Electronic Colloquium on Computational Complexity, Report No. 115 (2012)

Quasipolynomial-time Identity Testing of Non-Commutative and Read-Once Oblivious Algebraic Branching Programs

Michael A. Forbes* Amir Shpilka†

September 11, 2012

Abstract

We study the problem of obtaining efficient, deterministic, *black-box polynomial identity testing algorithms (PIT)* for read-once oblivious algebraic branching programs (ABPs). This class has an efficient, deterministic, white-box polynomial identity testing algorithm (due to Raz and Shpilka [RS05]), but prior to this work had no known such black-box algorithm. Here we obtain the first quasi-polynomial sized hitting sets for this class, when the order of the variables is known. This work can be seen as an algebraic analogue of the results of Nisan [Nis92] and Impagliazzo-Nisan-Wigderson [INW94] for space-bounded pseudorandom generators.

We also show that several other circuit classes can be black-box reduced to read-once oblivious ABPs, including set-multilinear ABPs (a generalization of depth 3 set-multilinear formulas), non-commutative ABPs (generalizing non-commutative formulas), and (semi-)diagonal depth-4 circuits (as introduced by Saxena [Sax08], and recently shown by Mulmuley [Mul12] to have implications for derandomizing Noether's Normalization Lemma). For set-multilinear ABPs and non-commutative ABPs, we give quasi-polynomial-time black-box PIT algorithms, where the latter case involves evaluations over the algebra of $(D+1) \times (D+1)$ matrices, where D is the depth of the ABP. For (semi-)diagonal depth-4 circuits, we obtain a black-box PIT algorithm (over any characteristic) whose run-time is quasi-polynomial in the runtime of Saxena's white-box algorithm, matching the concurrent work of Agrawal, Saha, and Saxena [ASS12]. Finally, by combing our results with the reconstruction algorithm of Klivans and Shpilka [KS06], we obtain deterministic reconstruction algorithms for the above circuit classes.

1 Introduction

Let C be an arithmetic circuit in the input variables $x_1, \ldots, x_n$, over a field $\mathbb{F}$. The output $C(x_1, \ldots, x_n)$ is a polynomial f in the ring $\mathbb{F}[x_1, \ldots, x_n]$. The polynomial identity testing (PIT) problem is to efficiently determine "$f \stackrel{?}{\equiv} 0$". In particular, we are asking if the formal expression f, as a polynomial in $\mathbb{F}[x_1, \ldots, x_n]$, is zero. Schwartz and Zippel [Zip79, Sch80] showed that if $0 \neq f \in \mathbb{F}[x_1, \ldots, x_n]$ is a polynomial of degree $\leq d$, and $\alpha_1, \ldots, \alpha_n \in S \subseteq \mathbb{F}$ are chosen uniformly at random, then $f(\alpha_1, \ldots, \alpha_n) = 0$ with probability at most[1] $\leq d/|S|$. Thus, given the circuit C, we can perform these evaluations efficiently, giving an efficient randomized procedure for answering

*Email: miforbes@mit.edu, Department of Electrical Engineering and Computer Science, MIT CSAIL, 32 Vassar St., Cambridge, MA 02139, Supported by NSF Grant CCF-0939370.

†Faculty of Computer Science, Technion — Israel Institute of Technology, Haifa, Israel, shpilka@cs.technion.ac.il. The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement number 257575.

[1]Note that this is meaningful only if $d < |S| \leq |\mathbb{F}|$, which in particular implies that f is not the zero function.

ISSN 1433-8092

"$f \stackrel{?}{\equiv} 0$". An important open problem is to find a derandomization of this algorithm, that is, to find a deterministic procedure for PIT that runs in polynomial time (in the size of the circuit C).

One interesting property of the above randomized algorithm of Schwartz-Zippel is that the algorithm does not need to "see" the circuit C. Namely, the algorithm only uses the circuit to compute the evaluations $f(\alpha_1, \ldots, \alpha_n)$. Such an algorithm is called a black-box algorithm. In contrast, an algorithm that can access the internal structure of the circuit C is called a white-box algorithm. Clearly, the designer of the algorithm has more resources in the white-box model and so one can expect that solving PIT in this model should be a simpler task than in the black-box model.

The problem of derandomizing PIT has received a lot of attention in the past few years. In particular, many works examine a particular class of circuits $\mathcal{C}$, and design PIT algorithms only for circuits in that class. One reason for this attention is the strong connection between deterministic PIT algorithms for a class $\mathcal{C}$ and lower bounds for $\mathcal{C}$. This connection was first observed by Heintz and Schnorr [HS80] (and also by Agrawal [Agr05]) for the black-box model and by Kabanets and Impagliazzo [KI04] for the white-box model (see also Dvir, Shpilka and Yehudayoff [DSY09]). Another motivation for studying the problem is its relation to algorithmic questions. Indeed, the famous deterministic primality testing algorithm of Agrawal, Kayal and Saxena [AKS04] is based on derandomizing a specific polynomial identity. Finally, the PIT problem is, in some sense, the most general problem that we know today for which we have randomized coRP algorithms but no polynomial time algorithms, thus studying it is a natural step towards a better understanding of the relation between RP and P. For more on the PIT problem we refer to the survey by Shpilka and Yehudayoff [SY10].

Although the white-box model seems to be simpler than the black-box model, for most models for which a white-box PIT algorithm is known, also a black-box PIT algorithm is known, albeit sometimes with worse parameters. Such examples include depth-2 circuits (also known as sparse polynomials) [BOT88, KS01], depth-3 $\Sigma\Pi\Sigma(k)$ circuits [SS11], Read-k formulas [AvMV11] and depth-3 tensors (also known as depth-3 set-multilinear circuits) [RS05, FS12]. While the running time of the algorithms for depth-2 circuits and $\Sigma\Pi\Sigma(k)$ circuits are essentially the same in the white-box and black-box models, for Read-k formulas and set-multilinear depth-3 circuits we encounter some loss that results in a quasi-polynomial running time in the black-box model compared to a polynomial time algorithm in the white-box model.

Until this work, the only model for which an efficient white-box algorithm was known without a (sub-exponential) black-box counterpart was the model of non-commutative arithmetic formulas, or, more generally, the models of non-commutative algebraic branching programs (ABPs) and set-multilinear algebraic branching programs [RS05] (see Subsection 1.1 for definitions).

The main result in this paper is a quasi-polynomial time PIT algorithm in the black-box model for read-once oblivious algebraic branching programs. Using this result we obtain black-box algorithms of similar running times for the models of set-multilinear ABPs, non-commutative ABPs, as well as for diagonal circuits as defined by Saxena [Sax08]. Although exponential lower bounds are known for these models, we note that the algebraic hardness-versus-randomness result of Kabanets and Impagliazzo [KI04] (as well as the extension of this result by Dvir, Shpilka and Yehudayoff [DSY09] to low-depth circuits) does not imply a black-box PIT algorithm for the model, since their technique does not work for the restricted models studied here.

Although the models which are considered here may seem a bit unnatural at first sight, we note that the non-commutative model received a lot of attention in the algebraic complexity literature (mostly in the last few years) as it gives a good starting point for understanding the hardness of computing the determinant and permanent [Nis91, CS07, AJS09, AS10, HWY10a, CHSS11,

HWY10b, HY12]. In addition, we show in the paper that the model of set-multilinear ABPs generalizes the so called model of diagonal circuits that was introduced by Saxena [Sax08]. Recently, Mulmuley [Mul12] has shown a connection between derandomizing PIT for diagonal circuits and the problem of derandomizing Noether's normalization lemma. We discuss this in more detail in Subsection 1.3. We also note that derandomizing PIT in the general ABP model is a very important question. Specifically, the work of Ben-Or and Cleve [BOC92] implies that black-box derandomization of PIT, even for width-3 ABPs, would imply derandomization of PIT for general arithmetic circuits of logarithmic depth and in general, a quasi-polynomial time derandomization of PIT for poly-size arithmetic circuits. Saha, Saptharishi and Saxena [SSS09] showed that black-box derandomization of PIT of width-2 ABPs implies derandomization of PIT for depth-3 circuits, a model for which our knowledge is quite limited at the moment.

Besides the natural goal to obtain the most general possible PIT result, the problem of obtaining a black-box version of the algorithm of Raz and Shpilka [RS05] is interesting because of the technique used there. Roughly, the PIT algorithm of [RS05] works for any model of computation that outputs polynomials whose space of partial derivatives is (relatively) low-dimensional. Set-multilinear ABPs, non-commutative ABPs, low-rank tensors, and so called pure arithmetic circuits (defined in Nisan and Wigderson [NW96]) are all examples of algebraic models that compute polynomials with that property, and so the algorithm of [RS05] works for them. In some sense using information on the dimension of partial derivatives of a given polynomial is the most applicable technique when trying to prove lower bounds for arithmetic circuits (see e.g., [Nis91, NW96, Raz06, Raz09, RSY08, RY09]) and so it was an interesting problem to understand whether this powerful technique could be carried over to the black-box setting. Prior to this work it was not known how to use this low-dimensional property in order to obtain a small hitting set, and this paper achieves this goal. Earlier, in [FS12], we obtained a black-box algorithm for the model of low-rank tensors, but could not prove that it also works for the more general case of set-multilinear ABPs (we still we do not know whether this is the case or not - we discuss the similarities and differences between this and our new algorithm in Subsection 1.3), so this work closes a gap in our understanding of such low-dimensional models.

We shall now define the model and state our results.

1.1 The model

The model that we consider in this paper is that of set-multilinear algebraic branching programs. In fact, we will work with a slightly more general model, but we first describe the model of set-multilinear ABPs.

Algebraic branching programs were first defined in the work of Nisan [Nis91] who proved exponential lower bounds on the size of non-commutative ABPs computing the determinant or permanent polynomials.

Definition 1.1 (Nisan)**.** *An* ***algebraic branching program (ABP)*** *is a directed acyclic graph with one vertex of in-degree zero, which is called the* source*, and one vertex of out-degree zero, which is called the* sink*. The vertices of the graph are partitioned into levels numbered* $0, \ldots, D$*. Edges may only go from level* $i-1$ *to level* i *for* $i = 1, \ldots, D$*. The source is the only vertex at level* 0 *and the sink is the only vertex at level* D*. Each edge is labeled with a affine function in the input variables. The width of an ABP is the maximum number of nodes in any layer, and the size of the ABP is the number of vertices.*

Each directed source-sink path in the ABP computes a polynomial, which is a product of the labels on the edges in that path. As this work concerns non-commutative computation, we specify

that the product of the labels is in the order of the path, from source to sink. The ABP itself computes the sum of all such polynomials.

We consider a slight variation of this model which we call set-multilinear ABP, in line with the term coined by Nisan and Wigderson [NW96]. In the set-multilinear scenario the variables are partitioned into sets

$$X = X_1 \sqcup X_2 \sqcup \cdots \sqcup X_D,$$

where

$$X_i = \{x_{i,1}, \ldots, x_{i,n}\}.$$

A set-multilinear monomial is a monomial of the form

$$m = x_{1,i_1} \cdot x_{2,i_2} \cdots x_{D,i_D}.$$

In words, a set-multilinear monomial is a multilinear monomial that contains exactly one variable from each X_i. A set-multilinear polynomial is a polynomial consisting of set-multilinear monomials. In other words, the coefficients of a set-multilinear polynomial can be viewed as map from $[n]^D$ to the field $\mathbb{F}$, and thus is an D-dimensional tensor.

Definition 1.2 (Set-multilinear ABP)**.** *A **set-multilinear algebraic branching program (ABP)** in the variable set $X = X_1 \sqcup X_2 \sqcup \cdots \sqcup X_D$ is an ABP of depth D, such that each edge between layer $i-1$ and layer i is labeled with a (homogeneous) linear function in the variable set X_i.*

It is clear from the definition that a set-multilinear ABP computes a set-multilinear polynomial. It is also not hard to see that any set-multilinear polynomial can be computed by a set-multilinear ABP.

In fact, our result holds for the following model, that we call read-once oblivious ABPs, as well.

Definition 1.3 (Read-Once Oblivious ABP)**.** *An **read-once oblivious ABP** in the variable set $X = \{x_1, \ldots, x_D\}$ is an ABP of depth D, such that each edge between layer $i-1$ and layer i is labeled with a univariate polynomial in x_i of degree $< n$.*

Note that unlike previous definitions, in read-once oblivious ABPs we allow edges to be labeled with arbitrary univariate polynomials (with a bound of n on the degree) and not just with linear forms. Observe that be mapping $x_{i,j} \leftrightarrow x_i^j$ transforms any set-multilinear ABP into an read-once oblivious ABP and vice versa (when we index j starting at zero).

1.2 Our results

A black-box PIT algorithm is also known as an explicit hitting set, which we now define. We phrase the definition in generality to capture the notion of hitting sets for non-commutative polynomials, generalizing the usual notion.

Definition 1.4 (Hitting Set)**.** *Let $\mathcal{C}$ be a class of non-commutative n-variate polynomials, with coefficients in $\mathbb{F}$. Let $\mathcal{R}$ be a non-commutative ring with a (commutative) ring homomorphism $\mathbb{F} \to \mathcal{R}$, so that polynomials in $\mathcal{C}$ are defined over $\mathcal{R}$. A set $\mathcal{H} \subseteq \mathcal{R}^n$ is a **hitting set for** $\mathcal{C}$ if for all $f \in \mathcal{H}$, $f \equiv 0$ iff $f|_{\mathcal{H}} \equiv 0$.*

*The hitting set $\mathcal{H}$ is $t(n)$-**explicit** if given an index into $\mathcal{H}$, the corresponding element of $\mathcal{H}$ can be computed in $t(n)$-time.*

When $\mathcal{C}$ is commutative, we will always use $\mathcal{R} = \mathbb{F}$, and when $\mathcal{C}$ is non-commutative, we will take $\mathcal{R} = \mathbb{F}^{m \times m}$ for some appropriate m.

Our main result is a quasi-polynomial time black-box PIT algorithm for read-once oblivious ABPs.

Theorem (Theorem 3.22, PIT for read-once oblivious ABPs)**.** *Let $\mathcal{C}$ be the set of D-variate polynomials computable by width r, depth D, individual degree $< n$ read-once oblivious ABPs. If $|\mathbb{F}| \geq \mathsf{poly}(D, n, r)$, then $\mathcal{C}$ has a $\mathsf{poly}(D, n, r)$-explicit hitting set, of size $\leq \mathsf{poly}(D, n, r)^{\mathcal{O}(\lg D)}$.*

We remark here that our result assumes that the read-once oblivious ABP takes the order $x_1 < \cdots < x_n$, and that our results do not hold under permutation of the variables. In Subsection 3.1 we explain our proof technique and give an overview of the proof. Using this theorem we obtain black-box PIT algorithms for several related models. We first observe that PIT for set-multilinear ABPs is an immediate corollary of Theorem 3.22. As with read-once oblivious ABPs, we assume that we know the partition of the variables into the D sets, and our results do not hold under permutation of the variables.

Theorem (Theorem 3.23, PIT for set-multilinear ABPs)**.** *Let $X = X_1 \sqcup X_2 \sqcup \cdots \sqcup X_D$, where $X_i = \{x_{i,1}, \ldots, x_{i,n}\}$, be a known partition. Let $\mathcal{C}$ be the set of set-multilinear polynomials $f(X_1, \ldots, X_D) : \mathbb{F}^{nD} \to \mathbb{F}$ computable by a width r, depth D, set-multilinear ABP. If $|\mathbb{F}| \geq \mathsf{poly}(D, n, r)$, then $\mathcal{C}$ has a $\mathsf{poly}(D, n, r)$-explicit hitting set, of size $\leq \mathsf{poly}(D, n, r)^{\mathcal{O}(\lg D)}$.*

Next, we consider the model of non-commutative ABPs (see Section 4). Raz and Shpilka [RS05] gave a polynomial time white-box PIT algorithm for this model and we obtain a quasi-polynomial time black-box PIT algorithm. The evaluation points in our hitting set are vectors of $(D+1)\times(D+1)$ matrices for ABPs of depth D. We explain this choice in Section 4. In contrast to the above two results, this result for non-commutative ABPs makes no assumption about the variable ordering, and thus still holds under permutation of the variables.

Theorem (Corollary 4.7, Black-box PIT for non-commutative ABPs)**.** *Let $\mathcal{NC}$ be the set of n-variate non-commutative polynomials computable by width r, depth D ABPs. If $|\mathbb{F}| \geq \mathsf{poly}(D, n, r)$, then $\mathcal{NC}$ has a $\mathsf{poly}(D, n, r)$-explicit hitting set over $(D+1) \times (D+1)$ matrices, of size $\leq \mathsf{poly}(D, n, r)^{\mathcal{O}(\lg D)}$.*

Saxena [Sax08] defined the model of diagonal circuits (see Section 5 for definition) in an attempt to capture some of the complexity of depth-4 circuits. Relying on the algorithm of [RS05], Saxena obtained a polynomial time white-box PIT algorithm for this model (for a certain setting of the parameters). Saha, Saptharishi and Saxena [SSS11], with the essentially same techniques, later extended Saxena's white-box results to the so-called semi-diagonal model. Simultaneously and independently of our work, Agrawal, Saha and Saxena [ASS12] gave a black-box PIT algorithm for semi-diagonal depth-4 circuits that runs in quasi-polynomial time in the runtime of the known white-box algorithm, and works over fields of large characteristic.

Using our main theorem and a new reduction from diagonal circuits to read-once oblivious ABPs (Lemma 5.8) we obtain a PIT algorithm for diagonal circuits that runs in time quasi-polynomial in the white-box algorithm given by Saxena. Further, we do so over any characteristic in a unified way. As with our result on non-commutative ABPs, this result makes no assumptions about variable order. With essentially no modification, we obtain similar results for semi-diagonal circuits.

Theorem (Theorem 5.11, Black-box PIT for semi-diagonal circuits)**.** *Let $\mathbb{F}$ be a field of arbitrary characteristic. Let $\mathcal{SDC}$ be the set of n-variate polynomials computable by semi-diagonal depth-4 circuits, that is, of the form $\Phi = \sum_{i \in [k]} m_i(\vec{x}) \cdot P_{i,1}^{e_{i,1}} \cdots P_{i,r_i}^{e_{i,r_i}}$, where $m_i(\vec{x})$ is a monomial of*

degree $\leq d$, $\prod_{j=1}^{r_i}(1+e_{i,j}) \leq e$ *and* $P_{i,j}$ *is a sum of univariate polynomials of degree* $\leq d$. *Then if* $|\mathbb{F}| \geq \mathsf{poly}(n,k,d,e)$ *then* $\mathcal{SDC}$ *has a* $\mathsf{poly}(n,k,e,d)$*-explicit hitting set of size* $\leq \mathsf{poly}(n,d,k,e)^{\mathcal{O}(\lg n)}$.

Klivans and Shpilka [KS06] gave a polynomial-time learning algorithm for read-once oblivious ABPs[2], given membership and equivalence queries. That is, they give a deterministic algorithm that can learn an unknown f computed be a small read-once oblivious ABP, given a oracle that evaluates f ("membership query"), as well an oracle such that for any hypothesis h computed by a small read-once oblivious ABP with $h \neq f$, the oracle returns an evaluation point $\vec{x}$ such that $h(\vec{x}) \neq f(\vec{x})$ ("equivalence query"). Membership queries can be implemented deterministically given black-box access to f, and equivalence queries can be implemented using random queries. That is, by appealing to the Schwartz-Zippel algorithm, distinguishing $h \neq f$ can be done using random evaluations.

Our above results construct hitting sets for read-once oblivious ABPs, and as these are closed under subtraction (that is, for equivalence queries, $h - f$ is also a small read-once oblivious ABP), our hitting set will always contain a distinguishing evaluation for h and f, if $h \neq f$. Thus, we can derandomize the equivalence queries needed for [KS06]. Further, our results on set-multilinear ABPs, non-commutative ABPs and semi-diagonal depth-4 circuits, all rely on reductions to read-once oblivious ABPs, and these reductions also hold in the learning setting. We omit further details, and state the following result.

Theorem 1.5 (Extension and Derandomization of [KS06])**.** *There is a deterministic polynomial-time learning algorithm from membership and equivalence queries that can learn each of the following models: read-once oblivious ABPs, set-multilinear ABPs, non-commutative ABPs and semi-diagonal depth-4 circuits. In the first three models the algorithm outputs an hypothesis from the same class as the unknown function, but in the case of semi-diagonal depth-4 circuits the outputted hypothesis is a read-once oblivious ABP. Given such query access, the running time of the algorithm is polynomial in the size*[3] *of the underlying branching program or circuit.*

Furthermore, such membership and equivalence queries can be implemented in randomized-polynomial time, or in deterministic quasi-polynomial time, for any of the classes mentioned above.

1.3 Related work

In [Mul12] Mulmuley studies the problem of derandomizing Noether's normalization lemma. Roughly, this lemma says that, for an algebraically closed field $\mathbb{K}$ and a projective variety $V \subset P(\mathbb{K}^s)$ ($P(\mathbb{K}^s)$ is the projective space over $\mathbb{K}^s$) of dimension n, if we pick a linear transformation $\Psi : \mathbb{K}^s \to \mathbb{K}^m$ at random for $m > n$, then Ψ will be well-defined on V. Furthermore, for any $v \in V$ the set $\Psi^{-1}(\Psi(v)) \cap V$, for such a random Ψ, is finite. The complexity theoretic question associated with the lemma, is given the set of polynomials defining V, how to explicitly construct the map Ψ. In fact, even given Ψ, it is not clear how to verify that it satisfies the required properties. In his paper Mulmuley considers two explicit varieties. The one that is related to diagonal circuits is the coordinate ring $\mathbb{K}[V]$ of a polynomial representation V of $G = \mathrm{SL}_m(\mathbb{K})$. An equivalent formulation of Noether's normalization lemma says that there exists a set $S \subseteq \mathbb{K}[V]^G$ (where $\mathbb{K}[V]^G$ is the subring of G-invariants in $\mathbb{K}[V]$) consisting of $\mathsf{poly}(n)$ homogeneous invariants such that $\mathbb{K}[V]^G$ is integral over the subring generated by S. Again, a random choice of S has the required property, but deterministic construction of such a small S is not known (nor an

[2]The terminology used in [KS06] is that of learning by multiplicity automata, but these are completely equivalent to read-once oblivious ABPs, as shown in [KS06].

[3]For convenience here, we define the "size" of a semi-diagonal depth-4 circuit to be some $\mathsf{poly}(n,k,e,d)$, using the notation of Theorem 5.11, even though the actual circuit could be much smaller.

efficient way of certifying whether a given S is good, see [Mul12] for a more detailed discussion). Theorem 1.1 of [Mul12] shows that black-box derandomization of PIT for diagonal circuits implies derandomization of Noether's lemma in the sense that one can construct such a set S efficiently. Moreover, Mulmuley views this connection as an evidence that derandomization of black-box PIT of even diagonal circuits is a hard problem. To support this view [Mul12] mentions that, currently, the best deterministic algorithms are doubly exponential in n.[4]

In contrast to this view, our Theorem 5.9 implies that for the variety studied in [Mul12] the set S can be constructed in quasi-polynomial time. Thus, instead of looking at the results in [Mul12] as indicating hardness of PIT, we suggest to view them as offering algorithms for several problems of giving explicit constructions.

Theorem 1.2 of [Mul12] gives derandomization of Noether's normalization lemma under certain assumptions such as EXP containing a function with a sub-exponential lower bound, or assuming that EXP is not contained in MA and that the generalized Riemann hypothesis holds. In both cases the implied running time is at least quasi-polynomial so our result achieves this without further assumptions.

It is interesting to note that to get a quasi-polynomial sized hitting set for depth-3 diagonal circuits one does not need our new result and in fact, such a hitting set can be constructed using the techniques of Shpilka and Volkovich [SV09].

The reader is encouraged to read [Mul12] to learn more on this connection.

While the work of Raz and Shpilka [RS05] establishes a white-box PIT algorithm for non-commutative ABPs, the black-box PIT question for this model is more intricate since one cannot immediately apply the usual Schwartz-Zippel algorithm over non-commutative domains. However, Bogdanov and Wee [BW05] showed how, leveraging the ideas in the Amitsur-Levitzki theorem [AL50], one can reduce non-commutative black-box PIT questions to commutative black-box PIT questions. By then appealing to Schwartz-Zippel, they give the first randomized algorithm for non-commutative PIT. They also discussed the possibility of derandomizing their result and raise a conjecture that if true would lead to a hitting set of size $\approx s^{\log^2 s}$ for non-commutative ABPs of size s. Our Theorem 4.6 gives a hitting set of size $\approx s^{\log s}$ and does not require unproven assumptions.

In particular, their conjecture asked about the minimal size of an ABP required to computed a polynomial identity of $n \times n$ matrices. Recall that a polynomial identity of matrices is a non-zero polynomial $f(\vec{x})$ such that no matter which $n \times n$ matrices we substitute for the variables x_i, f evaluates to the zero matrix. Our work bypasses this conjecture, in that we more directly translate the non-commutative polynomial into a (set-multilinear) commutative polynomial. Our construction consists of $(D+1) \times (D+1)$ strictly-upper-triangular matrices for depth D non-commutative ABPs. It is also not hard to see that there is a non-commutative formula of depth $D+1$ which is a polynomial identity for the space of $(D+1) \times (D+1)$ strictly-upper-triangular matrices. Thus, our result is tight in that it also illustrates that if we go just one dimension up above what is obviously necessary then we can already construct a hitting set.

In [AMS10], Arvind, Mukhopadhyay and Srinivasan gave a deterministic black-box algorithm for identity testing of sparse non-commutative polynomials. The algorithm runs in time polynomial in the number of variables, degree and sparsity of the unknown polynomial. This is similar to the running time achieved in the commutative setting for sparse polynomials (see e.g., [BOT88, KS01]) and in particular it is better than our quasi-polynomial time algorithm. On the other hand our algorithm is more general and works for any non-commutative polynomial that is computed by a

[4] More accurately, the doubly-exponential bound is for a general variety V. Because of the reduction to PIT there is a single exponential time algorithm for the specific varieties that he is considering.

small ABP.

We note that in the aforementioned [AMS10], the authors showed how to deterministically learn sparse non-commutative polynomials in time polynomial in the number of variables, degree and sparsity. In contrast, for such polynomials our deterministic algorithm requires quasi-polynomial time. For general non-commutative ABPs [AMS10] also obtained a deterministic polynomial time learning algorithm, but here they need to have the ability to query the ABP also at internal nodes and not just at the output node. Our deterministic algorithm runs in quasi-polynomial time but it does not need to query the ABP at intermediate computations.

In a previous paper [FS12] we gave a hitting set of quasi-polynomial size for the class of depth-3 set-multilinear polynomials. That result is mostly subsumed by the generality of Theorem 3.23, but the previous work has a better dependence on some of the parameters. More importantly, it is interesting to note that the proof in [FS12] is also based on the same intuitive idea of preserving dimension of linear spaces while reducing the number of variables, but in order to prove the results in this paper we had to take a different approach. At a high level the difference can be described as follows. We now summarize the algorithm of [FS12]. First the case of degree 2 is solved. In this case the tensor is simply a bilinear map. For larger D, the algorithms works by first reducing to the bivariate case using the Kronecker substitution $x_i \leftarrow x^{n^i}$ for $i \leq D/2$ and $x_{i+D/2} \leftarrow y^{n^i}$ for $1 \leq i \leq D/2$. Appealing now to the bivariate case, we can take y to be some multiple of x, and then undo the Kronecker substitution (and applying some degree reduction) to recover a tensor on $D/2$ variables. If we try to implement this approach with ABPs then we immediately run into problems, as the previous work requires that the layers of the ABP are commutative, in that we can re-order the layers. While this is true for depth-3 set-multilinear computation, it is not true for general ABPs. To generalize the ideas to general ABPs, this work respects the ordering of the layers in the ABP. In particular, while the previous work had a top-down recursion, this work follows a bottom-up recursion. We merge variables in adjacent levels of the ABP and then reduce the degree of the resulting polynomial using an (algebraic rank) extractor. We do this iteratively until we are left with a univariate polynomial. Perhaps surprisingly, this gives a set that is simpler to describe as compared to the hitting set in [FS12]. On the other hand, if we restrict ourselves to set-multilinear depth-3 circuits then the hitting set of [FS12] is polynomially smaller than the set that we construct here.

Independently and simultaneously with this work, Agrawal, Saha and Saxena [ASS12] obtained results on black-box derandomization of PIT for small-depth set-multilinear formulas, when the partition of the variables into sets is unknown. They obtained a hitting set of size $\exp((2h^2 \log(s))^{h+1})$, for size s formulas of multiplicative-depth h. We note that their model is both stronger and weaker compared to the models that we consider here. On the one hand, this model does not assume knowledge of the partition of the variables, whereas our model assumes this knowledge. On the other hand, they only handle small-depth formulas, and indeed, the size of their hitting grows doubly-exponentially in the depth, whereas our results handle arbitrary set-multilinear formulas, according to their definition, if we know the partition, and the size of the hitting set grows quasi-polynomially with the depth[5]. Using their results for set-multilinear formulas, Agrawal, Saha and Saxena [ASS12] also give a quasipolynomial-sized hitting set for semi-diagonal circuits over large characteristic fields. Our results, in particular our extension of Saxena's [Sax08] duality to all fields, can help extend the results of [ASS12] to small characteristics

[5]Their definition of set-multilinear formulas is actually that of a pure formula (see [NW96]). It is not hard to see that pure formulas form a sub-model of the set-multilinear ABPs we examine in this paper. That is, the usual transformation from formulas to ABPs can be done preserving set-multilinearity.

as well.

We also mention that in [JQS09, JQS10] Jansen, Qiao and Sarma studied black-box PIT in various models related to algebraic branching programs. Essentially, all these models can be cast as a problem of obtaining black-box PIT for read-once oblivious branching programs where each variable appears on a small number of edges. Their result gives a hitting set of size (roughly) $n^{O(k \log(k) \log(n))}$ when k is an upper bound on the number of edges that a variable can label and the ABP is of polynomial size. In comparison, our Theorem 3.22 gives a hitting set of size $n^{O(\log(n))}$ and works for $k = \mathsf{poly}(n)$ (as long as the size of the ABP is polynomial in n). Our techniques are very different from those of [JQS09, JQS10], which basically follow the proof technique of [SV09].

This work fits into the research program of derandomizing PIT, in particular derandomizing black-box PIT. However, for many of the models of algebraic circuits studied, there are corresponding boolean circuit models for which derandomization questions can also be asked. In particular, for a class $\mathcal{C}$ of boolean circuits, we can seek to construct a *pseudorandom generator* $\mathcal{G} : \{0,1\}^s \to \{0,1\}^n$ for $\mathcal{C}$, such that for any circuit $C \in \mathcal{C}$ on n inputs, we have the ϵ-closeness of distributions $C(\mathcal{G}(U_s)) \approx_\epsilon C(U_n)$, where U_k denotes the uniform distribution on $\{0,1\}^k$. Nisan [Nis92] studied pseudorandom generators for space-bounded computation, and for space S computation gave a generator with seed length $s = \mathcal{O}(\lg^2 S)$. Impagliazzo, Nisan and Wigderson [INW94] later gave a different construction with the same seed length. Randomized space-bounded computation can be modeled with read-once oblivious (boolean) branching programs, and these generators apply to this model of computation as well. Our work, at its core, studies read-once oblivious (algebraic) branching programs, and we achieve a quasi-polynomial-sized hitting set, and can thus be viewed as having a "seed length" of $\mathcal{O}(\lg^2 S)$ for ABPs of size S. Aside from the similarities in the statements of these results, there are some similarities in the high-level techniques as well. Indeed, an interpretation by Raz and Reingold [RR99] argues that the [INW94] generator for space-bounded computation can be seen as recursively partitioning the branching program, using an (boolean) extractor to recycle random bits between the partitions. Similarly, in our work, we use an (algebraic rank) extractor between the partitions. At present, it is unclear if there are formal (or further informal) connections between the boolean pseudorandom generators and black-box PIT.

2 Notation

We write $[n]$ to denote $\{1, \ldots, n\}$, and $[\![n]\!]$ to denote $\{0, \ldots, n-1\}$. We write $\mathbb{1}_P$ to denote the indicator function for the set/event P. We use $\sqcup$ to denote a disjoint union.

Matrices and vectors will most often be indexed starting at zero. In observance of this indexing scheme, we will write $S^{[\![n]\!]}$ to denote n-dimensional vectors with entries in S, and will write $S^{[\![n]\!] \times [\![m]\!]}$ to denote matrices of size $n \times m$ with entries in S. Indexing from 1 (as done in Section 5) will be indicated by the use of $[n]$ as opposed to $[\![n]\!]$. We denote I for the identity matrix, whose dimension will always be clear from the context. For a matrix M, we will write $M_{i,\bullet}$ to denote the i-th row of M, and $M_{\bullet,j}$ to denote the j-th column, where i and j are indexed from zero. Given a bit vector $\vec{b} \in \{0,1\}^{[\![n]\!]}$, we write b_i for the i-th bit of $\vec{b}$, where i is indexed from zero.. When $n = 0$, we use λ to denote the empty string in $\{0,1\}^{[\![0]\!]}$, and thus for $n \geq 0$, $|\{0,1\}^{[\![n]\!]}| = 2^n$.

Given a vector of polynomials $\vec{f} \in \mathbb{F}[\vec{x}]^n$ and an exponent vector $\vec{\alpha} \in \mathbb{N}^n$, we write $\vec{f}^{\vec{\alpha}}$ for $f_1^{\alpha_1} \cdots f_n^{\alpha_n}$.

We write $M \in \mathbb{F}[\vec{x}]^{[\![r]\!] \times [\![r]\!]}$ to indicate that M is an $r \times r$ matrix, with entries in $\mathbb{F}[\vec{x}]$. The (total) degree of M, written $\deg(M)$, is defined as the maximum degree of its entries. Given a collection of

matrices $\mathcal{M} \subseteq \mathbb{F}^{[\![r]\!]\times[\![r]\!]}$, span $\mathcal{M}$ denotes the span of these matrices, in the vector space $\mathbb{F}^{[\![r]\!]\times[\![r]\!]} \equiv \mathbb{F}^{[\![r^2]\!]}$. Given $\mathcal{M} \subseteq \mathbb{F}^{[\![r]\!]\times[\![r]\!]}$ and $\mathcal{N} \subseteq \mathbb{F}^{[\![r]\!]\times[\![r]\!]}$, we define $\mathcal{M}\cdot\mathcal{N} := \{MN : M \in \mathcal{M}, N \in \mathcal{N}\}$. Given a polynomial $f \in \mathbb{F}[\vec{x}]$, we write $\mathfrak{C}_{\vec{x}^{\vec{\alpha}}}(f)$ to denote the coefficient of $\vec{x}^{\vec{\alpha}}$ in f. For a matrix $M \in \mathbb{F}[\vec{x}]^{[\![r]\!]\times[\![r]\!]}$, we write $\mathfrak{C}_{\vec{x}^{\vec{\alpha}}}(M)$ to denote the $r \times r$ $\mathbb{F}$-matrix, with the $\mathfrak{C}_{\vec{x}^{\vec{\alpha}}}$ operator applied to each entry. When we write "$f \in \mathbb{F}[\vec{x}][\vec{y}]$", we will treat f as a polynomial in the variables $\vec{y}$, whose coefficients are polynomials in the variables $\vec{x}$, and correspondingly will write $\mathfrak{C}_{\vec{y}^{\vec{\beta}}}(f)$ to extract the polynomial in $\vec{x}$ that is the coefficient of the monomial $\vec{y}^{\vec{\beta}}$ in f. Given a polynomial f (in $\mathbb{F}[\vec{x}]$ or in the non-commutative $\mathbb{F}\{\vec{x}\}$), we write $H_k(f)$ for the homogeneous part of f of degree k.

If $M_i \in \mathbb{F}^{[\![r]\!]\times[\![r]\!]}$ for $i \in [\![D]\!]$, then we write $\prod_{i\in[\![D]\!]} M_i$ for the non-commutative multiplication $M_0 \cdots M_{D-1}$, with the understanding that the multiplication proceeds left to right, starting at the lower index of the product ($i = 0$) and ending at the higher index ($D-1$). When this non-commutative product is indexed over the range $\{0,1\}^{[\![d]\!]}$, we order the product by the lexicographic order on $\{0,1\}^{[\![d]\!]}$, where the least-significant bit is the rightmost bit.

3 Hitting Set for Read-Once Oblivious ABPs

In this section, we construct a hitting set for read-once oblivious ABPs. To make the questions more amenable to study, we give the following normal form for read-once oblivious ABPs.

Lemma 3.1. *Let A be a width $\leq r$, depth D, individual degree $< n$ read-once oblivious ABP, computing a polynomial $f \in \mathbb{F}[x_0, \ldots, x_{D-1}]$. Then for $i \in [\![D]\!]$, there are matrices $M_i \in \mathbb{F}[x_i]^{[\![r]\!]\times[\![r]\!]}$ of degree $< n$ such that*

$$f(\vec{x}) = \left(\prod_{i\in[\![D]\!]} M_i(x_i)\right)_{(0,0)}.$$

Conversely, any such function can be computed by a width r, depth D, read-once oblivious ABP.

Proof. ABP $\Longrightarrow$ matrices: Identify the nodes in the i-th layer of A with a subset of $[\![r]\!] \times \{i\}$, so that the nodes of A are identified with a subset of $[\![r]\!] \times [\![D+1]\!]$ and the source is 0×0 and the sink is $0 \times D$. Then, define the matrices M_i such that the entry $(M_i)_{\ell,\ell'}$ takes the label of the edge $(\ell \times (i-1), \ell' \times i)$. Note that such a label is a univariate polynomial in x_i of degree $< n$. It is straight-forward to see that f is computed by this matrix product.

matrices $\Longrightarrow$ ABP: This is similar to the above. □

Thus, instead of talking about such ABPs, we will be discussing products of matrices $\prod_i M_i(x_i)$. For induction purposes we will not be restricting ourselves to the $(0,0)$-th entry, but will consider the full matrices. Such products naturally correspond to multi-source multi-sink ABPs. With this normal form in mind, we can give the proof overview.

3.1 Proof overview

At a high level, our hitting set can be viewed as a repeated application (in parallel) of a "derandomized Kronecker product". That is, the usual Kronecker product allows us to merge two variables (that is, we merge x and y by taking $y \leftarrow x^m$ for large enough m) of a polynomial without losing any information. Once a single variable remains, one can resort to full-interpolation of this univariate polynomial. However, this generic transformation rapidly increases the degree of the polynomial and thus the final interpolation step requires too many evaluations to yield a good hitting set.

For our hitting set, we first observe that in an read-once oblivious ABP each layer has its own variable, so merging two variables in adjacent layers results in the merging of the two layers in the ABP. We then give a degree-reduction procedure for the results of such merges. That is, given two adjacent layers of our ABP, $M(x) \in \mathbb{F}[x]^{[\![r]\!]\times[\![r]\!]}$ and $N(y) \in \mathbb{F}[y]^{[\![r]\!]\times[\![r]\!]}$, we first set $y \leftarrow x^n$ (where $\deg M, N < n$) to obtain $M(x)N(y) \approx M(x)N(x^n)$, where the right-hand term is of degree $\approx n^2$. We then construct $f, g \in \mathbb{F}[z]$ of degree $< r^2$ such that $M(x)N(x^n) \approx M(f(z))N(g(z))$, where the degree of the right-hand term is now $\approx nr^2$. Thus we have that $M(x)N(y) \approx M(f(z))N(g(z))$, and we have only increased the degree by $\approx r^2$. As r, the dimension of the matrices (and the width of the ABP), stays constant throughout this merging process, one can repeat this merging process and have the degree of the polynomial scale linearly with the number of mergings. To then fully leverage this idea, we show that pairs of variables can be merged in parallel.

To discuss our degree-reduction process, we first need to state what it means for "$M(x)N(x^n) \approx M(f(z))N(g(z))$". In the Kronecker substitution, this equivalence typically means that there is a bijection between monomials. We will not use that notion, and instead will us a problem-specific notion, related to linear algebra. Specifically, for each $\alpha, \beta \in \mathbb{F}$, the matrix $M(\alpha)N(\beta)$ is some linear transformation $\mathbb{F}^{[\![r]\!]} \to \mathbb{F}^{[\![r]\!]}$. Ranging over all α and β, we get a space of such transformations, which we extend by linearity to a vector space of transformations. Crucially, the polynomial $M(x)N(y)$ is zero iff this vector space is zero. More generally, we will show that this linear space of transformations captures essentially all we need to know about the polynomial $M(x)N(y)$. Our degree reduction lemma (using the Kronecker substitution, and then doing the degree reduction) finds two curves $f, g \in \mathbb{F}[z]$ of degree $< r^2$ such that the space of linear transformations induced by $M(x)N(y)$ is the same as the space of linear transformations induced by $M(f(z))N(g(z))$. It follows then that $M(x)N(y) \approx M(f(z))N(g(z))$, and so we have merged two variables without increasing the degree too much.

To find such curves, we use a (seeded) rank extractor. That is, the n^2 matrices defined by the coefficients of $M(x)N(y)$ live in a r^2-dimensional space, so we seek to map these n^2 vectors to a smaller space while preserving rank. Gabizon and Raz [GR05] established such a lemma, and our prior work [FS12] improved the parameters in their lemma. Crucially, these rank extractors have the form such that the maps they define correspond to polynomial evaluations. Thus, these extractors establish (for most values of the seed) an explicit small set of evaluation points where the span of $M(x)N(y)$ is preserved. By creating curves that through these points, we obtain the desired f and g for our degree reduction.

3.2 Derandomized Kronecker Product for the Span of an ABP

We will begin the formal statements of this section by giving lemmas on the span of matrices, and how preserving the span of linear transformations can be used for PIT. We begin by showing that the span of ABPs can be preserved in a recursive fashion.

Lemma 3.2. *Let* $\mathcal{M}, \mathcal{M}' \subseteq \mathbb{F}^{[\![r]\!]\times[\![r]\!]}$ *and* $\mathcal{N}, \mathcal{N}' \subseteq \mathbb{F}^{[\![r]\!]\times[\![r]\!]}$, *such that* $\operatorname{span}\mathcal{M} = \operatorname{span}\mathcal{M}'$ *and* $\operatorname{span}\mathcal{N} = \operatorname{span}\mathcal{N}'$. *Then* $\operatorname{span}(\mathcal{M}\cdot\mathcal{N}) = \operatorname{span}(\operatorname{span}(\mathcal{M})\cdot\operatorname{span}(\mathcal{N}))$ *and* $\operatorname{span}(\mathcal{M}\cdot\mathcal{N}) = \operatorname{span}(\mathcal{M}'\cdot\mathcal{N}')$.

Proof. As $\mathcal{M}\cdot\mathcal{N} \subseteq \operatorname{span}(\mathcal{M})\cdot\operatorname{span}(\mathcal{N})$ we see that $\operatorname{span}(\mathcal{M}\cdot\mathcal{N}) \subseteq \operatorname{span}(\operatorname{span}(\mathcal{M})\cdot\operatorname{span}(\mathcal{N}))$ by linearity. To show the other direction, consider any $A \in \operatorname{span}(\operatorname{span}(\mathcal{M})\cdot\operatorname{span}(\mathcal{N}))$. Then $A = \sum_i c_i(\sum_j a_{i,j}M_j)(\sum_k b_{i,k}N_k)$, for $M_j \in \mathcal{M}$ and $N_k \in \mathcal{N}$. Thus, by bilinearity of the matrix product, $A = \sum_{i,j,k} c_i a_{i,j} b_{i,k} M_j N_k$, where $M_j N_k \in \mathcal{M}\cdot\mathcal{N}$, and thus $A \in \operatorname{span}(\mathcal{M}\cdot\mathcal{N})$, yielding the first claim.

The second claim follows from the first, by observing that $\mathrm{span}(\mathcal{M} \cdot \mathcal{N}) = \mathrm{span}(\mathrm{span}(\mathcal{M}) \cdot \mathrm{span}(\mathcal{N})) = \mathrm{span}(\mathrm{span}(\mathcal{M}') \cdot \mathrm{span}(\mathcal{N}')) = \mathrm{span}(\mathcal{M}' \cdot \mathcal{N}')$. □

We will use this as follows. By Lemma 3.1 we can restrict our attention to matrix-valued functions of the form $f(\vec{x}) = \prod_{i \in [\![D]\!]} M_i(x_i)$, and we work in this level of generality for ease of induction. As such, f defines a space of matrices $\mathcal{A}$ by evaluating the polynomial on all possible inputs. Crucially for PIT, we see that $\mathrm{span}(\mathcal{A})$ is zero iff f is zero. By breaking the matrix product in f into variable disjoint left- and right-halves (each on $\approx D/2$ variables), we can see that this space of matrices factorizes into $\mathcal{A} = \mathcal{A}_0 \cdot \mathcal{A}_1$, corresponding to the evaluations of the separate halves of the ABP. Now, because these halves are variable disjoint, we can *in parallel* find smaller spaces of matrices such that $\mathrm{span}(\mathcal{A}'_c) = \mathrm{span}(\mathcal{A}_c)$ for $c \in \{0, 1\}$. From the above lemma, we now see that $\mathrm{span}(\mathcal{A}) = \mathrm{span}(\mathcal{A}'_0 \cdot \mathcal{A}'_1)$. The operation "$\mathcal{A}'_0 \cdot \mathcal{A}'_1$" squares the number of matrices under consideration, so to complete the picture we "derandomize" this Kronecker product to yield a "small" family of matrices $\mathcal{A}'$ such that $\mathrm{span}(\mathcal{A}) = \mathrm{span}(\mathcal{A}')$. It follows then that PIT of the original function f reduces to testing each matrix in $\mathcal{A}'$ for non-zeroness, which will correspond to fully interpolating a univariate polynomial.

To implement the above program, our main challenge is to thus construct the derandomized Kronecker product. To do so, our next lemma shows that the span of a layer of an read-once oblivious ABP can be viewed either as the span of the evaluations of that layer, or of the (finite) number of (matrix-valued) coefficients of that layer. We will use both characterizations of the span of a layer.

Lemma 3.3. *Let $M \in \mathbb{F}[x]^{[\![r]\!] \times [\![r]\!]}$. Then for any set $S \subseteq \mathbb{F}$ with $|S| > \deg(M)$, $\mathrm{span}\{M(\alpha)\}_{\alpha \in S} = \mathrm{span}\{\mathfrak{C}_{x^i}(M)\}_{i=0}^{\deg(M)}$.*

Proof. $\subseteq$: Note that $M(\alpha) = \sum_{i=0}^{\deg(M)} \mathfrak{C}_{x^i}(M) \cdot \alpha^i$, and thus is in $\mathrm{span}\{\mathfrak{C}_{x^i}(M)\}_{i=0}^{\deg(M)}$, for any α.

$\supseteq$: Let $d = \deg(M) + 1$. As the span of a set of vectors is monotonic, it suffices to prove the claim when $|S| = d$. Recall that polynomial interpolation shows that the map $\mathbb{F}^{[\![d]\!]} \to \mathbb{F}^{[\![d]\!]}$ defined by mapping degree $< d$ polynomials to their evaluations on S is a bijective linear map. It follows that for any $0 \leq i < d$, there is a set of constants $\{a_{i,\alpha}\}_{\alpha \in S}$ such that for any polynomial $f \in \mathbb{F}[x]$ of degree $< d$, $\mathfrak{C}_{x^i}(f) = \sum_{\alpha \in S} a_{i,\alpha} f(\alpha)$. Applying this relation entry-wise we obtain $\mathfrak{C}_{x^i}(M) = \sum_{\alpha \in S} a_{i,\alpha} M(\alpha)$, and thus $\mathfrak{C}_{x^i}(M) \in \mathrm{span}\{M(\alpha)\}_{\alpha \in S}$. Varying this over all i yields the result. □

We now cite a dimension reduction lemma that shows how we can map a vector space of rank $\leq r$, in a large ambient dimension n, to a vector space of the same rank in an ambient dimension r (and thus, is a "rank extractor"). A lemma of this sort was first established by Gabizon and Raz [GR05]. That lemma would also work for us, but the version we cite has slightly better parameters.

Lemma 3.4 ([FS12])**.** *Let $1 \leq s \leq r \leq n$. Let $M \in \mathbb{F}^{[\![n]\!] \times [\![r']\!]}$ be of rank s, for $r' \geq s$. Let $\mathbb{K}$ be a field extending $\mathbb{F}$, and let $\omega \in \mathbb{K}$ be an element of order $\geq n$. For $\alpha \in \mathbb{K}$ define $A_\alpha \in \mathbb{K}^{[\![r]\!] \times [\![n]\!]}$ by $(A_\alpha)_{i,j} = (\omega^i \alpha)^j$. Then there are $\leq nr - \binom{r+1}{2} < nr$ values $\alpha \in \mathbb{K}$ such that the first s rows of $A_\alpha M$ have rank $< s$.*

We now apply this dimension reduction to read-once oblivious ABPs, by observing that even though the result $M(x)N(x^n)$ of the Kronecker product induces a linear space of transformations most naturally parameterized by the $\approx n^2$ coefficients of this polynomial (as in Lemma 3.3), each transformation is an $r \times r$ matrix, and so this linear space of transformations more naturally lives in an r^2-dimensional vector space. Thus, by applying dimension reduction, we make a first step in getting a derandomized Kronecker product.

Lemma 3.5. *Let $M \in \mathbb{F}[x]^{[\![r]\!]\times[\![r]\!]}$, $N \in \mathbb{F}[y]^{[\![r]\!]\times[\![r]\!]}$ be of degree $< n$. Let ω be an element of order $\geq n^2$. Then, for any $\alpha \in \mathbb{F}$,*

$$\operatorname{span}\{\mathfrak{C}_{x^i y^j}(M(x)N(y))\}_{i,j\in[\![n]\!]} \supseteq \operatorname{span}\{M(\omega^\ell\alpha)N((\omega^\ell\alpha)^n)\}_{\ell\in[\![r^2]\!]}$$

and except for $< n^2r^2$ values of α,

$$\operatorname{span}\{\mathfrak{C}_{x^i y^j}(M(x)N(y))\}_{i,j\in[\![n]\!]} = \operatorname{span}\{M(\omega^\ell\alpha)N((\omega^\ell\alpha)^n)\}_{\ell\in[\![r^2]\!]}$$

Proof. $\supseteq$: This follows from the same reasoning used in Lemma 3.3, in that $M(\omega^\ell\alpha)N((\omega^\ell\alpha)^n) = \sum_{i,j\in[\![n]\!]} \mathfrak{C}_{x^i y^j}(M(x)N(y)) \cdot (\omega^\ell\alpha)^{i+nj}$, and thus the evaluations of $M(x)N(y)$ are in the span of the coefficients of $M(x)N(y)$.

$\subseteq$: Consider the matrix $C \in \mathbb{F}^{[\![n^2]\!]\times[\![r^2]\!]}$ defined so that $C_{i+nj,\bullet} := \mathfrak{C}_{x^i y^j}(M(x)N(y))$, where we flatten $r \times r$ matrices into r^2-dimensional vectors. Observe, that as in the Kronecker substitution, the map $(i,j) \to i+nj$ is a bijection $[\![n]\!]^2 \to [\![n^2]\!]$, and thus the rows of C are exactly the coefficients of $M(x)N(y)$, and thus $\operatorname{span}\{\mathfrak{C}_{x^i y^j}(M(x)N(y))\}_{i,j\in[\![n]\!]} = \text{row-span}(C)$.

Taking $A_\alpha \in \mathbb{F}^{[\![r^2]\!]\times[\![n^2]\!]}$ as defined in Lemma 3.4, the lemma shows that except for $< n^2r^2$ values of α, we have that $\operatorname{rank}(A_\alpha C) = \operatorname{rank}(C)$. Recall that $(A_\alpha)_{\ell,i+nj} = (\omega^\ell\alpha)^{i+nj}$, which implies that

$$(A_\alpha)_{\ell,\bullet}C = \sum_{i,j\in[\![n]\!]} \mathfrak{C}_{x^i y^j}(M(x)N(y)) \cdot (\omega^\ell\alpha)^{i+nj} = M(\omega^\ell\alpha)N((\omega^\ell\alpha)^n),$$

and thus $\operatorname{span}\{M(\omega^\ell\alpha)N((\omega^\ell\alpha)^n)\}_{\ell\in[\![r^2]\!]} = \text{row-span}(A_\alpha C)$. Since $\text{row-span}(C) \supseteq \text{row-span}(A_\alpha C)$, it follows that if $\operatorname{rank}(C) = \operatorname{rank}(A_\alpha C)$, then $\text{row-span}(C) = \text{row-span}(A_\alpha C)$. As Lemma 3.4 shows there are $< n^2r^2$ many α for which $\operatorname{rank}(C) > \operatorname{rank}(A_\alpha C)$, and thus there are $< n^2r^2$ many α for which $\text{row-span}(C) \neq \text{row-span}(A_\alpha C)$, yielding the claim. □

The above result shows how to choose evaluation points for the two matrices M and N such that the span is preserved. However, by itself this does not lead naturally to recursion. Rather, we would like to take $M(x)N(y)$ and produce a pair of curves $f, g \in \mathbb{F}[z]$ satisfying $M(x)N(y) \approx M(f(z))N(g(z))$, such that we can apply recursion and merge the variable z with other variables. To do this, it is enough that f and g pass through all of the evaluation points for M and N in the above lemma, as then this captures the desired span of linear transformations. To create these curves, we need the following definition of the Lagrange interpolation polynomials, which will be featured in our hitting set. Note that these appeared in the prior work of Shpilka and Volkovich [SV09], who also used them in a somewhat similar fashion.

Definition 3.6. *Fix s, and distinct $\beta_i \in \mathbb{F}$, for $i \in [\![s]\!]$. Then the **Lagrange interpolation polynomials (with respect to s and the β_i's)** are the unique polynomials $p_\ell \in \mathbb{F}[t]$ of degree $< s$ such that $p_\ell(\beta_i) = \mathbb{1}_{i=\ell}$ for all $i, \ell \in [\![s]\!]$.*

We now use these polynomials to construct curves passing through the evaluation points shown in Lemma 3.5, to yield a reduction of a two-layer read-once oblivious ABP to a one-layer read-once oblivious ABP. We will phrase the result more generally, so that we can apply induction.

Lemma 3.7. *For $i \in [\![D]\!]$, let $M_i \in \mathbb{F}[x]^{[\![r]\!]\times[\![r]\!]}$, $N_i \in \mathbb{F}[y]^{[\![r]\!]\times[\![r]\!]}$ be of degree $< n$, and $f_i \in \mathbb{F}[x]$, $g_i \in \mathbb{F}[y]$ be of degree $\leq m$. Let $\omega \in \mathbb{F}$ be an element of order $\geq (Dnm)^2$. Let $\beta_j \in \mathbb{F}$ with $j \in [\![r^2]\!]$ be distinct, and let p_ℓ be the corresponding Lagrange interpolation polynomials with respect to r^2*

and the β_j's. Then, except for $< (Dnmr)^2$ values of α, we have

$$\operatorname{span}\left\{\prod_{i\in[\![D]\!]} M_i(f_i(x))\cdot\prod_{i\in[\![D]\!]} N_i(g_i(y))\right\}_{x,y\in\mathbb{F}}$$
$$\subseteq \operatorname{span}\left\{\prod_{i\in[\![D]\!]} M_i\left(\sum_{\ell\in[\![r^2]\!]} f_i(\omega^\ell\alpha)p_\ell(z)\right)\cdot\prod_{i\in[\![D]\!]} N_i\left(\sum_{\ell\in[\![r^2]\!]} g_i((\omega^\ell\alpha)^{Dnm})p_\ell(z)\right)\right\}_{z\in\mathbb{F}}$$

.

Proof. Define $R(x) := \prod_{i\in[\![D]\!]} M_i(f_i(x))$, so that $R \in \mathbb{F}[x]^{[\![r]\!]\times[\![r]\!]}$ is of degree $< Dnm$, and define $T(y) := \prod_{i\in[\![D]\!]} N_i(g_i(y))$, so that $T \in \mathbb{F}[y]^{[\![r]\!]\times[\![r]\!]}$ of degree $< Dnm$. As ω has order $\geq (Dnm)^2$, Lemma 3.5 implies that except for $< (Dnmr)^2$ values of α,

$$\operatorname{span}\{\mathfrak{C}_{x^iy^j}(R(x)T(y))\}_{i,j\in[\![Dnm]\!]} = \operatorname{span}\{R(\omega^\ell\alpha)T((\omega^\ell\alpha)^{Dnm})\}_{\ell\in[\![r^2]\!]}\,. \tag{3.8}$$

As $\omega \in \mathbb{F}$ has order $\geq (Dnm)^2$, it follows that $|\mathbb{F}| \geq (Dnm)^2$, so that applying Lemma 3.3 twice (first to the variable x, then to the variable y) we have that

$$\operatorname{span}\{R(x)T(y)\}_{x,y\in\mathbb{F}} = \operatorname{span}\{\mathfrak{C}_{x^iy^j}(R(x)T(y))\}_{i,j\in[\![Dnm]\!]}\,. \tag{3.9}$$

Now denote

$$U(z) := \prod_{i\in[\![D]\!]} M_i\left(\sum_{\ell\in[\![r^2]\!]} f_i(\omega^\ell\alpha)p_\ell(z)\right) \text{ and } V(z) := \prod_{i\in[\![D]\!]} N_i\left(\sum_{\ell\in[\![r^2]\!]} g_i((\omega^\ell\alpha)^{Dnm})p_\ell(z)\right)\,,$$

which are both elements of $\mathbb{F}[z]^{[\![r]\!]\times[\![r]\!]}$ of degree $< Dnr^2$. Then by construction of the Lagrange interpolation polynomials, we see that $U(\beta_\ell) = R(\omega^\ell\alpha)$ and $V(\beta_\ell) = T((\omega^\ell\alpha)^{Dnm})$, and thus by linearity

$$\operatorname{span}\{R(\omega^\ell\alpha)T((\omega^\ell\alpha)^{Dnm})\}_{\ell\in[\![r^2]\!]} \subseteq \operatorname{span}\{U(z)V(z)\}_{z\in\mathbb{F}}\,. \tag{3.10}$$

Putting equations (3.8), (3.9), and (3.10) together yields the claim. □

Remark 3.11. In the above lemma it is tempting to ask for the "$\subseteq$" to be an "$=$", for in PIT applications we wish to avoid causing a zero polynomial to become a non-zero polynomial (or in this case, avoid causing a zero-dimensional span becoming higher-dimensional). However, the proof does not give this in general, as inserting the $p_\ell(z)$ polynomial *outside* the functions f_i and g_i allows us to potentially evaluate the matrices M_i and N_i at points possibly outside the range of the polynomials f_i and g_i, seemingly allowing the dimension to increase.

Nevertheless, this lemma is enough for constructing a hitting set, as the resulting span is still inside that of $\prod_i M_i(x_i)\cdot\prod_i N_i(y_i)$, which is ultimately the polynomial whose span we are trying to replicate. In such applications, we will start off (by induction) with f_i and g_i such that

$$\operatorname{span}\left\{\prod_{i\in[\![D]\!]} M_i(f_i(x))\cdot\prod_{i\in[\![D]\!]} N_i(g_i(y))\right\} = \operatorname{span}\left\{\prod_{i\in[\![D]\!]} M_i(x_i)\cdot\prod_{i\in[\![D]\!]} N_i(y_i)\right\}\,,$$

and so as the "$\subseteq$" in Lemma 3.7 nests between the two terms in this equality, we will in fact get "$=$" in the lemma in this case, so will not turn zero polynomials into non-zero polynomials.

3.3 The Generator for Read-Once Oblivious ABPs

In this section we construct the hitting set for read-once oblivious ABPs. The hitting set is more naturally presented as a *generator*, which we now define. See also the survey [SY10] for more on generators.

Definition 3.12. *Let $\mathcal{C}$ be a class of circuits computing polynomials on n variables. A polynomial map $\mathcal{G} : \mathbb{F}^{[\![t]\!]} \to \mathbb{F}^{[\![n]\!]}$ is a **generator** for $\mathcal{C}$ if for every polynomial f computed by a circuit in $\mathcal{C}$, $f \equiv 0$ iff $f \circ \mathcal{G} \equiv 0$.*

Given a generator $\mathcal{G}$, one can show that $\mathcal{G}(S^t)$ is a hitting set for $\mathcal{C}$ as long as $|S|$ is polynomially large in the relevant parameters. We present this formally in Lemma 3.21. Given this relation to hitting sets, we now proceed to the construction of our generator, and then prove the requisite properties.

Construction 3.13. *Let $n, r \geq 1$, $d \geq 0$, and $D = 2^d$. Let $\mathbb{F}$ be a field of size $|\mathbb{F}| > (Dnr^3)^2$ and let $\omega \in \mathbb{F}$ be of order $\geq (Dnr^2)^2$. Let $\beta_i \in \mathbb{F}$ with $i \in [\![r^2]\!]$ be distinct, and let $p_\ell \in \mathbb{F}[t]$ be the Lagrange interpolation polynomials with respect to r^2 and the β_i's.*

We define a polynomial map $\mathcal{G}_d : \mathbb{F}^{[\![d+1]\!]} \to \mathbb{F}^{[\![D]\!]}$ as follows. We index the inputs to $\mathcal{G}_d$ by the variables α_i for $i \in [\![d+1]\!]$, and index the outputs of $\mathcal{G}_d$ by bit-vectors $\vec{b} \in \{0,1\}^{[\![d]\!]}$, so that we write $\mathcal{G}_{d,\vec{b}}$ for the output associated with $\vec{b}$.

Define,

$$\mathcal{G}_{d,\vec{b}} := \sum_{\ell_0,\ldots,\ell_{d-1} \in [\![r^2]\!]} \prod_{i \in [\![d]\!]} \left((1-b_i) \cdot p_{\ell_{i-1}}(\omega^{\ell_i}\alpha_i) + b_i \cdot p_{\ell_{i-1}}((\omega^{\ell_i}\alpha_i)^{2^i nr^2}) \right) \cdot p_{\ell_{d-1}}(\alpha_d) \,, \tag{3.14}$$

where we abuse notation and define $p_{\ell_{-1}}(t) = t$. In particular,

$$\mathcal{G}_{0,\vec{b}} := p_{\ell_{-1}}(\alpha_0) = \alpha_0. \tag{3.15}$$

We now establish properties of this construction. We first prove an easy lemma that gives an upper bound on the degrees of the variables α_i in $\mathcal{G}$.

Lemma 3.16 (Bounding the degree)**.** *Assume the setup of Construction 3.13. Let $\vec{b} \in \{0,1\}^{[\![d]\!]}$.*

- *For $i \in [\![d]\!]$, $\deg_{\alpha_i}(\mathcal{G}_{d,\vec{b}}(\vec{\alpha})) \leq Dnr^4$.*
- *For $i = d$, $\deg_{\alpha_d}(\mathcal{G}_{d,\vec{b}}(\vec{\alpha})) \leq r^2 \leq Dnr^4$.*

Proof. Recall that $\deg(p_{\ell_i}) \leq r^2$ (even for $p_{\ell_{-1}}$, as $r \geq 1$). Also, recall that $2^i \leq D$, for $i \in [\![d+1]\!]$. There are two cases, $0 \leq i < d$ and $i = d$. For $0 \leq i < d$, we see that $\deg_{\alpha_i}(\mathcal{G}_{d,\vec{b}}) \leq \deg(p_{\ell_{i-1}}) \cdot 2^i nr^2 \leq Dnr^4$. For $i = d$, we see that $\deg_{\alpha_d}(\mathcal{G}_{d,\vec{b}}) \leq \deg(p_{\ell_{d-1}}) \leq r^2 \leq Dnr^4$, as $D, n, r \geq 1$. □

The next lemma demonstrates the recursive structure of $\mathcal{G}$.

Lemma 3.17 (The recursive structure)**.** *Assume the setup of Construction 3.13. For $\vec{b} \in \{0,1\}^{[\![d-1]\!]}$, $b_{d-1} \in \{0,1\}$, and $\vec{\alpha}$ denoting the variables α_i for $i \in [\![d-1]\!]$,*

$$\mathcal{G}_{d,\vec{b}b_{d-1}}(\vec{\alpha}, \alpha_{d-1}, \alpha_d) = \begin{cases} \sum_{\ell_{d-1} \in [\![r^2]\!]} \mathcal{G}_{d-1,\vec{b}}(\vec{\alpha}, \omega^{\ell_{d-1}}\alpha_{d-1}) p_{\ell_{d-1}}(\alpha_d) & \text{if } b_{d-1} = 0 \\ \sum_{\ell_{d-1} \in [\![r^2]\!]} \mathcal{G}_{d-1,\vec{b}}(\vec{\alpha}, (\omega^{\ell_{d-1}}\alpha_{d-1})^{Dnr^2/2}) p_{\ell_{d-1}}(\alpha_d) & \text{else} \end{cases}.$$

Proof. This claim follows from the definitions, with some rearrangement of terms.

$$
\begin{aligned}
&\mathcal{G}_{d,\vec{b}b_{d-1}}(\vec{\alpha}, \alpha_{d-1}, \alpha_d) \\
&\quad= \sum_{\ell_0,\ldots,\ell_{d-1}\in[\![r^2]\!]} \prod_{i\in[\![d]\!]} \left((1-b_i)\cdot p_{\ell_{i-1}}(\omega^{\ell_i}\alpha_i) + b_i \cdot p_{\ell_{i-1}}((\omega^{\ell_i}\alpha_i)^{2^i nr^2})\right) \cdot p_{\ell_{d-1}}(\alpha_d) \\
&\quad= \sum_{\ell_{d-1}\in[\![r^2]\!]} \sum_{\substack{\ell_i\in[\![r^2]\!] \\ i\in[\![d-1]\!]}} \prod_{i\in[\![d-1]\!]} \left((1-b_i)\cdot p_{\ell_{i-1}}(\omega^{\ell_i}\alpha_i) + b_i \cdot p_{\ell_{i-1}}((\omega^{\ell_i}\alpha_i)^{2^i nr^2})\right) \\
&\qquad\qquad \cdot \left((1-b_{d-1})\cdot p_{\ell_{d-2}}(\omega^{\ell_{d-1}}\alpha_{d-1}) + b_{d-1} \cdot p_{\ell_{d-2}}((\omega^{\ell_{d-1}}\alpha_{d-1})^{2^{d-1} nr^2})\right) \cdot p_{\ell_{d-1}}(\alpha_d) \\
&\quad= \sum_{\ell_{d-1}\in[\![r^2]\!]} ((1-b_{d-1})\cdot \mathcal{G}_{d-1,\vec{b}}(\vec{\alpha}, \omega^{\ell_{d-1}}\alpha_{d-1}) + b_{d-1}\cdot \mathcal{G}_{d-1,\vec{b}}(\vec{\alpha}, (\omega^{\ell_{d-1}}\alpha_{d-1})^{2^{d-1}nr^2}) \cdot p_{\ell_{d-1}}(\alpha_d) \\
&\quad= \begin{cases} \sum_{\ell_{d-1}\in[\![r^2]\!]} \mathcal{G}_{d-1,\vec{b}}(\vec{\alpha}, \omega^{\ell_{d-1}}\alpha_{d-1}) p_{\ell_{d-1}}(\alpha_d) & \text{if } b_{d-1}=0 \\ \sum_{\ell_{d-1}\in[\![r^2]\!]} \mathcal{G}_{d-1,\vec{b}}(\vec{\alpha}, (\omega^{\ell_{d-1}}\alpha_{d-1})^{Dnr^2/2}) p_{\ell_{d-1}}(\alpha_d) & \text{else} \end{cases}
\end{aligned}
$$

□

We now show that the generator can be efficiently computed.

Lemma 3.18 (Efficiency)**.** *Assume the setup of Construction 3.13. For $\vec{b} \in \{0,1\}^{[\![d]\!]}$, $\mathcal{G}_{d,\vec{b}} : \mathbb{F}^{[\![d+1]\!]} \to \mathbb{F}^{[\![D]\!]}$ is computable by a depth $d+1$, width r^2, read-once oblivious ABP[6] in the variables $\vec{\alpha}$, of degree $\le Dnr^4$.*

Proof. For $i \in [\![d+1]\!]$, and $c \in \{0,1\}$ define $A_{i,c} \in \mathbb{F}[\alpha_i]^{[\![r^2]\!]\times[\![r^2]\!]}$, by

$$
(A_{i,0})_{\ell_{i-1},\ell_i}(\alpha_i) := p_{\ell_{i-1}}(\omega^{\ell_i}\alpha_i), \qquad\qquad (A_{i,1})_{\ell_{i-1},\ell_i}(\alpha_i) := p_{\ell_{i-1}}((\omega^{\ell_i}\alpha_i)^{2^i nr^2})
$$

for $\ell_{i-1}, \ell_i \in [\![r^2]\!]$, where ℓ_{-1} and ℓ_d are ranged over as well, and $p_{\ell_{-1}}$ is still the identity polynomial. Then we have that

$$
\begin{aligned}
\mathcal{G}_{d,\vec{b}}(\vec{\alpha}) &= \sum_{\ell_0,\ldots,\ell_{d-1}\in[\![r^2]\!]} \prod_{i\in[\![d]\!]} \left((1-b_i)\cdot p_{\ell_{i-1}}(\omega^{\ell_i}\alpha_i) + b_i \cdot p_{\ell_{i-1}}((\omega^{\ell_i}\alpha_i)^{2^i nr^2})\right) \cdot p_{\ell_{d-1}}(\alpha_d) \\
&= \sum_{\substack{\ell_{-1}=\ell_d=0 \\ \ell_0,\ldots,\ell_{d-1}\in[\![r^2]\!]}} \prod_{i\in[\![d]\!]} \left((1-b_i)\cdot A_{i,0}(\alpha_i)_{\ell_{i-1},\ell_i} + b_i \cdot A_{i,1}(\alpha_i)_{\ell_{i-1},\ell_i}\right) \cdot A_{d,0}(\alpha_d)_{\ell_{d-1},\ell_d} \\
&= \left(\left[\prod_{i\in[\![d]\!]} ((1-b_i)\cdot A_{i,0}(\alpha_i) + b_i \cdot A_{i,1}(\alpha_i)) \right] \cdot A_{d,0}(\alpha_d) \right)_{0,0}
\end{aligned}
$$

by the properties of matrix multiplication. By Lemma 3.1 we see that this matrix product can be computed by a width r^2, depth $d+1$, read-once oblivious ABP, and the degree bound follows from Lemma 3.16. □

We now turn to proving that our construction is indeed a generator for read-once oblivious ABPs. In particular, we first show the generator preserves span.

[6] That is, we are constructing a generator for read-once oblivious ABPs, and each coordinate is computed by an read-once oblivious ABP on many fewer variables. It is unclear at the moment whether this is amenable to recursion, since read-once oblivious ABPs are not closed under composition.

Lemma 3.19 (Span preserving)**.** *Assume the setup of Construction 3.13. Let $M_{\vec{b}} \in \mathbb{F}[x_{\vec{b}}]^{[\![r]\!]\times[\![r]\!]}$ for $\vec{b} \in \{0,1\}^{[\![d]\!]}$ be of degree $< n$. Then, denoting $\vec{\alpha}$ for the vector of variables $\alpha_0, \ldots, \alpha_d$,*

$$\operatorname{span}\left\{\prod_{\vec{b}\in\{0,1\}^{[\![d]\!]}} M_{\vec{b}}\left(x_{\vec{b}}\right)\right\}_{\vec{x}\in\mathbb{F}^{[\![D]\!]}} = \operatorname{span}\left\{\prod_{\vec{b}\in\{0,1\}^{[\![d]\!]}} M_{\vec{b}}\left(\mathcal{G}_{d,\vec{b}}(\vec{\alpha})\right)\right\}_{\vec{\alpha}\in\mathbb{F}^{[\![d+1]\!]}}. \tag{3.20}$$

Proof. We prove the claim by induction on d.

$\underline{d=0}$**:** Observe that for $d = 0$, $\{0,1\}^{[\![d]\!]} = \{\lambda\}$, where λ is the empty string. Further, we have that $\mathcal{G}_{0,\lambda} : \mathbb{F} \to \mathbb{F}$ is simply the identity map, by definition of $p_{\ell_{-1}}$. Thus, Equation 3.20 is stating that,

$$\operatorname{span}\{M_\lambda(x_\lambda)\}_{x_\lambda\in\mathbb{F}} = \operatorname{span}\{M_\lambda(\mathcal{G}_{0,\lambda}(\alpha_0))\}_{\alpha_0\in\mathbb{F}} = \operatorname{span}\{M_\lambda(\alpha_0)\}_{\alpha_0\in\mathbb{F}}$$

and is thus trivially true.

$\underline{d>0}$**:** We will split the span into the multiplication of two different spans, and then appeal to Lemma 3.7 to merge these two spans.

Specifically, denote

$$\mathcal{M} := \left\{\prod_{\vec{b}\in\{0,1\}^{[\![d]\!]}} M_{\vec{b}}\left(x_{\vec{b}}\right)\right\}_{\vec{x}\in\mathbb{F}^{[\![D]\!]}}$$

and for $c \in \{0,1\}$ denote

$$\mathcal{M}_c := \left\{\prod_{\vec{b}\in\{0,1\}^{[\![d-1]\!]}} M_{\vec{b}c}\left(x_{\vec{b}c}\right)\right\}_{\vec{x}\in\mathbb{F}^{[\![D/2]\!]}}.$$

Similarly, for $c \in \{0,1\}$, and $\vec{\alpha}$ being the vector of variables α_i for $i \in [\![d-1]\!]$, denote

$$\mathcal{M}' := \left\{\prod_{\vec{b}\in\{0,1\}^{[\![d]\!]}} M_{\vec{b}}\left(\mathcal{G}_{d,\vec{b}}\left(\vec{\alpha},\alpha_{d-1},\alpha_d\right)\right)\right\}_{\vec{\alpha}\alpha_{d-1}\alpha_d\in\mathbb{F}^{[\![d+1]\!]}}$$

and

$$\mathcal{M}'_c := \left\{\prod_{\vec{b}\in\{0,1\}^{[\![d-1]\!]}} M_{\vec{b}c}\left(\mathcal{G}_{d-1,\vec{b}c}\left(\vec{\alpha},\alpha_{d-1}\right)\right)\right\}_{\vec{\alpha}\alpha_{d-1}\in\mathbb{F}^{[\![d]\!]}}.$$

It follows from definition that $\mathcal{M} = \mathcal{M}_0 \cdot \mathcal{M}_1$, and by induction, we have that $\operatorname{span}\mathcal{M}_c = \operatorname{span}\mathcal{M}'_c$. Thus, Lemma 3.2 implies that $\operatorname{span}\mathcal{M} = \operatorname{span}(\operatorname{span}(\mathcal{M}'_0)\cdot\operatorname{span}(\mathcal{M}'_1)) = \operatorname{span}(\mathcal{M}'_0\cdot\mathcal{M}'_1)$. Observe that $\operatorname{span}\mathcal{M}' \subseteq \operatorname{span}\mathcal{M} = \operatorname{span}(\mathcal{M}'_0\cdot\mathcal{M}'_1)$, so the claim will follow from showing that $\operatorname{span}(\mathcal{M}'_0\cdot\mathcal{M}'_1) \subseteq \operatorname{span}\mathcal{M}'$.

We now show that $\operatorname{span}(\mathcal{M}'_0\cdot\mathcal{M}'_1) \subseteq \operatorname{span}\mathcal{M}'$. We will prove this claim for each fixed value of the variables $\vec{\alpha}$. Thus, the matrices in each of $\mathcal{M}'_c$ are parameterized only by the variable α_{d-1}. Further, each matrix in $\mathcal{M}'_c$ is the product of $D/2$ matrices each of the form $M(f(\alpha_{d-1}))$, for $M \in \mathbb{F}[\alpha_{d-1}]^{[\![r]\!]\times[\![r]\!]}$ of degree $< n$ and $f \in \mathbb{F}[\alpha_{d-1}]$ of degree $\leq r^2$ (where the degree bound is by

Lemma 3.16). As the order of ω is $\geq (Dnm)^2$, and $|\mathbb{F}| \geq (Dnmr)^2$ (taking $m = r^2$) we can apply Lemma 3.7 to see that there is some[7] value $\hat{\alpha}_{d-1} \in \mathbb{F}$ such that

$$\begin{aligned}
&\operatorname{span}\left\{ \prod_{\vec{b}\in\{0,1\}^{[\![d-1]\!]}} M_{\vec{b}0}\left(\mathcal{G}_{d-1,\vec{b}0}\left(\vec{\alpha},\alpha_{d-1}\right)\right) \cdot \prod_{\vec{b}\in\{0,1\}^{[\![d-1]\!]}} M_{\vec{b}1}\left(\mathcal{G}_{d-1,\vec{b}1}\left(\vec{\alpha},\alpha_{d-1}\right)\right) \right\}_{\alpha_{d-1}\in\mathbb{F}} \\
&\quad\subseteq \operatorname{span}\left\{ \prod_{\vec{b}\in\{0,1\}^{[\![d-1]\!]}} M_{\vec{b}0}\left(\sum_{\ell_{d-1}\in[\![r^2]\!]} \mathcal{G}_{d-1,\vec{b}0}\left(\vec{\alpha},\omega^{\ell}_{d-1}\hat{\alpha}_{d-1}\right) p_{\ell_{d-1}}(\alpha_d)\right) \right. \\
&\qquad\left. \cdot \prod_{\vec{b}\in\{0,1\}^{[\![d-1]\!]}} M_{\vec{b}1}\left(\sum_{\ell_{d-1}\in[\![r^2]\!]} \mathcal{G}_{d-1,\vec{b}1}\left(\vec{\alpha},(\omega^{\ell_{d-1}}\hat{\alpha}_{d-1})^{Dnr^2/2}\right) p_{\ell_{d-1}}(\alpha_d)\right) \right\}_{\alpha_d\in\mathbb{F}}
\end{aligned}$$

and rewriting this via Lemma 3.17,

$$\begin{aligned}
&= \operatorname{span}\left\{ \prod_{\vec{b}\in\{0,1\}^{[\![d-1]\!]},c\in\{0,1\}} M_{\vec{b}c}\left(\mathcal{G}_{d,\vec{b}c}(\vec{\alpha},\hat{\alpha}_{d-1},\alpha_d)\right) \right\}_{\alpha_d\in\mathbb{F}} \\
&\subseteq \operatorname{span}\left\{ \prod_{\vec{b}\in\{0,1\}^{[\![d-1]\!]},c\in\{0,1\}} M_{\vec{b}c}\left(\mathcal{G}_{d,\vec{b}c}(\vec{\alpha},\alpha_{d-1},\alpha_d)\right) \right\}_{\alpha_{d-1},\alpha_d\in\mathbb{F}} .
\end{aligned}$$

Thus taking this for each $\vec{\alpha}$ we get $\operatorname{span}(\mathcal{M}'_0 \cdot \mathcal{M}'_1) \subseteq \operatorname{span}\mathcal{M}'$, implying $\operatorname{span}\mathcal{M}' = \operatorname{span}\mathcal{M}$ as desired. □

The next lemma concludes that the span-preservation property shows that we indeed have a generator, and can thus construct a hitting set.

Lemma 3.21. *Assume the setup of Construction 3.13. Let $M_{\vec{b}} \in \mathbb{F}[x_{\vec{b}}]^{[\![r]\!]\times[\![r]\!]}$ for $\vec{b} \in \{0,1\}^{[\![d]\!]}$ be of degree $< n$, and let $f(\vec{x}) = \left(\prod_{b\in\{0,1\}^{[\![d]\!]}} M_{\vec{b}}(x_{\vec{b}})\right)_{0,0}$. Let $S \subseteq \mathbb{F}$, with $|S| \geq Dn^2r^4$. Then $f \equiv 0$ iff $f \circ \mathcal{G}_d \equiv 0$ iff $f|_{\mathcal{G}_d(S^{d+1})} \equiv 0$.*

Proof. As f is the $(0,0)$-entry of the matrix product $\prod_{b\in\{0,1\}^{[\![d]\!]}} M_{\vec{b}}(x_{\vec{b}})$, and this projection operator is linear, we see that Equation 3.19 implies that $\operatorname{span}\{f(\vec{x})\}_{\vec{x}\in\mathbb{F}^{[\![D]\!]}} = \operatorname{span}\{f(\mathcal{G}_d(\vec{\alpha}))\}_{\vec{\alpha}\in\mathbb{F}^{[\![d+1]\!]}}$, and as these spans are simply constant multiples of the output of f, we see that $f \equiv 0$ iff $f \circ \mathcal{G}_d \equiv 0$. Next, we observe that for $i \in [\![d+1]\!]$, $\deg_{\alpha_i}(f \circ \mathcal{G}_d) < Dn^2r^4$ using that f has individual degrees $< n$, and invoking the degree bound in Lemma 3.16. Thus, by basic (multivariate) polynomial interpolation, $f \circ \mathcal{G}_d \equiv 0$ iff $(f \circ \mathcal{G}_d)|_{S^{d+1}} \equiv 0$ iff $f|_{\mathcal{G}_d(S^{d+1})} \equiv 0$. □

We can now prove our main theorem.

Theorem 3.22 (PIT for read-once oblivious ABPs)**.** *Let $\mathcal{C}$ be the set of polynomials $f : \mathbb{F}^{[\![D]\!]} \to \mathbb{F}$ computable by a width r, depth D, individual degree $< n$ read-once oblivious ABP. If $|\mathbb{F}| \geq (2Dnr^3)^2$, then $\mathcal{C}$ has a $\mathsf{poly}(D,n,r)$-explicit hitting set, of size $\leq (2Dn^2r^4)^{\lceil \lg D\rceil+1}$.*

[7] Lemma 3.7 shows that there are $< (Dnmr)^2 = D^2n^2r^6$ possible bad values of $\hat{\alpha}_{d-1}$, but in Lemma 3.21 we show that there are "really" only Dn^2r^4 many bad values. It is unclear at present how to directly improve Lemma 3.7 to reflect this better bound.

Proof. By Lemma 3.1, any such read-once oblivious ABP computes a polynomial f expressible as $f(\vec{x}) = (\prod_{i \in [\![D]\!]} M_i(x_i))_{(0,0)}$, where $M_i(x_i) \in \mathbb{F}[x_i]^{[\![r]\!] \times [\![r]\!]}$ is of degree $< n$. Further, we can round D up to $2^{\lceil \lg D \rceil}$, and by setting $M_i(x_i) = \mathrm{I}$ for $i \geq D$ (ie. padding the product with identity matrices in new variables), we get that $f(\vec{x}) = (\prod_{i \in [\![2^{\lceil \lg D \rceil}]\!]} M_i(x_i))_{0,0}$.

We now show that there is some element $\omega \in \mathbb{F}$ of order $\geq (2^{\lceil \lg D \rceil} nr^2)^2$, that can be found in $\mathsf{poly}(D, n, r)$-time. As there at most d elements such that $x^d = 1$ in $\mathbb{F}$, it follows that there are at most $< (2Dnr^2)^4$ many non-zero elements of order $< (2Dnr^2)^2$, so any enumeration of non-zero elements in $\mathbb{F}$ will find such an element of large order, if $(2Dnr^2)^4 < |\mathbb{F}|$. If $(2Dnr^2)^4 \geq |\mathbb{F}|$, then recalling that the multiplicative group of a finite field $\mathbb{F}$ is cyclic, we see there is an element $g \in \mathbb{F}$ of order $|\mathbb{F}| - 1 \geq (2Dnr^3)^2 - 1 \geq (2^{\lceil \lg D \rceil} nr^3)^2$ (using that $2D > 2^{\lceil \lg D \rceil}$), and thus an enumeration can also find such an ω. Similarly, we can find r^2 distinct β_i, and can find some set $S \subseteq \mathbb{F}$ of size $2^{\lceil \lg D \rceil} n^2 r^4$. With ω, the β_i's and S, we see then, by Lemma 3.21, that $\mathcal{G}_d(S^{\lceil \lg D \rceil + 1})$ is a hitting set for f and has the desired size. Further, for any fixed $\vec{s} \in S^{\lceil \lg D \rceil + 1}$, $\mathcal{G}_d(\vec{s})$ can be computed in $\mathsf{poly}(D, n, r)$ time as each coordinate is computed by a small ABP, as shown in Lemma 3.18. □

We note that Theorem 3.23 is an immediate corollary.

Theorem 3.23 (PIT for set-multilinear ABPs)**.** *Let $\vec{X} = \sqcup_{i \in [\![D]\!]} \vec{X}_i$ where $\vec{X}_i = \{x_{i,j}\}_{j \in [\![n]\!]}$ be a known partition. Let $\mathcal{C}$ be the set of set-multilinear polynomials $f(\sqcup_{i \in [\![D]\!]} \vec{X}_i) : \mathbb{F}^{[\![nD]\!]} \to \mathbb{F}$ computable by a width r, depth D, set-multilinear ABP. If $|\mathbb{F}| \geq (2Dnr^3)^2$, then $\mathcal{C}$ has a $\mathsf{poly}(D, n, r)$-explicit hitting set, of size $\leq (2Dn^2r^4)^{\lceil \lg D \rceil + 1}$.*

Proof. By the degree bounds, the Kronecker map $x_{i,j} \leftarrow x_i^j$ induces a bijection among the monomials, and takes the original set-multilinear ABP to an read-once oblivious ABP, where we replace the set of variables $\vec{X}_i$ with the single (new) variable x_i. Structurally, this is still an width r, depth D ABP, and as we index from zero, the individual degrees are $< n$. Appealing to Theorem 3.22 gives the result. □

4 Non-commutative ABPs

In this section, we show how to do black-box identity testing of non-commutative ABPs in quasi-polynomial time. A non-commutative ABP is structurally the same as defined in Definition 1.1, but we now do not assume the variables commute. Formally, it is an ABP over the ring $\mathbb{F}\{\vec{x}\}$, of polynomials in non-commuting variables. Nisan [Nis91] first explored this model, noting that any commutative polynomial f can be computed non-commutatively (for example, by the complete monomial expansion) and thus non-commutative models of computation form a restricted class of computation we can explore. In the same work, he proved the first exponential lower bounds in the non-commutative ABP model for the (commutative) determinant and permanent polynomials. Later, Raz and Shpilka [RS05] gave a polynomial time PIT algorithm in the white-box model for this class.

In both of the above works, non-commutativity can be seen as "syntactic" in the sense that one can treat the variables as formally non-commutating free variables, and one doesn't seek to substitute values for these variables. However, black-box PIT of non-commutative polynomials in $\mathbb{F}\{\vec{x}\}$ by definition requires such a substitution. Such a substitution will occur in some ring $\mathcal{R}$ that is an $\mathbb{F}$-algebra (that is, there is a (commutative) ring homomorphism[8] $\mathbb{F} \to \mathcal{R}$). Clearly, $\mathcal{R}$

[8]We need this homomorphism to make sense of how elements of $\mathbb{F}$ act on elements of $\mathcal{R}$. That is, when evaluating a non-commutative polynomial f in $\mathbb{F}\{\vec{x}\}$ on elements of $\mathcal{R}$, we replace the coefficients in f with their images under this homomorphism.

must be non-commutative in order to witness the non-identity $xy - yx$ in $\mathbb{F}\{\vec{x}\}$. Trivially, we could choose "$\mathcal{R} = \mathbb{F}\{\vec{x}\}$" and get a black-box PIT algorithm that only requires a single query, but this simply pushes the complexity of the problem into the operations in $\mathcal{R}$. That is, one would want the black-box PIT queries to be efficiently implementable given white-box access to the circuit. As such, it seems natural to ask that $\mathcal{R}$ is a finite dimensional $\mathbb{F}$-vector space, and that the number of dimensions is polynomial in the relevant parameters. Further, another natural restriction is to take $\mathcal{R} = \mathbb{F}^{[\![m]\!]\times[\![m]\!]}$, the algebra of matrices[9], and that is what we do here.

However, once we have moved from evaluations in $\mathbb{F}\{\vec{x}\}$ to evaluations in $\mathcal{R}$, there is the concern that we have lost information, in that $f \in \mathbb{F}\{\vec{x}\}$ could vanish on all possible evaluations over $\mathcal{R}$. Note that this is also a problem in the commutative case, as in the standard example of the polynomial $x^2 - x \in \mathbb{F}_2[x]$ which vanishes over the field $\mathbb{F}_2$. In the case of matrices, which is the ring we shall work over, there are such identities given by the Amitsur-Levitzki theorem [AL50]: the polynomial $\sum_{\sigma\in S_{2m}} \operatorname{sgn}(\sigma) \prod_{i\in[\![2m]\!]} A_{\sigma(i)}$, where S_{2m} is the symmetric group on $2m$ elements, vanishes on every choice of $m \times m$ matrices A_i, over any field.

However, recall that in the commutative case, (multivariate) polynomial interpolation states that for a polynomial $f \in \mathbb{F}[\vec{x}]$ of (total) degree $< d$, f cannot vanish on all evaluations over $\mathbb{F}$ as long as $|\mathbb{F}| \geq d$. Extending this, the Schwartz-Zippel lemma shows that if $|\mathbb{F}| \gg d$ then f has a very low probability of vanishing on a random evaluation over $\mathbb{F}$. This result, applied via a union bound in a probabilistic argument, shows that efficient black-box PIT is (existentially) possible for small ABPs. In almost direct analogy, the Amitsur-Levitzki theorem shows that polynomials of (total) degree $< 2m$ in $\mathbb{F}\{\vec{x}\}$ cannot be identities over $\mathbb{F}^{[\![m]\!]\times[\![m]\!]}$. Bogdanov and Wee [BW05] observed that this result, in combination with the Schwartz-Zippel lemma, show that if $|\mathbb{F}|, m \gg d$ then a non-commutative polynomial f has a very low probability of vanishing on a random evaluation over $\mathbb{F}^{[\![m]\!]\times[\![m]\!]}$. And thus, as in the commutative case, this establishes existence of small hitting sets for non-commutative polynomials computed by small ABPs. Given this existence argument, we now proceed to construct quasi-polynomial sized, explicit hitting sets, for non-commutative ABPs.

We proceed in a fashion similar to Bogdanov and Wee [BW05], by giving a family of matrices with entries consisting of commutative variables, such that any non-zero non-commutative polynomial f over these matrices induces a non-zero family of commutative polynomials over the commutative variables. Further, we show that if f is computable by a small non-commutative ABP, then the resulting commutative polynomials are computable by small set-multilinear ABPs. We can then appeal to the hitting set for this class of polynomials, shown in Theorem 3.23. We first illustrate the construction of our matrices.

Construction 4.1. *Define $\varphi : \mathbb{F}\{x_i\}_{i\in[\![n]\!]} \to \mathbb{F}[x_{i,j}]_{i\in[\![n]\!],j\in\mathbb{N}}$ to be the unique $\mathbb{F}$-linear map defined by $\varphi(1) = 1$, and*

$$\varphi(x_{i_1}x_{i_2}\cdots x_{i_d}) := x_{i_1,1}x_{i_2,2}\cdots x_{i_d,d}\ .$$

For $i \in [\![n]\!]$, $D \geq 0$, define the upper-triangular matrix $X_{i,D} \in (\mathbb{F}[x_{i,j}]_{j\in\mathbb{N}})^{[\![D+1]\!]\times[\![D+1]\!]}$ by

$$(X_{i,D})_{k,\ell} := \begin{cases} x_{i,\ell} & \text{if } k+1 = \ell \\ 0 & \text{else} \end{cases}\ .$$

Define the vector $\vec{X}_D := (X_{i,D})_{i\in[\![n]\!]}$.

[9]The relevant homomorphism $\mathbb{F} \to \mathbb{F}^{[\![m]\!]\times[\![m]\!]}$ maps $a \mapsto a\mathrm{I}$, where I is the identity matrix.

Thus, pictorially, we have that the matrices $X_{i,D}$ look as follows.

$$X_{i,D} = \begin{bmatrix} 0 & x_{i,1} & 0 & 0 & \cdots & 0 \\ 0 & 0 & x_{i,2} & 0 & \cdots & 0 \\ 0 & 0 & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & x_{i,D-1} & 0 \\ 0 & 0 & \cdots & 0 & 0 & x_{i,D} \\ 0 & 0 & \cdots & 0 & 0 & 0 \end{bmatrix} .$$

These matrices are quite similar to the matrices used in [BW05] (see their Lemma 3.2), but they achieve a smaller dimension (in fact, matching the best possible, as seen by the Amitsur-Levitzki theorem). However, their construction does not seem to give the reduction to set-multilinear computation that we need. We now establish properties of our construction. In particular, we relate the evaluations of f on the matrices X_i, to the commutative polynomial $\varphi(f)$.

Lemma 4.2. *Assume the setup of Construction 4.1. Let $f \in \mathbb{F}\{x_i\}_{i\in[\![n]\!]}$ be a non-commutative polynomial. For $D \geq \ell \geq 0$,*

$$(f(\vec{X}_D))_{0,\ell} = \varphi(H_\ell(f)) .$$

Proof. By $\mathbb{F}$-linearity (of φ, $H_\ell(\bullet)$, and $(\bullet)_{0,\ell}$), it is enough to prove the claim for monomials $f(\vec{x}) = x_{i_1}x_{i_2}\cdots x_{i_d}$ for any $d \geq 0$. For $d = 0$, the only monomial is the identity matrix I, for which the claim is clear.

Now consider $d > 0$. Observe that

$$(X_{i_1}\cdots X_{i_d})_{0,\ell} = \sum_{\substack{\ell_0=0,\ell_d=\ell \\ \ell_1,\ldots,\ell_{d-1}\in[\![D+1]\!]}} (X_{i_1})_{\ell_0,\ell_1}(X_{i_2})_{\ell_1,\ell_2}\cdots(X_{i_d})_{\ell_{d-1},\ell_d}$$

and by construction, we see that if $\ell_{j-1}+1 \neq \ell_j$ then $(X_{i,j})_{\ell_{j-1},\ell_j} = 0$, and so,

$$\begin{aligned} &= \sum_{\substack{\ell_0=0,\ell_d=\ell \\ \ell_1,\ldots,\ell_{d-1}\in[\![D+1]\!] \\ \ell_{j-1}+1=\ell_j}} (X_{i_1})_{\ell_0,\ell_1}(X_{i_2})_{\ell_1,\ell_2}\cdots(X_{i_d})_{\ell_{d-1},\ell_d} \\ &=^{\dagger} \begin{cases} x_{i_1,1}\cdots x_{i_d,d} & d = \ell \\ 0 & \text{else} \end{cases} \\ &= \varphi(H_\ell(x_{i_1}\cdots x_{i_d})) \end{aligned}$$

as desired, where (†) follows from the observation that the only way a sequence of length $d+1$ can start at 0 and end at ℓ, increasing 1 at each step, is if $d = \ell$, and $\ell_j = j$. □

We now use this to give the black-box reduction from non-commutative polynomials to set-multilinear polynomials.

Lemma 4.3. *Assume the setup of Construction 4.1. Let $f \in \mathbb{F}\{x_i\}_{i\in[\![n]\!]}$ be a non-commutative polynomial of degree $\leq D$. Then for each $\ell \in [\![D+1]\!]$, the (commutative) polynomial $(f(\vec{X}_D))_{0,\ell}$ is set-multilinear, and $f \equiv 0$ iff for all $\ell \in [\![D+1]\!]$, $(f(\vec{X}_D))_{0,\ell} \equiv 0$. Consequently, $f \equiv 0$ iff $f(\vec{X}_D) \equiv 0$.*

Proof. As $\deg(f) \leq D$, it is clear that $f \equiv 0$ iff for all $\ell \in [\![D+1]\!]$, $H_\ell(f) \equiv 0$. Further, we see that φ is a bijection, so $H_\ell(f) \equiv 0$ iff $\varphi(H_\ell(f)) \equiv 0$, and that φ applied to any homogeneous non-commutative polynomial yields a set-multilinear polynomial. Finally, we use Lemma 4.2 to see that $\varphi(H_\ell(f)) \equiv 0$ iff $(f(\vec{X}_D))_{0,\ell} \equiv 0$. It follows then that $f(\vec{x}) \equiv 0$ iff $f(\vec{X}_D) \equiv 0$. □

Thus, this lemma says that to do black-box PIT of non-commutative polynomials, it is enough to do black-box PIT of set-multilinear polynomials, in order to hit each of the $(f(\vec{X}_D))_{0,\ell}$. However, in order for this reduction to be useful, we must show that $(f(\vec{X}_D))_{0,\ell}$ is not only set-multilinear, but is also computed by a small computational device, as otherwise no small hitting set may exist.

In the most general model, that of non-commutative circuits, it is not clear there is a simple bound on the complexity of $(f(\vec{X}_D))_{0,\ell}$ in terms of the complexity of f, as circuits have no global "order" in the computation. That is, in a non-commutative circuit each multiplication is ordered, and this local order is fixed by the circuit. However, as parts of the circuit may be reused, a partial computation in the circuit can appear in many different parts of the final computation. Concretely, in the circuit of the repeated squaring computation $((x)^2)^2$, we cannot assign "$x \to x_i$" in any way compatible with the fact that $\varphi(x^4) = x_1x_2x_3x_4$. Thus, it is not clear how to convert a non-commutative circuit for f into a circuit for $\varphi(H_\ell(f))$.

However, for weaker models of computation such as ABPs, there is such an notion of ordering as each multiplication proceeds from the source to the sink, in a single direction. This allows us to relate the ABP complexity of a (non-commutative) polynomial f to the ABP complexity of the (commutative) set-multilinear polynomial $\varphi(H_\ell(f))$. To do so, we first cite the following lemma, implicit in Nisan [Nis91] (see also Lemma 2 in Raz-Shpilka [RS05][10]).

Lemma 4.4 (Nisan, [Nis91])**.** *Let $f \in \mathbb{F}\{\vec{x}\}$ be a non-commutative polynomial computed by a non-commutative ABP of depth D and width r. Then for[11] $1 \leq \ell \leq D$, $H_\ell(f)$ is[12] computed by a non-commutative ABP of depth ℓ and width $\leq r(D+1)$. Further, each edge in this ABP is labeled with a homogeneous linear form.*

We now show that for ABPs as the above lemma outputs (those with only homogeneous linear forms on the edges) there is a clear connection between the ABP complexity of f and $\varphi(f)$.

Lemma 4.5. *Let $f \in \mathbb{F}\{\vec{x}\}$ be a non-commutative polynomial computed by an ABP A, whose edges are labelled with homogeneous linear forms. Let A' be the ABP with the same structure as A, but for each edge from layer $j-1$ to layer j with label $\sum_{i \in [\![n]\!]} a_i x_i$, we replace the label with $\sum_{i \in [\![n]\!]} a_i x_{i,j}$. Then A' computes $\varphi(f)$, and A' is a set-multilinear ABP.*

Proof. By linearity, it is enough to prove the claim for a single path in the ABP A. The path computes the product of its labels, and thus yields the product of linear forms

$$g = \prod_j \sum_{i \in [\![n]\!]} a_{i,j} x_i = \sum_{i_k \in [\![n]\!]} (\prod_j a_{i_j,j})(\prod_j x_{i_j}) \, .$$

Similarly, the same path in A' computes

$$\prod_j \sum_{i \in [\![n]\!]} a_{i,j} x_{i,j} = \sum_{i_k \in [\![n]\!]} (\prod_j a_{i_j,j})(\prod_j x_{i_j,j})$$

[10] In fact, the version that we use is slightly stronger than the version in [RS05], but the proof is the same and the only difference is that we use a less wasteful analysis, which is obvious from the proof there.

[11] The case when $\ell = 0$ does not fit into this technicalities of this lemma, so will be treated separately in what follows.

[12] In fact, these ABPs can efficiently computed given the original ABP, but we do not use this fact.

which equals $\varphi(g)$ by linearity, as desired. Finally, we see that the edges from layer $j-1$ to layer j only involve the variables $x_{\bullet,j}$, so the ABP is indeed set-multilinear. □

Combining the above results, we can now give the full reduction.

Theorem 4.6. *Assume the setup of Construction 4.1. Let $\mathcal{H}$ be a hitting set for set-multilinear (commutative) polynomials with the (known) variable partition $\sqcup_{j\in[\![D]\!]}\{x_{i,j}\}_{i\in[\![n]\!]}$ computed by set-multilinear ABPs of width $\leq r(D+1)$ and depth $\leq D$. Define $\mathcal{H}' \subseteq (\mathbb{F}^{[\![D+1]\!]\times[\![D+1]\!]})^{[\![n]\!]}$ by replacing each evaluation point in $\mathcal{H}$ with the corresponding evaluation point defined by the matrices $\vec{X}_D$. Then $\mathcal{H}$ is a hitting set for non-commutative ABPs of width r and depth D.*

Proof. Let f be computed by a non-commutative ABPs of width r and depth D. Then, by Lemma 4.3, $f \equiv 0$ iff for all $\ell \in [\![D+1]\!]$, the set-multilinear polynomial $(f(\vec{X}_D))_{0,\ell} \equiv 0$. If $\ell = 0$, then we can observe that as the X_i are strictly upper-triangular, it happens that $(f(\vec{X}_D))_{0,0}$ is constant for any setting of $\vec{X}_D$ (and in fact equals the constant term of f), so it is then clear that $(f(\vec{X}_D))_{0,0} \equiv 0$ iff $(f(\vec{X}_D))_{0,0}|_{\vec{X}_D\in\mathcal{H}'} \equiv 0$ as $|\mathcal{H}'| > 0$.

If $\ell > 0$ then we use Lemma 4.2 and Lemma 4.3 to see that $(f(\vec{X}_D))_{0,\ell} = \varphi(H_\ell(f))$ is a set-multilinear polynomial, and Lemma 4.4 and Lemma 4.5 imply that $\varphi(H_\ell(f))$ is computed by a width $\leq rD$, depth $\leq D$ set-multilinear ABP, and thus $(f(\vec{X}_D))_{0,\ell} \equiv 0$ iff $(f(\vec{X}_D))_{0,\ell}|_{\vec{X}_D\in\mathcal{H}'} \equiv 0$. □

Plugging in our hitting set for set-multilinear ABPs from Theorem 3.23, we obtain the following corollary.

Corollary 4.7 (PIT for non-commutative ABPs)**.** *Let $\mathcal{NC}$ be the set of n-variate non-commutative polynomials computable by width r, depth D ABPs. If $|\mathbb{F}| \geq (2(D+1)^5nr^4)^2$, then $\mathcal{NC}$ has a $\mathsf{poly}(D,n,r)$-explicit hitting set over $(D+1)\times(D+1)$ matrices, of size $\leq (2(D+1)^4n^2r^3)^{\lceil \lg D\rceil+1}$.*

Remark 4.8. The use of $(D+1)\times(D+1)$ strictly upper-triangular matrices in the above results is optimal, in the sense that if we used $m\times m$ strictly upper-triangular matrices for $m < D+1$ then no such result is possible, as any such matrix X has $X^D = 0$, and the monomial x^D is computable by a depth D ABP.

5 Diagonal Circuits

The model of diagonal circuits was first defined by Saxena [Sax08] in order to better understand depth-4 circuits. Much previous PIT research focused on small-depth circuits with restricted top fan-in. In contrast, the diagonal model allows unbounded top fan-in, but each multiplication gate only has few distinct factors (but the factors can have high multiplicity).

To ease reading we shall make use of the following shortened notation. Given a vector $\vec{e} \in \mathbb{N}^n$, we denote $|\vec{e}|_1 := e_1 + \cdots + e_n$, $|\vec{e}|_\infty = \max_i e_i$, $|\vec{e}|_\times = \prod_{i\in[n]}(1+e_i)$, and $\vec{e}! = e_1!\cdots e_n!$.

For depth-3 diagonal circuits, each multiplication gate has the form $\vec{\ell}(\vec{x})^{\vec{e}}$, for affine linear forms ℓ_i, where $|\vec{e}|_\times$ is bounded by $\mathsf{poly}(n)$. Saxena's techniques also extend to the case when the ℓ_i are sum of univariate polynomials, which is the depth-4 diagonal circuit model. We now give a formal definition.

Definition 5.1 (Diagonal Circuits)**.** *A **diagonal depth-4 circuit** has the form $\Phi = \sum_{i=1}^k \Psi_i$, where each product gate Ψ_i is of the form $\Psi_i = \vec{P}_i^{\vec{e}_i}$, and each $P_{i,j}(\vec{x})$ is a sum of univariate polynomials, $P_{i,j} = \sum_{m=1}^n g_{i,j,m}(x_m)$. The syntactic degree of the polynomial computed by the circuit is $\text{s-deg}(\Phi) = \max_i \deg(\Psi_i)$, where $\deg(\Psi_i) = \sum_j e_{i,j}\deg(P_{i,j})$ (i.e. the syntactic degree of Φ is the maximal degree of a multiplication gate Ψ_i in Φ).*

Saxena [Sax08] proved the following theorem, establishing a white-box PIT algorithm for diagonal depth-4 circuits.

Theorem 5.2 (Saxena, [Sax08])**.** *There is a deterministic algorithm that, given as input a diagonal depth-4 circuit Φ, runs in time $\mathsf{poly}(nk \cdot \text{s-deg}(\Phi), \max_{i\in[k]} |\vec{e}_i|_\times)$ and decides whether $\Phi \equiv 0$.*

We give a black-box PIT algorithm for diagonal depth-4 circuits whose running time is quasi-polynomial in the runtime established by Saxena. Saxena gave a white-box reduction from diagonal depth-4 circuits to non-commutative PIT over algebras, and claimed that the Raz-Shpilka [RS05] algorithm can be extended to handle this case. Our result uses a poly-time black-box reduction from diagonal depth-4 circuits to read-once oblivious ABPs. We follow Saxena's duality idea, but "pull" the complexity of the algebras he works with into the ABP itself. We can then apply our hitting sets on read-once oblivious ABPs, which incurs the quasi-polynomial overhead.

Our reduction will be based on the duality idea of Saxena [Sax08] (see his Lemma 2). However, the prior duality statements were cumbersome over fields of positive characteristic. Saxena [Sax08] only alluded briefly to this case, and later Saha, Saptharishi and Saxena [SSS11] gave a description, arguing that duality holds if one can move from $\mathbb{Z}_p$ to $\mathbb{Z}_{p^N}$ for some large N. This is not suitable for black-box reductions, as we cannot change the characteristic in a black-box way. Thus, we develop a unified duality statement that works for all fields, and also works in the black-box setting. This duality improves on that of Saxena [Sax08] by examining *individual degree* as opposed to *total degree*, and using this technical improvement we can extend the result to positive characteristic by using the Frobenius automorphisms.

We will denote the exponential power series as $\exp(x) = \sum_{\ell\geq 0} \frac{x^\ell}{\ell!}$, and the truncated exponential by $\lfloor \exp \rfloor_e(x) = \sum_{\ell=0}^{e} \frac{x^\ell}{\ell!}$. Further, recall our meaning, stated in Section 2, for extracting coefficients of polynomials, and its meaning when we take coefficients "over $\mathbb{F}[\vec{x}][\vec{y}]$" (as opposed to " over $\mathbb{F}[\vec{x},\vec{y}]$").

We first establish duality over $\mathbb{Q}$, treating the coefficients $\vec{c}$ of our polynomials as formal variables.

Lemma 5.3 (Duality over $\mathbb{Q}$)**.** *Let $\Psi = \vec{P}^{\vec{e}} \in \mathbb{Z}[\vec{x},\vec{c}]$, where $P_j(\vec{x},\vec{c}) = \sum_m g_{j,m}(x_m,\vec{c})$. Then, taking coefficients in $\mathbb{Q}[\vec{x},\vec{c}][\vec{z}]$,*

$$\frac{1}{\vec{e}!}\Psi = \mathfrak{C}_{\vec{z}^{\vec{e}}} \left(\prod_m \left(\prod_j \lfloor \exp \rfloor_{e_j} \left(\sum_j g_{j,m}(x_m,\vec{c}) z_j \right) \right) \right). \tag{5.4}$$

Proof. We will work over the power series $\mathbb{Q}[\![\vec{x},\vec{c},\vec{z}]\!]$. In particular, for $f \in \mathbb{Q}[\vec{x},\vec{c}]$, we have the identity

$$\frac{1}{e!} f(\vec{x},\vec{c})^e = \mathfrak{C}_{z^e} \exp(f(\vec{x},\vec{c})z) \ .$$

Note that this is a formal identity, that is, there is no notion of convergence needed as each coefficient in the power series of $\exp(f(\vec{x},\vec{c})z)$ is the sum of finitely many non-zero terms. It follows then that

$$\frac{1}{\vec{e}!}\Psi = \prod_j \mathfrak{C}_{z_j^{e_j}} \exp(P_j(\vec{x},\vec{c})z_j) = \mathfrak{C}_{\vec{z}^{\vec{e}}} \prod_j \exp(P_j(\vec{x},\vec{c})z_j)$$

using the identity $\exp(f+g) = \exp(f)\exp(g)$, which formally holds in $\mathbb{Q}[\![\vec{x},\vec{c},\vec{z}]\!]$,

$$= \mathfrak{C}_{\vec{z}^{\vec{e}}} \exp\left(\sum_j P_j(\vec{x},\vec{c}) z_j \right) = \mathfrak{C}_{\vec{z}^{\vec{e}}} \exp\left(\sum_j \sum_m g_{j,m}(x_m,\vec{c}) z_j \right)$$

reversing the order of summation,

$$= \mathfrak{C}_{\vec{z}^{\vec{e}}} \exp\left(\sum_m \sum_j g_{j,m}(x_m, \vec{c}) z_j\right)$$
$$= \mathfrak{C}_{\vec{z}^{\vec{e}}} \prod_m \prod_j \exp\left(g_{j,m}(x_m, \vec{c}) z_j\right)$$

and now discarding terms of degree $> e_j$ in z_j, as that do not affect the coefficient of $\vec{z}^{\vec{e}}$,

$$= \mathfrak{C}_{\vec{z}^{\vec{e}}} \prod_m \prod_j \lfloor \exp \rfloor_{e_j} \left(g_{j,m}(x_m, \vec{c}) z_j\right) \qquad \square$$

Even though the above proof used the exponential function, which has no exact analogue in positive characteristic, the final statement is just one of polynomials, and thus transfers to any field of sufficiently large characteristic. In particular, the characteristic must be $> |\vec{e}|_\infty$, the maximum exponent. We now transfer this duality to any field. We will abuse notation and treat the characteristic zero case as working over characteristic p, where $p = \infty$. Note that the "base-∞" expansion of any integer is just that integer itself, followed by zeroes.

Lemma 5.5 (Duality over any $\mathbb{F}$)**.** *Let $\mathbb{F}$ be a field of characteristic p (for characteristic zero, we abuse notation and use $p = \infty$). Let $\Psi = \vec{P}^{\vec{e}} \in \mathbb{F}[\vec{x}]$, where $P_j(\vec{x}) = \sum_m g_{j,m}(x_m)$.*

Write e_j in its base-p expansion, so that $e_j = \sum_{k \in [\![n_j]\!]} a_{j,k} p^k$ with $a_{j,k} \in [\![p]\!]$. Denote $Q_{j,k}(\vec{x}) := \sum_m g_{j,m}(x_m)^{p^k}$ and $\vec{Q}^{\vec{a}} := \prod_j \prod_k Q_{j,k}^{a_{j,k}}$. Then, $\vec{P}^{\vec{e}} = \vec{Q}^{\vec{a}}$ and

$$\frac{1}{\vec{a}!} \vec{Q}^{\vec{a}} = \mathfrak{C}_{\vec{z}^{\vec{a}}} \left(\prod_m \left(\prod_{j,k} \lfloor \exp \rfloor_{a_{j,k}} \left(\sum_{j,k} g_{j,m}(x_m)^{p^k} z_j\right)\right)\right), \tag{5.6}$$

and $|\vec{a}|_\times \le |\vec{e}|_\times$.

Proof. $\underline{\vec{P}^{\vec{e}} = \vec{Q}^{\vec{a}}}$: This follows from the Frobenius automorphism, that $(x+y)^{p^k} = x^{p^k} + y^{p^k}$ for any $k \ge 0$, in any field of characteristic p. That is,

$$\vec{P}^{\vec{e}} = \prod_j P_j^{e_j} = \prod_j P_j^{\sum_{k \in [\![n_j]\!]} a_{j,k} p^k} = \prod_j \prod_k P_j^{a_{j,k} p^k} = \prod_j \prod_k (P_j^{p^k})^{a_{j,k}}$$
$$= \prod_j \prod_k \left(\left(\sum_m g_{j,m}(x_m)\right)^{p^k}\right)^{a_{j,k}}$$

by the Frobenius automorphism,

$$= \prod_j \prod_k \left(\sum_m g_{j,m}(x_m)^{p^k}\right)^{a_{j,k}} = \prod_j \prod_k Q_{j,k}^{a_{j,k}} = \vec{Q}^{\vec{a}} .$$

$\underline{(5.6)}$: Note that the identity of Lemma 5.3 only involves rational numbers whose denominators $\le |\vec{e}|_\infty$, the maximum exponent. Thus, as $|\vec{a}|_\infty < p$ by construction, Lemma 5.3 is also an identity over $\mathbb{Z}_p[\vec{x}, \vec{c}]$ when applied with the exponent vector $\vec{a}$, and any g polynomials. In particular, we can

substitute values for the constants $\vec{c}$, implying that we can take the g polynomials to be polynomials in $\vec{x}$ with constants from any field extending $\mathbb{Z}_p$, such as $\mathbb{F}$. Thus, the identity can be applied to $\vec{Q}^{\vec{a}}$ over $\mathbb{F}$, yielding the result.

$\underline{|\vec{a}|_\times \leq |\vec{e}|_\times}$: As $|\vec{e}|_\times = \prod_j (1+e_j)$ and $|\vec{a}|_\times = \prod_j \prod_{k \in [\![n_j]\!]} (1+a_{j,k})$, it is sufficient to prove that $\prod_k (1+a_k) \leq 1+e$, for $e = \sum_k a_k p^k$ with $a_k \in [\![p]\!]$. To see this, first define $T := \{e' : 0 \leq e' \leq e\}$ and note that $|T| = 1+e$. Second, let $S = \{e' : e' = \sum_k a'_k p^k, 0 \leq a'_k \leq a_k\}$. Observe that $S \subseteq T$, and $|S| = \prod_k (1+a_k)$, so $\prod_k (1+a_k) = |S| \leq |T| = 1+e$. □

To use this lemma, we first need a structural result about ABPs. In this lemma we broaden the notion of an ABP to have arbitrary labels from a commutative ring $\mathcal{R}[\vec{x}]$, and will specialize this to our setting later.

Lemma 5.7. *Let A be an layered ABP of depth D and width $\leq r$, with edge labels in $\mathcal{R}[\vec{z}]$ for some commutative ring $\mathcal{R}$, computing an n-variate polynomial $f \in \mathcal{R}[\vec{z}]$. Then for any $\vec{e} \in \mathbb{N}^n$, there is an ABP A', with edge labels in $\mathcal{R}$, of depth D and width $\leq r \cdot |\vec{e}|_\times$ computing $\mathfrak{C}_{\vec{z}^{\vec{e}}}(f)$.*

Further, the edge labels between layers $i-1$ and i in A' occur as coefficients of the edge labels between layers $i-1$ and i in A.

Proof. Define $S := \{\vec{a} : \vec{0} \leq \vec{a} \leq \vec{e}, \vec{a} \in \mathbb{N}^n\}$, where we impose the natural partial order on $\mathbb{N}^n$. Clearly $|S| = |\vec{e}|_\times$. Index the nodes in A by $[r] \times [\![D+1]\!]$, so that the i-th layer in A consists of the nodes $[r] \times \{i\}$. Define the nodes of A' to be $[r] \times S \times [\![D+1]\!]$, so that the i-th layer of A' consists of the nodes $[r] \times S \times \{i\}$.

For each edge $(v \times (i-1), v' \times i)$ in A with label $g \in \mathcal{R}[\vec{z}]$, and each $\vec{a}, \vec{a}' \in S$ with $\vec{a} \leq \vec{a}'$, define the edge label $(v \times \vec{a} \times (i-1), v' \times \vec{a}' \times i)$ to be $\mathfrak{C}_{\vec{z}^{\vec{a}'-\vec{a}}}(g)$. Letting the source of A be denoted $s \times 0$, and the sink denotes $t \times D$, we define the source of A' to be $s \times \vec{0} \times 0$ and the sink to be $t \times \vec{e} \times D$. After removing nodes that belong to no source-sink path, it is clear that A' is a layered ABP, with depth D, width $\leq r \cdot |\vec{e}|_\times$. Further, the edge labels between layers $i-1$ and i in A' are coefficients of the edge labels between layers $i-1$ and i in A as desired. It remains to show that A' computes as desired, which follows from an induction on layers. Specifically, one can see that the paths from the source of A' to $v \times \vec{a} \times i$ compute $\mathfrak{C}_{\vec{z}^{\vec{a}}}(f_{v \times i})$, where $f_{v \times i}$ is the polynomial computed by the paths from the source in A to $v \times i$. □

We now apply this structural result, along with Saxena's dual form, to show that depth-4 diagonal circuits can be computed by small read-once oblivious ABPs.

Lemma 5.8. *Let $\mathbb{F}$ be any field. Let $f(x_1, \ldots, x_n)$ be computed by a diagonal circuit $\Phi = \sum_{i=1}^k \Psi_i$, where $\Psi_i = \vec{P}_i^{\vec{e}_i}$. Denote the syntactic degree of f to be $\text{s-deg}(f) = \max_i \text{s-deg}(\Psi_i)$. Then there is an read-once oblivious ABP A computing f with variable order $x_1 < \cdots < x_n$, with depth n and width $\leq k \cdot \max_{i \in [k]} |\vec{e}_i|_\times$ such that the edges in A are labeled with polynomials with degrees $\leq \text{s-deg}(f)$.*

Proof. Let p denote the characteristic of $\mathbb{F}$, with the convention that $p = \infty$ for characteristic zero fields. By the dual form, Lemma 5.5, we have that for each i,

$$\Psi_i = \vec{a}_i! \cdot \mathfrak{C}_{\vec{z}^{\vec{a}_i}}(f_i)$$

where $\vec{a}_i$ is the base-p expansion of $\vec{e}_i$ (and $|\vec{a}_i|_\times \leq |\vec{e}_i|_\times$), $f_i \in \mathbb{F}[\vec{x}][\vec{z}]$, and f_i is a product of n terms, where the j-th term only involves x_j amongst the $\vec{x}$ variables. It follows that f_i can be computed using a width-1, depth-n ABP in $\mathbb{F}[\vec{x}][\vec{z}]$, where the labels from layer $j-1$ to j only involve x_j amongst all $\vec{x}$ variables.

Applying Lemma 5.7 to f_i, taking $\mathcal{R} = \mathbb{F}[\vec{x}]$, we see (after a scalar multiplication by $\vec{a}_i$!) that Ψ_i is computable by an ABP A_i, which depth n, and width $\leq |\vec{a}_i|_\times \leq |\vec{e}_i|_\times$. Further, Lemma 5.7 establishes that the labels in A_i from layer $j-1$ to layer j come from the labels (by taking coefficients) on the edges from layer $j-1$ to layer j in the ABP computing the product f_i in $\mathbb{F}[\vec{x}][\vec{z}]$. Thus, we see that the labels from layer $j-1$ to layer j in A_i are polynomials in x_j, and have degree at most s-deg(Ψ_i), and thus A_i is an read-once oblivious ABP.

Now, observe that the read-once oblivious ABPs for the Ψ_i can be summed by merging all sources and merging all sinks, and that the resulting ABP A is read-once oblivious, as the variable ordering of $\vec{x}$ is consistent amongst the A_i. It follows that A has the desired properties. □

As the above lemma is constructive, it follows that white-box access to f implies white-box access to the read-once oblivious ABP A, and thus we could run the white-box algorithm of Raz-Shpilka [RS05] to derive a white-box PIT algorithm, in order to rederive Saxena's result (over any field). However, as we also have black-box PIT algorithms for read-once oblivious ABPs (and the above uses the fixed variable order $x_1 < \cdots < x_n$), and the above reduction does not need access to f, we can combine these results to deduce the following black-box PIT result for diagonal depth-4 circuits.

Theorem 5.9 (Black-box PIT for diagonal circuits)**.** *Let $\mathbb{F}$ be a field of arbitrary characteristic. Let $\mathcal{DC}$ be the set of n-variate polynomials computable by diagonal depth-4 circuits, that is, of the form $\Phi = \sum_{i\in[k]} \vec{P}_i^{\vec{e}_i}$, where $|\vec{e}_i|_\times \leq e$ and $P_{i,j}$ is of degree $\leq d$. Then if $|\mathbb{F}| \geq (2ndk^3e^4)^2$ then $\mathcal{DC}$ has a $\mathsf{poly}(n,k,e,d)$-explicit hitting set of size $\leq (2nd^2k^4e^6)^{\lceil \lg n \rceil + 1}$.*

Proof. From Lemma 5.8 we get that any $\Phi \in \mathcal{DC}$ can be computed by an read-once oblivious ABP on variable order $x_1 < \cdots < x_n$, of depth n, width $\leq ke$, with each edge label being a univariate of degree $\leq$ s-deg(Ψ). Noting that s-deg$(\vec{P}_i^{\vec{e}_i}) \leq |\vec{e}_i|_1 \cdot d < |\vec{e}_i|_\times \cdot d$, we invoke Theorem 3.22 to finish the claim. □

Remark 5.10. We remark here that the concurrent work of Agrawal, Saha and Saxena [ASS12] also obtain a quasi-polynomial hitting set for diagonal depth-4 circuits (when they assume each product gate is over a constant number of factors), but only over large characteristic, as they rely on the duality statements of Saxena [Sax08] (and later exposited by Saha, Saptharishi, and Saxena [SSS11]) which only hold over large characteristic. Over small characteristic, [Sax08] and [SSS11] gave more cumbersome duality statements working over prime-power characteristic, and [ASS12] do not extend their work to this case.

Our duality statements work any characteristic, and as such can show that the work of [ASS12] also implies results for diagonal circuits over any characteristic. In particular, instead of using Lemma 5.5 to construct an ABP, one can interpolate the coefficient $\vec{z}^{\vec{e}}$ in Lemma 5.5, and this can be done in depth-3, although the resulting formula is inherently larger than the resulting ABP would be.

5.1 Semi-diagonal Depth-4 circuits

In Saha, Saptharishi, and Saxena [SSS11] the model of semi-diagonal depth-4 circuits was introduced as a small extension of diagonal depth-4 circuits. The modification is that one is allowed to multiply each product gate $\Psi_i = \vec{P}^{\vec{e}}$ by an arbitrary monomial. [SSS11] used that the duality result of Saxena [Sax08] can also be shown to work in this setting. The concurrent work of Agrawal, Saha and Saxena [ASS12] thus present their results for semi-diagonal depth-4 circuits, as opposed to just diagonal depth-4 circuits. For ease of comparison, we also state our result for this model. As

multiplication by a single monomial in Lemma 5.5 preserves the variable disjoint product structure (but increases the degree), it follows that we can still convert semi-diagonal circuits to read-once oblivious ABPs, as stated in Lemma 5.8. Thus, as the details are the same, we conclude the following result.

Theorem 5.11 (Black-box PIT for semi-diagonal circuits)**.** *Let $\mathbb{F}$ be a field of arbitrary characteristic. Let $\mathcal{SDC}$ be the set of n-variate polynomials computable by semi-diagonal depth-4 circuits, that is, of the form $\Phi = \sum_{i\in[k]} m_i(\vec{x})\vec{P}_i^{\vec{e}_i}$, where $m_i(\vec{x})$ is a monomial of degree $\leq d$, $|\vec{e}_i|_\times \leq e$ and $P_{i,j}$ is of degree $\leq d$. Then if $|\mathbb{F}| \geq (4ndk^3e^4)^2$ then $\mathcal{SDC}$ has a $\mathsf{poly}(n,k,e,d)$-explicit hitting set of size $\leq (8nd^2k^4e^6)^{\lceil \lg n\rceil+1}$.*

6 Discussion

This work closes some gap in our understanding of white-box PIT vs. black-box PIT by transferring algorithms that used the partial derivative technique to the black-box world. The recent work of [ASS12] made another significant step by considering set-multilinear formulas (of small depth) where the partition is unknown. It will be very interesting to try and combine these two works to obtain a PIT algorithm for set-multilinear ABPs when the underlying partition is not known.

Another interesting goal is to truly close the gap between white-box and black-box. Specifically, all black-box algorithms for the models studied in this paper (as well as in [ASS12]) run with a quasi-polynomial overhead over the run-times of the corresponding known white-box algorithms. Ideally, this overhead could be made polynomial.

7 Acknowledgments

Much of this work was done when the first author was visiting the second author at the Technion, some while the first author was an intern at Microsoft Research Silicon Valley. We would like to thank Andy Drucker, Omer Reingold, Ilya Volkovich and Sergey Yekhanin for some helpful conversations. We thank Ketan Mulmuley for sharing [Mul12] with us.

References

[Agr05] M. Agrawal. Proving lower bounds via pseudo-random generators. In Proceedings of the 25th FSTTCS, volume 3821 of LNCS, pages 92–105, 2005. 2

[AJS09] V. Arvind, P. S. Joglekar, and S. Srinivasan. Arithmetic Circuits and the Hadamard Product of Polynomials. In FSTTCS, pages 25–36, 2009. 3

[AKS04] M. Agrawal, N. Kayal, and N. Saxena. Primes is in P. Annals of Mathematics, 160(2):781–793, 2004. 2

[AL50] S. A. Amitsur and J. Levitzki. Minimal identities for algebras. Proceedings of the of the American Mathematical Society, 1:449–463, 1950. 7, 20

[AMS10] V. Arvind, P. Mukhopadhyay, and S. Srinivasan. New results on noncommutative and commutative polynomial identity testing. Computational Complexity, 19(4):521–558, 2010. 7, 8

[AS10] Vikraman Arvind and Srikanth Srinivasan. On the hardness of the noncommutative determinant. In Proceedings of the 42nd Annual STOC, pages 677–686, 2010. 3

[ASS12] M. Agrawal, C. Saha, and N. Saxena. Quasi-polynomial hitting-set for set-depth-Δ formulas. Electronic Colloquium on Computational Complexity (ECCC), 19(113), 2012. 1, 5, 8, 27, 28

[AvMV11] M. Anderson, D. van Melkebeek, and I. Volkovich. Derandomizing polynomial identity testing for multilinear constant-read formulae. In Proceedings of the 26th Annual CCC, pages 273–282, 2011. 2

[BOC92] M. Ben-Or and R. Cleve. Computing algebraic formulas using a constant number of registers. SIAM J. Comput., 21(1):54–58, 1992. 3

[BOT88] M. Ben-Or and P. Tiwari. A deterministic algorithm for sparse multivariate polynominal interpolation. In Proceedings of the 20th Annual STOC, pages 301–309, 1988. 2, 7

[BW05] A. Bogdanov and H. Wee. More on noncommutative polynomial identity testing. In Proceedings of the 20th Annual IEEE Conference on Computational Complexity, pages 92–99, 2005. 7, 20, 21

[CHSS11] S. Chien, P. Harsha, A. Sinclair, and S. Srinivasan. Almost settling the hardness of noncommutative determinant. In Proceedings of the 43rd annual STOC, pages 499–508, 2011. 3

[CS07] S. Chien and A. Sinclair. Algebras with polynomial identities and computing the determinant. SIAM J. on Computing, 37:252–266, 2007. 3

[DSY09] Z. Dvir, A. Shpilka, and A. Yehudayoff. Hardness-randomness tradeoffs for bounded depth arithmetic circuits. SIAM J. on Computing, 39(4):1279–1293, 2009. 2

[FS12] M. A. Forbes and A. Shpilka. On identity testing of tensors, low-rank recovery and compressed sensing. In Proceedings of the 44th annual STOC, pages 163–172, 2012. 2, 3, 8, 11, 12

[GR05] A. Gabizon and R. Raz. Deterministic extractors for affine sources over large fields. In 46th Annual FOCS, pages 407–418, 2005. 11, 12

[HS80] J. Heintz and C. P. Schnorr. Testing polynomials which are easy to compute (extended abstract). In Proceedings of the 12th annual STOC, pages 262–272, 1980. 2

[HWY10a] P. Hrubeš, A. Wigderson, and A. Yehudayoff. Non-commutative circuits and the sum-of-squares problem. In Proceedings of the 42nd Annual ACM Symposium on Theory of Computing (STOC), pages 667–676, 2010. 3

[HWY10b] P. Hrubeš, A. Wigderson, and A. Yehudayoff. Relationless completeness and separations. In Proceedings of the 25th Conference on Computational Complexity, pages 280–290, 2010. 3

[HY12] P. Hrubeš and A. Yehudayoff. Formulas are exponentially stronger than monotone circuits in non-commutative setting. Electronic Colloquium on Computational Complexity (ECCC), 19:61, 2012. 3

[INW94] R. Impagliazzo, N. Nisan, and A. Wigderson. Pseudorandomness for network algorithms. In Proceedings of the 26th annual STOC, pages 356–364, 1994. 1, 9

[JQS09] M. Jansen, Y. Qiao, and J. Sarma. Deterministic identity testing of read-once algebraic branching programs. CoRR, abs/0912.2565, 2009. 9

[JQS10] M. J. Jansen, Y. Qiao, and J. M. N. Sarma. Deterministic black-box identity testing π-ordered algebraic branching programs. In Proceedinds of FSTTCS, pages 296–307, 2010. 9

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

[KS01] A. Klivans and D. Spielman. Randomness efficient identity testing of multivariate polynomials. In Proceedings of the 33rd Annual STOC, pages 216–223, 2001. 2, 7

[KS06] A. Klivans and A. Shpilka. Learning restricted models of arithmetic circuits. Theory of computing, 2(10):185–206, 2006. 1, 6

[Mul12] K. Mulmuley. Geometric complexity theory v: Equivalence between blackbox derandomization of polynomial identity testing and derandomization of noether's normalization lemma. To appear in FOCS 2012, 2012. 1, 3, 6, 7, 28

[Nis91] N. Nisan. Lower bounds for non-commutative computation. In Proceedings of the 23rd Annual STOC, pages 410–418, 1991. 3, 19, 22

[Nis92] N. Nisan. Pseudorandom generators for space-bounded computation. Combinatorica, 12(4):449–461, 1992. 1, 9

[NW96] N. Nisan and A. Wigderson. Lower bound on arithmetic circuits via partial derivatives. Computational Complexity, 6:217–234, 1996. 3, 4, 8

[Raz06] R. Raz. Separation of multilinear circuit and formula size. Theory of Computing, 2(1):121–135, 2006. 3

[Raz09] R. Raz. Multi-linear formulas for permanent and determinant are of super-polynomial size. J. ACM, 56(2), 2009. 3

[RR99] R. Raz and O. Reingold. On recycling the randomness of states in space bounded computation. In Proceedings of the 31st annual STOC, pages 159–168, 1999. 9

[RS05] R. Raz and A. Shpilka. Deterministic polynomial identity testing in non commutative models. Computational Complexity, 14(1):1–19, 2005. 1, 2, 3, 5, 7, 19, 22, 24, 27

[RSY08] R. Raz, A. Shpilka, and A. Yehudayoff. A lower bound for the size of syntactically multilinear arithmetic circuits. SIAM J. on Computing, 38(4):1624–1647, 2008. 3

[RY09] R. Raz and A. Yehudayoff. Lower bounds and separations for constant depth multilinear circuits. Computational Complexity, 18(2):171–207, 2009. 3

[Sax08] N. Saxena. Diagonal circuit identity testing and lower bounds. In ICALP (1), pages 60–71, 2008. 1, 2, 3, 5, 8, 23, 24, 27

[Sch80] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. J. ACM, 27(4):701–717, 1980. 1

[SS11] N. Saxena and C. Seshadhri. Blackbox identity testing for bounded top fanin depth-3 circuits: the field doesn't matter. In Proceedings of the 43rd Annual STOC, pages 431–440, 2011. 2

[SSS09] C. Saha, R. Saptharishi, and N. Saxena. The power of depth 2 circuits over algebras. In FSTTCS, pages 371–382, 2009. 3

[SSS11] C. Saha, R. Saptharishi, and N. Saxena. A case of depth-3 identity testing, sparse factorization and duality. Electronic Colloquium on Computational Complexity (ECCC), 18:21, 2011. 5, 24, 27

[SV09] A. Shpilka and I. Volkovich. Improved polynomial identity testing for read-once formulas. In APPROX-RANDOM, pages 700–713, 2009. 7, 9, 13

[SY10] A. Shpilka and A. Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. Foundations and Trends in Theoretical Computer Science, 5(3-4):207–388, 2010. 2, 15

[Zip79] R. Zippel. Probabilistic algorithms for sparse polynomials. In Symbolic and algebraic computation, pages 216–226. 1979. 1

ECCC ISSN 1433-8092

http://eccc.hpi-web.de