Subexponential Size Hitting Sets for Bounded Depth Multilinear Formulas*

Rafael Oliveira[1], Amir Shpilka[2], and Ben Lee Volk[2]

1 **Department of Computer Science, Princeton University, USA**
rmo@cs.princeton.edu

2 **Department of Computer Science Tel Aviv University, Israel**
shpilka@post.tau.ac.il, benleevolk@gmail.com

Abstract

In this paper we give subexponential size hitting sets for bounded depth multilinear arithmetic formulas. Using the known relation between black-box PIT and lower bounds we obtain lower bounds for these models.

For depth-3 multilinear formulas, of size $\exp(n^{\delta})$, we give a hitting set of size $\exp\left(\tilde{O}\left(n^{2/3+2\delta/3}\right)\right)$. This implies a lower bound of $\exp(\tilde{\Omega}(n^{1/2}))$ for depth-3 multilinear formulas, for some explicit polynomial.

For depth-4 multilinear formulas, of size $\exp(n^{\delta})$, we give a hitting set of size $\exp\left(\tilde{O}\left(n^{2/3+4\delta/3}\right)\right)$. This implies a lower bound of $\exp(\tilde{\Omega}(n^{1/4}))$ for depth-4 multilinear formulas, for some explicit polynomial.

A regular formula consists of alternating layers of $+, \times$ gates, where all gates at layer i have the same fan-in. We give a hitting set of size (roughly) $\exp\left(n^{1-\delta}\right)$, for regular depth-$d$ multilinear formulas of size $\exp(n^{\delta})$, where $\delta = O(\frac{1}{\sqrt{5}^d})$. This result implies a lower bound of roughly $\exp(\tilde{\Omega}(n^{\frac{1}{\sqrt{5}^d}}))$ for such formulas.

We note that better lower bounds are known for these models, but also that none of these bounds was achieved via construction of a hitting set. Moreover, no lower bound that implies such PIT results, even in the white-box model, is currently known.

Our results are combinatorial in nature and rely on reducing the underlying formula, first to a depth-4 formula, and then to a read-once algebraic branching program (from depth-3 formulas we go straight to read-once algebraic branching programs).

1998 ACM Subject Classification F.1.1 Models of Computation, F.1.3 Complexity Measures and Classes

Keywords and phrases Arithmetic Circuits, Derandomization, Polynomial Identity Testing

Digital Object Identifier 10.4230/LIPIcs.CCC.2015.304

1 Introduction

Arithmetic circuits are the standard model for computing polynomials. Roughly speaking, given a set of variables $X = \{x_1, \ldots, x_n\}$, an arithmetic circuit uses additions and multiplications to compute a polynomial f in the set of variables X. An arithmetic formula is an

* The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement number 257575, from the Israel Science Foundation (grant number 339/10), from NSF grant CCF-1217416 and from the Sloan fellowship.

arithmetic circuit whose computation graph is a tree. An arithmetic circuit (or formula) is multilinear if the polynomial computed at each of its gates is multilinear (as a formal polynomial), that is, in each of its monomials the power of every input variable is at most one (see Section 1.1 for definition of the models studied in this paper)

Two outstanding open problems in complexity theory are to prove exponential lower bounds on the size of arithmetic circuits, i.e., to prove a lower bound on the number of operations required to compute some polynomial f, and to give efficient deterministic polynomial identity testing (PIT for short) algorithms for them. The PIT problem for arithmetic circuits asks the following question: given an arithmetic circuit Φ computing a polynomial f, determine, *efficiently and deterministically*, whether "$f \equiv 0$". The black-box version of the PIT problem asks to construct a small *hitting set*, i.e., a set of evaluation points $\mathcal{H}$, for which any such non-zero f does not vanish on all the points in $\mathcal{H}$.

It is known that solving any one of the problems (proving lower bound or deterministic PIT), with appropriate parameters, for small depth (multilinear) formulas, is equivalent to solving it in the general (multilinear) case [37, 6, 24, 15, 36]. It is also known that these two problems are tightly connected and that solving one would imply a solution to the other, both in the general case [16, 17, 1] and in the bounded depth case[1] [11]. We note that in the multilinear case it is only known that hitting sets imply circuit lower bounds but not vice versa.

In this work we study the PIT problem for several models of bounded depth multilinear formulas. Our main results are subexponential size hitting sets for depth-3 and depth-4 multilinear formulas of subexponential size and for *regular* depth-d multilinear formulas of subexponential size (with construction size deteriorating among the different models). Using the connection between explicit hitting sets and circuit lower bounds we get, as corollaries, subexponential lower bounds for these models.

1.1 Models for Computing Multilinear Polynomials

An arithmetic circuit Φ over the field $\mathbb{F}$ and over the set of variables X is a directed acyclic graph as follows. Every vertex in Φ of in-degree 0 is labelled by either a variable in X or a field element in $\mathbb{F}$. Every other vertex in Φ is labelled by either $\times$ or $+$. An arithmetic circuit is called a formula if it is a directed tree (whose edges are directed from the leaves to the root). The vertices of Φ are also called gates. Every gate of in-degree 0 is called an input gate. Every gate of out-degree 0 is called an output gate. Every gate labelled by $\times$ is called a product gate. Every gate labelled by $+$ is called a sum gate. An arithmetic circuit computes a polynomial in a natural way. An input gate labelled by $y \in \mathbb{F} \cup X$ computes the polynomial y. A product gate computes the product of the polynomials computed by its children. A sum gate computes the sum of the polynomials computed by its children.

A polynomial $f \in \mathbb{F}[X]$ is called multilinear if the degree of each variable in f is at most one. An arithmetic circuit (formula) Φ is called multilinear if every gate in Φ computes a multilinear polynomial.

In this work we are interested in small depth multilinear formulas. A depth-3 $\Sigma\Pi\Sigma$ formula is a formula composed of three layers of alternating sum and product gates. Thus,

[1] The result of [11] is more restricted than the results for circuits with no depth restrictions.

every polynomial computed by a ΣΠΣ formula of size s has the following form

$$f = \sum_{i=1}^{s} \prod_{j=1}^{d_i} \ell_{i,j},$$

where the $\ell_{i,j}$ are linear functions. In a ΣΠΣ multilinear formula, it holds that in every product gate, $\prod_{j=1}^{d_i} \ell_{i,j}$, the linear functions $\ell_{i,1}, \ldots, \ell_{i,d_i}$ are supported on disjoint sets of variables.

Similarly, a depth-4 ΣΠΣΠ formula is a formula composed of four layers of alternating sum and product gates. Thus, every polynomial computed by a ΣΠΣΠ formula of size s has the following form

$$f = \sum_{i=1}^{s} \prod_{j=1}^{d_i} Q_{i,j},$$

where the $Q_{i,j}$ are computed at the bottom ΣΠ layers and are s-sparse polynomials, i.e., polynomials that have at most s monomials. As in the depth-3 case, we have that at every product gate the polynomials $Q_{i,1}, \ldots, Q_{i,d_i}$ are supported on disjoint sets of variables.

Another important definition for us is that of a regular depth-d formula. A regular depth-d formula is specified by a list of d integers $(a_1, p_1, a_2, p_2, \ldots)$. It has d layers of alternating sum and product gates. The fan-in of every sum gate at the $(2i-1)$'th layer is a_i and, similarly, the fan-in of every product gate at the $(2i)$'th layer is p_i. For example, a depth-4 formula that is specified by the list (a_1, p_1, a_2, p_2) has the following form:

$$f = \sum_{i=1}^{a_1} \prod_{j=1}^{p_1} Q_{i,j},$$

where each $Q_{i,j}$ is a polynomial of degree p_2 that has (at most) a_2 monomials. As before, a regular depth-d multilinear formula is a regular depth-d formula in which every gate computes a multilinear polynomial.

Regular formulas were first defined by Kayal, Saha and Saptharishi [21], who proved quasi-polynomial lower bounds for logarithmic-depth regular formulas. It is interesting to note that in the reductions from general (multilinear) circuits/formulas to depth-d (multilinear) formulas, one gets a regular depth-d (multilinear) formula [37, 6, 24, 36].

Finally, we also need to consider the model of Read-Once Algebraic Branching Programs (ROABPs) as our construction is based on a reduction to this case. Algebraic branching programs were first defined in the work of Nisan [25] who proved exponential lower bounds on the size of non-commutative ABPs computing the determinant or permanent polynomials. Roughly, an algebraic branching program (ABP) consists of a layered graph with edges going from the i'th layer to the $(i+1)$'th layer. The first layer consists of a single source node and the last layer contains a single sink. The edges of the graph are labeled with polynomials (in our case we only consider linear functions as labels). The weight of a path is the product of the weights of the edges in the path. The polynomial computed by the ABP is the sum of the weights of all the paths from the source to the sink. An ABP is called a read-once ABP (ROABP) if the only variable appearing on edges that connect the i'th and the $(i+1)$'th layer is x_i. It is clear that a ROABP whose edges are labeled with linear functions computes a multilinear polynomial.

1.2 Polynomial Identity Testing

In the PIT problem we are given an arithmetic circuit or formula Φ, computing some polynomial f, and we have to determine whether "$f \equiv 0$". That is, we are asking if f is the

zero polynomial in $\mathbb{F}[x_1, \ldots, x_n]$. By the Schwartz-Zippel-DeMillo-Lipton lemma [38, 32, 9], if $0 \neq f \in \mathbb{F}[x_1, \ldots, x_n]$ is a polynomial of degree $\leq d$, and $\alpha_1, \ldots, \alpha_n \in A \subseteq \mathbb{F}$ are chosen uniformly at random, then $f(\alpha_1, \ldots, \alpha_n) = 0$ with probability at most[2] $d/|A|$. Thus, given Φ, we can perform these evaluations efficiently, giving an efficient randomized procedure for answering "$f \equiv 0$?". It is an important open problem to find a derandomization of this algorithm, that is, to find a *deterministic* procedure for PIT that runs in polynomial time (in the size of Φ).

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

The problem of derandomizing PIT has received a lot of attention in the past few years. In particular, many works examine a specific class of circuits $\mathcal{C}$, and design PIT algorithms only for circuits in that class. One reason for this attention is the strong connection between deterministic PIT algorithms for a class $\mathcal{C}$ and lower bounds for $\mathcal{C}$. This connection was first observed by Heintz and Schnorr [16] (and later also by Agrawal [1]) for the black-box model and by Kabanets and Impagliazzo [18] for the white-box model (see also Dvir, Shpilka and Yehudayoff [11] for a similar result for bounded depth circuits). Another motivation for studying the problem is its relation to algorithmic questions. Indeed, the famous deterministic primality testing algorithm of Agrawal, Kayal and Saxena [3] is based on derandomizing a specific polynomial identity. Finally, the PIT problem is, in some sense, the most general problem that we know today for which we have randomized coRP algorithms but no polynomial time algorithms, thus studying it is a natural step towards a better understanding of the relation between RP and P. For more on the PIT problem we refer to the survey by Shpilka and Yehudayoff [35].

Among the most studied circuit classes we find Read-Once Algebraic Branching Programs [14, 12, 2], set-multilinear formulas [28, 13, 5], depth-3 formulas [10, 23, 20, 22, 31], multilinear depth-4 formulas (and some generalizations of them) [19, 30, 4] and bounded-read multilinear formulas [33, 34, 8, 4]. We note that none of these results follow from a reduction a la Kabanets-Impagliazzo [18] (or the reduction of [11] for bounded depth circuits) from PIT to lower bounds. Indeed, this reduction does not work for the restricted classes mentioned here. In particular, for the multilinear model no reduction from PIT to lower bounds is known. That is, even given lower bounds for multilinear circuits/formulas (e.g., the exponential lower bound of Raz and Yehudayoff [29] for constant depth multilinear formulas) we do not know how to construct a PIT algorithm for a related model.

The works on depth-3 and multilinear depth-4 formulas gave polynomial time algorithms only when the fan-in of the top gate (the output gate) is constant, and became exponential time when the top fan-in was $\Omega(n)$, both in the white-box and black-box models [23, 31, 30]. Raz and Shpilka [28] gave a polynomial time PIT for set-multilinear depth-3 circuits and Forbes and Shpilka [14] and Agrawal, Saha and Saxena [5] gave a quasi-polynomial size hitting set for the model. Recall that in a depth-3 set-multilinear formula, the variables are partitioned to sets, and each linear function at the bottom layer only involves variables

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

from a single set. Recently, Agrawal et al. [2] gave a subexponential white-box algorithm for a depth-3 formula that computes the sum of c set-multilinear formulas, each of size s, with respect to different partitions of the variables. The running time of their algorithm is $n^{O(2^c n^{1-\frac{2}{2^c}} \log s)}$. In particular, for $c = O(\log\log(n))$ the running time is $\exp(n)$.

Thus, prior to this work there were no subexponential PIT algorithms, even for depth-3 multilinear formulas with top fan-in n.

1.3 Our Results

▶ **Remark.** *Throughout this paper, we assume that for formulas of size 2^{n^δ}, the underlying field $\mathbb{F}$ is of size at least $|\mathbb{F}| \geq 2^{n^{2\delta}\,\mathsf{poly}\log(n)}$, and that if this is not the case then we are allowed to query the formula on inputs from an extension field of the appropriate size. In particular, all our results hold over fields of characteristic zero or over fields of size $\exp(n)$.*

We give subexponential size hitting sets for depth-3, depth-4 and regular depth-d multilinear formulas, of subexponential size. In particular we obtain the following results.

▶ **Theorem 1.1.** *There exists a hitting set $\mathcal{H}$ of size $|\mathcal{H}| = 2^{\tilde{O}(n^{\frac{2}{3}+\frac{2}{3}\delta})}$ for the class of $\Sigma\Pi\Sigma$ multilinear formulas of size 2^{n^δ}.*

This gives a significant improvement to the recent result, mentioned above, of Agrawal et al. [2] who studied sum of set-multilinear formulas. From the connection between hitting sets and circuit lower bounds [16, 1] we obtain the following corollary.

▶ **Corollary 1.2.** *There is an explicit multilinear polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$, whose coefficients can be computed in exponential time, such that any depth-3 multilinear formula for f has size $\exp(\tilde{\Omega}(\sqrt{n}))$.*

This lower bound is weaker than the exponential lower bound of Nisan and Wigderson for this model [26]. Yet, it is interesting to note that we can get such a strong lower bound from a PIT algorithm. Next, we present our result for depth-4 multilinear formulas.

▶ **Theorem 1.3.** *There exists a hitting set $\mathcal{H}$ of size $|\mathcal{H}| = 2^{\tilde{O}(n^{2/3+4\delta/3})}$ for the class of $\Sigma\Pi\Sigma\Pi$ multilinear formulas of size 2^{n^δ}.*

Again, from the connection between hitting sets and circuit lower bounds we obtain the following corollary.

▶ **Corollary 1.4.** *There is an explicit multilinear polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$, whose coefficients can be computed in exponential time, such that any depth-4 multilinear formula for f has size $\exp(\tilde{\Omega}(n^{1/4}))$.*

The best known lower bound for depth-4 multilinear formulas is $\exp(n^{1/2})$ due to Raz and Yehudayoff [29], thus, as in the previous case, the term in the exponent of our lower bound is the square root of the corresponding term in the best known lower bound. For regular depth-d multilinear formulas we obtain the following result.

▶ **Theorem 1.5.** *There exists a hitting set $\mathcal{H}$ of size $|\mathcal{H}| = 2^{\tilde{O}(n^{1-\delta/3})}$ for the class of regular depth-d multilinear formulas of size 2^{n^δ}, where $\delta \leq \frac{1}{5^{\lfloor d/2 \rfloor + 1}} = O\left(\frac{1}{\sqrt{5}^d}\right)$.*

As before we obtain a lower bound for such formulas.

▶ **Corollary 1.6.** *There is an explicit multilinear polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$, whose coefficients can be computed in exponential time, such that any regular depth-d multilinear formula for f has size $\exp(\tilde{\Omega}(n^{\frac{1}{5^{\lfloor d/2 \rfloor + 1}}}))$.*

We note that Raz and Yehudayoff gave an $\exp(n^{\Omega(\frac{1}{d})})$ lower bound for depth-d multilinear formulas, which is much stronger than what Corollary 1.6 gives. Yet, our result also gives a PIT algorithm, which does not follow from the results of [29]. As we later explain, we lose a square root in the term at the exponent for every increase of the depth and this is the reason that we get only $\exp(n^{1/\exp(d)})$ instead of $\exp(n^{1/d})$.

1.4 Proof Overview

We first discuss our proof technique for the case of depth-3 multilinear formulas. Our (idealized) aim is to reduce such a formula Φ to a depth-3 multilinear formula in which each linear function is of the form $\alpha x + \beta$. That is, each linear function contains at most one variable. If we manage to do that then we can use the quasi-polynomial sized hitting set of [13, 2] for this model.

Of course, the problem with the above argument is that in general, depth-3 formulas have more than one variable per linear function. To overcome this difficulty, we will partition the variables to several sets $T_1, \ldots, T_m$ and hope that each linear function in the formula contains at most one variable from each T_i. If we can do that then we would use the hitting set for each set of variables T_i and combine those sets together to get our hitting set. That is, the combined hitting set is composed of concatenation of all vectors of the form $v_1 \circ v_2 \circ \ldots \circ v_m$ where v_i comes from the hitting set restricted to the variables in T_i (the concatenation is performed in a way that respects the partition of course). Thus, if we can carry out this procedure then we will get a hitting set of size roughly $n^{m \log n}$. This step indeed yields a hitting set, since when we restrict our attention to each T_i and think of the other variables as constants in some huge extension field, then we do get a small ROABP (in the variables of T_i) and hence plugging in the hitting set of [13, 2] gives a non-zero polynomial. Thus, we can first do this for T_1 and obtain some good assignment v_1 that makes the polynomial non-zero after substituting v_1 to T_1. Then we can find v_2 etc.

There are two problems with the above argument. One problem is how to find such a good partition. The second is that this idea simply cannot work as is. For example, if we have the linear function $x_1 + \cdots + x_n$, then it will have a large intersection with each T_i.

We first deal with the second question. to overcome the difficulty posed by the example, we would like to somehow "get rid" of all linear functions of large support and then carry out the idea above. To remove linear functions with large support from the formula we use another trick. Consider a variable x_k that appears in a linear function ℓ_0 that has a large support. Assume that $\frac{\partial f}{\partial x_k} \not\equiv 0$ as otherwise we can ignore x_k. Now, because of multilinearity, we can transform our original formula Φ to a formula computing $\frac{\partial f}{\partial x_k}$. This is done by replacing each linear function $\ell(X) = \sum_{i=1}^{n} \alpha_i x_i + \alpha_0$ with the constant α_k. In particular, the function ℓ_0 that used to have a high support does not appear in the new formula. Furthermore, this process does not increase the support size of any other linear function. A possible issue is that if we have to repeat this process for every function of large support then it seems that we need to take a fresh derivative for every such linear function. The point is that because we only care about linear functions that have a large support to begin with, we can find a variable that simultaneously appears in many of those functions and thus one derivative will eliminate many of the "bad" linear functions. Working out the parameters, we see that we need to take roughly $n^{\epsilon} \cdot \log |\Phi|$ many derivatives to reduce to the case where all linear functions have support size at most $n^{1-\epsilon}$.

Now we go back to our first problem. We can assume that we have a depth-3 formula in which each linear function has support size at most $n^{1-\epsilon}$ and we wish to find a partition of the variables to sets $T_1, \ldots, T_m$ so that each T_i contains at most one variable from each

linear function. This cannot be achieved as a simple probabilistic argument shows, so we relax our requirement and only demand that in each multiplication gate (of the formula) only a few linear functions have a large intersection. If at most k linear functions in each gate have a large intersection, we can expand each multiplication gate to at most n^k new gates (by simply expanding all linear functions that have large intersection) and then apply our argument. As we will be able to handle subexponential size formulas, this blow up is tolerable for us.

Note that if we were to pick the partition at random, when $m = n^{1-\epsilon+\gamma}$, for some small γ, then we will get that with very high probability at most n^δ linear functions will have intersection at most n^δ with each T_i, where δ is such that $|\Phi| < \exp(n^\delta)$. To get a deterministic version of this partitioning, we simply use an n^δ-wise independent family of hash functions $\{h : [n] \to [m]\}$. Each hash function h induces a partition of the variables to $T_i = \{x_k \mid h(k) = i\}$. Because of the high independence, we are guaranteed that there is at least one hash function that induces a good partition.

Now we have all the ingredients in place. To get our hitting set we basically do the following (we describe the construction as a process, but it should be clear that every step can be performed using some evaluation vectors).

1. Pick $n^\epsilon \cdot \log|\Phi|$ many variables and compute a black-box for the polynomial that is obtained by taking the derivative of f with respect to those variables. The cost of this step is roughly $\binom{n}{n^\epsilon \cdot \log|\Phi|} \cdot 2^{n^\epsilon \cdot \log|\Phi|}$, where the first term is for picking the variables and the second is what we have to pay to get access to the derived polynomial.
2. Partition the remaining variables to (roughly) $n^{1-\epsilon/2}$ many sets using a (roughly) $\log|\Phi|$-wise independent family of hash functions. The cost of this step is roughly $n^{\log|\Phi|}$ as this is the size of the hash function family.
3. Plug in a fresh copy of the hitting set of [13, 2] to each of the sets of variables T_i. The cost is roughly $n^{\log n \cdot n^{1-\epsilon/2}}$.

Combining everything we get a hitting set of size roughly

$$\left(\binom{n}{n^\epsilon \cdot \log|\Phi|} \cdot 2^{n^\epsilon \cdot \log|\Phi|}\right) \cdot \left(n^{\log|\Phi|}\right) \cdot \left(n^{\log n \cdot n^{1-\epsilon/2}}\right) \approx 2^{\tilde{O}\left(n^{1-\epsilon/2} + n^\epsilon \log|\Phi|\right)}.$$

Optimizing the parameters we get our hitting set.

We would like to use the same approach also for the case of depth-4 formulas. Here the problem is that in the two bottom layers the formula computes a polynomial and not a linear function. In particular, when taking a derivative we are no longer removing functions that have large support. Nevertheless, we can still use a similar idea. We show there is a variable x_i that by either setting $f|_{x_i=0}$ or considering $\frac{\partial f}{\partial x_i}$, we are guaranteed that the total sparsity of all polynomials that have large support goes down by some non-negligible factor. Thus, repeating this process (of either setting a variable to 0 or taking a derivative) $n^\epsilon \cdot \log|\Phi|$ many times we reach a depth-4 formula where all polynomials computed at the bottom addition gate have small support. Next, we partition the variables to sets and consider a single set T_i. Now, another issue is that even if the intersection of a low-support polynomial with some T_i is rather small, the sparsity of the resulting polynomial (which is considered as a polynomial in the variable in the intersection) can still be exponential in the size of the intersection. This is why we lose a bit in the upper bound compared to the depth-3 case. Combining all steps again we get the result for depth-4 formulas.

The proof for regular formulas works by first reducing to the depth-4 case and then applying our hitting set. The reduction is obtained in a similar spirit to the reduction of Kayal et al. [21]. We break the formula at an appropriate layer and then express the top

layers as a $\Sigma\Pi$ circuit and the bottom layers as products of polynomials of not too high degree. We then use the trivial observation that if the degree of a polynomial is at most $n^{1-\epsilon}$ then its sparsity is at most $n^{n^{1-\epsilon}}$ and proceed as before. Due to the different requirements of the reduction and of the hashing part, we roughly lose a constant factor in the exponent of n, in the size of the hitting set, whenever the depth grows, resulting in a hitting set of size roughly $\exp(n^{1-1/\exp(d)})$.

To obtain the lower bounds we simply use the idea of [16, 1]. That is, given a hitting set we find a non-zero multilinear polynomial that vanishes on all points of the hitting set by solving a homogeneous system of linear equations.

1.5 Related Work

1.5.1 The work of Agrawal, Gurjar, Korwar and Saxena [2]

The closest work to ours is the one by Agrawal et al. [2]. In addition to other results, they gave a white-box PIT algorithm that runs in time $n^{O(2^c n^{1-\frac{2}{2^c}} \log s)}$ for depth-3 formulas that can be represented as a sum of c set-multilinear formulas, each of size s (potentially with respect to different partitions of the variables).

Theorem 1.1 improves upon this results in several ways. First, the theorem gives a hitting set, i.e., a black-box PIT. Secondly, for $c = O(\log \log n)$ the running time of the algorithm of [2] is $\exp(n)$, whereas our construction can handle a sum of $\exp(n^\beta)$ set-multilinear formulas and still maintain a subexponential complexity.

Nonetheless, there are some similarities behind the basic approach of this work and the work of Agrawal et al. Recall that a set-multilinear depth-3 formula is based on a partition of the variables, where each linear function in the formula contains variables from a single partition. Agrawal et al. start with a sum of c set-multilinear circuits, each with respect to a different partitioning of the variables, and their first goal is to reduce the formula to a set-multilinear formula, i.e., to have only one partition of the variables. For this they define a distance between different partitions and show, using an involved combinatorial argument, that one can find some partition $T_1, \ldots, T_m$ of the variables so that when restricting our attention to T_i, all the c set-multilinear formulas will be somewhat "close to each other". If the distance is Δ (according to their definition) then they prove that they can express the sum as a set-multilinear circuit of size roughly $s \cdot n^\Delta$, where s is the total size of the depth-3 formula. Unlike our work, they find the partition in a white-box manner by gradually refining the given c partitions of the set-multilinear circuits composing the formula. The final verification step is done, in a similar manner to ours, by substituting the hitting set of [5] (or that of [13]) to each of the sets T_i. The step of finding the partition $T_1, \ldots, T_m$ is technically involved and is the only step where white-box access is required.

1.6 Organization

In Section 2 we provide basic definitions and notations, and also state some general lemmas which will be helpful in the next sections. In Section 3, we explain how to reduce general depth-3 and depth-4 formulas to formulas such that every polynomial at the bottom has small support. Then, in Section 4, we construct a hitting set for those types of formulas. In Section 5, we explain how to combine the ideas of the previous two sections and construct our hitting set for depth-3 and depth-4 multilinear formulas.

We then move on to depth-d regular formulas in Section 6, and state our hitting set for this class. In the short Section 7 we spell out briefly how, using known observations about the

relation between PIT and lower bounds, we obtain our lower bounds for multilinear formulas. Finally, in Section 8 we discuss some open problems and future directions for research.

The proofs of the results regarding depth-4 and depth-d regular formulas are omitted from this version, and can be found in the full version of the paper ([27]).

2 Preliminaries

In this section, we establish notation, some definitions and useful lemmas that will be used throughout the paper.

2.1 Notations and Basic Definitions

For any positive integer n, we denote by $[n]$ the set of integers from 1 to n, and by $\binom{[n]}{\leq r}$ the family of subsets $A \subseteq [n]$ such that $|A| \leq r$. We often associate a subset $A \subseteq [n]$ with a subset of variables $\mathsf{var}(A) \subseteq \{x_1, \ldots, x_n\}$ in a natural way (i.e., $\mathsf{var}(A) = \{x_i \mid i \in A\}$). In those cases we make no distinction between the two and use A to refer to $\mathsf{var}(A)$. Additionally, if A and B are disjoint subsets of $[n]$, we denote their disjoint union by $A \sqcup B$. For a vector $v \in \mathbb{F}^n$ we denote with $v|_A$ the restriction of v to the coordinates A. In order to improve the readability of the text, we omit floor and ceiling notations.

Let $f(x_1, \ldots, x_n) \in \mathbb{F}[x_1, \ldots, x_n]$ be a polynomial. We will denote by $\partial_x f$ the formal derivative of f with respect to the variable x, and by $f|_{x=0}$ the polynomial obtained from f by setting $x = 0$. Moreover, if $A \subseteq [n]$, we will denote by $\partial_A f$ the polynomial obtained when taking the formal derivative of f with respect to all variables in A. In a similar fashion, we denote by $f|_{A=0}$ the polynomial obtained when we set all the variables in A to zero, and more generally, if $|A| = r$ and $\overline{\alpha} = (\alpha_1, \ldots, \alpha_r) \in \mathbb{F}^r$, $f|_{A=\overline{\alpha}}$ will denote the restriction of f obtained when setting the i'th variable in A to α_i, for $1 \leq i \leq r$.

In addition to the conventions above, the following definitions will be very useful in the next sections.

▶ **Definition 2.1** (Variable Set and Non-trivial Variable Set)**.** Let $f(x_1, \ldots, x_n) \in \mathbb{F}[x_1, \ldots, x_n]$ be a polynomial. Define the variable set (var) and the non-trivial variable set (var^*) as follows:

$$\mathsf{var}(f) = \{x \in \{x_1, \ldots, x_n\} \mid \partial_x f \neq 0\}$$
$$\mathsf{var}^*(f) = \{x \in \{x_1, \ldots, x_n\} \mid \partial_x f \neq 0 \text{ and } f|_{x=0} \neq 0\}.$$

That is, the variable set of a polynomial f is the set of variables $x \in \{x_1, \ldots, x_n\}$ that appear in the representation of f as a sum of monomials, whereas the non-trivial variable set is the set of variables of f that do not divide it.

We shall say that f has a small support if $\mathsf{var}(f)$ (or $\mathsf{var}^*(f)$) is not too large.

▶ **Definition 2.2** (Monomial Support and Sparsity)**.** Let $f(x_1, \ldots, x_n) \in \mathbb{F}[x_1, \ldots, x_n]$ be a polynomial. We define the *monomial support* of f, written $\mathsf{mon}(f)$, as the set of monomials that have a non-zero coefficient in f. In addition, we define the sparsity of f, written $\|f\|$, as the size of the set $\mathsf{mon}(f)$, that is,

$$\|f\| = |\mathsf{mon}(f)|.$$

In other words, the sparsity of f is the number of monomials that appear with a non-zero coefficient in f.

In the constructions of our hitting sets we will need to combine assignments to different subsets of variables. The following notation will be useful. For a partition of $[n]$, $T_1 \sqcup T_2 \sqcup \cdots \sqcup T_m = [n]$, and sets $\mathcal{H}_i \subseteq \mathbb{F}^{|T_i|}$, we denote with $\mathcal{H}_1^{T_1} \times \cdots \times \mathcal{H}_m^{T_m}$ the set of all vectors of length n whose restriction to T_i is an element of $\mathcal{H}_i$:

$$\mathcal{H}_1^{T_1} \times \cdots \times \mathcal{H}_m^{T_m} = \{v \in \mathbb{F}^n \mid \forall i \in [m],\ v|_{T_i} \in \mathcal{H}_i\}.$$

2.2 Depth-3 and Depth-4 Formulas

We define some special classes of depth-3 and depth-4 formulas that will be used throughout this paper.

▶ **Definition 2.3** (Restricted Top Fan-in)**.** Let Φ be a multilinear depth-4 formula. We say that Φ is a multilinear $\Sigma^{[M]}\Pi\Sigma\Pi$ formula if it is of the form $\sum_{i=1}^{m}\prod_{j=1}^{t_i} f_{i,j}$, where $m \leq M$. If, in addition to the conditions above, we have that each $f_{i,j}$ is a linear function, that is, Φ is actually a depth-3 formula, we will say that Φ is a multilinear $\Sigma^{[M]}\Pi\Sigma$ formula.

Our next definition considers the case where polynomials computed at the bottom layers do not contain too many variables, that is, they have small variable set.

▶ **Definition 2.4** (Restricted Top Fan-in and Variable Set)**.** Let Φ be a multilinear depth-4 formula. We say that Φ is a multilinear $\Sigma^{[M]}\Pi(\Sigma\Pi)^{\{\tau\}}$ formula if it is of the form $\sum_{i=1}^{m}\prod_{j=1}^{t_i} f_{i,j}$, where $m \leq M$ and for each $1 \leq i \leq m$ we have that

1. $|\mathsf{var}(f_{i,j})| \leq \tau$ for all $1 \leq j \leq t_i$
2. $\mathsf{var}(f_{i,j_1}) \cap \mathsf{var}(f_{i,j_2}) = \emptyset$, for any $j_1 \neq j_2$.

If, in addition to the conditions above, we have that each $f_{i,j}$ is a linear function, that is, Φ is actually a depth-3 formula, we will say that Φ is a multilinear $\Sigma^{[M]}\Pi\Sigma^{\{\tau\}}$ formula.

Since the formula will be given to us as a black-box, we can make some assumptions about it, which will help us to preserve non-zeroness when taking derivatives or setting variables to zero. To this end, we define a notion of simplicity of depth-4 formulas,[3] and prove that we can assume without loss of generality that any input formula is simple.

▶ **Definition 2.5.** Let $f(x_1, \ldots, x_n) \in \mathbb{F}[x_1, \ldots, x_n]$ be a multilinear polynomial and let

$$\Phi = \sum_{i=1}^{M}\prod_{j=1}^{t_i} f_{i,j}$$

be a multilinear depth-4 formula computing f. We say that Φ is a *simple* multilinear depth-4 formula if for each variable $x \in \mathsf{var}(f)$ that divides f, it must be the case that for every $1 \leq i \leq M$, there exists $j \in [t_i]$ such that $f_{i,j} = x$.

In words, Φ is simple if whenever a variable x divides f, it also divides every product gate. The following proposition tells us that we can indeed assume, without loss of generality, that any multilinear depth-4 formula given to us is a simple formula.

▶ **Proposition 2.6.** *If Φ is a depth-4 multilinear $\Sigma^{[M]}\Pi\Sigma\Pi$ formula computing $f(x_1, \ldots, x_n)$, then f can be computed by a simple depth-4 multilinear $\Sigma^{[M]}\Pi\Sigma\Pi$ formula Ψ where $|\Psi| \leq |\Phi|$.*

[3] Note that this is not the same notion as used, e.g., in [10].

As a corollary, together with the simple observation that any derivative or restriction of a multilinear formula results in a multilinear formula of at most the same size, we obtain that partial derivatives or restrictions of a multilinear polynomial can also be computed by simple formulas.

▶ **Corollary 2.7.** *If Φ is a depth-4 multilinear $\Sigma^{[M]}\Pi\Sigma\Pi$ formula computing $f(x_1, \ldots, x_n)$, then for any disjoint sets $A, B \subseteq \mathsf{var}(f)$, $\partial_A f|_{B=0}$ can be computed by a simple depth-4 multilinear $\Sigma^{[M]}\Pi\Sigma\Pi$ formula Ψ where $|\Psi| \leq |\Phi|$. We will refer to Ψ as $\partial_A \Phi|_{B=0}$.*

Therefore, from now on we will always assume that any depth-4 multilinear formula given to us is a simple formula.

2.3 ROABPs for Products of Sparse Polynomials

Another important model that we need for our constructions is that of Algebraic Branching Programs.

▶ **Definition 2.8** (Nisan [25])**.** An algebraic branching program (ABP) is a directed acyclic graph with one vertex s of in-degree zero (the *source*) and one vertex t of out-degree zero (the *sink*). The vertices of the graph are partitioned into levels labeled $0, 1, \ldots, D$. Edges in the graph can only go from level $\ell - 1$ to level ℓ, for $\ell \in [D]$. The source is the only vertex at level 0 and the sink is the only vertex at level D. Each edge is labeled with an affine function in the input variables. The *width* of an ABP is the maximum number of nodes in any layer, and the *size* of an ABP is the number of vertices in the ABP.

Each path from s to t computes the polynomial which is the product of the labels of the path edges, and the ABP computes the sum, over all $s \to t$ paths, of such polynomials.

▶ **Definition 2.9** (Ordered Read-Once Algebraic Branching Programs)**.** A *Ordered Read-Once Algebraic Branching Program (ROABP)* in the variable set $\{x_1, \ldots, x_D\}$ is an ABP of depth D, such that each edge between layer $\ell - 1$ and ℓ is labeled by a univariate polynomial in x_ℓ.

Below we show an elementary construction of ROABPs for a very specific class of polynomials, the proof of which can be found in [27].

▶ **Lemma 2.10.** *Let $\mathbb{F}$ be a field, and $f(y_1, \ldots, y_m) = \sum_{i=1}^{M} \prod_{j=1}^{t_i} f_{i,j}$ be a multivariate polynomial over $\mathbb{F}$, such that for every $1 \leq i \leq M$:*
1. *At most k different $1 \leq j \leq t_i$, satisfy $|\mathsf{var}(f_{i,j})| > 1$.*
2. *For every $1 \leq j \leq t_i$, $\|f_{i,j}\| \leq s$.*

Then f can be computed by an ROABP of width at most $M \cdot s^k$.

2.4 Hashing

In this section we present the basic hashing tools that we will use in our construction. We first recall the notion of a k-wise independent hash family.

▶ **Definition 2.11.** A family of hash functions $\mathcal{F} = \{h : [n] \to [m]\}$ is k-wise independent if for any k distinct elements $(a_1, \ldots, a_k) \in [n]^k$ and any k (not necessarily distinct) elements $(b_1, \ldots, b_k) \in [m]^k$, we have:

$$\Pr_{h \in \mathcal{F}}[h(a_1) = b_1 \wedge \cdots \wedge h(a_k) = b_k] = m^{-k}.$$

Our next lemma studies the case where several sets are hashed simultaneously by the same hash function. We present the lemma in a general form and only later, in the application, fix the parameters.

▶ **Lemma 2.12.** *Let $0 < \delta < \epsilon$, and $n, M \in \mathbb{N}$ such that $M = 2^{n^\delta}$. Assume $\mathcal{A}_1, \ldots, \mathcal{A}_M$ are families of pairwise disjoint subsets of $[n]$ such that for every $1 \leq i \leq M$ and every $A \in \mathcal{A}_i$, $|A| \leq n^{1-\epsilon}$. Let $\gamma > 0$ be such that $\gamma \geq (\epsilon - \delta)/2$. Let $\mathcal{F}$ be a family of k-wise independent hash functions from $[n]$ to $[m]$ for $k = n^\delta + 2\log n$ and $m = 10n^{1-\epsilon+\gamma}$.*

Then there exists $h \in \mathcal{F}$ such that for every $1 \leq i \leq M$ and every $1 \leq j \leq m$, both of the following conditions hold:

1. *For every set $A \in \mathcal{A}_i$, $\left|h^{-1}(j) \cap A\right| \leq k$.*
2. *The number of sets $A \in \mathcal{A}_i$ such that $\left|h^{-1}(j) \cap A\right| > 1$ is at most $k \log n$.*

We conclude this section with the following well known fact (see, e.g., Chapter 16 in [7], and the references therein):

▶ **Fact 2.13.** *There exists an explicitly constructible family $\mathcal{F}$ of k-wise independent hash functions from $[n]$ to $[10n^{1-\epsilon+\gamma}]$ of size $|\mathcal{F}| = n^{O(k)}$.*

3 Reducing the Bottom Variable Set of Depth-3 and Depth-4 Formulas

In this section we make the first step towards proving Theorems 1.1 and 1.3. As outlined in Section 1.4, our first step is making the functions computed at the bottom layers (linear functions in the case of depth-3 and "sparse" polynomials in the case of depth-4) have small variable set. Hence, we establish reductions from any $\Sigma^{[M]}\Pi\Sigma$ or $\Sigma^{[M]}\Pi\Sigma\Pi$ formula to a $\Sigma^{[M]}\Pi\Sigma^{\{\tau\}}$ or $\Sigma^{[M]}\Pi(\Sigma\Pi)^{\{\tau\}}$ formula, respectively. We first describe the simple depth-3 case. We then state without proofs the more general case of depth-4 formulas. The full proofs can be found in the full version ([27]).

3.1 Reducing Bottom Variable Set for Depth-3

Given a depth-3 formula $\sum_{i=1}^{M}\prod_{j=1}^{t_i}\ell_{i,j}$, we would like to eliminate all linear functions that contain many variables. To this end, we observe that there must exist a variable that appears in many of these functions, and that taking a derivative according to that variable eliminates those functions from the formula.

▶ **Lemma 3.1.** *Let $f(x_1, \ldots, x_n) = \sum_{i=1}^{M}\prod_{j=1}^{t_i}\ell_{i,j}$ be a non-zero multilinear polynomial computed by a multilinear $\Sigma^{[M]}\Pi\Sigma$ formula Φ and let $\epsilon > 0$. Then, there exists a set of variables A of size $|A| \leq \tilde{O}(n^\epsilon \cdot \log M)$ such that $\partial_A f$ is a non-zero multilinear polynomial that can be computed by a multilinear $\Sigma^{[M]}\Pi\Sigma^{\{n^{1-\epsilon}\}}$ formula.*

Proof. Define $\mathcal{B} = \{\ell_{i,j} \mid |\mathsf{var}(\ell_{i,j}) \cap \mathsf{var}(f)| \geq n^{1-\epsilon}\}$ to be the set of "bad" linear functions. Those are linear functions that contain more than $n^{1-\epsilon}$ variables that also appear in f. We show how to eliminate those linear functions from the formula while preserving non-zeroness.

Since for every $\ell \in \mathcal{B}$, $|\mathsf{var}(\ell) \cap \mathsf{var}(f)| \geq n^{1-\epsilon}$, there exists a variable x_i that appears in at least $|\mathcal{B}|n^{1-\epsilon}/n = |\mathcal{B}|/n^\epsilon$ linear functions in $\mathcal{B}$ (and also in f). The polynomial $\partial_{x_i} f$ is non-zero, since $x_i \in \mathsf{var}(f)$. Furthermore, using the fact that deriving with respect to a variable is a linear operation, and the fact that every multiplication gate in the formula multiplies linear functions with disjoint support, a formula for $\partial_{x_i} f$ can be obtained from Φ by replacing every linear function in which x_i appears with an appropriate constant. Therefore, every such function is removed from $\mathcal{B}$, and so the set of bad linear functions in $\partial_{x_i} f$ is of size at most $|\mathcal{B}| - |\mathcal{B}|/n^\epsilon = |\mathcal{B}| \cdot (1 - 1/n^\epsilon)$. We continue this process for at most $O(n^\epsilon \cdot \log|\mathcal{B}|)$ steps, until we reach a point where $|\mathcal{B}| < 1$ and so $|\mathcal{B}| = 0$.

Finally, it remains to be noted that $|\mathcal{B}| \leq Mn$, since by multilinearity each multiplication gate multiplies linear functions with disjoint support, and so its fan-in is at most n. ◀

The process for depth-4 formulas follows the same outline as the depth-3 case, but there are a few more details, which can be found in the full version [27]. The precise statement of the support reduction for depth-4 is as follows:

▶ **Lemma 3.2** (Reduction to Depth-4 with Small Bottom Variable Set). *Let Φ be a multilinear simple $\Sigma^{[M]}\Pi\Sigma\Pi$ formula computing a non-zero multilinear polynomial $f(x_1, \ldots, x_n) \in \mathbb{F}[x_1, \ldots, x_n]$. There exist disjoint sets $A, B \subset [n]$ with $|A \sqcup B| \leq \frac{2n}{\tau} \cdot \log(|\Phi|)$ such that the polynomial $\partial_A f|_{B=0}$ is non-zero and can be computed by a simple multilinear $\Sigma^{[M]}\Pi(\Sigma\Pi)^{\{\tau\}}$ formula Ψ.*

4 Hitting Set for $\Sigma\Pi\Sigma^{\{n^{1-\epsilon}\}}$ and $\Sigma\Pi(\Sigma\Pi)^{\{n^{1-\epsilon}\}}$ Formulas

In this section we construct subexponential sized hitting set for the classes of $\Sigma^{[M]}\Pi\Sigma^{\{n^{1-\epsilon}\}}$ and $\Sigma^{[M]}\Pi(\Sigma\Pi)^{\{n^{1-\epsilon}\}}$ multilinear formulas. Recall that in Section 3 we showed how to reduce general depth-3 and depth-4 formulas to these types of formulas. In the next section, we will show how to tie all loose edges and combine the arguments of Section 3 with those of this section in order to handle the general case.

An essential ingredient in our construction is a quasi-polynomial sized hitting set for Read-Once Algebraic Branching Programs (ROABPs) [14, 2]. We note that in our setting, we may assume that the reading order of the variables by the ABP is known.

▶ **Theorem 4.1** ([14, 2]). *Let $\mathcal{C}$ be the class of n-variate polynomials computed by a ROABP of width w, such that the degree of each variable is at most d, over a field $\mathbb{F}$ so that $|\mathbb{F}| \geq \mathsf{poly}(n, w, d)$. Then $\mathcal{C}$ has a hitting set of size $\mathsf{poly}(n, w, d)^{\log n}$ that can be constructed in time $\mathsf{poly}(n, w, d)^{\log n}$.*

We begin by describing a unified construction for both $\Sigma^{[M]}\Pi\Sigma^{\{n^{1-\epsilon}\}}$ and $\Sigma^{[M]}\Pi(\Sigma\Pi)^{\{n^{1-\epsilon}\}}$ formulas. We then describe how to set the parameters of the construction for each of the cases.

▶ **Construction 4.2** (Hitting set for multilinear $\Sigma^{[M]}\Pi\Sigma^{\{n^{1-\epsilon}\}}$ and $\Sigma^{[M]}\Pi(\Sigma\Pi)^{\{n^{1-\epsilon}\}}$ formulas). *Let $0 < \delta < \epsilon$ and n, k, s, M integers, such that $M = 2^{n^\delta}$ and $k = n^\delta + 2\log n$. Set $m = 10n^{1-(\epsilon+\delta)/2}$ and $t = k \log n$. Let $\mathcal{F}$ be a family of k-wise independent hash functions from $[n]$ to $[m]$, as in Lemma 2.12. For every $h \in \mathcal{F}$, define the set I_h as follows:*

1. *Partition the variables to sets[4] $T_1 \sqcup T_2 \sqcup \cdots \sqcup T_m = h^{-1}(1) \sqcup h^{-1}(2) \sqcup \cdots \sqcup h^{-1}(m)$.*
2. *For every $1 \leq i \leq m$, let $\mathcal{H}_i$ be a hitting set for ROABPs of width $M \cdot s^t$ and individual degree $d = 1$ (as promised by Theorem 4.1), on the variables of T_i (of course, $|T_i| \leq n$).*
3. *We define I_h as the set of all vectors v such that the restriction of v to the coordinates T_i, $v|_{T_i}$, is in $\mathcal{H}_i$. I.e., in the notation of Section 2.1,*

$$I_h = \mathcal{H}_1^{T_1} \times \mathcal{H}_2^{T_2} \times \cdots \times \mathcal{H}_m^{T_m}.$$

Finally, define $\mathcal{H} = \bigcup_{h \in \mathcal{F}} I_h$.

The following lemma gives an upper bound to the size of the hitting set constructed in Construction 4.2.

[4] Recall that we associate subsets of $[n]$ with subsets of the variables, and make no distinction in the notation.

▶ **Lemma 4.3.** *Let $\delta, \epsilon, k, n, s$ and M be the parameters of Construction 4.2. The set $\mathcal{H}$ constructed in Construction 4.2 has size $n^{O(k)} \cdot \left(M \cdot s^{k \log n}\right)^{\tilde{O}(n^{1-(\epsilon+\delta)/2})} = \left(M \cdot s^{k \log n}\right)^{\tilde{O}(n^{1-(\epsilon+\delta)/2})}$, and it can be constructed in time $\mathsf{poly}(|\mathcal{H}|)$.*

Proof. This is a direct consequence of the construction, Fact 2.13 and Theorem 4.1. ◀

4.1 Depth-3 Formulas

We begin by describing the argument for depth-3 formulas. The following lemma proves that indeed, by setting the proper parameters, the set $\mathcal{H}$ from Construction 4.2 does hit $\Sigma^{[M]}\Pi\Sigma^{\{n^{1-\epsilon}\}}$ formulas.

▶ **Lemma 4.4.** *Let $f(x_1, \ldots, x_n) \in \mathbb{F}[x_1, \ldots, x_n]$ be a multilinear polynomial computed by a multilinear $\Sigma^{[M]}\Pi\Sigma^{\{n^{1-\epsilon}\}}$ formula $\Phi = \sum_{i=1}^{M} \prod_{j=1}^{t_i} \ell_{i,j}$. Let $\mathcal{H}$ be the set constructed in Construction 4.2 with $s = k + 1$. Then there exists a point $\overline{\alpha} \in \mathcal{H}$ such that $f(\overline{\alpha}) \neq 0$.*

Proof. For every multiplication gate $1 \leq i \leq M$ in Φ, define a partition of the variables

$$\mathcal{A}_i = \{\mathsf{var}(\ell_{i,j}) \cap \mathsf{var}(f) \mid 1 \leq j \leq t_i\}.$$

Let $h \in \mathcal{F}$ be the function guaranteed by Lemma 2.12 with respect to the partitions $\mathcal{A}_1, \ldots \mathcal{A}_M$, and assume the setup of Construction 4.2. We claim that there exists $\overline{\alpha} \in I_h$ such that $f(\overline{\alpha}) \neq 0$.

To that end, consider the partition of the variables induced by h:

$$T_1 \sqcup T_2 \sqcup \cdots \sqcup T_m = h^{-1}(1) \sqcup h^{-1}(2) \sqcup \cdots \sqcup h^{-1}(m).$$

We view the polynomial as a polynomial f_1 in the variables of T_1, over the extension field $\mathbb{F}(T_2 \sqcup \cdots \sqcup T_n)$. We claim that f_1 can be computed by an ROABP of width $M \cdot (k+1)^{k \log n}$. To see this note that, by Lemma 2.12, in any multiplication gate, at most $k \log n$ linear functions contain more than one variable from T_1, and each contains at most k variables. It follows that the sparsity of every linear function (with respect to the variables in T_1) among those $k \log n$ functions, is at most $k+1$. By Lemma 2.10, f_1 can be computed by an ROABP over $\mathbb{F}(T_2 \sqcup \cdots \sqcup T_n)$ of width $M \cdot (k+1)^{k \log n}$. By the hitting set property of Theorem 4.1, there exists $\overline{\alpha_1} \in \mathcal{H}_1 \subseteq \mathbb{F}^{|T_1|}$ such that $f_2 \stackrel{\text{def}}{=} f_1|_{T_1 = \overline{\alpha_1}} \not\equiv 0$.

Similarly, the same conditions now hold for f_2, considered as a polynomial over the field $\mathbb{F}(T_3 \sqcup \cdots \sqcup T_n)$, and so there exists $\overline{\alpha_2} \in \mathcal{H}_2 \subseteq \mathbb{F}^{|T_2|}$ such that $f_3 \stackrel{\text{def}}{=} f_2|_{T_2 = \overline{\alpha_2}} \not\equiv 0$.

We continue this process for m steps, and at the last step we find $\overline{\alpha_m}$ such that $f_{m-1}(\overline{\alpha_m}) = f(\overline{\alpha_1}, \cdots, \overline{\alpha_m}) \neq 0$, with $(\overline{\alpha_1}, \cdots, \overline{\alpha_m}) \in \mathbb{F}^n$ being the length n vector obtained by filling the entires of $\overline{\alpha_i} \in \mathbb{F}^{|T_i|}$ in the positions indexed by T_i. ◀

4.2 Depth-4 Formulas

The argument for depth-4 formulas is very similar, apart from a small change in the setting of the parameters. Once again, the proof is omitted.

▶ **Lemma 4.5.** *Let $f(x_1, \ldots, x_n) \in \mathbb{F}[x_1, \ldots, x_n]$ be a multilinear polynomial computed by a multilinear $\Sigma^{[M]}\Pi(\Sigma\Pi)^{\{n^{1-\epsilon}\}}$ formula $\Phi = \sum_{i=1}^{M} \prod_{j=1}^{t_i} f_{i,j}$. Let $\mathcal{H}$ be the set constructed in Lemma 4.2 with $s = 2^k$. Then, there exists a point $\overline{\alpha} \in \mathcal{H}$ such that $f(\overline{\alpha}) \neq 0$.*

5 Hitting Set for Depth-3 and Depth-4 Multilinear Formulas

Recall that, in Section 3, we showed that any non-zero $\Sigma^{[M]}\Pi\Sigma$ or $\Sigma^{[M]}\Pi\Sigma\Pi$ formula has a non-zero partial derivative (and, possibly, a restriction) which is computed by a non-zero $\Sigma^{[M]}\Pi\Sigma^{\{n^{1-\epsilon}\}}$ or $\Sigma^{[M]}\Pi(\Sigma\Pi)^{\{n^{1-\epsilon}\}}$ formula, respectively. Then, in Section 4 we gave hitting sets for such formulas. In this section we provide the final ingredient, which is to show how to "lift" those hitting sets back to the general class, via a simple method, albeit one that requires some notation.

Handling restrictions is fairly easy, and causes no blow up in the hitting set size: If we have a set $\mathcal{H} \subseteq \mathbb{F}^{n-r}$ that hits $f|_{B=0}$ for some multilinear polynomial $f(x_1,\ldots,x_n)$ and $B \subseteq [n]$ with $|B| = r$, then simply extending $\mathcal{H}$ into a subset of $\mathbb{F}^n$ by filling out zeros in all the entries indexed by B will hit f itself.

In order to handle partial derivates, first note that if $f(x_1,\ldots,x_n)$ is a multilinear polynomial, then

$$\partial_{x_i} f = f(x_1,\ldots,x_{i-1},1,x_{i+1},\ldots,x_n) - f(x_1,\ldots,x_{i-1},0,x_{i+1},\ldots,x_n),$$

and so if $\partial_{x_i} f(\overline{\alpha}) \neq 0$ for some $\overline{\alpha} \in \mathbb{F}^n$ then at least one of the two evaluations on the right hand side must be non-zero as well.

Applying this fact repeatedly, given a set $A \subseteq [n]$ we can evaluate $\partial_A f$ at any point by making $2^{|A|}$ evaluations of f. Motivated by this, we introduce the following notation:

▶ **Definition 5.1.** Let $f(x_1,\ldots,x_n) \in \mathbb{F}[x_1,\ldots,x_n]$ be a multilinear polynomial and $A, B \subseteq [n]$ be two disjoint subsets of variables with $|A| = r, |B| = r'$. Let $\mathcal{H} \subseteq \mathbb{F}^{n-(r+r')}$.

We define the "lift" of $\mathcal{H}$ with respect to (A,B) to be

$$\mathcal{L}_A^B(\mathcal{H}) = \left(\{0,1\}^r\right)^A \times \left(\{0\}^{r'}\right)^B \times \mathcal{H}^{[n]\setminus(A\sqcup B)}.$$

In the special case where $B = \emptyset$, we simply denote $\mathcal{L}_A^B(\mathcal{H}) = \mathcal{L}_A(\mathcal{H})$.

That is, for all $\overline{\alpha} \in \mathcal{H}$, $\mathcal{L}_A^B(\mathcal{H})$ contains all the possible 2^r ways to extend $\overline{\alpha}$ into $\overline{\beta} \in \mathbb{F}^n$ by filling out zeros and ones within the r entries that are indexed by A, and zeros in all the r' entries indexed by B.

5.1 Depth-3 Formulas

In this section we prove Theorem 1.1. For the reader's convenience, we first restate the theorem:

▶ **Theorem 5.2** (Theorem 1.1, restated)**.** *Let $\mathcal{C}$ be the class of multilinear $\Sigma^{[M]}\Pi\Sigma$ formulas for $M = 2^{n^\delta}$. There exists a hitting set $\mathcal{H}$ of size $|\mathcal{H}| = 2^{\tilde{O}(n^{2/3+2\delta/3})}$ for $\mathcal{C}$, that can be constructed in time* $\mathsf{poly}(|\mathcal{H}|)$.

The size of the hitting set is subexponential for any constant $\delta < 1/2$. Also, if $M = \mathsf{poly}(n)$ then the size of the hitting set is $2^{\tilde{O}(n^{2/3})}$.

With Definition 5.1 in hand, we now provide our construction for $\Sigma^{[M]}\Pi\Sigma$ formulas, towards the proof of Theorem 5.2.

▶ **Construction 5.3** (Hitting set for multilinear $\Sigma^{[M]}\Pi\Sigma$ formulas)**.** *Let $M = 2^{n^\delta}$ and $\epsilon = 2/3 - \delta/3$. Let $r = \tilde{O}(n^\epsilon \log M) = \tilde{O}(n^{\frac{2}{3}+\frac{2}{3}\delta})$ as guaranteed by Lemma 3.1. For every $A \in \binom{[n]}{\leq r}$,*

construct a set $\mathcal{H}_A \in \mathbb{F}^{n-|A|}$ *using Construction 4.2 with parameters* $\delta, \epsilon, n, k, s = k+1$ *and* M *(recall that in Construction 5.3 we set* $k = n^\delta + 2\log n$*). Finally, let*

$$\mathcal{H} = \bigcup_{A \in \binom{[n]}{\leq r}} \mathcal{L}_A(\mathcal{H}_A).$$

We are now ready to prove Theorem 5.2:

Proof of Theorem 5.2. We show that the set $\mathcal{H}$ constructed in Construction 5.3 has the desired properties. First, note that by Lemma 4.3, for every $A \subseteq [n]$ with

$$|A| \leq \tilde{O}(n^\epsilon \log M) = \tilde{O}(n^{2/3-\delta/3} \log M) = \tilde{O}(n^{\frac{2}{3}+\frac{2}{3}\delta}),$$

the set $\mathcal{H}_A$ has size

$$(M \cdot (k+1)^{k \log n})^{\tilde{O}(n^{2/3-\delta/3})} = 2^{\tilde{O}(n^{2/3+2\delta/3})},$$

where we have used the fact that, in Construction 5.3, we take $k = n^\delta + 2\log n$. It therefore follows that

$$|\mathcal{L}_A(\mathcal{H}_A)| \leq 2^{|A|} \cdot |\mathcal{H}_A| = 2^{\tilde{O}(n^{2/3+2\delta/3})},$$

and that

$$|\mathcal{H}| \leq \sum_{i=0}^{\tilde{O}(n^{\frac{2}{3}+\frac{2}{3}\delta})} \sum_{A \subseteq [n], |A|=i} |\mathcal{L}_A(\mathcal{H}_A)| = 2^{\tilde{O}(n^{2/3+2\delta/3})}.$$

To show the hitting property of $\mathcal{H}$, let $f(x_1, \ldots, x_n)$ be a non-zero multilinear polynomial computed by a $\Sigma^{[M]}\Pi\Sigma$ formula, and let $A' \subseteq [n]$ be the set guaranteed by Lemma 3.1. Thus, $|A'| \leq \tilde{O}(n^\epsilon \log M) = \tilde{O}(n^{\frac{2}{3}+\frac{2}{3}\delta})$. Then by Lemma 4.4, there exists $\overline{\alpha} \in \mathcal{H}_{A'}$ such that $\partial_{A'} f(\overline{\alpha}) \neq 0$, and so there must exist

$$\overline{\beta} \in \mathcal{L}_{A'}(\mathcal{H}_{A'}) \subseteq \mathcal{H}$$

such that $f(\overline{\beta}) \neq 0$. ◀

In the depth-4 case, the construction and proof are both very similar, with a slight change in the parameters. For the detailed statements, proofs and construction of the hitting set, we refer the reader to the full version [27].

6 Multilinear Depth-*d* Regular Formulas

In this section, we consider *multilinear regular formulas*, which are regular formulas with the extra condition that each gate computes a multilinear polynomial. However, we will remove the bound on the formal degree of the formula. More precisely, we have the following definition:

▶ **Definition 6.1** (Multilinear Regular Formulas)**.** We say that a formula Φ is a multilinear $(a_1, p_1, a_2, p_2, \ldots, a_d, p_d, a_{d+1})$-regular formula computing a multilinear polynomial $f(x_1, \ldots, x_n)$ if it can be computed by a multilinear $\Sigma^{[a_1]}\Pi^{[p_1]}\Sigma^{[a_2]}\Pi^{[p_2]} \ldots \Sigma^{[a_d]}\Pi^{[p_d]}\Sigma^{[a_{d+1}]}$-formula. Notice that the size of such a formula is $(\prod_{1 \leq i \leq d+1} a_i) \cdot (\prod_{1 \leq i \leq d} p_i)$ and the formal degree of such a formula is given by $\deg(\Phi) = \prod_{1 \leq i \leq d} p_i$. Since the formula is multilinear, we have that $\deg(\Phi) \leq n$.

Comparing with the definition given in Section 1.1, an $(a_1, p_1, a_2, p_2, \ldots, a_d, p_d, a_{d+1})$-regular formula has depth $2d+1$.

Our main result for multilinear regular formulas is given by the following theorem, a proof of which can be found in the full version of this paper [27].

▶ **Theorem 6.2** (Theorem 1.5, restated). *For $d \geq 2$, let $\mathcal{C}_d$ be the class of multilinear polynomials computed by $(a_1, p_1, a_2, p_2, \ldots, a_d, p_d, a_{d+1})$-regular formulas of size $S \leq 2^{n^\delta}$ computing a multilinear polynomial $f(x_1, \ldots, x_n)$, where $\delta = \frac{1}{5^{d+1}}$. Then, there exists a hitting set $\mathcal{H}_d$ of size $|\mathcal{H}_d| = 2^{\tilde{O}(n^{1-\delta/3})}$ for $\mathcal{C}_d$, that can be constructed in time* $\mathsf{poly}(|\mathcal{H}_d|)$.

7 Lower Bounds for Bounded Depth Multilinear Formulas

As we noted earlier, the connection between construction of hitting sets and lower bounds for explicit polynomials is well established. The following theorem was proved by Heintz and Schnorr [16] and Agrawal [1], albeit we cite only a special case which matches our use of it:

▶ **Theorem 7.1** (A special case of [16, 1]). *Suppose there is a black-box deterministic PIT algorithm for a class $\mathcal{C}$ of multilinear circuits, that outputs a hitting set $\mathcal{H}$ of size $|\mathcal{H}| = 2^{n^\alpha} < 2^n$ and runs in time* $\mathsf{poly}(|\mathcal{H}|)$*, such that $\mathcal{H}$ hits circuits of size at most 2^{n^δ}. Then, there exists a multilinear polynomial $f(x_1, \ldots, x_n)$ such that any circuit from the class $\mathcal{C}$ computing f must be of size at least 2^{n^δ}, and the coefficients of f can be found in time $2^{O(n)}$.*

Theorem 7.1 is proved by finding a non-zero polynomial $f(x_1, \ldots, x_n)$ which vanishes on the entire hitting set $\mathcal{H}$ of size 2^{n^α}, and then, by definition, f cannot have circuits of size 2^{n^δ}. Finding f amounts to finding a non-zero solution to a homogenous system of linear equations whose unknowns are the coefficients of the 2^n possible multilinear monomials in $x_1, \ldots, x_n$. As long as $2^n > |\mathcal{H}| = 2^{n^\alpha}$, a non-zero solution is guaranteed to exist.

Our lower bounds now follow as a direct application of our hitting set constructions and Theorem 7.1.

Proofs of Corollaries 1.2, 1.4 and 1.6. In light of Theorem 7.1, we only need to pick δ so that the hitting sets we constructed have size less than 2^n. The appropriate choices, by Theorems 1.1, 1.3 and 1.5, respectively, can be seen to be $\delta = 1/2 - O(\log\log n/\log n)$ (for depth-3), $\delta = 1/4 - O(\log\log n/\log n)$ (for depth-4) and $\delta = \frac{1}{5^{\lfloor d/2 \rfloor + 1}} = O\left(\frac{1}{\sqrt{5}^d}\right)$ (for depth-d regular formulas). ◀

8 Conclusion and Open Questions

We conclude this paper with some obvious open problems. First, as noted in Section 1.3, the lower bounds that we get from our hitting sets are not as good as the best lower bounds for these models. Can one improve our construction to yield lower bounds matching the best known lower bounds?

Currently, the size of the hitting set that we get for depth-d regular multilinear formulas is roughly $\exp(n^{1-1/\exp(d)})$. Can the bound be improved to $\exp(n^{1-\Omega(1/d)})$? Finally, another natural question is to extend our argument from depth-d regular multilinear formulas to arbitrary depth-d multilinear formulas.

Acknowledgements. The authors would like to thank Zeev Dvir and Avi Wigderson for helpful discussions during the course of this work.

References

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

2 M. Agrawal, R. Gurjar, A. Korwar, and N. Saxena. Hitting-sets for ROABP and sum of set-multilinear circuits. *Electronic Colloquium on Computational Complexity (ECCC)*, 21:85, 2014.

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

4 M. Agrawal, C. Saha, R. Saptharishi, and N. Saxena. Jacobian hits circuits: hitting-sets, lower bounds for depth-d occur-k formulas & depth-3 transcendence degree-k circuits. In *STOC*, pages 599–614, 2012.

5 M. Agrawal, C. Saha, and N. Saxena. Quasi-polynomial hitting-set for set-depth-Δ formulas. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 321–330, 2013.

6 M. Agrawal and V. Vinay. Arithmetic circuits: A chasm at depth four. In *Proceedings of the 49th Annual FOCS*, pages 67–75, 2008.

7 N. Alon and J. H. Spencer. *The probabilistic method*. J. Wiley, 3 edition, 2008.

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

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

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

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

12 M. A. Forbes, R. Saptharishi, and A. Shpilka. Hitting sets for multilinear read-once algebraic branching programs, in any order. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 – June 03, 2014*, pages 867–875, 2014.

13 M. A. Forbes and A. Shpilka. On identity testing of tensors, low-rank recovery and compressed sensing. In *Proceedings of the 44th annual STOC*, pages 163–172, 2012.

14 M. A. Forbes and A. Shpilka. Quasipolynomial-time identity testing of non-commutative and read-once oblivious algebraic branching programs. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 243–252, 2013.

15 A. Gupta, P. Kamath, N. Kayal, and R. Saptharishi. Arithmetic circuits: A chasm at depth three. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013*, pages 578–587, 2013.

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

17 V. Kabanets and R. Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. In *Proceedings of the 35th Annual STOC*, pages 355–364, 2003.

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

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

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

21 N. Kayal, C. Saha, and R. Saptharishi. A super-polynomial lower bound for regular arithmetic formulas. In *Symposium on Theory of Computing, STOC 2014*, pages 146–153, 2014.

22 N. Kayal and S. Saraf. Blackbox polynomial identity testing for depth 3 circuits. In *Proceedings of the 50th Annual FOCS*, pages 198–207, 2009.

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

24 P. Koiran. Arithmetic circuits: the chasm at depth four gets wider. *CoRR*, abs/1006.4700, 2010.

25 N. Nisan. Lower bounds for non-commutative computation. In *Proceedings of the 23rd Annual STOC*, pages 410–418, 1991.

26 N. Nisan and A. Wigderson. Lower bound on arithmetic circuits via partial derivatives. *Computational Complexity*, 6:217–234, 1996.

27 R. Oliveira, A. Shpilka, and B. L. Volk. Subexponential size hitting sets for bounded depth multilinear formulas. *Electronic Colloquium on Computational Complexity (ECCC)*, 21:157, 2014.

28 R. Raz and A. Shpilka. Deterministic polynomial identity testing in non-commutative models. *Computational Complexity*, 14(1):1–19, 2005.

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

30 S. Saraf and I. Volkovich. Black-box identity testing of depth-4 multilinear circuits. In *Proceedings of the 43rd annual STOC*, pages 421–430, 2011.

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

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

33 A. Shpilka and I. Volkovich. Read-once polynomial identity testing. In *Proceedings of the 40th Annual STOC*, pages 507–516, 2008.

34 A. Shpilka and I. Volkovich. Improved polynomial identity testing for read-once formulas. In *APPROX-RANDOM*, pages 700–713, 2009.

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

36 S. Tavenas. Improved bounds for reduction to depth 4 and depth 3. In *Mathematical Foundations of Computer Science 2013 – 38th International Symposium, MFCS 2013*, pages 813–824, 2013.

37 L. G. Valiant, S. Skyum, S. Berkowitz, and C. Rackoff. Fast parallel computation of polynomials using few processors. *SIAM J. on Computing*, 12(4):641–644, November 1983.

38 R. Zippel. Probabilistic algorithms for sparse polynomials. In *EUROSAM*, pages 216–226, 1979.