# Derandomizing Polynomial Identity Testing for Multilinear Constant-Read Formulae

Matthew Anderson[*]
University of Wisconsin-Madison
mwa@cs.wisc.edu

Dieter van Melkebeek[†]
University of Wisconsin-Madison
dieter@cs.wisc.edu

Ilya Volkovich[‡]
Technion
ilyav@cs.technion.ac.il

December 8, 2010

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. Our algorithm runs in time $s^{O(1)} \cdot n^{k^{O(k)}}$, where $s$ denotes the size of the formula, $n$ denotes the number of variables, and $k$ bounds the number of occurrences of each variable. 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 time $n^{k^{O(k)}+O(k \log n)}$ in general, and time $n^{k^{O(k^2)}+O(kd)}$ for depth $d$. 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 formulae.

[*] Partially supported by NSF grants 0728809 and 1017597.

[†] Partially supported by NSF grants 0728809 and 1017597.

[‡] Partially supported by ISF grant 439/06.

# Contents

# 1. 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 want 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 [AB03] and finding perfect matchings in graphs [Lov79].

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 [Sch80, Zip79, DL78] 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. It works in a blackbox fashion in the sense that it does not need access to the representation of the polynomial $P$ but only to the value of $P$ at inputs of the algorithms choosing (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 [KI04, Agr05, KvMS09, AvM10].

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 [BOT88, KS01, Agr03, AM10, BHLV09]. 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 [SS10]. 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 [SV10]. There are also a few incomparable results for rather specialized classes of depth-four formulae [Sax08, AM10, SV09]. We refer to the excellent survey paper [Sax09] 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 constant-read formulae, namely formulae that are the sum of a constant number of read-once formulae. For such formulae 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 $2^{\operatorname{polylog} n}$ on formulae with $n$ variables [SV08, SV09].

## 1.1. 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 1.1 (PIT for Multilinear Constant-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 1.1 extends the class of formulae which Shpilka and Volkovich could handle since a sum of read-once formulae is always multilinear.

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. In the non-blackbox setting we can handle a further extension in which the leaf variables are replaced by *sparse multivariate* polynomials, as long as (i) each variable appears in at most $k$ of those multivariate polynomials, and (ii) for every multiplication gate of the original formula the different input branches of the gate are variable disjoint. 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 it satisfies (i), and *structurally multilinear* if it satisfies (ii). Note that structural multilinearity is a relaxation of multilinearity. In the blackbox setting we only know how to handle multilinear sparse substitutions but not structurally multilinear ones.

**Theorem 1.2. (PIT for Sparse-Substituted Constant-Read 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. 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 1.1 is a specialization of Theorem 1.2 obtained by setting $t = 1$.

We observe that any multilinear depth-four 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. [KMSV10], 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 [SV08, SV09] with identity tests for depth-four multilinear formulae [KMSV10] 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 1.3. (Blackbox PIT for Multilinear Sparse-Substituted Constant-Depth Constant-Read 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 with unbounded fanin.

For completeness we mention a couple related results regarding depth-three constant-read formulae that do not require multilinearity. In particular, [KS08] gives a $n^{2^{O(k^2)}}$ blackbox identity testing algorithm for read-$k$ depth-three formulae. Later work in [SS09] implies an improved running time of $n^{2^{O(k)}}$ for this algorithm.

## 1.2. 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 and PIT for multilinear read-$(k+1)$ formulae on half the number of variables.

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

We combine techniques developed for sums of read-once formulae (the SV-generator from [SV09]) and techniques developed for multilinear depth-four formulae (the rank bound for depth-three formulae [SS09] as in [KMSV10]) with a novel fragmentation technique. The latter enables us to realize Step 1 in the blackbox setting, as well as Step 2 in both the blackbox and non-blackbox settings. The key technical difficulty lies in Step 2, where we show how the fragmentation technique brings the rank bound for depth-three formulae to bear on multilinear $\sum^2$-read-$k$ formulae. We now discuss the two steps and their key ingredients in more detail, with a focus on the role of fragmentation.

### 1.2.1. Fragmenting Multilinear Formulae

Our fragmentation technique for multilinear formulae involves partial derivatives with respect to well-chosen variables. For a read-once formula $F$, it 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. For a general multilinear read-$(k+1)$ formula on $n$ variables, a similar procedure yields the following.

**Lemma 1.4 (Simplified Fragmentation Lemma).** *Given a nonzero multilinear read-$(k+1)$ formula $F$ on $n$ variables, there exists a variable $x$ such that $\partial_x F$ is nonzero and can be written as the product of subformulae on at most $n/2$ variables each, and possibly one other formula that is the derivative of a $\sum^2$-read-$k$ subformula.*

The Fragmentation Lemma helps us in realizing the blackbox version of Step 1 as follows. 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 $F \in \mathcal{F}$. A hitting set generator for $\mathcal{F}$ also hits all products of elements 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 $n$-variable $\sum^2$-read-$k$ formulae into one for $n$-variable read-$(k+1)$ formulae.

The Fragmentation Lemma also plays a critical role in Step 2, both in the blackbox and in the non-blackbox setting. This is the most involved step in our construction and entails two more ingredients: the SV-generator and the rank bound for depth-three formulae. We first introduce those two ingredients and then explain how they combine with the Fragmentation Lemma. At a high level, the Fragmentation 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.

### 1.2.2. The SV-Generator

Shpilka and Volkovich [SV09] defined a hitting set generator $G_w$ that interpolates all 0-1-vectors of weight at most $w$ and has some additional closure properties. Their approach for sums of a constant number of read-once formulae is based on two facts. Let $\mathcal{T}_d$ denote the set of all terms of degree exactly $d$.

**Fact 1.5 ([SV09]).** *$G_w$ 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 $d > w$.*

**Fact 1.6 ([SV09]).** [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 $d \geq 3k$.*

For a formula $F$ as in Fact 1.6, consider applying Fact 1.5 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 1.5, 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 1.6 for $F$, then it also satisfies that hypothesis for any substitution $F'$ of the above type. By Fact 1.6, 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 1.5. We conclude that for any $\bar{\sigma}$ satisfying the condition of Fact 1.6, $G_{3k}+\bar{\sigma}$ 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 1.6 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. This is how Shpilka and Volkovich obtained their polynomial-time non-blackbox test [SV08]. Alternately, one can use a hitting set generator $\mathcal{G}$ for read-once formulae to generate a shift $\bar{\sigma}$ satisfying the conditions of Fact 1.6. Fact 1.5 then

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

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 [SV09].

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 1.5 as is, and develop the following equivalent of Fact 1.6 for sums of (two) multilinear read-$k$ formulae.

**Lemma 1.7 (Informal Simplified Key Lemma).** *Let $F$ be a $\sum^2$-read-k formula, and $\bar{\sigma}$ a point where none of the nonzero partial derivatives of small order of any subformula of $F$ vanish. Then $F(\bar{x} + \bar{\sigma}) \notin \mathcal{T}_d$ for any $d$ that is sufficiently large with respect to $k$.*

Note that the condition of the Key Lemma involves higher-order derivatives, whereas the corresponding condition in Fact 1.6 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 for PIT on 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 $k$. This gives our non-blackbox reduction from PIT on multilinear $\sum^2$-read-$k$ formulae to PIT on multilinear read-$k$ formulae. In the blackbox setting, we can generate the shift $\bar{\sigma}$ we need 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.

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.

### 1.2.3. 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 [DS07] 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 [SS09] showed that a zero $\Sigma\Pi\Sigma$-formula of syntactic degree $d$ that satisfies some relatively minor conditions can have rank at most $O(m^3 \log d)$. We refer to this structural property as the *rank bound*. The "relatively minor" conditions are (i) that the linear functions at the bottom have no nontrivial factor that is common to all of them, and (ii) that no nontrivial subset of the $m$ branches of $F$ sums to zero. Condition (i) is referred to as the simplicity of $F$, and condition (ii) as the minimality of $F$.

The rank bound leads in a fairly straightforward way to a deterministic quasi-polynomial-time non-blackbox PIT algorithm for $\Sigma\Pi\Sigma$-formulae with constant top fanin $m$. If the "relatively minor" conditions are not met, we can reduce to easier cases. Otherwise, the rank bound either tells us that $F$ is guaranteed to be nonzero or else we can reduce $F$ to an equivalent PIT instance $\tilde{F}$ on $r$ variables for some small $r$, which we can solve by brute force in time roughly $(d+1)^r$.

To derive their results for multilinear depth-four formulae, Karnin et al. [KMSV10] established the following implication of the rank bound for “split” formulae. A “split” formula is the product of subformulae that each only depend on a fraction of the variables. We only give a qualitative statement here; see Section 2.3 for the quantitative version.

**Fact 1.8 (Informal and implicit in [KMSV10]).** [2] *A simple minimal sum of a constant number of sufficiently split multilinear formulae cannot be zero.*

We use the rank bound via Fact 1.8, not to directly construct our PIT algorithm, but to establish the Key Lemma. 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 $F' - a \cdot \prod_{i \in I} x_i \equiv 0$. Fact 1.8 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”.

### 1.2.4. Shattering Multilinear Formulae

Let us first 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. We can repeat this process on each of the resulting factors. After $O(1/\alpha)$ applications, each of the remaining factors only depend on a fraction $\alpha$ 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 come into play in 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.

We handle this situation by explicitly expanding the $\sum^2$-read-$(k-1)$ formula into the sum of two read-$(k-1)$ formulae and propagating the sum up to the top addition gate of $F'$. This increases the top fanin of $F'$ and duplicates some of the variable occurrences. However, by restricting to the variables that only appear in the $\sum^2$-read-$(k-1)$ formula and further restricting to the largest group that appear the exact same number of times in the larger of the two terms, we can ensure that the sum of the read-values of the children of $F'$ does not increase. As a result, we need to apply this expansion operation no more than $2k$ times, and the top fanin of $F'$ never grows above $2k$. Since

[2] Karnin et al.[KMSV10] refer to “split” formulae as “compressed”.

we only apply the operation when the $\sum^2$-read-$(k-1)$ formula depends on many variables, the subset of the variables to which we restrict remains large. This result of this process is qualitatively captured in the following lemma.

**Lemma 1.9 (Informal Simplified Shattering Lemma).** *For any $\sum^2$-read-$k$ formula $F$, there exist disjoint sets of indices $P$ and $V$, with $P$ small and $V$ relatively large, such that $\partial_P F$ can be written as $\sum_{j=1}^{m} F'_j$ where $m \leq 2k$ and each $F'_j$ is split with respect to $V$ and is the product of subformulae of partial derivatives of $F_1$ and $F_2$.*

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 Fact 1.8 is obtained from $F'$ by setting the variables outside of $V$ appropriately. If none of 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 for the shattering. Since the variables always appear as subformulae, the condition in the statement of the Key Lemma suffices.

### 1.2.5. 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 [KMSV10] 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.

## 1.3. Organization

In Section 2 we introduce our notation and formally define the classes of arithmetic formulae that we study. Section 2 also reviews some preliminaries in more detail, including the SV-generator and the rank bound for depth-three formulae. In Section 3 we develop the Fragmentation Lemma in a step-wise fashion – for read-once formulae, read-$k$ formulae, and sparse-substituted formulae – and the Shattering Lemma that is based on it. The actual development of our results follows the structure from the introduction, but we prove our results right away for multilinear sparse-substituted formulae. We develop our blackbox and non-blackbox algorithms in parallel. In Section 4 we reduce PIT for multilinear sparse-substituted read-$(k+1)$ formulae to PIT for multilinear sparse-substituted $\sum^2$-read-$k$ formulae. In Section 5 we reduce PIT for multilinear sparse-substituted $\sum^2$-read-$k$ formulae to PIT for multilinear sparse-substituted read-$k$ formulae. In Section 6 we prove our two main theorems for identity testing multilinear constant-read formulae: a deterministic polynomial-time non-blackbox algorithm and a deterministic quasi-polynomial-time blackbox algorithm. We end with a specialization of our approach that gives a deterministic polynomial-time blackbox algorithm for multilinear constant-read constant-depth formulae.

# 2. Notation and Preliminaries

In this section we first review some basics and notation regarding polynomials and arithmetic formulae. We then describe two ingredients we need from prior work, namely the SV-generator and the rank bound.

## 2.1. Polynomials and Arithmetic Formulae

Let $\mathbb{F}$ denote a field. A polynomial $P \in \mathbb{F}[x_1, ..., x_n]$ *depends* on a variable $x_i$ if there are two inputs $\bar{\alpha}, \bar{\beta}$ differing only in the $i^{th}$ coordinate for which $P(\bar{\alpha}) \neq P(\bar{\beta})$. We denote by $\mathrm{var}(P)$ the set of variables that $P$ depends on.

For a subset of the variables $X \subseteq \{x_1, ..., x_n\}$ and an assignment $\bar{\alpha}$, $P|_{X \leftarrow \bar{\alpha}}$ denotes the polynomial $P$ with the variables in $X$ substituted by the corresponding values in $\bar{\alpha}$. We often denote variables interchangeably by their index or by their label: $i$ versus $x_i$, and $[n] \doteq \{1, 2, \ldots, n\}$ versus $\{x_1, ..., x_n\}$.

An *arithmetic formula* is a tree where the leaves are labeled with variables or field elements and internal nodes labeled with addition or multiplication. The singular node with no outgoing edges is the output gate of the formula. We allow generalized input, addition, and multiplication gates, where the result can be multiplied by and/or added to a constant. Formally, for arbitrary constants $\alpha, \beta \in \mathbb{F}$, input and arithmetic gates produce the following output.

- Input gates: $\alpha \cdot x + \beta$.
- Addition gates: $g = \alpha \cdot (\sum_i g_i) + \beta$.
- Multiplication gates: $g = \alpha \cdot (\prod_i g_i) + \beta$.

We interchangeably use the notions of a gate and the polynomial computed by that gate. The size of an arithmetic formula is the number of gates in the formula. The depth of an arithmetic formula is the length of the longest path from the output gate to an input variable. Except for the constant depth case we can assume that the fanin of multiplication and addition gates is two.

### 2.1.1. Restricted Types of Arithmetic Formulae

We consider several restricted classes of arithmetic formulae. An arithmetic formula is *multilinear* if every gate of the formula computes a polynomial that has degree at most one in every variable. This means that only one child of a multiplication gate may depend on a particular variable. However, more than one child may contain occurrences of some variable.

We also consider the restriction that each variable occurs only a bounded number of times.

**Definition 2.1 (read-$k$ formula).** *For $k \in \mathbb{N}$, a* read-$k$ *formula is an arithmetic formula that has at most $k$ occurrences of each variable. For a subset $V \subseteq [n]$, a* read$_V$-$k$ *formula is an arithmetic formula that has at most $k$ occurrences of each variable in $V$ (and an unrestricted number of occurrences of variables outside of $V$).*

Observe that for $V = [n]$ the notion of read$_V$-$k$ coincides with read-$k$.

Given a read-$k$ formula, we can transform it in polynomial time into an equivalent formula that has at most $O(kn)$ gates and where constants only occur in the $\alpha$ and $\beta$ of gates. The transformation is straightforward and is included in Appendix A for completeness. It preserves multilinearity. In a blackbox setting we can assume without loss of generality that the underlying read-$k$ formula is in standard form. Our non-blackbox algorithms will always implicitly start by running the transformation.

We can build more complex formulae by adding several formulae together.

**Definition 2.2 ($\sum^m$-read-$k$ formula).** *For $k, m \in \mathbb{N}$, a* $\sum^m$-read-$k$ *formula is the sum of $m$ read-$k$ formulae.*

Note that this class is not distinct from the class of constant-read formulae because any $\sum^m$-read-$k$ formula is a read-$(km)$ formula.

Finally, we also consider the above types of formulae in which variables can be replaced by sparse polynomials. We call a polynomial is *$t$-sparse* if it consists of at most $t$ terms.

**Definition 2.3 (sparse-substituted formula).** *A* sparse-substituted *formula is an arithmetic formula where each leaf is replaced by a sparse multivariate polynomial given as a list of terms. Further,*

1. *if every variable occurs in at most $k$ of the sparse polynomials, we say that the formula is* read-$k$, *and*

2. *if for every multiplication gate $g$ and every variable $x$ there is at most one multiplicand of $g$ that depends on $x$, we say that the formula is* structurally multilinear.

A sparse-substituted formula is multilinear if every gate (including the substituted input gates) computes a multilinear function. This is equivalent to all multiplication gates in the backbone formula being variable-disjoint, and the sparse substitutions being multilinear. The corresponding interpretation of structural multilinearity is that the multiplication gates in the backbone formula are variable-disjoint, but the substituted sparse polynomials do not need to be multilinear. Thus, structural multilinearity is more general than multilinearity.

### 2.1.2. Partial Derivatives

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 $\frac{\partial P}{\partial x} = Q$. We often shorten this notation to $\partial_x P$. Observe that $R = P|_{x \leftarrow 0}$.

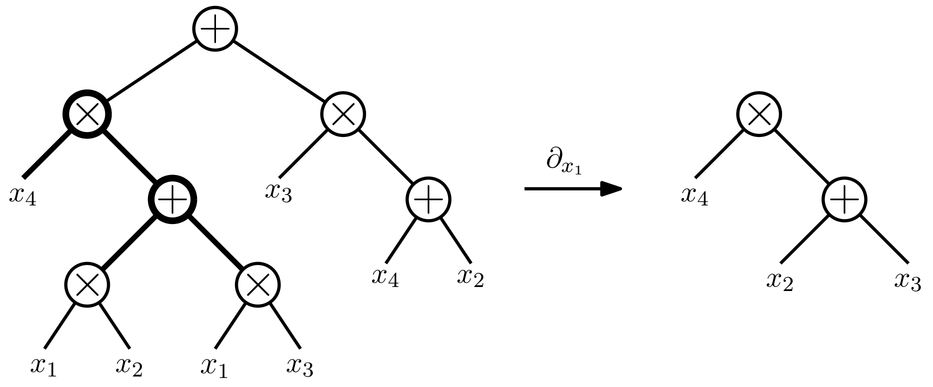

Figure 1: An example of taking the partial derivative of a multilinear read-2 formula.

For a multilinear read-$k$ 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 $g = \alpha \cdot \sum_i g_i + \beta$, we have that $\partial_x g = \alpha \cdot \sum_i \partial_x g_i$. Thus, we essentially recursively replace each of the children by their partial derivative. The structure of the formula is maintained, except that some children may disappear because they do not depend on $x$. In the case of a multiplication gate $g = \alpha \cdot (\prod_i g_i) + \beta$, the derivative $\partial_x g$ is a sum of products, namely $\partial_x g = \alpha \cdot \sum_i (\prod_{j \neq i} g_j) \cdot \partial_x g_i$. However, by the multilinearity condition 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, recursively replace $g_i$ by its partial derivative, and set $\beta = 0$. The structure of the formula is again maintained or simplified. Overall, the resulting formula $\partial_x F$ is multilinear and read-$k$. See Figure 1 for an example with each $\alpha = 1$ and $\beta = 0$. Similarly, the partial derivatives of multilinear $\sum^m$-read-$k$ formulae are multilinear $\sum^m$-read-$k$ formulae.

### 2.1.3. Polynomial Identity Testing and Hitting Set Generators

Arithmetic formula identity testing denotes the problem of deciding whether a given arithmetic formula is identically zero as a formal polynomial. More precisely, let $F$ be an arithmetic formula on $n$ variables over the field $\mathbb{F}$. The formula $F$ is identically zero iff all coefficients of the formal polynomial that $F$ defines vanish. For example, the formula $(x-1)(x+1) - (x^2 - 1)$ is identically zero but the formula $x^2 - x$ is nonzero (even over the field with 2 elements).

There are two general paradigms for identity testing algorithms: *blackbox* and *non-blackbox*. In the non-blackbox setting, the algorithm is given the description of the arithmetic formula as input. In the blackbox setting, the algorithm is allowed only to make queries to an oracle that evaluates the formula on a given input. Observe that non-blackbox identity testing reduces to blackbox identity testing because the description of a formula can be used to efficiently evaluate the formula on each query the blackbox algorithm makes. There is one caveat – in the blackbox case the algorithm should be allowed to query inputs from a sufficiently large field. This may be an extension field if the base field is too small. Otherwise, it is impossible to distinguish a polynomial that is functionally zero over $\mathbb{F}$ but not zero as a formal polynomial, from the formal zero polynomial (e.g., consider the formula $x^2 - x$ restricted to the field with two elements).

Blackbox algorithms for a class of polynomials naturally produce a *hitting set*, i.e., a set $H$ of points such that each nonzero polynomial $P$ from the class does not vanish at some point in $H$. We say that $H$ *hits* the class, and $P$ in particular. To see the connection, observe that when a blackbox algorithm queries a point that is nonzero it can immediately stop. Conversely, when the result of every query is zero, the algorithm must conclude that the polynomial is zero; otherwise, it fails to correctly decide the zero polynomial.

A related notion is that of a *hitting set generator*. Formally, a polynomial map $\mathcal{G} = (\mathcal{G}_1, \mathcal{G}_2, \dots, \mathcal{G}_n)$ where each $\mathcal{G}_i \in \mathbb{F}[y_1, y_2, \dots, y_\ell]$ is a hitting set generator (or generator for short) for a class of polynomials $\mathcal{P}$, if for each nonzero polynomial $P \in \mathcal{P}$, $\mathcal{G}$ *hits* $P$, that is $P \circ \mathcal{G} \not\equiv 0$. Suppose that $\mathcal{G}$ hits a class of polynomials $\mathcal{P}$, then $\mathcal{G}$ can be used to construct a blackbox identity test for $P \in \mathcal{P}$ by collecting all elements in the image of $\mathcal{G}$ when we let the input variables to $\mathcal{G}$ range over some small set.

**Proposition 2.4.** *Let $\mathcal{P}$ be a class of $n$-variate polynomials of total degree at most $d$. Let $\mathcal{G} \in (\mathbb{F}[y_1, ..., y_\ell])^n$ be a generator for $\mathcal{P}$ such that the total degree of each polynomial in $\mathcal{G}$ is at most $d_{\mathcal{G}}$. There is a deterministic blackbox polynomial identity testing algorithm for $\mathcal{P}$ that runs in time $O((d \cdot d_{\mathcal{G}})^\ell)$ and queries points from an extension field of size $O(d \cdot d_{\mathcal{G}})$.*

*Proof.* Let $P$ be a nonzero polynomial in $\mathcal{P}$. Since $\mathcal{G}$ is a generator for $\mathcal{P}$, the polynomial $P \circ \mathcal{G} \in \mathbb{F}[y_1, y_2, \dots, y_\ell]$ is nonzero. The total degree of $P \circ \mathcal{G}$ is at most $d \cdot d_{\mathcal{G}}$. By the Schwartz-Zippel Lemma [Sch80, Zip79, DL78] any set $V^\ell \subseteq \mathbb{E}^\ell$ where $|V| \geq d \cdot d_{\mathcal{G}} + 1$ and $\mathbb{E}$ is an extension field of $\mathbb{F}$, contains a point at which $P \circ \mathcal{G}$ does not vanish. Note that the extension field $\mathbb{E} \supseteq \mathbb{F}$ must be sufficiently large to support the subset $V$ of the required size. The algorithm tests $P$ at all points in $\mathcal{G}(V^\ell)$ and outputs zero iff all test points are zero. ∎

Note that this approach is only efficient when $\ell \ll n$ and the degrees are not too large.

Hitting set generators and a hitting sets are closely related. By Proposition 2.4 a hitting set generator implies a hitting set. It is also known that a hitting set generator can be efficiently constructed from a hitting set using polynomial interpolation [SV09].

## 2.2. The SV-Generator

One example of such a generator is the one Shpilka and Volkovich obtained by interpolating the set $H_w^n$ of all points in $\{0,1\}^n$ with at most $w$ nonzero components. The resulting generator $G_w$ is a polynomial map of total degree $n$ on $2w$ variables. Shpilka and Volkovich [SV09] showed that it hits $\sum^k$-read-once formulae for $w \geq 3k + \log n$. Karnin et al. [KMSV10] also used it to construct a hitting set generator for multilinear depth-four formulae with bounded top fanin.

For completeness, we include the definition of the generator $G_w$.

**Definition 2.5 (SV-Generator [SV09]).** *Let $a_1, \dots, a_n$ denote $n$ distinct elements from $\mathbb{E} \supseteq \mathbb{F}$, an $u_i(x) \doteq \prod_{j \neq i} \frac{x - a_j}{a_i - a_j}$, for $i \in [n]$, the corresponding Lagrange interpolants. For every $w \geq 1$, define*

$$G_w(y_1, ..., y_w, z_1, ..., z_w) \doteq \left( \sum_{j=1}^{w} u_1(y_j) z_j, \sum_{j=1}^{w} u_2(y_j) z_j, ..., \sum_{j=1}^{w} u_n(y_j) z_j \right).$$

The map $G_w$ has a number of useful properties that follow immediately from its definition. We list the ones we need.

**Proposition 2.6 ([KMSV10, Observations 4.2, 4.3]).** *Let $w_1, w_2$ be positive integers.*

1. $G_{w_1}(\bar{y}, \bar{0}) \equiv 0$.
2. $G_{w_1}|_{y_{w_1} \leftarrow a_i} = G_{w_1 - 1} + z_{w_1} \cdot \bar{e}_i$*, where $\bar{e}_i$ denotes the 0-1-vector with a single 1 in position $i$.*
3. $G_{w_1} + G_{w_2} = G_{w_1+w_2}$.

The first item states that zero is in the image of $G$. The second item shows how to make a single output component (and no others) depend on a particular $z_j$. The final item shows that sums of independent copies of $G$ are equivalent to a single copy of $G$ with the appropriate weight parameter $w$.

Proposition 2.6 implies the following: if $\mathcal{G}$ is a generator that hits some partial derivative of a multilinear polynomial then $\mathcal{G} + G_1$ hits the multilinear polynomial itself.

**Lemma 2.7 (Implicit in [KMSV10]).** *Let $P$ be a multilinear polynomial. If $\mathcal{G}$ hits a nonzero $\partial_x P$ for some variable $x$, then $P \circ (\mathcal{G} + G_1)$ is non-constant.*

*Proof.* Since $\partial_x P \not\equiv 0$, both $|\mathrm{var}(P)| \geq 1$ and $x \in \mathrm{var}(P)$. Consider $P(\mathcal{G} + G_1(a_x, z_1))$, that is, evaluating $P$ at $\mathcal{G} + G_1(y_1, z_1)$ and selecting the seed $y_1$ so that $z_1$ only appears in variable $x$. By the multilinearity of $P$, we can write $P$ as $P = x \cdot \partial_x P + P|_{x \leftarrow 0}$. Property 2 of Proposition 2.6 then implies that

$$P(\mathcal{G} + G_1(a_x, z_1)) = ((\mathcal{G})_x + z_1) \cdot \partial_x P(\mathcal{G}) + P|_{x \leftarrow 0}(\mathcal{G}).$$

As $z_1$ does not occur in $(\mathcal{G})_x$ we know that that $(\mathcal{G})_x + z_1$ is nonzero and depends on $z_1$. Because $\mathcal{G}$ hits $\partial_x P$ we see that the second factor of the first term is also nonzero implying that the entire first term is nonzero. Since the second term is independent of $z_1$, there is no way that the $z_1$ dependence of the first term can be canceled out by the second term. This implies that $P \circ (\mathcal{G} + G_1)$ depends on $z_1$ and hence is non-constant as claimed. ■

We require one further property of $G$ that is implicit in [SV08, SV09], namely Fact 1.5 from the introduction.

**Lemma 2.8 (Implicit in [SV08, Theorem 8.2]).** *For $d \in \mathbb{N}$, let $\mathcal{T}_d$ denote the class of all terms of degree $d$, i.e., all monomials of degree $d$ multiplied by an arbitrary nonzero coefficient. The map $G_w$ is a hitting set generator for any class $\mathcal{P}$ of multilinear polynomials that is closed under zero-substitutions and is disjoint from $\mathcal{T}_d$ for every $d > w$.*

*Proof.* Define the set

$$H_w^n \doteq \{\bar{x} \in \{0,1\}^n \mid \sum_{i=1}^{n} x_i \leq w\}.$$

The set $H_w^n$ is the set of vectors in $\{0,1\}^n \subseteq \mathbb{F}^n$ with at most $w$ nonzero components. The set $H_w^n$ is in the image of $G_w$. To see this, consider $\bar{x} \in H_w^n$. Let $\{c_i\}_{i=1}^{r}$ be the index set of at most $w$ nonzero components of $\bar{x}$. We can set the seed to $G_w$ so that $G_w$ evaluates to $\bar{x}$: For $i \in [r]$, set

$y_i \leftarrow a_{c_i}$ (that is, the constant selecting the $c_i^{th}$ component) and $z_i \leftarrow 1$; set $y_i = z_i \leftarrow 0$ for $i > r$. Then $G_w(\bar{y}, \bar{z}) = \bar{x}$.

Fix $P \in \mathbb{F}[x_1, \ldots, x_n]$ in $\mathcal{P}$. Since the image of $G_w$ contains $H_w^n$, it is sufficient to prove that if $P|_{H_w^n} \equiv 0$ then $P \equiv 0$.

We prove the lemma by induction on $n$. If $n \leq w$, then $H_w^n$ is all of $\{0,1\}^n$. Since $P$ is multilinear, it is uniquely defined by its values on $\{0,1\}^n$. Therefore, $P|_{H_w^n} \equiv 0$ implies $P \equiv 0$, completing the base case.

Now, consider $n > w$ and suppose that $P|_{H_w^n} \equiv 0$. For some $j \in [n]$, let $P' \doteq P|_{x_j \leftarrow 0}$. By the closure under zero-substitutions of $\mathcal{P}$, $P' \in \mathcal{P}$. Since $H_w^{n-1}$ is a projection of $H_w^n \cap \{\bar{x} \in \{0,1\}^n | x_j = 0\}$, we have that $P'|_{H_w^{n-1}} \equiv 0$. By the induction hypothesis $P|_{x_j \leftarrow 0} = P' \equiv 0$. By Gauss' Lemma this implies that $x_j | P$.

The above argument works for any $j \in [n]$, so $x_j | P$ for all $j \in [n]$. Hence, $(\prod_{i=1}^n x_i) | P$. Since $P$ is multilinear we have that $P = a \cdot \prod_{i=1}^n x_i$ for some constant $a$. Since $P \in \mathcal{P}$ and $\mathcal{P} \cap \mathcal{T}_n = \emptyset$ for $n > w$, we conclude that $a = 0$. Thus $P \equiv 0$, completing the proof. ∎

## 2.3. The Rank Bound

The rank bound is a property that allows us to witness that certain formulae are nonzero without exhibiting a point where they do not vanish. It was incepted for depth-three formulae in [DS07] and later also applied to multilinear depth-four formulae in [KMSV10].

Let us start with some terminology. An additive top fanin $m$, formula $F = \sum_{i=1}^m F_i$ is said to be *simple* if the greatest common divisor of the $F_i$'s is in $\mathbb{F}$. $F$ is said to be *minimal* if for all non-trivial subsets $S \subsetneq [m]$, $\sum_{i \in S} F_i \not\equiv 0$. If $F$ has depth three, each of the $F_i$'s are products of linear functions. The *rank* of such a formula $F$ is the dimension of the linear space spanned by all those linear functions. A line of research initiated by [DS07] studies the connection between the rank of depth-three formulae and whether it can be zero. For multilinear depth-three formulae, the following represents the state of the art.

**Lemma 2.9 (Rank Bound for Multilinear Depth-Three Formulae [SS09]).** [3] *There is an increasing function $R'(m)$, bounded above by $O(m^3 \log m)$, such that the following holds for any multilinear depth-three formula with additive top fanin $m$: If $F$ is simple and minimal and has rank at least $R'(m)$ then $F \not\equiv 0$.*

For their application to multilinear depth-four formulae, [KMSV10] 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 $\alpha$ of the variables. In such a case we call the formula $F$ $\alpha$-split.

**Definition 2.10 ($\alpha$-split).** *Let $F = \sum_{i=1}^m F_i \in \mathbb{F}[x_1, \ldots, x_n]$, $\alpha \in [0,1]$, and $V \subseteq [n]$. $F$ is* $\alpha$-split *if each $F_i$ is of the form $\prod_j F_{i,j}$ where $|var(F_{i,j})| \leq \alpha n$. $F$ is* $\alpha$-split with respect to $V$ *(in shorthand, $\alpha$-split$_V$) if $|var(F_{i,j}) \cap V| \leq \alpha |V|$ for all $i, j$.*

For $V = [n]$, the two definitions coincide. Note in the definition of split we do not require that $\text{var}(F) = [n]$.

[3] [SS09] only discusses the case of linear *forms*, i.e., linear functions without constant terms. The more general result follows by homogenization. See Appendix B for the details.

Karnin et al. [KMSV10] show how to transform a split simple minimal multilinear formula $F = \sum_{i=1}^{m} F_i$ into a simple minimal depth-three formula $F' = \sum_{i=1}^{m} F'_i$ and then apply the rank bound for $F'$ in order to show that $F \not\equiv 0$. The following formalization quantifies Fact 1.8 from the introduction. We include a proof for completeness.

**Lemma 2.11 (Rank Bound for Split Multilinear Formulae [KMSV10, Lemma 4.5]).** *For $R(m) = m \cdot R'(m)$, where $R$ is the function given by Lemma 2.9, 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 $\alpha$-split for $\alpha = (R(m))^{-1}$, then $F \not\equiv 0$.*

*Proof.* We show that $R(m) = m \cdot R'(m)$ does the job, where $R'(m)$ is the function from the rank bound for multilinear depth-three formulae (Lemma 2.9).

Without loss of generality write: $F = \sum_{i=1}^{m} \prod_j F_{i,j}$, where the $F_{i,j}$ are irreducible. We can construct a set $U \subseteq [n]$ of size $|U| \geq \frac{1}{\alpha \cdot m} = R'(m)$ such that for all $i, j$, $|U \cap \text{var}(F_{ij})| \leq 1$. A greedy construction works (see [KMSV10, Lemma 3.2] for the details).

Assigning all variables outside of $U$ linearizes each $F_{ij}$ – in fact, each $F_{ij}$ becomes a univariate linear function – and turns $F$ into a depth-three formula $F'$ with an addition gate on top of fanin $m$. Moreover, as we will argue, a typical assignment (chosen from a sufficiently large extension field $\mathbb{E} \supseteq F$) to the variables outside of $U$ keeps $F'$ simple and minimal, and ensures that its rank is at least the number $|U|$ of remaining variables. Since $|U| \geq R'(m)$, the rank bound for multilinear depth-three formulae then implies that $F' \not\equiv 0$, and therefore that $F \not\equiv 0$.

All that remains is to establish the above claim about a typical assignment $\bar{\beta}$ to the variables in $[n] \setminus U$.

To argue simplicity, we make use of the following property of multilinear polynomials $P$ and $Q$: If $P$ is irreducible and depends on a variable $x$, then $P|Q$ iff $P|_{x\leftarrow 0} \cdot Q - P \cdot Q|_{x\leftarrow 0} \equiv 0$. Since $F$ is simple, for every irreducible subformula $F_{ij}$ that depends on some $u \in U$, there is branch, say $F_{i'}$, such that $F_{ij}$ does not divide $F_{i'}$. Thus, by the above property, $D \doteq F_{ij}|_{U\leftarrow 0} \cdot F_{i'} - F_{ij} \cdot F_{i'}|_{U\leftarrow 0} \not\equiv 0$. Let $F'_{ij}$ be the result of applying $\bar{\beta}$ to $F_{ij}$, and define $F'_{i'}$ and $F'$ similarly. A typical assignment $\bar{\beta}$ keeps $D$ nonzero and $F'_{ij}$ dependent on $u$. Since $F'_{ij}$ remains irreducible as a univariate polynomial, the above property implies that $F'_{ij}$ does not divide $F'_{i'}$. Therefore, $F'$ is simple.

Minimality is maintained by a typical assignment since if $\sum_{i\in S} F_i$ is a nonzero polynomial for all $\emptyset \subsetneq S \subsetneq [m]$, then the same holds after a typical partial assignment $\bar{\beta}$.

Finally, for any $u \in U$ there exists at least one $F_{ij}$ for which $u \in \text{var}(F_{ij})$. A typical assignment to the variables in $F_{ij}$ other than $u$ turns $F_{ij}$ into a non-constant linear function of $u$. Since distinct $u \in U$ are independent, this means that the rank under a typical assignment $\bar{\beta}$ is at least $|U|$. ∎

# 3. Fragmenting and Shattering Multilinear Formulae

In this section we describe a means of splitting up or *fragmenting* multilinear sparse-substituted formulae. We build up towards this goal by first fragmenting read-once formulae, and then multilinear read-$k$ formulae. We conclude by extending our fragmentation technique to work for sparse-substituted formulae, proving our final Fragmentation Lemma (Lemma 3.3).

We view the Fragmentation Lemma as an atomic operation that breaks a read-$k$ formula into a product of easier formulae, at the cost of some partial derivatives and zero-substitutions. By greedily applying the Fragmentation Lemma and using some other ideas we are able to *shatter*

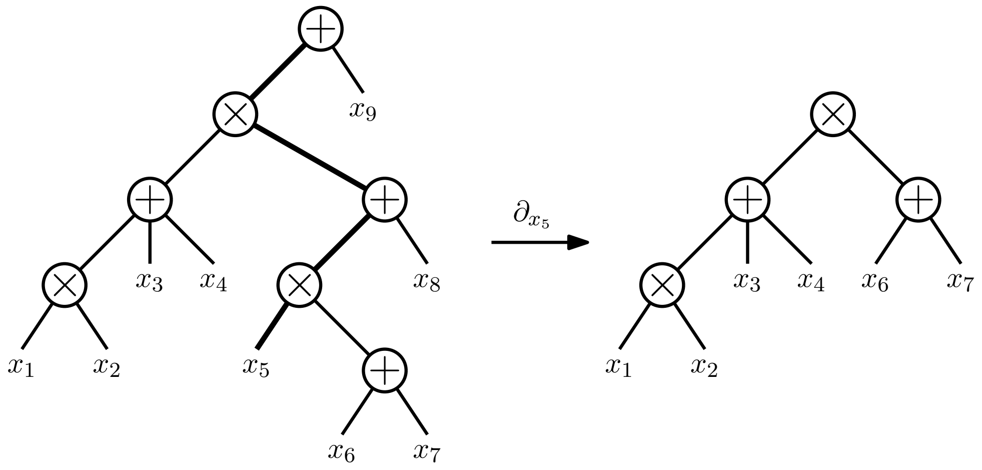

Figure 2: An example of fragmenting a read-once formula.

multilinear sparse-substituted $\sum^m$-read-$k$ formulae, that is, simultaneously split all the top-level branches so that they are the product of factors that each only depend on a fraction of the variables. The Shattering Lemma (Lemma 3.4) is the main result of this section and formalizes Lemma 1.9 from the introduction.

## 3.1. Fragmenting Read-Once Formulae

Let $F$ be a read-once formula. Consider a traversal of the variables of $F$ in leaf order. For the median variable $x$ in this traversal, $\partial_x F$ is $\frac{1}{2}$-split. This is because the path from the root of $F$ to $x$ partitions the formula into halves that can only combine at the top multiplication gate of $\partial_x F$. Moreover, by only considering variables on which $F$ depends, we can make sure that $\partial_x F \not\equiv 0$ assuming $F$ is non-constant. We make this intuition more formal in the following lemma and refer to Figure 2 for an example. For generality, we state the lemma with respect to a restricted variable set $V \subseteq [n]$. Recall that a formula $F$ is $\alpha$-split$_V$ if it is the product of formulae each depending on fewer than $\alpha|V|$ variables from $V$ (and possibly more variables outside of $V$).

**Lemma 3.1 (Fragmenting Read-Once Formulae).** *Let $\emptyset \subsetneq V \subseteq [n]$ and let $F$ be an $n$-variable multilinear read$_V$-once 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 subformulae of $F$ that each depend on at most $\frac{|V|}{2}$ variables from $V$ (and possibly more variables outside of $V$).*

*Proof.* Without loss of generality, it can be assumed that $V$ only contains variables on which $F$ depends. Let $S$ be a sequence of the variables in $V$ produced by an in-order traversal of $F$. Select the variable $x \doteq S_{\lfloor |V|/2 \rfloor}$, which has no more than $|V|/2$ of the variables in $V$ to the left or right of it in the traversal. Then $\partial_x F$ is nonzero (since $F$ depends on $x$) and can be written in the required form. To see the latter, follow the procedure for computing $\partial_x F$ described in Section 2.1.2 by tracing the path in $F$ from the root to the leaf labeled $x$. By the sum rule, $\partial_x$ eliminates the diverging addition branches along this path as those branches to not depend on $x$. Multiplicative branches that do not depend on $x$ are split off as factors. None of those factors of $\partial_x F$ can depend on more than $|V|/2$ variables from $V$. This is because the variables within each factor cannot span both sides of the formula around the path from the output gate to $x$. ■

## 3.2. Fragmenting Multilinear Read-$k$ Formulae

While illustrating the basic idea of fragmenting, Lemma 3.1 is insufficient for our purposes. A key reason the proof of Lemma 3.1 goes through is that in read-once formulae each addition gate has children that are variable disjoint. This property allows the argument to recurse into a single addition branch. In read-$k$ formulae this is no longer the case. Our solution is to follow the largest branch that depends on a variable that is only present within that branch. This allows us to mimic the behavior of the read-once approach 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 $x$. This means that these children are read-$(k-1)$ formulae. Taking a partial derivative with respect to a variable that only occurs within the current gate eliminates all diverging addition branches above the gate. This makes the resulting formula multiplicative in all the unvisited (and small) multiplication branches. This intuition can be formalized in the following lemma, which generalizes Lemma 1.4 from the introduction.

**Lemma 3.2 (Fragmenting Multilinear Read-*k* Formulae).** *Let $\emptyset \subsetneq V \subseteq [n]$, $k \geq 2$, and let $F$ be an $n$-variable multilinear read$_V$-$k$ 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. *at most one $\sum^2$-read$_V$-$(k-1)$ formula, which is the derivative with respect to $x$ of some subformula of $F$.*

*Proof.* Assume without loss of generality that $V$ only contains variables on which $F$ depends, and that the children of multiplication gates are variable disjoint with respect to $V$.

If none of the variables in $V$ occur $k$ times in $F$, any choice of variable $x \in V$ does the job. So, let us assume that is at least one variable in $V$ occurs $k$ times. We use Algorithm 1 to select the variable $x$, where we assume without loss of generality that $F$ has fanin 2.

The algorithm recurses through the structure of $F$, maintaining the following invariant: The current gate $g$ being visited contains below it $k$ occurrences of some variable in $V$. Setting $g$ to be the output gate of $F$ satisfies this invariant initially.

If $g$ is a multiplication gate (steps (1)-(3)), recurse to the child of $g$ that depends on more than $\frac{|V|}{2}$ of the variables in $V$ and contains $k$ occurrences of some variable in $V$. If no such child exists, return a variable from $V$ that occurs $k$ times in $g$. Such a variable must exist by the invariant.

If $g$ is an addition gate (steps (4)-(6)), and at least one of its children, $g_i$, has a variable in $V$ that occurs $k$ times in $g_i$, recurse to $g_i$. Otherwise, both children of $g$ are read$_V$-$(k-1)$ formulae. Select a variable $x \in V$ that occurs $k$ times in $g$ ending the recursion.

In the partial derivative $\partial_x F$, all unvisited addition branches along the path from the output gate of $F$ to the final $g$ have been eliminated. Also, all unvisited multiplication branches along the path become factors of $\partial_x F$ together with $\partial_x g$. More formally, $\partial_x F = (\partial_x g) \cdot \prod_i F_i$, where the $F_i$

**Algorithm 1** SPLIT$(g, k, V)$ - A read$_V$-$k$ fragmenting algorithm

```
Input: $k \geq 2, n \in \mathbb{N}, \emptyset \subsetneq V \subseteq [n]$ and $g$ is an $n$-variate, multilinear read$_V$-$k$ formula that depends
    on all variables in $V$. There exists a variable in $V$ that occurs $k$ times in $g$.
Output: $x \in V$, such that $\partial_x g$ meets the conditions of Lemma 3.2
case $g = \alpha(g_1 \cdot g_2) + \beta$
    if $\exists i \in \{1,2\}, x \in V$ where $x$ occurs $k$ times in $g_i$ and $|\mathrm{var}(g_i) \cap V| > \frac{|V|}{2}$ then
        return SPLIT$(g_i, k, V)$
case $g = \alpha(g_1 + g_2) + \beta$
    if $\exists i \in \{1,2\}, x \in V$ where $x$ occurs $k$ times in $g_i$ then
        return SPLIT$(g_i, k, V)$
{Otherwise}
return $x \in V$ that occurs $k$ times in $g$.
```

are the unvisited multiplication branches. The $F_i$ are read$_V$-$k$ formulae that depend on at most $\frac{|V|}{2}$ variables from $V$, because the largest branch is always taken at multiplication gates. When the process stops at a multiplication gate, no large child contains all occurrences of some variable in $V$. In this case, there is at most one factor of $\partial_x g$ that depends on more than $\frac{|V|}{2}$ variables from $V$. If this factor exists it is a read$_V$-$(k-1)$ formula (and thus also a $\sum^2$-read$_V$-$(k-1)$ formula). The remaining factors are read$_V$-$k$ formulae depending on at most $\frac{|V|}{2}$ variables from $V$. When the process stops at an addition gate, $\partial_x g$ is a $\sum^2$-read$_V$-$(k-1)$ formula that may depend on many variables from $V$. In either case, the resulting $\partial_x F$ meets the requirements of the lemma. Note that since we assumed $F$ depends on all variables in $V$, $\partial_x F \not\equiv 0$. ■

Lemma 3.1 can be viewed as a simplified version of Lemma 3.2 for the case $k = 1$.

### 3.3. Fragmenting Multilinear Sparse-Substituted Formulae

In this subsection we extend our fragmenting arguments to work for sparse-substituted formulae.

First consider a multilinear sparse-substituted read-once formula $F$. The idea is to apply the argument from Lemma 3.1 and the chain rule to locate a variable $x$ such that $\partial_x F$ is *almost* fragmented. By this we mean that each of the factors of $\partial_x F$ depends on at most half of the variables except the factor that was originally a sparse polynomial that depends on $x$. The sparse polynomial, say $f$, may depend on too many variables. In that case we perform further operations so that $f$ factors into small pieces. Through a sequence of partial derivatives and zero-substitutions we eliminate all but one term in $f$. This implies that the sparse polynomial and hence the overall resulting formula $F'$ is $\frac{1}{2}$-split. To perform the additional step, observe that for any variable $x$, either at most half of the terms in $f$ depend on $x$ or at most half do not. In the former case, taking the partial derivative with respect to $x$ eliminates at least half of the terms; setting $x$ to 0 has the same effect in the latter case. Repeating this process a number of times logarithmic in the maximum number of terms eliminates all but one of the terms, resulting in a trivially split formula.

This is the intuition behind the sparse-substituted extension of Lemma 3.1 and corresponds to the first part of the next lemma. The second part is the sparse-substituted extension of Lemma 3.2 and follows from that lemma by a simple observation.

**Lemma 3.3 (Fragmentation Lemma).** *Let $\emptyset \subsetneq V \subseteq [n]$, $k \geq 1$, and let $F$ be an $n$-variable multilinear sparse-substituted read$_V$-$k$ formula that depends on at least one variable in $V$. Let $t$ denote the maximum number of terms in each substituted polynomial.*

1. *If $k = 1$ there exist disjoint sets of variables $P, Z \subseteq V$ with $|P \cup Z| \leq (\log(t) + 1)$ such that $\partial_P F|_{Z \leftarrow 0}$ is nonzero and is a product of factors that each depend on at most $\frac{|V|}{2}$ variables from $V$ (and possibly more variables outside of $V$). Moreover, the factors are subformulae of $F$ and at most one formula of the form $\partial_P f|_{Z \leftarrow 0}$ where $f$ is one of the sparse substitutions.*
2. *Otherwise, there exists a variable $x \in V$ such that $\partial_x F$ is nonzero and is the product of*
   - a) *subformulae of $F$ each depending on at most $\frac{|V|}{2}$ variables from $V$ (and possibly more variables outside of $V$), and*
   - b) *at most one multilinear sparse-substituted $\sum^2$-read$_V$-$(k-1)$ formula, which is the derivative with respect to $x$ of some subformula of $F$.*

*Proof. Part 1.* Assume without loss of generality that $F$ has fanin 2, depends on all variables in $V$, and that the children of multiplication gates are variable disjoint with respect to $V$.

Use Lemma 3.1 to select a variable, $x$, such that $\partial_x F$ is the product of multilinear sparse-substituted read$_V$-once formulae on at most $\frac{|V|}{2}$ variables from $V$ and possibly a single sparse polynomial on more than $\frac{|V|}{2}$ variables from $V$ (which originally depended on $x$). If the large sparse factor is not present, the lemma is complete with $P = \{x\}$ and $Z = \emptyset$. Therefore, assume otherwise.

Let $f$ be the large sparse polynomial factor of $\partial_x F$. A sparse polynomial can be thought of as a sparse sum of terms over the variables in $V$ where the coefficients are sparse polynomials in $\mathbb{F}[[n]\backslash V]$ (the constant term counts). If the number of terms in $f$ is less than two, $f$ can be represented as a product of multilinear sparse-substituted read$_V$-once formulae, each depending on a single variable from $V$. Also, for each variable in $V$ that $f$ depends on, we can assume that variable is present in at least one term, but not all of them. Otherwise, that variable can be pulled out as a factor multilinear sparse-substituted read$_V$-once formula. Therefore, we can assume that the number of terms in $f$ is at least two and every variable in $V$ that $f$ depends on is present in at least one term but not in every term. Thus, for each variable $y \in \text{var}(f) \cap V$, at least one of $f|_{y \leftarrow 0}$ or $\partial_y f$ has at most half as many terms as $f$. Since $f$ has at most $t$ terms, at most $\log t$ many partial derivatives and zero-substitutions are sufficient to eliminate all but one of the terms in $f$. Therefore, there are choices for disjoint sets $P'$ and $Z$ such that $\partial_{P'} f|_{Z \leftarrow 0}$ becomes a term over $V$ (which is the product of univariate read$_V$-once formulae). Choosing $P \doteq \{x\} \cup P'$ and $Z$, lifts the required property to $\partial_P F|_{Z \leftarrow 0}$. Since $F$ is multilinear, the operations we perform ensure that $\partial_P F|_{Z \leftarrow 0} \not\equiv 0$.

*Part 2.* Here the proof is essentially the same as the proof of Lemma 3.2. Since $k \geq 2$, the argument always halts at an internal gate and never reaches a sparse-substituted input. Only the number of occurrences of each variable is relevant to the decisions the argument makes. This implies that the argument does not change when sparse-substituted formulae are considered (and is even independent of the sparsity parameter). Thus, this part of the proof is immediate as a corollary to the proof of Lemma 3.2. ■

Observe that the cost of applying the Fragmentation Lemma to a read-once formula is $\log(t) + 1$ partial derivatives and zero-substitutions, whereas applying it to a formula that is not read-once requires only a single partial derivative (though the promised result is weaker in this case).

### 3.4. Shattering Multilinear Formulae

The previous subsections establish a method for fragmenting multilinear sparse-substituted read-$k$ formulae. We now apply the Fragmentation Lemma to shatter multilinear sparse-substituted $\sum^m$-read-$k$ formulae. Recall that shattering is the act of simultaneously splitting all branches of a $\sum^m$-read-$k$ formula. When $k = 1$, that is, in the case of multilinear sparse-substituted $\sum^m$-read-once formulae, applying the Fragmentation Lemma greedily to a factor that depends on the largest number of variables suffices to shatter a multilinear sparse-substituted $\sum^m$-read-once formula to an arbitrary level. To obtain an $\alpha$-split formula in the end, we need $O(m^{\frac{(\log(t)+1)}{\alpha}})$ partial derivatives and zero-substitutions.

In the case of arbitrary read-value $k > 1$ the Fragmentation Lemma is not immediately sufficient for the task. As in the read-once case, we can apply the lemma greedily to a largest factor of a read-$k$ branch to $\alpha$-split the branch within at most $\frac{2}{\alpha}$ applications. However, this is assuming that case 2b of the Fragmentation Lemma never occurs where the $\sum^2$-read-$(k-1)$ factor depends on more than half (possibly all) of the variables. When this case occurs the fragmentation process fails to split the formula into pieces each depending on few variables. To resolve the issue, we leverage the fact that the blocking factor is both large and a $\sum^2$-read-$(k-1)$ formula.

Consider a specific read-$k$ formula $F$ on $n$ variables. Apply the Fragmentation Lemma to $F$. Suppose that case 2b 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 $V \doteq \mathrm{var}(H_1 + H_2)$. $F'$ is explicitly a $\sum^2$-read$_V$-$(k-1)$ formula and a read$_V$-$k$ formula. However, $F'$ is almost certainly not a read-$k$ 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 $k$. Note that prior to this restriction the upper bound on this sum is $2(k-1)$. 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-once formulae with respect to some subset $V$ of $[n]$. In the latter case we are effectively in the situation we first described with $k = 1$, and all subsequent splittings will succeed. In either case we obtain a formula that is shattered with respect to a subset $V$ that is at most a factor $k^{O(k)}$ smaller than $n$.

In summary, the Shattering Lemma splits multilinear sparse-substituted $\sum^m$-read-$k$ formulae to arbitrary degree, albeit with some restriction of the variable set and an increase in top fanin. Moreover, each of the branches in the shattered formula are present in the original input formula, either as such or after taking some partial derivatives and zero-substitutions. This technical property follows from the properties of the Fragmentation Lemma and will be needed in the eventual application.

**Lemma 3.4 (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 sparse-substituted read-$k_i$ formula. Let $t$ denote the maximum number of terms in each substituted polynomial. There exist disjoint subsets $P, Z, V \subseteq [n]$ such that $\partial_P F|_{Z\leftarrow 0}$ 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$_V$,*
- $|P \cup Z| \leq (k-m+1) \cdot \frac{4k}{\alpha(k+1)} \cdot (\log(t)+1)$, *and*
- $|V| \geq \left(\frac{\alpha(k+1)}{8k}\right)^{k-m} \cdot n - \frac{8k}{\alpha(k+1)} \cdot (\log(t)+1)$.

*Moreover, the factors of each of the $F'_i$'s are of the form $\partial_{\tilde{P}} f|_{Z\leftarrow 0}$ where $f$ is some subformula of some $F_j$ and $\tilde{P} \subseteq P$.*

*Proof.* We iteratively construct disjoint subsets $P, Z, V \subseteq [n]$, maintaining the invariant that $\partial_P F|_{Z\leftarrow 0}$ can be written as $F' \doteq \sum_{i=1}^{m'} F'_i$ where (1) each $F'_i$ is a read$_V$-$k'_i$ formula, (2) $m' \leq k$, (3) $\sum_{i=1}^{m'} k'_i \doteq k' \leq k$, and (4) each $F'_i$ is the product of factors of the form $\partial_{\tilde{P}} f|_{Z\leftarrow 0}$ where $f$ is some subformula of some $F_j$ and $\tilde{P} \subseteq P$. Setting $P \leftarrow \emptyset, Z \leftarrow \emptyset, V \leftarrow [n]$ and $F' \leftarrow F$ realizes the invariant initially. The fact that $m' \leq k$ follows because each $F_i$ is non-constant.

The goal of our algorithm is to $\alpha(m'+1)$-split$_V$ the formula $F'$. Each iteration (but the last) consists of two phases: a splitting phase, and a rewriting phase. In the splitting phase we attempt to split $F'$ by greedily applying the Fragmentation Lemma (Lemma 3.3) on each of the branches $F'_i$. The splitting phase may get stuck because of a $\sum^2$-read$_V$-$(k'_i-1)$ subformula that blocks further splitting. If not and the resulting $F'$ is sufficiently split, the algorithm halts. Otherwise, the algorithm enters the rewriting phase where it expands the subformula that blocked the Fragmentation Lemma and reasserts the invariant, after which the next iteration starts. A potential argument shows that the number of iterations until a successful splitting phase is bounded by $k-m$. We first describe the splitting and rewriting phases in more detail, then argue termination and analyze what bounds we obtain for the sizes of the sets $P$, $Z$, and $V$.

*Splitting.* Assume that $F'$ is not $\frac{\alpha(m'+1)}{2}$-split$_V$, otherwise halt. Let $F'_{ij}$ be a subformula of $F'$ that depends on the most variables in $V$ out of all the factors of the $F'_i$'s. Apply the Fragmentation Lemma (Lemma 3.3) with respect to the set $V \cap \mathrm{var}(F'_{ij})$ to produce sets $P', Z' \subseteq [n]$. By the Fragmentation Lemma exactly one of the following holds: (i) the factors of $\partial_{P'} F'_{ij}|_{Z'\leftarrow 0}$ depend on at most $\frac{|V \cap \mathrm{var}(F'_{ij})|}{2}$ variables in $V$, or (ii) $\partial_{P'} F'_{ij}|_{Z'\leftarrow 0}$ has one multilinear sparse-substituted $\sum^2$-read$_V$-$(k'_i-1)$ factor which depends on more than $\frac{|V \cap \mathrm{var}(F'_{ij})|}{2}$ variables in $V$.

Repeatedly perform this greedy application, adding elements to the sets $P'$ and $Z'$ until either case (ii) above occurs or $\partial_{P'} F'|_{Z'\leftarrow 0}$ is $\frac{\alpha(m'+1)}{2}$-split$_V$. In the former case we start a rewriting phase and modify $\partial_{P'} F'|_{Z'\leftarrow 0}$ before we re-attempt to split. In the latter case our goal has been achieved provided that $|P' \cup Z'| \leq \frac{|V|}{2}$: We can add $P'$ to the set $P$ we already had, similarly add $Z'$ to

$Z$, and replace $V$ by $V' \doteq V \setminus (P' \cup Z')$. The assumption that $|P' \cup Z'| \leq \frac{|V|}{2}$ guarantees that $|V'| \geq \frac{|V|}{2}$. Since $\partial_P F|_{Z \leftarrow 0}$ (which equals $\partial_{P'} F'|_{Z' \leftarrow 0}$) is $\frac{\alpha(m'+1)}{2}$-split$_V$, the latter inequality implies that the formula is $\alpha(m'+1)$-split$_{V'}$. If the assumption that $|P' \cup Z'| \leq \frac{|V|}{2}$ does not hold, then outputting $V' = \emptyset$ will meet the size bound for that set and trivially make the formula $\partial_P F|_{Z \leftarrow 0}$ $\alpha(m'+1)$-split$_{V'}$.

The splitting phase maintains the invariant. Regarding part (4) of the invariant, observe that the factors produced by the Fragmentation Lemma are subformulae of the input to the Fragmentation Lemma (for which the invariant initially held).

*Rewriting.* We now describe the rewriting phase. Let $V$ refer to the situation at the start of the preceding splitting phase. Let $F'_{ij}$ be the subformula the splitting phase blocked on, and let $H_1$ and $H_2$ denote the two branches of the multilinear sparse-substituted $\sum^2$-read$_V$-$(k'_i - 1)$ subformula of $\partial_x F'_i$ that caused the blocking case 2b of the Fragmentation Lemma to happen. We have that $\partial_{P'} F'_{ij}|_{Z' \leftarrow 0} = H \cdot (H_1 + H_2)$, where $H$ is some read$_V$-$k'_i$ formula that is independent of the variables in $V \cap \text{var}(H_1 + H_2)$. Let $V' \doteq V \cap \text{var}(H_1 + H_2)$. Partition $V'$ into sets $\{V'_0, ..., V'_{k'_i - 1}\}$ based on the exact number of occurrences of each variable in $H_1$. Let $V''$ be any set from this partitioning excluding the set $V'_0$ (we will restrict the choice of $V''$ later). This implies that $H_1$ is read$_{V''}$-$k'_{i1}$ and $H_2$ is read$_{V''}$-$k'_{i2}$ for some integers $k'_{i1}$ and $k'_{i2}$ such that $k'_{i1}, k'_{i2} < k'_i$ and $k'_{i1} + k'_{i2} \leq k'_i$.

Rewrite $\partial_{P'} F'|_{Z' \leftarrow 0}$ as a top fanin $m' + 1$ formula by distributing multiplication over addition in the term $\partial_{P'} F'_i|_{Z' \leftarrow 0}$:

$$\partial_{P'} F'|_{Z' \leftarrow 0} \equiv (H \cdot H_1) + (H \cdot H_2) + \sum_{j \neq i} \partial_{P'} F'_j|_{Z' \leftarrow 0}. \tag{1}$$

Observe that $\sum_{j \neq i} \partial_{P'} F'_j|_{Z' \leftarrow 0}$ is a read$_{V''}$-$(\sum_{j \neq i} k'_j)$ formula as partial derivatives and substitutions do not increase the read-value, and $V'' \subseteq V$. The term $(H \cdot H_1) + (H \cdot H_2)$ may not be a read$_V$-$k'_i$ formula, but it must be a read$_{V'}$-$k'_i$ formula. It is explicitly the sum of a read$_{V''}$-$k'_{i1}$ formula and a read$_{V''}$-$k'_{i2}$ formula for some $k'_{i1}, k'_{i2} < k'_i$ with $k'_{i1} + k'_{i2} \leq k'_i$. The representation of $\partial_{P'} F'|_{Z' \leftarrow 0}$ in Equation (1) is therefore a read$_{V''}$-$k'$ formula with top fanin $m' + 1$.

Set $F'$ to be this representation of $\partial_{P'} F'|_{Z' \leftarrow 0}$. Merge branches that have become constant into non-constant branches. This maintains the invariant that $m' \leq k' \leq k$. Setting $V \leftarrow V''$ makes $F'$ a top fanin $m'$ read$_V$-$k'$ formula. As for part (4) of the invariant, note that the subformula $F'_{ij}$ which blocked the Fragmentation Lemma originally satisfied it during the splitting phase. This means that with respect to the additional partial derivatives and zero-substitutions performed for the attempted split, $H_1$ and $H_2$ as well as $H$ satisfy the invariant as new factors of the branches $F'_i$. Thus, the new $F'$ satisfies the full invariant. This completes the rewriting phase and one full iteration of the algorithm.

*Correctness.* We repeat the sequence of splitting and rewriting phases until a splitting phase runs till completion. In that case the algorithm produces disjoint sets $P, Z, V \subseteq [n]$ such that $\partial_P F|_{Z \leftarrow 0}$ can be written as a $\alpha(m'+1)$-split$_V$ formula with top fanin $m' \leq k$.

Apart from the size bounds on the sets $P$, $Z$, and $V$, all that remains to establish correctness is termination. To argue the latter we use the following potential argument. Consider the sum $\sum_{i=1}^{m'} k'_i$ and view it as $m'$ blocks of integer size, where $k'_i$ is the size of the $i$th block. Over the course of the algorithm blocks can only stay the same, shrink, or be split in a nontrivial way. The latter is what happens in a rewriting phase. As soon as all blocks are of size at most 1, the splitting phase is guaranteed to run successfully because case 2b of the Fragmentation Lemma cannot occur for read-once formulae, and the algorithm terminates. As we start out with $m$ nontrivial blocks and a value of $k$ for the sum, there can be no more than $k-m$ nontrivial splits. Therefore, there are no more than $k-m$ rewriting phases and $k-m+1$ splitting phases.

*Analysis.* We now bound the size of $P \cup Z$. We first analyze how many times the Fragmentation Lemma is applied in each splitting phase. The goal is to $\frac{\alpha(m'+1)}{2}$-split$_V$ each of the $m'$ branches. To $\frac{\alpha(m'+1)}{2}$-split$_V$ one branch, $\frac{4}{\alpha(m'+1)}$ applications of the Fragmentation Lemma are sufficient, since each application reduces the intersection of the factors with $V$ to at most half the original amount. Since the invariant maintains $m' \le k$ and $\alpha$ is non-increasing, we can upper bound the number of applications of the Fragmentation Lemma during an arbitrary iteration by $\frac{4m'}{\alpha(m'+1)} \le \frac{4k}{\alpha(k+1)}$. Each single application of the Fragmentation Lemma adds at most $(\log(t)+1)$ variables to $P'$ and $Z'$. Since there are at most $k-m+1$ splitting phases, across all iterations at most $(k-m+1)(\log(t)+1)\frac{4k}{\alpha(k+1)}$ variables are added to $P \cup Z$.

We finish by lower bounding the size of $V$. Consider the change in $|V|$ over one combined splitting/rewriting iteration. We have that $|V'| \ge \frac{\alpha(m'+1)}{4}|V|$, because $F'$ was not $\frac{\alpha(m'+1)}{2}$-split$_V$ before attempting to split $F'_{ij}$ (so the largest number of variables in $V$ that a factor depends on is at least $\frac{\alpha(m'+1)}{2} \cdot |V|$), $F'_{ij}$ was chosen for its maximal dependence on variables from $V$, and $|\mathrm{var}(H_1+H_2) \cap V| \ge |\mathrm{var}(F'_{ij}) \cap V|/2$. If we pick $V''$ to be the largest set from the partitioning $\{V'_0, V'_1, ..., V'_{k'_i-1}\}$ excluding $V'_0$, and we assume without loss of generality that $|\mathrm{var}(H_1)| \ge |\mathrm{var}(H_2)|$, we have that $|V''| \ge \frac{1}{2(k'_i-1)}|V'|$. Combining these inequalities and using the facts that $\alpha$ is non-increasing and $k \ge k'_i, m'$ gives:

$$|V''| \ge \frac{1}{2(k'_i-1)}|V'| \ge \frac{\alpha(m'+1)}{8(k'_i-1)}|V| \ge \frac{\alpha(k+1)}{8k}|V|.$$

This means that $|V|$ decreases by a factor of at most $\frac{8k}{\alpha(k+1)}$ in each combined splitting/rewriting iteration. At the end of the final splitting phase $|V'| \ge |V| - 2|P' \cup Z'|$ because $V'$ is set to the empty set when $|P' \cup Z'| \ge \frac{|V|}{2}$. Recall that $|P' \cup Z'| \le \frac{4k}{\alpha(k+1)}(\log(t)+1)$. Since there are at most $k-m$ combined splitting/rewriting iterations, this gives the following lower bound at the end:

$$|V| \ge \left(\frac{\alpha(k+1)}{8k}\right)^{k-m} \cdot n - \frac{8k}{\alpha(k+1)} \cdot (\log(t)+1). \qquad \blacksquare$$

## 4. Reducing Testing Multilinear Read-$(k+1)$ Formulae to Testing Multilinear $\sum^2$-Read-$k$ Formulae

In this section we describe two methods of reducing identity testing multilinear sparse-substituted read-$(k+1)$ formulae to identity testing multilinear sparse-substituted $\sum^2$-read-$k$ formulae. The first reduction is non-blackbox and is entirely self-contained. The second reduction is blackbox and makes use of the Fragmentation Lemma of the preceding section.

## 4.1. A Non-Blackbox Reduction

The intuition for this reduction is somewhat similar to that for the Fragmentation Lemma. Consider a subformula $g$ of a multilinear sparse-substituted read-$(k+1)$ formula $F$ where $g$ is of the form $g = g_1 + g_2$ and $g_1$ is read-$(k+1)$ but not read-$k$. There must be some variable $x$ that appears $k+1$ times in $g_1$ and nowhere else in $F$. If $g_1$ actually depends on $x$, then $g$ is nonzero. This is irrespective of whether $g_2$ is nonzero, because it is not possible for $g_2$ to cancel out the contribution of $x$ present in $g_1$. In general, if all occurrences of a variable $x$ are contained in an addition branch and that branch depends on $x$, the addition gate must be nonzero. The polynomials computed by gates above $g$ can only be zero if the zero polynomial is multiplied in. Now, consider replacing $g_1$ with a fresh variable. The above reasoning shows that this transformation does not change whether the overall formula is nonzero. If a branch contains all occurrences of $x$ but does not depend on $x$, setting $x$ to 0 does not affect the value computed by the formula. This observation allows us to eliminate variables that are read $k+1$ times, and thereby transform a multilinear sparse-substituted read-$(k+1)$ formula into a multilinear sparse-substituted read-$k$ formula without affecting the (non)zeroness of the formula.

In order to execute the transformation, we need to be able to decide whether the subformula $g_1$ depends on the variable $x$. If we apply the transformation in a bottom-up fashion, by the time we need to make that decision the formula $g_1$ is multilinear sparse-substituted $\sum^2$-read-$k$, and we can use a polynomial identity test for such formulae to check whether $\partial_x g_1 \equiv 0$. This is the idea behind the non-blackbox reduction from identity testing multilinear sparse-substituted read-$(k+1)$ formulae to identity testing multilinear sparse-substituted $\sum^2$-read-$k$ formulae.

**Lemma 4.1 (Read-$(k+1)$ PIT $\leq \sum^2$-Read-$k$ PIT – Non-Blackbox).** *For an integer $k \geq 1$, given a deterministic identity testing algorithm for $n$-variate size-$s$ multilinear sparse-substituted $\sum^2$-read-$k$ formulae that runs in time $T(k,n,s,t)$, where $t$ denotes the maximum number of terms in each substituted polynomial, there is a deterministic algorithm that tests $n$-variate size-$s$ multilinear sparse-substituted read-$(k+1)$ formulae that runs in time $O(kn^2 \cdot T(k,n,s,t) + \mathrm{poly}(k,n,s,t))$.*

*Proof.* Let $F$ be a multilinear sparse-substituted read-$(k+1)$ formula. The goal of the algorithm is to transform the gates of $F$ in a bottom-up fashion into read-$k$ formulae while preserving the (non)zeroness of $F$. The transformation also eliminates variables which gates do not depend on. Because $F$ is multilinear, this second property ensures that multiplication gates are explicitly variable disjoint. Once $F$ is transformed in this way, the identity test is immediate from the assumed identity testing algorithm.

As a first step we argue that the following transformation preserves the (non)zeroness of $F$. Consider a gate $g$ in $F$ that contains all occurrences of some variable $x$ in $F$ and depends on $x$. Define $F_{-g}(\bar{x}, y)$ as the formula where the gate $g$ has been replaced by a new variable $y$ (note, $F_{-g}(\bar{x}, g) = F$). $F_{-g}$ does not depend on $x$ because all occurrences of $x$ are in $g$. Then, because $F_{-g}$ is a formula and $y$ occurs only once, without loss of generality, write:

$$F \equiv F_{-g}(\bar{x}, g) \equiv P(\bar{x}) + Q(\bar{x}) \cdot g,$$

for two polynomials $P$ and $Q$ that do not depend on $x$. If $Q \equiv 0$, then $F$ is independent of $g$, so $F_{-g} \equiv F$. If $Q \not\equiv 0$, then $F$ is nonzero because $g$ depends on $x$, but $P$ does not depend on $x$, so

**Algorithm 2** Transforming a read-$(k+1)$ gate into an "equivalent" read-$k$ gate

**Input:** $k \geq 1$, $F$ is a multilinear sparse-substituted read-$(k+1)$ formula, $g$ is a multilinear sparse-substituted read-$(k+1)$ subformula of $F$ whose children are read-$k$ formulae that depend on all variables that appear within them, and $\mathcal{A}$ is a deterministic identity testing algorithm for multilinear sparse-substituted $\sum^2$-read-$k$ formulae.

**Output:** $g$ is a read-$k$ formula that depends on all variables that appear within it. Except for variables that do not appear in $F_{-g}$, the number of occurrences of a variable in $g$ does not increase. The (non)zeroness of $F$ is unchanged with respect to the transformation of $g$.

```
case $g = \alpha(g_1 + g_2) + \beta$
  for all $x \in \mathrm{var}(g)$
    if $\partial_x(g_1 + g_2) \equiv 0$ {Invoking the subroutine $\mathcal{A}$} then
      Replace $g$ by $g|_{x \leftarrow 0}$
    else if $x$ occurs $k+1$ times in $g$ then
      Replace $g$ by a new variable $y$
case $g = \alpha(g_1 \cdot g_2) + \beta$
  if $g_1 \equiv 0$ OR $g_2 \equiv 0$ {Invoking the subroutine $\mathcal{A}$} then
    Replace $g$ by $\beta$
case $g$ is a sparse-substituted input
  Simplify the list of terms
```

the $x$ component cannot be canceled. By definition, $F \not\equiv 0$ implies $F_{-g} \not\equiv 0$. Therefore we can conclude that for such gates $g$, $F \equiv 0$ iff $F_{-g} \equiv 0$.

Now, process the gates $g$ of $F$ in a bottom-up fashion. Note that the algorithm realizes the following properties: (1) the (non)zeroness of $F$ does not change, (2) $g$ is a multilinear sparse-substituted read-$k$ formula, (3) except for variables that only appear in $g$, the number of occurrences of a variable in $g$ does not increase, and (4) $g$ depends on all variables that appear in it. There are three possible cases for $g$.

1. Case $g = \alpha(g_1 + g_2) + \beta$. We first go over all original variables $x$ that appear in $g$. Let $x$ be such a variable. Since the children $g_1$ and $g_2$ of $g$ are read-$k$, $g$ is a multilinear sparse-substituted $\sum^2$-read-$k$ formula. Compute $\partial_x g$ as a multilinear sparse-substituted $\sum^2$-read-$k$ formula. This is easy because the multiplication gates within $g_1$ and $g_2$ are variable disjoint. Now, test whether $\partial_x g \equiv 0$ using the hypothesized identity testing algorithm. If $\partial_x g \equiv 0$, replace $g$ by $g|_{x \leftarrow 0}$; otherwise, if $x$ occurs $k+1$ times in $g$, replace $g$ by a fresh variable $y$, if not, do nothing.

2. Case $g = \alpha(g_1 \cdot g_2) + \beta$. Because property (4) holds for the children of $g$, and $g$ is multilinear, $g$ is a read-$k$ formula. Check whether $g_1$ or $g_2$ are identically zero, using the identity test for read-$k$ formulae. If either formula is zero, replace $g$ with $\beta$. Otherwise, do nothing.

3. Case $g$ is a sparse-substituted input. In order to realize properties (1-4), all we need to do is to simplify the list of terms by collecting duplicate monomials and dropping them if the coefficient is zero.

For clarity, the algorithm is described in Algorithm 2. Overall, it transforms $F$ from a multilinear sparse-substituted read-$(k+1)$ formula into a multilinear sparse-substituted read-$k$ formula without affecting the (non)zeroness. The $\sum^2$-read-$k$ formula identity test is applied at most $n$ times at each gate to determine the dependence on the original variables. Since $F$ is in standard form, this takes at most $O(kn^2)$ identity tests overall. Adding polynomial-time for computing the standard form, traversing $F$, computing the partial derivatives, and doing the field arithmetic gives the running time claimed. ■

### 4.2. A Blackbox Reduction

Let $F$ be a multilinear sparse-substituted read-$(k+1)$ formula. We construct a generator for $F$ using a generator $\mathcal{G}$ for multilinear sparse-substituted $\sum^2$-read-$k$ formulae. If $F$ is a read-$k$ formula, the assumed generator alone suffices. Otherwise, we apply the Fragmentation Lemma (Lemma 3.3 to show that there is a partial derivative of $F$ that has mostly small factors and, possibly, one factor that is a large multilinear sparse-substituted $\sum^2$-read-$k$ formula. In the former case the factors are small enough to be hit recursively and in the latter case the factor is hit by the assumed generator for multilinear sparse-substituted $\sum^2$-read-$k$ formulae. The properties of the SV-generator $G$ (Proposition 2.6 and Lemma 2.7) imply that if $\mathcal{G}$ is a generator for the partial derivative of a polynomial, then $\mathcal{G} + G_1$ is a generator for the original polynomial.

**Lemma 4.2 (Read-$(k+1)$ PIT $\leq \sum^2$-Read-$k$ PIT – Blackbox).** *For an integer $k \geq 1$, let $\mathcal{G}$ be a generator for $n$-variate multilinear sparse-substituted $\sum^2$-read-$k$ formulae, and let $F$ be a nonzero $n$-variable multilinear sparse-substituted read-$(k+1)$ formula. Then $\mathcal{G} + G_{\log|var(F)|}$ hits $F$.*

*Proof.* First observe that if $F$ is read-$k$, we are immediately done because $F \circ \mathcal{G} \not\equiv 0$ and $\bar{0}$ is in the range of $G$ (by the first item of Proposition 2.6).

The proof goes by induction on $|\text{var}(F)|$. If $|\text{var}(F)| = 0$, the lemma holds trivially as $F$ is constant. If $|\text{var}(F)| = 1$, $F$ is a read-once formula, which is covered by the above observation. For the induction step, by the above observation we can assume that $F$ is read-$(k+1)$ and not read-$k$. Therefore, $F$ meets the conditions to apply the second part of the Fragmentation Lemma (Lemma 3.3). The lemma produces a variable $x \in \text{var}(F)$. The factors of $\partial_x F$ all depend on at most $\frac{|\text{var}(F)|}{2}$ variables and are read-$(k+1)$ formulae, except for at most one which is a $\sum^2$-read-$k$ formula. The induction hypothesis gives that the former factors of $\partial_x F$ are all hit by $\mathcal{G} + G_{\log(|\text{var}(F)|/2)}$. The latter factor (if it occurs) is hit by $\mathcal{G}$. Applying Lemma 2.7 gives that $\mathcal{G} + G_{\log(|\text{var}(F)|/2)} + G_1$ hits $F$. Recalling Proposition 2.6 implies that $\mathcal{G} + G_{\log|\text{var}(F)|}$ hits $F$. ■

## 5. Reducing Testing Multilinear $\sum^2$-Read-$k$ Formulae to Testing Multilinear Read-$k$ Formulae

In this section we present two methods of reducing identity testing multilinear sparse-substituted $\sum^2$-read-$k$ formulae to identity testing multilinear sparse-substituted read-$k$ formulae. Both reductions rely on a common theorem (Theorem 5.4) proved in the next subsection. Informally, that theorem says that for a nonzero multilinear sparse-substituted $\sum^2$-read-$k$ formula $F$ and a shift $\bar{\sigma}$ satisfying some simple conditions, the shifted formula $F(\bar{x} + \bar{\sigma})$ is hit by the SV-generator $G_w$

with $w = k^{O(k)}(\log(t) + 1)$, where $t$ denotes the maximum number of terms in each substituted polynomial.

Note that, since $F$ is a nonzero polynomial, such a theorem is trivially true for a typical shift $\bar{\sigma}$, even with $w = 0$. The interesting part of the theorem is the simplicity of the conditions on $\bar{\sigma}$ that guarantee the hitting property. In particular, the properties needed of $\bar{\sigma}$ allow it to be computed efficiently either by an identity test for multilinear sparse-substituted read-$k$ formulae, or as an element in the range of a hitting set generator for such formulae, where the efficiency of the former is better than the latter.

## 5.1. A Generator for Shifted Formulae

In this subsection we show that the SV-generator hits small sums of specially shifted multilinear sparse-substituted constant-read formulae. Our argument critically relies on the property given in Lemma 2.8 – that the SV-generator hits any class of polynomials that is closed under zero-substitutions and such that no term of high degree belongs to the class. Thus, we first argue that small sums of specially shifted multilinear sparse-substituted constant-read formulae cannot compute a term of high degree. The latter is a formalization and generalization of Lemma 1.7 from the introduction.

In order to prove that key lemma, we first establish a similar lemma for *split* multilinear sparse-substituted 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 sparse-substituted formula on $n$ variables. By applying the rank bound for split formulae (Lemma 2.11) we can argue that if none of the $F_i$'s are 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 $M_n$ immediately give simplicity. Minimality essentially comes for free because we are working in the blackbox setting. The splitting required by the rank bound immediately follows from the splitting of $F$. Formalizing this idea yields the following lemma.

**Lemma 5.1.** *Let $F = \sum_{i=1}^{m} F_i$ be a multilinear sparse-substituted $\alpha(m+1)$-split formula on $n \geq 1$ variables, where $\alpha \doteq \frac{1}{R}$ and $R$ is the function given by Lemma 2.11. If no $F_i$ is divisible by any variable, then $F \not\equiv a \cdot \prod_{i=1}^{n} x_i$ for any nonzero constant $a$.*

Note that for a non-constant formula $F$ on $n$ variables to be $\alpha(m+1)$-split, $n$ needs to be at least $1/\alpha(m+1)$.

*Proof.* Suppose for the sake of contradiction that $F \equiv a \cdot M_n$ for some nonzero constant $a$, where $M_n \doteq \prod_{i=1}^{n} x_i$.

If there is some subsum of the branches of $F$ that equals 0, eliminate all those branches. Not all branches of $F$ may be eliminated in this way as this contradicts $a \cdot M_n \not\equiv 0$. Let $0 < m' \leq m$ be the remaining number of branches, and let $F'$ denote the remaining branches. The formula $F' - a \cdot M_n$ is minimal and has top fanin $m' + 1$.

Now, suppose that there is some non-constant polynomial $D$ that divides every remaining $F_i$. Since $F' \equiv a \cdot M_n$, then $D$ also divides $M_n$. Because $D$ is non-constant, some variable $x$ divides $D$

and hence divides each remaining $F_i$. This contradicts the hypothesized non-divisibility property of the $F_i$. Therefore $F' - a \cdot M_n$ is simple as a formula with top fanin $m' + 1$.

The previous two paragraphs establish that the top fanin $m'+1$ formula $F' - a \cdot M_n$ is simple and minimal. Further, for every variable, there is some branch that depends on it – the $M_n$ branch does. Observe that the $M_n$ branch is trivially $\alpha(m'+1)$-split and every other branch is also $\alpha(m'+1)$-split as $m' \leq m$ and $\alpha \doteq \frac{1}{R}$ is decreasing. The rank bound for split formulae (Lemma 2.11) then implies that $F' - a \cdot M_n \not\equiv 0$, and thus that $F \not\equiv a \cdot M_n$. This contradicts the initial assumption and concludes the proof. ■

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 $F_i(\bar{x} + \bar{\sigma})$. This reasoning is formalized in the following corollary.

**Corollary 5.2.** *Let $F = \sum_{i=1}^{m} F_i$ be a multilinear sparse-substituted $\alpha(m+1)$-split formula on $n \geq 1$ variables, where $\alpha \doteq \frac{1}{R}$ and $R$ is the function given by Lemma 2.11. 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 $a$.*

*Proof.* Since the branches of $F$ are $\alpha(m+1)$-split, the branches of $F(\bar{x}+\bar{\sigma})$ are also $\alpha(m+1)$-split. By assumption, $F_i(\bar{\sigma}) \neq 0$. Therefore, for each branch $i \in [m]$ and variable $x \in [n]$, $F_i(\bar{x} + \bar{\sigma})|_{x \leftarrow 0} \not\equiv 0$. This implies that no variables divide any branch $F_i(\bar{x} + \bar{\sigma})$. With this property established, apply Lemma 5.1 on $F(\bar{x} + \bar{\sigma})$ to conclude the proof. ■

We now show how to lift Corollary 5.2 from split multilinear sparse-substituted formulae to sums of multilinear sparse-substituted 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 $F' = \partial_P F|_{Z \leftarrow 0}$ using the Shattering Lemma (Lemma 3.4), and apply the same operations that shatter $F$ to $M_n$. Observe that zero-substitutions are shifted into substitutions by $\bar{\sigma}$, and that $\partial_P M_n|_{Z \leftarrow (-\bar{\sigma})}$ is a nonzero term of degree $n - |P \cup Z|$ 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 Corollary 5.2 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 $\partial_{\tilde{P}} f|_{Z \leftarrow 0}$ 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 $M_n$. This step is in the same spirit as the argument in the proof of the rank bound for split formulae (Lemma 2.11), namely that a typical assignment suffices.

With these ideas in mind, the key lemma is as follows.

**Lemma 5.3 (Key Lemma).** *Let $F = \sum_{i=1}^{m} F_i$, where each $F_i$ is a non-constant multilinear sparse-substituted read-$k_i$ formula. If $\bar{\sigma}$ is a common nonzero of the nonzero formulae of the form $\partial_P f|_{Z\leftarrow 0}$ where $f$ is a subformula of the $F_i$'s and $|P \cup Z| \leq b \doteq (k-m+1)\cdot 4k\cdot R(k+1)\cdot(\log(t)+1)$, then*

$$F(\bar{x}+\bar{\sigma}) \notin \mathcal{T}_n,$$

*for $n > w \doteq (8k\cdot R(k+1))^{k-m+1}(\log(t)+1)$, where $k \doteq \sum_{i=1}^{m} k_i$, $t$ denotes the maximum number of terms in each substituted polynomial, and $R$ is the function given by Lemma 2.11.*

*Proof.* Assume the contrary, without loss of generality, that $F(\bar{x}+\bar{\sigma}) \equiv a\cdot M_n$ for some nonzero constant $a$ and $n > w$.

We first argue that, without loss of generality, $\mathrm{var}(F_i) \subseteq [n]$ for all $i \in [m]$. Suppose that some $F_i$ depends on variables outside $[n]$. Let $W \doteq (\cup_{i=1}^{m}\mathrm{var}(F_i))\backslash[n]$. Replace $F$ with $F|_{W\leftarrow\bar{\sigma}}$. Then $M_n|_{W\leftarrow\bar{\sigma}} = M_n$. $\bar{\sigma}$ remains a common nonzero assignment of the stated type of formulae, now with $F_i$ replaced by $F_i|_{W\leftarrow\bar{\sigma}}$. The substitution may make some branches $F_i$ constant. If this is the case combine those branches into some non-constant branch (if no non-constant branches exist we have already derived a contradiction). Since all branches are non-constant, the quantity $k-m$ has not increased.

Define $\alpha \doteq \frac{1}{R}$. Shatter $F$ using Lemma 3.4. This produces the sets $P, Z$ and $V$. Let $F' \doteq \partial_P F|_{Z\leftarrow 0}$. By the Shattering Lemma $F' = \sum_{i=1}^{m'} F'_i$ is a multilinear sparse-substituted formula that has top fanin $m' \leq k$, is $\alpha(m'+1)$-split$_V$, and each $F'_i$ is a product of factors of formulae of the form $\partial_{\tilde{P}} f|_{Z\leftarrow 0}$ where $f$ is a subformula of an $F_i$ and $\tilde{P} \subseteq P$. Assume without loss of generality that each $F'_i$ is nonzero. By the lemma, $|P\cup Z| \leq b$. By hypothesis, the subformulae of the above form do not vanish at $\bar{\sigma}$. These properties imply that $F'_i(\bar{\sigma}) \neq 0$ for each $i \in [m']$.

There is an assignment to the variables in $[n]\backslash V$ that: (1) preserves $\bar{\sigma}$ as a nonzero of the $F'_i$'s on the remaining variables $V$, and (2) differs in every component from $\bar{\sigma}$. In fact, a typical assignment suffices. To see this, consider the polynomial:

$$\Phi \doteq \left(\prod_{i=1}^{m'} F'_i|_{V\leftarrow\bar{\sigma}}\right)\cdot \prod_{j\in([n]\backslash V)} (x_j - \sigma_j).$$

The polynomial $\Phi$ is nonzero because the $F'_i$'s do not vanish at $\bar{\sigma}$. Thus, a nonzero assignment for $\Phi$ satisfies the requirements above. Pick $\bar{\beta}$ to be any such assignment.

Let $F'' \doteq F'|_{([n]\backslash V)\leftarrow\bar{\beta}}$, where the $F''_i$ are defined similarly. By the first property of $\bar{\beta}$, $F''_i(\bar{\sigma}) \neq 0$. By the second property of $\bar{\beta}$, $M_n|_{([n]\backslash V)\leftarrow(\bar{\beta}-\bar{\sigma})}$ is a nonzero term over the variables $V$. Then using the initial assumption write

$$F''(\bar{x}+\bar{\sigma}) \equiv F'(\bar{x}+\bar{\sigma})|_{([n]\backslash V)\leftarrow(\bar{\beta}-\bar{\sigma})} \equiv a\cdot M_n|_{([n]\backslash V)\leftarrow(\bar{\beta}-\bar{\sigma})} \equiv a'\cdot M_V,$$

for some nonzero constant $a'$. Now, $F'' \in \mathbb{F}[V]$ is a multilinear sparse-substituted top $\alpha(m'+1)$-split$_V$ formula with top fanin $m'$, where no branch vanishes at $\bar{\sigma}$. Thus, we obtain a contradiction with Corollary 5.2 as long as $|V| > 0$. By the bound on $|V|$ given in the Shattering Lemma and then condition that $n > w$, the latter is the case for $w \geq (8k\cdot R(k+1))^{k-m+1}(\log(t)+1)$. ■

In order to prove a usable theorem for our applications, we need to massage Lemma 5.3 to apply Lemma 2.8. Let $F$ be a formula and $\bar{\sigma}$ be a shift as in the statement of Lemma 5.3. The property

$F(\bar{x}+\bar{\sigma}) \notin \mathcal{T}_n$ holds for arbitrary zero substitutions of $F(\bar{x}+\bar{\sigma})$ as well, because if the required properties of $\bar{\sigma}$ hold for the formula itself then they also hold for all zero-substitutions. The set of polynomials corresponding to all zero-substituted versions of $F(\bar{x}+\bar{\sigma})$ serves as the set $\mathcal{P}$ in the application of Lemma 2.8. This, in turn, implies that $G_w$ hits $F(\bar{x}+\bar{\sigma})$, since it is a member of this set of polynomials, where the seed length $w$ is bounded as in Lemma 5.3.

**Theorem 5.4.** *Let $F = \sum_{i=1}^{m} F_i$, where each $F_i$ is a non-constant multilinear sparse-substituted read-$k_i$ formula. If $\bar{\sigma}$ is a common nonzero of the nonzero formulae of the form $\partial_P f|_{Z\leftarrow 0}$ where $f$ is a subformula of the $F_i$'s and $|P \cup Z| \le b \doteq (k-m+1)\cdot 4k \cdot R(k+1)\cdot(\log(t)+1)$, then*

$$F \not\equiv 0 \Rightarrow F(G_w+\bar{\sigma}) \not\equiv 0,$$

*for $w > (8k \cdot R(k+1))^{k-m+1}(\log(t)+1)$, where $k \doteq \sum_{i=1}^{m} k_i$, $t$ denotes the maximum number of terms in each substituted polynomial, and $R$ is the function given by Lemma 2.11.*

*Proof.* Define the class of formulae

$$\mathcal{F} \doteq \{(F|_{S\leftarrow\bar{\sigma}}) \mid S \subseteq [n]\}.$$

Without loss of generality each non-constant $f \in \mathcal{F}$ only has top level branches that are non-constant, since constant branches can be collected into some non-constant branch without changing the read of that branch. Observe that for each $f \in \mathcal{F}$, the non-vanishing property with respect to $\bar{\sigma}$ is maintained because $\bar{\sigma}$ is being used as the partial assignment. Lemma 5.3 then implies that $f(\bar{x}+\bar{\sigma}) \notin \mathcal{T}_n$ for $n > w$. Shifting all subformulae in $\mathcal{F}$ by $\bar{\sigma}$ gives a class $\mathcal{P}$ of multilinear polynomials that is closed under zero-substitutions and does not intersect $\mathcal{T}_n$ for $n > w$. Lemma 2.8 then says that $G_w$ hits $\mathcal{P}$, and $F(\bar{x}+\bar{\sigma})$ in particular. ■

## 5.2. A Non-Blackbox Reduction

We now give a non-blackbox reduction using Theorem 5.4. The first step of the reduction is to compute an appropriate shift $\bar{\sigma}$ using an identity testing algorithm for multilinear sparse-substituted read-$k$ formulae. A technical complication is to ensure that the formula has gates that are explicitly multilinear, so that partial derivatives can be computed efficiently. This can also be done using an identity test for multilinear sparse-substituted read-$k$ formulae. Once we have $\bar{\sigma}$, we simply evaluate $F(G_w+\bar{\sigma})$ on sufficiently many points and see whether we obtain a nonzero value.

**Lemma 5.5 ($\sum^m$-Read-$k$ PIT $\le$ Read-$k$ PIT – Non-Blackbox).** *For any integer $k \ge 1$, given a deterministic identity testing algorithm for $n$-variate size-$s$ multilinear sparse-substituted read-$k$ formulae that runs in time $T(k,n,s,t)$, where $t$ denotes the maximum number of terms in each substituted polynomial, there is a deterministic algorithm that tests $n$-variate size-$s$ multilinear sparse-substituted $\sum^m$-read-$k$ formulae that runs in time*

$$k^2m^2n^{O(b)} \cdot T(k,n,s,t) + n^{O(w_{m,k}\cdot(\log(t)+1))}\,\mathrm{poly}(k,n,s,t),$$

*where $b \doteq ((k-1)m+1)\cdot 4km \cdot R(km+1)\cdot(\log(t)+1)$, $w_{m,k} \doteq (8km \cdot R(km+1))^{(k-1)m+1}$, and $R$ is the function given by Lemma 2.11.*

*Proof.* Let $F \doteq \sum_{i=1}^{m} F_i$, where each $F_i$ is a multilinear sparse-substituted read-$k$ formula. Let $b$ be sufficient to apply Lemma 5.3 with the parameters $k_i = k$, $m$, and $n$.

Process each of the $F_i$ from the bottom up, making the children of multiplication gates variable disjoint. To do this, at each gate $g$ compute the set of variables that $g$ depends on. This can be done using the hypothesized identity test on the first order partial derivatives of $g$ with respect to each variable. These partial derivatives can be efficiently computed as the children have been previously processed to have variable disjoint multiplication gates. Set variables that $g$ does not depend on to 0, though only within the subformula $g$. Note, this does not affect the polynomial computed at each gate of $F_i$, it merely removes extraneous variable occurrences. As we can assume $F$ to be in standard form, each $F_i$ has at most $O(kn)$ gates and this step uses at most $O(kmn^2)$ applications of the identity test.

Let $\mathcal{F}$ be the set of all nonzero formulae of the form $\partial_P f|_{Z \leftarrow 0}$ where $f$ is a subformula of one of the $F_i$'s and $|P \cup Z| \leq b$. Notice that the elements of $\mathcal{F}$ are multilinear sparse-substituted read-$k$ formulae because the latter class of formulae is closed under partial derivatives and substitutions.

There are at most $O(kmn)$ gates in $F$, thus $|\mathcal{F}| = O(kmn^{b+1})$. The formulae in $\mathcal{F}$ can be efficiently enumerated. To see this, observe that for a choice of a gate $g$ in $F_i$, and sets $P$ and $Z$, the formula $\partial_P g|_{Z \leftarrow 0}$ can be computed in time polynomial in the size of $F$, because we preprocessed the multiplication gates of $F$ to be variable disjoint. Further, we can determine efficiently whether each of these formulae is nonzero using the hypothesized identity test for multilinear sparse-substituted read-$k$ formulae.

Define the polynomial, $\Phi \doteq \prod_{f \in \mathcal{F}} f$. Since $\Phi \not\equiv 0$, there is a point in a finite extension $\mathbb{E}^n \supseteq \mathbb{F}^n$ that witnesses the nonzeroness of $\Phi$. We can use trial substitution to determine a point, $\bar{\sigma} \in \mathbb{E}^n$ where $\Phi$ is nonzero. $F$ is multilinear and all the formulae in $\mathcal{F}$ are multilinear as well. This means that $\Phi$ has total degree at most $O(kmn^{b+2})$. By the Schwartz-Zippel Lemma we only need to test elements from a subset $W \subseteq \mathbb{E}$ of size at most the degree of $\Phi$ plus one (i.e., a set of size $O(kmn^{b+2})$). For each variable, in turn, determine a value from $W$ that keeps $\Phi$ nonzero. Fix the variable to this value then move on to consider the next variable. This uses $O(kmn^{b+3})$ identity tests on a partially substituted version of $\Phi$. Each of these identity tests uses $O(kmn^{b+1})$ identity tests on the component read-$k$ formulae. In total, our algorithm uses $O(k^2m^2n^{2b+4})$ identity tests on multilinear sparse-substituted read-$k$ formulae to compute $\bar{\sigma}$. This can be completed in $O(k^2m^2n^{2b+4}T(k,n,s,t))$ time, using the assumed identity test.

Using Theorem 5.4 gives that $G_{w_{m,k}(\log t+1)}$ hits $F(\bar{x} + \bar{\sigma})$. Therefore, $F \equiv 0$ iff $F(G_w + \bar{\sigma}) \equiv 0$. By multilinearity, the formula $F$ has degree at most $n$ and, by definition, $G_w$ has degree at most $n$. Applying Proposition 2.4 gives a test for $F(G_w + \bar{\sigma})$ that runs in time $O((n^2)^{2w})$. This completes the identity test. The cost of performing this part of the algorithm is at most $n^{O(w)} \cdot \mathrm{poly}(k, m, s, t)$. Combining this with the preprocessing and the computation of $\bar{\sigma}$ gives the total running time claimed. ∎

## 5.3. A Blackbox Reduction

We now describe a blackbox version of Lemma 5.5. The overall approach is the same, though the details are somewhat simpler. With Theorem 5.4 in hand, all that remains is to leverage a generator $\mathcal{G}$ for multilinear sparse-substituted read-$k$ formulae to generate an appropriate shift $\bar{\sigma}$ and then apply the theorem to complete the reduction.

Let $F = \sum_{i=1}^{m} F_i$ be a multilinear sparse-substituted $\sum^{m}$-read-$k$ formula. Let $\mathcal{F}$ be the set of all nonzero formulae of the form $\partial_P f|_{Z \leftarrow 0}$ where $f$ is a subformula of one of the $F_i$'s and $|P \cup Z| \leq b$. $\mathcal{F}$ is composed of nonzero multilinear sparse-substituted read-$k$ formulae. Therefore, $\mathcal{G}$ hits the product $\prod_{f \in \mathcal{F}} f$, and a suitable shift $\bar{\sigma}$ is in the image of $\mathcal{G}$. Applying Theorem 5.4 then gives that $\mathcal{G} + G_w$ is a generator for multilinear sparse-substituted $\sum^{m}$-read-$k$ formulae, where $w$ is bounded as in the theorem.

**Lemma 5.6 ($\sum^{m}$-Read-$k$ PIT $\leq$ Read-$k$ PIT – Blackbox).** *For an integer $k \geq 1$, let $\mathcal{G}$ be a generator for $n$-variate multilinear sparse-substituted read-$k$ formulae. Then $\mathcal{G} + G_{w_{m,k} \cdot (\log(t)+1)}$ is a generator for $n$-variate multilinear sparse-substituted $\sum^{m}$-read-$k$ formulae, where $w_{m,k} \doteq (8km \cdot R(km+1))^{(k-1)m+1}$, $t$ denotes the maximum number of terms in each substituted polynomial, and $R$ is the function given by Lemma 2.11.*

*Proof.* Let $F$ be a multilinear sparse-substituted $\sum^{m}$-read-$k$ formula. Write $F \doteq \sum_{i=1}^{m} F_i$, where each $F_i$ is a multilinear sparse-substituted read-$k$ formula. Let $b \doteq ((k-1)m+1) \cdot 4km \cdot R(km+1) \cdot (\log(t)+1)$ and $w \doteq w_{m,k} \cdot (\log(t)+1)$; in other words, sufficient parameters for applying Lemma 5.3 with $m$ and $k_i = k$.

Let $\mathcal{F}$ be the same set of formulae as in the proof of Lemma 5.5 with respect to the current version of $F$, and consider the polynomial $\Phi \doteq \prod_{f \in \mathcal{F}} f$. Note that $\Phi \not\equiv 0$.

Since $\mathcal{G}$ is a generator for multilinear sparse-substituted read-$k$ formulae and $\Phi$ is the product of multilinear sparse-substituted read-$k$ formula, $\mathcal{G}$ hits $\Phi$. There is a point with components in a finite extension $\mathbb{E} \supseteq \mathbb{F}$ that witnesses the nonzeroness of $\Phi \circ \mathcal{G}$. Let $\bar{\beta}$ be such a point and define $\bar{\sigma} \doteq \mathcal{G}(\bar{\beta})$. Thus $\Phi(\bar{\sigma}) \neq 0$. This implies that all nonzero formulae of the form $\partial_P f|_{Z \leftarrow 0}$ where $f$ is a subformula of one of the $F_i$'s and $|P \cup Z| \leq b$, do not vanish at $\bar{\sigma}$. By Theorem 5.4, $G_{w_{m,k} \cdot (\log(t)+1)}$ hits $F(\bar{x} + \bar{\sigma})$. Finally, since $\bar{\sigma}$ is in the image of $\mathcal{G}$, $\mathcal{G} + G_{w_{m,k} \cdot (\log(t)+1)}$ hits $F$, completing the reduction. ■

## 6. Identity Testing Multilinear Read-$k$ Formulae

Before moving on to prove our main theorems, we briefly stop to recall the overall approach. For clarity we only state the non-blackbox approach; the blackbox approach follows the same pattern. We construct an identity test for multilinear sparse-substituted read-$k$ formulae using three tools.

**Lemma 4.1** – a reduction from identity testing multilinear sparse-substituted read-($k$+1) formulae to identity testing multilinear sparse-substituted $\sum^{2}$-read-$k$ formulae.

**Lemma 5.5** – a reduction from identity testing multilinear sparse-substituted $\sum^{2}$-read-$k$ formulae to identity testing multilinear sparse-substituted read-$k$ formulae.

**Lemma 6.1** – an identity test for multilinear sparse-substituted read-once formulae.

Observe that combining the first two reductions reduces identity testing multilinear sparse-substituted read-($k$+1) formulae to identity testing multilinear sparse-substituted read-$k$ formulae. Applying this observation recursively and combining it with Lemma 6.1 as the base case $k = 1$, establishes our main theorem – an identity test for multilinear sparse-substituted read-$k$ for arbitrary $k$. Lemma 6.1 and its corresponding blackbox version are proved in the following subsections immediately before the corresponding main theorem. In the last subsection we develop a specialized blackbox identity test for multilinear sparse-substituted read-$k$ formulae of constant depth.

## 6.1. A Non-Blackbox Identity Test

We begin by describing a simple identity test for multilinear sparse-substituted read-once formula. Note that here *multilinear* is not a redundant qualifier because the sparse substitutions could make the read-once formula non-multilinear.

**Lemma 6.1.** *There is a deterministic algorithm for identity testing multilinear sparse-substituted read-once formulae that runs in time* $\mathrm{poly}(n, s, t)$*, where $s$ denotes the size of the formula, $n$ the number of variables, and $t$ the maximum number of terms in each substituted polynomial.*

*Proof.* First, consider how to identity test sparse polynomials. Examine the list of terms and merge duplicate monomials. The sparse polynomial is nonzero iff any term remains (with a nonzero coefficient). Notice that the same procedure can be used to determine whether a sparse polynomial is constant. This process takes time polynomial in the number of variables and the bound on the sparsity of the substitutions.

To identity test multilinear sparse-substituted read-once formulae, first apply the above procedure to each sparse substitution. If a sparse substitution is non-constant, replace it with a unique new variable; otherwise, replace it with the constant value. The resulting formula is read-once because each variable only occurs in one sparse input. This procedure does not affect the (non)zeroness of the formula. Moreover, the procedure runs in time polynomial in the size of the formula and reduces the problem to testing read-once formulae. There can be no additive cancellation of variables in read-once formulae, therefore the only way for such a formula to be zero is if it is multiplied by the constant zero. Thus, (non)zeroness can be determined by traversing the read-once formula from the bottom up, simplifying gates over constants and eliminating gates that have a multiplication by zero. ■

Combining Lemmas 4.1, 5.5, and 6.1 in the way suggested above proves the following main result.

**Theorem 6.2.** *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.*

*Proof.* We proceed by induction on $k$. The base case is $k = 1$, which is handled by the identity test from Lemma 6.1. Consider the induction step for arbitrary $k+1$. Assume there is an identity test for multilinear sparse-substituted read-$k$ formulae that runs in time $T(k, n, s, t)$. Lemma 5.5 implies there is a deterministic algorithm that runs in time $k^2 n^b \cdot T(k, n, s, t) + n^w \cdot \mathrm{poly}(k, n, s, t))$ that tests multilinear sparse-substituted $\sum^2$-read-$k$ formulae. The lemma bounds $b = O(k^6 \log k \cdot (\log(t)+1))$ and $w = k^{O(k)}(\log(t) + 1)$. Given this identity test for multilinear sparse-substituted $\sum^2$-read-$k$ formulae, Lemma 4.1 results in an identity test for multilinear sparse-substituted read-$(k+1)$ formulae that runs in time $T(k+1, n, s, t) = O(kn^2(k^2 n^b \cdot T(k, n, s, t) + n^w \cdot \mathrm{poly}(k, n, s, t)) + \mathrm{poly}(k, n, s, t))$. Solving this recurrence results in the bound claimed. ■

In the non-blackbox setting we can take things one step further, and identity test *structurally*-multilinear sparse-substituted constant-read formulae. This follows from the next lemma, which reduces the case of non-multilinear sparse substitutions to multilinear sparse substitutions. The argument is based on the following observation: since the multiplication gates in the backbone formula are variable disjoint, the only way to achieve a factor $x^d$ with $d > 1$ in a term is for $x^d$ to be present within one of the sparse substitutions. This allows high-degree terms to be rewritten as multilinear terms over new variables without changing the (non)zeroness of the formula.

**Lemma 6.3.** *Polynomial identity testing structurally-multilinear sparse-substituted formulae polynomial-time mapping reduces to polynomial identity testing multilinear sparse-substituted formulae such that an $n$-variable sparse-substituted read-$k$ formula is mapped to an $O(n \log(knt))$-variable sparse-substituted read-$k$ formula, where $t$ denotes the maximum number of terms in each substituted polynomial.*

*Proof.* Let $F$ be a $n$-variable structurally-multilinear sparse-substituted read-$k$ formula. Write $F = B(f_1, ..., f_r)$, where each of the $f_i$ are sparse polynomials with at most $t$ terms and $B$ is the backbone. Each $f_i$ can be written formally as $f_i = \sum_{j=1}^{t} \alpha_{ij} \prod_{\ell=1}^{n} x_\ell^{d_{ij\ell}}$, where $\alpha_{ij} \in \mathbb{F}$ is the coefficient of the $j^{th}$ term in $f_i$ and $d_{ij\ell}$ is the degree of $x_\ell$ in the $j^{th}$ term of $f_i$. Let $\tau$ be the total number of terms that occur within all $f_i$. Note that $\tau = O(knt)$ as we can assume that $F$ is in standard form. The quantity $\tau$ also bounds the number of distinct degrees of $x_\ell$ occurring in $F$; label these distinct degrees $\{d_{j\ell}\}_{j=1}^{\leq\tau}$. Introduce a new set of variables $\{y_{\ell q}\}_{q=1}^{\lceil \log \tau \rceil}$. Within each $f_i$ replace the occurrence of $x_\ell^{d_{j\ell}}$ with $\prod_{q=1}^{\lceil \log \tau \rceil} y_{\ell q}^{b(j,q)}$, forming $f_i'$, where $b(j,q) \in \{0,1\}$ is the $q^{th}$ bit in the binary expansion of $j$. This process replaces the original $n$ variables with $n\lceil \log \tau \rceil$ new variables. Since $\tau = O(knt)$, there are at most $O(n \log(knt))$ new variables. Define $F' \doteq B(f_1', ..., f_r')$. We argue that $F \equiv 0$ iff $F' \equiv 0$.

Observe that $F'$ is multilinear because $F$ is structurally-multilinear. Also, each $y_{\ell q}$ occurs at most as many times in the $f_i'$'s as $x_\ell$ does in the $f_i$'s, so $F'$ is a sparse-substituted read-$k$ formula.

There is a one-to-one mapping between monomials in the formal expansions $F$ and $F'$. This follows because the only way the product $x_\ell^d$ for $d > 1$ appears in a monomial of $F$ is if it comes from a single $f_i$, since the $x_\ell$ are never multiplied outside of $f_i$. Moreover, the coefficients of the corresponding monomials are also the same. Therefore $F \equiv 0$ iff $F' \equiv 0$. ■

The combination of Lemma 6.3 and Theorem 6.2 proves the non-blackbox part of Theorem 1.2. It yields the following corollary for constant $k$.

**Corollary 6.4.** *There exists a deterministic polynomial identity testing algorithm for structurally-multilinear sparse-substituted constant-read formulae that runs in time $s^{O(1)} \cdot (n \log t)^{O(\log t)}$, where $s$ denotes the size of the formula, $n$ the number of variables, and $t$ the maximum number of terms a substitution consists of.*

When $t$ is constant the algorithm is runs in polynomial time. In particular, we obtain the following corollary.

**Corollary 6.5.** *There exists a deterministic polynomial-time algorithm for identity testing multilinear constant-read formulae.*

## 6.2. A Blackbox Identity Test

We proceed analogously to the previous subsection. We first argue that the SV-generator $G$ works for multilinear sparse-substituted read-once formulae – this extends the argument in [SV09], which worked for read-once formulae. Additionally, the argument is stated with respect to a depth parameter to make a later specialization to constant-depth more concise.

The idea is the following. We recurse on the structure of the multilinear sparse-substituted read-once formula $F$ and argue that the SV-generator takes non-constant subformulae to non-constant subformulae. There are three generic cases, based on the top gate of $F$: (i) addition, (ii) multiplication, and (iii) a sparse-substituted input.

In case (i), the fact that $F$ is read-once implies that addition branches are variable disjoint. This means that there is a variable whose partial derivative eliminates at least half of the formula and reduces the depth by one. Combining this fact with Lemma 2.7 completes the case. In case (ii), the fact that the SV-generator takes non-constant subformulae to non-constant subformulae immediately implies that if $G$ hits the children of a multiplication gate it also hits the gate itself. In case (iii) we can use a similar idea as in the first part of Lemma 3.3. For any variable $x$, either at least half of the terms of the sparse-substituted input depend on $x$, or at least half of the terms the do not. In the former situation setting $x$ to zero eliminates at least half of the terms in the sparse-substituted input; in the latter situation taking $\partial_x$ has the same effect. Combining this with an argument similar to Lemma 2.7 completes the case.

**Lemma 6.6.** *Let $F$ be a nonzero multilinear sparse-substituted depth-$d$ read-once formula. Then $G_w$ hits $F$ for $w \doteq \min\{\lceil\log|var(F)|\rceil, d\} + \lceil\log t\rceil + 1$, where $t$ denotes the maximum number of terms in each substituted polynomial. Moreover, if $F$ is non-constant then so is $F \circ G_w$.*

*Proof.* We proceed by structural induction on $F$. When $|\mathrm{var}(F)| = 0$, $F \circ G_w = F$ and the lemma trivially holds. In the induction step $|\mathrm{var}(F)| \geq 1$, hence $F$ is non-constant. There are three induction cases.

1. The top gate of $F$ is an addition gate. Say $F = \alpha \cdot \sum_{i=1}^m F_i + \beta$, where the $F_i$ are multilinear sparse-substituted depth-$(d-1)$ read-once formulae. Because $F$ is in standard form, it has at least two non-constant branches $F_1$ and $F_2$. Without loss of generality $|\mathrm{var}(F_1)| \leq \frac{|\mathrm{var}(F)|}{2}$. Then, because $F$ is read-once: $F_1$ and $F_2$ are variable disjoint, and for any $x \in \mathrm{var}(F_1)$, $\partial_x F = \partial_x(\alpha \cdot \sum_{i=1}^m F_i + \beta) = \alpha \cdot \partial_x F_1 \not\equiv 0$. Thus, $\partial_x F$ has depth at most $d-1$ and depends on at most $\frac{|\mathrm{var}(F)|}{2}$ variables. Observe that
$$\min\left\{\left\lceil \log\frac{|\mathrm{var}(F)|}{2}\right\rceil, d-1\right\} + \lceil\log t\rceil + 1 = w - 1.$$
The induction hypothesis immediately gives that the $\partial_x F \not\equiv 0$ is hit by $G_{w-1}$. Applying Lemma 2.7 implies that $F \circ (G_{w-1} + G_1)$ is non-constant. By the Proposition 2.6, $G_{w-1} + G_1 = G_w$, completing this case.

2. The top gate of $F$ is a multiplication gate. Say $F = \alpha \cdot \prod_{i=1}^m F_i + \beta$, where the $F_i$ are multilinear sparse-substituted depth-$(d-1)$ read-once formulae. The fact that $F$ is in standard form implies that each $F_i$ is non-constant. The induction hypothesis immediately implies that $G_{w'}$ hits each $F_i$, where $w' = \min\{\lceil\log|\mathrm{var}(F)|\rceil, d-1\} + \lceil\log t\rceil + 1$. Further, each $F_i \circ G_{w'}$ is

non-constant. Combining this with the fact that $w \geq w'$ implies that $\alpha \cdot (\prod_{i=1}^{m} F_i) \circ G_w + \beta$ is non-constant, completing this case.

3. $F$ is a sparse-substituted input. Assume, without loss of generality, that there are no duplicate monomials. Let $\ell \leq t$ be the number of terms in $g$.

   Suppose there exists a variable $x \in \mathrm{var}(F)$ for which $F|_{x \leftarrow 0}$ is non-constant. For such an $x$ write $F = x \cdot \partial_x F + F|_{x \leftarrow 0}$ – this is possible because $F$ is multilinear. If $(\partial_x F) \circ G_{w-1} \not\equiv 0$, applying Lemma 2.7 gives that $F \circ (G_{w-1} + G_1)$ is non-constant and we are done. Otherwise $(\partial_x F) \circ G_{w-1} \equiv 0$. This implies that $\partial_x F$ has more than $\ell/2$ terms, otherwise $G_{w-1}$ would have hit $\partial_x F$ by the induction hypothesis. Therefore $F|_{x \leftarrow 0}$ has at most $\ell/2$ terms. Since $F|_{x \leftarrow 0}$ is non-constant, the induction hypothesis implies that $F|_{x \leftarrow 0} \circ G_{w-1}$ is non-constant, hence is $F \circ G_w$ is non-constant.

   We are left with the case where for all variables $x \in \mathrm{var}(F)$, $F|_{x \leftarrow 0}$ is constant. The facts that $F$ is non-constant and multilinear imply that $F$ contains exactly one term which is non-constant. Therefore $F \circ G_w$ is non-constant for $w \geq 1$, because each coordinate of $G_w$ computes a non-constant polynomial. ■

We formally conclude using Lemmas 4.2, 5.6, and 6.6 to prove the following main result.

**Theorem 6.7.** *For some function $w_k = k^{O(k)}$, the polynomial map $G_{w_k \cdot (\log(t)+1) + k \log n}$ is a hitting set generator for $n$-variate multilinear sparse-substituted read-$k$ formulae, where $t$ denotes the maximum number of terms a substitution consists of.*

*Proof.* We proceed by induction on $k$ and argue that we can set $w_k$ equal to the value $w_{2,k}$ from Lemma 5.6. The base case is immediate from Lemma 6.6. Consider the induction step for arbitrary $k$. Assume that $\mathcal{G} \doteq G_{w_k \cdot (\log(t)+1) + k \log n}$ is a generator for multilinear sparse-substituted read-$k$ formulae. Lemma 5.6 with $m = 2$ implies that $\mathcal{G} + G_{w_k \cdot (\log(t)+1)}$ is a generator for multilinear sparse-substituted $\sum^2$-read-$k$ formulae. Apply Lemma 4.2 to $\mathcal{G}' \doteq \mathcal{G} + G_{w_k \cdot (\log(t)+1)}$. This gives that $G_{w_k \cdot (\log(t)+1) + k \log n} + G_{w_k \cdot (\log(t)+1)} + G_{\log n}$ is a generator for multilinear sparse-substituted read-$(k+1)$ formulae. Apply the basic properties of $G$ from Proposition 2.6 to get that a total seed length of $2w_k \cdot (\log(t) + 1) + (k+1) \log n$ suffices to hit multilinear sparse-substituted read-$(k+1)$ formulae. As we can assume without loss of generality that $2w_k \leq w_{k+1}$, the theorem follows. ■

A multilinear polynomial $F$ on $n$ variables has degree at most $n$. The SV-generator $G_w$ with output length $n$ has total degree at most $n$. Combining these facts and Proposition 2.4 with the previous theorem establishes the blackbox part of Theorem 1.2. In particular, it gives a quasi-polynomial-time blackbox algorithm for identity testing multilinear sparse-substituted constant-read formulae.

**Corollary 6.8.** *There exists a deterministic blackbox polynomial identity testing algorithm for multilinear sparse-substituted constant-read formulae that runs in time $n^{O(\log(n)+\log(t))}$ and queries points from an extension field of size $O(n^2)$, where $n$ denotes the number of variables and $t$ the maximum number of terms a substitution consists of.*

## 6.3. The Special Case of Constant-Depth

We can improve the running time of our blackbox constant-read identity test by further restricting formulae to be constant-depth. We consider only the blackbox case because that is where we can get a substantial improvement. In the constant-depth setting we allow addition and multiplication gates that have arbitrary fanin. In order to specialize our previous argument to the constant depth case, we first give a version of the Fragmentation Lemma (Lemma 3.3) parameterized with respect to the depth. We then carry through the different parameterization in Lemma 4.2 and Theorem 6.7.

**Lemma 6.9 (Bounded-Depth Fragmentation Lemma).** *Let $\emptyset \subsetneq V \subseteq [n]$, $k \geq 1$, and let $F$ be an $n$-variable multilinear sparse-substituted depth-$d$ read$_V$-$k$ formula that depends on at least one variable in $V$. Let $t$ denote the maximum number of terms in each substituted polynomial.*

1. *If $k = 1$ there exist disjoint sets of variables $P, Z \subseteq V$ with $|P \cup Z| \leq (\log(t) + 1)$ such that $\partial_P F|_{Z \leftarrow 0}$ is nonzero and is a product of factors that each depend on at most $\frac{|V|}{2}$ variables from $V$ (and possibly more variables outside of $V$). Moreover, the factors are subformulae of $F$ and at most one formula of the form $\partial_P f|_{Z \leftarrow 0}$ where $f$ is one of the sparse substitutions.*

2. *Otherwise, there exists a variable $x \in V$ such that $\partial_x F$ is nonzero and is the product of*

   a) *subformulae of $F$ that have depth at most $d - 1$, and*

   b) *at most one multilinear sparse-substituted $\sum^k$-read$_V$-$(k-1)$ formula, which is the derivative with respect to $x$ of some subformula of $F$.*

Observe that the read-once case of this lemma is identical to that of the original Fragmentation Lemma. This is because regardless of the depth it takes one derivative to reach a product of smaller formula with at most one large substituted polynomial that needs to reduced. The second half of the lemma is quite similar to the original, however, we make some different choices based on the depth.

*Proof.* Given the original Fragmentation Lemma, we only need to argue the second part. Assume without loss of generality that $V$ only contains variables on which $F$ depends, and that the children of multiplication gates are variable disjoint with respect to $V$.

If none of the variables in $V$ occur $k$ times in $F$, any choice of variable $x \in V$ does the job. So, let us assume that at least one variable in $V$ occurs $k$ times.

The algorithm recurses through the structure of $F$, maintaining the following invariant: The current gate being $g$ visited, $g$, contains below it $k$ occurrences of some variable in $V$. Setting $g$ to be the output gate of $F$ satisfies this invariant initially.

If $g$ is a multiplication gate, recurse on a child of $g$ that depends on a variable from $V$ that occurs $k$ times in $g$. Such a child must exist by the invariant and because $F$ is multilinear.

If $g$ is an addition gate, and at least one of its children, $g_i$, has a variable in $V$ that occurs $k$ times in $g_i$, recurse to $g_i$. Otherwise, select a variable $x \in V$ that occurs $k$ times in $g$ ending the recursion. In this case all of the children of $g$ are multilinear sparse-substituted read$_V$-$(k-1)$ formulae. Since $g$ has at most $k$ children that contain $x$, $\partial_x g$ can be represented as a $\sum^k$-read$_V$-$(k-1)$ formula.

In the partial derivative $\partial_x F$, all unvisited addition branches along the path from the output gate of $F$ to the final $g$ have been eliminated. Also, all unvisited multiplication branches along the path become factors of $\partial_x F$ together with $\partial_x g$. More formally, $\partial_x F = (\partial_x g) \prod_i F_i$, where the $F_i$ are the unvisited multiplication branches. The $F_i$'s are multilinear sparse-substituted read$_V$-$k$ formulae that have depth at most $d-1$, because they are the children of some multiplication gate in $F$. When the process stops at an addition gate, $\partial_x g$ is a multilinear sparse-substituted $\sum^k$-read$_V$-$(k-1)$ formula that may depend on many variables from $V$. Note that since we assumed $F$ depends on all variables in $V$, $\partial_x F \not\equiv 0$. ■

Lemma 6.9 leads to the following variant of Lemma 4.2 in the bounded-depth setting.

**Lemma 6.10.** *For an integer $k \geq 1$, let $\mathcal{G}$ be a generator for $n$-variate multilinear sparse-substituted depth-$d$ $\sum^{k+1}$-read-$k$ formulae and let $F$ be a nonzero $n$-variable multilinear sparse-substituted depth-$d$ read-$(k+1)$ formula. Then $\mathcal{G} + G_d$ hits $F$.*

*Proof.* First observe that if $F$ is read-$k$, we are immediately done because $F \circ \mathcal{G} \not\equiv 0$ and $\bar{0}$ is in the range of $G$ (by the first item of Proposition 2.6).

The proof goes by induction on $d$. If $d = 0$, the lemma holds trivially as $F$ is constant. If $d = 1$, $F$ is a read-once formula, which is covered by the above observation. For the induction step, by the above observation we can assume that $F$ is read-$(k+1)$ and not read-$k$. Therefore, $F$ meets the conditions to apply the second part of the Fragmentation Lemma for bounded depth formulae (Lemma 6.9). The lemma produces a variable $x \in \mathrm{var}(F)$. The factors of $\partial_x F$ all have depth at most $d-1$ and are multilinear sparse-substituted read-$(k+1)$ formulae, except for at most one which is a $\sum^{k+1}$-read-$k$ formula. The induction hypothesis gives that the former factors of $\partial_x F$ are all hit by $\mathcal{G} + G_{d-1}$. The latter factor (if it occurs) is hit by $\mathcal{G}$. Applying Lemma 2.7 gives that $\mathcal{G} + G_{d-1} + G_1$ hits $F$. Recalling Proposition 2.6 implies that $\mathcal{G} + G_d$ hits $F$. ■

We can use the previous lemma with Lemmas 5.6 and 6.6 to construct a hitting set generator specialized to bounded depth. The proof is almost identical to Theorem 6.7, except that fanin of the reduced instance increases to $k+1$ from 2. This weakens the parameterization of the seed length with respect to $k$.

**Theorem 6.11.** *For some function $w_k = k^{O(k^2)}$, the polynomial map $G_{w_k \cdot (\log(t)+1)+kd}$ is a hitting set generator for $n$-variate multilinear sparse-substituted depth-$d$ read-$k$ formulae, where $t$ denotes the maximum number of terms a substitution consists of.*

*Proof.* We proceed by induction on $k$ and argue that we can set $w_k$ equal to the value $w_{k+1,k}$ from Lemma 5.6. The base case is immediate from Lemma 6.6. Consider the induction step for arbitrary $k$. Assume that $\mathcal{G} \doteq G_{w_k \cdot (\log(t)+1)+kd}$ is a generator for multilinear sparse-substituted depth-$d$ read-$k$ formulae. Lemma 5.6 with $m = k+1$ implies that $\mathcal{G} + G_{w_k \cdot (\log(t)+1)}$ is a generator for multilinear sparse-substituted depth-$d$ $\sum^{k+1}$-read-$k$ formulae. Apply Lemma 6.10 to $\mathcal{G}' \doteq \mathcal{G} + G_{w_k \cdot (\log(t)+1)}$. This gives that $G_{w_k \cdot (\log(t)+1)+kd} + G_{w_k \cdot (\log(t)+1)} + G_d$ is a generator for multilinear sparse-substituted depth-$d$ read-$(k+1)$ formulae. Apply the basic properties of $G$ from Proposition 2.6 to get that a total seed length of $2w_{k+1} \cdot (\log(t)+1) + (k+1)d$ suffices to hit multilinear sparse-substituted depth-$d$ read-$(k+1)$ formulae. As we can assume without loss of generality that $2w_k \leq w_{k+1}$, the theorem follows. ■

Analogous to the unbounded depth setting, combining Theorem 6.11 with Proposition 2.4 establishes Theorem 1.3 and the following corollary for constant-read constant-depth formulae.

**Corollary 6.12.** *There exists a deterministic blackbox polynomial identity testing algorithm for multilinear sparse-substituted constant-depth constant-read formulae that runs in time $n^{O(\log t)}$ and queries points from an extension field of size $O(n^2)$, where $n$ denotes the number of variables and $t$ the maximum number of terms a substitution consists of.*

The important difference between the above corollary and Corollary 6.8 is that the exponent no longer depends on $n$. Additionally, if the sparsity of substituted polynomials is constant the algorithm becomes polynomial-time. In particular, we obtain the following corollary.

**Corollary 6.13.** *There is a deterministic polynomial-time blackbox algorithm for identity testing multilinear constant-depth constant-read formulae.*

## Acknowledgements

MA and DvM would like to thank Amir Shpilka for bringing them into touch with IV.

## References

[AB03] M. Agrawal and S. Biswas. Primality and identity testing via chinese remaindering. *Journal of the ACM*, 50(4):429–443, 2003.

[Agr03] M. Agrawal. On derandomizing tests for certain polynomial identities. In *Proceedings of 18th Annual IEEE Conference on Computational Complexity*, pages 355–359, 2003.

[Agr05] M. Agrawal. Proving lower bounds via pseudo-random generators. *Foundations of Software Technology and Theoretical Computer Science*, pages 92–105, 2005.

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

[AvM10] S. Aaronson and D. van Melkebeek. A note on circuit lower bounds from derandomization. *Electronic Colloquium on Computational Complexity*, 2010. TR10-105.

[BHLV09] M. Bläser, M. Hardt, R. Lipton, and N. Vishnoi. Deterministically testing sparse polynomial identities of unbounded degree. *Information Processing Letters*, 109(3):187–192, 2009.

[BOT88] 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*, pages 301–309, 1988.

[DL78] R. DeMillo and R. Lipton. A probabilistic remark on algebraic program testing. *Information Processing Letters*, 7(4):193–195, 1978.

[DS07] Z. Dvir and A. Shpilka. Locally decodable codes with two queries and polynomial identity testing for depth 3 circuits. *SIAM Journal on Computing*, 36(5):1404–1434, 2007.

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

[KMSV10] 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*, pages 649–658, 2010.

[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*, pages 216–223, 2001.

[KS08] 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*, pages 280–291, 2008.

[KvMS09] 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*, pages 574–587, 2009.

[Lov79] L. Lovász. On determinants, matchings and random algorithms. In *Fundamentals of Computation Theory*, volume 79, pages 565–574, 1979.

[Sax08] N. Saxena. Diagonal circuit identity testing and lower bounds. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming*, pages 60–71, 2008.

[Sax09] N. Saxena. Progress on polynomial identity testing. *Bulletin of the EATCS*, 99:49–79, 2009.

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

[SS09] 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*, pages 137–148, 2009.

[SS10] N. Saxena and C. Seshadhri. Blackbox identity testing for bounded top fanin depth-3 circuits: The field doesn't matter. *Electronic Colloquium on Computational Complexity*, 2010. TR10-167.

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

[SV09] A. Shpilka and I. Volkovich. Improved polynomial identity testing for read-once formulas. In *Proceedings of the 13th International Workshop on Randomization and Computation*, pages 700–713, 2009.

[SV10] S. Saraf and I. Volkovich. Black-box identity testing of depth-4 multilinear circuits, 2010. Manuscript.

[Zip79] R. Zippel. Probabilistic algorithms for sparse polynomials. *Symbolic and Algebraic Computation*, pages 216–226, 1979.

# A. Standard Form of Read-$k$ Formulae

Given a read-$k$ formula, it can be transformed into a standard form read-$k$ formula where constants only occur in the $\alpha$ and $\beta$ of gates and the formula has at most $O(kn)$ gates.

**Proposition A.1.** *There is an algorithm that transforms a given read-k formula $F$ on $n$ variables into an equivalent read-k formula $F'$, such that $F'$ has at most $kn$ gates. Moreover, the algorithm preserves multilinearity and runs in time polynomial in the size of $F$.*

*Proof.* Without loss of generality assume has $F$ fanin at most 2. Consider the following simplification rules for a gate $g$ in $F$.

1. Suppose $g$ has one child. Then $g = \alpha \cdot g' + \beta$. If $g'$ is a constant, replace $g$ with the constant $\alpha \cdot g' + \beta$. If $g'$ is an input variable do nothing. If $g'$ is another gate, where $g' = \alpha' \cdot g'' + \beta'$, replace $g$ with $(\alpha\alpha') \cdot g'' + (\alpha\beta' + \beta)$, explicitly computing the new constants.

2. Suppose $g$ has two children. Then $g = \alpha \cdot (g_1 \text{ op } g_2) + \beta$. If both children are constant replace $g$ with the constant it computes. If one child is constant (wlog $g_1$), and op $=$ $+$, replace $g$ with $\alpha \cdot g_2 + (\alpha g_1 + \beta)$; if op $=$ $\times$, replace $g$ with $(\alpha g_1) \cdot g_2 + \beta$; otherwise do nothing.

Note that the simplification rules preserve multilinearity. Repeatedly apply these rules until a fixed point is reached, call this $F'$. Inspection of the rules gives that $g$ is always replaced by an equivalent formula, therefore $F' \equiv F$. Since non-constant parts of the formula are never duplicated $F'$ is read-$k$. If $g$ is an internal gate with only one child, it is eliminated; if $g$ is an internal gate with two children at least one of which is a constant, $g$ is changed by these rules. Thus, if a fixed point is reached, all internal gates must have two children that are not constants.

This implies that all the original constants have been moved into the $\alpha$'s and $\beta$'s of the gates and inputs. Therefore $F'$ can be viewed as a binary tree where the at most $kn$ inputs are leaves. Thus, the total number of gates in $F'$ is at most $kn$. This means that there are at most $2kn$ gates and input pairs $(\alpha, \beta)$.

Since the total number of gates and constants decreases with each step, this process can repeat at most the size of $F$ many times before the fixed point is reached. ■

The previous lemma only bounds the number of gates in the resulting formula. The constants $\alpha$ and $\beta$ at each gate and inputs in $F'$ are bounded in bit-length by the size of the original formula $F$. This means that evaluating the formula $F'$ still incurs cost polynomial in the size of $F$.

The standard form can be easily generalized to sparse-substituted formulae, because the internal gate structure is the same where the sparse polynomials are treated as inputs. The standard form can also be specialized to bounded depth formulae where the gate fanin is unbounded.

# B. Homogeneous Rank Bound to Inhomogeneous Rank Bound

In Section 3 we briefly discussed the rank bound of [SS09]. Formally, they show the following theorem.

**Lemma B.1 ([SS09, Theorem 2]).** *The rank of a homogeneous, simple, and minimal depth-three formula that is zero has rank at most $O(m^3 \log d)$, where $m$ denotes the top fanin of the formula and $d$ the degree.*

The previous lemma requires that the formula be homogeneous. This means that the bottom level addition gates compute linear *forms* (which contain no constant terms) rather than more general linear *functions.* In the paper originating the rank bound, Dvir and Shpilka [DS07] prove the following homogenization lemma. For completeness, we state and prove the lemma (almost verbatim) and then argue that the version of the rank bound we use (Lemma 2.9) follows.

**Lemma B.2 ([DS07, Lemma 3.5]).** *There exists a polynomial-time algorithm that transforms a depth-three formula $F$ into a homogeneous depth-three formula $F'$ such that $F \equiv 0$ iff $F' \equiv 0$. The algorithm preserves simplicity, minimality, and rank, and does not increase the top fanin or degree*

*Proof.* Let $F = \sum_{i=1}^{m} \prod_{j=1}^{d} L_{ij}(X)$, where

$$L_{ij}(x) = L_{ij}^0 + \sum_{\ell=1}^{n} L_{ij}^{\ell} \cdot x_\ell$$

are the linear functions appearing in $F$ over the input variables $X = \{x_1, ..., x_n\}$. Introduce a new variable $y$. Over the input variables $x_1, ..., x_n, y$ define

$$L'_{ij}(x, y) \doteq L_{ij}^0 \cdot y + \sum_{\ell=1}^{n} L_{ij}^{\ell} \cdot x_\ell,$$

and

$$F'(x, y) \doteq \sum_{i=1}^{m} y^{d-d_i} \prod_{j=1}^{d_i} L'_{ij}.$$

Clearly, $F'$ is a homogeneous depth-three degree-$d$ formula with top fanin $m$. Further, $F'$ can be computed in time polynomial in the size of $F$. Write

$$F(x) \equiv \sum_{i=0}^{d} P_i(x),$$

where $P_i(x)$ denotes the homogeneous part of degree $i$ of $F(x)$, then

$$F'(x, y) \equiv \sum_{i=0}^{d} P_i(x) y^{d-i}.$$

Therefore $F \equiv 0$ iff $F' \equiv 0$.

The preservation properties follow from the construction. ■

Applying Lemma B.2 to Theorem B.1 generalizes the latter to inhomogeneous formulae (using the property that simplicity, minimality, and rank are preserved). To get the version specialized to multilinear formulae that we use (Lemma 2.9), one observes that for multilinear formulae the rank is at least the degree. This implies that the rank $r$ satisfies $r = O(m^3 \log r)$, from which an upper bound of $O(m^3 \log m)$ on $r$ follows.