing a (multiplication or input) gate Φ to get the parse tree T, the gate Φ always takes the position of node $v(\Phi)$.)

We state below some simple structural facts about UPT circuits. (See [17] for proof.)

▶ **Proposition 7.**

1. *Let C be a UPT* formula. *Then C is in normal form.*
2. *For any UPT circuit C of size s and shape T, there is another UPT circuit C' of size $O(s^2)$ and shape T in normal form computing the same polynomial as C. Further, given C and T, such a C' can be constructed in time* poly(s).

Let C be either a UPT formula or a UPT circuit of shape T in normal form. We say that a $+$ gate Φ in C is a $(v,+)$ gate if $v(\Phi) = v$. Similarly, we refer to a $\times$ gate Φ in C as a $(v,\times)$ gate if $v(\Phi) = v$. For simplicity of notation, we also refer to an *input* gate Φ as a $(v,\times)$ gate if $v(\Phi) = v$. Note that the output gate is a $(v_0,+)$ gate where v_0 is the root of T.

We now observe that any UPT formula or circuit in normal form can be converted to another (of a possibly different shape) where each multiplication gate has fan-in at most 2.

▶ **Lemma 8.** *Let C be a normal form UPT circuit (resp. formula) of size s and shape T. Then there is a tree T' and normal form UPT circuit (resp. formula) C' of size $O(s)$ and shape T' such that C' computes the same polynomial as C and every multiplication gate in C' has fan-in at most 2. (This implies that every internal node of T' also has fan-in at most 2.) Further, there is a deterministic polynomial-time algorithm, which when given C, computes C' as above.*

Proof. See the full version [17]. ◀

Let C be a UPT circuit of shape T computing a homogeneous polynomial f of degree d. Given any node $u \in V(T)$, we define partition Π_u of $[d]$ so that $\Pi_u = (Y_u, Z_u)$ where $Y_u = \{\text{pos}(v) \mid v \text{ a leaf and descendant of } u\}$.

We will need the following lemma of Lagarde et al. [18].

▶ **Lemma 9** ([18]). *Let C be a normal form UPT circuit of size s computing a homogeneous polynomial $f \in \mathbb{F}\langle X\rangle$ of degree d. Assume that the fan-in of each multiplication gate is bounded by 2. Then, for any $u \in V(T)$, $\text{rank}(f, \Pi_u) \leq s$, where Π_u is as defined above.*

2.5 A polynomial that is full rank w.r.t. all partitions

The following was shown in [19].

▶ **Theorem 10.** *For any even d and any positive $N \in \mathbb{N}$, there is a $q_0(N,d)$ such that the following holds over any field of size at least $q_0(N,d)$. There is an explicit homogeneous polynomial $F_{N,d} \in \mathbb{F}\langle X\rangle$ of degree d such that for any balanced partition $\Pi = (Y,Z)$ of $[d]$, $\text{rank}(f,\Pi) = N^{d/2}$ (equivalently, $\text{rel-rank}(f,\Pi) = 1$). Further, $F_{N,d}$ can be computed by an explicit homogeneous non-commutative arithmetic circuit of size* poly(N,d).

3 Lower bounds for k-PT circuits

In this section, we show that any k-PT circuit computing a polynomial of degree d where k is subexponential in d cannot compute the polynomial $F_{N,d}$ from Theorem 10. We will show that if both k and the size of the circuit are subexponential in d, then there is a Π such that $\text{rel-rank}(f,\Pi) < 1$.

Our proof is based on the following lemmas.

▶ **Lemma 11.** *Let C be a k-PT circuit (resp. formula) of size s with $\mathcal{T}(C) = \{T_1, \ldots, T_k\}$ computing $f \in \mathbb{F}\langle X\rangle$. Then there exist normal form UPT circuits (resp. formulas) $C_1, \ldots, C_k$ of size at most s^2 each such that $\mathcal{T}(C_i) = \{T_i\}$ and $f = \sum_{i=1}^{k} f_i$, where f_i the polynomial computed by C_i.*

Proof. See the full version [17]. ◀

▶ **Lemma 12.** *Let C be a UPT circuit in normal form over $\mathbb{F}\langle X\rangle$ of size $s = N^c$ and f a homogeneous polynomial of degree d computed by C. Let Π be a uniformly random partition of the variables of $[d]$ into two sets. Then for any parameter $b \in \mathbb{N}$,*

$$\Pr_{\Pi}\left[\operatorname{rank}(f,\Pi) \geq N^{d/2-b}\right] \leq \exp(-\Omega(d/(b+c)^2)).$$

Before proving Lemma 12, let us see that the above lemmas imply the following lower bound for homogeneous non-commutative circuits with few parse trees. Note that when the field $\mathbb{F}$ is large enough, this proves a lower bound for $F_{N,d}$ from Theorem 10.

▶ **Theorem 13.** *Assume that $N \geq 2$ is any constant and d an even integer parameter that is growing. Let $F \in \mathbb{F}\langle X\rangle$ be any polynomial such that for each balanced partition Π, $\operatorname{rank}(F,\Pi) = N^{d/2}$. Then, for any constant $\varepsilon \in (0,1)$, any circuit that computes F and satisfies $|\mathcal{T}(C)| = k \leq 2^{d^{\frac{1}{3}-\varepsilon}}$ must have size at least $2^{d^{\frac{1}{3}-\frac{\varepsilon}{2}}}$.*

Proof. Let C be any circuit of size $s \leq N^c$ for $c = d^{1/3-\varepsilon/2}$ with $|\mathcal{T}(C)| = k \leq 2^{d^{1/3-\varepsilon}}$ and computing $f \in \mathbb{F}\langle X\rangle$. We show that there is a balanced partition Π such that $\operatorname{rank}(f,\Pi) < N^{d/2}$. This will prove the theorem.

To show this, we proceed as follows. Using Lemma 11, we can write $f = \sum_{i\in[k]} f_i$ where each $f_i \in \mathbb{F}\langle X\rangle$ is computed by a normal form UPT circuit C_i of size at most $s^2 \leq N^{2c}$.

Fix any $i \in [k]$. By Lemma 12, the number of partitions Π for which $\operatorname{rank}(f_i,\Pi) \geq N^{\frac{d}{2}-c}$ is at most $2^d \cdot \exp(-\Omega(d/c^2))$. In particular, since the number of balanced partitions is $\binom{d}{d/2} = \Theta(\frac{2^d}{\sqrt{d}})$, we see that for a random *balanced* partition Π,

$$\Pr_{\Pi \text{ balanced}}\left[\operatorname{rank}(f_i,\Pi) \geq N^{d/2-c}\right] \leq \sqrt{d}\cdot \exp(-\Omega(d/c^2)) \leq \exp(-d^{1/3}).$$

Say f_i is good for Π if $\operatorname{rank}(f_i,\Pi) \geq N^{d/2-c}$. By the above, we have

$$\Pr_{\Pi \text{ balanced}}\left[\exists i \in [k] \text{ s.t. } f_i \text{ good for } \Pi\right] \leq k \cdot \exp(-d^{1/3}) \leq 2^{d^{1/3-\varepsilon}} \cdot \exp(-d^{1/3}) < 1.$$

In particular, there is a balanced Π such that no f_i is good for Π. Fix such a balanced partition Π. By the subadditivity of rank, we have

$$\begin{aligned}\operatorname{rank}(f,\Pi) &\leq \sum_{i\in[k]} \operatorname{rank}(f_i,\Pi) \leq k \cdot N^{d/2-c} \leq 2^{d^{1/3-\varepsilon}} \cdot N^{d/2-c}\\ &= N^{d/2} \cdot \exp(O(d^{1/3-\varepsilon}) - \Omega(d^{1/3-\varepsilon/2})) < N^{d/2}.\end{aligned}$$

This proves the theorem. ◀

Proof of Lemma 12 . Recall from Section 2.2 that we identify each partition Π with an element of $\{-1,1\}^d$. Given partitions $\Pi_1, \Pi_2 \in \{-1,1\}^d$ we use $\langle \Pi_1, \Pi_2\rangle$ to denote their inner product: i.e., $\langle \Pi_1, \Pi_2\rangle := \sum_{i\in[d]} \Pi_1(i)\Pi_2(i)$. Note that the Hamming distance $\Delta(\Pi_1, \Pi_2)$ is

$$\Delta(\Pi_1,\Pi_2) = \frac{d}{2} - \frac{1}{2}\langle \Pi_1, \Pi_2\rangle. \tag{1}$$

Let $\mathcal{T}(C) = \{T\}$. Recall that $|\mathcal{L}(T)| = d$ and by Lemma 8, we can assume that the fan-in of each internal node of T is bounded by 2. For any $u \in \mathcal{I}(T)$ (recall $\mathcal{I}(T)$ is the set of internal nodes of T), let $\mathcal{L}(u)$ denote the set of leaves of the subtree rooted at u. We identify each leaf $\ell \in V(T)$ with $\mathrm{pos}(\ell) \in [d]$. For each $u \in \mathcal{I}(T)$, we can define the partition Π_u from Section 2.4 by $\Pi_u(\ell) = -1$ iff $\ell \in \mathcal{L}(u)$.

For $\gamma > 0$, define a partition Π to be *γ-correlated to T* if for each $u \in \mathcal{I}(T)$, we have $\left|\sum_{\ell\in\mathcal{L}(u)} \Pi(\ell)\right| \leq \gamma$.

Lemma 12 immediately follows from Claims 14 and 15, stated below.

▶ **Claim 14.** *Let Π be any partition of $[d]$ such that $\mathrm{rank}(f,\Pi) \geq N^{d/2-b}$. Then Π is $O(b+c)$-correlated to T.*

Proof. We know from Lemma 9 and Proposition 5 that for each $u \in \mathcal{I}(T)$, $\mathrm{rank}(f,\Pi_u)$ and $\mathrm{rank}(f,-\Pi_u)$ are at most N^c. If Π is a partition such that either $\Delta(\Pi,\Pi_u)$ or $\Delta(\Pi,-\Pi_u)$ is strictly smaller than $\frac{d}{2} - (b+c)$ for *some* $u \in \mathcal{I}(T)$, then by Lemma 6 we would have $\mathrm{rank}(f,\Pi) < N^{d/2-b}$.

Thus, if $\mathrm{rank}(f,\Pi) \geq N^{d/2-b}$, we must have $\min\{\Delta(\Pi,\Pi_u),\Delta(\Pi,-\Pi_u)\} \geq \frac{d}{2} - (b+c)$ for each $u \in \mathcal{I}(T)$. By (1), this means that for each $u \in \mathcal{I}(T)$, $|\langle\Pi,\Pi_u\rangle| \leq \gamma$ for some $\gamma = O(b+c)$.

Let v be the root of T. Note that $\Pi_v \in \{-1,1\}^d$ is the vector with all its entries being -1. Hence, we have for any $u \in \mathcal{I}(T)$,

$$\left|\sum_{\ell\in\mathcal{L}(u)} \Pi(\ell)\right| = \left|\langle\Pi, \frac{-(\Pi_u+\Pi_v)}{2}\rangle\right| \leq \frac{1}{2}(|\langle\Pi,\Pi_u\rangle| + |\langle\Pi,\Pi_v\rangle|) \leq O(\gamma).$$

This proves the claim. ◀

▶ **Claim 15.** *Say $\Pi \in \{-1,1\}^d$ is chosen uniformly at random and $\gamma \leq \sqrt{d}$. Then* $\Pr_\Pi[\Pi$ *is γ-correlated to T*$] \leq \exp(-\Omega(\frac{d}{\gamma^2}))$.

The following technical subclaim is useful for proving Claim 15. (See [17] for proof.)

▶ **Subclaim 16.** *Assume that $r,t \in \mathbb{N}$ such that $rt \leq d/4$. Then we can find a sequence $u_1,\ldots,u_r \in \mathcal{I}(T)$ such that for each $i \in [r]$ we have $|\mathcal{L}(u_i) \setminus \bigcup_{j=1}^{i-1}\mathcal{L}(u_j)| \geq t$.*

Proof of Claim 15. We apply Subclaim 16 with $t = \Theta(\gamma^2)$ and $r = \Theta(d/\gamma^2)$ to get a sequence $u_1,\ldots,u_r \in \mathcal{I}(T)$ such that for each $i \in [r]$, we have $|\mathcal{L}(u_i) \setminus \bigcup_{j=1}^{i-1}\mathcal{L}(u_j)| \geq t$.

By the definition of γ-correlation, we have

$$\begin{aligned}\Pr_\Pi[\Pi\ \gamma\text{-correlated to } T] &\leq \Pr_\Pi\left[\forall i\in[r], \left|\sum_{\ell\in\mathcal{L}(u_i)}\Pi(\ell)\right| \leq \gamma\right] \\ &\leq \prod_{i\in[r]}\Pr_\Pi\left[\left|\sum_{\ell\in\mathcal{L}(u_i)}\Pi(\ell)\right| \leq \gamma \;\middle|\; \{\Pi(\ell) \mid \ell \in \bigcup_{j<i}\mathcal{L}(u_j)\}\right] \qquad (2)\end{aligned}$$

Fix any $i \in [r]$ and $\Pi(\ell)$ for each $\ell \in \mathcal{L}_{<i} := \bigcup_{j<i}\mathcal{L}(u_j)$. The event $|\sum_{\ell\in\mathcal{L}(u_i)}\Pi(\ell)| \leq \gamma$ is equivalent to $\sum_{\ell\in\mathcal{L}(u_i)\setminus\mathcal{L}_{<i}}\Pi(\ell) \in I$ for some interval I of length $2\gamma = O(\sqrt{t})$. This is the probability that the sum of at least t $\{-1,1\}$-valued random variables chosen i.u.a.r. lies in an interval of length $O(\sqrt{t})$. By the Central Limit theorem, this is at most $1-\Omega(1)$. By (2), we get $\Pr_\Pi[\Pi\ \gamma\text{-correlated to } T] \leq \exp\{-\Omega(r)\}$, which proves the claim. ◀

4 Other results

We refer the reader to the full version of the paper [17] for proofs of the results stated below.

Lower bounds for circuits with rotations of one parse tree. Given two parse trees T_1 and T_2 with the same number of leaves, we say that T_1 is a *rotation* of T_2, denoted $T_1 \sim T_2$, if T_1 can be obtained from T_2 by repeatedly reordering the children of various nodes in T_2. Clearly, $\sim$ is an equivalence relation. We use $[\![T]\!]$ to denote the equivalence class of tree T. We say that a homogeneous circuit C is *rotation UPT or rotUPT* if there is a tree T such that $\mathcal{T}(C) \subseteq [\![T]\!]$. We can show the following result.

▶ **Theorem 17.** *Let $N, d \in \mathbb{N}$ be parameters with d even. Let C be a rotUPT circuit of size s computing a polynomial $f \in \mathbb{F}\langle X\rangle$ of degree d over N variables, then there exists a partition $\Pi = \Pi_C$ s.t. $\mathrm{rel\text{-}rank}(f, \Pi)$ is at most $\mathrm{poly}(s) \cdot N^{-\Omega(d)}$. In particular, if $|\mathbb{F}| > q_0(N, d)$ where $q_0(N, d)$ is as in Theorem 10, any rotUPT circuit for $F_{N,d}$ has size $N^{\Omega(d)}$.*

Separation between Few PT formulas and ABPs. We now state two lower bounds for formulas against $\mathrm{IMM}_{n,d}$, yelding separations with ABPs.

We define the $\times$-*depth* of a formula to be the maximum number of $\times$-gates that one can meet on a path from the root to a leaf. Note that if a formula has alternating $+$ and $\times$ gates on each path and has depth Δ' and $\times$-depth Δ, then $\Delta' \geq \Delta \geq \lceil \frac{\Delta'}{2} \rceil$. We will state our first lower bound in terms of $\times$-depth.

▶ **Theorem 18.** *Let F be a UPT formula of $\times$-depth Δ, size s, computing $\mathrm{IMM}_{n,d} \in \mathbb{F}\langle X\rangle$. Then, $s \geq n^{\Omega(\Delta d^{1/\Delta})}$. In particular, any UPT formula for $\mathrm{IMM}_{n,d}$ must have size $n^{\Omega(\log d)}$.*

This lower bound is actually tight for every $\times$-depth Δ, since the standard divide and conquer approach to computing $\mathrm{IMM}_{n,d}$ gives in fact a UPT formula of size $n^{O(\Delta d^{1/\Delta})}$ and $\times$-depth Δ, for any $\Delta \leq \log d$.

We can also prove a lower bound on the size of k-PT formulas computing $\mathrm{IMM}_{n,d}$ as long as k is significantly smaller than 2^d and $d \leq \log n$.

▶ **Theorem 19.** *Let n, d be growing parameters with $d \leq \log n$. Then, any k-PT formula F computing $\mathrm{IMM}_{n,d}$ has size at least n^ℓ where $\ell = \Omega(\lg d - \lg\lg k)$. In particular, if $k = 2^{o(d)}$, the $\mathrm{size}(F) \geq n^{\omega(1)}$ and if $k = 2^{d^{1-\Omega(1)}}$, then $\mathrm{size}(F) \geq n^{\Omega(\log d)}$.*

Deterministic PIT for UPT and *k*-PT circuits. We now state two results regarding deterministic whitebox PIT algorithms for UPT and k-PT circuits. The first result was already known in characteristic 0 via the result of Lagarde et al. [18]. However, the algorithm we give, adapting the work of Raz and Shpilka [24], works over fields of any characteristic and runs in time polynomial in the size of the circuit. The second result uses additionally the ideas of Gurjar et al. [10] to extend the above algorithm to a deterministic PIT for sums of k UPT circuits; the algorithm runs in polynomial time as long as k is a constant.

▶ **Theorem 20** (PIT for UPT circuits). *Let $N, s \in \mathbb{N}$ be parameters. There is a deterministic algorithm running in time $\mathrm{poly}(s)$ which, on input a UPT circuit C of size at most s over N variables, checks if C computes the zero polynomial or not.*

▶ **Theorem 21** (PIT for sums of k UPT circuits). *Let $N, s, k \in \mathbb{N}$ be parameters. There is a deterministic algorithm running in time $s^{O(2^k)}$ which, on input k UPT circuits $C_1, \ldots, C_k$ (of possibly differing shapes) each of of size at most s over N variables, checks if $\sum_{i=0}^{k} C_i$ computes the zero polynomial or not.*

References

1 Eric Allender, Jia Jiao, Meena Mahajan, and V. Vinay. Non-commutative arithmetic circuits: Depth reduction and size lower bounds. *Theor. Comput. Sci.*, 209(1-2):47–86, 1998. doi:10.1016/S0304-3975(97)00227-2.

2 Vikraman Arvind, Pushkar S. Joglekar, Partha Mukhopadhyay, and S Raja. Identity testing for +-regular noncommutative arithmetic circuits. *Electronic Colloquium on Computational Complexity (ECCC)*, 23:193, 2016. URL: http://eccc.hpi-web.de/report/2016/193.

3 Vikraman Arvind, Pushkar S. Joglekar, and Srikanth Srinivasan. Arithmetic circuits and the hadamard product of polynomials. In Ravi Kannan and K. Narayan Kumar, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2009, December 15-17, 2009, IIT Kanpur, India*, volume 4 of *LIPIcs*, pages 25–36. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2009. doi:10.4230/LIPIcs.FSTTCS.2009.2304.

4 Vikraman Arvind, Partha Mukhopadhyay, and S Raja. Randomized polynomial time identity testing for noncommutative circuits. *Electronic Colloquium on Computational Complexity (ECCC)*, 23:89, 2016. URL: http://eccc.hpi-web.de/report/2016/089.

5 Vikraman Arvind and S. Raja. The complexity of two register and skew arithmetic computation. *Electronic Colloquium on Computational Complexity (ECCC)*, 21:28, 2014. URL: http://eccc.hpi-web.de/report/2014/028.

6 Steve Chien, Lars Eilstrup Rasmussen, and Alistair Sinclair. Clifford algebras and approximating the permanent. *J. Comput. Syst. Sci.*, 67(2):263–290, 2003. doi:10.1016/S0022-0000(03)00010-2.

7 Steve Chien and Alistair Sinclair. Algebras with polynomial identities and computing the determinant. In *45th Symposium on Foundations of Computer Science (FOCS 2004), 17-19 October 2004, Rome, Italy, Proceedings*, pages 352–361, 2004. doi:10.1109/FOCS.2004.9.

8 Hervé Fournier, Nutan Limaye, Guillaume Malod, and Srikanth Srinivasan. Lower bounds for depth-4 formulas computing iterated matrix multiplication. *SIAM J. Comput.*, 44(5):1173–1201, 2015. doi:10.1137/140990280.

9 Ankit Gupta, Pritish Kamath, Neeraj Kayal, and Ramprasad Saptharishi. Approaching the chasm at depth four. In *Proceedings of the Conference on Computational Complexity (CCC)*, 2013.

10 Rohit Gurjar, Arpita Korwar, Nitin Saxena, and Thomas Thierauf. Deterministic identity testing for sum of read-once oblivious arithmetic branching programs. In *30th Conference on Computational Complexity, CCC 2015, June 17-19, 2015, Portland, Oregon, USA*, pages 323–346, 2015. doi:10.4230/LIPIcs.CCC.2015.323.

11 Pavel Hrubeš, Avi Wigderson, and Amir Yehudayoff. Non-commutative circuits and the sum-of-squares problem. *Journal of the American Mathematical Society*, 24(3):871–898, 2011.

12 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.

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

14 Neeraj Kayal, Nutan Limaye, Chandan Saha, and Srikanth Srinivasan. An exponential lower bound for homogeneous depth four arithmetic formulas. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 61–70, 2014. doi:10.1109/FOCS.2014.15.

15 Neeraj Kayal, Chandan Saha, and Ramprasad Saptharishi. A super-polynomial lower bound for regular arithmetic formulas. In David B. Shmoys, editor, *Symposium on Theory*

of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014, pages 146–153. ACM, 2014. doi:10.1145/2591796.2591847.

16 Mrinal Kumar and Shubhangi Saraf. On the power of homogeneous depth 4 arithmetic circuits. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 364–373, 2014. doi:10.1109/FOCS.2014.46.

17 Guillaume Lagarde, Nutan Limaye, and Srikanth Srinivasan. Lower bounds and PIT for non-commutative arithmetic circuits with restricted parse trees. *Electronic Colloquium on Computational Complexity (ECCC)*, 24:77, 2017. URL: https://eccc.weizmann.ac.il/report/2017/077.

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

19 Nutan Limaye, Guillaume Malod, and Srikanth Srinivasan. Lower bounds for non-commutative skew circuits. *Theory of Computing*, 12(1):1–38, 2016. doi:10.4086/toc.2016.v012a012.

20 Guillaume Malod and Natacha Portier. Characterizing valiant's algebraic complexity classes. *J. Complexity*, 24(1):16–38, 2008. doi:10.1016/j.jco.2006.09.006.

21 Noam Nisan. Lower bounds for non-commutative computation (extended abstract). In Cris Koutsougeras and Jeffrey Scott Vitter, editors, *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 410–418. ACM, 1991. doi:10.1145/103418.103462.

22 Noam Nisan and Avi Wigderson. Lower bounds on arithmetic circuits via partial derivatives. *Computational Complexity*, 6(3):217–234, 1997. doi:10.1007/BF01294256.

23 Ran Raz. Multi-linear formulas for permanent and determinant are of super-polynomial size. *J. ACM*, 56(2):8:1–8:17, 2009. doi:10.1145/1502793.1502797.

24 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.

25 Amir Shpilka and Avi Wigderson. Depth-3 arithmetic circuits over fields of characteristic zero. *Computational Complexity*, 10(1):1–27, 2001. doi:10.1007/PL00001609.

26 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. doi:10.1561/0400000039.