2013 IEEE 54th Annual Symposium on Foundations of Computer Science

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

Michael A. Forbes[†]
Department of EECS
MIT CSAIL
32 Vassar St.
Cambridge, MA 02139
miforbes@mit.edu

Amir Shpilka[‡]
Faculty of Computer Science
Technion — Israel Institute of Technology
Haifa, Israel
shpilka@cs.technion.ac.il

***Abstract*—[1] We study the problem of obtaining efficient, deterministic, *black-box polynomial identity testing algorithms (PIT)* for algebraic branching programs (ABPs) that are read-once and oblivious. This class has an efficient, deterministic, white-box polynomial identity testing algorithm (due to Raz and Shpilka [1]), but prior to this work there was no known such black-box algorithm.**

The main result of this work gives the first quasi-polynomial sized hitting sets for size S circuits from this class, when the order of the variables is known. As our hitting set is of size $\exp(\lg^2 S)$, this is analogous (in the terminology of boolean pseudorandomness) to a seed-length of $\lg^2 S$, which is the seed length of the pseudorandom generators of Nisan [2] and Impagliazzo-Nisan-Wigderson [3] for read-once oblivious boolean branching programs. Thus our work can be seen as an algebraic analogue of these foundational results in boolean pseudorandomness.

Our results are stronger for branching programs of bounded width, where we give a hitting set of size $\exp(\lg^2 S/\lg\lg S)$, corresponding to a seed length of $\lg^2 S/\lg\lg S$. This is in stark contrast to the known results for read-once oblivious boolean branching programs of bounded width, where no pseudorandom generator (or hitting set) with seed length $o(\lg^2 S)$ is known. Thus, while our work is in some sense an algebraic analogue of existing boolean results, the two regimes seem to have non-trivial differences.

In follow up work ([4]), we strengthened a result of Mulmuley [5], and showed that derandomizing a particular case of the Noether Normalization Lemma is reducible to black-box PIT of read-once oblivious ABPs. Using the results of the present work, this gives a derandomization of Noether Normalization in that case, which Mulmuley conjectured would difficult due to its relations to problems in algebraic geometry.

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 [6]). 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 run-time of Saxena's white-box algorithm, matching the concurrent work of Agrawal, Saha, and Saxena [7]. Finally, by combining our results with the reconstruction algorithm of Klivans and Shpilka [8], we obtain deterministic reconstruction algorithms for the above circuit classes.

***Keywords*-branching programs, derandomization, non-commutative polynomials, polynomial identity testing**

I. Introduction

Let C be an algebraic 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 "$f\equiv 0$?". 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 [9], [10] 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 $\leq d/|S|$. Thus, given the circuit C, we can perform these evaluations efficiently, giving an efficient randomized procedure for answering "$f\equiv 0$?". 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.

[†]This work supported by the Center for Science of Information (CSoI), an NSF Science and Technology Center, under grant agreement CCF-0939370.

[‡]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]A full version of this work can be found at http://arxiv.org/abs/1209.2408.

0272-5428/13 $26.00 © 2013 IEEE
DOI 10.1109/FOCS.2013.34

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 [11] (and also by Agrawal [12]) for the black-box model and by Kabanets and Impagliazzo [13] for the white-box model (see also Dvir, Shpilka and Yehudayoff [14]). Another motivation for studying the problem is its relation to algorithmic questions. Indeed, the famous deterministic primality testing algorithm of Agrawal, Kayal and Saxena [15] 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 [16].

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, a black-box PIT algorithm is also known, albeit sometimes with worse parameters. Such examples include depth-2 circuits (also known as sparse polynomials) [17], [18], depth-3 $\Sigma\Pi\Sigma(k)$ circuits [19], Read-k formulas [20] and depth-3 tensors (also known as depth-3 set-multilinear circuits) [1], [21]. 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 algebraic formulas, or, more generally, the models of non-commutative algebraic branching programs (ABPs) and set-multilinear algebraic branching programs [1] (see Subsection I-A 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. Equivalently, we give a *hitting set* of size $2^{\mathcal{O}(\lg^2 S)}$ for size S circuits from this model. By tuning parameters in our recursion, we obtain hitting sets of size $2^{\mathcal{O}(\lg^2 S/\lg\lg S)}$ when the branching programs are also of bounded width. Using our main 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 [6]. Although exponential lower bounds are known for these models, we note that the algebraic hardness-versus-randomness result of Kabanets and Impagliazzo [13] (as well as the extension of this result by Dvir, Shpilka and Yehudayoff [14] 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.

The algebraic circuit models considered in this work, though restricted, have received significant attention in existing work on lower bounds and pseudorandomness, and are strong enough to capture non-trivial derandomization questions arising elsewhere. In Subsection I-A we define these models, and state our results in Subsection I-B. In Subsection I-C we discuss various work concerning these models, and explain relations to other areas such as (boolean) space-bounded derandomization, the Noether Normalization Lemma from algebraic geometry, and an algebraic analogue of the natural proofs barrier in boolean lower bounds. In Subsection I-D, we outline the main proof ideas of the hitting set for read-once ABPs.

A. 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 [22] who proved exponential lower bounds on the size of non-commutative ABPs computing the determinant or permanent polynomials.

Definition I.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 [23]. 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 I.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 I.3 (Read-Once Oblivious ABP)**.** *A **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 the mapping $x_{i,j} \leftrightarrow x_i^j$ transforms any set-multilinear ABP into a read-once oblivious ABP and vice versa (when we index j starting at zero).

B. 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 I.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 $f \in \mathcal{C}$, $f \equiv 0$ 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 I.5 (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 $\leq \mathsf{poly}(D, n, r)^{\mathcal{O}(\lg D)}$.*

This theorem plays a crucial role in future work by the authors ([4]), were we give a derandomization of Noether's Normalization Lemma in a certain case. In Subsection I-D we explain our proof technique and give an overview of the proof. Our technique also yields an improved derandomization when the branching program has small width, which we now state.

Theorem I.6 (PIT for small width read-once oblivious ABPs)**.** *Let $\mathcal{C}$ be the set of D-variate polynomials computable by width $r \leq \mathcal{O}(1)$, depth D, individual degree $< n$ read-once oblivious ABPs. If $|\mathbb{F}| \geq \mathsf{poly}(D, n)$, then $\mathcal{C}$ has a $\mathsf{poly}(D, n)$-explicit hitting set, of size $\leq \mathsf{poly}(D, n)^{\mathcal{O}(\lg D / \lg \lg D)}$.*

Using Theorem I.5 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 I.5. 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 I.7 (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. Raz and Shpilka [1] 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. 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 I.8 (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 [6] defined the model of diagonal circuits (see the full paper for a definition) in an attempt to capture

some of the complexity of depth-4 circuits. Relying on the algorithm of [1], Saxena obtained a polynomial time white-box PIT algorithm for this model (for a certain setting of the parameters). Saha, Saptharishi and Saxena [24], 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 [7] 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 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 I.9 (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 $\leq \mathsf{poly}(n,d,k,e)^{\mathcal{O}(\lg n)}$.*

Ramprasad Saptharishi [25] showed that a new notion called evaluation dimension extends the partial derivative technique as used in [6] and that our hitting set holds for this model as well, thus obtaining an alternate proof of Theorem I.9. We discuss this in the full paper.

Klivans and Shpilka [8] 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.

[2]The terminology used in [8] is that of learning by multiplicity automata, but these are completely equivalent to read-once oblivious ABPs, as shown in [8].

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 [8]. 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 I.10 (Extension and Derandomization of [8])**.** *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.

C. Related work

Boolean Pseudorandomness: 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 [2] 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 [3] 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, corresponding to a seed length

[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 I.9, even though the actual circuit could be much smaller.

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 [26] argues that the [3] 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.

However, despite the similarities to the boolean regime, our results improve when the branching programs have bounded width. Specifically, our hitting sets (Theorem I.6) achieve a seed length of $\mathcal{O}(\lg^2 S/\lg\lg S)$, and it has been a long-standing open problem (see [27, Open Problem 8.6]) to achieve a seed length (for pseudorandom generators, or even hitting sets) of $o(\lg^2 S)$ for boolean read-once oblivious branching programs of constant width. Despite much recent work (see [28]–[35]), such seed-lengths are only known for branching programs that are restricted even further, such as regular or permutation branching programs. It is an interesting question as to whether any insights of this paper can achieve a similar seed length in the boolean regime, or in general whether there are any formal connections between algebraic and boolean pseudorandomness for read-once oblivious branching programs.

Derandomizing Noether Normalization: Mulmuley [5] (see the full version [36]) recently showed that the Noether Normalization Lemma (of commutative algebra and algebraic geometry) can in a sense be made constructive, assuming that PIT can be derandomized. This lemma shows, roughly, that in any commutative ring R, there is a smaller subring $S \subseteq R$ that captures many of the interesting properties of R. The usual proof of this lemma is to take the subring S to be generated by a sufficiently large set of "random" elements from R. Thus, one can hope to get a constructive version of this lemma by invoking the appropriate derandomization hypothesis from complexity theory, and Mulmuley shows that derandomization of PIT suffices.

A particular focus of Mulmuley's work is when R is the ring of polynomials, whose variables are the $n^2 r$ variables in r symbolic $n \times n$ matrices, such that these polynomials are invariant under the simultaneous conjugation of the r matrices by any scalar matrix. This ring of invariants has an explicit set T of generators, of size $\exp(r, n)$. Noether Normalization shows that there exists a set T' of size $\mathsf{poly}(r, n)$, such that T' generates a subring $S \subseteq R$, and R is *integral*[4] over S. Mulmuley shows that derandomizing black-box PIT would yield an explicit such set T', and because of the relations with algebraic geometry, conjectures that finding such a set T' is hard. Mulmuley also derives weaker results, for other rings, just from the assumption that black-box PIT can be derandomized for diagonal circuits.

In this work, we give a quasi-polynomial hitting sets for diagonal circuits (Theorem I.9), making some of Mulmuley's weaker results unconditional. More interestingly, in follow-up work ([4]), we improved Mulmuley's reduction from Noether Normalization to PIT in the case of the above ring R of invariants, and showed that derandomizing PIT for read-once oblivious ABPs is sufficient for finding any explicit set T' of invariants generating the desired subring S. By using the results of this paper (Theorem I.5), one can construct an explicit set T' of size $\mathsf{poly}(n, r)^{\log r}$, despite the conjectured hardness of this problem.

Algebraically Natural Proofs: Razborov and Rudich [37] defined the notion of a *natural proof*, and showed that no natural proof can yield strong lower bounds for many interesting boolean models of computation, assuming widely believed conjectures in cryptography about the existence of pseudorandom functions. Further, they showed that this barrier explains the lack of progress in obtaining such lower bounds, as essentially all of the lower bound techniques known are natural in their sense.

In algebraic complexity, there is also a notion of an algebraically natural proof, and essentially all known lower bounds are natural in this sense (see [38] and [16, §3.9]). However, there is no formal evidence to date that algebraically natural proofs cannot prove strong lower bounds, as there are no candidate constructions for (algebraic) pseudorandom functions. The main known boolean construction, by Goldreich, Goldwasser and Micali [39], does not work in the algebraic regime as the construction uses repeated function composition, which results in a polynomial of exponential degree and as such does not fit in the framework of algebraic complexity theory, which only studies polynomials of polynomial degree.

In this work, we give limited informal evidence that some variant of the GGM construction could yield an algebraically natural proofs barrier. That is, Naor (see [40]) observed that the GGM construction is superficially similar to Nisan's [2] pseudorandom generator, in that they both use recursive trees of function composition. In this work, we give an algebraic analogue of Nisan's [2] boolean pseudorandom generator, and as such use repeated function composition. As in the naive algebraization of the GGM construction, a naive implementation of our proof strategy would incur a degree blow-up. While the blow-up would only be quasi-polynomial, it would ultimately result in a hitting set of size $\exp(\lg^3 S)$ for read-once oblivious ABPs of size S, instead of the $\exp(\lg^2 S)$ that we achieve. To obtain this better result, we introduce an interpolation trick that allows us to control the degree blow-up, even though we use repeated function composition. While this trick alone does not yield the desired algebraic analogue of the GGM construction, it potentially removes the primary obstacle to constructing an

[4]A ring R is integral over a subring S, if for every $r \in R$, there is some monic polynomial $p(x) \in S[x]$ such that $p(r) = 0$. For example, the algebraic integers are integral over $\mathbb{Z}$, but $\mathbb{Q}$ is not integral over $\mathbb{Z}$.

algebraic analogue of GGM.

The Partial Derivative Method: Besides the natural goal of obtaining the most general possible PIT result, the problem of obtaining a black-box version of the algorithm of Raz and Shpilka [1] is interesting because of the technique used there. Roughly, the PIT algorithm of [1] 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 algebraic circuits (defined in Nisan and Wigderson [23]) are all examples of algebraic models that compute polynomials with that property, and so the algorithm of [1] 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 algebraic circuits (see e.g., [22], [23], [41]–[44]) 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 [21], 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 below), so this work closes a gap in our understanding of such low-dimensional models.

Non-commutative ABPs: While the primary focus of algebraic complexity are polynomials over commutative rings, it has proved challenging to prove lower bounds in sufficiently powerful circuit models because any lower bound would have to exclude any possible "massive cancellation". Thus, in recent years, attention has turned to non-commutative computation, in the hopes that strong lower bounds would be easier to obtain (see [22], [45]–[51]).

Similarly, the PIT problem has also been studied in the non-commutative model. While the work of Raz and Shpilka [1] 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 [52] showed how, leveraging the ideas in the Amitsur-Levitzki theorem [53], 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 $\approx s^{\lg^2 s}$ for non-commutative ABPs of size s. Our Theorem I.8 gives a hitting set of size $\approx s^{\lg s}$ 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, as we instead give an improved reduction from non-commutative to commutative computation, such that for ABPs the resulting computation is set-multilinear. 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 [54], 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., [17], [18]) 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 small ABP.

We note that in the aforementioned [54], 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 [54] 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.

Previous Work: In a previous paper [21] 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 I.7, but the previous work has a better dependence on some of the parameters. More importantly, it is interesting to note that the proof in [21] 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 [21]. First the case of degree 2 is solved. In this case the tensor is simply a bilinear map. For larger D, the algorithm 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 [21]. On the other hand, if we restrict ourselves to set-multilinear depth-3 circuits then the hitting set of [21] is polynomially smaller than the set that we construct here.

Independently and simultaneously with this work, Agrawal, Saha and Saxena [7] 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 $\exp((2h^2 \lg(s))^{h+1})$, 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 [7] also give a quasipolynomial-sized hitting set for semi-diagonal circuits over large characteristic fields. Our results, in particular our extension of Saxena's [6] duality to all fields, can help extend the results of [7] to small characteristics as well.

We also mention that in [55], [56] 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) $n^{\mathcal{O}(k \lg(k) \lg(n))}$ 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 I.5 gives a hitting set of size $n^{\mathcal{O}(\lg(n))}$ 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 [55], which follows the proof technique of [57]. The later paper [56] use an algebraic analogue of the Impagliazzo-Nisan-Wigderson [3] generator, as we do, but the details are different.

D. Proof overview

The first step in our proof is to work with a normal form for read-once oblivious ABPs. Specifically, an easy lemma shows that any polynomial f computed in this model can be written as[6] $f(\vec{x}) = (\prod_{i \in [\![D]\!]} M_i(x_i))_{(0,0)}$, where each M_i is a matrix whose entries are univariate polynomials in the variable x_i. Next, we observe that for purposes of a stronger inductive hypothesis, we work with matrix-valued polynomials computed as $f(\vec{x}) = \prod_{i \in [\![D]\!]} M_i(x_i)$. With this view in mind, we can now consider the recursion scheme.

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 a 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 "$M(x)N(x^n) \approx M(f(z))N(g(z))$".

[5]Their definition of set-multilinear formulas is actually that of a pure formula (see [23]). 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.

[6]We use $[\![n]\!]$ to denote $\{0, \ldots, n-1\}$.

In the Kronecker substitution, this equivalence typically means that there is a bijection between monomials. We will not use that notion, and instead will use 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 $M(f(z))N(g(z))$. Namely, the space spanned by $M(x)N(y)$, where we range over all assignments to (x, y) is the same as the spaces spanned by $M(f(z))N(g(z))$, where we range over all assignments to 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 an r^2-dimensional space, so we seek to map these n^2 vectors to a smaller space while preserving rank. Gabizon and Raz [58] established such a lemma, and our prior work [21] 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.

While the strategy above will succeed, it will be deficient as the resulting hitting set will be quasi-polynomially larger than desired, and each point in the hitting set will be quasi-polynomially explicit, instead of being polynomially explicit. This deficiency arises from repeated function composition, as we now explain by example. Consider a depth 4, read-once oblivious ABP, written as $M(x, y, z, w) := A(x)B(y)C(z)D(w)$, where A, B, C, D are $r \times r$ matrices with entries that are univariate polynomials degree $< n$. After the first step we have $A(x)B(y)C(z)D(w) \approx A(f(x'))B(g(x'))C(f(y'))D(g(y'))$ for two new variables x' and y'. Viewing this matrix product as a read once oblivious ABP in the two variables x' and y', we can repeat the variable merging procedure, to obtain

$$\begin{aligned} A(x)B(y)C(z)D(w) &\approx A(f(x'))B(g(x'))C(f(y'))D(g(y')) \\ &\approx A((f \circ f)(x''))B((g \circ f)(x'')) \\ &\quad C((f \circ g)(x''))D((g \circ g)(x'')), \end{aligned}$$

for a new variable x''. While this reduction is desirable, as we can now fully interpolate the single final variable x'' to obtain a hitting set, the degree in the final x'' is $r^2 \cdot r^2 = r^4$, instead of remaining r^2. That is, to reduce from D variables to a single variable, we will use at least $\lg D$ compositions of the f, g polynomials, yielding a degree of at least $r^{\Omega(\lg D)}$. As there are $\lg D$ levels of recursion, and each requires a seed matching the degree of the ABPs, this will imply that the resulting hitting set is of size $r^{\Omega(\lg^2 D)}$, which is larger than what we achieve in this work. Further, this hitting set is not even polynomially explicit, as computing it requires evaluating polynomials of quasipolynomial degree, which potentially requires quasipolynomial bit-length when computing over the rationals.

Such degree blow-up seems inherent in this naive composition method, so the "extract and recurse" paradigm by itself is insufficient for an algebraic analogue of boolean results in space-bounded derandomization. To overcome this, we avoid treating $M'(x', y') := A(f(x'))B(g(x'))C(f(y'))D(g(y'))$ as a entirely generic ABP, and recognize that function composition has occurred, and that we know precisely what those functions are. That is, the extractor will give a seed β, and a $\mathsf{poly}(r)$ set of points $P'_\beta \subseteq \mathbb{F}^2$ such that the span of $M'(x', y')$ is equal to the span of $M'|_{P'_\beta}$, for most values of β. However, realizing that x', y' define values of x, y, z, w via the known curves f, g, we can use the points P'_β to construct a new set of points $P_\beta \subseteq \mathbb{F}^4$ such that the span of $M(x, y, z, w)$ is equal to the span of $M|_{P_\beta}$ for most values of β. Finally, we can then interpolate curves in a new variable x'' to pass through the points P_β, and these curves will be of degree $|P_\beta|$. Noting that $|P_\beta| = |P'_\beta| = \mathsf{poly}(r)$, and that this $\mathsf{poly}(r)$ is the number of samples of the extractor, which only depends on the width of the ABPs considered, and this width never changes, we see that there is no degree blow-up throughout the recursion.

To achieve better results when the width r of the ABP is small, we keep the paradigm from above, but change the branching factor of the recursion. That is, in the above recursion we merged pairs of variables at each level, so that if we start with D variables we need $\lg D$ levels of recursion, and the curves involved will have $\mathsf{poly}(r)$ degree. By changing to a branching factor of B, we use $\log_B D$ levels of recursion, and the curves involved will have $\approx \mathsf{poly}(r)^B$ degree. As the number of levels in the recursion equals the number of seeds we use, and the degree of the curves (along with the degree n of the ABP) governs the number of values to try for each seed, we can see that choosing $B = \log_r D$ yields $\lg D / \lg \lg D \cdot \lg r$ seeds, and each seed will take $\mathsf{poly}(n, D)$ values. For $r = \mathcal{O}(1)$, this yields a hitting set of size $\mathsf{poly}(n, D)^{O(\lg D / \lg \lg D)}$.

In the boolean regime, changing the branching factor of recursion in the pseudorandom generators of Nisan [2] or Impagliazzo-Nisan-Wigderson [3] is not known to achieve such savings. This seems to be because each application of the extractor (viewed as an averaging sampler) has a sample complexity that depends on the total number of variables in

the branching program, which is needed so the error does not become too large. It follows that increasing the branching factor simply multiples the seed length by $B/\lg B$, which only becomes worse as B increases. In contrast, the rank extractor used here has a sample complexity that depends only on the width of the ABP, and the number of total variables only affects the extractor in the number of seeds needed. Thus, the seed-length grows as $(B \lg r + \lg n + \lg D) \lg D / \lg B$, which allows us to balance parameters by taking $B \lg r \approx \lg D$.

II. 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, Ramprasad Saptharishi, Ilya Volkovich and Sergey Yekhanin for some helpful conversations. We thank Ketan Mulmuley for sharing [5] with us. We thank Ramprasad for allowing us to include his result on evaluation dimension here. We also thank Avi Wigderson for raising the question of whether our technique could yield better results for the bounded width case.

References

[1] R. Raz and A. Shpilka, "Deterministic polynomial identity testing in non-commutative models," Computational Complexity, vol. 14, no. 1, pp. 1–19, 2005.

[2] N. Nisan, "Pseudorandom generators for space-bounded computation," Combinatorica, vol. 12, no. 4, pp. 449–461, 1992.

[3] R. Impagliazzo, N. Nisan, and A. Wigderson, "Pseudorandomness for network algorithms," in Proceedings of the 26th annual STOC, 1994, pp. 356–364.

[4] M. A. Forbes and A. Shpilka, "Explicit noether normalization for simultaneous conjugation via polynomial identity testing," Electronic Colloquium on Computational Complexity (ECCC), vol. 20, p. 33, 2013.

[5] K. Mulmuley, "Geometric complexity theory V: Equivalence between blackbox derandomization of polynomial identity testing and derandomization of Noether's normalization lemma," in FOCS, 2012, pp. 629–638.

[6] N. Saxena, "Diagonal circuit identity testing and lower bounds," in ICALP (1), 2008, pp. 60–71.

[7] M. Agrawal, C. Saha, and N. Saxena, "Quasi-polynomial hitting-set for set-depth-Δ formulas," Electronic Colloquium on Computational Complexity (ECCC), vol. 19, no. 113, 2012.

[8] A. Klivans and A. Shpilka, "Learning restricted models of arithmetic circuits." Theory of computing, vol. 2, no. 10, pp. 185–206, 2006.

[9] R. Zippel, "Probabilistic algorithms for sparse polynomials," in EUROSAM, 1979, pp. 216–226.

[10] J. T. Schwartz, "Fast probabilistic algorithms for verification of polynomial identities," J. ACM, vol. 27, no. 4, pp. 701–717, 1980.

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

[12] M. Agrawal, "Proving lower bounds via pseudo-random generators," in Proceedings of the 25th FSTTCS, ser. LNCS, vol. 3821, 2005, pp. 92–105.

[13] V. Kabanets and R. Impagliazzo, "Derandomizing polynomial identity tests means proving circuit lower bounds," Computational Complexity, vol. 13, no. 1-2, pp. 1–46, 2004.

[14] Z. Dvir, A. Shpilka, and A. Yehudayoff, "Hardness-randomness tradeoffs for bounded depth arithmetic circuits," SIAM J. on Computing, vol. 39, no. 4, pp. 1279–1293, 2009.

[15] M. Agrawal, N. Kayal, and N. Saxena, "Primes is in P," Annals of Mathematics, vol. 160, no. 2, pp. 781–793, 2004.

[16] A. Shpilka and A. Yehudayoff, "Arithmetic circuits: A survey of recent results and open questions," Foundations and Trends in Theoretical Computer Science, vol. 5, no. 3-4, pp. 207–388, 2010.

[17] M. Ben-Or and P. Tiwari, "A deterministic algorithm for sparse multivariate polynominal interpolation," in Proceedings of the 20th Annual STOC, 1988, pp. 301–309.

[18] A. Klivans and D. Spielman, "Randomness efficient identity testing of multivariate polynomials," in Proceedings of the 33rd Annual STOC, 2001, pp. 216–223.

[19] 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, 2011, pp. 431–440.

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

[21] M. A. Forbes and A. Shpilka, "On identity testing of tensors, low-rank recovery and compressed sensing," in Proceedings of the 44th annual STOC, 2012, pp. 163–172.

[22] N. Nisan, "Lower bounds for non-commutative computation," in Proceedings of the 23rd Annual STOC, 1991, pp. 410–418.

[23] N. Nisan and A. Wigderson, "Lower bound on arithmetic circuits via partial derivatives," Computational Complexity, vol. 6, pp. 217–234, 1996.

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

[25] R. Saptharishi, 2012, private communication.

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

[27] S. P. Vadhan, "Pseudorandomness," Foundations and Trends in Theoretical Computer Science, vol. 7, no. 1-3, pp. 1–336, 2012.

[28] A. Bogdanov, Z. Dvir, E. Verbin, and A. Yehudayoff, "Pseudorandomness for width-2 branching programs," Theory of Computing, vol. 9, pp. 283–293, 2013.

[29] J. Síma and S. Zák, "Almost k-wise independent sets establish hitting sets for width-3 1-branching programs," in CSR, 2011, pp. 120–133.

[30] P. Gopalan, R. Meka, O. Reingold, L. Trevisan, and S. P. Vadhan, "Better pseudorandom generators from milder pseudorandom restrictions," in FOCS, 2012, pp. 120–129.

[31] M. Braverman, A. Rao, R. Raz, and A. Yehudayoff, "Pseudorandom generators for regular branching programs," in Proceedings of the 51st annual FOCS, 2010, pp. 40–47.

[32] J. Brody and E. Verbin, "The coin problem and pseudorandomness for branching programs," in Proceedings of the 51st annual FOCS, 2010, pp. 30–39.

[33] M. Koucký, P. Nimbhorkar, and P. Pudlák, "Pseudorandom generators for group products: extended abstract," in STOC, 2011, pp. 263–272.

[34] A. De, "Pseudorandomness for permutation and regular branching programs," in IEEE Conference on Computational Complexity, 2011, pp. 221–231.

[35] T. Steinke, "Pseudorandomness for permutation branching programs without the group theory," Electronic Colloquium on Computational Complexity (ECCC), vol. 19, p. 83, 2012.

[36] K. Mulmuley, "Geometric complexity theory V: Equivalence between blackbox derandomization of polynomial identity testing and derandomization of Noether's normalization lemma," arXiv, vol. abs/1209.5993, 2012, full version of the FOCS 2012 paper.

[37] A. A. Razboeov and S. Rudich, "Natural proofs," J. of Computer and System Sciences, vol. 55, no. 1, pp. 24–35, 1997.

[38] S. Aaronson, "Arithmetic natural proofs theory is sought," 2008, http://scottaaronson.com/blog/?p=336. [Online]. Available: http://scottaaronson.com/blog/?p=336

[39] O. Goldreich, S. Goldwasser, and S. Micali, "How to construct random functions," J. ACM, vol. 33, no. 4, pp. 792–807, 1986.

[40] O. Reingold. (2013) Research-life stories — Omer Reingold. http://windowsontheory.org/2013/02/13/research-life-stories-omer-reingold/. [Online]. Available: http://windowsontheory.org/2013/02/13/research-life-stories-omer-reingold/

[41] R. Raz, "Separation of multilinear circuit and formula size," Theory of Computing, vol. 2, no. 1, pp. 121–135, 2006.

[42] ——, "Multi-linear formulas for permanent and determinant are of super-polynomial size," J. ACM, vol. 56, no. 2, 2009.

[43] R. Raz, A. Shpilka, and A. Yehudayoff, "A lower bound for the size of syntactically multilinear arithmetic circuits," SIAM J. on Computing, vol. 38, no. 4, pp. 1624–1647, 2008.

[44] R. Raz and A. Yehudayoff, "Lower bounds and separations for constant depth multilinear circuits," Computational Complexity, vol. 18, no. 2, pp. 171–207, 2009.

[45] S. Chien and A. Sinclair, "Algebras with polynomial identities and computing the determinant," SIAM J. on Computing, vol. 37, pp. 252–266, 2007.

[46] V. Arvind, P. S. Joglekar, and S. Srinivasan, "Arithmetic Circuits and the Hadamard Product of Polynomials," in FSTTCS, 2009, pp. 25–36.

[47] V. Arvind and S. Srinivasan, "On the hardness of the noncommutative determinant," in Proceedings of the 42nd Annual STOC, 2010, pp. 677–686.

[48] P. Hrubeš, A. Wigderson, and A. Yehudayoff, "Noncommutative circuits and the sum-of-squares problem," in Proceedings of the 42nd Annual ACM Symposium on Theory of Computing (STOC), 2010, pp. 667–676.

[49] S. Chien, P. Harsha, A. Sinclair, and S. Srinivasan, "Almost settling the hardness of noncommutative determinant," in Proceedings of the 43rd annual STOC, 2011, pp. 499–508.

[50] P. Hrubeš, A. Wigderson, and A. Yehudayoff, "Relationless completeness and separations," in Proceedings of the 25th Conference on Computational Complexity, 2010, pp. 280–290.

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

[52] A. Bogdanov and H. Wee, "More on noncommutative polynomial identity testing," in Proceedings of the 20th Annual IEEE Conference on Computational Complexity, 2005, pp. 92–99.

[53] S. A. Amitsur and J. Levitzki, "Minimal identities for algebras," Proceedings of the of the American Mathematical Society, vol. 1, pp. 449–463, 1950.

[54] V. Arvind, P. Mukhopadhyay, and S. Srinivasan, "New results on noncommutative and commutative polynomial identity testing," Computational Complexity, vol. 19, no. 4, pp. 521–558, 2010.

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

[56] M. J. Jansen, Y. Qiao, and J. M. N. Sarma, "Deterministic black-box identity testing π-ordered algebraic branching programs," in Proceedinds of FSTTCS, 2010, pp. 296–307.

[57] A. Shpilka and I. Volkovich, "Improved polynomial identity testing for read-once formulas," in APPROX-RANDOM, 2009, pp. 700–713.

[58] A. Gabizon and R. Raz, "Deterministic extractors for affine sources over large fields," Combinatorica, vol. 28, no. 4, pp. 415–440, 2008.