Electronic Colloquium on Computational Complexity, Report No. 32 (2023)

# Linear Independence, Alternants and Applications

Vishwas Bhargava* Shubhangi Saraf† Ilya Volkovich ‡

**Abstract**

We develop a new technique for analyzing linear independence of multivariate polynomials. One of our main technical contributions is a *Small Witness for Linear Independence* (SWLI) lemma which states the following. If the polynomials $f_1, f_2, \ldots, f_k \in \mathbb{F}[X]$ over $X = \{x_1, \ldots, x_n\}$ are $\mathbb{F}$-linearly independent then there exists a subset $S \subseteq X$ of size at most $k-1$ such that $f_1, f_2, \ldots, f_k$ are also $\mathbb{F}(X \setminus S)$-linearly independent.

We show how to effectively combine this lemma with the use of the *alternant* matrix to analyze linear independence of polynomials. We also give applications of our technique to the questions of polynomial identity testing and arithmetic circuit reconstruction.

1. We give a general technique for lifting efficient polynomial identity testing algorithms from basic classes of circuits, satisfying some closure properties, to more general classes of circuits. As one of the corollaries of this result, we obtain the first algorithm for polynomial identity testing for depth-4, constant-occur circuits that works over **all** fields. This strengthens a result by [ASSS16] (*STOC '12*) that works in the case when the characteristic is 0 or sufficiently large. Another corollary is an identity testing algorithm for a special case of depth-5 circuits. To the best of our knowledge, this is the first algorithm for this class of circuits.
2. We give new and efficient black-box reconstruction algorithms for the class of set-multilinear depth-3 circuits of constant top fan-in, where the set-multilinear variable partition is *unknown*. This generalizes the results of [BSV21] (*STOC '21*) and [PSV22] (*ECCC '22*) which work in the case of known variable partition, and correspond to tensor decomposition of constant-rank tensors.

*David R. Cheriton School of Computer Science, Univeristy of Waterloo, Waterloo, Canada. Email: vishwas1384@gmail.com.

†Department of Mathematics & Department of Computer Science, University of Toronto, Toronto, Canada. Research partially supported by a Sloan research fellowship and an NSERC Discovery Grant. Email: shubhangi.saraf@gmail.com.

‡Department of Computer Science, Boston College, Chestnut Hill, MA 02467. Email: ilya.volkovich@bc.edu.

ISSN 1433-8092

# 1 Introduction

*Arithmetic circuits* are directed acyclic graphs (DAG) computing multivariate polynomials succinctly, building up from variables using $(+)$ addition and $(\times)$ multiplication operations. Two central algorithmic questions in the study of algebraic circuit complexity are those of polynomial identity testing (PIT) and arithmetic circuit reconstruction.

*Polynomial Identity Testing* of arithmetic circuits is the following problem: Given an arithmetic circuit $C$ over a field $\mathbb{F}$ with input variables $x_1, x_2, \cdots, x_n$, can we check efficiently whether $C$ computes the identically zero polynomial in the polynomial ring $\mathbb{F}[x_1, x_2, \cdots, x_n]$? The same question can be asked in the *black-box* setting. There, $C$ is accessed by via a black-box (i.e. oracle) where we are allowed to substitute field elements $a_i \in \mathbb{F}$ for $x_i$ and the black-box returns the value of $C(a_1, a_2, \cdots, a_n)$. A simple randomized polynomial-time algorithm for this problem is known due to the Schwartz-Zippel Lemma [Sch80, Zip79]. However *deterministic* polynomial-time (or at least subexponential-time) algorithms for PIT are believed to be quite challenging to obtain and are intimately connected with the question of obtaining lower bounds for general arithmetic circuits [KI04, HS80].

*Reconstruction* of arithmetic circuits is the following problem: given black-box (a.k.a. oracle/ membership query) access to a polynomial computed by a circuit $C$ of size $s$ from some class of circuits $\mathcal{C}$, give an efficient algorithm (deterministic or randomized) for recovering $C$ or some circuit $C'$ that computes the same polynomial as $C$. This problem is the algebraic analogue of exact learning in Boolean circuit complexity [Ang88]. If one additionally requires that the output circuit belongs to the same class $\mathcal{C}$ as the input circuit, then it is called *proper learning*.

Reconstruction of arithmetic circuits is an extremely natural problem, but also a really hard one. Just like PIT, much attention has focused on reconstruction algorithms for various interesting subclasses of arithmetic circuits [BBB+00, KS01, KS06, FS12]. In particular, much attention has focused on depth-3 and depth-4 arithmetic circuits [KS09a, GKL12, Sin16, BSV20, Sin20, BSV21].

Given the depth reduction results of [AV08, Koi10, Tav13, GKKS13], we know that depth-3 and depth-4 arithmetic circuits are very expressive, and good enough PIT or reconstruction algorithms for these models would have major implications for general circuits. Thus perhaps not surprisingly, we are quite far from obtaining efficient reconstruction or PIT algorithms even for depth-3 circuits.

## 1.1 Our Methods: Algebraic Independence vs Linear Independence

One of the frontiers of our understanding of PIT algorithms is the work of Agrawal et al. [ASSS16] which used the notion of *algebraic independence* to simultaneously unify and strengthen a wide variety of known PIT results for different classes of bounded-depth circuits that had previously been analyzed using a diverse set of techniques [GKPS11, BMS13, AvMV15, SV18]. Over fields of large characteristic, the notion of *algebraic independence* of a collection of polynomials is captured by the Jacobian matrix (which is full rank if the polynomials are algebraically independent). In several settings for which efficient deterministic PIT algorithms were known, the underlying circuit class has some bounded parameters - bounded read, bounded transcendence degree, bounded top fan-in etc. In these situations, the [ASSS16] work shows how to construct a "bounded" Jacobian such that constructing a hitting set for the Jacobian would suffice for constructing a hitting set for the underlying circuit class. It then analyzed properties of the Jacobian to construct the hitting set.

One drawback of Jacobian-based algorithms is that they involve taking partial derivatives and they work under the assumption that the field characteristic is zero or sufficiently large. In fact the Jacobian criterion fails over low characteristic fields. For instance, the polynomials $x^{p-1}y, y^{p-1}x$ are algebraically independent over $\mathbb{F}_p$ but the Jacobian is not full-rank over $\mathbb{F}_p$.

Our main contribution is to introduce and analyze another technique, very analogous to the Jacobian, which instead captures *linear independence* of polynomials. Using this method, we are able to recover some of the results that the Jacobian method was able to obtain. We are also able to obtain efficient PIT algorithms for some other general classes of circuits for which, to the best of our knowledge, no such algorithms were known prior to this work. Unlike the Jacobian method, our technique does not employ partial derivatives

and works over **all** fields. Moreover, linear independence is a simpler notion than algebraic independence, and hence the overall proof feels conceptually simpler.

We also demonstrate the potential of our technique with an application to arithmetic circuit reconstruction and show how to obtain the first randomized polynomial-time algorithm for reconstruction of set-multilinear $\Sigma\Pi\Sigma(k)$ circuits of *unknown* variable partition.

**Small Witness for Linear Independence (SWLI) Lemma:** The starting point for our method for analyzing linear independence of a collection of polynomials is what we call the *SWLI Lemma*. We prove that if $k$ multivariate polynomials in $\mathbb{F}[x_1, \ldots, x_n]$ are linearly independent over the base field $\mathbb{F}$, then there exists a subset $S \subseteq \{x_1, \ldots, x_n\}$ of variables such that the polynomials remain linearly independent over the extension field $\mathbb{F}(\bar{S})$. Here, $\bar{S}$ is the complement of $S$. That is, if we keep just those $k-1$ variables "alive" and add the other variables to the extension field, then the polynomials continue to be linearly independent over the extension field. For a formal statement, see Lemma 4.3.

Our results on PIT are obtained by combining the SWLI lemma with the notion of an *alternant* of a collection of polynomials. The alternant is a well-known tool to check the linear independence of a collection of functions $f_1, f_2, \ldots, f_k$ in a function space, see [Ait17, Chapter 6].

Roughly, the alternant matrix for $f_1, f_2, \ldots, f_k$ is a $k \times k$ matrix where each row is an independent evaluation of the vector $\langle f_1, f_2, \ldots, f_k \rangle$. The alternant is full rank if and only if $f_1, f_2, \ldots, f_k$ are linearly independent. Combining the SWLI Lemma along with alternants allows us to obtain hitting sets in a manner similar to the way the Jacobian was used in [ASSS16]. Our results on reconstruction are obtained by combining the SWLI lemma with a notion of generalized alternants.

## 1.2 Our Results

In this paper, we develop methods for analyzing linear independence of multivariate polynomials and give applications to polynomial identity testing and reconstruction that we describe below.

Let $\mathcal{C}$ be an arbitrary circuit class. We define two parametric families of associated circuit classes. For $k, \ell \in \mathbb{N}$, we define the class of $\Sigma^{[k]}\Pi^{[\ell]}\mathcal{C}$ *formulas* as the class of polynomials $f(\bar{x})$ that can be expressed as:

$$f(\bar{x}) = \sum_{i=1}^{k} \prod_{j=1}^{\ell} C_{ij}$$

where $C_{ij} \in \mathcal{C}$.

Note that $k$ and $\ell$ are upper bounds since some $C_{ij}$ can be taken to be field constants. That is, $\Sigma^{[k]}\Pi^{[\ell]}\mathcal{C}$ corresponds to taking a sum of (at most) $k$ products of (at most) $\ell$ polynomials from $\mathcal{C}$.

Next, we define the class *occur-k* $\Sigma\Pi\mathcal{C}$ *formulas* as the class of all the polynomials of the form:

$$f(\bar{x}) = \sum_{i} \prod_{j} C_{ij}$$

where again $C_{ij} \in \mathcal{C}$ and in addition each variable $x_t$ occurs in at most $k$ of the sub-circuits $C_{ij}$. That is, for every $x_t$ at most $k$ of the $C_{ij}$-s depend on it.
Note that, unlike the previous definition, there is **no restriction** on the number of additions or multiplications taken.

**Results for Polynomial Identity Testing** Our main result here is a general technique for lifting efficient polynomial identity testing algorithms from basic classes of circuits, satisfying some closure properties, to more general classes of circuits. Informally, let $\mathcal{C}$ be any class of circuits such that we can efficiently do black-box PIT for circuits from $\mathcal{C}$. Moreover, assume that we can also efficiently get black-box PIT algorithms

for constant-wise sums and products of circuits from $\mathcal{C}$, i.e. for circuits from $\Sigma^{[r]}\Pi^{[\ell]}\mathcal{C}$, where $r$ and $\ell$ are constant. Then we can "lift" this algorithm to an efficient black-box PIT algorithm for circuits from the class occur-$k$ $\Sigma\Pi\mathcal{C}$ (for any constant $k$). This class has **unbounded** fan-in sums of products of circuits from $\mathcal{C}$, but with the constraint that each variable appears in at most $k$ of the circuits from $\mathcal{C}$.

**Theorem 1.** *Let $n, k, s \in \mathbb{N}$ and let $\mathbb{F}$ be an arbitrary field. Let $\mathcal{H}$ be a hitting set for $\Sigma^{[(2k)!]}\Pi^{[4k^2]}\mathcal{C}$ formulas of size $s$ over $\mathbb{F}[x_1, x_2, \ldots, x_n]$. Then there exists a deterministic algorithm that given $n, k, s$ outputs a hitting set $\mathcal{H}'$ of size $|\mathcal{H}'| = |\mathcal{H}|^2 \cdot s^{\mathcal{O}(k)}$ for occur-$k$ $\Sigma\Pi\mathcal{C}$ formulas of size $s$ over $\mathbb{F}[x_1, x_2, \ldots, x_n]$,*

Once we have this general theorem, we immediately get some nice corollaries of this result by applying it to circuit classes $\mathcal{C}$ where we have efficient PIT algorithms even for circuits from $\Sigma^{[k]}\Pi^{[\ell]}\mathcal{C}$, where $k$ and $\ell$ are constant. The first class we apply it to is that of sparse polynomials (a.k.a depth-2 circuits). Notice that for this class, $\Sigma^{[k]}\Pi^{[\ell]}\mathcal{C}$ circuits are also reasonably sparse (when $k$ and $\ell$ are constant) and we can use the black-box PIT algorithm from [KS01].

**Theorem 2.** *Let $n, k, s \in \mathbb{N}$ and let $\mathbb{F}$ be an arbitrary field. Suppose that $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ is a polynomial computed by an occur-$k$ depth-4 $\Sigma\Pi\Sigma\Pi$ formula of size $s$. Then there exists a deterministic algorithm that given $n, k, s$ and a black-box access to $f$ decides if $f \equiv 0$, in time $s^{\mathcal{O}(k^2)}$.*

A special case of Theorem 2 is when the underlying circuit class is that of depth-4 multilinear circuits of top fan-in $k$. This special case was handled in [SV18] using very different techniques, and that proof does work over all fields. Moreover, the time complexity of that algorithm was $s^{\mathcal{O}(k^3)}$, and hence slightly worse than the running time bound we obtain here.

Prior to this work, this model (among others) was studied in a breakthrough work by [ASSS16] where the authors used the Jacobian very effectively. One drawback of the Jacobian method is that it only works over fields of characteristic 0 or large characteristic. Thus the above result was only known for $\text{char}(\mathbb{F}) = 0$ or $\text{char}(\mathbb{F}) > s^{2k}$. It since has been an open question to extend the result to all fields, and indeed there were other attempts in this direction [KS17, PSS18, CS19]. The Jacobian method captures the notion of algebraic independence of polynomials. We sidestep the Jacobian by showing that it suffices to use just *linear independence*.

An additional point of distinction is that one can also view the approach behind the result of [ASSS16] as a way to "lift" efficient black-box PIT algorithms of a certain kind to an efficient black-box PIT algorithm for circuits from the class occur-$k$ $\Sigma\Pi\mathcal{C}$. Yet, the assumption of [ASSS16] is arguably stronger: the assumed PIT algorithms should, among other things, work not just for $\Sigma^{[k]}\Pi^{[\ell]}\mathcal{C}$, where $k$ and $\ell$ are constant, but also for the class $\Sigma^{[k]}\Pi^{[\ell]}\partial\mathcal{C}$, i.e. for constant-wise sums and products of *partial derivatives* of circuits from $\mathcal{C}$. It is to be noted that not all circuit classes are closed under taking partial derivatives.

In particular, our weakened assumption allows us to prove Theorem 3, which is, to the best of our knowledge, the first efficient PIT algorithm for this circuit class. We obtain this result by another interesting instantiation of Theorem 1 when the circuit class $\mathcal{C}$ is that of depth-3 circuits of constant top fan-in, i.e. $\Sigma\Pi\Sigma(k)$ circuits. Since we have polynomial-time black-box PIT algorithms for this model for any constant value of $k$ [DS07, KS07, KS09b, SS12], we also have it for $\Sigma^{[k]}\Pi^{[\ell]}\mathcal{C}$ formulas, where $k$ and $\ell$ are constant. Thus, in this case, we get the following result.

**Theorem 3.** *Let $n, k_1, k_2, s \in \mathbb{N}$ and let $\mathbb{F}$ be an arbitrary field. Suppose that $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ is a polynomial computed by an occur-$k_1$ depth-5 $\Sigma\Pi(\Sigma^{[k_2]}\Pi\Sigma)$ formula of size $s$. Then there exists a deterministic algorithm that given $n, k_1, k_2, s$ and a black-box access to $f$ decides if $f \equiv 0$, in time $s^{k_2^{\mathcal{O}(k_1^2)}}$.*

**Remark 1.1.** *The Jacobian-based techniques of [ASSS16] were extended to handle the "bounded-depth and bounded-occur" setting. Yet, the key difference is that in their model, the occurrence is "charged" for each variable that occurs in the bottom* two *layers of the formula (i.e. how many of the bottom depth-*2 *sub-formulas depend on each variable). Whereas in our model, the occurrence is "charged" for each variable that occurs in the bottom* three layers *of the formula (i.e. how many of the bottom depth-*3 *sub-formulas depend on each variable).*

*It is not immediately clear whether the Jacobian-based techniques could be used to handle our way of "charging" as it would require to hit the Jacobian of depth-3 $\Sigma\Pi\Sigma(k)$ circuits. To the best our knowledge, no efficient PIT algorithm is known for this model, partially due to the fact that a naïve computation of a partial derivative of a depth-3 $\Sigma\Pi\Sigma(k)$ circuit could, potentially, result in a depth-3 circuit with* unbounded *top fan-in.*

**Results for Arithmetic Circuit Reconstruction** We now describe our applications to the problem of arithmetic circuit reconstruction.

A polynomial $f \in \mathbb{F}[X]$ is called *set-multilinear*, if there exists a partition $X = \cup_{j\in[d]} X_j$ such that every monomial that appears in $f$ is of the form $x_{i_1}x_{i_2}\cdots x_{i_d}$ where $x_{i_j} \in X_j$. In other words, each monomial in such $f$ picks up exactly one variable from each part in the partition. We say that a $\Sigma\Pi\Sigma(k)$ circuit $C$ is *set-multilinear* if there exists a partition $X = \cup_{j\in[d]} X_j$ such that every gate in $C$ computes a set-multilinear polynomial w.r.t. this partition. In this case, $C$ is a $\Sigma\Pi\Sigma_{\{\sqcup_j X_j\}}(k)$ circuit.

In this work we study the model of set-multilinear $\Sigma\Pi\Sigma(k)$ circuits of *unknown* variable partition. Thus the black-box reconstruction algorithm knows the set of variables $X$, but does not know the partition of variables under which the polynomial can be represented as a $\Sigma\Pi\Sigma_{\{\sqcup_j X_j\}}(k)$ circuit. We obtain the first polynomial or even subexponential-time proper learning algorithm for this model. Prior to our work, the model of proper learning for $\Sigma\Pi\Sigma_{\{\sqcup_j X_j\}}(k)$ circuits with *known* variable partition was studied in the work of [BSV21] as well as the recent work of [PSV22] where polynomial-time algorithms were obtained. These works also gave efficient reconstruction algorithms for multilinear $\Sigma\Pi\Sigma(k)$ circuits, a more general model. However, this does not immediately imply a *proper* reconstruction model for simpler models, such as set-multilinear $\Sigma\Pi\Sigma(k)$. We would like to remark that [PSV22] did give a way to adapt the multilinear $\Sigma\Pi\Sigma(k)$ learning algorithm to proper learning set-multilinear circuit, but their argument crucially uses knowledge of the variable partition (for instance, the step of "finding the basis of a particular vector space" in [PSV22, Section 6.2]) and thus does not work in the unknown partition case.

**Theorem 4.** *Let $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a degree $d$ polynomial computed by a set-multilinear $\Sigma\Pi\Sigma(k)$ circuit (of unknown partition) and $\mathrm{char}(\mathbb{F}) > d$ or $\mathrm{char}(\mathbb{F}) = 0$. Then there exists a randomized algorithm that given black-box access to $f$ outputs a set-multilinear $\Sigma\Pi\Sigma(k)$ circuit $C$ that computes $f$, in time* $\mathrm{poly}(d^{k^{\mathcal{O}(1)}}, k^{k^{\mathcal{O}(1)}}, n)$.

We would like to comment that given a set-multilinear $\Sigma\Pi\Sigma$ circuit of an *unknown* partition (and arbitrary top fan-in), the task of actually finding the set-multilinear partition is NP-hard! Indeed, one can embed the problem of graph 3-coloring into a problem of recovering the partition. See Section 7.5 for more details.

Thus in general the task of proper reconstruction of set-multilinear circuits of known and unknown partition (proper reconstruction is important here, since in this model we do need to compute the partition) can potentially have vastly different complexities. In this paper we are able to give efficient algorithms in the setting of unknown partition when the top fan-in is constant.

## 1.3 Related Work

**Linear independence**: Just like the alternant, there are other tools to study the linear independence of functions. For instance, the *Wronskian*. The Wronskian is defined for a finite family $f_1, \ldots, f_n$ of $(k-1)$-times differentiable univariate functions, and is defined as the determinant of the Wronskian matrix $W = (\partial^{=(i-1)} f_j)_{i,j\in[k]}$. If $f_i \in \mathbb{F}[x_1, \ldots, x_n]$ and $\mathbb{F}$ has characteristic 0[1], then a necessary and sufficient criterion for the polynomials to be linearly dependent is that the Wronskian is the identically zero polynomial. Again, this criterion fails for finite fields, since the derivatives of higher-degree polynomials might be zero over low-characteristic. There have been works to adapt this criterion for low-characteristic fields using "folded-Wronskian". We refer the reader to [GK16] and the references therein for a detailed discussion on Wronskian, folded Wronskian, and its applications. Note that, this criterion (as stated above) works

[1] or sufficiently large

for univariate polynomials only[2], as opposed to alternant which works for multivariate polynomials as well. Also, the use of partial derivatives here can complicate simple models, for instance, derivatives of $\Sigma\Pi\Sigma(k)$ circuits. However, the alternant matrix which consists only of evaluations does not suffer from this fate.

**Polynomial Identity Testing**: In the constant depth regime, some of the major PIT results have been the construction of poly-size hitting sets for $\Sigma\Pi\Sigma(k)$ circuits [DS07, KS07, KS09b, SS12], hitting sets for restricted $\Sigma\Pi\Sigma\Pi(k)$ circuits [SV18, DDS21, PS21] and the recent breakthrough of [LST22] that gave sub-exponential size hitting sets for constant-depth circuits. A striking fact about all these PIT results is the diverse set of techniques used to prove them.

The result of [ASSS16] used the notion of algebraic independence to unify a number of PIT results proven with completely different techniques [GKPS11, BMS13, AvMV15, SV18]. However, since the Jacobian criteria fails over low-characteristic fields, [ASSS16] can only get PIT algorithms for large characteristic fields. Nevertheless, there have been attempts to adapt the Jacobian criteria for finite fields, and further use it for PIT. In particular, the work of [PSS18] adapted the Jacobian criteria in the special case when a certain parameter called the inseparable degree of the underlying polynomials is bounded. Using these criteria, [PSS18] (and later [CS19]) had applications in PIT for special models. We surpassed the intricate nature of algebraic independence over low-characteristic altogether by working with linear independence. Thus our PIT results work for fields of all characteristics.

**Reconstruction**: The work of [BSV21] studied proper-learning of set-multilinear depth-3 $\Sigma\Pi\Sigma(k)$ circuits for constant top fan-in. This work was further improved by [PSV22] from $n^{a(k)}$ to $\text{poly}(n) \cdot b(k)$, for some function $a$ and $b$. The main motivation for studying this comes from its connection to tensor decomposition [BSV21, Section 1.1].

In both of these works, the learning algorithms critically need the variable partition. For instance, the "width-reduction" procedure in [BSV21] and the "finding the basis of a particular vector space" in [PSV22, Section 6.2], both crucially use the variable partition.

As another application of our SWLI Lemma, we show an algorithm for learning set-multilinear $\Sigma\Pi\Sigma(k)$ circuits, with an *unknown* partition. In order to do this, we first prove a generalized version of Carlini's variable reduction [Car06]. Secondly, we introduce a new operator called the *Generalized Alternant*. Our operator matrix has a specific decomposition when the input polynomial is a set-multilinear $\Sigma\Pi\Sigma(k)$ circuit. In fact, the matrix has a nice decomposition under a somewhat more general condition. We hope this operator will be utilized for future purposes. (See Section 5 for more details). We would like to remark that similar operator matrices (and their matrix decompositions) have been used in the literature for sparsity testing and PIT algorithms for sparse polynomials [BOT88, GJR10]. Thus, our work can be seen as generalizing those explicit decompositions to the case of set-multilinear $\Sigma\Pi\Sigma$ circuits.

## 2 Proof Overview

The starting point of our results is a new method for analyzing linear independence of a collection of polynomials. In Lemma 4.3 we show that if $k$ multivariate polynomials over any field $\mathbb{F}$ are linearly independent over the base field $\mathbb{F}$, then there exists a subset of $k-1$ variables such that if we keep just those $k-1$ variables "alive" and set the rest to random values, then the polynomials continue to be linearly independent with high probability. More precisely, we show that if $f_1, \ldots, f_k \in \mathbb{F}[X]$ are $\mathbb{F}$-linearly independent polynomials, then there exists a subset $S \subseteq X$ of size $|S| \leq k-1$ such that the polynomials are $\mathbb{F}(X \setminus S)$-linearly independent.

To the best of our knowledge this result is new. Moreover, it seems like a valuable and fundamental tool for analyzing arithmetic circuits. We demonstrate why such a result is so interesting by using it to obtain new and efficient polynomial identity testing and reconstruction algorithms for some classes of circuits.

[2]There is a multivariate generalization of the Wronskian (see [BD10]), but this is a collection of exponentially many "determinants" with a promise that at least one of them is non-zero. Thus, it is not clear how to use it efficiently.

## 2.1 Application to PIT

One may wonder how to use such a result for PIT, since though when $k$ is small we can efficiently "guess" the set of $k$ variables by going over all possibilities, setting the remaining variables to "random values" is not really a luxury we can afford for a deterministic identity test. To harness the power of this lemma, we couple it with the *alternant* - a well-known tool to check the linear independence of functions [Ait17].

Formally, let $\mathbf{f} = \{f_1(X), f_2(X), \ldots, f_k(X)\} \subseteq \mathbb{F}[X]$ be a set of polynomials over $\mathbb{F}[X]$, where $X$ is a set of size $n$. Let $X_1, \ldots, X_k$ be disjoint sets of variables each of size $n$. Then the polynomials in $\mathbf{f}$ are $\mathbb{F}$-linearly independent iff $\det(\mathrm{Alt}_{\mathbf{f}}(X_1, X_2, \ldots, X_k)) \not\equiv 0$, where $\mathrm{Alt}_{\mathbf{f}}(X_1, X_2, \ldots, X_k)$ is defined to be the following matrix:

$$\mathrm{Alt}_{\mathbf{f}}(X_1, X_2, \ldots, X_k) \triangleq \begin{pmatrix} f_1(X_1) & f_2(X_1) & \cdots & \cdots & f_k(X_1) \\ f_1(X_2) & f_2(X_2) & \cdots & \cdots & f_k(X_2) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ f_1(X_k) & f_2(X_k) & \cdots & \cdots & f_k(X_k) \end{pmatrix}.$$

Now, if the number of polynomials, $k$, is constant and the number of variables, $n$, is large, then we can couple these two lemmas very effectively in certain situations. Notice that to test $\mathbb{F}$-linear independence of $f_1, \ldots, f_k$, the SWLI Lemma shows that it suffices to test $\mathbb{F}(X \setminus S)$-linear independence of $f_1, \ldots, f_k$. In other words, we are viewing $f_1, \ldots, f_k$ as polynomials in just $k-1$ variables, but over the larger field of rational functions $\mathbb{F}(X \setminus S)$ and testing linear independence there. This in some sense is a "variable reduction" procedure, though we have made our base field more complex.

Since we now have polynomials over just $k-1$ variables, we can now use the above result but just make multiple copies of the special $k-1$ variables! We now have to test non-zeroness of the determinant over $\mathbb{F}(X \setminus S)$, but what is very nice is that is just the same thing as being nonzero over $\mathbb{F}$. To set up some notation, let the set of variables $X = Y \cup Z$ where $Y$ has size $k-1$ and we are checking linear independence over $\mathbb{F}(Z)$. Thus each $f_i$ is a member of $\mathbb{F}[Y, Z]$. Let $Y_1, \ldots, Y_k$ be disjoint sets of variables each of size $k-1$. These will be the multiple copies of the $Y$ variables. Define:

$$\mathrm{Alt}^{Y}_{\mathbf{f}}(Y_1, Y_2, \ldots, Y_k, Z) \triangleq \begin{pmatrix} f_1(Y_1, Z) & f_2(Y_1, Z) & \cdots & \cdots & f_k(Y_1, Z) \\ f_1(Y_2, Z) & f_2(Y_2, Z) & \cdots & \cdots & f_k(Y_2, Z) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ f_1(Y_k, Z) & f_2(Y_k, Z) & \cdots & \cdots & f_k(Y_k, Z) \end{pmatrix}.$$

Then it is not hard to see that the polynomials in $\mathbf{f}$ are $\mathbb{F}(Z)$-linearly independent iff $\det(\mathrm{Alt}_{\mathbf{f}}(Y_1, Y_2, \ldots, Y_k, Z)) \not\equiv 0$.

Thus our task of testing if $f_1, \ldots, f_k \in \mathbb{F}[X]$ are $\mathbb{F}$-linearly independent has reduced to testing if $\det(\mathrm{Alt}^{Y}_{\mathbf{f}}(Y_1, Y_2, \ldots, Y_k, Z)) \not\equiv 0$. We now show how to do precisely this, if the $f_i$-s satisfy some additional "nice" properties.

To understand how our general PIT algorithms work, it will be instructive to focus on the special case of depth-4 multilinear circuits of constant top fan-in. The first polynomial-time black-box hitting set for this model was provided in [SV18]. In [ASSS16] this was simplified, generalized, and the parameters were improved, though the results of [ASSS16] only work over fields of characteristic 0[3]. Our work gives an alternate simple proof, with the added benefit of working over all fields. Like the work of [ASSS16], we are also able to generalize the hitting set result to work for the model of depth-4 occur-$k$ formulas. Our proof technique is arguably simpler and is the first work that gives an efficient black-box hitting set for depth-4 occur-$k$ formulas that works over all fields.

**Black-box PIT for Depth-4 Multilinear Circuits of Constant Top Fan-in** Let $k$ be the top fan-in. In this model, we are looking at circuits of the form $C = \sum_{i=1}^{k} f_i$, where each $f_i$ is of the form $\prod_{j=1}^{d} A_{ij}$. Each $A_{ij}$ is a multilinear sparse polynomial and each $f_i$ is multilinear, and hence has factors that are variable

[3]Or sufficiently large characteristic.

disjoint. First, we observe that if $C$ is nonzero and is represented as a multilinear circuit with the smallest top fan-in, then we can essentially assume that the $f_i$-s are linearly independent: if there was a linear dependency then we can use it to express the same polynomial as a depth-4 multilinear circuit of smaller top fan-in. Thus, in order to certify the nonzeroness of $C$, it is sufficient to certify the linear independence of the $f_i$-s. Now, by the SWLI lemma (Lemma 4.3), there exists a subset $Y \subseteq X$ of size $k-1$ (and we can find this subset by iterating over all possible subsets of this size) such that for $Z = X \setminus Y$ the $f_i$-s are linearly independent iff $\det(\mathrm{Alt}^Y_{\mathbf{f}}(Y_1, Y_2, \ldots, Y_k, Z)) \not\equiv 0$.

Thus, all we need to do is to find a hitting set for $\det(\mathrm{Alt}^Y_{\mathbf{f}}(Y_1, Y_2, \ldots, Y_k, Z))$. We will observe that $\det(\mathrm{Alt}^Y_{\mathbf{f}}(Y_1, Y_2, \ldots, Y_k, Z))$ can actually be written as the product of sparse polynomials. Hitting set generators for such polynomials are well-known (e.g. [KS01]). Consequently, this immediately gives a hitting set generator for $\det(\mathrm{Alt}^Y_{\mathbf{f}}(Y_1, Y_2, \ldots, Y_k, Z))$.

Here is why $\det(\mathrm{Alt}^Y_{\mathbf{f}}(Y_1, Y_2, \ldots, Y_k, Z))$ is a product of sparse polynomials. Let us inspect any column of $\mathrm{Alt}^Y_{\mathbf{f}}(Y_1, Y_2, \ldots, Y_k, Z)$, say the first. We observe that only the $Y$ variables vary in the different entries. And while $f_1$ might be a product of "many" sparse polynomials, the $Y$-variables "touch" only (at most) $k-1$ of them since $f_1$ is multilinear and $|Y| \leq k-1$. The remaining factors of $f_1$ remain unaffected and the same holds true for all the entries of the first column. These factors can then be pulled out of the matrix and will be factors of the determinant. We repeat this operation per column and pull out all factors of each column that do not have the $Y$-variables. Then $\det(\mathrm{Alt}^Y_{\mathbf{f}}(Y_1, Y_2, \ldots, Y_k, Z))$ is a product of all these sparse factors and the determinant of the new "reduced" matrix $D'$. $D'$ is a $k \times k$ matrix and each entry is a product of at most $k-1$ sparse polynomials. Since each sparse polynomial has at most $s$ monomials (where $s$ is the size of $C$) it is easy to see $\det(D')$ has at most $k! \times s^{k^2}$ monomials, which is still relatively sparse if $k$ is constant.

**Black-Box PIT for Constant-Occur Depth-4 Circuits** The main observation here is that if $C$ is a occur-$k$ depth-4 circuit then $C$ is nonzero[4] iff some discrete first order partial derivative[5] of $C$ is nonzero. (We use discrete derivatives so that we do not run into issues with the characteristic of the underlying field). Moreover, we have black-box access to these derivatives. Thus, it suffices to hit the first-order discrete derivatives of $C$. Notice, however, that since $C$ is occur-$k$, each discrete derivative of $C$ is an occur-$2k$ depth-4 circuit of top fan-in at most $2k$. Thus we have reduced our question to doing black-box PIT for a circuit of the form $C' = \sum_{i=1}^{2k} f_i$, where each $f_i$ is a product of sparse polynomials where each variable occurs in only $2k$ of the sparse polynomials. Now the rest of the argument is almost identical to the that for multilinear depth-4 circuits of top fan-in $k$. We analyze the analogous matrix $\mathrm{Alt}^Y_{\mathbf{f}}(Y_1, Y_2, \ldots, Y_k, Z)$. The only change now is that each $Y$ variable might appear in up to $2k$ factors of any $f_i$. However after pulling out $Y$-untouched factors from each column, it is easy to see that the resulting determinant is still sparse, albeit with slightly worse parameters.

**PIT for Occur-$k$ $\Sigma\Pi\mathcal{C}$ Circuits** More generally, let $\mathcal{C}$ be any class of circuits for which there are efficient black-box PIT algorithms even for constant-wise sums of constant-wise products of circuits from $\mathcal{C}$, i.e. the model of circuits from $\Sigma^{[k]}\Pi^{[\ell]}\mathcal{C}$, when $k$ and $\ell$ are constant. We show that we can also obtain efficient black-box PIT algorithms for occur-$k$ $\Sigma\Pi\mathcal{C}$ circuits. The skeleton of the proof is just like that for black-box PIT for constant-occur depth-4 circuits (this is the special case when the class $\mathcal{C}$ is that of depth-2 circuits, i.e. sparse polynomials). The key difference is in the structure of the matrix $D'$ after the common factors are pulled out and in the structure of the factors themselves. Upon inspection, it is not hard to see that $\det(\mathrm{Alt}^Y_{\mathbf{f}}(Y_1, Y_2, \ldots, Y_k, Z))$ can be computed by a circuit of the form $\hat{C} \cdot C'$ where $\hat{C}$ is a $\Sigma^{[(2k)!]}\Pi^{[4k^2]}\mathcal{C}$ circuit (for which we assumed there are efficient black-box PIT algorithms when $k$ is constant) and $C'$ is a circuit from $\Pi\mathcal{C}$ (and since we have efficient black-box PIT algorithms for circuits in $\mathcal{C}$, we also have it for $\Pi\mathcal{C}$ using hitting set generators.) The formal argument is given in Lemma 6.5.

[4] In fact, non-constant.

[5] $C'_m \triangleq C - C|_{x_m=0}$

## 2.2 Application to Reconstruction

We show how to use the SWLI lemma as well as a variant of alternants to obtain efficient proper reconstruction algorithms for set-multilinear $\Sigma\Pi\Sigma(k)$ circuits of *unknown* partition. Let $C$ be a set-multilinear $\Sigma\Pi\Sigma(k)$ circuit to which we have black-box access. There are two cases to consider - when the underlying linear forms have "low" linear rank (or dimension), and when they have "high" rank.

We handle the low-rank case by setting up a suitable system of polynomial equations whose solution gives the circuit representation we are looking for. These equations need to be carefully set up to ensure that their solution will correspond to a set-multilinear circuit representation. Moreover, we need to be able to do this even without knowing the variable partition!

The high-rank case needs several additional ideas. At the heart of our algorithm is a procedure for learning even just a *few* of the linear forms appearing in $C$. Once we have even one or two, we can then leverage this information to learn *many more* of the linear forms by looking at a restriction where the known linear forms are set to 0, and then inductively learning the restricted circuit (which has a smaller top fan-in). A key observation is that since a set-multilinear circuit is a special case of a multilinear circuit, consequently, using rank bounds for *multilinear* circuits one can then argue that (any representation of) the restricted circuit that is learnt will have *many* linear forms in common with the original underlying circuit that we were hoping to learn. There are several other additional details. Even once we learn many or most of the linear forms, learning the last few can pose some challenges. We set up a suitable system of polynomial equations in a few variables whose solution eventually gives us all the linear forms. For the purpose of this proof overview, we will focus on what was the most challenging and interesting step – getting our hands on at least one of the linear forms appearing in $C$.

**Learning One or More Linear Forms of $C$ using Generalized Alternants** We introduce a novel, alternant-based technique that would work for any polynomial $F \in \mathbb{F}[W]$ of the following form. Suppose there exists a partition $W = X \cup Y \cup Z$ for which $F$ can be written as

$$F(X, Y, Z) = \sum_{t=1}^{k} R_t(X) \cdot Q_t(Y) \cdot H_t(Z) \tag{1}$$

and, in addition, the polynomials in $\mathbf{R} \triangleq \{R_1(X), \ldots, R_k(X)\}$ and $\mathbf{Q} \triangleq \{Q_1(Y), \ldots, R_k(Y)\}$ are $\mathbb{F}$-linearly independent. (With a little bit of effort we show that set-multilinear $\Sigma\Pi\Sigma(k)$ circuits in some sense have this property.) We then show (see Theorem 5.3) how to output a list of polynomials that contains (among other things) all the irreducible factors of $H_1(Z) \cdot \ldots \cdot H_k(Z)$.

As mentioned before, if our input polynomial $F$ is computable by a set-multilinear $\Sigma\Pi\Sigma(k)$ circuit $C$, there are two cases to consider. If the set of all linear forms appearing in $C$ (after stripping off the common GCD of all product gates) has small rank, then we are in the "low-rank" case which we handle using systems of polynomial equations (see above). Otherwise, we are in the "high-rank" case. We then show that if $C$ has "high rank" then there exists a partition of $F$'s variables such that $F$ can be expressed as in Equation 1 above when the $H_i$-s all appear in the original representation of $C$. Therefore, by learning the $H_i$-s, we actually learn "many" linear forms of $C$! The proof of this fact is nontrivial and crucially uses the SWLI Lemma. The precise technical details can be found in Lemma 7.4.

Let us now discuss the idea behind our technique. To achieve the goal, we introduce an operator that we have called *generalized alternant*. See Section 5 for more details. Let $F(X, Y, Z) \in \mathbb{F}[X, Y, Z]$ and let $k \geq 1$. Let $X_1, \ldots, X_k$ and $Y_1, \ldots, Y_k$ be disjoint copies of $X$ and $Y$, respectively. Consider the following matrix which we call the *generalized alternant of $F$ with respect to $X$ and $Y$ of order $k$*. Notice that we can simulate black-box access to the entries of this matrix *if we know the variable partition.*

$$M = \begin{pmatrix} F(X_1,Y_1,Z) & F(X_1,Y_2,Z) & \cdots & F(X_1,Y_k,Z) \\ F(X_2,Y_1,Z) & F(X_2,Y_2,Z) & \cdots & F(X_2,Y_k,Z) \\ \vdots & \vdots & \vdots & \vdots \\ F(X_k,Y_1,Z) & F(X_k,Y_2,Z) & \cdots & F(X_k,Y_k,Z) \end{pmatrix}. \quad \text{Formally, } \{M_{ij}\}_{\{i,j\in[k]\}} = F(X_i,Y_j,Z).$$

The main observation here is that if $F$ can be expressed as in Equation 1 above, the matrix factors as

$$M = \begin{pmatrix} R_1(X_1) & R_2(X_1) & \cdots & R_k(X_1) \\ R_1(X_2) & R_2(X_2) & \cdots & R_k(X_2) \\ \vdots & \vdots & \vdots & \vdots \\ R_1(X_k) & R_2(X_k) & \cdots & R_k(X_k) \end{pmatrix} \cdot \begin{pmatrix} H_1(Z) & & & \\ & \ddots & & \\ & & \ddots & \\ & & & H_k(Z) \end{pmatrix} \cdot \begin{pmatrix} Q_1(Y_1) & Q_1(Y_2) & \cdots & Q_1(Y_k) \\ Q_2(Y_1) & Q_2(Y_2) & \cdots & Q_2(Y_k) \\ \vdots & \vdots & \vdots & \vdots \\ Q_k(Y_1) & Q_k(Y_2) & \cdots & Q_k(Y_k) \end{pmatrix}$$

Observe that the first and third matrices are precisely the alternant for $\mathbf{R}$ and the transposed alternant for $\mathbf{Q}$, respectively. Consequently: $\det(M) = \det(\text{Alt}^X_{\mathbf{R}}) \cdot \det(\text{Alt}^Y_{\mathbf{Q}}) \cdot H_1(Z) \cdot \ldots \cdot H_k(Z)$. The last piece of the puzzle would be to recall that the polynomials in both $\mathbf{R}$ and $\mathbf{Q}$ are $\mathbb{F}$-linearly independent and hence by Lemma 4.7: $\det(\text{Alt}^X_{\mathbf{R}}), \det(\text{Alt}^Y_{\mathbf{Q}}) \not\equiv 0$ which in turn implies that $\det(M) \not\equiv 0$. As a result, we use the black-box factorization algorithm of [KT90] (see Lemma 3.11) to learn the irreducible factors of $H_1(Z) \cdot \ldots \cdot H_k(Z)$.

What remains to discuss is how we find the appropriate variable partition as the algorithm operates under the mere assumption that a partition $W = X \cup Y \cup Z$ *exists* (but is not given as input). The naïve approach of simply guessing a partition would be too costly since there are $3^n$ possible options. Once again, the SWLI lemma comes to our rescue.

Since the polynomials in both $\mathbf{R}$ and $\mathbf{Q}$ are $\mathbb{F}$-linearly independent, by the SWLI lemma, we know that there is a subset $X' \subseteq X$ of $k-1$ variables from $X$, and similarly a subset $Y' \subseteq Y$ of $k-1$ from $Y$ such that the polynomials in both $\mathbf{R}$ and $\mathbf{Q}$ remain linearly independent when regarded as polynomials in $X'$ and $Y'$, respectively. Consequently, if we defined the generalized alternant $M'$ only with respect to those variables, the two alternant matrices will still remain full rank resulting in $\det(M') \not\equiv 0$. Yet, since $X'$ and $Y'$ are "small", we can "guess" them by iterating over all possibilities. For additional details, see Theorem 5.3.

# 3 Notations and Preliminaries

Throughout the paper, we use uppercase $X, Y$ to denote sets of variables, lowercase $x_i$ to denote single variables and $\bar{x}, \bar{y}$ to denote vector/tuple of variables and $\bar{v}$ to denote a vector/tuple of field constants. We sometimes abuse notations by referring to a circuit as a collection of multiplication $\Sigma\Pi$ gates. Let $\mathbb{F}$ denote a field, finite or otherwise, and let $\overline{\mathbb{F}}$ denote its algebraic closure

.

## 3.1 Polynomials

A polynomial $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ *depends* on a variable $x_i$ if there are two inputs $\bar{\alpha}, \bar{\beta} \in \overline{\mathbb{F}}^n$ differing only in the $i^{th}$ coordinate for which $f(\bar{\alpha}) \neq f(\bar{\beta})$. Equivalently, $f$ *depends* on a variable $x_i$ if there is a monomial in $f$ which contains $x_i$. We denote by $\text{var}(f)$ the set of variables that $f$ depends on.

For a polynomial $f(x_1, \ldots, x_n)$, a variable $x_i$ and a field element $\alpha$, we denote with $f|_{x_i=\alpha}$ the polynomial resulting from substituting $\alpha$ to $x_i$. Similarly given a subset $I \subseteq [n]$ and an assignment $\bar{a} \in \mathbb{F}^n$, we define $f|_{\bar{x}_I=\bar{a}_I}$ to be the polynomial resulting from substituting $a_i$ to $x_i$ for every $i \in I$.

Let $f, g \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be polynomials. We say that $g$ *divides* $f$, or equivalently $g$ is a factor of $f$, and denote it by $g \mid f$ if there exists a polynomial $h \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ such that $f = g \cdot h$. We say that $f$ is *irreducible* if $f$ is non-constant and cannot be written as a product of two non-constant polynomials.

Given the notion of divisibility, we define the gcd of a set of polynomials in a natural way to be the highest degree polynomial dividing them all (suitably scaled)[6]. A *linear function* is a polynomial of the form $L(X) = \sum_{i=1}^{n} a_i x_i + a_0$ with $a_i \in \mathbb{F}$.
Inspired by a similar notion of [KS09a], in [BSV21] a distance measure between polynomials was defined.

**Definition 3.1** ([BSV21])**.** *For $f, g \in \mathbb{F}[x_1, x_2, \ldots, x_n]$, we define a* distance *function:*

$$\Delta(f,g) \triangleq \frac{\max\{\deg(f), \deg(g)\}}{\deg(\gcd(f,g))}.$$

## 3.2 Generators for Circuit Classes

In this section, we formally define the notion of generators and hitting sets for polynomials as well as describe a few of their useful properties. For a further discussion see [SV15, SY10].

A map $\mathcal{G} = (\mathcal{G}^1, \ldots, \mathcal{G}^n) : \mathbb{F}^r \to \mathbb{F}^n$ is a *generator* for a circuit class $\mathcal{C}$, if for every non-zero $n$-variate polynomial $P \in \mathcal{C}$, it holds that $P(\mathcal{G}) \not\equiv 0$. For $Z \subseteq [n]$, we denote by $\mathcal{G}^Z$ the projection of $\mathcal{G}$ to coordinates that correspond to the variables in $Z$.

The image of the map $\mathcal{G}$ is denoted as $\mathrm{Im}(\mathcal{G}) \triangleq \mathcal{G}(\overline{\mathbb{F}}^r)$. Ideally, $r$ should be very small compared to $n$. A set $\mathcal{H} \subseteq \mathbb{F}^n$ is a *hitting set* for a circuit class $\mathcal{C}$, if for every non-zero polynomial $P \in \mathcal{C}$, there exists $\bar{a} \in \mathcal{H}$, such that $P(\bar{a}) \neq 0$. A generator can also be viewed as a map containing a hitting set for $\mathcal{C}$ in its image. That is, for every non-zero $P \in \mathcal{C}$, there exists $\bar{a} \in \mathrm{Im}(\mathcal{G})$ such that $P(\bar{a}) \neq 0$. In identity testing, generators and hitting sets play the same role. Given a generator one can easily construct a hitting set by evaluating the generator on a large enough set of points (see Lemma 3.9 for more details). Conversely in [SV15], an efficient method for constructing a generator from a hitting set was given.

**Lemma 3.2** ([SV15])**.** *There exists a deterministic algorithm that given a set $\mathcal{H} \subseteq \mathbb{F}^n$ and $\delta > 1$ runs in time $\mathrm{poly}(|\mathcal{H}|, \delta)$ and constructs a map $\mathcal{G}(\bar{w}) : \mathbb{F}^r \to \mathbb{F}^n$ with $r = \lceil \log_\delta |\mathcal{H}| \rceil$ such that $\mathcal{H} \subseteq \mathrm{Im}(\mathcal{G})$ and each $\mathcal{G}^i$ has individual degree at most $\delta - 1$.*

**Remark 3.3.** *Since polynomial rings over fields are integral domains, one very useful property of generators is multiplicativity. That is, a generator for a class $\mathcal{C}$ is also a generator for the class $\Pi\mathcal{C}$.*

Since generators are polynomial maps, it is natural to define the sum of two generators $\mathcal{G}_1$ and $\mathcal{G}_2$ by their component-wise sum. We take the convention that for two generators $\mathcal{G}_1$ and $\mathcal{G}_2$ with the same output length the sum $\mathcal{G}_1 + \mathcal{G}_2$ is defined over disjoint input variables. That is $(\mathcal{G}_1 + \mathcal{G}_2)(X_1, X_2) \triangleq \mathcal{G}_1(X_1) + \mathcal{G}_2(X_2)$. Intuitively, one can think of adding a sample from $\mathcal{G}_1$ to an independent sample from $\mathcal{G}_2$.

### 3.2.1 The $G_{n,k}$ Generator of [SV15]

The $G_{n,k}$ generator was defined in [SV15].

**Definition 3.4** ([SV15])**.** *Let $a_1, \ldots, a_n$ denote $n$ distinct elements from a field $\mathbb{F}$ and for $i \in [n]$ let $L_i(x) \triangleq \prod_{j \neq i} \frac{x - a_j}{a_i - a_j}$ denote the corresponding Lagrange interpolant. For every $k \in \mathbb{N}$, define*

$$G_{n,k}(y_1, \ldots, y_k, z_1, \ldots, z_k) \triangleq \left( \sum_{j=1}^{k} L_1(y_j) z_j, \sum_{j=1}^{k} L_2(y_j) z_j, \ldots, \sum_{j=1}^{k} L_n(y_j) z_j \right).$$

Below are a number of useful properties that follow from its definition. For a formal proof see [SV15, Vol15].

[6] Such a polynomial is unique up to scaling, and one can fix a canonical polynomial in this class for instance by requiring that the leading monomial has coefficient 1. With this definition, two polynomials are pairwise coprime if their gcd is of degree 0, and in particular the gcd equals 1.

**Lemma 3.5** ([SV15, Vol15]). *Let $k, k'$ be positive integers, $f \in \mathbb{F}[X]$ and $H : \mathbb{F}^r \to \mathbb{F}^{|X|}$ be a polynomial map.*

1. *Variable "revival": let $X = Y \cup Z$ such that $|Y| \leq k$. Then there exists a substitution of field element and polynomials to the variables of $G_{n,k}$ for which the map $G_{n,k} + H$ results in $(Y, H^Z)$.*
2. *Additivity: $G_{n,k} + G_{n,k'} = G_{n,k+k'}$.*
3. *Let $x_m \in X$. Define $f_m \stackrel{\Delta}{=} f - f|_{x_m=0}$. Suppose $f_m(H) \not\equiv 0$. Then $f(H + G_{n,1}) \not\equiv 0$.*

## 3.3 Depth-3 Circuits & Our Models

In this section, we formally introduce the general model of depth-3 circuits and the specialization of set-multilinear depth-3 circuits, which is the focus of our paper. It is to be noted that depth-3 circuits were a subject of a long line of study [DS07, KS07, KS09b, SV15, AM10, KS11, SS11, SS12, SS13].

**Definition 3.6.** *A depth-3 $\Sigma\Pi\Sigma(k)$ circuit $C$ computes a polynomial of the form*

$$C(X) = \sum_{i=1}^{k} T_i(X) = \sum_{i=1}^{k} \prod_{j=1}^{d_i} \ell_{i,j}(X),$$

*where the $\ell_{i,j}$-s are linear functions; $\ell_{i,j}(X) = \sum_{t=1}^{n} a_{i,j}^t x_t + a_{i,j}^0$ with $a_{i,j}^t \in \mathbb{F}$. A* multilinear $\Sigma\Pi\Sigma(k)$ *circuit is a $\Sigma\Pi\Sigma(k)$ circuit in which each $T_i$ is a multilinear polynomial. In particular, each such $T_i$ is a product of variable-disjoint linear functions. Given a partition $X = \cup_{j\in[d]} X_j$ of $X$, a* set-multilinear $\Sigma\Pi\Sigma_{\{\sqcup_j X_j\}}(k)$ *circuit is a further specialization of a multilinear circuit to the case when each $\ell_{i,j}$ is a linear* form *in $\mathbb{F}[X_j]$. That is, each $\ell_{i,j}$ is defined over the variables in $X_j$ and $a_{i,j}^0 = 0$.*

When the partition is unknown we will simply refer to them as set-multilinear $\Sigma\Pi\Sigma$ circuits. It should be noted that a polynomial (and its circuit representation) can be set-multilinear w.r.t. to numerous variable partitions. For instance, $f = x_1 x_2 \cdots x_n + y_1 y_2 \cdots y_n$.

Observe that if a polynomial $f$ is computed by a set-multilinear circuit w.r.t (the partition) $X = \cup_{j\in[d]} X_j$, then every monomial that appears in $f$ is of the form $x_{i_1} x_{i_2} \cdots x_{i_d}$. where $x_{i_j} \in X_j$. In other words, each monomial in such $f$ picks up exactly one variable from each part in the partition.

We say that a set-multilinear depth-3 circuit is *optimal* if no circuit with a smaller fan-in (in that respective class) can compute the same polynomial.

We say that $C$ is *minimal* if no subset of the multiplication gates sums to zero. We define $\gcd(C)$ as the linear product of all the non-constant linear functions that belong to all the $T_i$-s. I.e. $\gcd(C) = \gcd(T_1, \ldots, T_k)$. We say that $C$ is *simple* if $\gcd(C) = 1$. The simplification of $C$, denoted by $\mathrm{sim}(C)$, is defined as $C/\gcd(C)$. In other words, the circuit results upon the removal of all the linear functions that appear in $\gcd(C)$.

For a depth-3 circuit $C$, its rank is defined as $\mathrm{rank}(C) = \dim(\mathrm{span}\{\ell_{i,j}\})$. Observe that for a multilinear circuit: $d \leq \mathrm{rank}(C)$.

The following result known as the *Rank Bound* provides a structural property for multilinear depth-3 computing the zero polynomial, under some technical conditions.

**Theorem 3.7** ([SS11]). *There exists a monotone function $R_M(k) \leq \mathcal{O}(k^3 \log k)$ such that any simple and minimal, multilinear $\Sigma\Pi\Sigma(k)$ circuit $C$, computing the zero polynomial satisfies $\mathrm{rank}(C) \leq R_M(k)$.*

**Definition 3.8** (Parametric Families of Circuits). *Let $\mathcal{C}$ be an arbitrary circuit class. For $k, \ell \in \mathbb{N}$, we define the class of $\Sigma^{[k]}\Pi^{[\ell]}\mathcal{C}$* formulas *as the class of polynomials of the form $\sum_{i=1}^{k} \prod_{j=1}^{\ell} C_{ij}$ where $C_{ij} \in \mathcal{C}$.*

*We define the class* occur-$k$ $\Sigma\Pi\mathcal{C}$ formulas *as the class of all the polynomials of the form:*

$$f(\bar{x}) = \sum_i \prod_j C_{ij}$$

*where each variable $x_t$ occurs in at most $k$ of the sub-circuits $C_{ij}$. That is, for every $x_t$ at most $k$ of the $C_{ij}$-s depend on it.*

**Remark:** Note that unlike the definition of $\Sigma^{[k]}\Pi^{[\ell]}\mathcal{C}$ formulas, there is **no restriction** on the number of additions or multiplications taken for occur-$k$ $\Sigma\Pi\mathcal{C}$ formulas.
**Remark:** Note that $k$ and $\ell$ are upper bounds since some $C_{ij}$ can be taken to be field elements. That is, $\Sigma^{[k]}\Pi^{[\ell]}\mathcal{C}$ corresponds to taking sums of (at most) $k$ products of (at most) $\ell$ polynomials from $\mathcal{C}$.

### 3.3.1 Previous Results

We will now state a collection of known results/algorithms that will be useful in our final proof.

**Lemma 3.9** ([Alo99])**.** *Let $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a polynomial. Suppose that for every $i \in [n]$ the individual degree of $x_i$ is bounded by $d_i$, and let $S_i \subseteq \mathbb{F}$ be such that $|S_i| > d_i$. We denote $S = S_1 \times S_2 \times \cdots \times S_n$. Then, $f \equiv 0$ iff $f|_S \equiv 0$.*

**Lemma 3.10** ([Sch80, Zip79, DL78])**.** *Let $f(x_1, ..., x_n)$ be a nonzero polynomial of degree at most $d$, and let $S \subseteq \mathbb{F}$. If we choose $\bar{a} = (a_1, \ldots, a_n) \in S^n$ uniformly at random, then $Pr[f(\bar{a}) = 0] \leq d/|S|$.*

**Lemma 3.11** ([KT90])**.** *There exists a randomized algorithm that given black-box access to a polynomial $f(\bar{x}) \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ of degree $d$ outputs its irreducible factors, in time* $\text{poly}(n, d)$.

**Lemma 3.12.** *Set-multilinear $\Sigma\Pi\Sigma(k)$ circuits are closed under factoring. Formally, if $f = g \cdot h$ and $f$ is a degree $d$ polynomial computed by a $\Sigma\Pi\Sigma_{\{\sqcup_j X_j\}}(k)$ circuit. Then $g$ (similarly $h$) is computed by a set-multilinear $\Sigma\Pi\Sigma(k)$ circuit. Also, there is a partition of $[d]$ into two disjoint sets $A_1, A_2 \subseteq$ s.t. $g$ is set-multilinear w.r.t to partition $\cup_{i\in A_1} X_i$ and $h$ is set-multilinear w.r.t. to partition $\cup_{i\in A_2} X_i$.*

*Proof.* We have that $f \in \Sigma\Pi\Sigma_{\{\sqcup_j X_j\}}(k)$, and $f = g \cdot h$. Let $X = \text{var}(f), Y = \text{var}(g)$, and $Z = \text{var}(h)$. Note that, $f$ is homogeneous by definition, and thus $g, h$ are homogeneous as well. Further, $Y \cap Z = \phi$.

We claim that for all $i \in [d]$ either $X_i \cap Y \neq \emptyset$ or $X_i \cap Y \neq \emptyset$. Note that, assuming the claim, the lemma follows by restricting the $\Sigma\Pi\Sigma_{\{\sqcup_j X_j\}}(k)$circuit computing $f$ to $Y$ (by setting $Z$ variables randomly) to give a circuit for $g$.

To prove the claim, suppose $\exists i$ s.t. $x_1 \in Y \cap X_i$ and $x_2 \in Z \cap X_i$. Let $g = a_1x_1 + a_0$ and $h = b_1x_2 + b_0$, where $a_1, b_1 \in \mathbb{F}[X \setminus \{x_1, x_2\}] \neq 0$. This gives that $f = g \cdot h$ has a monomial that depends on both $x_1, x_2$ which contradicts the set-multilinearity of $f$.

□

As a corollary, we can efficiently simulate a black-box access to $\text{sim}(C)$ given a black-box access to a multilinear or set-multilinear $C$. The reason this holds is that by the above lemma it follows that once we strip away all the linear factors dividing the underlying polynomial, what remains is still a multilinear/set-multilinear $\Sigma\Pi\Sigma(k)$ circuit which is also now simple.

**Corollary 3.13.** *There is a randomized algorithm that given a black-box access to a multilinear/set-multilinear $\Sigma\Pi\Sigma(k)$ circuit $C$ outputs linear functions $L_1, \ldots, L_r$ and black-box access to a simple multilinear/set-multilinear $\Sigma\Pi\Sigma(k)$ circuit $\hat{C}$ such that $C = \prod_{i=1}^{r} L_i \cdot \hat{C}$, in time* $\text{poly}(n)$.

### 3.3.2 Variable Reduction

In this section, we discuss how to reduce the number of variables in a polynomial. Before describing this procedure we have to formally define the notion of the number of essential variables in a polynomial.

**Definition 3.14** (Number of essential variables)**.** *For $f(\bar{x}) \in \mathbb{F}[\bar{x}]$, we will say that the number of essential variables in $f(\bar{x})$ is $t$ if there exist an invertible linear transformation $A \in \mathbb{F}^{n\times n}$ s.t. $f(A\bar{x})$ just depends on $t$ variables. Moreover any such transformation doesn't exist for mapping to fewer than $t$ variables.*

The next lemma is from the work of Carlini [Car06], adopted by Kayal [Kay11] in the language of circuits. This lemma eliminates redundant variables from a polynomial and plays a crucial role in our reconstruction results.

We state the lemma below in the setting of black-box access to the input polynomial. The original version of the lemma was in the white-box setting, but by inspecting the proof in [Kay11] one can see that it works in the black-box setting as well by noting that given black-box access to a circuit computing a polynomial $f$, one can get black-box access to the circuits computing its first order partial derivatives.

**Lemma 3.15.** *[Kay11, Car06] Given black-box access to an $n$-variate polynomial $f(X) \in \mathbb{F}[X]$ of degree $d$ with $m$ essential variables, s.t. $char(\mathbb{F}) > d$ or $0$, there is a randomized $\mathrm{poly}(n,d)$ time algorithm that computes an invertible linear transformation $A \in \mathbb{F}^{(n\times n)}$ such that $f(A\cdot\bar{x})$ depends on the first $m$-variables only.*

We will also need the following generalization of Carlini's Lemma. The proof is similar to the proof of previous lemma.

For any $Y \subseteq X$, and $f(X) \in \mathbb{F}[X]$ define $\langle \frac{\partial f}{\partial Y}\rangle := \mathrm{Span}_{\mathbb{F}}\left(\frac{\partial f}{\partial x_i} \mid x_i \in Y\right)$.

**Lemma 3.16.** *Given black-box access to an $n$-variate, degree $d$ polynomial $f(X,Y) \in \mathbb{F}[X,Y]$, s.t. $X = \{x_1, x_2, \ldots, x_t\}$ and $char(\mathbb{F}) > d$ or $0$. Suppose $\dim_{\mathbb{F}}\left(\langle \frac{\partial f}{\partial X}\rangle\right) = m \leq t$, then there is a randomized $\mathrm{poly}(n,d)$ time algorithm that computes an invertible linear transformation $A \in \mathbb{F}^{t\times t}$ such that $f(A\cdot\bar{x}, Y) \in \mathbb{F}[x_1, \ldots, x_m, Y]$.*

*Proof.* We have that $\dim_{\mathbb{F}}\left(\langle \frac{\partial f}{\partial X}\rangle\right) = m$. Let $m < t$, otherwise, we have nothing to show. Let $u_{m+1}, , \ldots, u_t \in \mathbb{F}^t$ be the basis vector of the null-space of $\langle \frac{\partial f}{\partial X}\rangle$. And, $u_1, u_2, \ldots u_t \in \mathbb{F}^t$ be vectors s.t. $u_1, u_2, \ldots, u_t \in \mathbb{F}^t$ is a basis for $\mathbb{F}^t$. Let $A \in \mathbb{F}^{t\times t}$ be the matrix with $i$-th columns $u_i$. Note that, we can compute the null-space of $\langle \frac{\partial f}{\partial X}\rangle$ (and thus the matrix A) in $\mathrm{poly}(n,d)$-time using Lemma 4.9.

By the chain rule we have,

$$\begin{pmatrix} \frac{\partial f(A\cdot\bar{x},Y)}{\partial x_1} \\ \vdots \\ \frac{\partial f(A\cdot\bar{x},Y)}{\partial x_t} \end{pmatrix} = A^T \begin{pmatrix} \frac{\partial f}{\partial x_1}(AX,Y) \\ \vdots \\ \frac{\partial f}{\partial x_t}(AX,Y) \end{pmatrix}.$$

This gives that, $\frac{\partial f(\cdot\bar{x},Y)}{\partial x_i} = u_i \cdot \left(\frac{\partial f}{\partial x_1}(A\cdot\bar{x},Y), \cdots, \frac{\partial f}{\partial x_t}(A\cdot\bar{x},Y)\right)$. Since $A$ is invertible, we have that $u_i \cdot \left(\frac{\partial f}{\partial x_1}(A\cdot\bar{x},Y), \cdots, \frac{\partial f}{\partial x_t}(A\cdot\bar{x},Y)\right) = 0 \iff u_i \cdot \left(\frac{\partial f}{\partial x_1}(X,Y), \cdots, \frac{\partial f}{\partial x_t}(X,Y)\right) = 0$. Recall for $i > m$, $u_i \in$ null-space of $\langle \frac{\partial f}{\partial X}\rangle$. Thus, $\frac{\partial f(A\cdot\bar{x},Y)}{\partial x_j} = 0$, for $j > m$.

□

# 4 Polynomial Linear Dependence & Alternants

**Definition 4.1** (Polynomial Linear Dependence)**.** *We say that the polynomials $f_1, \ldots, f_k \in \mathbb{F}[X]$ are* $\mathbb{F}$-linearly dependent*, if there exist field constants $\alpha_1, \ldots, \alpha_k \in \mathbb{F}$, not all zeros, such that $\alpha_1\cdot f_1 + \ldots + \alpha_k \cdot f_k \equiv$*

0. *We define the notion of* linear independence *accordingly. In particular, note that a set of* linearly-independent *polynomials cannot contain the identically zero polynomial.*

*We can extend this notion by considering linear dependence over a field of rational functions $\mathbb{F}(S)$ for some set of variables $S$.*

*Let $S$ be a set of variables. We say that the polynomials $f_1, \ldots, f_k \in \mathbb{F}[X]$ are* $\mathbb{F}(S)$-linearly dependent*, if there exist rational functions $H_1(S), \ldots, H_k(S) \in \mathbb{F}(S)$, not all zeros, such that $H_1(S) \cdot f_1 + \ldots + H_k(S) \cdot f_k \equiv 0$. We define* linear independence *accordingly.*

The following properties are immediate extensions of the definition.

**Lemma 4.2.** *Let $\mathbf{f} = \{f_1, \ldots, f_k\} \subseteq \mathbb{F}[X]$ be a set of $\mathbb{F}(S)$-linearly independent polynomials for a set of variables $S$ (could be empty). Then*

- *Let $T$ be a set of variables* disjoint *to $X$ (i.e. $X \cap T = \emptyset$). Then the polynomials in $\mathbf{f}$ are also $\mathbb{F}(S \cup T)$-linearly independent.*
- *Let $x_j \in S$. Then the polynomials $f_1, \ldots, f_k$ are also $(\mathbb{F}(S \setminus \{x_j\}))(x_j)$-linearly independent and vice versa.*

We now state and prove our main technical result: any set of linearly independent polynomials has a small "witness set" for their independence. More formally, a set of $k$ linearly independent polynomials remain linearly independent even if we regard them as polynomials in a chosen set $S$ of (at most) $k-1$ variables. Equivalently, one can think about random projections. That is, for any set of $k$ linearly independent polynomials there exists a set $S$ of at most $k-1$ variables such that the polynomials remain linearly independent, even if we sets all the variables **outside** of $S$ to random values.

**Lemma 4.3** (SWLI Lemma)**.** *Let $\mathbb{F}$ be an arbitrary field and let $\mathbf{f} = \{f_1, \ldots, f_k\} \subseteq \mathbb{F}[X]$ be a set of $\mathbb{F}$-linearly independent polynomials. Then there exists a subset $S \subseteq X$ of size $|S| \leq k-1$ such that the polynomials in $\mathbf{f}$ are (also) $\mathbb{F}(X \setminus S)$-linearly independent.*

To provide more intuition, we first prove a weaker statement. In fact, this statement will be used as a "base case" subsequently.

**Lemma 4.4.** *Let $\mathbb{F}$ be an arbitrary field and let $f_1, f_2 \in \mathbb{F}[X]$ such that $|X| \geq 2$. Suppose that the polynomials $f_1, f_2$ are $\mathbb{F}$-linearly independent. Then there exists a variable $x_i \in X$ such that the polynomials are (also) $\mathbb{F}(x_i)$-linearly independent.*

*Proof.* We prove something somewhat stronger: for any pair of variables $x_i \neq x_j \in X$ the polynomials $f_1$ and $f_2$ are either $\mathbb{F}(x_i)$-linearly independent or $\mathbb{F}(x_j)$-linearly independent or both. Assume the contrary. By the definition, there exist $u_1(x_i), u_2(x_i) \in \mathbb{F}(x_i)$ and $v_1(x_j), v_2(x_j) \in \mathbb{F}(x_j)$, not all zeros, such that:

$$u_1(x_i) \cdot f_1 + u_2(x_i) \cdot f_2 \equiv 0$$
$$v_1(x_j) \cdot f_1 + v_2(x_j) \cdot f_2 \equiv 0$$

Further, observe that $u_1, u_2, v_1, v_2$ are all nonzero as otherwise $f_1 \equiv 0$ or $f_2 \equiv 0$ Hence, we obtain that:

$$-u_1(x_i)/u_2(x_i) \cdot f_1 = f_2$$
$$-v_1(x_j)/v_2(x_j) \cdot f_1 = f_2$$

As (again) $f_1 \not\equiv 0$ we obtain that

$$u_1(x_i)/u_2(x_i) = v_1(x_j)/v_2(x_j).$$

As $u_1(x_i)/u_2(x_i) \in \mathbb{F}(x_i)$ and $v_1(x_j)/v_2(x_j) \in \mathbb{F}(x_j)$, both expressions must in fact lie in the base field $\mathbb{F}$. This, in turn, implies that $f_1$ and $f_2$ are $\mathbb{F}$-linear dependent, leading to a contradiction. □

We now extend the claim to an arbitrary number of polynomials.

**Lemma 4.5.** *Let $\mathbb{F}$ be an arbitrary field and let $f_1, \ldots, f_k \in \mathbb{F}[X]$ such that $|X| \geq k \geq 2$. Suppose that the polynomials $f_1, \ldots, f_k$ are $\mathbb{F}$-linearly independent. Then there exists a variable $x_i \in X$ such that the polynomials are (also) $\mathbb{F}(x_i)$-linearly independent.*

*Proof.* The proof is by induction on $k$. The base case $k = 2$ is covered by Lemma 4.4. Suppose $k \geq 3$ and consider $f_1 \ldots f_{k-1}$. As $|X| \geq k > k-1$, by the induction hypothesis, there exists $x_i \in X$ such that the polynomials $f_1 \ldots f_{k-1}$ $\mathbb{F}(x_i)$-linearly independent.

Set $Y \triangleq X \setminus \{x_i\}$ and $\mathbb{E} \triangleq \mathbb{F}(x_i)$ and let us now regard the polynomials $f_1 \ldots f_{k-1}$ as polynomials in the variables $Y$ over the field $\mathbb{E}$. That is, as polynomials over $\mathbb{E}[Y]$. From the above, the polynomials are $\mathbb{E}$-linearly independent. And now we make a crucial observation that $|Y| = |X| - 1 \geq k - 1$ and since the inductive hypothesis applies to **all** fields, we can invoke it again to obtain that there exists $x_j \in Y$ (different from $x_i$) such that the polynomials $f_1 \ldots f_{k-1}$ are $\mathbb{E}(x_j)$-linearly independent. By Lemma 4.2, these polynomials are (also) $\mathbb{F}(x_i, x_j)$-linearly independent.

Now consider all $k$ polynomials $f_1 \ldots f_k$. To complete the argument, we claim that these polynomials are either $\mathbb{F}(x_i)$-linearly independent or $\mathbb{F}(x_j)$-linearly independent (for the same choice of $x_i$ and $x_j$ above). The proof is by contradiction and is similar to the proof of Lemma 4.4. We give it here for completeness.

If possible the $k$ polynomials are linearly dependent over $\mathbb{F}(x_i)$ and over $\mathbb{F}(x_j)$. By the definition, there exist $u_1(x_i), \ldots, u_k(x_i) \in \mathbb{F}(x_i)$ and $v_1(x_j), \ldots, v_k(x_j) \in \mathbb{F}(x_j)$, not all zeros, such that:

$$u_1(x_i) \cdot f_1 + \ldots + u_{k-1}(x_i) \cdot f_{k-1} + u_k(x_i) \cdot f_k \equiv 0$$
$$v_1(x_j) \cdot f_1 + \ldots + v_{k-1}(x_j) \cdot f_{k-1} + v_k(x_j) \cdot f_k \equiv 0$$

First, observe that $u_k(x_i), v_k(x_j) \not\equiv 0$ as otherwise the polynomials $f_1 \ldots f_{k-1}$ would be $\mathbb{F}(x_i, x_j)$-linearly dependent. Hence, we obtain:

$$-u_1(x_i)/u_k(x_i) \cdot f_1 - \ldots - u_{k-1}(x_i)/u_k(x_i) \cdot f_{k-1} = f_k$$
$$-v_1(x_j)/v_k(x_j) \cdot f_1 - \ldots - v_{k-1}(x_j)/v_k(x_j) \cdot f_{k-1} = f_k$$

As $f_k$ has a unique representation as a linear combination of $\mathbb{F}(x_i, x_j)$-linearly independent polynomials $f_1 \ldots f_{k-1}$ we obtain that:

$$\forall \ell \in [k-1] : u_\ell(x_i)/u_k(x_i) = v_\ell(x_j)/v_k(x_j).$$

Consequently, all these expressions lie in the base field $\mathbb{F}$. This, in turn, implies that $f_1, \ldots, f_k$ are $\mathbb{F}$-linearly dependent, leading to a contradiction. □

*Proof of Lemma 4.3.* We will proceed by induction on $|X|$. The base case is when $|X| < k$. In this case we can simply take $S = X$. Now suppose that $|X| \geq k$. By Lemma 4.5, there exist a variable $x_i \in X$ such that the polynomials in $\mathbf{f}$ are $\mathbb{F}(x_i)$-linearly independent. Set $Y \triangleq X \setminus \{x_i\}$ and $\mathbb{E} \triangleq \mathbb{F}(x_i)$ and let us now regard the polynomials in $\mathbf{f}$ as polynomials in the variables $Y$ over the field $\mathbb{E}$. That is, as polynomials over $\mathbb{E}[Y]$. From the above, the polynomials are $\mathbb{E}$-linearly independent. By the inductive hypothesis, there exists a subset $S \subseteq Y$ of size $|S| \leq k - 1$ such that polynomials in $\mathbf{f}$ are $\mathbb{E}(Y \setminus S)$-linearly independent. In other words, the polynomials are $(\mathbb{F}(x_i))(X \setminus (S \cup \{x_i\}))$-linearly independent. By Lemma 4.2, this is equivalent to being $\mathbb{F}(X \setminus S)$-linearly independent. □

## 4.1 Alternants

Alterant Matrices (or alternants, for short) have played an important role in many application such as the *alternant codes*.

**Definition 4.6.** *Let $\mathbf{f} = (f_1, f_2, \dots f_k)$, where $f_i(X) \in \mathbb{F}[X]$, be a vector of polynomials over a field $\mathbb{F}$ and let $X_1, \dots, X_k$ be disjoint sets of variables each of size $|X|$ (i.e. $k$ disjoint copies of $X$). We define the corresponding* alternate matrix *(or alternant, for short) as:*

$$\mathrm{Alt}_{\mathbf{f}}(X_1, X_2, \dots, X_k) \triangleq \begin{pmatrix} f_1(X_1) & f_2(X_1) & \cdots & \cdots & f_k(X_1) \\ f_1(X_2) & f_2(X_2) & \cdots & \cdots & f_k(X_2) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ f_1(X_k) & f_2(X_k) & \cdots & \cdots & f_k(X_k) \end{pmatrix}.$$

*For our purposes, we would like to consider a simple extension of the alternant that can be thought of as a "partial" alternant. That is, we treat a subset of the variable as field elements and regard the polynomials as polynomials in the remaining variables. Formally, let $\mathbf{f} = (f_1, f_2, \dots f_k)$, where $f_i(X, Z) \in \mathbb{F}[X, Z]$, be a vector of polynomials over a field $\mathbb{F}$ and let $X_1, \dots, X_k$ be disjoint sets of variables each of size $|X|$ (i.e. $k$ disjoint copies of $X$). We define the corresponding* partial alternate *(or alternant, for short) as:*

$$\mathrm{Alt}^X_{\mathbf{f}}(X_1, X_2, \dots, X_k, Z) \triangleq \begin{pmatrix} f_1(X_1, Z) & f_2(X_1, Z) & \cdots & \cdots & f_k(X_1, Z) \\ f_1(X_2, Z) & f_2(X_2, Z) & \cdots & \cdots & f_k(X_2, Z) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ f_1(X_k, Z) & f_2(X_k, Z) & \cdots & \cdots & f_k(X_k, Z) \end{pmatrix}.$$

The following is a fundamental property of the alternant which captures linear (in)dependence of polynomials. The special case where $Z$ is the empty set is also very interesting. A proof can be found in [Kay11, Claim 7]. For completeness, see Section A in the Appendix.

**Lemma 4.7.** *Let $\boldsymbol{f} = \{f_1(X, Z), f_2(X, Z), \dots, f_k(X, Z)\} \subseteq \mathbb{F}[X, Z]$. Then the polynomials in $\boldsymbol{f}$ are $\mathbb{F}(Z)$-linearly independent iff* $\det(\mathrm{Alt}^X_{\mathbf{f}}) \not\equiv 0$.

## 4.2 $\mathbb{F}$-Linear Dependencies

**Definition 4.8.** *Let $\boldsymbol{f} := (f_1, f_2, \dots f_m)$, where $f_i(X) \in \mathbb{F}[X]$, be a vector of polynomials over a field $\mathbb{F}$. The set of $\mathbb{F}$-linear dependencies in $\boldsymbol{f}$, denoted $\boldsymbol{f}^\perp$, is the set of all vectors $\boldsymbol{v} \in \mathbb{F}^m$ whose inner product with $\boldsymbol{f}$ is the zero polynomial, i.e.,*

$$\boldsymbol{f}^\perp = \{(a_1, \dots, a_m) \in \mathbb{F}^m : a_1 f_1(X) + \dots + a_m f_m(X) = 0\}.$$

The set $\mathbf{f}^\perp$ is clearly a linear subspace of $\mathbb{F}^m$. This notion is helpful to state and prove some useful lemmas. The main observation here is that given (by arithmetic circuits or black-box access) a collection of polynomials, then in randomized polynomial time we can find the $\mathbb{F}$-linear dependencies among these polynomials. In order to show this, we will need the following technical lemma which we state without proof but it can be found in [Kay11].

**Lemma 4.9.** *[Kay11, Lem 4.1] Given $m$ polynomials $\boldsymbol{f} = \{f_1, f_2, \dots f_m\}$, each in $\mathbb{F}[x_1, \dots, x_n]$ of degree at most $d$, either by a circuit (or black-box access) [7], s.t. with rank(maximal number of linearly independent $f_i$-s) of $\boldsymbol{f} = k$, and $|\mathbb{F}| > \binom{m}{k} \cdot dnk$ (if $|\mathbb{F}| \leq \binom{m}{k} \cdot dnk$ then we can work with an extension) then:*

1. *There is a randomized* $\mathrm{poly}(m, n, k)$ *time algorithm to compute a basis for the space $\mathbb{F}$-span$\{f_1, f_2, \dots f_m\}$. Along with the basis(say $f_{i_1}, f_{i_2}, \dots f_{i_k}$ be a basis of linear space of $f_i$-s), the aforementioned algorithm also outputs a matrix $M$ s.t.*

$$M \begin{pmatrix} f_{i_1} \\ \vdots \\ f_{i_k} \end{pmatrix} = \begin{pmatrix} f_1 \\ \vdots \\ f_m \end{pmatrix}.$$

2. *Also, there is a randomized* $\mathrm{poly}(m, n, k)$ *time algorithm that given a vector of these polynomials $\boldsymbol{f} = (f_1, f_2, \dots, f_m)$ computes a basis for the space $\boldsymbol{f}^\perp$.*

[7] The lemma statement in [Kay11] just mentions the case when a circuit is given explicitly, however it is easy to observe that even black-box/oracle access suffices.

# 5 Generalized Alternants

**Definition 5.1.** *Generalized Alternant Let $F(X,Y,Z) \in \mathbb{F}[X,Y,Z]$ and let $k \geq 1$. Let $X_1,\ldots,X_k$ and $Y_1,\ldots,Y_k$ be disjoint copies of $X$ and $Y$, respectively. Define:*

$$\mathrm{GAlt}_k^{X,Y}[F] = M = \begin{pmatrix} F(X_1,Y_1,Z) & F(X_1,Y_2,Z) & \cdots & F(X_1,Y_k,Z) \\ F(X_2,Y_1,Z) & F(X_2,Y_2,Z) & \cdots & F(X_2,Y_k,Z) \\ \vdots & \vdots & \vdots & \vdots \\ F(X_k,Y_1,Z) & F(X_k,Y_2,Z) & \cdots & F(X_k,Y_k,Z) \end{pmatrix}.$$

*Formally, let $\{M_{ij}\}_{\{i,j\in[k]\}} = F(X_i,Y_j,Z)$.*

The main property, as well as, the relation to the "classical" alternant is given in the following lemma.

**Lemma 5.2.** *Let $F(X,Y,Z) \in \mathbb{F}[X,Y,Z]$ be a polynomial that can be expressed in the form:*

$$F(X,Y,Z) = \sum_{t=1}^{k} R_t(X,Z) \cdot Q_t(Y,Z) \cdot H_t(Z)$$

*for some polynomials $\{R_t,Q_t,H_t\}_{t\in[k]}$. Let $\mathbf{R} \triangleq \{R_1(X,Z),\ldots,R_k(X,Z)\}$ and $\mathbf{Q} \triangleq \{Q_1(Y,Z),\ldots,R_k(Y,Z)\}$. Then*

$$\mathrm{GAlt}_k^{X,Y}[F] = \mathrm{Alt}_{\mathbf{R}}^{X} \cdot \begin{pmatrix} H_1(Z) & & & \\ & \ddots & & \\ & & \ddots & \\ & & & H_k(Z) \end{pmatrix} \cdot (\mathrm{Alt}_{\mathbf{Q}}^{Y})^t$$

*and consequently,* $\det(\mathrm{GAlt}_k^{X,Y}[F]) = \det(\mathrm{Alt}_{\mathbf{R}}^{X}) \cdot \det(\mathrm{Alt}_{\mathbf{Q}}^{Y}) \cdot H_1(Z) \cdot \ldots \cdot H_k(Z)$.

*Proof.* By inspection.

$$\begin{pmatrix} R_1(X_1,Z) & R_2(X_1,Z) & \cdots & R_k(X_1,Z) \\ R_1(X_2,Z) & R_2(X_2,Z) & \cdots & R_k(X_2,Z) \\ \vdots & \vdots & \vdots & \vdots \\ R_1(X_k,Z) & R_2(X_k,Z) & \cdots & R_k(X_k,Z) \end{pmatrix} \cdot \begin{pmatrix} H_1(Z) & & & \\ & \ddots & & \\ & & \ddots & \\ & & & H_k(Z) \end{pmatrix} \cdot \begin{pmatrix} Q_1(Y_1,Z) & Q_1(Y_2,Z) & \cdots & Q_1(Y_k,Z) \\ Q_2(Y_1,Z) & Q_2(Y_2,Z) & \cdots & Q_2(Y_k,Z) \\ \vdots & \vdots & \vdots & \vdots \\ Q_k(Y_1,Z) & Q_k(Y_2,Z) & \cdots & Q_k(Y_k,Z) \end{pmatrix}$$

□

The following is our second technical contribution: an algorithm that can learn "many" parts of a hidden circuit under some technical conditions.

**Theorem 5.3.** *Suppose $F(W) \in \mathbb{F}[W]$, with $|W| = n$ is a polynomial of degree $d$ such there exists a partition $W = X \cup Y \cup Z$ for which $F$ can be written as*

$$F(X,Y,Z) = \sum_{t=1}^{k} R_t(X) \cdot Q_t(Y) \cdot H_t(Z)$$

*In addition, suppose that the polynomials in $\mathbf{R} \triangleq \{R_1(X),\ldots,R_k(X)\}$ and $\mathbf{Q} \triangleq \{Q_1(Y),\ldots,Q_k(Y)\}$ are $\mathbb{F}$-linearly independent. Then there exists a randomized algorithm that given a black-box access to $F$ and $k$ as input, outputs a list of polynomials that contains (among other things) all the irreducible factors of $H_1(Z) \cdot \ldots \cdot H_k(Z)$, in time $n^{\mathcal{O}(k)} \cdot \mathrm{poly}(d)$.*

*Proof.* Consider the following algorithm:

For each pair of subsets $X', Y' \subseteq W$ such that $X' \cap Y' = \emptyset$ and $|X'|, |Y'| \leq k-1$ :

- Let $Z' \triangleq W \setminus \{X' \cup Y'\}$. Write $F$ as $F(X', Y', Z')$
- Compute $\mathrm{GAlt}_k^{X',Y'}[F]$
- Factor $\det(\mathrm{GAlt}_k^{X',Y'}[F])$ as a multivariate polynomial over $X'_1, \ldots X'_k, Y'_1, \ldots, Y'_k, Z'$ using the algorithm from Lemma 3.11.
- Add factors to the list.

**Analysis:** By Lemma 4.3, there exist subsets $S \subseteq X$ and $T \subseteq Y$ of sizes $|S|, |T| \leq k-1$ such that the polynomials in $\mathbf{R}$ and $\mathbf{Q}$ are $\mathbb{F}(X \setminus S)$-linearly independent and $\mathbb{F}(Y \setminus T)$-linearly independent, respectively. By Lemma 4.2, the polynomials in both $\mathbf{R}$ and $\mathbf{Q}$ are $\mathbb{F}((X \setminus S) \cup (Y \setminus T) \cup Z)$-linearly independent.

Suppose that algorithm "guessed" the sets $X' = S$ and $Y' = T$. By definition, $Z' = W \setminus \{X' \cup Y'\} = (X \setminus S) \cup (Y \setminus T) \cup Z$. Hence, we have that:

- $X \subseteq S \cup Z' = X' \cup Z'$. Thus for all $t \in [k] : R_t \in \mathbb{F}[X', Z']$
- $Y \subseteq T \cup Z' = Y' \cup Z'$. Thus for all $t \in [k] : Q_t \in \mathbb{F}[Y', Z']$
- $Z \subseteq Z'$. Thus for all $t \in [k] : H_t \in \mathbb{F}[Z']$

Consequently, $F$ can be expressed in the form: $F(X', Y', Z') = \sum_{t=1}^{k} R_t(X', Z') \cdot Q_t(Y', Z') \cdot H_t(Z')$ where the polynomials in both $\mathbf{R}$ and $\mathbf{Q}$ are $\mathbb{F}(Z')$-linearly independent. By Lemma 5.2:

$$\det(\mathrm{GAlt}_k^{X',Y'}[F]) = \det(\mathrm{Alt}_{\mathbf{R}}^{X'}) \cdot \det(\mathrm{Alt}_{\mathbf{Q}}^{Y'}) \cdot H_1(Z') \cdot \ldots \cdot H_k(Z')$$

By Lemma 4.7: $\det(\mathrm{Alt}_{\mathbf{R}}^{X'}), \det(\mathrm{Alt}_{\mathbf{Q}}^{Y'}) \not\equiv 0$ and hence $\det(\mathrm{GAlt}_k^{X',Y'}[F]) \not\equiv 0$. Hence the factoring algorithm will output the irreducible factors of $H_1(Z) \cdot \ldots \cdot H_k(Z)$.

**Runtime:** There are at most $\binom{n}{k}^2$ choices for $X'$ and $Y'$. For each pair $X'$ and $Y'$, evaluating $\det(\mathrm{GAlt}_k^{X',Y'}[F])$ on an input can be carried out in time $\mathrm{poly}(n)$. Finally, by Lemma 3.11 each factorization takes $\mathrm{poly}(n, d)$ time. In total, we get a $n^{\mathcal{O}(k)} \cdot \mathrm{poly}(d)$-time algorithm.

□

# 6 PIT via Alternants

Let $\mathbf{T} = \{T_1, \ldots, T_k\} \subseteq \mathbb{F}[X]$ and let $C = T_1(X) + \ldots + T_k(X) \not\equiv 0$. The main idea behind the alternants approach to PIT is based on the following observations:

- We can assume wlog that the polynomials in $\mathbf{T}$ are $\mathbb{F}$-linearly independent.
- If we project $C(X)$ to a subset of variables while ensuring that the polynomials in $\mathbf{T}$ remain linearly independent, the projected sum will be non-zero.
- Alternants capture linear independence of polynomials (see Lemma 4.7).

In conclusion, "all" we need to do is to hit the alternant. The next two lemmas formalize the above intuition.

**Lemma 6.1.** *Let $C(X) = T_1(X) + \ldots + T_k(X)$ be a circuit computing a non-zero polynomial. Then there exists a non-empty subset $A \subseteq [k]$ and non-zero field elements $\{\alpha_i\}_{i\in A} \subseteq \mathbb{F}$ such that*

1. $C(X) = \sum_{i\in A} \alpha_i T_i(X)$
2. *The polynomials $\{\alpha_i T_i(X)\}_{i\in A}$ are $\mathbb{F}$-linearly independent.*

*Proof idea.* Find a basis for $\mathbf{T}$ and express the polynomials as linear combinations in the basis. □

**Lemma 6.2.** *Let $\mathbf{T} = \{T_1(Y,Z), \ldots, T_\ell(Y,Z)\} \subseteq \mathbb{F}[Y,Z]$ and $\bar{a} \in \mathbb{F}^{|Z|}$ such that $\det(\mathrm{Alt}^Y_{\mathbf{T}}|_{Z=\bar{a}}) \not\equiv 0$. Then $\sum_{i=1}^{\ell} T_i(Y,\bar{a}) \not\equiv 0$.*

Before proceeding with the proof, we remark that the premise $\det(\mathrm{Alt}^Y_{\mathbf{T}}|_{Z=\bar{a}}) \not\equiv 0$ in particular implies that $\det(\mathrm{Alt}^Y_{\mathbf{T}}) \not\equiv 0$. By the main property of the alternant (Lemma 4.7) this means that the polynomials in $\mathbf{T}$ are assumed to $\mathbb{F}$-linearly independent.

*Proof.* Let $\tilde{\mathbf{T}} \triangleq \{T_1(Y,\bar{a}), \ldots, T_\ell(Y,\bar{a})\}$. Observe that $\mathrm{Alt}^Y_{\mathbf{T}}|_{Z=\bar{a}} = \mathrm{Alt}_{\tilde{\mathbf{T}}}$. Consequently, by Lemma 4.7, the polynomials in $\tilde{\mathbf{T}}$ are $\mathbb{F}$-linearly independent. In other words, no $\mathbb{F}$-linear combination of the polynomials in $\tilde{\mathbf{T}}$ results in the zero polynomial and in particular, $\sum_{i=1}^{\ell} 1 \cdot T_i(Y,\bar{a}) \not\equiv 0$. □

Utilizing the properties of generators and the map $G_{n,k}$ (see Lemma 3.5), we can extend the claim to the case when the actual partition $X = Y \cup Z$ is *unknown* and we rather have an upper bound on the size of $Y$.

**Lemma 6.3.** *Suppose $X = Y \cup Z$ such that $|Y| \leq b$. Denote $|X| = n$. Let $\mathbf{T} = \{T_1(Y,Z), \ldots, T_\ell(Y,Z)\} \subseteq \mathbb{F}[Y,Z]$ and $\mathcal{G}_n = (\mathcal{G}^Y_n, \mathcal{G}^Z_n) : \mathbb{F}^r \to \mathbb{F}^n$ be a polynomial map such that $\det(\mathrm{Alt}^Y_{\mathbf{T}}|_{Z=\mathcal{G}^Z_n}) \not\equiv 0$.*
*Then $\sum_{i=1}^{\ell} T_i(\mathcal{G}_n + G_{n,b}) \not\equiv 0$.*

*Proof.* By Lemma 3.5, there exist a substitution to the variables of $G_{n,b}$ for which $\sum_{i=1}^{\ell} T_i(\mathcal{G}_n + G_{n,b})$ results in $\sum_{i=1}^{\ell} T_i(Y, \mathcal{G}^Z_n)$. Next, by definition, there exist $\bar{a} \in \mathrm{Im}\,(\mathcal{G}_n)$ such that $\det(\mathrm{Alt}^Y_{\mathbf{T}}|_{Z=\bar{a}^Z}) \not\equiv 0$. Consequently, there exists a substitution to the variables of $\sum_{i=1}^{\ell} T_i(\mathcal{G}_n + G_{n,b})$ that results in $\sum_{i=1}^{\ell} T_i(Y, \bar{a}^Z)$. By Lemma 6.2, $\sum_{i=1}^{\ell} T_i(Y, \bar{a}^Z) \not\equiv 0$ and hence $\sum_{i=1}^{\ell} T_i(\mathcal{G}_n + G_{n,b}) \not\equiv 0$. □

## 6.1 Identity Testing for Occur-$k$ Formulas

In this section we lay the foundations for the proof of our main result, Theorem 1, which states that for any circuit class $\mathcal{C}$, identity testing for occur-$k$ $\Sigma\Pi\mathcal{C}$ formulas reduces to identity testing of $\Sigma^{[(2k)!]}\Pi^{[4k^2]}\mathcal{C}$ formulas. To this end, we fix an arbitrary circuit class $\mathcal{C}$ throughout this section. The main result follows as a corollary of the following theorem given below, and the proof of the main result is given in Section 6.2.

**Theorem 6.4.** *Let $\mathcal{G}_n : \mathbb{F}^r \to \mathbb{F}^n$ be a generator for $\Sigma^{[(2k)!]}\Pi^{[4k^2]}\mathcal{C}$ formulas. Then $\mathcal{G}_n + G_{n,2k}$ is a generator for occur-$k$ $\Sigma\Pi\mathcal{C}$ formulas.*

We start by showing that a (partial) alternant of an occur-$k$ $\Sigma\Pi\mathcal{C}$ formula can be computed by a $\Sigma^{[\ell!]}\Pi^{[|Y|\cdot k]}\mathcal{C}$ formula (for parameters $|Y|$ and $\ell$ which we make explicit below), up to a product by circuits from $\mathcal{C}$ itself.

**Lemma 6.5.** *Let* $C(Y,Z) = \sum_{i=1}^{\ell} T_i(Y,Z) = \sum_{i=1}^{\ell} \prod_{j=1}^{d_j} C_{ij}(Y,Z)$ *be an occur-*$k$ $\Sigma\Pi\mathcal{C}$ *circuit (i.e.* $C_{ij} \in \mathcal{C}$*). Denote* $\mathbf{T} \triangleq \{T_1, \ldots, T_\ell\}$*. Then* $\det(\mathrm{Alt}^Y_{\mathbf{T}})$ *can be computed by a circuit of the form* $C'(Y_1, \ldots, Y_\ell, Z) \cdot C''(Z)$ *where* $C' \in \Sigma^{[\ell!]}\Pi^{[|Y|\cdot k]}\mathcal{C}$ *and* $C'' \in \prod \mathcal{C}$*.*

*Proof.* For each $i \in [\ell]$ we can write $T_i(Y,Z) = T'_i(Y,Z) \cdot U_i(Z)$, where $T'_i(Y,Z)$ is the product of all the $C_{ij}$-s in $T_i$ that depend on the variables of $Y$. Formally:

$$T'_i(Y,Z) \triangleq \prod_{\{j \mid \mathrm{var}(C_{ij}) \cap Y \neq \emptyset\}} C_{ij} \quad , \quad U_i(Z) \triangleq T_i(Y,Z)/T'_i(Y,Z).$$

We obtain the following:

$$\mathrm{Alt}^Y_{\mathbf{T}} = \begin{pmatrix} T_1(Y_1,Z) & T_2(Y_1,Z) & \cdots & \cdots & T_\ell(Y_1,Z) \\ T_1(Y_2,Z) & T_2(Y_2,Z) & \cdots & \cdots & T_\ell(Y_2,Z) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ T_1(Y_\ell,Z) & T_2(Y_k,Z) & \cdots & \cdots & T_\ell(Y_\ell,Z) \end{pmatrix} =$$

$$= \begin{pmatrix} T'_1(Y_1,Z) & T'_2(Y_1,Z) & \cdots & T'_\ell(Y_1,Z) \\ T'_1(Y_2,Z) & T'_2(Y_2,Z) & \cdots & T'_\ell(Y_2,Z) \\ \vdots & \vdots & \vdots & \vdots \\ T'_1(Y_\ell,Z) & T'_2(Y_\ell,Z) & \cdots & T'_\ell(Y_\ell,Z) \end{pmatrix} \cdot \begin{pmatrix} U_1(Z) & & & \\ & \ddots & & \\ & & \ddots & \\ & & & U_\ell(Z) \end{pmatrix} =$$

$$= \mathrm{Alt}^Y_{\mathbf{T}'} \cdot \begin{pmatrix} U_1(Z) & & & \\ & \ddots & & \\ & & \ddots & \\ & & & U_\ell(Z) \end{pmatrix}$$

Consequently, $\det(\mathrm{Alt}^Y_{\mathbf{T}}) = \det(\mathrm{Alt}^Y_{\mathbf{T}'}) \cdot \prod_{i=1}^{\ell} U_\ell(Z)$. Finally, by definition:

$$\det(\mathrm{Alt}^Y_{\mathbf{T}'}) = \sum_{\sigma \in S_\ell} \left( \mathrm{sign}(\sigma) \prod_{i=1}^{\ell} T'_i\left(Y_{\sigma(i)}, Z\right) \right).$$

Finally, observe that there are $\ell!$ permutations and each product $\prod_{i=1}^{\ell} T'_i$ contains exactly all the $C_{ij}$-s in $C$ that depend on variables of $Y$. Since $C(X)$ is an occur-$k$ formula, there are most $|Y| \cdot k$ such $C_{ij}$-s. □

Before we continue with the proof of Theorem 6.4, we note that the circuit $C'$ actually depends on multiple copies of the variables $Y$. At the same time, by Lemma 6.3, it is actually sufficient to hit the $Z$ variables. Hence, in order to maintain the same input size, for the sake of the analysis we consider $C'$ restricted to a single copy of the $Y$.

*Proof of Theorem 6.4.* Let $C(X) = \sum_{i=1}^{\ell} T_i(X) = \sum_{i=1}^{\ell} \prod_{j=1}^{d_j} C_{ij}(X)$ be an occur-$k$ $\Sigma\Pi\mathcal{C}$ circuit. If $C(X)$ is a constant, the claim follows immediately. Therefore, assume that $\mathrm{var}(C) \neq \emptyset$. Pick $x_m \in \mathrm{var}(C)$ and let wlog $T_1, \ldots, T_{\ell'}$ be the polynomials that depend on $x_m$. Since $C$ is an occur-$k$ circuit, we have that $\ell' \leq k$. Consider the circuit $C_m \triangleq C - C|_{x_m=0}$. Observe that

$$0 \not\equiv C_m = \sum_{i=1}^{\ell'} T_i - \sum_{i=1}^{\ell'} T_i|_{x_m=0}.$$

In addition, as each $T_i$ appears at most twice in $C_m$, each variable occurs at most $2k$ times. Hence, $C_m$ is occur-$2k$. Finally, by Lemma 6.1, we can assume wlog that $C_m(X)$ is computed by an occur-$2k$ $\Sigma\Pi\mathcal{C}$ formula: $C_m(X) = \sum_{i=1}^{\hat{\ell}} \hat{T}_i(X)$ such that $\hat{\ell} \leq 2k$ and the polynomials $\hat{\mathbf{T}} \triangleq \left\{\hat{T}_1(X), \ldots, \hat{T}_{\hat{\ell}}(X)\right\}$ are $\mathbb{F}$-linearly independent.

By Lemma 4.3, there exist a subset $Y \subseteq X$ of variables of size $|Y| \leq \hat{\ell}-1 \leq 2k-1$ such the polynomials in $\hat{\mathbf{T}}$ are $\mathbb{F}(X\backslash Y)$-linearly independent. Let us denote $Z \triangleq X\backslash Y$. By Lemma 4.7, $\det\left(\mathrm{Alt}^Y_{\hat{\mathbf{T}}}(Y_1, \ldots, Y_{\hat{\ell}}, Z)\right) \not\equiv 0$. Let $(\bar{a}^1, \ldots, \bar{a}^{\hat{\ell}-1}) \in \left(\mathbb{F}^{|Y|}\right)^{\hat{\ell}-1}$ be an assignment such that $\det\left(\mathrm{Alt}^Y_{\hat{\mathbf{T}}}(\bar{a}^1, \ldots, \bar{a}^{\hat{\ell}-1}, Y_{\hat{\ell}}, Z)\right) \not\equiv 0$ [8]. Define:

$$f(X) = f(Y, Z) \triangleq \det\left(\mathrm{Alt}^Y_{\hat{\mathbf{T}}}(\bar{a}^1, \ldots, \bar{a}^{\hat{\ell}-1}, Y, Z)\right).$$

By the above, $f \not\equiv 0$. Furthermore, by Lemma 6.5, $f(Y, Z)$ can be computed by a circuit of the form $C'(Y, Z) \cdot C''(Z)$ where $C' \in \Sigma^{[\hat{\ell}!]}\Pi^{[|Y|\cdot 2k]}\mathcal{C}$ and $C'' \in \prod\mathcal{C}$. By the assumption, $\mathcal{G}_n = (\mathcal{G}^Y_n, \mathcal{G}^Z_n)$ is a generator for $\Sigma^{[(2k)!]}\Pi^{[4k^2]}\mathcal{C}$ formulas and hence, by the multiplicative properties of generators (see Remark 3.3), $C''(\mathcal{G}^Z_n) \not\equiv 0$. In addition, as $|Y| \leq \hat{\ell} - 1 \leq 2k - 1$ we have that $C' \in \Sigma^{[(2k)!]}\Pi^{[4k^2]}\mathcal{C}$. This, in turn, implies that

$$\det\left(\mathrm{Alt}^Y_{\hat{\mathbf{T}}}(\bar{a}^1, \ldots, \bar{a}^{\hat{\ell}-1}, \mathcal{G}^Y_n, \mathcal{G}^Z_n)\right) = f(\mathcal{G}^Y_n, \mathcal{G}^Z_n) = C'(\mathcal{G}^Y_n, \mathcal{G}^Z_n) \cdot C''(\mathcal{G}^Z_n) \not\equiv 0$$

and in particular that $\det(\mathrm{Alt}^Y_{\hat{\mathbf{T}}}|_{Z=\mathcal{G}^Z_n}) \not\equiv 0$. Consequently, by Lemma 6.3:

$$C_m\left(\mathcal{G}_n + G_{n,2k-1}\right) = \sum_{i=1}^{\hat{\ell}} \hat{T}_i(\mathcal{G}_n + G_{n,2k-1}) \not\equiv 0.$$

Finally, recall that $C_m \triangleq C - C|_{x_m=0}$ and hence by the properties of $G_{n,k}$ (see Lemma 3.5): we have that

$$C(\mathcal{G}_n + G_{n,2k}) = C(\mathcal{G}_n + G_{n,2k-1} + G_{n,1}) \not\equiv 0$$

as required. □

## 6.2 Proofs of Theorems 1,2 and 3

The proof of Theorem 1 follows from Theorem 6.4 by applying the standard connections between hitting sets and generators. We provide a sketch of the proof below for completeness.

**Theorem 6.6** (Theorem 1 restated)**.** *Let $n, k, s \in \mathbb{N}$ and let $\mathbb{F}$ be an arbitrary field. Let $\mathcal{H}$ be a hitting set for $\Sigma^{[(2k)!]}\Pi^{[4k^2]}\mathcal{C}$ formulas of size $s$ over $\mathbb{F}[x_1, x_2, \ldots, x_n]$. Then there exists a deterministic algorithm that given $n, k, s$ outputs a hitting set $\mathcal{H}'$ of size $|\mathcal{H}'| = |\mathcal{H}|^2 \cdot s^{\mathcal{O}(k)}$ for occur-$k$ $\Sigma\Pi\mathcal{C}$ formulas of size $s$ over $\mathbb{F}[x_1, x_2, \ldots, x_n]$,*

*Proof.* First, we invoke the algorithm from Lemma 3.2 with $\mathcal{H}$ and $\delta = s$ to construct a generator $\mathcal{G}_n : \mathbb{F}^r \to \mathbb{F}^n$ for $\Sigma^{[(2k)!]}\Pi^{[4k^2]}\mathcal{C}$ formulas of size $s$ such that $r = \lceil \log_s |\mathcal{H}| \rceil$ and each $\mathcal{G}^i_n$ has individual degree at most $s-1$. Let $f$ be a polynomial computed by an occur-$k$ $\Sigma\Pi\mathcal{C}$ formula. By Theorem 6.4, $f(\mathcal{G}_n + G_{n,2k}) \not\equiv 0$. The individual degree of each of the $r$ variables of $\mathcal{G}_n$ is at most $s(s-1) < s^2$ and the individual degree of each of the $4k$ variables of $G_{n,2k}$ is at most $ns < s^2$. Let $\mathcal{H}'$ be a result of evaluating $\mathcal{G}_n + G_{n,2k}$ on a set $V^{r+4k}$ when $V \subseteq \mathbb{F}$ is of size $|V| = s^2$. By Lemma 3.9: $f|_{\mathcal{H}'} \not\equiv 0$. Finally, observe that $|\mathcal{H}'| = \left|s^2\right|^{r+4k} = |\mathcal{H}|^2 \cdot s^{\mathcal{O}(k)}$. □

Theorems 2 and 3 follow from Theorem 1 by instantiating the black-box PIT results for sparse polynomials [KS01] and $\Sigma\Pi\Sigma(k)$ circuits [SS12], respectively.

[8] As was mentioned earlier, we are only using these $\bar{a}^i$-s for analysis and will not be required to find them explicitly.

**Theorem 6.7** (Theorem 2 restated)**.** *Let $n, k, s \in \mathbb{N}$ and let $\mathbb{F}$ be an arbitrary field. Suppose that $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ is a polynomial computed by an occur-$k$ depth-4 $\Sigma\Pi\Sigma\Pi$ formula of size $s$. Then there exists a deterministic algorithm that given $n, k, s$ and black-box access to $f$ decides if $f \equiv 0$, in time $s^{\mathcal{O}(k^2)}$.*

**Lemma 6.8** ([KS01])**.** *There is a deterministic algorithm that given black-box access to a sparse polynomial of size[9] $s$ decides if $C \equiv 0$, in time* $\text{poly}(s)$

*Proof of Theorem 2.* If $\mathcal{C}$ is a sparse polynomial of size $s$ then a $\Sigma^{[(2k)!]}\Pi^{[4k^2]}\mathcal{C}$ formula is a sparse polynomial of size at most $s^{\mathcal{O}(k^2)}$. By the above result, the appropriate hitting set $\mathcal{H}$ will be of size $s^{\mathcal{O}(k^2)}$. By Theorem 1: $|\mathcal{H}'| = |\mathcal{H}|^2 \cdot s^{\mathcal{O}(k)} = s^{\mathcal{O}(k^2)}$. □

**Theorem 6.9** (Theorem 3 restated)**.** *Let $n, k_1, k_2, s \in \mathbb{N}$ and let $\mathbb{F}$ be an arbitrary field. Suppose that $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ is a polynomial computed by an occur-$k_1$ depth-5 $\Sigma\Pi(\Sigma^{[k_2]}\Pi\Sigma)$ formula of size $s$. Then there exists a deterministic algorithm that given $n, k_1, k_2, s$ and a black-box access to $f$ decides if $f \equiv 0$, in time $s^{k_2^{\mathcal{O}(k_1^2)}}$.*

**Lemma 6.10** ([SS12])**.** *There is a deterministic algorithm that given black-box access to a $\Sigma\Pi\Sigma(k)$[10] circuit $C$ of size $s$ decides if $C \equiv 0$, in time $s^{\mathcal{O}(k)}$.*

*Proof of Theorem 3.* The main observation is that, if $\mathcal{C}$ is a $\Sigma\Pi\Sigma(k_2)$ circuit of size $s$, then a $\Sigma^{[(2k)!]}\Pi^{[4k^2]}\mathcal{C}$ formula can expressed as a $\Sigma\Pi\Sigma(k_2^{\mathcal{O}(k_1^2)})$ circuit of size at most $s^{\mathcal{O}(k_1^2)}$. By the above result, the appropriate hitting set $\mathcal{H}$ will be of size $\left(s^{\mathcal{O}(k_1^2)}\right)^{(k_2^{\mathcal{O}(k_1^2)})} = s^{k_2^{\mathcal{O}(k_1^2)}}$. By Theorem 1: $|\mathcal{H}'| = |\mathcal{H}|^2 \cdot s^{\mathcal{O}(k)} = s^{k_2^{\mathcal{O}(k_1^2)}}$. □

# 7 Reconstructing Set-Multilinear $\Sigma\Pi\Sigma(k)$ Circuits with Unknown Variable Partition

In this section, we will show how to learn low-degree set-multilinear circuits with unknown partitions. The following observation gives a necessary and sufficient polynomial system for a polynomial to have a set-multilinear $\Sigma\Pi\Sigma$ representation. We get these equations by adding the constraints that each product gate has factors that are variable disjoint, and the variable partition is consistent across all product gates.

**Observation 7.1.** *Let $t, d, k, n \in \mathbb{N}_{>0}$, and $\{C_\ell\}_{\ell \in [t]}$ be a set of depth-3 circuits, such that $C_\ell \equiv \sum_{i \in [k]} \prod_{j \in [d]} \sum_{m \in [n]} a_{i,j,\ell}^{(m)} x_m$.*

- *For any fixed $\ell \in [t]$, if $\forall i, i', k, j \neq j'$, $a_{ij}^{(k)} a_{i'j'}^{(k)} = 0$. Then $C_\ell$ is a set-multilinear circuit. And, the converse holds up to a relabelling of linear forms in each product gate.*

- *Further, if $\forall i, i', \ell, \ell', m, j \neq j'$, $a_{i,j,\ell}^{(m)} \cdot a_{i',j',\ell'}^{(m)} = 0$. Then, $\{C_\ell\}_{\ell \in [t]}$ are set-multilinear circuits with respect to a same partition of variables. Again, the converse holds up to a relabelling of linear forms in each product gate (in each $C_\ell$).*

  *Note that the conditions for checking set-multilinearity are captured by at most $k^2 \cdot t^2 \cdot n \cdot d^2$ quadratic equations in $k \cdot t \cdot n \cdot d$ unknowns.*

Let $\text{Sys}_{\mathbb{F}}(n, m, d)$ denote the randomized time complexity of finding a solution $\in \mathbb{F}^n$ (or a suitable extension) to a system of $m$ polynomial equations $\in \mathbb{F}[x_1, \ldots, x_n]$ of total degree $d$ (if one exists). This is a well-studied problem and we refer the reader to Section 3.8 in [BSV21] for a detailed discussion. For our purpose, we will use the following bound $\text{Sys}(n, m, d) = \text{poly}((nmd)^{n^n})$, which follows directly from Theorem 3.36 in [BSV21].

[9] A sparse polynomial of size $s$ has at most $s$ non-zero terms and is of degree at most $s$

[10] Also denoted as $\Sigma^{[k]}\Pi\Sigma$

## 7.1 Reconstructing Low-Degree Set-Multilinear $\Sigma\Pi\Sigma(k)$ Circuits

As a basic step in reconstructing general set-multilinear $\Sigma\Pi\Sigma(k)$ circuits, we show how to reconstruct set-multilinear $\Sigma\Pi\Sigma(k)$ circuits efficiently when the degree of the computed polynomial is small.

**Lemma 7.2.** *Given black-box access to a degree $d$ polynomial $f \in \mathbb{F}[X]$ such that $f$ is computable by a set-multilinear $\Sigma\Pi\Sigma(k)$ circuit $C_f$ over the field $\mathbb{F}$ with characteristic $0$ or $> d$, there is a randomized $\mathrm{poly}(n, \mathrm{Sys}(d^2k^2, k^2d^2n + \binom{dk+d}{k}, d)) \leq (dkn)^{\mathcal{O}(d^2k^3)^{(d^2k^2)}}$ time algorithm that outputs a $\Sigma\Pi\Sigma_{\{\sqcup_j X_j\}}(k)$ circuit computing $f$, where $X = \uplus_i X_i$.*

*Proof.* Let $m$ be the number of essential variables in $f$. Since there at most $kd$ linear forms appearing in $C$, it is easy to see that $m \leq kd$. By Lemma 3.15, there is a polynomial-time randomized algorithm that given black-box access to $f$, computes an invertible linear transformation $A \in \mathbb{F}^{n\times n}$ such that $f(A \cdot \bar{x})$ only depends on the first $m$ variables.

Let $g(X) = f(A \cdot \bar{x})$. Observe that given black-box access to $f$, one can easily simulate black-box access to $g$, since in order to evaluate $g$ at any input $\alpha \in \mathbb{F}[x_1, x_2, \ldots, x_n]$, one has to simply evaluate $f$ at $A \cdot \alpha$.

Also observe that $g(A^{-1} \cdot \bar{x}) = f(\bar{x})$. Thus any algorithm that can efficiently learn $g$ can also efficiently learn $f$ in the following way. For each $i \in [n]$, suppose that $R_i$ denote the $i$th row of $A^{-1}$. Then in the $i$th input to $g$ simply input the linear polynomial $L_i = \langle R_i, \bar{x}\rangle$, which is the inner product of $R_i$ and the vector $\bar{x}$ of formal input variables. Since $g$ only depends on the first $m$ variables, we only really need to do this operation for $i \in [m]$.

Since $f$ is computed by a degree $d$ set-multilinear $\Sigma\Pi\Sigma(k)$ circuit, hence $g(\bar{x}) = f(A \cdot \bar{x})$ is also has a natural degree $d$ $\Sigma\Pi\Sigma(k)$ circuit representation, where the linear forms of that representation are obtained by applying the transformation $A$ to corresponding linear forms of $C$. Let us call this circuit $C_g$. Notice that $C_g$ may not be set-multilinear (or even multilinear). However, if we're somehow able to learn the precise circuit $C_g$, then by substituting each variable $x_i$ to $L_i$ then we would recover the circuit $C_f$ which is indeed set-multilinear.

Thus our goal is now the following. We have black-box access to $g$ which only depends on $m$ variables. We would like to devise an algorithm for reconstructing $C_g$. Note that, $C_g$ is a particular degree $d$ $\Sigma\Pi\Sigma(k)$ representation of $g$ with a nice property that when we plug in $x_i = L_i$ in this representation, then we recover a set-multilinear $\Sigma\Pi\Sigma(k)$ representation of $f$. Let us call the new object obtained by plugging in $x_i = L_i$ for each $i$, the "lift" of $C_g$.

However, $g$ might have multiple representations as a degree $d$ $\Sigma\Pi\Sigma(k)$ circuit. And, there is no guarantee that their lift will be set-multilinear. However the existence of $C_g$ tells us that there exists a $\Sigma\Pi\Sigma(k)$ representation of $g$ whose lift is a set-multilinear $\Sigma\Pi\Sigma(k)$ circuit. Can we find such a representation of $g$?

We will now see that we can actually do this. In order to learn a degree $d$ $\Sigma\Pi\Sigma(k)$ representation of $g$ we will set up a system of polynomial equations whose solution will give as a degree $d$ $\Sigma\Pi\Sigma(k)$ representation. We will be able to impose additional polynomial constraints to this system that will further ensure that whatever $\Sigma\Pi\Sigma(k)$ representation is learned will be such that its lift will be a set-multilinear $\Sigma\Pi\Sigma(k)$ circuit.

The algorithm first learns $g$ as a sum of monomials. Since $g$ is of degree at most $d$ and depends on at most $kd$ variable, such a representation of $g$ can be found in time $\mathrm{poly}\left(\binom{kd+d}{d}\right)$ using known sparse polynomial reconstruction algorithms [KS01, BOT88]. Let $S$ be the set of $m$-tuples of non-negative integers that sum to $d$. Then the algorithm finds a collection of coefficients $\{c_{\bar{e}} \in \mathbb{F} | \bar{e} \in S\}$ such that $g = \sum_{\bar{e}\in S} c_{\bar{e}} \cdot \bar{x}^{\bar{e}}$.

Any degree $d$ $\Sigma\Pi\Sigma(k)$ representation of $g$ looks like the following:

$$\sum_{i=1}^{k}\prod_{j=1}^{d}(a_{j,1}^{(i)}x_1 + a_{j,2}^{(i)}x_2 + \ldots + a_{j,m}^{(i)}x_m + a_{j,m+1}^{(i)}) = \sum_{\bar{e}\in S} c_{\bar{e}} \cdot \bar{x}^{\bar{e}}.$$

The algorithm already knows the set of coefficients $\{c_{\bar{e}} \in \mathbb{F} | \bar{e} \in S\}$. In order to learn a $\Sigma\Pi\Sigma(k)$ representation it needs to learn values for the coefficients in the LHS, i.e. the $a_{j,r}^{(i)}$ for various choices of $i, j, r$. These $a_{j,r}^{(i)}$ are the unknown variables.

Now for each monomial $x^{\bar{e}}$ that appears in $g$, we can compare the coefficient of it on the LHS and RHS of the above expression, set them equal to each other, and get a polynomial equation in the unknown variables. We do this for all the monomials and hence set up a system of polynomial equations in the unknown variables. Each solution to this system of equations corresponds to a degree $d$ $\Sigma\Pi\Sigma(k)$ representation of $g$ and vice versa.

We are looking for a degree $d$ $\Sigma\Pi\Sigma(k)$ representation whose lift is set-multilinear. To ensure this, we will add some additional polynomial constraints to our system of polynomial equations.

Now suppose that

$$\sum_{i=1}^{k}\prod_{j=1}^{d}(a_{j,1}^{(i)}x_1 + a_{j,2}^{(i)}x_2 + \ldots + a_{j,m}^{(i)}x_m + a_{j,m+1}^{(i)})$$

represents some degree $d$ $\Sigma\Pi\Sigma(k)$ representation of $g$. (We still treat the $a_{j,r}^{(i)}$ as unknown variables). Also, denote the linear polynomials

$$\ell_j^{(i)} := (a_{j,1}^{(i)}L_1 + a_{j,2}^{(i)}L_2 + \ldots + a_{j,m}^{(i)}L_m + a_{j,m+1}^{(i)})$$

appearing in the expression. Each $L_i$ is a linear form in $x_1, \ldots, x_n$, and the algorithm knows what these $L_i$ are. Further, let $X = \uplus_{w\in[d]}X_w$ be the variable partition of C (and $f$).

In order for its lift to be set-multilinear with respect to the variable partition $\uplus_{w\in[d]}X_w$, we would need to look at the expression

$$\sum_{i=1}^{k}\prod_{j=1}^{d}(a_{j,1}^{(i)}L_1 + a_{j,2}^{(i)}L_2 + \ldots + a_{j,m}^{(i)}L_m + a_{j,m+1}^{(i)})$$

and use Observation 7.1.

We add the polynomial equations given by Observation 7.1 to our system of polynomial equations.

Observe that any solution to the new system will have the property that the lift will be set-multilinear. Moreover the existence of $C_g$ guarantees that the system will have at least one solution, and hence it is solvable in time $\mathrm{Sys}(mdk, k^2d^2n + \binom{dk+d}{k}, d)$. And the overall time complexity of the algorithm is bounded by $\mathrm{poly}(n, \mathrm{Sys}(d^2k^2, k^2d^2n + \binom{dk+d}{k}, d)) \leq (dkn)^{\mathcal{O}(d^2k^3)^{(d^2k^2)}}$.

□

## 7.2 Reconstructing High-Degree Set-Multilinear $\Sigma\Pi\Sigma(k)$ Circuits

In this section, we show how to apply the generalized alternant - a tool we have developed in Section 5, to the scenario of set-multilinear polynomials. In what follows, we fix a partition $X = \uplus_{j\in[d]}X_j$ of $X$ and consider a polynomial $F \in \mathbb{F}[X]$ that can be written as

$$F(X) = \sum_{i=1}^{k}F_i(X) = \sum_{i=1}^{k}\prod_{j=1}^{d}P_{ij}(X_j).$$

We remark that we fix the partition for the purpose of analysis only. Indeed, knowing the partition is **not** required to execute the algorithm.

Our main result in this section is a structural result that shows that a set-multilinear $\Sigma\Pi\Sigma(k)$ circuit with sufficiently many parts can be expressed in a form that satisfies the premises of Theorem 5.3. We first prove a lemma that will be helpful.

**Lemma 7.3.** *Suppose that* $\mathrm{rank}_{\mathbb{F}}\{F_1, \ldots, F_k\} = r$. *Then there exists* $J \subseteq [d]$ *of size* $|J| \leq r-1$ *such that*

$$\mathrm{rank}_{\mathbb{F}}\left\{\prod_{j\in J}P_{ij}(X_j) \mid i \in [k]\right\} = r.$$

*Proof.* Assume w.l.o.g that $\text{rank}_{\mathbb{F}}\{F_1, \ldots, F_r\} = r$. By Lemma 4.3, there exists a subset $S$ of variables of size of $|S| \leq r-1$ such that the polynomials $F_1, \ldots, F_r$ are $\mathbb{F}(X \setminus S)$-linearly independent. Let $J$ be the set of parts that contain the variables of $S$. Formally:

$$J \triangleq \{j \mid X_j \cap S \neq \emptyset\} \text{ and let } X_J \triangleq \cup_{j \in J} X_j.$$

Observe that $S \subseteq X_J$ and hence $X \setminus X_J \subseteq X \setminus S$. Furthermore, as $X_j$-s are disjoint, we have that $|J| \leq |S| \leq r-1$. For $i \in [k]$ we define:

$$Q_i(X_J) \triangleq \prod_{j \in J} P_{ij}(X_j) \text{ and } W_i(X \setminus X_J) \triangleq \frac{F_i}{Q_i}.$$

We now claim that $Q_1, \ldots, Q_r$ are $\mathbb{F}$-linearly independent. Consider an $\mathbb{F}$-linear combination $\sum_{i=1}^{r} \beta_i \cdot Q_i \equiv 0$. Let us multiply the equation by the product $\prod_{i=t}^{r} W_t$. We have that:

$$0 \equiv \sum_{i=1}^{r} \beta_i \cdot Q_i \cdot \prod_{t=1}^{r} W_t = \sum_{i=1}^{r} \left( \beta_i \cdot \prod_{t \neq i}^{r} W_t \right) \cdot F_i$$

Hence, the RHS of the equation constitutes an $\mathbb{F}(X \setminus S)$-linear combination of $F_1, \ldots, F_r$ and consequently, $\forall i \in [r] : \beta_i \cdot \prod_{t \neq i}^{r} W_t \equiv 0$. As $\forall t : W_t \not\equiv 0$, we obtain that $\forall i \in [r] : \beta_i = 0$. □

**Lemma 7.4.** *Let $d \geq k$ and suppose that $\text{rank}_{\mathbb{F}}\{F_1, \ldots, F_k\} = k$. Then there exists $r \in [k]$ and a partition $J \cup J' \cup J'' = [d]$ such that $F(X) = \sum_{i=1}^{k} F_i(X)$ can also be written as*

$$F(X) = \sum_{i=1}^{r} F'_i(X) = \sum_{i=1}^{r} R_i(X_J) \cdot Q_i(X_{J'}) \cdot H_i(X_{J''})$$

*where*

1. *$|J| \leq k-1$, $|J'| \leq r-1$ and hence $|J''| \geq d-r-k+2$.*
2. *There exists a 1-1 mapping $\sigma : [r] \to [k]$ such that $\forall i \in [r] : H_i = \prod_{j \in J''} P_{\sigma(i),j}(X_j)$.*
3. *$\text{rank}_{\mathbb{F}}\{R_i \mid i \in [r]\} = \text{rank}_{\mathbb{F}}\{Q_i \mid i \in [r]\} = r$.*

*Proof.* By Lemma 7.3, there exists $J \subseteq [d]$ of size $|J| \leq k-1$ such that

$$\text{rank}_{\mathbb{F}} \left\{ \prod_{j \in J} P_{ij}(X_j) \mid i \in [k] \right\} = k.$$

Consider the "remaining" parts of $F : \bar{J} \triangleq [d] \setminus J$ and let $r$ denote the corresponding rank. That is:

$$r \triangleq \text{rank}_{\mathbb{F}} \left\{ \prod_{j \in \bar{J}} P_{ij}(X_j) \mid i \in [k] \right\}.$$

Observe that $\left|\bar{J}\right| \geq 1$. Hence, by definition, $1 \leq r \leq k$. By Lemma 7.3 (again), there exists $J' \subseteq \bar{J}$ of size $|J'| \leq r-1$ such that:

$$\text{rank}_{\mathbb{F}} \left\{ \prod_{j \in J'} P_{ij}(X_j) \mid i \in [k] \right\} = r.$$

Finally, let $J'' \triangleq [d] \setminus (J \cup J')$. Observe that $J, J'$ and $J''$ form a *partition* of $[d]$. For $i \in [k]$ we define:

$$R_i \triangleq \prod_{j \in J} P_{ij}(X_j) \ , \ Q_i \triangleq \prod_{j \in J'} P_{ij}(X_j) \ , \ H_i \triangleq \prod_{j \in J''} P_{ij}(X_j).$$

That is, $F = \sum_{i=1}^{k} F_i = \sum_{i=1}^{k} R_i \cdot Q_i \cdot H_i$ where

$$\text{rank}_{\mathbb{F}}\{R_i \mid i \in [k]\} = k \text{ and } \text{rank}_{\mathbb{F}}\{Q_i \mid i \in [k]\} = \text{rank}_{\mathbb{F}}\{Q_i \cdot H_i \mid i \in [k]\} = r.$$

If $r = k$ then we are done. Otherwise, $r < k$. Assume w.l.o.g that $\text{rank}_{\mathbb{F}}\{Q_i \cdot H_i \mid i \in [r]\} = r$. Otherwise, we can relabel the terms. By definition, for any $\ell \geq r+1$ there exist $\alpha_{\ell,i} \in \mathbb{F}$ such that: $Q_\ell \cdot H_\ell = \sum_{i=1}^{r} \alpha_{\ell,i} \cdot Q_i \cdot H_i$. Hence, we can write:

$$F = \sum_{i=1}^{k} F_i = \sum_{i=1}^{k} R_i \cdot Q_i \cdot H_i = \sum_{i=1}^{r} \left[ R_i + \sum_{\ell=r+1}^{k} \alpha_{\ell,i} \cdot R_\ell \right] \cdot Q_i \cdot H_i.$$

It remains to show that

$$\text{rank}_{\mathbb{F}} \left\{ R_i + \sum_{\ell=r+1}^{k} \alpha_{\ell,i} \cdot R_\ell \mid i \in [r] \right\} = r.$$

Consider an $\mathbb{F}$-linear combination

$$\sum_{i=1}^{r} \beta_i \cdot \left[ R_i + \sum_{\ell=r+1}^{k} \alpha_{\ell,i} \cdot R_\ell \right] \equiv 0.$$

We can rewrite this as:

$$\sum_{i=1}^{r} \beta_i \cdot R_i + \sum_{\ell=r+1}^{k} \left( \sum_{i=1}^{r} \beta_i \cdot \alpha_{\ell,i} \right) \cdot R_\ell \equiv 0.$$

Hence, we have a linear combination of $R_1, \ldots R_k$. As $\text{rank}_{\mathbb{F}}\{R_i \mid i \in [k]\} = k$ we obtain that $\forall i \in [r] : \beta_i = 0$, as required. □

### 7.2.1 Learning two Linear Forms appearing in the Circuit

As a first step towards learning a general (high degree) $\Sigma\Pi\Sigma_{\{\sqcup_j X_j\}}(k)$ circuit $C$, our algorithm will try to learn a single linear form appearing in the circuit $C$. In fact, it will be convenient to learn 2 linear forms appearing in $C$ such that each multiplication gate of $C$ contains at most one of them.

**Lemma 7.5.** *Let $f$ be a nonzero polynomial computable by a simple and minimal set-multilinear $\Sigma\Pi\Sigma(k)$ circuit such that $k \geq 2$ and the degree $d > 2k$. Further, assume that there are no linear factors of $f$. Then, given black-box access to $f$, there is a randomized algorithm that runs in time $n^{\mathcal{O}(k)}\text{poly}(d)$, and does the following. It outputs a set $L$ of $\text{poly}(d,k)$ pairs of linear forms which has the following property. Let $C \equiv \sum_{i=1}^{k} T_i$ be any simple and minimal set-multilinear representation of $f$. One of the pairs $(\ell_1, \ell_2)$ in $L$ is such that for some $i \in [d]$, $\ell_1$ and $\ell_2$ are supported on the variables of $X_i$, and both $\ell_1$ and $\ell_2$ appear in $C$.*

*Proof.* We know that $f$ is computable by a set-multilinear $\Sigma\Pi\Sigma$ circuit. Pick any *optimal* representation (w.r.t top fan-in) $f = T_1 + T_2 + \ldots T_k$, where $T_i = \prod_{j \in [d]} \ell_{i,j}(X_j)$ for some variable partition $X = \cup X_j$. By optimality of top fan-in, we have that $\text{rank}_{\mathbb{F}}\{T_1, T_2, \ldots, T_k\} = k$. Using lemma 7.4 we can rewrite $f = \sum_{i \in [k]} T_i = \sum_{i \in [r]} T'_i = \sum_{i=1}^{r} R_i(X_J) \cdot Q_i(X_{J'}) \cdot H_i(X_{J''})$ where $J \cup J' \cup J'' = [d]$ and $|J| \leq k-1$, $|J'| \leq r-1$. Recall for any set $J$, $X_J = \cup_{j \in J} X_j$. Further, there exists a 1-1 mapping $\sigma : [r] \to [k]$ such

that $\forall i \in [r] : H_i = \prod_{j \in J''} \ell_{\sigma(i),j}(X_j)$ and $\text{rank}_{\mathbb{F}}\{R_i \mid i \in [r]\} = \text{rank}_{\mathbb{F}}\{Q_i \mid i \in [r]\} = r$. Now, $f$ satisfies the premises of Theorem 5.3 and in $n^{\mathcal{O}(k)}\text{poly}(d)$ we will get a list $L'$ which contains all the linear factors of $H_1, H_2, \ldots, H_k$. As $d > 2k$, the number of $H_i$'s is non-trivial. Moreover, for any linear factor $\ell(X_j)$ appearing in any one of the $H_i$'s, there must exist another linear factor $\ell'(X_j)$ appearing in one of the other $H_{i'}$'s (and supported on the same variable set), otherwise $f$ will have a linear factor, which we assumed is not the case. The output set is simply the collection of all pairs of linear forms in $L'$ that are supported on the same variable partition. Note that, $|L'| \leq \text{poly}(k, d)$ and the set $L$ has the required property. □

### 7.2.2 Learning most of the Linear Forms appearing in the Circuit

In this section, we will see how to use the two linear forms learnt in the previous subsection to learn a small set $S$ of multiplication gates, such that each multiplication gate $T_i$ of $C$ is very "close" to some gate of $S$. From this, we will then see how to essentially learn *most* of the linear forms appearing in each gate of $C$. Our approach is recursive, so we will assume using an induction hypothesis that there is $\mathcal{A}(n, k-1, d)$ time randomized algorithm for reconstructing degree $d$, set-multilinear $\Sigma\Pi\Sigma(k-1)$ circuits. Note that, the base case of $k = 1$ follows directly from black-box factoring result of [KT90] i.e. $\mathcal{A}(n, 1, d) = \text{poly}(n, d)$.

**Lemma 7.6.** *Let $C \equiv \sum_{i=1}^{k} T_i$ be a minimal $\Sigma\Pi\Sigma_{\{\sqcup_j X_j\}}(k)$ circuit of degree $d$ and let $\ell_1$ and $\ell_2$ be two distinct linear forms supported on variables of $X_i$ for some $i \in [d]$ such that each of $\ell_1$ and $\ell_2$ appears in $C$. Then there is a randomized algorithm that given black-box access to $C$, given $\ell_1$ and $\ell_2$, and two oracle calls to algorithm for learning set-multilinear $\Sigma\Pi\Sigma(k-1)$ circuits, runs in time at most $2\mathcal{A}(n, k-1, d) + \text{poly}(n, k, d)$ and outputs a set $S = \{M_1, M_2 \ldots, M_{|S|}\}$ of at most $2k-2$ $\Pi\Sigma$ circuits of degree $d-1$ such that with high probability, for all $i \in [k]$, there exists $j \in [2k-2]$ such that $\Delta(T_i, M_j) = O(k^4)$.*

The proof of the above lemma is almost identical to the proof of Lemma 5.7 in [BSV21] and thus omitted. The only place where it differs is the use of multilinear rank bounds (Theorem 3.7) instead of set-multilinear rank bounds, as the variable partition can be different across two representations.

Thus we can now assume that our algorithm can compute a set $S$ consisting of multiplication gates such that $|S| \leq 2(k-1)$, and each gate of the circuit $C$ that we are trying to learn is close to some element of $S$. The next lemma shows how we can polish our set S to ensure that the multiplication gates we learned *divide* the original gates and are supported on the same variable sets.

**Lemma 7.7.** *Let $C \equiv \sum_{i=1}^{k} T_i$ be a $\Sigma\Pi\Sigma_{\{\sqcup_j X_j\}}(k)$ circuit of degree $d$ and let $S = \{M_1, M_2 \ldots, M_{|S|}\}$ be a set of at most $2k-2$ $\Pi\Sigma$ circuits of degree $d-1$ such that for all $i \in [k]$, there exists $j \in [2k-2]$ such that $\Delta(T_i, M_j) = O(k^4)$. Then there is a $\text{poly}((kd^{k^3}))$-time algorithm for computing another set $\bar{S}$ which has the following properties.*

1. $|\bar{S}| \leq \left(|S| \cdot \binom{d-1}{\mathcal{O}(k^5)}\right)^k$
2. *The elements of $\bar{S}$ are $k$-tuples of $\Pi\Sigma$ circuits of degree $d - \mathcal{O}(k^5)$*
3. *One of the elements of $\bar{S}$ is of the form $(G_1, G_2, \ldots G_k)$ where for each $i \in [k]$, $G_i$ divides $T_i$. Moreover, all of the $G_i$s are set-multilinear $\Pi\Sigma$ circuits sharing the same variable partition.*

Again, the proof of the above lemma is identical (with minor changes in parameters) to that of Lemma 5.8 in [BSV21] and thus omitted.

## 7.3 Learning the Full Circuit

This part of our learning algorithm differs substantially from that of [BSV21]. This is because we don't have the "width reduction" procedure (Section 5.1 [BSV21]), Lemma 5.9 from [BSV21] will no longer be efficient (as width can be $\Omega(n)$). Fortunately, there is a workaround using our generalization of Carlini's variable reduction lemma (Lemma 3.16).

**Lemma 7.8.** *Let $C \equiv \sum_{i=1}^{k} T_i$ be a $\Sigma\Pi\Sigma_{\{\sqcup_j X_j\}}(k)$ circuit of degree $d$ computing a polynomial $f(X) \in \mathbb{F}[X]$, where char$(\mathbb{F}) > d$ or char$(\mathbb{F}) = 0$. Let $(G_1, G_2, \dots G_k)$ be a $k$-tuple where for each $i \in [k]$, $G_i$ divides $T_i$. Moreover, all of the $G_i$s are set-multilinear $\Pi\Sigma$ circuits of degree $d - \mathcal{O}(k^5)$, sharing the same variable partition. Then, given black-box access to $C$ and given the $k$-tuple $(G_1, G_2, \dots G_k)$, there is a randomized* $\text{poly}\left(n, k^{k^{k^{\mathcal{O}(1)}}}\right)$ *time algorithm that outputs a set-multilinear $\Sigma\Pi\Sigma(k)$ representation of $f$.*

*Proof.* We are given $(G_1, G_2, \dots G_k)$ and our aim is to find $H_i$-s such that $f = \sum_{i\in[k]} G_i H_i$ is a set-multilinear $\Sigma\Pi\Sigma(k)$ representation of $f$. Let $Y = \text{var}(G_1, G_2, \dots G_k)$ and $Z = X \setminus Y$. Note that, $H_i \in \mathbb{F}[Z]$. Notice that, if we have black-box access to the individual $H_i$'s, then we can learn them just by black-box factorization followed by sparse reconstruction of the linear factors. We will achieve something close to this in principle.

As a first step, we find the linear dependency structure among the $G_i$-s using Lemma 4.9. Let $G_1, G_2, \dots G_c$ be a basis of linear space of $G_i$-s (we can always ensure this by relabelling of gates). Also, let $M$ be a $k \times c$ matrix which is the corresponding linear dependence matrix we get from Lemma 4.9, that is,

$$M_{k\times c} \begin{pmatrix} G_1 \\ \vdots \\ G_c \end{pmatrix} = \begin{pmatrix} G_1 \\ \vdots \\ G_k \end{pmatrix}. \tag{2}$$

Note that,

$$f = \begin{pmatrix} H_1, \cdots, H_k \end{pmatrix} \cdot \begin{pmatrix} G_1 \\ \vdots \\ G_k \end{pmatrix} = \begin{pmatrix} H_1, \cdots, H_k \end{pmatrix} \cdot M \cdot \begin{pmatrix} G_1 \\ \vdots \\ G_c \end{pmatrix}$$

For $i \in [c]$, define $\tilde{H}_i$ by the following equality

$$\begin{pmatrix} \tilde{H}_1 \\ \vdots \\ \tilde{H}_c \end{pmatrix} := M^T \begin{pmatrix} H_1 \\ \vdots \\ . \\ H_k \end{pmatrix}. \tag{3}$$

Thus,

$$f(Y, Z) = \sum_{i\in[k]} H_i(Z) G_i(Y) = \sum_{i\in[c]} \tilde{H}_i(Z) G_i(Y). \tag{4}$$

By the proof of Lemma 3.6 in [BSV21] we can get black-box access to $\tilde{H}$.

Note that, we can't directly set up a system for $H_i$'s because they can potentially depend on $\Omega(n)$ variables, and hence our system will have a large number of unknowns. Instead, we will apply an invertible linear transformation on Z variables and reduce our variables to poly$(k)$.

Observe that,

$$\left\langle \frac{\partial f}{\partial Z} \right\rangle \subseteq G_1(Y) \cdot \left\langle \frac{\partial H_1}{\partial Z} \right\rangle + \cdots + G_k(Y) \cdot \left\langle \frac{\partial H_k}{\partial Z} \right\rangle$$

Thus,

$$\dim \left\langle \frac{\partial f}{\partial Z} \right\rangle \leq k \cdot ev(H_i) = \mathcal{O}(k^6).$$

By Lemma 3.16 we can get an invertible linear transformation $A$ s.t. $f(AZ, Y) \in \mathbb{F}[z_1, \dots, z_m, Y]$ and $m = \mathcal{O}(k^6)$.

Let $H_i(AZ) = \prod_{j\in[k^5]}(a_{i,j,1}z_1 + a_{i,j,2}z_2 \dots a_{i,j,r}z_m)$, where $a_{i,j,k}$ are unknowns that we intend to find.

And we have, the following equality along with BB access to $\tilde{H}_i(AZ)$'s.

$$\begin{pmatrix} \tilde{H}_1(AZ) \\ \vdots \\ \tilde{H}_c(AZ) \end{pmatrix} = M^T \begin{pmatrix} H_1(AZ) \\ \vdots \\ H_k(AZ) \end{pmatrix}.$$

Now, we will set up a system of polynomial equations with $\mathcal{O}(k^6)$ variables s.t. the lift to $H_i(A^{-1} \cdot AZ)$ is set multilinear w.r.t to the same partition across all $H_i$'s (Observation 7.1). Note that, a solution exists and every solution will give us a valid representation.

For time complexity analysis, we have linear-algebraic steps which can be done in poly$(n)$-time. And for system-solving, we have $\mathcal{O}(k^6)$ unknowns, with $k^{\mathcal{O}(k)}$ equations of degree at most $\mathcal{O}(k^5)$. This gives that the total time complexity is bounded by poly$\left(n, k^{k^{k^{\mathcal{O}(1)}}}\right)$. □

## 7.4 Putting it All Together

We now show how to combine all the lemmas and subroutines developed so far to get the full reconstruction algorithm for set-multilinear $\Sigma\Pi\Sigma(k)$ circuits with an unknown partition. The theorem below is basically a restatement of Theorem 3 (only for the randomized algorithm over general fields).

**Theorem 7.9.** *Given black-box access to a degree $d$, $n$-variate polynomial $f \in \mathbb{F}[X]$ such that $f$ is computable by a set-multilinear $\Sigma\Pi\Sigma(k)$ circuit $C_f$ over the field $\mathbb{F}$ with char$(\mathbb{F}) > d$ or char$(\mathbb{F}) = 0$, there is a randomized* poly$(d^{k^3}, k^{k^{k^{10}}}, n)$ *time algorithm that outputs a $\Sigma\Pi\Sigma_{\{\sqcup_j X_j\}}(k)$ circuit computing $f$.*

*Proof.* Our algorithm is recursive and we assume that we have an efficient algorithm for reconstructing a set-multilinear $\Sigma\Pi\Sigma(k-1)$ circuit that runs in time $\mathcal{A}(n, k-1, d)$. For the base case of $k = 1$, the reconstruction algorithm follows directly from black-box factoring result of [KT90].

Now assume $k \geq 2$. By stripping off all the linear factors Corollary 3.13, and closure of factoring Lemma 3.12, we can assume that $C_f$ is simple and has $f$ has no linear factors. We can also assume that $C_f$ is minimal. This is because if $f$ has a $\Sigma\Pi\Sigma_{\{\sqcup_j X_j\}}(k)$ representation with a non-minimal circuit, then it also has $\Sigma\Pi\Sigma_{\{\sqcup_j X_j\}}(k)$ representation with a minimal circuit, which is obtained by just deleting any subset of multiplication gates in the non-minimal circuit which sums to zero.

If $d = \mathcal{O}(k^5)$, then we invoke the algorithm in Lemma 7.2 to learn a $\Sigma\Pi\Sigma_{\{\sqcup_j X_j\}}(k)$ representation of $C$.

Otherwise, we invoke the algorithm from Lemma 7.5 to compute the set $L$ of pairs of linear forms. For each pair $(\ell_1, \ell_2) \in L$ we do the following: We invoke Lemma 7.6 to compute a set $S$ of at most $2k-2$ $\Pi\Sigma$ circuits and then invoke Lemma 7.7 to compute a set $\bar{S}$ of $k$-tuples. Let us call this final set $\bar{S}_{(\ell_1,\ell_2)}$. For each $k$-tuple $(G_1, G_2, \ldots, G_k) \in \bar{S}_{(\ell_1,\ell_2)}$ we invoke the algorithm of Lemma 7.8 to output a circuit. We then verify that the output circuit indeed has the $\Sigma\Pi\Sigma_{\{\sqcup_j X_j\}}(k)$ format and then we check (by running a polynomial identity testing algorithm) if it computes $f$. If it passes both these verification steps then the algorithm halts and outputs that circuit. By Lemmas 7.5, 7.6, 7.7 and 7.8, we do know that for some choice of $(\ell_1, \ell_2)$ and for some choice of $(G_1, G_2, \ldots, G_k) \in \bar{S}_{(\ell_1,\ell_2)}$, the algorithm will succeed with high probability.

**Time complexity analysis**: Recall, $\mathcal{A}(n, k, d)$ is the time complexity of learning degree $d$, set-multilinear $\Sigma\Pi\Sigma(k)$ circuit computing $f$. We now upper bound $\mathcal{A}(n, k, d)$. Note that,

$$\begin{aligned} \mathcal{A}(n,k,d) &\leq 2 \cdot \mathcal{A}(n, k-1, d) + \text{poly}\left(n, k^{k^{k^{\mathcal{O}(1)}}}\right) + d^{k^{\mathcal{O}(1)}}. \\ &\leq 2^k \cdot \mathcal{A}(n,1,d) + k \cdot \text{poly}\left(n, k^{k^{k^{\mathcal{O}(1)}}}, d^{k^{\mathcal{O}(1)}}\right) \leq \text{poly}(d^{k^{\mathcal{O}(1)}}, k^{k^{k^{\mathcal{O}(1)}}}, n). \end{aligned}$$

□

### 7.5 NP-Hardness

We complement our reconstruction algorithm by showing that given a polynomial computed by a set-multilinear $\Sigma\Pi\Sigma$ circuit of *unknown* partition (and of arbitrary top fan-in), the task of merely finding a partition is already NP-hard! Indeed, one can embed the problem of graph 3-coloring into a problem of recovering the partition. The intuition is that one can think of each part as a color.

**Lemma 7.10.** *Let $G = (V, E)$ be a graph. Assume wlog that $V = [n]$. Define $f_G \triangleq \sum_{(u,v)\in E} x_u x_v y_{u,v}$. Then $f_G$ is computable by set-multilinear $\Sigma\Pi\Sigma$ circuit iff $G$ is 3-colorable.*

*Proof.* Suppose that $G$ is 3-colorable and let $c : V \to [3]$ be a coloring function. For $j \in [3]$, define

$$X_j \triangleq \{x_u \mid c(x_u) = j\} \cup \{y_{u,v} \mid c(x_u), c(x_v) \neq j\}$$

Pick an edge $(u, v) \in E$ and consider the corresponding summand $x_u x_v y_{u,v}$. Since $G$ is 3-colorable, $c(x_u) \neq c(x_v)$ hence $x_u$ and $x_v$ will be assigned to different parts. In addition, the variable $y_{u,v}$ to the remaining part $X_{[3]\setminus\{c(x_u),c(x_v)\}}$. Note that the variable $y_{u,v}$ appears only once. Consequently, $f_G$ is computable by a set-multilinear $\Sigma\Pi\Sigma$ circuit.

For the other direction, recall that if a polynomial is computed by a set-multilinear circuit, each monomial should pick up exactly one variable from each part in the partition. That is, there exists a partition of all the variables into $X_1, X_2, X_3$ such that for every $(u, v) \in E$, each variable in $x_u x_v y_{u,v}$ should belong to different parts. We can now define a coloring function $c : V \to [3]$ as: $c(x_u) = j$ iff $x_u \in X_j$. This implies that $u$ and $v$ should have different colors and hence $c$ constitutes a 3-coloring of $G$. □

## 8 Discussion & Open Questions

In this paper we introduced a new technique and showed applications for identity testing and circuit reconstruction.

1. One can extend our technique to "lift" PIT algorithms for constant-wise sums and products of circuits of a class $\mathcal{C}$ to occur-$k$ formulas of higher, **constant** depth (in this paper we just mentioned it for depth 3 circuits, but this lifting process can be iterated) with the addition restriction that all the intermediate addition gates must have bounded fan-in. It is an interesting question to try to remove this restriction on fan-in.

2. In addition to characteristic requirements, a difference between the result of [ASSS16] and our result is the ability of [ASSS16] to handle powers. In particular, consider the following "symmetric" version of the sparsity testing problem of [GK85]: given two sparse polynomials $f, g$ and an integer $e$, decide if $f = g^e$. In the setting of [ASSS16], $C \triangleq f - g^e$ is considered an occur-2 formula and hence can be tested efficiently, albeit for a sufficiently large characteristic[11]. In our setting, $C$ is an occur-$(e + 1)$-formula hence rendering our PIT algorithms inefficient when $e$ is super-constant. Can one leverage our techniques to solve this problem for all fields?

## Acknowledgments

The authors would like to thank the anonymous referees for useful comments that improved the presentation of the results.

[11] A similar result was obtained in [Vol17].

# References


[Ait17] Alexander Craig Aitken. *Determinants and matrices*. Read Books Ltd, 2017.

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

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

[Ang88] D. Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1988.

[ASSS16] M. Agrawal, C. Saha, R. Saptharishi, and N. Saxena. Jacobian hits circuits: Hitting sets, lower bounds for depth-d occur-k formulas and depth-3 transcendence degree-k circuits. *SIAM J. Comput.*, 45(4):1533–1562, 2016.

[AV08] M. Agrawal and V. Vinay. Arithmetic circuits: A chasm at depth four. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 67–75, 2008.

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

[BBB+00] A. Beimel, F. Bergadano, N. H. Bshouty, E. Kushilevitz, and S. Varricchio. Learning functions represented as multiplicity automata. *J. ACM*, 47(3):506–530, 2000.

[BD10] Alin Bostan and Philippe Dumas. Wronskians and linear independence. *The American Mathematical Monthly*, 117(8):722, 2010.

[BMS13] M. Beecken, J. Mittmann, and N. Saxena. Algebraic independence and blackbox identity testing. *Information & Computation*, 222:2–19, 2013.

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

[BSV20] V. Bhargava, S. Saraf, and I. Volkovich. Reconstruction of depth-4 multilinear circuits. In Shuchi Chawla, editor, *Proceedings of the 31st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2144–2160. SIAM, 2020.

[BSV21] V. Bhargava, S. Saraf, and I. Volkovich. Reconstruction algorithms for low-rank tensors and depth-3 multilinear circuits. In *53rd Annual ACM SIGACT Symposium on Theory of Computing STOC*, pages 809–822. ACM, 2021.

[Car06] E. Carlini. Reducing the number of variables of a polynomial. In Mohamed Elkadi, Bernard Mourrain, and Ragni Piene, editors, *Algebraic Geometry and Geometric Modeling*, pages 237–247. Springer, 2006.

[CS19] P. Chatterjee and R. Saptharishi. Constructing faithful homomorphisms over fields of finite characteristic. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS*, volume 150 of *LIPIcs*, pages 11:1–11:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[DDS21] P. Dutta, P. Dwivedi, and N. Saxena. Deterministic identity testing paradigms for bounded top-fanin depth-4 circuits. In *36th Conference on Computational Complexity (CCC 2021)*, volume 5, page 9, 2021.

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

[DS07] 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, 2007.

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

[GJR10] E. Grigorescu, K. Jung, and R. Rubinfeld. A local decision test for sparse polynomials. *Inf. Process. Lett.*, 110(20):898–901, 2010.

[GK85] J. von zur Gathen and E. Kaltofen. Factoring sparse multivariate polynomials. *Journal of Computer and System Sciences*, 31(2):265–287, 1985.

[GK16] Venkatesan Guruswami and Swastik Kopparty. Explicit subspace designs. *Combinatorica*, 36(2):161–185, 2016.

[GKKS13] A. Gupta, P. Kamath, N. Kayal, and R. Saptharishi. Arithmetic circuits: A chasm at depth three. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 578–587, 2013.

[GKL12] A. Gupta, N. Kayal, and S. V. Lokam. Reconstruction of depth-4 multilinear circuits with top fanin 2. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC)*, pages 625–642, 2012. Full version at https://eccc.weizmann.ac.il/report/2011/153.

[GKPS11] B. Grenet, P. Koiran, N. Portier, and Y. Strozecki. The limited power of powering: Polynomial identity testing and a depth-four lower bound for the permanent. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS*, volume 13 of *LIPIcs*, pages 127–139. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2011.

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

[Kay11] N. Kayal. Efficient algorithms for some special cases of the polynomial equivalence problem. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete algorithms*, pages 1409–1421. SIAM, 2011.

[KI04] V. Kabanets and R. Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004.

[Koi10] P. Koiran. Arithmetic circuits: the chasm at depth four gets wider. *CoRR*, abs/1006.4700, 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 (STOC)*, pages 216–223, 2001.

[KS06] A. Klivans and A. Shpilka. Learning restricted models of arithmetic circuits. *Theory of computing*, 2(10):185–206, 2006.

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

[KS09a] Z. S. Karnin and A. Shpilka. Reconstruction of generalized depth-3 arithmetic circuits with bounded top fan-in. In *Proceedings of the 24th Annual IEEE Conference on Computational Complexity (CCC)*, pages 274–285, 2009. Full version at http://www.cs.technion.ac.il/ shpilka/publications/KarninShpilka09.pdf.

[KS09b] N. Kayal and S. Saraf. Blackbox polynomial identity testing for depth 3 circuits. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 198–207, 2009. Full version at https://eccc.weizmann.ac.il/report/2009/032.

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

[KS17] M. Kumar and S. Saraf. Arithmetic circuits with locally low algebraic rank. *Theory Comput.*, 13(1):1–33, 2017.

[KT90] E. Kaltofen and B. M. Trager. Computing with polynomials given by black boxes for their evaluations: Greatest common divisors, factorization, separation of numerators and denominators. *J. of Symbolic Computation*, 9(3):301–320, 1990.

[LST22] N. Limaye, S. Srinivasan, and S. Tavenas. Superpolynomial lower bounds against low-depth algebraic circuits. In *FOCS 2021*, 2022.

[PS21] S. Peleg and A. Shpilka. Polynomial time deterministic identity testing algorithm for $\Sigma^{[3]}\Pi\Sigma\Pi^{[2]}$ circuits via edelstein–kelly type theorem for quadratic polynomials. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 259–271, 2021.

[PSS18] A. Pandey, N. Saxena, and A. Sinhababu. Algebraic independence over positive characteristic: New criterion and applications to locally low-algebraic-rank circuits. *Comput. Complex.*, 27(4):617–670, 2018.

[PSV22] Sh. Peleg, A. Shpilka, and B. Volk. Tensor reconstruction beyond constant rank. *Electron. Colloquium Comput. Complex.*, TR22-125, 2022.

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

[Sin16] G. Sinha. Reconstruction of real depth-3 circuits with top fan-in 2. In *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, pages 31:1–31:53, 2016.

[Sin20] G. Sinha. Efficient reconstruction of depth three circuits with top fan-in two. *Electron. Colloquium Comput. Complex.*, 27:125, 2020.

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

[SS12] N. Saxena and C. Seshadhri. Blackbox identity testing for bounded top-fanin depth-3 circuits: The field doesn't matter. *SIAM J. Comput.*, 41(5):1285–1298, 2012.

[SS13] N. Saxena and C. Seshadhri. From sylvester-gallai configurations to rank bounds: Improved blackbox identity test for depth-3 circuits. *J. ACM*, 60(5):33, 2013.

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

[SV18] S. Saraf and I. Volkovich. Blackbox identity testing for depth-4 multilinear circuits. *Combinatorica*, 38(5):1205–1238, 2018.

[SY10] A. Shpilka and A. Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5(3-4):207–388, 2010.

[Tav13] S. Tavenas. Improved bounds for reduction to depth 4 and depth 3. In *MFCS*, pages 813–824, 2013.

[Vol15] I. Volkovich. Deterministically factoring sparse polynomials into multilinear factors and sums of univariate polynomials. In *APPROX-RANDOM*, pages 943–958, 2015.

[Vol17] I. Volkovich. On some computations on sparse polynomials. In *APPROX-RANDOM*, pages 48:1–4:21, 2017.

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

# A The Fundamental Property of the Alternant

**Lemma A.1.** *[Kay11, Claim 7] Let* $\boldsymbol{f} = \{f_1(X), f_2(X), \ldots, f_k(X)\} \subseteq \mathbb{F}[X]$. *Let* $X_1, \ldots, X_k$ *be disjoint sets of variables each of size* $n$. *Define:*

$$Q \triangleq \begin{pmatrix} f_1(X_1) & f_2(X_1) & \cdots & \cdots & f_m(X_1) \\ f_1(X_2) & f_2(X_2) & \cdots & \cdots & f_m(X_2) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ f_1(X_k) & f_2(X_k) & \cdots & \cdots & f_m(X_k) \end{pmatrix}.$$

*Then the polynomials in* $\boldsymbol{f}$ *are* $\mathbb{F}$*-linearly independent iff* $\det(Q) \not\equiv 0$.

*Proof.* We will show, via induction on $k$, that $Q$ has full rank, or equivalently, the determinant of $Q$ is a non-zero polynomial. Note that $k = 1$ follows directly. On expanding $Det(Q)$ along the first row we get,

$$Det(Q) = \sum_{j=1}^{k} (-1)^{j+1} f_j(X_1) Q_{1j}$$

where $Q_{ij}$ is the determinant of the $ij$-th minor. Notice that every $Q_{1j}$ , $j \in [k]$, is a polynomial in the set of variables $X_2, ..., X_k$. By induction, every $Q_{1j}$ is a nonzero polynomial (since every subset of a set of $\mathbb{F}$-linearly independent polynomials is also $\mathbb{F}$-linearly independent). If $Det(Q)$ was the zero polynomial then plugging in random values for $X_2, ..., X_k$ would give us a nonzero $\mathbb{F}$-linear dependence among $f_1(X_1), f_2(X_1), ..., f_k(X_1)$, which is a contradiction. Hence, $Det(Q)$ must be nonzero. □

To obtain the full version of Lemma 4.7 is obtained by treating a (different) set of variables $Z$ as field constants in $\mathbb{F}(Z)$.



ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il