Quasipolynomial Hitting Sets for Circuits with Restricted Parse Trees

Ramprasad Saptharishi[1]
Tata Institute of Fundamental Research, Mumbai, India
ramprasad@tifr.res.in

Anamay Tengse[2]
Tata Institute of Fundamental Research, Mumbai, India
tengse.anamay@tifr.res.in

Abstract

We study the class of non-commutative *Unambiguous circuits or Unique-Parse-Tree (UPT) circuits*, and a related model of *Few-Parse-Trees (FewPT)* circuits (which were recently introduced by Lagarde, Malod and Perifel [18] and Lagarde, Limaye and Srinivasan [17]) and give the following constructions:

- An explicit hitting set of *quasipolynomial* size for UPT circuits,
- An explicit hitting set of *quasipolynomial* size for FewPT circuits (circuits with constantly many parse tree shapes),
- An explicit hitting set of *polynomial* size for UPT circuits (of known parse tree shape), when a parameter of *preimage-width* is bounded by a constant.

The above three results are extensions of the results of [2], [10] and [9] to the setting of UPT circuits, and hence also generalize their results in the commutative world from *read-once oblivious algebraic branching programs (ROABPs)* to *UPT-set-multilinear* circuits.

The main idea is to study *shufflings* of non-commutative polynomials, which can then be used to prove suitable depth reduction results for UPT circuits and thereby allow a careful translation of the ideas in [2], [10] and [9].

2012 ACM Subject Classification Theory of computation → Algebraic complexity theory, Theory of computation → Complexity classes

Keywords and phrases Unambiguous Circuits, Read-once Oblivious ABPs, Polynomial Identity Testing, Lower Bounds, Algebraic Circuit Complexity

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2018.6

Related Version A full version of the paper is available at `https://arxiv.org/abs/1709.03068`.

Acknowledgements We thank the organizers of the NMI Workshop on Arithmetic Complexity 2017 where we learned of the circuit classes that we study in this paper. We thank Nutan Limaye and Srikanth Srinivasan for numerous discussions that eventually led to these results. We thank Rohit Gurjar for pointing out a subtlety in a previous draft of this paper, and also thank Amir Shpilka for inviting RS to Tel Aviv University (where this discussion took place). We also thank the anonymous reviewers for their valuable suggestions.

[1] Research supported by Ramanujan Fellowship of DST.
[2] Supported by a fellowship of the DAE.

1 Introduction

The field of algebraic complexity deals with classifying multivariate polynomials based on their hardness. Typically, the complexity of a polynomial is measured by the size of the smallest circuit computing it (an arithmetic circuit is a directed acyclic graph made up of internal nodes that are labeled with $+$ or $\times$ and leaves labelled with variables or constants from the field; size of the circuit is the number of nodes). The central question in this field is to construct an explicit family of polynomials ($\{\mathrm{Perm}_n\}$ is the top candidate) that requires large arithmetic circuits to compute it. This is also called the "VP vs VNP" question (named after Valiant [26]), and thought of as an algebraic analogue of the "P vs NP" question.

So far, the best lower bound we have for general arithmetic circuits computing an n-variate degree d polynomial is a barely super-linear $\Omega(n \log d)$ lower bound by Baur and Strassen [6]. Recent research has focused on proving lower bounds for restricted classes of circuits, either by bounding the depth of such circuits or by focusing on other syntactic restrictions. One such syntactic restriction is to consider *non-commutative circuits*, where we assume that the underlying variables $x_1, \ldots, x_n$ do not commute. In the non-commutative model, there is an inherent order in which elements are multiplied and this adds restrictions on the way monomials can be computed ($xy \neq yx$ here and hence $x^2 + 2xy + y^2 \neq (x+y)^2 = x^2 + xy + yx + y^2$). It is therefore natural to expect that it should be easier to prove lower bounds in this model.

Nisan [20] introduced the non-commutative model, specifically the non-commutative algebraic branching programs (ABP). In his seminal paper, he showed exponential lower bounds against non-commutative ABPs for the non-commutative versions of the determinant and permanent polynomials (among others). In fact, using his technique, one could even reconstruct the smallest non-commutative ABP given just oracle access to that polynomial (cf. [16])! Although we have exponential lower bounds for non-commutative ABPs, we do not have any non-trivial lower bounds for non-commutative circuits. Hrubeš, Wigderson and Yehudayoff [12] presented an approach via *sum-of-squares* lower bounds but we do not have any non-trivial lower bounds for the class of general non-commutative circuits.

Limaye, Malod and Srinivasan [19] extended Nisan's lower bound to non-commutative skew circuits, which are circuits where every multiplication gate has at most one child that is a non-leaf. Lagarde, Malod and Perifel [18] initiated the study of non-commutative *unambiguous circuits*, or *Unique Parse Tree (UPT)* circuits. These circuits and their generalizations are the main models of study in this paper.

Arvind and Raja [5] also studied lower bounds for various subclasses of commutative set-multilinear circuits. Some of the models they studied include analogues of UPT and FewPT circuits. They proved lower bounds for UPT and FewPT set-multilinear circuits, and also for other subclasses of set-multilinear circuits called *narrow* set-multilinear circuits and *interval* set-multilinear circuits, the latter of which assumes the sum-of-squares conjecture of Hrubeš, Wigderson and Yehudayoff [12].

1.1 The model of study

A parse tree of a circuit is obtained by starting at the root, and at every $+$ gate choosing exactly one child, and at every $\times$ gate choosing all its children (formally defined in Theorem 2.1). Informally, a parse tree of a circuit is basically a *certificate* of computation of a monomial in a circuit. Lagarde, Malod and Perifel [18] introduced a subclass of non-commutative circuits called *Unique Parse Tree (UPT) circuits* or *unambiguous circuits* where all parse trees of the circuit have the same shape (formally defined in Theorem 2.2). The class of non-commutative UPT circuits subsumes the class of non-commutative ABPs as any ABP can be expressed as a left-skew circuit. A related model of *set-depth-*Δ *formulas* was studied by Agrawal, Saha and Saxena [3] that is a subclass of UPT circuits where the underlying parse trees are extremely regular[3].

Lagarde, Malod and Perifel [18] extended the techniques of Nisan [20] to give exponential lower bounds for UPT circuits. Subsequently, Lagarde, Limaye and Srinivasan [17] extended the lower bounds to the class of circuits with parse trees of not-too-many shapes (at most $2^{o(n)}$ shapes).

1.2 Polynomial identity testing

A *Polynomial Identity Test (PIT)* is an algorithm that, given a circuit as input, checks if the circuit is computing the zero polynomial or not. The standard Ore-DeMillo-Lipton-Schwartz-Zippel lemma [22, 7, 24, 28] provides a simple randomized algorithm but the goal is to construct an efficient deterministic PIT. A stronger test is what is called a *black-box PIT* where we are only provided evaluation access to the circuit. Hence, a black-box PIT is essentially equivalent to constructing a *hitting set*, i.e., a set of points (or matrices, in the non-commutative case) $\mathcal{H}$ such that every non-zero polynomial from the class of interest is guaranteed to evaluate to a nonzero value on some element $\mathbf{a} \in \mathcal{H}$. PITs that use the structure of the circuit are called *white-box* PITs.

The task of constructing efficient PITs is intimately connected to the task of proving lower bounds [11, 15, 1]. Once we have a lower bound for a class $\mathcal{C}$, it is natural to ask if we can also construct efficient PITs for that class. Raz and Shpilka [23] gave the first deterministic polynomial time white-box PIT for the class of non-commutative ABPs. Forbes and Shpilka [8] gave a quasipolynomial ($n^{O(\log n)}$) size hitting set for non-commutative ABPs. This was achieved by studying a natural commutative analogue of non-commutative ABPs, and this was the class of *Read-Once Oblivious Algebraic Branching Programs (ROABPs)* where the variables are read in a "known order".

The class of ROABPs is interesting in its own right owing to the connection with the "RL vs L" question. In fact, much of the hitting set constructions for ROABPs has been inspired by Nisan's [21] pseudorandom generator for RL (which has seed length $O(\log^2 n)$). As mentioned earlier, Forbes and Shpilka gave a hitting set of size $n^{O(\log n)}$ for polynomial sized ROABPs when the order in which variables are read was known. Agrawal, Gurjar, Korwar and Saxena [2] presented a different hitting set for the class of commutative ROABPs that did not need the knowledge of the order in which the variables were read. Subsequently, Gurjar, Korwar, Saxena and Thierauf [10] studied polynomials that can be computed as a sum of constantly many ROABPs (of possibly different orders) and presented a polynomial time white-box PIT, and also a quasipolynomial time black-box PIT for this class.

Lagarde, Malod and Perifel [18], besides presenting lower bounds for non-commutative UPT circuits, also gave a polynomial time white-box PIT for this class. This was extended by Lagarde, Limaye and Srinivasan [17] to a white-box algorithm for non-commutative circuits with constantly many parse tree shapes (analogous to the result of [10]). The question of constructing black-box PITs was left open by them, and we answer this in our paper.

[3] the formula is levelled, and all nodes at a level have the same fan-in

1.3 Our results

Polynomial Identity Testing

Our main results are hitting sets for the class of polynomials computed by UPT circuits and related classes.

▶ **Theorem 1.1** (Hitting sets for UPT circuits). *There is an explicit hitting set $\mathcal{H}_{d,n,s}$ of at most $(snd)^{O(\log d)}$ size for the class of degree d n-variate homogeneous non-commutative polynomials in $\mathbb{F}\langle x_1, \ldots, x_n\rangle$ that are computed by UPT circuits of size at most s.*

This result builds on the technique of *basis isolating weight assignments* introduced by [2] for constructing hitting sets for ROABPs. Furthermore, we can also extend the hitting set to the class of non-commutative circuits that have *few shapes* (analogous to [10]'s hitting set for sums of few ROABPs).

▶ **Theorem 1.2** (Hitting sets for circuits with few parse tree shapes). *There is an explicit hitting set $\mathcal{H}_{d,n,s,k}$ of size at most $(s^{2^k} nd)^{O(\log d)}$ for the class of n-variate degree d homogeneous non-commutative polynomials in $\mathbb{F}\langle x_1, \ldots, x_n\rangle$ that are computed by non-commutative circuits of size at most s consisting of parse trees of at most k shapes.*

Both the above theorems are fully black-box in the sense that it is not required to know the underlying shape(s). For the case of non-commutative ABPs (and more generally, ROABPs in a known order), Gurjar, Korwar and Saxena [9] presented a more efficient hitting set when the width of the ABP is small. For UPT circuits, there is a natural notion of *preimage-width* of a UPT circuit (formally defined in Theorem 2.3) that corresponds to the notion of width of an ABP. We show an analogue of the hitting set of Gurjar, Korwar and Saxena for the class of UPT circuits of small *preimage-width* if the underlying shape of the parse trees is known.

▶ **Theorem 1.3** (Hitting sets for known-shape low-width UPT circuits). *Let $\mathcal{C}_{n,d,T,w}$ be the class of n-variate degree d non-commutative polynomials that are computable by UPT circuits of preimage-width at most w and underlying parse-tree shape as T. Over any field of zero or large characteristic, there is an explicit hitting set $\mathcal{H}_{n,d,T,w}$ of size $w^{O(\log d)}\operatorname{poly}(nd)$ for $\mathcal{C}_{n,d,T,w}$.*

These hitting sets also translate to the natural commutative analogues of *UPT set-multilinear circuits* etc. (formally defined in Theorem 5.1).

Structural results

If f is a non-commutative polynomial of degree d and if $\sigma \in S_d$ is a permutation on d letters, we define the *shuffling* of f by σ (denoted by $\Delta_\sigma(f)$) as the natural operation of permuting each *word* of f according to σ.

The three PIT statements stated above begin with the following depth reduction statement about UPT circuits.

▶ **Theorem 1.4** (Depth reduction for UPT circuits). *Let f be an n-variate degree d polynomial that is computable by a UPT circuit of preimage-width w. Then, there is some $\sigma \in S_d$ such that $\Delta_\sigma(f)$ can be computed by a UPT circuit of $O(\log d)$ depth and preimage-width $O(w^2)$.*

The above theorem implies that $\Delta_\sigma(f)$ is computable by an ABP of quasipolynomial size. We also show that this blow-up of quasipolynomial size is tight.

▶ **Theorem 1.5** (Separating UPT circuits and ABPs, under shuffling). *There is an explicit n-variate degree d non-commutative polynomial f that is computable by UPT circuits of preimage-width $w = \mathrm{poly}(n,d)$ such that for every $\sigma \in S_d$, the polynomial $\Delta_\sigma(f)$ requires non-commutative ABPs of size $(nd)^{\Omega(\log nd)}$ to compute it.*

We also extend the lower bound of [18] to give a polynomial computed by a *skew circuit* that requires exponential sized UPT circuits under any shuffling. Details can be found in the full version.

1.4 Proof ideas

As mentioned, the starting point of all these results is the depth reduction. From a result of Nisan [20], the palindrome polynomial Pal_d is known to require ABPs of size $2^{\Omega(d)}$ even though it can be computed by a polynomial sized UPT circuit. Therefore, Pal_d cannot be computed by a circuit of depth $o(d/\log d)$. The key insight here is that even though Pal_d cannot be computed by small depth non-commutative circuits, a shuffling of the palindrome is

$$\sum_{w_1,\ldots,w_d \in [n]} x_{w_1}x_{w_1}x_{w_2}x_{w_2}\cdots x_{w_d}x_{w_d} = \prod_{i=1}^{d}\left(x_1x_1 + \cdots + x_nx_n\right),$$

which is of course computable by an $O(\log d)$ depth UPT formula even. Hence we attempt to reduce the depth under a suitable shuffling.

In order to establish the depth reduction (Theorem 1.4) we follow the strategy of Valiant, Skyum, Berkowitz and Rackoff [27] and Allender, Jiao, Mahajan and Vinay [4] but make use of the UPT structure (work with different *frontier nodes* and *gate quotients*) based on the underlying shape of the parse trees. It was pointed out to us that the key ideas in our proof of depth reduction were used by Arvind and Raja ([5]) for a commutative analogue of UPT circuits.

This depth reduction immediately yields that there is a quasipolynomial sized ABP computing a shuffling of f. We show that this blow-up is tight (Theorem 1.5) by essentially following the proof of Hrubeš and Yehudayoff [13] to separate monotone ABPs and monotone circuits in the commutative world.

In order to obtain hitting sets for UPT circuits, one could potentially just use the fact that there is a quasipolynomial sized ABP computing a shuffling of f and just use the known hitting sets for non-commutative ABPs [8] to obtain a hitting set of $\mathrm{poly}(ndw)^{O(\log^2 d)}$. However, we directly work with the UPT circuit and lift the technique of *basis isolating weight assignments* of Agrawal, Gurjar, Korwar and Saxena [2] to this more general setting to obtain Theorem 1.1. Theorem 1.3 is an easy generalization of the ideas of Gurjar, Korwar and Saxena [9] once we observe that the depth reduction keeps the preimage-width small.

Theorem 1.2 essentially follows the same ideas of Gurjar, Korwar, Saxena and Thierauf [10]. The techniques of [10] are general enough that once a circuit class has a *characterizing set of dependencies* and a *basis isolating weight assignment*, there is a natural method to lift the techniques to work with the sum of few elements from this class. [10] use this for ROABPs and we use this for UPT circuits.

To summarize, once we obtain the depth reduction, much of the results in this paper is a careful translation of prior work of [13], [2], [10], [9] to the setting of UPT (or FewPT) circuits. Consequently, this also generalizes the hitting sets of [2, 10, 9] from ROABPs to *UPT (or FewPT) set-multilinear* circuits. Such a generalization was unknown prior to this work.

2 Preliminaries

2.1 Notation

- We use $\mathbb{F}\langle x_1, \ldots, x_n \rangle$ to refer to the ring of polynomials in non-commuting variables $\{x_1, \ldots, x_n\}$. For a parameter d, we use $\mathbb{F}\langle x_1, \ldots, x_n \rangle_{\deg = d}$ to refer to the set of polynomials in $\mathbb{F}\langle x_1, \ldots, x_n \rangle$ that are homogeneous and of degree d. Similarly, the set of polynomials of degree at most d will be denoted by $\mathbb{F}\langle x_1, \ldots, x_n \rangle_{\deg \leq d}$.
- We use boldface letters $\mathbf{x}$ and $\mathbf{y}$ to denote sets of variables (the number of variables would be clear from context). We shall also use $[d]$ to refer to the set $\{1, 2, \ldots, d\}$.
- The paper will sometimes shift between the commutative and the non-commutative domains. We use $\mathbf{x}$ whenever we are talking about non-commutative variables, and $\mathbf{y}$, $\mathbf{z}$ for variables in the commutative domain.

2.2 Basic definitions

UPT and FewPT circuits

▶ **Definition 2.1** (Parse trees). A parse tree T of a circuit C is a tree obtained as follows:
- the root of C is the root of T,
- if $v \in T$ is a $\times$ gate, then all the children of v in C are the children of v in T in the same order,
- if $v \in T$ is a $+$ gate, then exactly one child of v in C is a child of v in T.

Gates are replicated to ensure that T is a tree. The value of the parse tree T, denoted by $[T]$, is just the product of the leaf labels in T.

Intuitively, a parse tree is a *certificate* that a monomial was produced in the computation of C (though it could potentially be canceled by other parse trees computing the same monomial). Therefore, if f is the polynomial computed by C, then

$$f = \sum_{T \text{ is a parse tree}} [T].$$

▶ **Definition 2.2.** (UPT and FewPT circuits) A circuit C computing a homogeneous polynomial is said to be a *Unique Parse Tree (UPT) circuit* if all parse trees of C have the same shape (that is, they are identical except perhaps for the gate names).

A circuit C that computes a homogeneous polynomial is said to be a FewPT(k) circuit if the parse trees of C have at most k distinct shapes.

▶ **Definition 2.3** (Preimage-width). Suppose C is a UPT circuit and say T is the shape of the underlying parse trees. For a node $\tau \in T$ and a gate $g \in C$, we shall say that g is a *preimage* of τ, denoted by $g \sim \tau$, if and only if there is some parse tree T' of C where the gate g appears in position τ.

The *preimage-width* of a UPT circuit C is the largest size of preimages of any node $\tau \in T$. That is,

$$\text{preimage-width}(C) = \max_{\tau \in T} |\{g \in C \ : \ g \sim \tau\}| .$$

It is clear that if C is a UPT circuit of preimage-width w computing a homogeneous degree d polynomial, then the size of C is at most dw. The preimage-width of a UPT circuit is a more useful measure to study than the size of the circuit. A simple concrete example of this is that the standard conversion of homogeneous ABPs to homogeneous circuits in fact yields UPT circuits. Furthermore, the width of the ABP is directly related to the preimage-width of the resulting UPT circuit.

▶ **Observation 2.4.** *If f is computable by a width w homogeneous algebraic branching program, then f can be equivalently computed by UPT circuits of preimage-width w^2.*

$\times_p$-products

▶ **Definition 2.5** ($\times_p$-products)**.** For any $d_1, d_2 \geq 0$ and p satisfying $0 \leq p \leq d_2$, define a map $\times_p : \mathbb{F}\langle x_1, \ldots, x_n\rangle_{\deg=d_1} \times \mathbb{F}\langle x_1, \ldots, x_n\rangle_{\deg=d_2} \to \mathbb{F}\langle x_1, \ldots, x_n\rangle_{\deg=d_1+d_2}$ as the unique bilinear that satisfies

$$x_{w_1} \cdots x_{w_{d_1}} \times_p x_{v_1} \cdots x_{v_{d_2}} = x_{v_1} \cdots x_{v_p} x_{w_1} \cdots x_{w_{d_1}} x_{v_{p+1}} \cdots x_{v_{d_2}}.$$

For instance, the usual multiplication (or concatenation) operation is just $\times_0$.

Shuffling of a polynomial

▶ **Definition 2.6** (Shuffling of a non-commutative polynomial)**.** Let $P_d(x_1, \ldots, x_n)$ be a homogeneous degree d non-commutative polynomial from $\mathbb{F}\langle x_1, \ldots, x_n\rangle_{\deg=d}$. Given any permutation $\sigma \in S_d$ over d-letters, we can define the *shuffling of P_d via σ* as the unique linear map $\Delta_\sigma : \mathbb{F}\langle x_1, \ldots, x_n\rangle_{\deg=d} \to \mathbb{F}\langle x_1, \ldots, x_n\rangle_{\deg=d}$ that is obtained by linearly extending

$$\Delta_\sigma(x_{w_1} \cdots x_{w_d}) = x_{w_{\sigma(1)}} \cdots x_{w_{\sigma(d)}}.$$

2.3 Basic lemmas

Canonical UPT circuits, and types of gates

We shall say that a UPT circuit C with underlying parse tree shape T is *canonical* if for every gate $g \in C$ there is some node $\tau \in T$ such that every parse tree of C involving g has g only in position τ. That is, every gate of the circuit has a unique type associated with it.

▶ **Lemma 2.7** ([18])**.** *Suppose if $f \in \mathbb{F}\langle x_1, \ldots, x_n\rangle$ is a homogeneous, degree d, non-commutative polynomial computed by a non-commutative UPT circuit of preimage-width w. Then, f can be equivalently computed by a canonical UPT circuit of preimage-width w as well.*

For a canonical UPT circuit where the parse trees have shape T, we shall say that g has type τ if $\tau \in T$ is the unique node in T such that $g \sim \tau$.

Fix a $\tau \in T$ and let i be the number of leaves of the subtree rooted at τ, and let p be the number of leaves to the left of τ in the inorder traversal of T. We shall then say that τ (or a gate $g \in C$ of type τ) has *position-type* (i, p). The following lemma allows us to write the polynomial computed by the circuit as a small sum of $\times_p$-products.

▶ **Lemma 2.8** ([18])**.** *Let f be a polynomial computed by a canonical UPT circuit C of preimage-width w and say T is the shape of the underlying parse trees. If $\tau \in T$ with position-type (i, p), then we can write f as*

$$f(\mathbf{x}) = \sum_{r=1}^{w} g_r(\mathbf{x}) \times_p h_r(\mathbf{x}),$$

where $\deg g_r = i$ *and* $\deg h_r = \deg(f) - i$ *for all* $r = 1, \ldots, w$.

3 Depth reduction for UPT circuits

This section shall address Theorem 1.4, which we recall below.

▶ **Theorem 1.4** (Depth reduction for UPT circuits). *Let f be an n-variate degree d polynomial that is computable by a UPT circuit of preimage-width w. Then, there is some $\sigma \in S_d$ such that $\Delta_\sigma(f)$ can be computed by a UPT circuit of $O(\log d)$ depth and preimage-width $O(w^2)$.*

It was pointed out to us that a similar depth reduction was also proved by Arvind and Raja [5]. They showed that a commutative UPT set-multilinear circuit can be depth-reduced to a corresponding quasipolynomial sized $O(\log d)$ depth UPT set-multilinar formula via Hyafil's [14] depth reduction. Using techniques similar to [27], one can directly obtain a polynomial sized UPT circuit of depth $O(\log d)$. Though this can be inferred from the results in [5], we state and prove it in the form needed for the non-commutative setting.

3.1 UPT $\otimes$-circuits

To prove the depth reduction, we will move to an intermediate model of UPT $\otimes$-circuits.

▶ **Definition 3.1** (UPT $\otimes$-circuits). The class of UPT $\otimes$-circuits is a generalization of homogeneous non-commutative circuits in that the internal gates are $+$ gates and $\times_p$ gates instead of the usual $+$ and $\times$ gates. We shall also say that the circuit is *semi-unbounded* if all $\times_p$ gates have fan-in bounded by 2 (with no restriction on $+$ gates).

A *parse tree* for an $\otimes$-circuit is similar to parse trees in a general non-commutative circuit but the internal nodes of the parse tree are labelled by $+$ and $\times_p$ (with the p specified at each gate).

We shall say that an $\otimes$-circuit C is UPT if every parse tree is of the same shape, i.e. two parse trees in C can differ only in the gate names.

To prove Theorem 1.4, we begin by depth reducing the circuit to get an $\otimes$-circuit computing f of $O(\log d)$ depth. We then convert that to a UPT circuit computing a shuffling of f.

▶ **Lemma 3.2** (Depth reducing to $\otimes$-circuits). *Let $f \in \mathbb{F}\langle x_1, \ldots, x_n\rangle$ be a homogeneous degree d polynomial that is computable by a UPT circuit of size s. Then, f can equivalently be computed by a semi-unbounded UPT $\otimes$-circuit of size $O(s^2)$ and depth $O(\log d)$.*

The rough sketch is to follow a similar process as in [27] by defining a suitable notion of a *gate quotient* $[u : v]$ for this setting. The set of *frontier* nodes is different from the previous depth reduction results but Theorem 2.8 allows us to essentially follow the same strategy to obtain the above depth-reduced UPT circuit.

Proof. Let C be the UPT circuit computing $f(x_1, \ldots, x_n)$ and say T is the shape of the parse trees of C. For any node $\tau \in T$, let $\mathcal{F}_\tau$ be the set of all gates in C whose position in T is τ. For two gates $u, v \in C$, we shall say that $u \succeq v$ if the place of u in T is an ancestor of the place of v in T. We shall abuse notation and use $u \succeq \tau$ to mean that u's position in T is an ancestor of $\tau \in T$. For a gate $u \in C$, let $[u]$ refer to the polynomial computed at that gate. Similar to [27, 4], we define inductively the following notion of a *gate quotient* for any pair of gates $u, v \in C$:

$$[u : v] = \begin{cases} 0 & \text{if } u \not\succeq v, \\ 1 & \text{if } u = v, \\ [u_1 : v] + [u_2 : v] & \text{if } u = u_1 + u_2, \\ [u_1 : v] \cdot [u_2] & \text{if } u = u_1 \times u_2 \text{ and } u_1 \succeq v, \\ [u_1] \cdot [u_2 : v] & \text{if } u = u_1 \times u_2 \text{ and } u_2 \succeq v. \end{cases}$$

▶ **Claim 3.3.** *For any $u \in C$, if $\tau \in T$ such that $u \succeq \tau$, then*

$$[u] = \sum_{\substack{w \in C \\ w \sim \tau}} [w] \times_p [u : w] \tag{3.1}$$

for a suitable p depending just on τ and the type of u. Furthermore, suppose $u, v \in C$ with v being a multiplication gate and if $\tau \in T$ such that $u \succeq \tau \succeq v$ then

$$[u : v] = \sum_{\substack{w \in C \\ w \sim \tau}} [w : v] \times_p [u : w]. \tag{3.2}$$

for a suitable p depending just on τ and the type of u and v.

We'll defer this proof to later and first finish the proof of Theorem 3.2. With (3.1) and (3.2), we can construct the $\otimes$-circuit C' for f just as in [27, 4]. The circuit C' would have gates computing each $[u]$ and $[u : v]$ for nodes $u, v \in C$ with $u \succeq v$ and v being a multiplication gate. The wirings in C' is built by appropriate applications of (3.1) and (3.2).

Let $u \in C$ and say $\deg[u] = d_u$. The plan would be to set up the computation in C' so that using an $O(1)$ depth computation, we can compute $[u]$ using gates whose degrees are a constant factor smaller than d_u. Consider any parse tree rooted at u, and starting from u follow the *higher degree* child. Let τ be the last point on the path with degree $\geq d_u/2$ (degree of its children will be $< d_u/2$). Applying (3.1),

$$\begin{aligned} [u] &= \sum_{w \sim \tau} [w] \times_p [u : w] \\ &= \sum_{w \sim \tau} ([w_1] \times [w_2]) \times_p [u : w] && \text{where } w = w_1 \times w_2. \end{aligned}$$

Now observe that each of the terms on the RHS, $[u : w], [w_1], [w_2]$ have degree at most $d_u/2$, as we wanted. Furthermore, each coordinate of tuple $([u : w], [w_1], [w_2])$ are all of the same *type* as we run over all $w \sim \tau$.

We now need to show how to compute $[u : v]$ for a pair $u \succ v$. Say $\deg[u] = d_u$ and $\deg[v] = d_v$. For this, start with some parse tree rooted at u and walk down the path leading to the place of v, and let τ be the last point on this path such that $\deg \tau \geq \frac{d_u + d_v}{2}$. Using (3.2),

$$\begin{aligned} [u : v] &= \sum_{w \sim \tau} [w : v] \times_p [u : w] \\ &= \sum_{w \sim \tau} ([w_1] \times [w_2 : v]) \times_p [u : w] \end{aligned}$$

where $w = w_1 \times w_2$ and $w_2 \succeq v$ (the other possibility is identical). By the choice of τ, we have $\deg[u : w], \deg[w_2 : v] \leq \frac{d_u - d_v}{2}$. However, the best bound we can give on $\deg[w_1]$

is $d_u - d_v$. Nevertheless, we can apply (3.1) again on $[w_1]$ by finding a suitable $\tau' \prec w_1$ satisfying $\deg \tau' \geq \frac{\deg w_1}{2}$ and write

$$\begin{aligned}
[u : v] &= \sum_{w \sim \tau} ([w_1] \times [w_2 : v]) \times_p [u : w] \\
&= \sum_{w \sim \tau} \left(\left(\sum_{w' \sim \tau'} [w'] \times_{p'} [w_1 : w'] \right) \times [w_2 : v] \right) \times_p [u : w] \\
&= \sum_{w \sim \tau} \sum_{w' \sim \tau'} ((([w_1'] \times [w_2']) \times_{p'} [w_1 : w']) \times [w_2 : v]) \times_p [u : w]
\end{aligned}$$

By the choice of τ and τ', each of the *factors* on the RHS have degree at most $\frac{(d_u - d_v)}{2}$ as we wanted. Furthermore, once again, all of the summands consists of similarly typed factors.

This naturally yields an $\otimes$-circuit computing f of depth $O(\log d)$ and size $O(s^2)$. Since all summands consist of similarly typed factors, it follows that the circuit is UPT as well. ◀

Proof of Claim 3.3. The proof is by induction. As a base case, suppose $u \sim \tau$. Then, $[u]$ is just the sum of the values of parse trees. Some of the parse trees use u. Of all nodes $w \in C$ such that $w \sim \tau$, only $[u : u] = 1$ and every other $[u : w] = 0$. Therefore, clearly $[u] = \sum_{w \sim \tau} [w] \cdot [u : w]$.

Now suppose $u \succ \tau$ and say we already know that $[u'] = \sum_{w \sim \tau} [w] \times_p [u' : w]$ for every $u \succ u' \succeq \tau$. If $u = u_1 + u_2$, then

$$\begin{aligned}
[u] &= [u_1] + [u_2] \\
&= \left(\sum_{w \sim \tau} [w] \times_p [u_1 : w] \right) + \left(\sum_{w \sim \tau} [w] \times_p [u_2 : w] \right) \\
&= \sum_{w \sim \tau} [w] \times_p ([u_1 : w] + [u_2 : w]) \\
&= \sum_{w \sim \tau} [w] \times_p [u : w].
\end{aligned}$$

Similarly, suppose $[u] = [u_1] \times [u_2]$. We have two cases depending on whether $u_1 \succeq \tau$ or $u_2 \succeq \tau$.

If $u_1 \succeq \tau$, then

$$\begin{aligned}
[u] &= [u_1] \times [u_2] \\
&= \left(\sum_{w \sim \tau} [w] \times_p [u_1 : w] \right) \times [u_2] \\
&= \sum_{w \sim \tau} [w] \times_p ([u_1 : w] \times [u_2]) \\
&= \sum_{w \sim \tau} [w] \times_p [u : w].
\end{aligned}$$

If $u_2 \succeq \tau$, then

$$\begin{aligned}
[u] &= [u_1] \times [u_2] \\
&= [u_1] \times \left(\sum_{w \sim \tau} [w] \times_p [u_2 : w] \right) \\
&= \sum_{w \sim \tau} [w] \times_{p + \deg u_1} ([u_1] \times [u_2 : w]) \\
&= \sum_{w \sim \tau} [w] \times_{p + d_1} [u : w].
\end{aligned}$$

Essentially the same proof works for (3.2) as well. ◀

▶ **Lemma 3.4** ($\otimes$-circuits to circuits for a shuffling). *Let $f \in \mathbb{F}\langle x_1, \ldots, x_n \rangle$ be a homogeneous degree d polynomial that is computable by a UPT $\otimes$-circuit C' of size s. Consider the circuit C'' obtained by replacing all $\otimes$ gates in C' by $\times$ gates. Then, C'' computes $\Delta_\sigma(f)$ for some $\sigma \in S_d$.*

Proof. We shall prove this by induction. We need a slightly stronger inductive hypothesis which is that the choice of permutation σ depends only on the shape of the parse trees in C'.

Say u is the root of C'. Suppose u is a $+$ gate and say $u = u_1 + u_2 + \cdots + u_r$. If $u' = u_1' + \cdots + u_r'$ is the resulting computation in C'' then by the inductive hypothesis, we know that there is a $\sigma \in S_d$ such that $[u_i'] = \Delta_\sigma([u_i])$. Therefore,

$$[u'] = \sum_{i=1}^{r} \Delta_\sigma([u_i]) = \Delta_\sigma([u]).$$

Suppose $u = u_1 \times_p u_2$ with $\deg[u_1] = d_1$ and $\deg[u_2] = d_2$. Say $u_1 = \sum_{\alpha \in [n]^{d_1}} a_\alpha x_\alpha$ and $\sum_{\beta \in [n]^{d_2}} b_\beta x_\beta$. Then, $[u] = \sum_{\alpha,\beta} a_\alpha b_\beta \cdot x_\alpha \times_p x_\beta$. If u', u_1' and u_2' is the resulting computation in C'', then

$$\begin{aligned}
[u'] &= [u_1'] \times [u_2'] \\
&= \Delta_{\sigma_1}([u_1]) \times \Delta_{\sigma_2}([u_2]) && \text{for some } \sigma_1 \in S_{d_1}, \sigma_2 \in S_{d_2}, \\
&= \sum_{\alpha,\beta} a_\alpha b_\beta \cdot (\Delta_{\sigma_1}(x_\alpha) \times \Delta_{\sigma_2}(x_\beta)) \\
&= \sum_{\alpha,\beta} a_\alpha b_\beta \cdot \Delta_\sigma(x_\alpha \times_p x_\beta) && \text{for some } \sigma \in S_d, \\
&= \Delta_\sigma([u])
\end{aligned}$$ ◀

The following corollary is immediate from the fact that any circuit of depth d and size s can be computed by a formula of size $s^{O(d)}$ and hence an ABP of size $s^{O(d)}$.

▶ **Corollary 3.5.** *If $f \in \mathbb{F}\langle x_1, \ldots, x_n \rangle$ is a homogeneous degree d polynomial that is computable by a UPT circuit of size s, then there is some $\sigma \in S_d$ such that $\Delta_\sigma(f)$ is computable by a non-commutative algebraic branching program of size $s^{O(\log d)}$.*

Furthermore, the shuffling σ that permits this can also be efficiently computed given the underlying shape for the circuit computing f.

3.2 UPT circuits of constant width

For a UPT circuit C, recall that we say that its preimage-width is w if for every node τ in the shape T, there are at most w gates of C that have type τ. The following observation is evident from the proof of the above depth reduction.

▶ **Observation 3.6.** *If C is a UPT circuit of width w, then the depth reduced circuit C' as obtained in Theorem 1.4 has width $O(w^2)$.*

This observation would allow us to yield a more efficient hitting set for the class of *small width known shape* UPT circuits. Details are present in the full version.

4 Separating ROABPs and UPT circuits

▶ **Theorem 1.5** (Separating UPT circuits and ABPs, under shuffling). *There is an explicit n-variate degree d non-commutative polynomial f that is computable by UPT circuits of preimage-width $w = \mathrm{poly}(n, d)$ such that for every $\sigma \in S_d$, the polynomial $\Delta_\sigma(f)$ requires non-commutative ABPs of size $(nd)^{\Omega(\log nd)}$ to compute it.*

The polynomial and the proof technique described here were introduced by Hrubeš and Yehudayoff [13] to separate monotone circuits and monotone ABPs in the commutative regime. The polynomial used here is a non-commutative analogue of the polynomial used by [13]. Much of the proof is the argument of [13] tailored to the non-commutative setting.

4.1 The polynomial

Let T_d denote the complete binary tree of depth d (with 2^d leaves) and let $D = 2^{d+1} - 1$ refer to the number of nodes in T_d. We shall say that a colouring $\gamma : T_d \to \mathbb{Z}_m$ is *legal* if for every node $u \in T$, if v and w are the children of u then $\gamma(u) = \gamma(v) + \gamma(w) \bmod m$.

Let $v_1, \ldots, v_D$ be the vertices of T_d listed in an *in-order* manner (left-subtree listed inductively, then the root, and then the right-subtree listed inductively). We now define the non-commutative polynomial $P_d(x_1, \ldots, x_m) \in \mathbb{F}\langle x_1, \ldots, x_m \rangle$ of degree $D = 2^{d+1} - 1$ as

$$P_d(x_1, \ldots, x_m) = \sum_{\substack{\gamma \in [m]^D \\ \gamma \text{ is legal}}} x_{\gamma(v_1)} x_{\gamma(v_2)} \cdots x_{\gamma(v_D)}. \tag{4.1}$$

▶ **Lemma 4.1** (Upper bound). *For every $m, d > 0$, the polynomial $P_d(y_1, \ldots, y_m)$ can be computed by a non-commutative UPT circuit of size $O(m^2 d)$.*

(Refer to the full version for a proof.)

▶ **Theorem 4.2** (Lower bound). *For every permutation $\sigma \in S_D$, any non-commutative ABP computing the polynomial $\Delta_\sigma(P_d)$ has width $m^{\Omega(d)}$.*

Hence for $d = \log m$, we have that $P_d(x_1, \ldots, x_m)$ is computable by a UPT circuit of size $O(m^2 \log m)$ but for every $\sigma \in S_D$ the above theorem tells us that $\Delta_\sigma(P_d)$ requires ABPs of width $m^{\Omega(\log m)}$ to compute it. The lower bound follows on exactly same lines as the [13]. A proof is present in the full version.

5 Hitting sets for non-commutative models

Commutative brethren of non-commutative models

This reduction to an appropriate commutative case was used by Forbes and Shpilka [8] to reduce constructing hitting sets for non-commutative ABPs to hitting sets for commutative ROABPs (more precisely, to set-multilinear ABPs). They studied the image of the non-commutative polynomial under the map $\Psi : \mathbb{F}\langle x_1, \ldots, x_n \rangle_{\text{deg}=d} \to \mathbb{F}[y_{1,1}, \ldots, y_{d,n}]$ which is the unique $\mathbb{F}$-linear map given by $\Psi : x_{w_1} \cdots x_{w_d} \mapsto y_{1,w_1} \cdots y_{d,w_d}$.

For the model of non-commutative UPT circuits, the appropriate commutative model is a restriction of set-multilinear circuits that we call UPT set-multilinear (UPT-SML) circuits.

▶ **Definition 5.1** (Set-multilinear circuits). Let $\mathbf{y} = \mathbf{y}_1 \sqcup \cdots \sqcup \mathbf{y}_d$ be a partition of the variables. A circuit C computing a polynomial $f \in \mathbb{F}[\mathbf{y}]$ is said to be a *set-multilinear circuit* with respect to the above partition if:

- each gate $g \in C$ is labelled by a subset $S_g \subseteq [d]$ and g computes a polynomial over variables $\bigcup_{i \in S_g} \mathbf{y}_i$ where every monomial of $[g]$ is divisible by exactly one variable in $\mathbf{y}_i$ for each $i \in S_g$,
- if g is a $+$ gate, then the subset that labels g also labels each of its children,
- if g is a $\times$ gate with g_1 and g_2 being its children, then the subsets S_{g_1} and S_{g_2} labelling g_1 and g_2 respectively is a partition of S_g, i.e. $S_g = S_{g_1} \sqcup S_{g_2}$.

We shall say the circuit C is *UPT set-multilinear* if every parse tree of C is of the same shape and identically labelled. That is, if g and g' are $\times$ gates labelled by a set $S \subseteq [d]$, and if $g = g_1 \times g_2$ with S_1 and S_2 labelling g_1 and g_2, then the children of g' are also labelled by S_1 and S_2 respectively.

We shall say the set-multilinear circuit C is *FewPT(k) set-multilinear* if the circuit consists of parse trees of at most k different shapes.

A natural generalization that will be useful later is a *multi-output UPT set-multilinear* circuit, which is a UPT set-multilinear circuit that potentially has multiple output gates, which are all labelled with the same subset.

Forbes and Shpilka [8] showed that constructing hitting sets for these commutative models suffices for the non-commutative models by a simple reduction (details in the full version). We shall therefore focus on these commutative models for the hitting set constructions. And since we have already seen that such circuits can be depth reduced[4] to $O(\log d)$ depth, it suffices to construct a hitting set for $O(\log d)$-depth UPT and FewPT set-multilinear circuits.

5.1 Hitting sets for UPT set-multilinear circuits

▶ **Theorem 5.2** (Hitting sets for UPT set-multilinear circuits). *Let $\mathcal{C}$ be the class of n-variate degree d set-multilinear polynomials (with respect to $\mathbf{y} = \mathbf{y}_1 \sqcup \cdots \sqcup \mathbf{y}_d$) that are computable by UPT set-multlinear circuits of preimage-width w and depth r. Then, for $M = \left(\binom{w}{2} n^2 d + 1\right)^2$, the set*

$$\mathcal{H} = \left\{ (b_{11}, \ldots, b_{dn}) \ : \ \mathbf{p} \in [M]^r \ , \ a_k \in A \ , \ b_{ij} = \prod_{k=1}^{r+1} a_k^{2^{(i-1)n+(j-1)} \bmod p_i} \right\}$$

is a hitting set for $\mathcal{C}$ of size $\operatorname{poly}(ndw)^r$.

The proof of this theorem is obtained by constructing what is called a *basis isolating weight assignment* for polynomials simultaneously computed by a multi-output UPT-SML circuit, heavily borrowing from the ideas in [2]. The details of the hitting set construction are present in the full version.

5.2 Poly-sized hitting sets for constant width UPT circuits

▶ **Theorem 1.3** (Hitting sets for known-shape low-width UPT circuits). *Let $\mathcal{C}_{n,d,T,w}$ be the class of n-variate degree d non-commutative polynomials that are computable by UPT circuits of preimage-width at most w and underlying parse-tree shape as T. Over any field of zero or large characteristic, there is an explicit hitting set $\mathcal{H}_{n,d,T,w}$ of size $w^{O(\log d)}$* $\operatorname{poly}(nd)$ *for $\mathcal{C}_{n,d,T,w}$.*

This is an easy extension of the ideas from [9], a proof can be found in the full version.

6 FewPT circuits

In this section we describe the black-box identity test for FewPT(k) circuits. The following lemma from [17] shows that this class is equivalent to polynomials computed by sum of k UPT circuits (of possibly different shapes).

6.1 Preliminaries

▶ **Lemma 6.1** ([17], Lemma 16). *Let $f(\mathbf{x})$ be a polynomial computed by FewPT(k) circuit of preimage-width w. Then f can be equivalently computed by a sum of k UPT circuits of preimage-width w each.*

[4] the shuffling just reorders the partition of the set-multilinear circuit

Like in [17], we will refer to this class by Σ^k-UPT. We shall further qualify this notation to use Σ^k-UPT(w) to denote the class of circuits that is a sum of k UPT circuits of preimage-width w.

From this lemma, we can focus on constructing hitting sets for Σ^k-UPT-SML circuits. The proof largely follows the ideas of Gurjar, Korwar, Saxena and Thierauf [10][5].

Notation

Let $\mathbf{y} = \mathbf{y}_1 \sqcup \cdots \sqcup \mathbf{y}_d$ be a partition of the variables and let $S = \{s_1, \ldots, s_p\}$ be a subset of $[d]$. Define the set of variables $\mathbf{y}_S = \mathbf{y}_{s_1} \cup \cdots \cup \mathbf{y}_{s_p}$ and the set of monomials $\mathbf{y}^S = \mathbf{y}_{s_1} \times \cdots \times \mathbf{y}_{s_p}$. Also, define $\mathbf{y}_{-S} = \mathbf{y} \setminus \mathbf{y}_S$ and $\mathbf{y}^{-S} = \mathbf{y}^{[d] \setminus S}$.

▶ **Definition 6.2** (Coefficient operator)**.** Let $f = \sum_{m \in \mathbf{y}^{[d]}} \alpha_m m$ be a set-multilinear polynomial of degree d, for $S \subseteq [d]$ and a monomial $m \in \mathbf{y}^S$, define $\operatorname{coeff}_m : \mathbb{F}[\mathbf{y}] \to \mathbb{F}[\mathbf{y}_{-S}]$ to be as follows.

$$\operatorname{coeff}_m(f) = \sum_{m' \in \mathbf{y}^{-S}} \alpha_{(m \cdot m')} m'$$

where $\alpha_{(m \cdot m')}$ is the coefficient of mm' in f.

▶ **Lemma 6.3.** *Let $\mathbf{y} = \mathbf{y}_1 \sqcup \ldots \sqcup \mathbf{y}_d$ be a partition and $f(\mathbf{y})$ be a set-multilinear polynomial (with respect to the above partition) computed by a* UPT-SML *circuit of preimage-width w and underlying parse-tree shape T. Suppose $g(\mathbf{y})$ is another set-multilinear polynomial (under the same partition) that* cannot *be computed by a* UPT-SML *circuit of preimage-width w with the same shape T.*

Then, there exists $S \subseteq [d]$ and $R \in \mathbb{F}[\mathbf{y}_S]^{1 \times w'}$, and $P, Q \in \mathbb{F}[y_{-S}]^{w' \times 1}$ satisfying $f = RP$, $g = RQ$ with $w' \le w^2$ such that:

- *For each $i \in [w']$, there is a monomial $m_i \in \mathbf{y}^S$ such that the i-th element of P and Q is* $\operatorname{coeff}_{m_i}(f)$ *and* $\operatorname{coeff}_{m_i}(g)$ *respectively,*
- *there is a vector $\Gamma \in \mathbb{F}^{1 \times w'}$ of support size at most $w+1$ such that $\Gamma P = 0$ and $\Gamma Q \neq 0$,*
- *the coefficient space of R is full-rank, i.e. if we interpret R as a matrix over $\mathbb{F}$ by listing each of its w' entries as a column vector of coefficients, then this matrix has full column-rank.*
- *the vector of polynomials R is simultaneously computable by a* UPT-SML *circuit of preimage-width at most w'.*

This lemma is a fairly natural and straightforward generalization of [10, Lemma 4.5] and a proof of this is provided in the full version.

▶ **Lemma 6.4.** *Suppose $f(\mathbf{y})$ is a non-zero polynomial computed by a Σ^k-UPT-SML(w) circuit. Suppose* $\operatorname{wt} : \mathbf{y} \to M^r$ *is a weight assignment that satisfies the following properties:*

- wt *is a BIWA for spaces of polynomials simultaneously computed by* UPT-SML *circuits of preimage-width at most $w(w+1)$,*
- *For any g in Σ^{k-1}-UPT-SML$(w(w+1))$, the polynomial $g(\mathbf{y} + \mathbf{t}^{\operatorname{wt}}) \in \mathbb{F}(\mathbf{t})[\mathbf{y}]$ has a monomial with non-zero coefficient that depends on at most ℓ distinct variables in $\mathbf{y}$.*

[5] [10] constructed hitting sets for sums of ROABPs and we use similar techniques for sums of UPT circuits. Roughly speaking, if we have a class $\mathcal{C}$ that has a *characterizing set of dependencies* for which we know how to construct BIWAs, then we can also construct hitting sets for $\Sigma^k\mathcal{C}$.

Then, the polynomial $f(\mathbf{y}+\mathbf{t}^{\mathrm{wt}})$ *has a monomial, depending on at most* $\log(w(w+1))+\ell$ *distinct variables in* $\mathbf{y}$*, with a non-zero coefficient.*

This is essentially a restatement of [10, Lemma 4.6, Lemma 4.8] and follows from their proof. Unravelling the recursion, we get the following corollary.

▶ **Corollary 6.5.** *Let* $f(\mathbf{y})$ *be a non-zero polynomial that can computed by a* Σ^k-UPT-SML(w) *circuit. Suppose* $\mathrm{wt} : \mathbf{y} \to M^r$ *is a BIWA for the class of polynomials simultaneously computed by* UPT-SML *circuits of preimage-width at most* $w^{2^{O(k)}}$*. Then, the polynomial* $f(\mathbf{y}+\mathbf{t}^{\mathrm{wt}}) \in \mathbb{F}(\mathbf{t})[\mathbf{y}]$ *has a monomial with a non-zero coefficient that depends on at most* $2^{O(k)}\log w$ *variables in* $\mathbf{y}$*.*

Once we are guaranteed to retain a monomial of small-support, we can construct a hitting set by enumerating over all possible supports and applying the Schwartz-Zippel lemma [22, 7, 24, 28] (or apply standard generators such as the Shpilka-Volkovich generator [25]). This completes the proof of Theorem 1.2, which we restate below for convenience.

▶ **Theorem 1.2** (Hitting sets for circuits with few parse tree shapes)**.** *There is an explicit hitting set* $\mathcal{H}_{d,n,s,k}$ *of size at most* $(s^{2^k}nd)^{O(\log d)}$ *for the class of* n*-variate degree* d *homogeneous non-commutative polynomials in* $\mathbb{F}\langle x_1,\ldots,x_n\rangle$ *that are computed by non-commutative circuits of size at most* s *consisting of parse trees of at most* k *shapes.*

7 Open problems

An interesting open problem (at least to us) is whether we can give non-trivial hitting sets for the class of *non-commutative skew circuits.* Lagarde, Limaye and Srinivasan [17] provide a white-box PIT in some restricted settings when the skew circuits are somewhat closer to UPT (with some restriction on what sort of parse trees they can have) but removing this restriction would be a great step forward.

Another issue is that the current construction of hitting sets for FewPT circuits (which build on [10]) incurs quasipolynomial losses at two different places. The first is in the construction of the *basis isolating weight assignment (BIWA)*, and we only know to construct that using quasipolynomially large weights. The other is in a brute-force enumeration of all monomials of support $O(\log s)$. As a result, even if at a later date we have a construction of a BIWA with polynomially large weights, this proof would still only yield a quasipolynomially large hitting set for FewPT circuits. It would be interesting to see if this brute-force enumeration could be circumvented.

References

1 Manindra Agrawal. Proving Lower Bounds Via Pseudo-random Generators. In *Proceedings of the 25th International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2005)*, pages 92–105, 2005. `doi:10.1007/11590156_6`.

2 Manindra Agrawal, Rohit Gurjar, Arpita Korwar, and Nitin Saxena. Hitting-Sets for RO-ABP and Sum of Set-Multilinear Circuits. *SIAM Journal of Computing*, 44(3):669–697, 2015. `doi:10.1137/140975103`.

3 Manindra Agrawal, Chandan Saha, and Nitin Saxena. Quasi-polynomial hitting-set for set-depth-Δ formulas. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC 2013)*, pages 321–330, 2013. `eccc:TR12-113`. `doi:10.1145/2488608.2488649`.

4 Eric Allender, Jia Jiao, Meena Mahajan, and V. Vinay. Non-Commutative Arithmetic Circuits: Depth Reduction and Size Lower Bounds. *Theoretical Computer Science*, 209(1-2):47–86, 1998. doi:10.1016/S0304-3975(97)00227-2.

5 Vikraman Arvind and S. Raja. Some Lower Bound Results for Set-Multilinear Arithmetic Computations. *Chicago Journal of Theoretical Computer Science*, 2016. URL: http://cjtcs.cs.uchicago.edu/articles/2016/6/contents.html.

6 Walter Baur and Volker Strassen. The Complexity of Partial Derivatives. *Theoretical Computer Science*, 22:317–330, 1983. doi:10.1016/0304-3975(83)90110-X.

7 Richard A. DeMillo and Richard J. Lipton. A Probabilistic Remark on Algebraic Program Testing. *Information Processing Letters*, 7(4):193–195, 1978. doi:10.1016/0020-0190(78)90067-4.

8 Michael A. Forbes and Amir Shpilka. Quasipolynomial-Time Identity Testing of Non-commutative and Read-Once Oblivious Algebraic Branching Programs. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2013)*, pages 243–252, 2013. Full version at arXiv:1209.2408. doi:10.1109/FOCS.2013.34.

9 Rohit Gurjar, Arpita Korwar, and Nitin Saxena. Identity Testing for Constant-Width, and Commutative, Read-Once Oblivious ABPs. In *Proceedings of the 31st Annual Computational Complexity Conference (CCC 2016)*, pages 29:1–29:16, 2016. arXiv:1601.08031. doi:10.4230/LIPIcs.CCC.2016.29.

10 Rohit Gurjar, Arpita Korwar, Nitin Saxena, and Thomas Thierauf. Deterministic Identity Testing for Sum of Read-once Oblivious Arithmetic Branching Programs. In *Proceedings of the 30th Annual Computational Complexity Conference (CCC 2015)*, pages 323–346, 2015. arXiv:1411.7341. doi:10.4230/LIPIcs.CCC.2015.323.

11 Joos Heintz and Claus-Peter Schnorr. Testing Polynomials which Are Easy to Compute (Extended Abstract). In *Proceedings of the 12th Annual ACM Symposium on Theory of Computing (STOC 1980)*, pages 262–272, 1980. doi:10.1145/800141.804674.

12 Pavel Hrubes, Avi Wigderson, and Amir Yehudayoff. Non-commutative circuits and the sum-of-squares problem. In *Proceedings of the 42nd Annual ACM Symposium on Theory of Computing (STOC 2010)*, pages 667–676, 2010. doi:10.1145/1806689.1806781.

13 Pavel Hrubeš and Amir Yehudayoff. On Isoperimetric Profiles and Computational Complexity. In *Proceedings of the 43rd International Colloquium on Automata, Languages and Programming (ICALP 2016)*, pages 89:1–89:12, 2016. eccc:TR15-164. doi:10.4230/LIPIcs.ICALP.2016.89.

14 Laurent Hyafil. On the Parallel Evaluation of Multivariate Polynomials. *SIAM Journal of Computing*, 8(2):120–123, 1979. Preliminary version in the *10th Annual ACM Symposium on Theory of Computing (STOC 1978)*. doi:10.1137/0208010.

15 Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004. Preliminary version in the *35th Annual ACM Symposium on Theory of Computing (STOC 2003)*. doi:10.1007/s00037-004-0182-6.

16 Adam R. Klivans and Amir Shpilka. Learning Restricted Models of Arithmetic Circuits. *Theory of Computing*, 2(10):185–206, 2006. Preliminary version in the *16th Annual Conference on Computational Learning Theory (COLT 2003)*. doi:10.4086/toc.2006.v002a010.

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. eccc:TR17-077. 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 Computa-*

tional 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. `eccc:TR15-22`. `doi:10.4086/toc.2016.v012a012`.

20 Noam Nisan. Lower bounds for non-commutative computation. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing (STOC 1991)*, pages 410–418, 1991. Available on `citeseer:10.1.1.17.5067`. `doi:10.1145/103418.103462`.

21 Noam Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992. `doi:10.1007/BF01305237`.

22 Øystein Ore. Über höhere kongruenzen. *Norsk Mat. Forenings Skrifter*, 1(7):15, 1922.

23 Ran Raz and Amir Shpilka. Deterministic polynomial identity testing in non-commutative models. *Computational Complexity*, 14(1):1–19, 2005. Preliminary version in the *19th Annual IEEE Conference on Computational Complexity (CCC 2004)*. `doi:10.1007/s00037-005-0188-8`.

24 Jacob T. Schwartz. Fast Probabilistic Algorithms for Verification of Polynomial Identities. *Journal of the ACM*, 27(4):701–717, 1980. `doi:10.1145/322217.322225`.

25 Amir Shpilka and Ilya Volkovich. Read-once polynomial identity testing. *Computational Complexity*, 24(3):477–532, 2015. Preliminary version in the *40th Annual ACM Symposium on Theory of Computing (STOC 2008)*. `doi:10.1007/s00037-015-0105-8`.

26 Leslie G. Valiant. Completeness Classes in Algebra. In *Proceedings of the 11th Annual ACM Symposium on Theory of Computing (STOC 1979)*, pages 249–261, 1979. `doi:10.1145/800135.804419`.

27 Leslie G. Valiant, Sven Skyum, S. Berkowitz, and Charles Rackoff. Fast Parallel Computation of Polynomials Using Few Processors. *SIAM Journal of Computing*, 12(4):641–644, 1983. Preliminary version in the *6th Internationl Symposium on the Mathematical Foundations of Computer Science (MFCS 1981)*. `doi:10.1137/0212043`.

28 Richard Zippel. Probabilistic algorithms for sparse polynomials. In *Symbolic and Algebraic Computation, EUROSAM '79, An International Symposiumon Symbolic and Algebraic Computation*, volume 72 of *Lecture Notes in Computer Science*, pages 216–226. Springer, 1979. `doi:10.1007/3-540-09519-5_73`.

A Relative computational power of restricted parse tree models

In this section we discuss the relative computational power of the models studied in the paper and their algebraic branching program counterparts.

We begin by recalling that Lagarde, Malod and Perifel [18] show that the computational power of non-commutative UPT circuits lies *strictly* between that of ABPs and circuits. In other words their work refines the strict separation between ABPs and circuits given by the seminal work of Nisan [20]. Each of these results use a generalization of the notion of *partial derivative matrix* which was first introduced by Nisan. We skip defining the generalized partial derivative matrix formally for brevity. A formal definition can be found in the proof of Theorem 1.5 in the full version. The following statements will help in understanding the rest of this section.

- Nisan [20] showed that for any homogeneous non-commutative polynomial f the *width* of a (homogeneous) ABP computing it, in the layer i, is exactly equal to the rank of the partial derivative matrix of f for degree i.
- Lagarde et al. [18] show that in the smallest UPT circuit of shape T computing a polynomial f, the number of gates of a type $\tau \in T$ is equal to the rank of the generalized partial derivative matrix for the type τ.

Thus, the ranks of the appropriate generalized partial derivative matrices for f characterize the ABP complexity and the UPT complexity of f. As discussed in section 1.3, we have strict separations between ABPs, UPT circuits and general circuits, even under shufflings. We now provide an informal discussion about constant width (or preimage-width) models for the sake of completeness.

A.1 Constant width models

For ease of exposition, we will use the term *width* to refer to both the width of an ABP and the preimage-width of a UPT circuit. The intended meaning will be clear from the context.

We can obtain a strict separation between constant width ABPs and constant width UPT circuits using the proof of Theorem 1.5. This is done by working with a constant variate version of the polynomial described in the proof. We skip the details to avoid repeating the proof. A more interesting comparison is that of ABPs of unrestricted (poly(n)) width and constant width UPT circuits, which we discuss now.

A.1.1 ABPs vs constant width UPT circuits

Let f_n be the following non-commutative variant of the elementary symmetric polynomial on n-variables of degree $d = n/2$.

$$f_n := \sum_{1 \le i_1 < \cdots < i_d \le n} x_{i_1} x_{i_2} \cdots x_{i_d}$$

Note that the generalized partial derivative matrix of f_n for any interval of positions $I \subset [d]$ will have rank $\geq n - d = \Omega(n)$. Moreover, any shuffling of f_n has the same property, since the rank does not depend on how we "order" the indices in $[n]$. Hence, any UPT circuit computing f_n or even a shuffling of f_n, requires poly(n) width. However, it is easy to see that f_n has a poly(n) sized ABP.

▶ **Fact A.1.** *(Informal) There is a polynomial f_n that is computable by a* poly(n) *sized ABP, but any shuffling of f_n requires UPT circuits of width $\Omega(n)$.*

Consider the bivariate palindrome polynomial of degree $2n$, for a growing parameter n.

$$P_n(x_1, x_2) = \sum_{(i_1,\ldots,i_n)\in[2]^n} (x_{i_1} x_{i_2} \cdots x_{i_n}) \cdot (x_{i_n} \cdots x_{i_2} x_{i_1})$$

It is easy to verify that the rank of the partial derivative matrix of $P_n(x_1, x_2)$ for degree n, is exactly 2^n. Also note that the polynomial has a UPT circuit of constant width. However, as remarked before in the paper, there is a shuffling of the palindrome polynomial that makes it simple for ABPs. Therefore we get the following fact, *when shufflings are not allowed.*

▶ **Fact A.2.** *(Informal) The classes of constant width UPT circuits and ABPs are incomparable.*

Let us now look at the case when shufflings are allowed.

Say $f_n(\mathbf{x})$ is an n-variate homogeneous polynomial of degree poly(n). If f has a UPT circuit C of width w, then Theorem 1.4 gives us that a shuffling of f_n, say f_n', has a UPT circuit C' of depth $O(\log n)$ and width $O(w^2)$. Let T be the shape of C' and let $g \in C'$ be a gate with type $\tau \in T$. Note that any path in C', from g to the root of C', goes through $O(\log n)$ gates, each of which is one out of the $O(w^2)$ gates of its type. Therefore even a trivial conversion of C' to a formula replicates any gate $g \in C'$ at most $w^{O(\log n)}$ times. We therefore have the following.

▶ **Fact A.3.** *(Informal) If $f_n(\mathbf{x})$ is computable by a constant width UPT circuit of size* $\mathrm{poly}(n)$*, then a shuffling of f_n is computable by an ABP of size* $\mathrm{poly}(n)$*.*