Electronic Colloquium on Computational Complexity, Report No. 171 (2016)

Complete Derandomization of Identity Testing and Reconstruction of Read-Once Formulas

Daniel Minahan [*] Ilya Volkovich [†]

Abstract

In this paper we study the identity testing problem of *arithmetic read-once formulas* (ROF) and some related models. A read-once formula is formula (a circuit whose underlying graph is a tree) in which the operations are $\{+, \times\}$ and such that every input variable labels at most one leaf. We obtain the first polynomial-time deterministic identity testing algorithm that operates in the black-box setting for read-once formulas, as well as some other related models. As an application, we obtain the first polynomial-time deterministic reconstruction algorithm for such formulas. Our results are obtained by improving and extending the analysis of the algorithm of [SV15].

1 Introduction

In this paper we study the problem of Polynomial Identity Testing (PIT): given an arithmetic circuit C over a field $\mathbb{F}$, with input variables $x_1, x_2, \dots, x_n$, determine whether C computes the identically zero polynomial. Given its connections to a wide range of problems, PIT is considered a central problem in algebraic complexity theory and algorithms design. Particular instances include: perfect matchings in graphs [Lov79, MVV87], primality testing [AKS04], $\mathsf{IP} = \mathsf{PSPACE}$ [LFKN92, Sha90] the PCP theorem [AS98, ALM$^+$98] and many more. PIT is one of a few natural problems which have a simple efficient randomized algorithm [DL78, Sch80, Zip79] back lack a deterministic one. Indeed, it has been a long standing open question to come up with an efficient deterministic algorithm for this problem.

In this paper we consider the PIT problem in the *black-box* setting. In this setting, one is not given the full description of the circuit C but only allowed black-box (oracle) access to C. The problem of derandomizing identity testing in this setting reduces to that of finding for every s an explicit set of points $\mathcal{H} \subseteq \mathbb{F}^n$ of size $\mathrm{poly}(s)$ such that any non-zero circuit of size s does not vanish on $\mathcal{H}$. We refer to such sets as *hitting sets*. Indeed, the randomized algorithm of [DL78, Sch80, Zip79] provides an exponential-size hitting set. Furthermore, applying standard probabilistic arguments one can show existence of "small" hitting sets. Yet, coming up with an explicit hitting set is believed to be very difficult task as it would immediately imply explicit exponential lowers bounds [HS80, Agr05].

[*] Departments of Mathematics and EECS, CSE Division, University of Michigan, Ann Arbor, MI. Email: dminahan@umich.edu. Research partially supported by NSF.

[†] Department of EECS, CSE Division, University of Michigan, Ann Arbor, MI. Email: ilyavol@umich.edu.

ISSN 1433-8092

Yet, for several restricted classes of arithmetic circuits, efficient deterministic black-box PIT algorithms were found. For example, efficient black-box PIT algorithms were shown for depth-2 arithmetic circuits [BOT88, KS01, LV03] and depth-3 arithmetic circuits with bounded top fan-in (also known as $\Sigma\Pi\Sigma(k)$ circuits) [DS06, KS07, KS09, AM10, SS10, KS11, SS11, ASSS12]. There has also been a lot of progress on PIT for restricted classes of depth-4 circuits [Sax08, AM10, BMS11, ASSS12, KMSV13, Gup14, AvMV15, Muk15, SV15, KS16a, KS16b]. Another body of research has been focused on PIT algorithms for bounded-read models. That is, classes of circuits where each variable appears some bounded number of times [FS12, ASS13, FS13b, FS13a, FSS14, AvMV15, SV15, GKST15, AFS+16, GKS16, FGT16], with the simplest case being the read-once formulas.

A *read-once formula* (ROF for short) is an arithmetic formula (i.e. a tree) in which the operations are $\{+, \times\}$ and such that every input variable labels at most one leaf. These formulas can be thought of as the smallest formulas that depend on all their variables and the simplest non-trivial subclass of multilinear formulas. Although ROFs form a very restricted model of computation they have received a lot of attention in both the Boolean [KLN+93, AHK93, BHH95b] and the algebraic [HH91, BHH95a, BB98, BC98, SV14, SV15, Vol16] worlds.

The first sub-exponential time $n^{\mathcal{O}(\sqrt{n})}$ black-box PIT algorithm for ROFs was given in [SV08]. Later on, in [SV15][1] the result was improved to $n^{\mathcal{O}(\log n)}$ via another algorithm. A different analysis for the latter algorithm, resulting in roughly the same run time, was given in [AvMV15]. Yet, despite the rich body of work devoted to the problem, prior to our work no polynomial-time black-box PIT algorithm was known even for ROFs. In this paper we give the first black-box PIT algorithm for ROFs, and some related classes of formulas, thus achieving a complete derandomization of the PIT problem for these classes. For more information on PIT we refer the reader to the survey [SY10].

It is important to point out that while PIT asks whether the resulting polynomial is identically zero as a formal sum of monomials, some non-identically zero polynomials might evaluate to the zero function. For example, $x^4 - x$ will always evaluate to zero over the field of five elements. For this reason we will allow our algorithm to evaluate the polynomial on elements from a polynomially large extension field of $\mathbb{F}$. In [SV15] it was shown one cannot achieve polynomial-time black-box PIT algorithms if $|\mathbb{F}| = o(n/\log n)$.

1.1 Our Results

In this section we describe and discuss our results. In fact, our results hold for a slightly richer class of preprocessed read-once formulas. A *preprocessed ROF* (PROF for short) is a ROF in which we are allowed to replace each variable x_i with a univariate polynomial $T_i(x_i)$. A polynomial $P(\bar{x})$ is a *Preprocessed Read-Once Polynomial* (PROP for short) if it can be computed by a preprocessed read-once formula (see Section 3.2 for a formal definition). We begin with our main result: polynomial-time black-box PIT algorithm for PROFs.

Theorem 1. *Let $n, d \in \mathbb{N}$. There exist a deterministic algorithm that given black-box (oracle) access to a preprocessed read-once formula Φ on n variables and individual degrees (of the preprocessing) at most d, checks whether $\Phi \equiv 0$. The running time of the algorithm is polynomial in n and d.*

In [SV15] it was shown how to extend a PIT algorithm for a single PROF into a PIT algorithm for a sum of PROFs. By plugging in our main result we obtain a black-box PIT algorithm for sums of PROFs.

[1] Conference version first appeared in [SV09].

Theorem 2. *Let $k, n, d \in \mathbb{N}$. There exist a deterministic algorithm that given black-box (oracle) access to $\Phi = \Phi_1 + \ldots + \Phi_k$, where the Φ_i-s are preprocessed read-once formulas in n variables, with individual degrees at most d, checks whether $\Phi \equiv 0$. The running time of the algorithm is $(nd)^{\mathcal{O}(k)}$.*

Observe that for a fixed $k \in \mathbb{N}$ the algorithm runs in polynomial time with respect to n and d. Furthermore, observe that if $\mathcal{H}$ is hitting set for a sum of two PROPs then $\mathcal{H}$ is an *interpolating* set for a single PROP. That is, the values of a single PROP P on $\mathcal{H}$ contain enough information to uniquely identify P. Indeed, a consequence of Theorem 2 is an interpolating set of polynomial size for PROPs. However in general, $\mathcal{H}$ does not provide us with an efficient algorithm to reconstruct a corresponding PROF.

In [BHH95a], a randomized polynomial-time reconstruction algorithm for ROFs was given. In [SV14], the algorithm was extended to PROFs. Moreover, it was shown how to convert a black-box PIT algorithm into a reconstruction algorithm paying a polynomial overhead. Indeed, by plugging in the result of [SV15], the first deterministic sub-exponential (and, in fact, quasi-polynomial) time reconstruction algorithm for PROFs was given. By plugging in our main result, we achieve a complete derandomization of the reconstruction algorithm by obtaining a deterministic polynomial-time reconstruction algorithm for PROFs.

Theorem 3. *There exist a deterministic algorithm that given black-box (oracle) access to a preprocessed read-one formula Φ, on n variables and individual degrees at most d, reconstructs Φ. Namely, the algorithm outputs a PROF $\hat{\Phi}$ that computes the same polynomial. The running time of the algorithm is polynomial in n and d.*

1.2 Organization

The paper is organized as follows. In Section 2 we give the basic definitions and notations. In Section 3 we formally introduce ROFs and its generalizations along with some structural properties, when in Section 3.3 we discuss the PIT algorithm of [SV15]. In Section 3.4 we prove some additional properties of the algorithm, which is the main technical contribution of the paper. Next, in Section 4 we give our main result, thus proving Theorem 1. We discuss the applications of our main result in Section 5 proving Theorems 2 and 3. We conclude the paper with some open questions in Section 6.

2 Preliminaries

For a positive integer n, let $[n]$ denote the set $\{1, \ldots, n\}$. We now give some definitions that apply to polynomials $P, Q \in \mathbb{F}[x_1, \ldots, x_n]$. For a polynomial P, a variable x_i, and $\alpha \in \mathbb{F}$, let $P|_{x_i=\alpha}$ denote the polynomial that results upon setting $x_i = \alpha$. We say that P *depends* on x_i if there exist $\bar{a}, \bar{b} \in \mathbb{F}^n$ that differ in the i-coordinate such that $P(\bar{a}) \neq P(\bar{b})$. We denote $\mathrm{var}(P) \triangleq \{i \ : \ P \text{ depends on } x_i\}$. Intuitively, P depends on x_i if x_i appears when P is written as a sum of monomials.

We say that P is a *homogeneous* polynomial if every monomial in P has the same total degree. For $i \in \mathbb{N}$ we define $H_i[P]$ as the *homogeneous part* of degree i of P. That is, all the monomials of total degree i that appear in P. If P does not have monomials of degree i then $H_i[P] \equiv 0$. We say that P and Q are *similar* and denote $P \sim Q$ if there exist $\alpha, \beta \in \mathbb{F} \setminus \{0\}$ such that $\alpha \cdot P = \beta \cdot Q$.

In order to actually calculate the complexity of our algorithm we need to define a formal of computational for polynomials.

Definition 2.1 (Arithmetic formula)**.** *A arithmetic formula is a binary tree where each leaf has a variable $x_i \in \{x_1, \ldots, x_n\}$ and each internal node, called a gate, has an operation $+$ or $\times$. Additionally, each leaf and node are labeled with some $(\alpha, \beta) \in \mathbb{F}^2$. The tree is evaluated by recursively calculating the values of the left subtree P_1 and the right subtree P_2 and then combining them by $\alpha(P_1 * P_2) + \beta$ where $*$ is the operation at the top gate.*

The efficiency of an algorithm over a set of formulas $\mathcal{C}$ is therefore based on the number of gates in $\mathcal{C}$. Often times, we will implicitly associate a class of formulas $\mathcal{C}$ with the class of polynomials computed by these formulas.

We consider formula in the *black-box* (or oracle) setting. That is, the algorithm cannot directly look at the formula and is only allowed to query the polynomial on $\mathbb{F}^n$. Hereafter, we assume that an evaluation query can be carried out in $\mathcal{O}(1)$ time. In case $\mathbb{F}$ is small, we allow to query the formula on a polynomially-large extension field of $\mathbb{F}$.

2.1 Generators and Hitting Sets

Our black-box PIT algorithms use the notion of *generators.* In this section, we formally define this notion describe a few of their useful properties and give the connection to hitting sets. Intuitively, a generator $\mathcal{G}$ for a polynomial class $\mathcal{C}$, is a function that stretches t independent variables into $n \gg t$ dependent variables that can be *plugged* into any polynomial $P \in \mathcal{C}$ without causing it to vanish. Recall that a hitting set $\mathcal{H} \subseteq \mathbb{F}^n$ for a class of polynomials $\mathcal{C}$ is a set such that for any nonzero polynomial $P \in \mathcal{C}$, there exists $\bar{a} \in \mathcal{H}$, such that $P(\bar{a}) \neq 0$.

Definition 2.2 (Hitting Set)**.** *Let $\mathcal{C}$ be a class of polynomials in $\mathbb{F}[x_1, \ldots, x_n]$. A set $\mathcal{H}$ is called a* hitting set *for $\mathcal{C}$ provided that $\forall P \in \mathcal{C}$ with $P \not\equiv 0$ we have that $P|_{\mathcal{H}} \not\equiv 0$.*

This leads us to a basic algorithm for PIT.

Algorithm 1. *Let $\mathcal{C}$ be a class of polynomial in $\mathbb{F}[x_1, \ldots, x_n]$ and let $\mathcal{H}$ be a hitting set for $\mathcal{C}$. Then there exists a deterministic PIT algorithm for $\mathcal{C}$ that runs in time $\mathcal{O}(|\mathcal{H}|)$.*

The following generalization of the fundamental theorem of algebra provides hitting sets of exponential size for every polynomial. A proof can be found in [Alo99].

Lemma 2.3. *Let $P \not\equiv 0 \in \mathbb{F}[x_1, \ldots, x_n]$ and suppose the individual degree of any variable in P is bounded by some $d \in \mathbb{N}$. Pick $S \subseteq \mathbb{F}$ with $|S| > d$. Then $P|_S \not\equiv 0$.*

Remark: The precondition of the lemma implies that $|\mathbb{F}| > d$. In case that $\mathbb{F}$ is small, this assumption is met by choosing elements from an appropriately large extension field of $\mathbb{F}$.

A related notion is the notion of generators. Many hitting sets are constructed by means of generators.

Definition 2.4 (Generator)**.** *Let $\mathcal{C}$ be a class of polynomials over $\mathbb{F}$. A polynomial map $G : \mathbb{F}^t \to \mathbb{F}^n$ is a* generator *for $\mathcal{C}$ provided that $\forall P \not\equiv 0 \in \mathcal{C}$ we have $P(G) \not\equiv 0$.*

Intuitively, a generator $\mathcal{G}$ for $\mathcal{C}$ is a polynomial mapping that has a hitting set for $\mathcal{C}$ in its image. More specifically, Lemma 2.3 allows us to convert a generator into a hitting set by observing that a polynomial composed with a polynomial map results in other polynomial. Such composition typically reduces the number of variables that the polynomial depends on, but it may increase the degree the total degree. Thus, since the size of the hitting set produced by Lemma 2.3 depends on both parameters, we want to find a generator that reduces the number of variables without drastically increasing its degree.

3 Read-Once Formulas

In this section we discuss our computational model. We first consider the basic model of read-once formulas and cover some of its main properties. Then, we introduce the model of preprocessed-read-once formulas and give its corresponding properties.

3.1 Read-Once Formulas and Read-Once Polynomials

Most of the definitions that we give in this section are from [HH91] and [SV15] or some small variants. We start by formally defining the notions of a read-once formula and a read-once polynomial.

Definition 3.1 (Read-Once Formula)**.** *A* read-once formula *(ROF) is an arithmetic formula where each variable appears at most once. A polynomial $P(\bar{x})$ is* read-once polynomial *(ROP for short) if it can be computed by a read-once formula.*

Clearly, ROPs form a subclass of multilinear polynomials. In addition, note that the number of gates in a ROF is at most twice the number of variables. This means that our complexity scales with n, so we need only be concerned about how the runtime of our algorithm scales with respect to the number of variables. Thus, our ideal efficiency for an algorithm is $n^{\mathcal{O}(1)}$. The next lemma also follows easily from the definition.

Lemma 3.2 (ROP Structural Lemma)**.** *Every ROP $P(\bar{x})$ that depends on at least two variables can be presented in one of the following forms:*

1. $P(\bar{x}) = P_1(\bar{x}) + P_2(\bar{x})$
2. $P(\bar{x}) = P_1(\bar{x}) \cdot P_2(\bar{x}) + c$

where P_1 and P_2 are non-constant, variable-disjoint ROPs and $c \in \mathbb{F}$ is a constant.

3.2 Preprocessed Read-Once Polynomials

In this section we extend the model of ROFs by allowing a *preprocessing* step of the input variables. While the basic model is read-once in its variables, the extended model can be considered as read-once in univariate polynomials.

Definition 3.3. *A* preprocessing *is a transformation $T(\bar{x}) : \mathbb{F}^n \to \mathbb{F}^n$ of the form $T(\bar{x}) \triangleq (T_1(x_1), T_2(x_2), \ldots, T_n(x_n))$ such that each T_i is a* non-constant univariate polynomial.

Notice that preprocessings do not affect the PIT problem in the white-box setting as for every n-variate polynomial $P(\bar{y})$ it holds that $P(\bar{y}) \equiv 0$ if and only if $P(T(\bar{x})) \equiv 0$. We now give a formal definition and list some immediate properties.

Definition 3.4. *A* preprocessed arithmetic read-once formula *(PROF for short) over a field $\mathbb{F}$ in the variables $\bar{x} = (x_1, \ldots, x_n)$ is a binary tree whose leafs are labelled with non-constant univariate polynomials $T_1(x_1), T_2(x_2), \ldots, T_n(x_n)$ (all together forming a preprocessing) and whose internal nodes are labelled with the arithmetic operations $\{+, \times\}$ and with a pair of field elements $(\alpha, \beta) \in \mathbb{F}^2$. Each T_i can label at most one leaf. The computation is performed in the following way. A leaf labelled with the polynomial $T_i(x_i)$ and with (α, β) computes the polynomial $\alpha \cdot T_i(x_i) + \beta$. If a node v is labelled with the operation op and with (α, β), and its children compute the polynomials Φ_{v_1} and Φ_{v_2} then the polynomial computed at v is $\Phi_v = \alpha \cdot (\Phi_{v_1}\ op\ \Phi_{v_2}) + \beta$.*

A polynomial $P(\bar{x})$ is a *Preprocessed Read-Once Polynomial* (PROP for short) if it can be computed by a preprocessed read-once formula. A *Decomposition* of a polynomial P is a pair $Q(\bar{z}), T(\bar{x})$ such that $P(\bar{x}) = Q(T(\bar{x}))$ when Q is a ROP and T is a preprocessing. An immediate consequence from the definition is that each PROP admits a decomposition. The following lemma is the PROPs analog of Lemma 3.2.

Lemma 3.5 (PROP Structural Lemma)**.** *Every PROP $P(\bar{x})$ with $|\mathrm{var}(P)| \geq 2$ can be presented in one of the following forms:*

1. $P(\bar{x}) = P_1(\bar{x}) + P_2(\bar{x})$
2. $P(\bar{x}) = P_1(\bar{x}) \cdot P_2(\bar{x}) + c$

where P_1 and P_2 are non-constant, variable-disjoint PROPs and $c \in \mathbb{F}$ is a constant.

3.3 The Algorithm of [SV15]

In this paper we improve the complexity analysis of the PIT algorithm of [SV15]. We begin by describing their algorithm. The heart of the algorithm is a construction of polynomial map $G_{n,t}$ which is shown to a generator for PROPs for a certain range of parameters.

As in [SV15], we fix a set $A = \{\alpha_1, \alpha_2, \ldots, \alpha_n\} \subseteq \mathbb{F}$ of n distinct elements. It also assumed that in case that $\mathbb{F}$ is small, we have access to some extension field of $\mathbb{F}$ with more than n elements. As was shown in [SV15], this assumption is necessary in order to achieve a polynomial-time algorithm.

Definition 3.6 (The generator of [SV15])**.** *Let $t \in \mathbb{N}$. For each $i \in [n]$ let $L_i(y)$ denote the i-th Lagrange Interpolation polynomial. Formally: $L_i(y) \triangleq \frac{\prod_{j\neq i}(y-\alpha_j)}{\prod_{j\neq i}(\alpha_i-\alpha_j)}$. That is, $L_i(y)$ is a degree $n-1$ polynomial satisfying: $L_i(\alpha_j) = 1$ when $j = i$ and $L_i(\alpha_j) = 0$ when $j \neq i$. For each $i \in [n]$, let $G_t^i(y_1, \ldots, y_t, z_1, \ldots, z_t) \triangleq \sum_{k=1}^{t} L_i(y_k) \cdot z_k$. Finally, let $G_{n,t}(y_1, \ldots, y_t, z_1, \ldots, z_t) \triangleq \big(G_t^1(y_1, \ldots, z_t), \ldots, G_t^n(y_1, \ldots, z_t)\big)$.*

$G_{n,t}$ can be seen as a sum of t variable-disjoint copies of $G_{n,1}$, resembling a sum of t independent copies of the same random variable. The main part of the analysis of the algorithm is to establish that for every $n \in \mathbb{N}$ the map $G_{n,\log n}$ is a generator for PROPs on n variables.

Lemma 3.7 ([SV15])**.** *Let $P \in \mathbb{F}[x_1, \ldots, x_n]$ be a non-constant PROP. Then $P(G_{n,\log n})$ is non-constant.*

The intuition behind the proof is that a PROP can be written as either a sum or a product of two variable-disjoint polynomials (Lemma 3.5). Hence, (at least) one of these polynomials contains at most half of the variables. The map $G_{n,t}$ allows to "move" to a smaller polynomial by "shaving" a copy of $G_{n,1}$. Finally, applying Lemma 2.3 one could show that if $G_{n,t}$ is a generator for a class of polynomials, then it can be converted into a relatively small hitting set for that same class.

Lemma 3.8. *Let $P \in \mathbb{F}[x_1, \ldots, x_n]$ be a polynomial of degree d such that $P(G_{n,t}) \not\equiv 0$ for some $t, d \in \mathbb{N}$. Then P has a hitting set of size $(nd)^{\mathcal{O}(t)}$.*

Consequently, the result of Lemma 3.7 translates into a hitting set of size $(nd)^{\mathcal{O}(\log n)}$ for PROPs.

We note that the generator of [SV15] has been used as an ingredient in some subsequent PIT algorithms (e.g. [AvMV15, AFS+16]).

3.4 Our Technical Contribution

In this section we explore additional properties of the generator of [SV15]. The main observation is a structural property of the generator when applied to a polynomial that depends only on a "small" subset of variables. This is the main technical contribution of the paper.

Let $A = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ be the set of elements that is used to define the generator.

Definition 3.9. *For $I \subseteq [n]$, define $\Phi_I(y) \triangleq \prod\limits_{i \in I} (y - \alpha_i)$. For notational convenience, $\Phi_\emptyset(y) \triangleq 1$.*

In order to provide some intuition for the definition, we observe that for any $i \in [n]$ we have that $L_i \sim \Phi_{[n]\setminus\{i\}}$.

Lemma 3.10. *Let $P \in \mathbb{F}[x_1, \ldots, x_n]$ be a homogeneous polynomial of a total degree d and let δ be an upper bound on the individual degrees of all variables x_i in P. Then there exists a polynomial $P'(y)$ of degree at most $\delta \cdot |\mathrm{var}(P)| - d$ such that*

$$P(G_{n,1}(y,z)) = z^d \cdot \Phi_{[n]}^{d-\delta}(y) \cdot P'(y) \cdot \Phi^{\delta}_{[n]\setminus \mathrm{var}(P)}(y).$$

In particular, there exist a polynomial $P'(y)$ of degree at most $d \cdot (|\mathrm{var}(P)| - 1)$ such that

$$P(G_{n,1}(y,z)) = z^d \cdot P'(y) \cdot \Phi^{d}_{[n]\setminus \mathrm{var}(P)}(y).$$

Proof. Let $V \subseteq [n]$ and let $m(\bar{x}) = \alpha \prod\limits_{i \in V} x_i^{e_i}$ be a monomial s.t. $\sum\limits_{i \in V} e_i = d$ and $\forall i \in V : 0 \le e_i \le \delta$.

$$m\left(G_{n,1}(y,z)\right) = \alpha z^d \cdot \prod_{i \in V} L_i^{e_i}(y) = \beta z^d \cdot \prod_{i \in V} \Phi^{e_i}_{[n]\setminus\{i\}}(y) = \beta z^d \cdot \Phi^d_{[n]}(y) / \prod_{i \in V} (y - \alpha_i)^{e_i} =$$

$$\beta z^d \cdot \Phi^{d-\delta}_{[n]}(y) \cdot \Phi^{\delta}_{V}(y) / \prod_{i \in V} (y - \alpha_i)^{e_i} \cdot \Phi^{\delta}_{[n]\setminus V}(y) = z^d \cdot \Phi^{d-\delta}_{[n]}(y) \cdot \beta \prod_{i \in V} (y - \alpha_i)^{\delta - e_i} \cdot \Phi^{\delta}_{[n]\setminus V}(y).$$

Take $m'(y) = \beta \prod_{i \in V} (y - \alpha_i)^{\delta - e_i}$ and observe that degree of $m'(y)$ is $\delta \cdot |V| - d$. By definition, the polynomial P consists of a sum of such monomial where $V = \mathrm{var}(P)$. Therefore, the first claim follows by a linearity argument. The second claim follows by taking $\delta = d$. □

4 Main Result

In this section we prove our main result Theorem 1. We begin by showing that $P(G_{n,1})$ hits sums of univariate polynomials. This proof is available in [SV15] but we reproduce it here for completeness.

Lemma 4.1. *Let $P \in \mathbb{F}[x_1, \ldots, x_n]$ be a non-constant polynomial of the form $P = \sum\limits_{i=1}^{n} T_i(x_i)$. Then $P(G_{n,1}(y,z))$ is non-constant.*

Proof. Pick x_i such that $T_i(x_i)$ is non-constant. Then observe that: $P(G_{n,1})|_{y=\alpha_i} = T_i(z) + \sum\limits_{j \neq i} T_j(0)$. This is a non-constant polynomial and so $P(G_{n,1})$ is non-constant as well. □

We now move to the proof our main result. We want to show that $G_{n,1}$ is a generator for the set of PROPs. The idea is to proceed by induction using Lemma 3.5. Recall that for any $P \in \mathbb{F}[x_1, \ldots, x_n]$ and $i \in \mathbb{N}$, $H_i[P]$ denotes the homogeneous part of degree i of P. Consequently, we can write $P = \sum_{i=0}^{d} H_i[P]$.

Theorem 4.2. *Let $P \in \mathbb{F}[x_1, \ldots, x_n]$ be a non-constant PROP. Then $P(G_{n,1})$ is non-constant.*

Proof. Let d denote the total degree of P. We induct on $m = |\mathrm{var}(P)|$. The base case where $m = 1$ follows from Lemma 4.1. Now, suppose that $m \geq 2$. By the PROP Structural Lemma (Lemma 3.5) we have two cases.

1. $P = P_1 \cdot P_2 + c$. Note that $|\mathrm{var}(P_1)|, |\mathrm{var}(P_2)| \leq m-1$, so by the inductive hypothesis $P_1(G_{n,1})$ and $P_2(G_{n,1})$ are non-constant polynomials and hence their product is non-constant as well. Adding a constant does not affect this.

2. $P = P_1 + P_2$. For $j = 1, 2$: we can write $P_j = \sum\limits_{i=0}^{d} P_{i,j}$ where $P_{i,j} = H_i[P_j]$. By Lemma 3.10, for each $0 \leq i \leq d$ and $j = 1, 2$ there exists a polynomial $P'_{i,j}(y)$ of degree at most $i \cdot (|\mathrm{var}(P_{i,j})| - 1)$ such that

$$P_j(G_{n,1}(y,z)) = \sum_{i=0}^{d} P_{i,j}(G_{n,1}(y,z)) = \sum_{i=0}^{d} z^i \cdot P'_{i,j}(y) \cdot \Phi^i_{[n]\setminus \mathrm{var}(P_{i,j})}(y)$$

and hence

$$P(G_{n,1}(y,z)) = \sum_{i=0}^{d} z^i \cdot \left(P'_{i,1}(y) \cdot \Phi^i_{[n]\setminus \mathrm{var}(P_{i,1})}(y) + P'_{i,2}(y) \cdot \Phi^i_{[n]\setminus \mathrm{var}(P_{i,2})}(y) \right). \tag{1}$$

As before, by the inductive hypothesis $P_1(G_{n,1})$ and $P_2(G_{n,1})$ are non-constant polynomials. Therefore, there exist $1 \leq k \leq d$ such that

$$z^k \cdot P'_{k,1}(y) \cdot \Phi^k_{[n]\setminus \mathrm{var}(P_{k,1})}(y) \not\equiv 0$$

and in particular $P'_{k,1}(y) \not\equiv 0$. Let us denote $V_j = \mathrm{var}(P_{k,j})$ and $W = [n] \setminus (V_1 \cup V_2)$. Consider the expression that corresponds to z^k term in Equation 1:

$$P'_{k,1}(y) \cdot \Phi^k_{[n]\setminus \mathrm{var}(P_{k,1})}(y) + P'_{k,2}(y) \cdot \Phi^k_{[n]\setminus \mathrm{var}(P_{k,2})}(y) \tag{2}$$

As $P_{k,1}$ and $P_{k,2}$ are variable-disjoint, Equation 2 can be rewritten as:

$$P'_{k,1}(y) \cdot \Phi^k_{V_2 \cup W}(y) + P'_{k,2}(y) \cdot \Phi^k_{V_1 \cup W}(y) =$$
$$\Phi^k_W(y) \cdot \left(P'_{k,1}(y) \cdot \Phi^k_{V_2}(y) + P'_{k,2}(y) \cdot \Phi^k_{V_1}(y) \right)$$

The last equality follows from the properties of Φ (see Definition 3.9).

We claim that the obtained expression is non-constant. To this end, it sufficient to show that $P'_{k,1}(y) \cdot \Phi^k_{V_2}(y) + P'_{k,2}(y) \cdot \Phi^k_{V_1}(y) \not\equiv 0$. Assume the contrary. We obtain that $P'_{k,1}(y) \cdot \Phi^k_{V_2}(y) = -P'_{k,2}(y) \cdot \Phi^k_{V_1}(y)$. As V_1 and V_2 are disjoint sets, $\Phi_{V_1}(y)$ and $\Phi_{V_2}(y)$ have no common roots. Therefore, it must be the case that $\Phi^k_{V_1}$ divides $P'_{k,1}$. As $P'_{k,1} \not\equiv 0$, we get that

$$\deg\left(P'_{k,1}\right) \geq \deg\left(\Phi^k_{V_1}\right) = k\,|V_1|$$

while by Lemma 3.10, $P'_{k,1}(y)$ is a polynomial of degree at most $k \cdot (|V_1| - 1)$. Consequently, the coefficient of z^k in $P(G_{n,1}(y,z))$ is non-constant and the claim follows.

□

Theorem 1 follows by combining Theorem 4.2 with Lemma 3.8.

5 Applications

In this section we show two application for our main result, proving Theorems 2 and 3. The first application is testing whether several PROPs sum up to the zero polynomial. To this end, we require the following result which shows that a generator for the class of PROPs can be extended to yield a generator for the class of sums of PROPs.

Lemma 5.1 ([SV15])**.** *Let $\mathcal{G}_n$ be a generator for PROPs on n variables. Then for any $k \in \mathbb{N}$, $\mathcal{G}_n + G_{n,3k}$ is a generator for sums of k PROPs on n variables.*

Remark: As both $\mathcal{G}_n$ and $G_{n,3k}$ represent polynomial maps with the same output length, the sum $\mathcal{G} + G_{n,3k}$ should be interpreted as component-wise sum, where we implicitly assume the variables of $\mathcal{G}_n$ and $G_{n,3k}$ have been relabelled so as to be *disjoint*. The next corollary follows by combining Lemma 5.1 with Theorem 4.2 and the properties of $G_{n,t}$ (see Definition 3.6).

Corollary 5.2. *For any $k, n \in \mathbb{N}$, the map $G_{n,3k+1}$ is a generator for sums of k PROPs on n variables.*

Theorem 2 follows by applying Lemma 3.8. Observe that if $\mathcal{H}$ is hitting set for a sum of two PROPs then $\mathcal{H}$ is an *interpolating* set for a single PROP. That is, the values of a single PROP P on $\mathcal{H}$ contain enough information to uniquely identify P. Indeed, a consequence of Theorem 2 is an interpolating set of polynomial size for PROPs. However in general, $\mathcal{H}$ does not provide us with an efficient algorithm to reconstruct a corresponding PROF. In [SV14] it was shown how to use an interpolating set to devise a reconstruction algorithm with a polynomial overhead.

Lemma 5.3 ([SV14])**.** *Let $n, d \in \mathbb{N}$. There exists a deterministic algorithm that given a hitting set $\mathcal{H}_{n,d}$ for PROPs on n variable and degree at most d, and black-box (oracle) access to a PROP P as above, outputs a PROF Φ that computes P, in time polynomial in n, d and $|\mathcal{H}_{n,d}|$.*

Combining the Lemma with Theorem 1 results in Theorem 3.

6 Conclusions & Open Questions

In this paper we present the first polynomial-time black-box identity testing and reconstruction algorithms for read-once formulas, which form a subclass of multilinear formulas. In [AvMV15], quasi-polynomial-time and polynomial-time PIT algorithms were given for multilinear read-k formulas in the black-box and the white-box settings, respectively for constant values of k. At the high-level, both algorithms goes by alternating the following two steps:

- Step 1: Reduce PIT of a read-$(k+1)$ formula to PIT of sum of two read-k formulas
- Step 2: Reduce PIT of sum of two read-k formulas to PIT of a (single) read-k formula

While Step 2 introduces an overhead of (roughly) $n^{k^{\mathcal{O}(k)}}$ in both settings, the gap in the final complexity results from the overhead introduced by Step 1. Indeed, in the whitebox setting, the overhead is $\text{poly}(n, k)$ while in black-box setting the overhead is $n^{\mathcal{O}(\log n)}$. Moreover, for $k = 0$ the the analysis of Step 2 can be seen as a different analysis of the black-box PIT algorithm for ROFs of [SV15], resulting in roughly the same run time. We hope that the ideas presented in this paper could be extended further to improve the analysis of the black-box PIT algorithms of [AvMV15], and, perhaps lead to new PIT algorithms.

Particular open questions: can one obtain a polynomial-time black-box PIT algorithm for multilinear read-k formula with a constant k? What about $k = 2$? I.e. multilinear read-twice formulas. Even more specifically, can one show a black-box reduction from a PIT instance of a multilinear read-twice formula to polynomially-many PIT instances of sums of constantly-many read-once formulas, introducing only a polynomial overhead?

References

[AFS$^+$16] M. Anderson, M. A. Forbes, R. Saptharishi, A. Shpilka, and B. L. Volk. Identity testing and lower bounds for read-k oblivious algebraic branching programs. In *31st Conference on Computational Complexity, CCC*, pages 30:1–30:25, 2016.

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

[AHK93] D. Angluin, L. Hellerstein, and M. Karpinski. Learning read-once formulas with queries. *J. ACM*, 40(1):185–210, 1993.

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

[ALM$^+$98] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *JACM*, 45(3):501–555, 1998.

[Alo99] N. Alon. Combinatorial nullstellensatz. *Combinatorics, Probability and Computing*, 8:7–29, 1999.

[AM10] V. Arvind and P. Mukhopadhyay. The monomial ideal membership problem and polynomial identity testing. *Information and Computation*, 208(4):351–363, 2010.

[AS98] S. Arora and S. Safra. Probabilistic checking of proofs: A new characterization of NP. *JACM*, 45(1):70–122, 1998.

[ASS13] M. Agrawal, C. Saha, and N. Saxena. Quasi-polynomial hitting-set for set-depth- formulas. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC)*, pages 321–330, 2013.

[ASSS12] M. Agrawal, C. Saha, R. Saptharishi, and N. Saxena. Jacobian hits circuits: Hitting-sets, lower bounds for depth-d occur-k formulas & depth-3 transcendence degree-k circuits. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC)*, pages 599–614, 2012.

[AvMV15] M. Anderson, D. van Melkebeek, and I. Volkovich. Derandomizing polynomial identity testing for multilinear constant-read formulae. *Computational Complexity*, 24(4):695–776, 2015.

[BB98] D. Bshouty and N. H. Bshouty. On interpolating arithmetic read-once formulas with exponentiation. *JCSS*, 56(1):112–124, 1998.

[BC98] N. H. Bshouty and R. Cleve. Interpolating arithmetic read-once formulas in parallel. *SIAM J. on Computing*, 27(2):401–413, 1998.

[BHH95a] N. H. Bshouty, T. R. Hancock, and L. Hellerstein. Learning arithmetic read-once formulas. *SIAM J. on Computing*, 24(4):706–735, 1995.

[BHH95b] N. H. Bshouty, T. R. Hancock, and L. Hellerstein. Learning boolean read-once formulas with arbitrary symmetric and constant fan-in gates. *JCSS*, 50:521–542, 1995.

[BMS11] M. Beecken, J. Mittmann, and N. Saxena. Algebraic independence and blackbox identity testing. In *Automata, Languages and Programming, 38th International Colloquium (ICALP)*, pages 137–148, 2011.

[BOT88] M. Ben-Or and P. Tiwari. A deterministic algorithm for sparse multivariate polynominal interpolation. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC)*, pages 301–309, 1988.

[DL78] R. A. DeMillo and R. J. Lipton. A probabilistic remark on algebraic program testing. *Inf. Process. Lett.*, 7(4):193–195, 1978.

[DS06] Z. Dvir and A. Shpilka. Locally decodable codes with 2 queries and polynomial identity testing for depth 3 circuits. *SIAM J. on Computing*, 36(5):1404–1434, 2006.

[FGT16] S. A. Fenner, R. Gurjar, and T. Thierauf. Bipartite perfect matching is in quasi-nc. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 754–763, 2016.

[FS12] M. Forbes and A. Shpilka. Quasipolynomial-time identity testing of non-commutative and read-once oblivious algebraic branching programs. *Electronic Colloquium on Computational Complexity (ECCC)*, 19:115, 2012.

[FS13a] M. Forbes and A. Shpilka. Explicit noether normalization for simultaneous conjugation via polynomial identity testing. In *APPROX-RANDOM*, pages 527–542, 2013.

[FS13b] M. Forbes and A. Shpilka. Quasipolynomial-time identity testing of non-commutative and read-once oblivious algebraic branching programs. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 243–252, 2013. Full version at http://eccc.hpi-web.de/report/2012/115.

[FSS14] M. Forbes, R. Saptharishi, and A. Shpilka. Pseudorandomness for multilinear read-once algebraic branching programs, in any order. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC)*, pages 867–875, 2014. Full version at http://eccc.hpi-web.de/report/2013/132.

[GKS16] R. Gurjar, A. Korwar, and N. Saxena. Identity testing for constant-width, and commutative, read-once oblivious abps. In *31st Conference on Computational Complexity, CCC*, pages 29:1–29:16, 2016.

[GKST15] R. Gurjar, A. Korwar, N. Saxena, and N. Thierauf. Deterministic identity testing for sum of read-once oblivious arithmetic branching programs. In *30th Conference on Computational Complexity, CCC*, pages 323–346, 2015.

[Gup14] A. Gupta. Algebraic geometric techniques for depth-4 PIT & sylvester-gallai conjectures for varieties. *Electronic Colloquium on Computational Complexity (ECCC)*, 21:130, 2014.

[HH91] T. R. Hancock and L. Hellerstein. Learning read-once formulas over fields and extended bases. In *Proceedings of the 4th Annual Workshop on Computational Learning Theory (COLT)*, pages 326–336, 1991.

[HS80] J. Heintz and C. P. Schnorr. Testing polynomials which are easy to compute (extended abstract). In *Proceedings of the 12th Annual ACM Symposium on Theory of Computing (STOC)*, pages 262–272, 1980.

[KLN$^+$93] M. Karchmer, N. Linial, I. Newman, M. E. Saks, and A. Wigderson. Combinatorial characterization of read-once formulae. *Discrete Mathematics*, 114(1-3):275–282, 1993.

[KMSV13] Z. S. Karnin, P. Mukhopadhyay, A. Shpilka, and I. Volkovich. Deterministic identity testing of depth 4 multilinear circuits with bounded top fan-in. *SIAM J. on Computing*, 42(6):2114–2131, 2013.

[KS01] A. Klivans and D. Spielman. Randomness efficient identity testing of multivariate polynomials. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 216–223, 2001.

[KS07] N. Kayal and N. Saxena. Polynomial identity testing for depth 3 circuits. *Computational Complexity*, 16(2):115–138, 2007.

[KS09] N. Kayal and S. Saraf. Blackbox polynomial identity testing for depth 3 circuits. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 198–207, 2009. Full version at http://eccc.hpi-web.de/report/2009/032.

[KS11] Z. S. Karnin and A. Shpilka. Black box polynomial identity testing of generalized depth-3 arithmetic circuits with bounded top fan-in. *Combinatorica*, 31(3):333–364, 2011.

[KS16a] M. Kumar and S. Saraf. Arithmetic circuits with locally low algebraic rank. In *31st Conference on Computational Complexity, CCC*, pages 34:1–34:27, 2016.

[KS16b] M. Kumar and S. Saraf. Sums of products of polynomials in few variables: Lower bounds and polynomial identity testing. In *31st Conference on Computational Complexity, CCC*, pages 35:1–35:29, 2016.

[LFKN92] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. *JACM*, 39(4):859–868, 1992.

[Lov79] L. Lovasz. On determinants, matchings, and random algorithms. In L. Budach, editor, *Fundamentals of Computing Theory*. Akademia-Verlag, 1979.

[LV03] R. J. Lipton and N. K. Vishnoi. Deterministic identity testing for multivariate polynomials. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 756–760, 2003.

[Muk15] P. Mukhopadhyay. Depth-4 identity testing and noether's normalization lemma. *Electronic Colloquium on Computational Complexity (ECCC)*, 2015.

[MVV87] K. Mulmuley, U. Vazirani, and V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987.

[Sax08] N. Saxena. Diagonal circuit identity testing and lower bounds. In *Automata, Languages and Programming, 35th International Colloquium*, pages 60–71, 2008. Full version at http://eccc.hpi-web.de/eccc-reports/2007/TR07-124/index.html.

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

[Sha90] A. Shamir. IP=PSPACE. In *Proceedings of the Thirty First Annual Symposium on Foundations of Computer Science*, pages 11–15, 1990.

[SS10] N. Saxena and C. Seshadhri. From Sylvester-Gallai Configurations to Rank Bounds: Improved Black-Box Identity Test for Deph-3 Circuits. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 21–30, 2010.

[SS11] N. Saxena and C. Seshadhri. An almost optimal rank bound for depth-3 identities. *SIAM J. Comput.*, 40(1):200–224, 2011.

[SV08] A. Shpilka and I. Volkovich. Read-once polynomial identity testing. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC)*, pages 507–516, 2008.

[SV09] A. Shpilka and I. Volkovich. Improved polynomial identity testing for read-once formulas. In *APPROX-RANDOM*, pages 700–713, 2009. Full version at http://eccc.hpi-web.de/report/2010/011.

[SV14] A. Shpilka and I. Volkovich. On reconstruction and testing of read-once formulas. *Theory of Computing*, 10:465–514, 2014.

[SV15] A. Shpilka and I. Volkovich. Read-once polynomial identity testing. *Computational Complexity*, 24(3):477–532, 2015.

[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.

[Vol16] I. Volkovich. Characterizing arithmetic read-once formulae. *TOCT*, 8(1):2, 2016.

[Zip79] R. Zippel. Probabilistic algorithms for sparse polynomials. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation*, pages 216–226, 1979.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de