Derandomizing Polynomial Identity Testing for Multilinear Constant-Read Formulae

Matthew Anderson
Computer Sciences Dept.
University of Wisconsin
Madison, USA
Email: mwa@cs.wisc.edu

Dieter van Melkebeek
Computer Sciences Dept.
University of Wisconsin
Madison, USA
Email: dieter@cs.wisc.edu

Ilya Volkovich
Computer Science Dept.
Technion
Haifa, Israel
Email: ilyav@cs.technion.ac.il

***Abstract*—We present a polynomial-time deterministic algorithm for testing whether constant-read multilinear arithmetic formulae are identically zero. In such a formula each variable occurs only a constant number of times and each subformula computes a multilinear polynomial. Before our work no subexponential-time deterministic algorithm was known for this class of formulae. We also present a deterministic algorithm that works in a blackbox fashion and runs in quasi-polynomial time in general, and polynomial time for constant depth. Finally, we extend our results and allow the inputs to be replaced with sparse polynomials. Our results encompass recent deterministic identity tests for sums of a constant number of read-once formulae, and for multilinear depth-four circuits.**

***Keywords*-arithmetic circuit; bounded-depth circuit; derandomization; polynomial identity testing;**

I. Introduction

Polynomial identity testing (PIT) denotes the fundamental problem of deciding whether a given polynomial identity holds. More precisely, we are given an arithmetic circuit or formula F on n inputs over a given field $\mathbb{F}$, and wish to know whether all the coefficients of the formal polynomial P computed by F vanish. Due to its basic nature, PIT shows up in many constructions in theory of computing. Particular problems that reduce to PIT include integer primality testing [1] and finding perfect matchings in graphs [2].

PIT has a very natural randomized algorithm – pick the values of the variables uniformly at random from a small set S, and accept iff P evaluates to zero on that input. If $P \equiv 0$ then the algorithm never errs; if $P \not\equiv 0$ then by Schwartz-Zippel [3], [4], [5] the probability of error is at most $d/|S|$, where d denotes the total degree of P. This results in an efficient randomized algorithm for PIT. This algorithm works in a blackbox fashion in the sense that it does not access the representation of the polynomial P, rather it only examines the value of P at certain points (from $\mathbb{F}$ or an extension field of $\mathbb{F}$).

Despite the simplicity of the above randomized algorithm, no efficient deterministic algorithm for PIT is known. In fact, the development of a deterministic subexponential-time algorithm for PIT would imply Boolean or arithmetic circuit / formula lower bounds that have been a central elusive goal in theory of computing for a very long time [6], [7], [8], [9].

Recent years have seen considerable progress on deterministic PIT algorithms for restricted classes of arithmetic formulae, in particular for constant-depth formulae. For depth two several deterministic polynomial-time blackbox algorithms are known [10], [11], [12], [13], [14]. For depth three the state-of-the-art is a deterministic polynomial-time blackbox algorithm when the fanin of the top gate is fixed to any constant [15]. The same is known for depth four but only when the formulae are multilinear, i.e., when every gate in the formula computes a polynomial of degree at most one in each variable [16]. There are also a few incomparable results for rather specialized classes of depth-four formulae [17], [13], [18]. We refer to the excellent survey paper [19] for more information.

Another natural restriction are arithmetic formulae in which each variable appears only a limited number of times. We call such formulae *read-k* , where k denotes the limit. PIT for read-once formulae is trivial in the non-blackbox setting as there can be no cancellation of monomials. Shpilka and Volkovich considered a special type of bounded-read formulae, namely formulae that are the sum of k read-once formulae. For such formulae and constant k they established a deterministic polynomial-time non-blackbox algorithm as well as a deterministic blackbox algorithm that runs in quasi-polynomial time, i.e., in time $n^{O(k+\log n)}$ on formulae with n variables [20], [18].

A. Results

As our main result we present a deterministic polynomial-time PIT algorithm for multilinear constant-read formulae, as well as a deterministic quasi-polynomial-time blackbox algorithm.

Theorem 1. (PIT for Multilinear Bounded-Read Formulae). *There exists a deterministic polynomial identity testing algorithm for multilinear formulae that runs in time $s^{O(1)} \cdot n^{k^{O(k)}}$, where s denotes the size of the formula, n the number of variables, and k the maximum number of times a variable appears in the formula. There also exists a deterministic blackbox algorithm that runs in time $n^{k^{O(k)}+O(k \log n)}$ and queries points from an extension field of size $O(n^2)$.*

Note that Theorem 1 extends the class of formulae which Shpilka and Volkovich could handle since a sum of read-once formulae is always multilinear. This is a *strict* extension – in the full version of this paper [21] we exhibit an explicit read-2 formula with n variables that requires $\Omega(n)$ terms when written as a sum of read-once formulae. The separating example also shows that the efficiency of the PIT algorithm in Theorem 1 cannot be obtained by first expressing the given formula as a sum of read-once formulae and then applying the known PIT tests [18] for sums of read-once formulae to it.

Shpilka and Volkovich actually proved their result for sums of a somewhat more general type of formulae than read-once, namely read-once formulae in which each leaf variable is replaced by a low-degree univariate polynomial in that variable. We can handle an extension in which the leaf variables are replaced by *sparse multivariate* polynomials. We use the term *sparse-substituted* formula for a formula along with substitutions for the leaf variables by multivariate polynomials that are each given as a list of terms (monomials). We call a sparse-substituted formula *read-k* if each variable appears in at most k of those multivariate polynomials.

Theorem 2. (Extension to Sparse-Substituted Formulae). *There exists a deterministic polynomial identity testing algorithm for multilinear sparse-substituted formulae that runs in time $s^{O(1)} \cdot n^{k^{O(k)}(\log(t)+1)}$, where s denotes the size of the formula, n the number of variables, k the maximum number of substitutions in which a variable appears, and t the maximum number of terms a substitution consists of.*

There also exists a deterministic blackbox algorithm for multilinear sparse-substituted formulae that runs in time $n^{k^{O(k)}(\log(t)+1)+O(k \log n)}$ and queries points from an extension field of size $O(n^2)$.

Note Theorem 1 is a specialization of Theorem 2 obtained by setting $t = 1$.

We can further extend our non-blackbox identity test by introducing a relaxed notion of multilinearity for sparse-substituted formulae which requires only that for every multiplication gate of the original formula the different input branches of the gate are variable disjoint. We call such sparse-substituted formulae *structurally-multilinear*. Note that this definition allows the substituted polynomials to be non-multilinear.

Theorem 3. (Extension to Structurally-Multilinear Formulae). *There exists a deterministic polynomial identity testing algorithm for structurally-multilinear sparse-substituted formulae that runs in time $s^{O(1)} \cdot (n \log t)^{k^{O(k)}(\log(t)+1)}$, where s denotes the size of the formula, n the number of variables, k the maximum number of substitutions in which a variable appears, and t the maximum number of terms a substitution consists of.*

We observe that any multilinear depth-four alternating formula with an addition gate of fanin k as the output can be written as the sum of k sparse-substituted read-once formulae, where the read-once formulae are single monomials and the substitutions correspond to multilinear depth-two formulae. This implies that our blackbox algorithm also extends the work by Karnin et al. [22], who established a deterministic quasi-polynomial-time blackbox algorithm for multilinear formulae of depth four. Thus, our results can be seen as unifying identity tests for sums of read-once formulae [18] with identity tests for depth-four multilinear formulae [22] while achieving comparable running times in each of those restricted settings.

We can improve the running time of our blackbox algorithm in the case where the formulae have small depth.

Theorem 4. (Improvement for Bounded-Depth Formulae). *There exists a deterministic blackbox polynomial identity testing algorithm for multilinear sparse-substituted formulae with unbounded fanin that uses $n^{k^{O(k^2)}(\log(t)+1)+O(kd)}$ time and queries points from an extension field of size $O(n^2)$, where n denotes the number of variables, d the depth of the formula, k the maximum number of substitutions in which a variable appears, and t the maximum number of terms a substitution consists of.*

In particular, we obtain a polynomial-time blackbox algorithm for constant-read constant-depth formulae.

For completeness we mention a couple of related results regarding depth-three constant-read formulae that do not require multilinearity. In particular, [23] gives a $n^{2^{O(k^2)}}$ blackbox identity testing algorithm for read-k depth-three formulae. Later work in [24] implies an improved running time of $n^{2^{O(k)}}$ for this algorithm.

B. Techniques

As we have mentioned earlier, polynomial identity testing is trivial for read-once formulae. Our overall approach for multilinear constant-read formulae is a recursive one in which we reduce to instances with smaller read-value and/or fewer variables until we reach a trivial case. Our reduction alternates between two steps and uses as an intermediate stage formulae that are the sum of two multilinear read-k formulae. We refer to such formulae as multilinear $\sum^2$-read-k formulae.

Step 1. Reduce PIT for multilinear read-$(k+1)$ formulae to PIT for multilinear $\sum^2$-read-k formulae.

Step 2. Reduce PIT for multilinear $\sum^2$-read-k formulae to PIT for multilinear read-k formulae.

We unify techniques developed for sums of read-once formulae (the SV-generator from [18]) and techniques developed for multilinear depth-four formulae (the rank bound for depth-three formulae [25], [26], as seen through [22]) with a novel technique for exploring the structure of formulae. We refer to our technique as "shattering". It allows us to bring the rank bound for depth-three formulae to bear on multilinear $\sum^2$-read-k formulae, and enables us to realize Step 2 in both the blackbox and non-blackbox settings. The technique builds on a simpler technique of "fragmentation", which generalizes an idea of [22] and which we also use to realize Step 1 in the blackbox setting. The key technical difficulty lies in showing how fragmentation enables shattering.

Because of space limitations, we focus only on the blackbox result without sparse substitutions in the body of this extended abstract.

In general, a blackbox PIT algorithm for a class $\mathcal{F}$ of formulae is equivalent to the construction of a low-degree polynomial mapping G on few variables such that $F \circ G$ is nonzero for every nonzero $F \in \mathcal{F}$. We refer to such a mapping as a *hitting set generator* for $\mathcal{F}$, and say that G *hits* every $F \in \mathcal{F}$. Testing F on all elements in the image of G when the input variables to G range over some small set produces an identity test. The converse is also true – identity tests imply hitting set generators (see e.g. [18]).

C. Organization

In Section II we develop our ideas and present an overview of the proof. We briefly discuss the techniques that we use and their context within the overall argument. Section III delves more deeply into our fragmentation and shattering technique for multilinear read-k formulae. Section IV discusses how these methods are combined to prove the key technical lemma and the main blackbox result.

The non-blackbox result, the constant depth result, and further extensions as well as all formal proofs are in the full version of the paper, which is available on ECCC [21].

II. Proof Overview

We now discuss the two steps in our construction and their key ingredients in more detail, with a focus on the role of fragmentation and shattering.

A. Fragmenting Multilinear Formulae

Our fragmentation technique for multilinear formulae involves partial derivatives with respect to well-chosen variables. We discuss the technique in detail in Section 3. For now, the statement of the following lemma suffices. For technical reasons, we generalize the notion of read-k to restrict only the occurrences of variables from some subset V. We refer to the generalization as read$_V$-k.

Lemma 5 (Fragmentation Lemma). *Let $\emptyset \subsetneq V \subseteq [n]$, $k \geq 0$, and let F be an n-variate multilinear read$_V$-$(k+1)$ formula that depends on at least one variable in V. There exists a variable $x \in V$ such that $\partial_x F$ is nonzero and is the product of*

1) *subformulae of F each depending on at most $\frac{|V|}{2}$ variables from V (and possibly more variables outside of V), and*
2) *when $k \geq 1$, at most one $\sum^2$-read$_V$-k formula, which is the derivative with respect to x of some subformula of F.*

The lemma helps us in realizing the blackbox version of Step 1 as follows. A hitting set generator for a class $\mathcal{F}$ of formulae also hits products of formulae from $\mathcal{F}$. Thus, by the Fragmentation Lemma, a hitting set generator that hits multilinear $\sum^2$-read-k formulae on n variables as well as multilinear read-$(k+1)$ formulae that depend on at most $n/2$ of the n variables, also hits some nonzero partial derivative of any nonzero multilinear read-$(k+1)$ formula on n variables. Adding an independent random field element turns such a hitting set generator into one that hits every multilinear read-$(k+1)$ formula on n variables. A logarithmic number of

applications of this transformation then turns a hitting set generator for n-variate $\sum^2$-read-k formulae into one for n-variate read-$(k+1)$ formulae. These ideas are formalized in the following reduction (Step 1), the proof of which can be found in the full version of this paper [21]. We use the notation $\mathrm{var}(F)$ to denote the set of variables that a formula F depends on, and for two polynomial maps $\mathcal{G}_1$, $\mathcal{G}_2$ with the same range the notation $\mathcal{G}_1+\mathcal{G}_2$ denotes their component-wise addition.

Lemma 6 (Read-$(k+1)$ PIT $\leq \sum^2$-Read-k PIT). *For an integer $k \geq 1$, let $\mathcal{G}$ be a generator for n-variate multilinear $\sum^2$-read-k formulae, and let F be a nonzero n-variate multilinear read-$(k+1)$ formula. Then, there is a polynomial-time computable polynomial map $G_w : \mathbb{F}^{O(w)} \to \mathbb{F}^n$ with total degree n such that $\mathcal{G} + G_{\log |var(F)|}$ hits F.*

We also use the Fragmentation Lemma as a building block to establish our Shattering Lemma. This is the most involved step in our construction. Once we have our Shattering Lemma, we employ two more ingredients: the SV-generator and the rank bound for depth-three formulae. At a high level, the Shattering Lemma allows us to transform multilinear read-k formulae into depth-three formulae to which the rank bound applies, and the latter enables us to apply the SV-generator and realize Step 2. We first introduce these additional ingredients, then explain how they combine with the Shattering Lemma, and finally sketch how to obtain shattering from fragmentation.

B. The Key Lemma

A polynomial map $G_w : \mathbb{F}^{2w} \to \mathbb{F}^n$ sufficient for Lemma 6 was introduced by Shpilka and Volkovich in [18]. It interpolates all 0-1-vectors of weight at most w and has the following critical property, where $\mathcal{T}_d$ denotes the set of all terms of degree exactly d.

Fact 7 ([18]). *G_w is a hitting set generator for any class $\mathcal{F}$ of multilinear polynomials that is closed under zero-substitutions and is disjoint from $\mathcal{T}_d$ for every $d > w$.*

The approach in [18] for sums of a constant number of read-once formulae is based on Fact 7 and the following fact.

Fact 8 ([18]). [1] *Let $F = \sum_{i=1}^{k} F_i$ be a nonzero formula with each F_i read-once. Let $\bar{\sigma}$ be a point where none of the nonzero first-order partial derivatives of the F_i's vanish. Then $F(\bar{x}+\bar{\sigma}) \notin \mathcal{T}_d$ for any $d \geq 3k$.*

[1] Shpilka and Volkovich refer to this fact as a hardness of representation result and use the term "justifying assignment" for $\bar{\sigma}$.

For a formula F as in Fact 8, consider applying Fact 7 to the class $\mathcal{F}$ consisting of $F(\bar{x}+\bar{\sigma})$ and all its zero-substitutions, for some fixed $\bar{\sigma}$. The first condition of Fact 7, the closure under zero-substitutions of $\mathcal{F}$, holds by construction. As for the second condition, consider a formula F' obtained by substituting into F the components of $\bar{\sigma}$ for some subset X of the variables. For any variable $x \notin X$, we have that $\frac{\partial F'}{\partial x}(\bar{\sigma}) = \frac{\partial F}{\partial x}(\bar{\sigma})$. Thus, if $\bar{\sigma}$ satisfies the hypothesis of Fact 8 for F, then it also satisfies that hypothesis for any substitution F' of the above type. By Fact 8, this shows that $F'(\bar{x}+\bar{\sigma}) \notin \mathcal{T}_d$ for $d \geq 3k$. Noting that $F'(\bar{x}+\bar{\sigma})$ coincides with $F(\bar{x}+\bar{\sigma})$ where all variables in X have been substituted by zero, this means that $\mathcal{F}$ satisfies the second condition of Fact 7. We conclude that for any $\bar{\sigma}$ satisfying the condition of Fact 8, $G_{3k}+\bar{\sigma}$ hits F.

Moreover, since the partial derivatives $\partial_x F_i$ are read-once formulae and PIT for read-once formulae is trivial, we can efficiently find a shift $\bar{\sigma}$ satisfying the conditions of Fact 8 when given access to the formula F – select values for the components of $\bar{\sigma}$ one by one so as to maintain nonzeroness of the nonzero partial derivatives under that setting. One can also use a hitting set generator $\mathcal{G}$ for read-once formulae to generate a shift $\bar{\sigma}$ satisfying the conditions of Fact 8. Fact 7 then shows that $\mathcal{G}+G_{3k}$ is a hitting set generator for sums of k read-once formulae. This is how Shpilka and Volkovich obtained their quasi-polynomial-time blackbox test [18].

We follow the same strategy for Step 2 of our approach, i.e., to reduce PIT for multilinear $\sum^2$-read-k formulae to PIT for multilinear read-k formulae. We use Fact 7 as is, and develop the following equivalent of Fact 8 for sums of (two) multilinear read-k formulae. We use the notation ∂_P to denote the partial derivative with respect to the set of variables P.

Lemma 9 (Key Lemma). *The following holds for some monotone function $R(m) = O(m^3 \log m)$. Let $F = \sum_{i=1}^{m} F_i$, where each F_i is a non-constant multilinear read-k_i formula. If $\bar{\sigma}$ is a common nonzero of the nonzero formulae of the form $\partial_P f$ where f is a subformula of the F_i's and $|P| \leq b \doteq (k-m+1)\cdot 4k\cdot R(k+1)$, then $F(\bar{x}+\bar{\sigma}) \notin \mathcal{T}_n$ for $n > w \doteq (8k\cdot R(k+1))^{k-m+1}$, where $k \doteq \sum_{i=1}^{m} k_i$.*

We sketch the proof idea for the Key Lemma in Section 4. Note that the condition of the Key Lemma involves higher-order derivatives, whereas the corresponding condition in Fact 8 only uses first-order derivatives. Nevertheless, the important properties are preserved: (i) the condition implies that the conclusion holds for $F(\bar{x}+\bar{\sigma})$ as well as for all its zero-substitutions, and

(ii) the condition states that $\bar{\sigma}$ is a common nonzero of some nonzero multilinear read-k formulae which we can easily compute from F.

Thus, given access to F and to an oracle to a polynomial identity test for multilinear read-k formulae, we can efficiently construct a shift $\bar{\sigma}$ such that G_w hits $F(\bar{x}+\bar{\sigma})$ for w sufficiently large with respect to k. We can generate the shift $\bar{\sigma}$ we need, by using a hitting set generator $\mathcal{G}$ for multilinear read-k formulae, resulting in $\mathcal{G}+G_w$ as a hitting set generator for multilinear $\sum^2$-read-k formulae. These ideas are formalized in the following reduction (Step 2) and worked out in the full version of this paper [21].

Lemma 10 ($\sum^m$-Read-k PIT $\leq$ Read-k PIT). *The following holds for some monotone function $R(m) = O(m^3 \log m)$ and any integer $k \geq 1$. Let $\mathcal{G}$ be a generator for n-variate multilinear read-k formulae. Then $\mathcal{G} + G_{w_{m,k}}$ is a generator for n-variate multilinear $\sum^m$-read-k formulae, where $w_{m,k} \doteq (8km \cdot R(km+1))^{(k-1)m+1}$.*

Shpilka and Volkovich show Fact 8 by arguing that applying a sequence of partial derivatives and nonzero substitutions F reduces the degree of the terms in $\mathcal{T}_d$ and zeroes some of the F_i's. If $\mathcal{T}_d$ remains non-trivial after all F_i's are zeroed, the fact is proved. The bound they derive on d depends on how quickly the fanin k is reduced relative to the number of operations performed by the argument. Strong structural properties of read-once formulae make it relatively easy to argue that few partial derivatives and substitutions suffice to zero any particular read-once formula. For formulae of arbitrary read these properties are not readily present. In order to prove the Key Lemma, we employ our fragmentation technique to bring the rank bound for depth-three formulae to bear on multilinear constant-read formulae.

C. The Rank Bound

For constant-depth formulae the difficult cases of PIT are those where the top gate is an addition. In particular, for depth-three these are $\Sigma\Pi\Sigma$-formulae – sums of products of linear functions. A relevant structural property of such formulae F is their *rank*, which Dvir and Shpilka [25] defined as the dimension of the span of the linear functions at the bottom level of F. One way to think about the rank is as the true number r of independent variables of F – we can efficiently transform F into a $\Sigma\Pi\Sigma$-formula $\tilde{F}$ on r variables such that F is identically zero iff $\tilde{F}$ is. In recent years much progress has been made on upper bounds for the rank of $\Sigma\Pi\Sigma$-formulae that are identically zero, where the upper bounds are expressed as a function of the fanin m of the top addition gate. In particular, Saxena and Seshadhri [26] showed that a zero $\Sigma\Pi\Sigma$-formula of syntactic degree d that satisfies some relatively minor conditions can have rank at most $O(m^2 \log d)$ in general, and $O(m^2 \log m)$ if the formula is multilinear. We refer to this structural property as the *rank bound*. The "relatively minor" conditions are (i) that the multiplication gates at the bottom have no non-trivial factor that is common to all of them, and (ii) that no non-trivial subset of the m branches of F sums to zero. A formula is called *simple* when satisfying condition (i) and, *minimal* when satisfying condition (ii).

The rank bound has been frequently used to witness the nonzeroness of depth-three formulae for the purposes of identity testing. For their application to multilinear depth-four formulae, [22] consider multilinear formulae of the form $F = \sum_{i=1}^{m} F_i$ where the F_i's factor into subformulae each depending only on a fraction α of the variables. In such a case we call the formula F α-split.

Definition 11 (α-split). *Let $F = \sum_{i=1}^{m} F_i \in \mathbb{F}[x_1, \ldots, x_n]$, $\alpha \in [0,1]$, and $V \subseteq [n]$. We say that F is* α-split *if each F_i is of the form $\prod_j F_{i,j}$ where $|var(F_{i,j})| \leq \alpha n$. F is* α-split with respect to V *(in shorthand, α-split$_V$) if $|var(F_{i,j}) \cap V| \leq \alpha |V|$ for all i, j.*

[22] lifts the depth-three rank bound to achieve a similar implication for split multilinear depth-four formulae.

Lemma 12. (Rank Bound for Split Multilinear Formulae [22, Lemma 4.5]) *For some monotone function $R(m) = O(m^3 \log m)$ the following holds for any multilinear formula $F = \sum_{i=1}^{m} F_i$ on $n \geq 1$ variables with $\cup_{i \in [m]} var(F_i) = [n]$. If F is simple, minimal, and α-split for $\alpha = (R(m))^{-1}$, then $F \not\equiv 0$.*

We use the rank bound via Lemma 12, not to directly construct our PIT algorithm, but to establish the Key Lemma (Lemma 9). The connection between the two is as follows. Let F' be a sum of a constant number of multilinear formulae. Note that $F' \in \mathcal{T}_{d'}$ iff there exists a nonzero scalar a and an index set I of size d' such that $F' - a \cdot \prod_{i \in I} x_i \equiv 0$. Lemma 12 shows that the latter cannot happen for $d' > 0$ if each of the summands of F' is (i) sufficiently split and (ii) not divisible by any variable. For a shifted formula $F'(\bar{x}+\bar{\sigma})$ the latter condition is met if the summands do not vanish at $\bar{\sigma}$. Thus, in order to establish the Key Lemma, all that

remains is to transform a multilinear $\sum^2$-read-k formula F for which $F(\bar{x}+\bar{\sigma}) \in \mathcal{T}_d$ into a sum F' of a constant number of sufficiently split multilinear formulae such that $F'(\bar{x}+\bar{\sigma}) \in \mathcal{T}_{d'}$ for some $d' > 0$. Moreover, the transformation should be sufficiently simple such that the condition that none of the summands of F' vanish at $\bar{\sigma}$ translates into a simple condition about $\bar{\sigma}$ and the original formula F. Repeated applications of the Fragmentation Lemma allow us to do so (for d sufficiently large compared to k) in a process we refer to as "shattering".

D. Shattering Multilinear Formulae

For the purpose of exposition, let us consider the case $k = 1$, i.e., let $F = F_1 + F_2$ be the sum of two read-once formulae. The Fragmentation Lemma applied to F_i gives a formula $\partial_x F_i$ that is a product of subformulae on at most half of the variables each. When we greedily apply the Fragmentation lemma to a factor which depends on the most variables, $O(1/\alpha)$ applications suffice to ensure that each of the remaining factors depend on at most a fraction α of the variables. If we denote by P the set of variables we used for the partial derivatives, multilinearity implies that the product of all those factors equals $\partial_P F_i$. Thus, the formula $F' \doteq \partial_P F$ is the sum of two split multilinear formulae. Moreover, if $F(\bar{x}+\bar{\sigma}) \in \mathcal{T}_d$ then $F'(\bar{x}+\bar{\sigma}) \in \mathcal{T}_{d'}$ for $d' = d-|P|$, which is positive as long as d is sufficiently large compared to $1/\alpha$. Let F_i' coincide with $\partial_P F_i$, so the condition that F_i' does not vanish at $\bar{\sigma}$ is equivalent to $\partial_P F_i$ not vanishing at $\bar{\sigma}$. This is how higher-order derivatives enter the conditions of the Key Lemma.

In the cases where $k \geq 2$ the shattering process becomes more complicated as it no longer holds that all the factors produced by the Fragmentation Lemma depend on at most half the number of variables – the one $\sum^2$-read-$(k-1)$ factor may depend on more. In Section 3.2 we sketch how we can handle the general situation. The following is the resulting statement, which for generality is given for sums of an arbitrary number of bounded-read formulae.

Lemma 13 (Shattering Lemma). *Let $\alpha : \mathbb{N} \to (0,1]$ be a non-increasing function. Let $F \in \mathbb{F}[x_1, \ldots, x_n]$ be a formula of the form $F = \sum_{i=1}^{m} F_i$, where each F_i is a non-constant multilinear read-k_i formula. There exist disjoint subsets $P, V \subseteq [n]$ such that $\partial_P F$ can be written as $\sum_{i=1}^{m'} F_i'$, where $m' \leq k \doteq \sum_{i=1}^{m} k_i$, each F_i' is multilinear and $\alpha(m'+1)$-split$_V$, the factors of each of the F_i''s are of the form $\partial_{\tilde{P}} f$ where f is some subformula of some F_j and $\tilde{P} \subseteq P$. In addition,*

$$|P| \leq (k-m+1) \cdot \frac{4k}{\alpha(k+1)},$$

and

$$|V| \geq \left(\frac{\alpha(k+1)}{8k}\right)^{k-m} \cdot n - \frac{8k}{\alpha(k+1)}.$$

Note that the formula F' given by the Shattering Lemma may depend on variables outside of V, and that the F_j''s are only split with respect to V, i.e., they are the products of factors that each only depend on a fraction of the variables of V but may depend on many variables outside of V. The formula to which we apply Lemma 12 is obtained from F' by setting the variables outside of V appropriately. If neither the projections nor any of the F_j vanish at $\bar{\sigma}$, we can conclude that $F(\bar{x}+\bar{\sigma}) \notin \mathcal{T}_d$ for any d larger than the number of partial derivatives we needed. Since the variables always appear as subformulae, the condition in the statement of the Key Lemma suffices.

E. Extension to Sparse-Substituted Formulae

The above arguments require the formulae to be multilinear for two reasons. First, they make heavy use of partial derivatives, and multilinear formulae are closed under the partial derivative operation. Second, factors of multilinear formulae are variable disjoint.

We can relax the multilinearity condition somewhat. Few modifications are needed in order to extend our results to multilinear sparse-substituted formulae, i.e., multilinear formulae in which each leaf variable is replaced by a sparse multilinear polynomial such that all multiplication gates of the original formula remain variable disjoint. The main extension happens in the Fragmentation Lemma. A combination of partial derivatives and zero-substitutions similar to one used in [22] allows us to fragment the sparse substitutions. For substitutions that consist of at most t terms, this results in an overall multiplicative increase in the number of such operations by $\log t$. This factor propagates to the exponent of the running time of our algorithms.

A further extension to *structurally*-multilinear sparse-substituted formulae follows by a simple reduction from general sparse substitutions to multilinear sparse substitutions.

III. Fragmenting and Shattering Multilinear Formulae

In this section we expand on two critical ingredients for our arguments – fragmenting and shattering.

A. Fragmenting

We first describe a means of splitting up or *fragmenting* multilinear formulae using partial derivatives. We build up towards this goal by describing how partial derivatives act on multilinear read-$(k+1)$ formulae. We then explain how to fragment such formulae.

Partial derivatives of polynomials can be defined formally over any field $\mathbb{F}$ by stipulating the partial derivative of monomials consistent with standard calculus, and imposing linearity. The well-known sum, product, and chain rules then carry over. For a multilinear polynomial $P \in \mathbb{F}[x_1, ..., x_n]$ and a variable $x = x_i$ with $i \in [n]$, we can write P as $P = Q \cdot x + R$, where $Q, R \in \mathbb{F}[x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n]$. In this case the partial derivative of P with respect to x is $\partial_x P = Q$.

For a multilinear read-$(k+1)$ formula F, $\partial_x F$ is easily obtained from F and results in a formula with the same or a simpler structure than F. Start from the output gate and recurse through the formula, applying at each gate the sum or product rule as appropriate. In the case of an addition gate $g = \sum_i g_i$, we have that $\partial_x g = \sum_i \partial_x g_i$. In the case of a multiplication gate $g = \prod_i g_i$, the derivative $\partial_x g$ is a sum of products, namely $\partial_x g = \sum_i (\prod_{j \neq i} g_j) \cdot \partial_x g_i$. By the multilinearity at most one of the terms in the sum is nonzero because at most one g_i can depend on x. Thus, we leave the branches g_j for $j \neq i$ untouched and recursively replace g_i by its partial derivative. Overall, the resulting formula $\partial_x F$ is multilinear and read-$(k+1)$.

For a *read-once* formula F, fragmenting boils down to the following observation: Taking the partial derivative with respect to the median variable x on which F depends in leaf order, yields a nonzero formula $\partial_x F$ that is the product of subformulae each of which depends on at most half the variables. See Fig. 1 for an example. A key reason this argument goes through is that in read-once formulae every addition gate has children that are variable disjoint. This property implies that at most one branch of an addition gate is nonzero after a partial derivative.

In read-$(k+1)$ formulae with $k \geq 1$, this property no longer holds. We overcome this hurdle by selecting a variable x after repeatedly recursing into the largest branch that depends on a variable that is only present within that branch. This mimics the behavior of the partial derivative of a read-once formula as long as such a branch exists. Once no such branch exists, each child of the current gate cannot contain all the occurrences of any variable. Therefore, these children are read-k formulae. Taking a partial derivative with respect to a variable that only occurs within the current

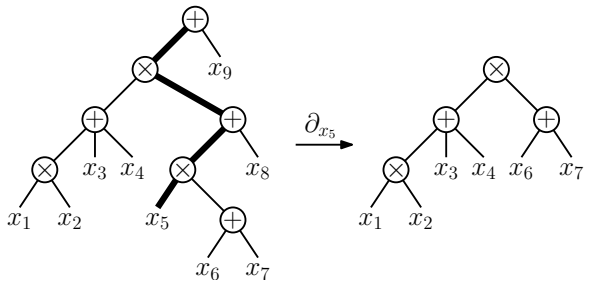

Figure 1. A partial derivative of a read-once formula.

gate eliminates all diverging addition branches above the gate. The resulting formula is a product of all the unvisited (and small) multiplication branches and possibly one (large) $\sum^2$-read-k formulae. This intuition is formalized in the proof of the Fragmentation Lemma (see the full version [21]).

B. Shattering

As mentioned at the beginning of Section 2.4, a $\sum^2$-read-once formula can be shattered by greedily applying the Fragmentation Lemma. In the case of arbitrary read-value $k > 1$, the Fragmentation Lemma is not immediately sufficient for shattering. As in the read-once case, we can apply the lemma greedily to a largest factor of a read-k branch to α-split the branch within at most $O(\frac{1}{\alpha})$ applications. However, this assumes that case 1 of the Fragmentation Lemma always occurs. Otherwise, the greedy process fails to split the formula. To resolve this issue, we leverage the fact that the blocking factor is both large and a $\sum^2$-read-$(k-1)$ formula.

Consider a read-k formula F on n variables. Apply the Fragmentation Lemma to F. Suppose that case 2 of the lemma occurs, producing a variable x, and that the corresponding $\sum^2$-read-$(k-1)$ factor of $\partial_x F$ depends on more than $\frac{n}{2}$ of the variables. Without loss of generality, $\partial_x F = H \cdot (H_1 + H_2)$, where H is a product of read-k formulae each depending on at most $\frac{n}{2}$ variables, and both H_1 and H_2 are read-$(k-1)$ formulae. Rewrite F by distributing the top level multiplication over addition:

$$F' \doteq (H \cdot H_1) + (H \cdot H_2) \equiv H \cdot (H_1 + H_2) = \partial_x F.$$

Let $V \doteq \text{var}(H_1 + H_2)$. F' is explicitly a $\sum^2$-read$_V$-$(k-1)$ formula and a read$_V$-k formula. By further restricting to the largest set of variables that appear the exact same number of times in the larger of the two subformulae H_1 and H_2, we can argue the existence of a subset $V' \subseteq V$ that contains at least a $\frac{1}{2k}$ fraction

of the variables in V such that the read of H_1 and H_2 with respect to V' sum to at most k. This action effectively breaks up the original formula F into two branches without increasing the sum of the read values of the branches. Since $|V| \geq \frac{n}{2}$, the set V' is at most a factor $4k$ smaller than n, and the number of branches increased by one.

This operation can be performed at most $k-1$ times on a read-k formula before either: (i) the attempted greedy splitting is successful, or (ii) the formula becomes the sum of k read$_V$-once formulae, for some non-trivial $V \subseteq [n]$. In the latter case all subsequent splittings will succeed because case 2 of the Fragmentation Lemma cannot occur for read-once formulae. Thus, in either case we obtain a formula with at most k branches that is shattered with respect to a subset V, where $|V|$ is at most a factor $k^{O(k)}$ smaller than n.

Finally, notice that each of the branches in the shattered formula are present in the original input formula, either as such or after taking some partial derivatives. This technical property follows from the properties of the Fragmentation Lemma, and is critical for the application in the Key Lemma. This is the idea behind the Shattering Lemma (see the full version for the formal proof [21]).

IV. Proof of the Key Lemma and Main Result

In order to prove Lemma 9, we first establish a similar lemma for *split* multilinear formulae, and then apply the Shattering Lemma to lift the result to the constant-read setting.

Let $F = \sum_{i=1}^{m} F_i$ be a sufficiently split multilinear formula on n variables. By applying the rank bound for split formulae (Lemma 12) we can argue that if no F_i is divisible by any variable then F cannot compute a term of the form $a \cdot M_n$, where a is a nonzero constant and M_n denotes the monomial $\prod_{i=1}^{n} x_i$. The idea is to consider the formula $F - a \cdot M_n$ and apply the rank bound to it in order to show that it is nonzero. The non-divisibility condition and the natural properties of M_n give simplicity. Minimality effectively comes for free because we are working in the blackbox setting. The splitting required by the rank bound immediately follows from the splitting of F. The property that the branches F_i are not divisible by any variable can be easily established by shifting the formula by a point $\bar{\sigma}$ that is a common nonzero of all the branches F_i. Indeed, if we pick $\bar{\sigma}$ such that $F_i(\bar{\sigma}) \neq 0$ then no variable can divide $F_i(\bar{x} + \bar{\sigma})$. Formalizing this idea yields the following lemma.

Lemma 14. *Let $F = \sum_{i=1}^{m} F_i$ be a multilinear $\alpha(m+1)$-split formula on $n \geq 1$ variables, where $\alpha \doteq \frac{1}{R}$ and R is the function given by Lemma 12. If no F_i vanishes at $\bar{\sigma}$, then $F(\bar{x} + \bar{\sigma}) \not\equiv a \cdot \prod_{i=1}^{n} x_i$ for any nonzero constant a.*

We now show how to lift Lemma 14 from split multilinear formulae to sums of multilinear constant-read formulae. This yields our key lemma – that for such formulae F and a "good" shift $\bar{\sigma}$, $F(\bar{x}+\bar{\sigma})$ cannot compute a term of large degree.

For the sake of contradiction suppose the opposite, i.e., that $F(\bar{x}+\bar{\sigma}) \equiv a \cdot M_n$ for some nonzero constant a and large n. Shatter F into $F' = \partial_P F$ using the Shattering Lemma (Lemma 13), and apply the same operations that shatter F to M_n. $\partial_P M_n$ is a nonzero term of degree $n-|P|$ provided that no component of $\bar{\sigma}$ vanishes. After an appropriate substitution for variables outside of the set V from the Shattering Lemma, we obtain that $F'(\bar{x} + \bar{\sigma}) \equiv a' \cdot M_V$ for some nonzero constant a' and $V \subseteq [n]$, where M_V denotes the product of the variables in V.

At this point we would like to apply Lemma 14 to derive a contradiction. However, we need to have that $|V| > 0$ and that $\bar{\sigma}$ is a common nonzero of all the branches of F'. The former follows from the bounds in the Shattering Lemma provided n is sufficiently large. To achieve the latter condition we impose a stronger requirement on the shift $\bar{\sigma}$ prior to shattering so that afterward $\bar{\sigma}$ is a common nonzero of the shattered branches. The Shattering Lemma tells us that the factors of the branches of the shattered formula are of the form $\partial_{\tilde{P}} f$ where f is some subformula of the F_i's and $\tilde{P} \subseteq P$. Therefore, we require that $\bar{\sigma}$ is a common nonzero for all such subformulae that are nonzero. This is what we mean by a "good" shift.

One additional technical detail is that we must apply a substitution to the variables outside of V that preserves the properties of $\bar{\sigma}$ and does not zero M_n; a typical assignment suffices.

This is the idea behind the Key Lemma.

Combining the Key Lemma with Fact 7 yields Lemma 10, which represents Step 2 of our overall approach as described in Section 1.2 from the Introduction. Lemma 6 represents Step 1. Combining these two steps provides a means of constructing a hitting set generator for multilinear read-$(k+1)$ formulae from a hitting set generator for multilinear read-k formulae. [18] showed that $G_{\lceil \log n \rceil + 1}$ is a hitting set generator for read-once formulae. Iteratively applying the combined reduction and some simple properties of the SV-generator, shows that G_w with seed length $w =$

$k^{O(k)} + k \log n$, is a hitting set generator for multilinear read-k formulae. This implies a blackbox identity test for an n-variate multilinear read-k formula F that runs in time $n^{k^{O(k)}+O(k \log n)}$ and queries elements from field of size $O(n^2)$, as the total degree of $F \circ G_w$ is at most $O(n^2)$. This argues the blackbox part of Theorem 1. The formal proof may be found in the full version of this paper [21].

ACKNOWLEDGMENT

The authors would like to thank Amir Shpilka for bringing them into touch. This research was partially supported by NSF grants 0728809 and 1017597 (M.A. and D.v.M.) and by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement number 257575 (I.V.).

REFERENCES

[1] M. Agrawal and S. Biswas, "Primality and identity testing via chinese remaindering," *Journal of the ACM*, vol. 50, no. 4, pp. 429–443, 2003.

[2] L. Lovász, "On determinants, matchings and random algorithms," in *Fundamentals of Computation Theory*, vol. 79, 1979, pp. 565–574.

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

[4] R. Zippel, "Probabilistic algorithms for sparse polynomials," *Symbolic and Algebraic Computation*, pp. 216–226, 1979.

[5] R. DeMillo and R. Lipton, "A probabilistic remark on algebraic program testing," *Information Processing Letters*, vol. 7, no. 4, pp. 193–195, 1978.

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

[7] M. Agrawal, "Proving lower bounds via pseudo-random generators," *Foundations of Software Technology and Theoretical Computer Science*, pp. 92–105, 2005.

[8] J. Kinne, D. van Melkebeek, and R. Shaltiel, "Pseudorandom generators and typically-correct derandomization," in *Proceedings of the 13th International Workshop on Randomization and Computation*, 2009, pp. 574–587.

[9] S. Aaronson and D. van Melkebeek, "A note on circuit lower bounds from derandomization," Electronic Colloquium on Computational Complexity, Tech. Rep. 105, 2010.

[10] M. Ben-Or and P. Tiwari, "A deterministic algorithm for sparse multivariate polynomial interpolation," in *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, 1988, pp. 301–309.

[11] A. Klivans and D. Spielman, "Randomness efficient identity testing of multivariate polynomials," in *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing*, 2001, pp. 216–223.

[12] M. Agrawal, "On derandomizing tests for certain polynomial identities," in *Proceedings of the 18th Annual IEEE Conference on Computational Complexity*, 2003, pp. 355–359.

[13] V. Arvind and P. Mukhopadhyay, "The ideal membership problem and polynomial identity testing," *Information and Computation*, vol. 208, no. 4, pp. 351–363, 2010.

[14] M. Bläser, M. Hardt, R. Lipton, and N. Vishnoi, "Deterministically testing sparse polynomial identities of unbounded degree," *Information Processing Letters*, vol. 109, no. 3, pp. 187–192, 2009.

[15] 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 ACM Symposium on Theory of Computing*, 2011, to appear.

[16] S. Saraf and I. Volkovich, "Black-box identity testing of depth-4 multilinear circuits," in *Proceedings of the 43rd ACM Symposium on Theory of Computing*, 2011, to appear.

[17] N. Saxena, "Diagonal circuit identity testing and lower bounds," in *Proceedings of the 35th International Colloquium on Automata, Languages and Programming*, 2008, pp. 60–71.

[18] A. Shpilka and I. Volkovich, "Improved polynomial identity testing for read-once formulas," in *Proceedings of the 13th International Workshop on Randomization and Computation*, 2009, pp. 700–713.

[19] N. Saxena, "Progress on polynomial identity testing," *Bulletin of the EATCS*, vol. 99, pp. 49–79, 2009.

[20] A. Shpilka and I. Volkovich, "Read-once polynomial identity testing," in *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, 2008, pp. 507–516.

[21] M. Anderson, D. van Melkebeek, and I. Volkovich, "Derandomizing polynomial identity testing for multilinear constant-read formulae," Electronic Colloquium on Computational Complexity, Tech. Rep. 188, 2010.

[22] Z. Karnin, P. Mukhopadhyay, A. Shpilka, and I. Volkovich, "Deterministic identity testing of depth-4 multilinear circuits with bounded top fan-in," in

Proceedings of the 42nd ACM Symposium on Theory of Computing, 2010, pp. 649–658.

[23] Z. Karnin and A. Shpilka, "Black box polynomial identity testing of generalized depth-3 arithmetic circuits with bounded top fan-in," in *Proceedings of the 23rd Annual IEEE Conference on Computational Complexity*, 2008, pp. 280–291.

[24] N. Saxena and C. Seshadhri, "An almost optimal rank bound for depth-3 identities," in *Proceedings of the 24th Annual IEEE Conference on Computational Complexity*, 2009, pp. 137–148.

[25] Z. Dvir and A. Shpilka, "Locally decodable codes with two queries and polynomial identity testing for depth 3 circuits," *SIAM Journal on Computing*, vol. 36, no. 5, pp. 1404–1434, 2007.

[26] N. Saxena and C. Seshadhri, "From Sylvester-Gallai configurations to rank bounds: Improved black-box identity test for depth-3 circuits," Electronic Colloquium on Computational Complexity, Tech. Rep. 13, 2010.