Electronic Colloquium on Computational Complexity, Report No. 125 (2022)

Tensor Reconstruction Beyond Constant Rank

Shir Peleg[*] Amir Shpilka[*] Ben Lee Volk[†]

Abstract

We give reconstruction algorithms for subclasses of depth-3 arithmetic circuits. In particular, we obtain the first efficient algorithm for finding tensor rank, and an optimal tensor decomposition as a sum of rank-one tensors, when given black-box access to a tensor of super-constant rank. Specifically, we obtain the following results:

1. A deterministic algorithm that reconstructs polynomials computed by $\Sigma^{[k]} \bigwedge^{[d]} \Sigma$ circuits in time $\mathsf{poly}(n, d, c) \cdot \mathsf{poly}(k)^{k^{k^{10}}}$,
2. A randomized algorithm that reconstructs polynomials computed by multilinear $\Sigma^{[k]} \prod^{[d]} \Sigma$ circuits in time $\mathsf{poly}(n, d, c) \cdot k^{k^{k^{k^{O(k)}}}}$,
3. A randomized algorithm that reconstructs polynomials computed by set-multilinear $\Sigma^{[k]} \prod^{[d]} \Sigma$ circuits in time $\mathsf{poly}(n, d, c) \cdot k^{k^{k^{k^{O(k)}}}}$,

where $c = \log q$ if $\mathbb{F} = \mathbb{F}_q$ is a finite field, and c equals the maximum bit complexity of any coefficient of f if $\mathbb{F}$ is infinite.

Prior to our work, polynomial time algorithms for the case when the rank, k, is constant, were given by Bhargava, Saraf and Volkovich [BSV21].

Another contribution of this work is correcting an error from a paper of Karnin and Shpilka [KS09a] (with some loss in parameters) that also affected Theorem 1.6 of [BSV21]. Consequently, the results of [KS09a, BSV21] continue to hold, with a slightly worse setting of parameters. For fixing the error we systematically study the relation between syntactic and semantic notions of rank of $\Sigma\Pi\Sigma$ circuits, and the corresponding partitions of such circuits.

We obtain our improved running time by introducing a technique for learning rank preserving coordinate-subspaces. Both [KS09a] and [BSV21] tried all choices of finding the "correct" coordinates, which, due to the size of the set, led to having a fast growing function of k at the exponent of n. We manage to find these spaces in time that is still growing fast with k, yet it is only a fixed polynomial in n.

[*] Blavatnik School of Computer Science, Tel Aviv University, Tel Aviv, Israel. Emails: shirpele@tauex.tau.ac.il, shpilka@tauex.tau.ac.il. The research leading to these results has received funding from the Israel Science Foundation (grant number 514/20) and from the Len Blavatnik and the Blavatnik Family foundation.

[†] Efi Arazi School of Computer Science, Reichman University, Israel. Email: benleevolk@gmail.com.

ISSN 1433-8092

Contents

1 Introduction

Reconstruction of algebraic circuits is a natural algorithmic problem that asks, given a black box access to a polynomial f from some circuit class $\mathcal{C}$, to efficiently output an algebraic circuit computing f. Algebraic circuits are computational devices that compute multivariate polynomials using basic arithmetic operations, much like boolean circuits compute boolean functions using boolean bit operations. Thus, the reconstruction problem is a natural algebraic analog for well studied boolean learning problems [Bsh13].

It is often desired that the output of the algorithm will also be a circuit from the class $\mathcal{C}$ (which is called *proper learning*). Requiring the learning algorithm to be efficient imposes an obvious upper bound on the size of the output, but it is also desirable to output a circuit as small as possible, ideally the smallest possible circuit from the class $\mathcal{C}$ that computes f.

Reconstruction, however, is also a hard algorithmic problem. Results such as the NP hardness of computing or even approximating tensor rank [Hås90, Shi16, BIJL18, Swe18] force us to carefully manage our expectations regarding what's possible to compute efficiently, since it turns out that even for weak classes $\mathcal{C}$ (such as depth-3 set-multilinear circuits) it's unlikely to find an efficient algorithm that outputs the smallest possible circuit. Furthermore, reconstruction appears to be an even harder problem than black box Polynomial Identity Testing (PIT), the problem of determining whether the black box f computes the identically zero polynomial. While PIT can be efficiently solved using randomness, efficient deterministic algorithms are known only for a handful of restricted circuit classes (we note, though, that in the reconstruction problem even giving a randomized algorithm is a non-trivial task). For a survey on algebraic circuits, PIT and reconstruction, see [SY10].

Nevertheless, for some restricted classes, or when the constraints are sufficiently relaxed, it is possible to give many non-trivial efficient reconstruction algorithms. For example, many works have dealt with *random* algebraic circuits (see, e.g., [GKL11, KNS19, GKS20], among others). In this setting, we think of the black box as being chosen randomly from the class $\mathcal{C}$ under some natural distribution on circuits from $\mathcal{C}$, and we require the algorithm to reconstruct f with high probably over the chosen circuit (and perhaps over the random coins of the algorithm as well). Random circuits often avoid the degeneracies and pathologies that are associated with the clever cancellations that facilitate sophisticated algebraic algorithms, and are thus easier to handle and argue about.

Another line of study, more relevant to our work, has to do with reconstruction of small depth algebraic circuits. The simplest non-trivial case is depth-2 circuits, for which the reconstruction problem is pretty well understood and can be done efficiently [BT88, KS01]. However, even slightly larger depths, like depth-3 and depth-4 circuits, already pose a much greater challenge. This is perhaps explained by a sequence of depth reduction results [AV08, Koi12, Tav15, GKKS16] that show that low depth circuits are expressive enough to non-trivially simulate any algebraic circuit of polynomial size (and arbitrary depth). Thus, most attention has focused on restricted classes of depth-3 and depth-4 circuits [KS09a, GKL12, Sin16, Sin22, BSV20, BSV21].

1.1 Circuit Classes

A depth-3 circuit with top fan-in (that is, the in-degree of the top sum gate) k computes a polynomial of the form $\sum_{i=1}^{k}\prod_{j=1}^{d_i}\ell_{i,j}(\mathbf{x})$, where each $\ell_{i,j}$ is a linear function in the input variables $\mathbf{x}$. We denote this class $\Sigma^k\Pi\Sigma$. When k is constant, this is a subclass of general depth-3 circuits that has been extensively studied (see Section 4.6 of [SY10]).

The circuit is called *multilinear* if every gate in the circuit computes a multilinear polynomial. An even stronger restriction is *set-multilinearity*. A polynomial f is set-multilinear if the set of variables $\mathbf{x}$ can be partitioned to disjoint sets $\mathbf{x}_1, \dots, \mathbf{x}_d$ such that every monomial appearing in f is a product of variables $x_{1,i_1}x_{2,i_2}\cdots x_{d,i_d}$ such that x_{j,i_j} is in $\mathbf{x}_j$. That is, a degree-d set-multilinear polynomial is simply a d-dimensional tensor. Depth-3 set-multilinear circuits, which are circuits in which every gate computes a multilinear polynomial, are a natural model for computing tensors. Each product of linear functions $\prod_{j=1}^{d}\ell_j(\mathbf{x}_j)$ corresponds to a rank one tensor, and thus we see that f can be computed by a set-multilinear circuit of top fan-in k if and only if its rank is at most k.

Finally, the most restricted model we study is *depth-3 powering circuits*. In this model, multiplication gates are replaced by powering gates. Such gates get as input a single linear function and their output is that function raised to some power. We denote the class of depth-3 powering circuit by $\Sigma^k\wedge\Sigma$. This is a natural computational model for computing *symmetric tensors*, where again the top fan-in corresponds to the rank.

In a recent work, Bhargava, Saraf, and Volkovich [BSV21] presented proper reconstruction algorithms for these circuit models. The running times of their algorithms are polynomial in n, the number of variables, and the degree d, assuming k is constant, but not when k is any growing function of n or d. The exact running time is a polynomial whose exponent is a somewhat complicated expression that involves some quickly growing function of k. We describe their results more precisely vis-à-vis our results in Section 1.2.

In particular, given a constant upper bound on the rank, they obtain efficient algorithms that given a tensor (or a symmetric tensor) can exactly compute its rank, and also obtain a decomposition as a sum of rank-one tensors. Since for large enough ranks the problem of computing the tensor rank becomes NP hard, it's natural to wonder at which point the intractability kicks in. That is, is there an efficient polynomial time algorithm that can compute the rank and obtain a decomposition even when the upper bound k is super-constant?

In this paper we obtain faster algorithms that remain polynomial time algorithms (in n and d) even when k is slightly super-constant. Our running times are of the form $\mathsf{poly}(n, d, T(k))$ where T is some quickly growing function of k. Like the algorithms of Bhargava, Saraf, and Volkovich [BSV21], our learning algorithms are proper and return the smallest possible representation of f in the relevant circuit model. In particular, they imply efficient randomized algorithms for computing tensor rank even when the rank is slightly super-constant.

Another contribution of this work is correcting an error that appeared in previous work. This error originated in [KS09a] and affected Theorem 1.6 of [BSV21] as well.

Explaining the nature of the error requires some technical details that we present in Section 1.3.4. Our correction recovers the affected results of [KS09a, BSV21], albeit with a slight change in the parameters that implies a somewhat worse dependence on the parameter k.

Our algorithms require the field $\mathbb{F}$ to be large enough. The precise meaning of what "large enough" means depends on each case. The largeness assumption can always be guaranteed without loss of generality by considering field extensions, if necessary (in which case the output will also be a circuit over the extension field). In certain cases, we also assume that the characteristic of the field is large enough.

1.2 Our Results

We start by describing our results for depth-3 powering circuits.

Theorem 1.1. *There exists a randomized algorithm that, given a black box access to a polynomial f with n variables and degree d that is computed by a $\Sigma^k \wedge \Sigma$ circuit, reconstructs f in time $\mathsf{poly}(n,d,c) \cdot \mathsf{poly}(k)^{k^{k^{10}}}$, where $c = \log q$ if $\mathbb{F} = \mathbb{F}_q$ is a finite field, and c equals the maximum bit complexity of any coefficient of f if $\mathbb{F}$ is infinite.*

In [BSV21], the authors give an algorithm for a similar task that runs in time $\mathsf{poly}((dk)^{k^{k^{10}}}, n, c)$. Note that unlike the algorithm in [BSV21], when $d = \mathsf{poly}(n)$ our algorithms runs in polynomial time even when k is a slightly super-constant function of n (e.g., $k = (\log\log n / \log\log\log n - O(1))^{1/10}$). As in [BSV21], we can derandomize the algorithm from Theorem 1.1 over $\mathbb{R}$ or $\mathbb{C}$ and obtain a deterministic algorithm that runs in roughly the same time. Theorem 1.1 is proved in Section 3.

We also provide reconstruction algorithms for multilinear depth-3 circuits with top fan-in k.

Theorem 1.2. *There exists a randomized algorithm that, given a black box access to a polynomial f with n variables and degree d, which is computed by a $\Sigma^k\Pi\Sigma$ multilinear circuit, reconstructs f in time $\mathsf{poly}(n,d,c) \cdot k^{k^{k^{k^{O(k)}}}}$, where $c = \log q$ if $\mathbb{F} = \mathbb{F}_q$ is a finite field, and c equals the maximum bit complexity of any coefficient of f if $\mathbb{F}$ is infinite.*

Note that again, the algorithm in Theorem 1.2 runs in polynomial time for small enough (but super-constant) k, whereas the corresponding algorithm of [BSV21] had running time of roughly $n^{T(k)}$ for some quickly growing function $T(k)$.

Finally, we also present a reconstruction algorithm for set-multilinear depth-3 circuits. Note that even though this class is a subclass of the previous model of multilinear circuits, as long as we insist on proper learning, reconstruction algorithms for a more general class don't imply reconstruction algorithms for its subclasses.

Theorem 1.3. *There exists a randomized algorithm that, given a black box access to a polynomial $f(\mathbf{x}_1, \ldots \mathbf{x}_d)$ such that $|\mathbf{x}_i| \leq n$ for every $i \in [d]$, such that f is computed by a depth-3 set-multilinear circuit, reconstructs f in time $\mathsf{poly}(n,d,c) \cdot k^{k^{k^{O(k)}}}$, where $c = \log q$ if $\mathbb{F} = \mathbb{F}_q$ is a finite field, and c equals the maximum bit complexity of any coefficient of f if $\mathbb{F}$ is infinite.*

Unlike the algorithms from [BSV21] and our algorithm from Theorem 1.1, we don't know how to derandomize the algorithms from Theorem 1.2 and Theorem 1.3, even over $\mathbb{R}$ or $\mathbb{C}$. This remains an interesting open problem.

1.3 Proof Technique

There are two main factors contributing to the doubly or triply exponential dependence on k in the time complexity of the algorithms in [BSV21].

The first is the fact that their algorithms solve systems of polynomial equations. This is required in order to find brute force solutions for the reconstruction problem over various projections of f to a few variables, making the number of variables in the polynomial system of equations rather small (that is, only a function of k, and not of n). They then calculate the running time using the best known algorithms for solving such systems of polynomial equations. The exact running time depends on the field, and it is typically singly or doubly exponential time in the number of variables.

Our main observation is that in all of these cases, it is also possible to modify the algorithms so that the *degree* of the polynomial system of equations and the *number of equations* are also only functions of k (and not of n or d, the number of variables and degree of the original polynomial f).

The second reason their algorithms run in time $n^{T(k)}$ is a construction of an object called "rank preserving subspace", which is a subset of the coordinates that preserves certain properties of the polynomial, as we explain in Section 1.3.2. The dimension of this subspace depends on k, but finding it involves enumerating over all possible subsets of coordinates of the relevant size. As we soon explain, overcoming this difficulty requires a substantial amount of work.

1.3.1 $\Sigma^k \wedge \Sigma$ circuits

In the case of Theorem 1.1, getting the degree of the polynomial system of equations to be small is done in a simple way, by simply learning a high order derivative of f instead of f itself. Of course, when we take derivatives of f we have to make sure that we don't lose too much information so that the learning problem for the derivative becomes trivial but useless. We find a small sets of vectors $S \subseteq \mathbb{F}^n$ such that given any black box access to a polynomial computed by a depth-3 powering circuit, $f = \sum_{i=1}^{k} c_i \ell_i^d$, there exists $\mathbf{u} \in S$ such that in the iterated directional derivative $g := \partial^{d-2k-1} f / \partial \mathbf{u}^{d-2k-1}$ none of the linear functions "disappear": that is, g itself is a depth-3 powering circuit with top fan-in k and exactly the same k linear functions $\ell_1, \dots, \ell_k$ appearing in the circuit, perhaps with different coefficients (that are easily computable functions of the original coefficients). We can then learn g using the algorithm of [BSV21], except that g is a polynomial of degree $2k+1$, so that the dependence on d of our algorithm is much more tame.

1.3.2 Multilinear and Set-Multilinear $\Sigma^k\Pi\Sigma$ Circuits

The proofs of Theorem 1.2 and Theorem 1.3 can be broken down to two parts, the first handles low degree polynomials and the second high degree polynomials. The analysis of both parts in [BSV21] incurs factors of the form $n^{T(k)}$, which we would like to eliminate. While the proof of the low degree case follows the general outline of [BSV21], the proof of the high degree case is significantly more challenging and requires new ideas. As we describe later, the proof of the high degree case in [BSV21], for multilinear $\Sigma^k\Pi\Sigma$ circuits, contains an error originating in [KS09a]. We are able to correct this error (see Theorem 4.11), but even this correction doesn't suffice for improving the running time and a new approach is needed. Their result for set-multilinear circuits were not affected by this error as they use a different proof technique in the high degree case.

The low degree case: when the degree d is small, the number of linear functions in the circuit, which is bounded by kd, is also small so that we can allow ourselves to try and learn the circuit for f in an almost brute-force manner by solving a system of polynomial equations. Following [BSV21] we first find, in polynomial time, an invertible linear transformation A so that $g := f(A\mathbf{x})$ depends on a few *variables* (and not merely linear functions). We then obtain a low-degree polynomial in a small number of variables, so that we can allow ourselves to learn the new circuit by solving a set of polynomial equations whose variables are the coefficients of the purported small circuit.

We then wish to output $g(A^{-1}\mathbf{x})$. The problem is that this circuit may not be multilinear. To solve this, [BSV21] introduce an additional set of $\approx \mathsf{poly}(n)$ low degree polynomial equations to guarantee that the output circuit is multilinear. This results in a running time of about $n^{T(k)}$ for this part alone. We observe however that this set is highly redundant in the sense that, by dimension arguments, many of these equations are linearly dependent. By finding a basis to the polynomial system of equations and solving that basis alone, we're able to reduce the running time to $n \cdot T'(k)$ for some different function $T'(k)$.

Our algorithm for low-degree set-multilinear circuits is very similar but a bit simpler. By slightly tweaking the polynomial system of equations that describes the circuit, we can learn $f(A\mathbf{x})$ as a set-multilinear circuit. Further, in this case it's possible to find A such that $g(A^{-1}\mathbf{x})$ will automatically be set-multilinear, so that the challenge described in the previous paragraph doesn't exists in this setting.

The high degree case: this is the more complicated and tedious part of the argument. We start by explaining the high level approach of [BSV21].

The *(syntactic) rank* of a $\Sigma^k\Pi\Sigma$ circuit is defined to be the dimension of the span of the linear functions appearing in its multiplication gates, after factoring out the greatest common divisors of these gates (that is, the linear functions appearing in all of them). For more details see Definition 4.1. This is a well studied notion originating in the work of Dvir and Shpilka [DS07] and used in many later works [KS08, KS09a, SS11, SS13, KS09b]. The rank function allows one to define the *distance* between two circuits C_1 and C_2 as the rank of their sum.

The algorithm of [BSV21] for learning multilinear $\Sigma^k\Pi\Sigma$ circuits relies on a structural property of such circuits claimed by Karnin and Shpilka [KS09a]. Karnin and Shpilka [KS09a] partition the k multiplication gates in the circuit to *clusters*, so that each cluster has a low rank and each two distinct clusters have a large distance. In [KS09a], it is claimed that for some choice of parameters, this partition is unique and depends only on the polynomial computed by the circuit and not on the circuit itself (this is where the error is, and this is what is being corrected in Theorem 4.11). Thus, the authors of [BSV21] try to obtain black box access to each of the clusters. Then, factoring out their greatest common divisors they can reconstruct them as, by multilinearity, the remaining part is a low degree polynomial.

Obtaining black box access to the clusters is most of the technical work in the proof of Theorem 1.6 of [BSV21]. In a high level, using their uniqueness result, Karnin and Shpilka [KS09a] claimed to prove the existence of a small "rank preserving subset" of the variables B, such that after randomly fixing the variables outside of B, the remaining circuit $C|_B$ has the property that its clusters are in one-to-one correspondence with the original clusters restricted to B. The circuit $C|_B$ can again be reconstructed using the low-degree case, as it only involves a small number of variables, and thus we can get direct access to its clusters. Using a clever algorithm, Bhargava, Saraf and Volkovich [BSV21] are able to obtain evaluations of the original clusters using evaluations of the restricted clusters.

Regardless of the correctness issue that we discuss soon, a big bottleneck of this argument is that one needs to iterate over all subsets B of $[n]$ up to a certain size bound (that depends only on k). Clearly such a procedure requires running time of the form $n^{T(k)}$.

Thus, we would like to obtain an algorithm that explicitly constructs a set B. One natural approach is to start with the empty set and add one variable at a time. This can be done by reconstructing the polynomial f restricted to the current set B, and its clusters, and checking whether adding a variable to B changes one of the parameters. If so then we add the variable and repeat the process. We continue as long as either the number of clusters or the rank of a cluster increases. The challenge with this approach is that the uniqueness guarantee of Theorem 4.11 does not suffice. Note that if f has a $\Sigma^k\Pi\Sigma$ circuit C, f restricted to B as a natural $\Sigma^k\Pi\Sigma$ circuit obtained by restricting the circuit C to B. However, our low-degree algorithm learns *some*, and potentially different, $\Sigma^k\Pi\Sigma$ circuit that computes the restriction of f to B. While Theorem 4.11 guarantees both circuits would have the same number of clusters, computing the same polynomials, we don't have the guarantee that the *rank* of each cluster is the same a the different circuits, as the rank may depend on the circuit. Thus, as we gradually increase B, it seems hard to compare rank of clusters between different representations.

To circumvent that we introduce *semantic* versions of ranks and distances, which are properties of a *polynomial* and not of a circuit computing it. In fact, our version of the semantic rank was already introduced by Karnin and Shpilka in [KS08, KS09a], in the context of learning so-called $\Sigma\Pi\Sigma(k,d,\rho)$ circuits. Their notion of "rank" for such circuits is a certain hybrid between syntactic and semantic rank. Since in our case the distinction is important, we try to mention explicitly whether we mean syntactic or semantic rank.

1.3.3 Semantic Notions of Rank

The semantic rank of a polynomial f is defined as follows: first write $f = \prod_i \ell_i \cdot h$, where the ℓ_i's are linear functions and h has no linear factors. Then define the semantic rank of f to be the minimal number r such that h depends on r linear functions.

This number is well defined and doesn't depend on any representation of f as a $\Sigma^k\Pi\Sigma$ circuit. Working with semantic rank has advantages and disadvantages. On the one hand, it is now possible to prove stronger uniqueness properties regarding the clusters, since if two clusters compute the same polynomial then they also have the same rank. Indeed we prove such a uniqueness statement for some parameters. On the other hand, analyzing the semantic rank and its behavior under various operations (such as restricting the circuit to a subset of the variables, or increasing the set B using the approach mentioned above) is significantly more difficult. Thus, we also prove various connections between semantic and syntactic ranks and we are able to show that if f is computed by an $\Sigma^k\Pi\Sigma$ circuit C, then the semantic and syntactic ranks of C are not too far apart.

Recall that our main challenge is to explicitly construct a cluster-preserving subset B of the variables, whose existence for syntactic ranks was proved by [KS09a] (see Section 1.3.4 for a discussion of this result). In the context of semantic rank, proving such an analogous statement is significantly more challenging. In fact, while the proof of [KS09a] is existential (and then the algorithm of [BSV21] essentially enumerates over all possible subspaces), our proof is algorithmic (see Algorithm 3). In essence, our algorithm follows the outline described above: it starts with the empty set and on each iteration adds a few variables to B until the cluster structure "stabilizes", i.e., their number and their ranks stay the same. Proving that this algorithm works requires a significant amount of technical work.

1.3.4 The Errors in Previous Work and Our Corrections

Explaining the nature of the erroneous statements appearing in [BSV21, KS09a] requires giving some more technical details.

As mentioned earlier, one of the main components in the reconstruction algorithm for multilinear $\Sigma^k\Pi\Sigma$ circuits given in [BSV21] is the uniqueness of clusters property for such circuits, which is claimed by Karnin and Shpilka [KS09a]. Note that the rank and distance measures for $\Sigma^k\Pi\Sigma$ circuits are *syntactic* and inherently tied to a circuit. Karnin and Shpilka [KS09a] define a *clustering* algorithm that, given a circuit, partitions the k multiplication gates into several sets such that the rank of the subcircuit corresponding to each set is small, and the distance between every pair of subcircuits is large.

In Corollary 6.8 of [BSV21] it is claimed, based on [KS09a], that these clusters are unique, even among different circuits that compute the same polynomial. That is, if C and C' are two circuits computing the same polynomial f, the clustering algorithm of [KS09a] would return the same clusters (perhaps up to a permutation). Such a claim can indeed be read from Theorem 5.3 of [KS09a]. However, in our judgment, the paper [KS09a] does not contain a valid mathematical proof for such a statement.

Karnin and Shpilka associate with each partition to clusters two parameters, κ

and r. The parameter r upper bounds the rank of each cluster, and the parameter κ controls the distance: their clustering algorithm guarantees that each pair of clusters has distance at least κr. Consequentially, their clustering algorithm receives κ as an additional input, and outputs a clustering with parameters κ and r for some value of r that can be upper bounded as a function of κ and k.

The proof of Theorem 5.3 of [KS09a] assumes without justification that, given two different circuits C and C' computing the same polynomial, the clustering algorithm with parameters κ would return partitions with the same value of the parameter r, which is crucially used in their proof.

In this work we provide a corrected proof of Theorem 5.3 of [KS09a] (Theorem 4.11). While the corrected version is not identical to the original statement word-for-word (as our parameter κ is much larger than originally stated as a function of k), it suffices for fixing the arguments in [KS09a] and [BSV21], with the straightforward corresponding changes in parameters throughout.

We wish to stress again that Theorems 1.1 and 1.4 of [BSV21], that give algorithms for learning depth-3 set-multilinear and depth-3 powering circuits, respectively, are not affected by the error in [KS09a].

1.4 Open Problems

One natural problem our work raises is the question of how large the top fan-in k needs to be before reconstruction problem becomes intractable. The NP-hardness results for tensor rank imply that clearly when $k = \mathsf{poly}(n)$ we shouldn't expect to find exact proper learning algorithm, whereas we show that the intractability barrier is not at the regime when k is constant. It remains an interesting problem to bridge the gap.

Another interesting problem is derandomizing our algorithms from Theorem 1.2 and Theorem 1.3. In Remark 7.3 we explain why we cannot derandomize our algorithm with the same improved running time.

2 Preliminaries

The following notation will be very useful throughout our paper.

Definition 2.1. *For $\mathbf{a} \in \mathbb{F}^n$, $B \subseteq [n]$, and a polynomial $f \in \mathbb{F}[x_1, \dots, x_n]$ we define $f|_{B,\mathbf{a}}$ the polynomial obtained by fixing $x_j = \mathbf{a}_j$ for every $j \notin B$.* ♢

2.1 Black Box Access to Directional Derivatives

Lemma 2.2. *Let $\mathbb{F}$ be a field of size at least $d+1$ and let $f(x_1, \dots, x_n) \in \mathbb{F}[x_1, \dots, x_n]$ be a polynomial of degree d. Given a black box access to f, for every $e \leq d$ and for every variable $x \in \{x_1, \dots, x_n\}$, we can simulate a black box access to $g := \partial^e f / \partial x^e$ using at most $d+1$ queries to f.*

Proof. Without loss of generality assume $x = x_1$. Write $f(x) = \sum_{i=0}^{d} f_i(x_2, \dots, x_n) \cdot x_1^i$, so that

$$g = \frac{\partial f}{\partial x_1^e} = \sum_{i=e}^{d} \left(\prod_{t=0}^{e-1} (i-t) \right) f_i(x_2, \ldots, x_n) x_1^{i-e}. \tag{2.3}$$

Pick arbitrary distinct $\alpha_1, \ldots, \alpha_{d+1} \in \mathbb{F}$. Since $P(X) := \sum_{i=0}^{d} f_i(x_2, \ldots, x_n) X^i$ is a univariate polynomial in X of degree d, by standard polynomial interpolation there are (efficiently computable) elements $\beta_{i,j} \in \mathbb{F}$, where $i, j \in \{0, \ldots, d\}$, such that for every $i \in \{0, \ldots, d\}$,

$$f_i(x_2, \ldots, x_n) = \sum_{j=0}^{d} \beta_{i,j} P(\alpha_j). \tag{2.4}$$

Suppose now we would like to evaluate g at the point $\mathbf{c} = (c_1, \ldots, c_n)$. We first compute $f_i(c_2, \ldots, c_n)$ for every $i \in \{0, \ldots, d\}$ using the relation (2.4). This requires a total of $d+1$ black box evaluations of f, for evaluating $f(\alpha_j, c_2, \ldots, c_n) = P(\alpha_j)$ for every $j \in \{0, \ldots, d\}$. Given $f_0, \ldots, f_d$, we can evaluate g at $\mathbf{c}$ directly using the relation (2.3) by plugging in $x_1 = c_1$. □

Lemma 2.2 can be generalized to directional derivatives.

Lemma 2.5. *Let $\mathbb{F}$ be a field of size at least $d+1$ and let $f(x_1, \ldots, x_n) \in \mathbb{F}[x_1, \ldots, x_n]$ be a polynomial of degree d. Given a black box access to f, for every $e \le d$ and for every $0 \neq \mathbf{u} \in \mathbb{F}^n$, we can simulate a black box access to $g := \frac{\partial f}{\partial \mathbf{u}^e}$ using at most $d+1$ queries to f.*

Proof. Let A be an invertible matrix such that $A\mathbf{e}_1 = \mathbf{u}$ (that is, the first column of A is $\mathbf{u}$). Further, define $f_A(\mathbf{x}) = f(A\mathbf{x})$. Clearly, we can simulate black box access to f_A using black box access to f. Further, note that by the chain rule, for every $\mathbf{c} \in \mathbb{F}^n$

$$\frac{\partial f_A}{\partial x_1}(\mathbf{c}) = \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}(A\mathbf{c}) \cdot A_{i,1} = \sum_{i=1}^{n} u_i \frac{\partial f}{\partial x_i}(A\mathbf{c}).$$

Thus, in order to evaluate

$$\frac{\partial f}{\partial \mathbf{u}}(\mathbf{x}) = \sum_{i=1}^{n} u_i \frac{\partial f}{\partial x_i}(\mathbf{x})$$

at any point $\mathbf{c}'$, we can evaluate $\frac{\partial f_A}{\partial x_1}$ at the point $A^{-1}\mathbf{c}'$ using Lemma 2.2.

For higher order derivative, we use the same method, that is, we simulate $\partial^e f / \partial \mathbf{u}^e$ using black box access to $\partial^e f_A / \partial x_1^e$ (as guaranteed by Lemma 2.2), which is in turn simulated by black box access to f itself. □

2.2 Essential Variables

Let f be an n-variate polynomial. We say that f depends on m essential variables if there exists an invertible linear transformation A such that $f(A\mathbf{x})$ depends on m variables. An interesting fact is that it's possible, given a black box access to f, to compute a linear transformation A such that $g := f(A\mathbf{x})$ depends only on $x_1, \ldots, x_m$.

Lemma 2.6 ([Kay11, Car06])**.** *Let $f \in \mathbb{F}[\mathbf{x}]$ be an n-variate polynomial of degree d with m essential variables, where $\mathrm{char}(\mathbb{F}) = 0$ or $\mathrm{char}(\mathbb{F}) > d$. Suppose f is computed by a circuit of size s. Then, there's an efficient randomized algorithm that, given black box access to f, runs in time $\mathsf{poly}(n, d, s)$ and computes an invertible linear transformation A such that $f(A\mathbf{x})$ depends on the first m variables $x_1, \ldots, x_m$.*

Bhargava, Saraf and Volkovich [BSV21] derandomize this lemma when f is computed by a $\Sigma^k \wedge \Sigma$ circuit, a depth-3 set-multilinear circuit of top fan-in k or a depth-3 multilinear circuit of top fan-in k. However the time required for their derandomization involves factors of $n^{O(k)}$ and thus we want to obtain an improved running time.

For a class of polynomials $\mathcal{C}$, we denote by $\Sigma^t\mathcal{C}$ the class of polynomials of the form $\alpha_1 f_1 + \alpha_2 f_2 + \cdots + \alpha_t f_t$ with $\alpha_i \in \mathbb{F}$ and $f_i \in \mathcal{C}$ for every i.

Lemma 2.7. *Let $\mathcal{C}$ be a class of polynomials and let $f_1, \ldots, f_t \in \mathcal{C}$. Let $\mathcal{H}$ be a hitting set for $\Sigma^t\mathcal{C}$. Denote by $f_i|_{\mathcal{H}}$ the vector (of length $|\mathcal{H}|$) $(f_i(\beta))_{\beta \in \mathcal{H}}$. Then for any $\alpha_1, \ldots, \alpha_t \in \mathbb{F}$,*

$$\sum_{i=1}^{t} \alpha_i f_i = 0 \iff \sum_{i=1}^{t} \alpha_i f_i|_{\mathcal{H}} = 0.$$

In particular, the polynomials $f_1, \ldots, f_t$ are linearly independent if and only if the vectors $f_1|_{\mathcal{H}}, \ldots, f_t|_{\mathcal{H}}$ are linearly independent.

Proof. The implication from left to right is clear. In the other direction, $\sum_{i=1}^{t} \alpha_i f_i|_{\mathcal{H}} = (\sum_{i=1}^{t} \alpha_i f_i)|_{\mathcal{H}}$. Since $\sum_{i=1}^{t} \alpha_i f_i \in \Sigma^t\mathcal{C}$ and $\mathcal{H}$ is a hitting set, it follows that $\sum_{i=1}^{t} \alpha_i f_i = 0$. □

Lemma 2.7 gives an efficient way to test for dependency of polynomials assuming the existence of a small and efficiently constructible hitting set for $\Sigma^t\mathcal{C}$.

A derandomized version of Lemma 2.6 is given below.

Lemma 2.8. *Let $\mathcal{C}$ be a class of polynomials closed under taking first order partial derivatives. Given a black box access to a degree-d polynomial $f(\mathbf{x}) \in \mathbb{F}[\mathbf{x}]$ that has t essential variables, denote by $|\mathcal{H}|$ a hitting set for $\Sigma^{t+1}\mathcal{C}$. Then, there's a deterministic algorithm that runs in time $\mathsf{poly}(n, d, |\mathcal{H}|)$ and outputs an invertible matrix $A \in \mathbb{F}^{n \times n}$ such that $f(A\mathbf{x})$ depends only the first t variables.*

Proof. As noted by [BSV21], the only place in which randomness is used in the proof of Lemma 2.6 is in finding a basis for the vector space

$$V = \left\{ \mathbf{a} \in \mathbb{F}^n : \sum_{i=1}^{n} a_i \frac{\partial f}{\partial x_i} = 0 \right\}.$$

It turns out that $\operatorname{codim} V = t$ where t the number of essential variables of f. Let $f_i = \frac{\partial f}{\partial x_i}$. By assumption $f_i \in \mathcal{C}$, and further using Lemma 2.2 we can obtain black box access to each f_i. Using Lemma 2.7, we can greedily pick t linearly independent polynomials among $f_1, \ldots, f_n$, as well as, for each element f_j, compute the coefficients that express it as a linear combination of basis vectors (note that this requires applying

Lemma 2.7 on at most $t+1$ polynomials). That is, as in [BSV21], we compute a matrix $M \in \mathbb{F}^{n \times t}$ and indices $i_1, \ldots, i_t$ such that

$$M \begin{pmatrix} f_{i_1} \\ \vdots \\ f_{i_t} \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_{n-1} \\ f_n \end{pmatrix}$$

The left kernel of M is V. □

We note that the models we consider in this work are all closed under first order partial derivatives.

2.3 Hitting Sets for Depth-3 Circuits

While there exist hitting sets of size $n^{O(\log\log n)}$ for depth-3 powering circuits [FSS14] and quasi-polynomial size hitting sets for depth-3 set-multilinear circuits [FS13, FSS14, AGKS15], we insist on obtaining polynomial size hitting sets for these models when k is slightly super-constant (when k is constant there are hitting sets of size $n^{\mathsf{poly}(k)}$ for general depth-3 circuits of top fan-in k, see, e.g., Section 4.6.2 of [SY10] and [SS12]). Guo and Gurjar constructed such explicit polynomial size hitting sets for read-once algebraic branching programs (roABPs) of super-constant width.

Theorem 2.9 ([GG20])**.** *There's an explicit hitting set of size* $\mathsf{poly}(n,d)$ *for the class of n-variate, individual degree d polynomials computed by any-order roABPs of width w, assuming there's a constant $\varepsilon > 0$ such that $w = 2^{O(\log^{1-\varepsilon}(nd))}$.*

Corollary 2.10. *There's an explicit hitting set of size* $\mathsf{poly}(n,d)$ *for the class of set-multilinear polynomials computed by depth-3 set-multilinear circuits of degree d and top fan-in $k = 2^{O(\log^{1-\varepsilon}(nd))}$.*

Proof. Write $\mathbf{x} = \mathbf{x}_1 \cup \cdots \cup \mathbf{x}_d$ for the set of variables of f. Note that by substituting $x_{i,j}$ by y_i^j we obtain a polynomial $\tilde{f}$ in $\mathbf{y} = (y_1, \ldots, y_d)$, of individual degree n, which is non-zero if and only if f is non-zero. Further, $\tilde{f}$ is naturally computed by an any-order roABP of width k: we convert every multiplication gate to a path of width 1, and connect them in parallel. Thus we get a hitting set for f of the required size. □

We can also reduce depth-3 powering circuits to roABPs.

Lemma 2.11 ([FS13])**.** *Suppose f is computed by a depth-3 powering circuit of top fan-in k and degree d. Then f is computable by an any-order roABP of width $O(dk)$, size* $\mathsf{poly}(dk, n)$ *and degree d.*

We only use Lemma 2.11 when k is very small and the degree d is commensurate with k. Thus, since the parameters of Theorem 2.9 are quite comfortable, we can deduce.

Corollary 2.12. *Suppose d and k are both $2^{O(\log^{1-\varepsilon}(n))}$. Then, there's an explicit hitting set of size $\mathsf{poly}(n)$ for the class of set-multilinear polynomials computed by a $\Sigma^k \wedge \Sigma$ circuits of degree d and top fan-in k.*

For general depth-3 circuits with top fan-in k, the known results are slightly weaker.

Lemma 2.13 ([SS12])**.** *There exists an explicit hitting set for the class of n-variate polynomials computed by multilinear $\Sigma^k\Pi\Sigma$ circuits of degree d of size $n^{(O(k^2 \log k)}$.*

We remark that the hitting set presented in [SY10] is of size $n^{O(R(k,r))}$ where $R(k,r)$ is the so-called rank bound for $\Sigma^k\Pi\Sigma$ circuits, which (for some fields, as explained in [SY10]) depends on d. However for *multilinear* circuits the above result is a corollary of Corollary 6.9 of [DS07] and the rank bounds of [SS13].

We further note that had we used Lemma 2.13, our algorithm wouldn't run in polynomial time for super-constant k, which is one of the reasons we use a randomized PIT algorithm for this class in our reconstruction algorithm. However, this is not the major obstacle for derandomization: derandomizing our algorithm in polynomial time would require a deterministic PIT for much larger classes than multilinear $\Sigma^k\Pi\Sigma$ circuits. It's an interesting open problem to obtain a derandomization for our algorithm even modulo Lemma 2.13.

2.4 Solving a System of Polynomial Equations

Let $\mathbb{F}$ be a field and let $\mathrm{Sys}_{\mathbb{F}}(n,m,d)$ denote the randomized time complexity of finding a solution to a polynomial system of m equations in n variables of degree d. A detailed analysis of this function for various fields $\mathbb{F}$ appears in Section 3.8 of [BSV21]. For our purposes, it is enough to note that for every field $\mathbb{F}$, $\mathrm{Sys}_{\mathbb{F}}(n,m,d) = \mathsf{poly}(nmd)^{n^n}$, if we allow solutions from an algebraic extension of $\mathbb{F}$. Further, for $\mathbb{F} = \mathbb{R}$, $\mathbb{C}$ or $\mathbb{F}_q$, extensions are not needed, and if $\mathbb{F} = \mathbb{R}$ or $\mathbb{C}$ then the algorithm is in fact deterministic.

2.5 Resultants

Let $f(x), g(x)$ be two polynomials of degrees m and ℓ in the variable x, respectively. Suppose $m, \ell > 0$, and write

$$f(x) = c_m x^m + x_{m-1}x^{m-1} + \cdots + c_0$$
$$g(x) = d_\ell x^\ell + d_{\ell-1}x^{\ell-1} + \cdots + d_0.$$

The *Sylvester matrix* of the polynomials f and g with respect to the variable x is the following $(m+\ell) \times (m+\ell)$ matrix:

$$\begin{pmatrix}
c_m & & & & d_\ell & & & \\
c_{m-1} & c_m & & & d_{\ell-1} & & & \\
c_{m-2} & c_{m-1} & \ddots & & d_{\ell-2} & d_{\ell-1} & \ddots & \\
\vdots & & \ddots & c_m & \vdots & & \ddots & d_\ell \\
 & \vdots & & c_{m-1} & & \vdots & & d_{\ell-1} \\
c_0 & & & & d_0 & & & \\
 & c_0 & & \vdots & & d_0 & & \vdots \\
 & & \ddots & & & & \ddots & \\
 & & & c_0 & & & & d_0
\end{pmatrix}$$

The determinant of this matrix is called the *resultant* of f and g with respect to the variable x and is denoted $\mathrm{Res}_x(f,g)$.

In our case we often think of $f,g \in \mathbb{F}[x_1,\ldots,x_n]$ interchangeably as n-variate polynomials or as univariate polynomials in some variable, say x_1, over the ring $\mathbb{F}[x_2,\ldots,x_n]$, in which case the resultant is a polynomial in $x_2,\ldots,x_n$. The main property of resultant we use is that, assuming the degree in x_1 of both f and g is positive, f and g have a common factor in $\mathbb{F}[x_2,\ldots,x_n]$ if and only if $\mathrm{Res}_{x_1}(f,g) = 0$ (see, e.g., Proposition 3 in Chapter 3, Section 6 of [CLO07]).

3 A Reconstruction Algorithm for Depth-3 Powering Circuits of Super-Constant Top Fan-in

For two non-zero affine functions $\ell_i, \ell_j \in \mathbb{F}[x_1,\ldots,x_n]$, we use the notation $\ell_i \sim \ell_j$ if ℓ_i is a constant multiple of ℓ_j.

Definition 3.1. *Let $\Sigma^k\wedge\Sigma$ denote the class of depth-3 powering circuits of top fan-in k. That is, if a degree d polynomial $f \in \mathbb{F}[x_1,\ldots,x_n]$ is computed by a circuit $C_f \in \Sigma^k\wedge\Sigma$ then $f = \sum_{i=1}^{k} c_i\ell_i^d$, and $\ell_i \in \mathbb{F}[x_1,\ldots,x_n]$ are of total degree 1.*

The representation $f = \sum_{i=1}^{k} c_i\ell_i^d$ is called minimal *if every i, $c_i \neq 0$ and every two indices $i \neq j$ satisfy $\ell_i \not\sim \ell_j$.* ◊

Theorem 3.2 (Theorem 4.1 of [BSV21])**.** *Given black-box access to a degree d polynomial $f \in \mathbb{F}[x_1,x_2,\ldots,x_n]$ such that f is computable by a $\Sigma^k\wedge\Sigma$ circuit C_f over a field $\mathbb{F}$ (of characteristic 0 or more than d), there is a randomized $\mathsf{poly}(n,c,(dk)^{k^{k^{10}}})$ time algorithm that outputs a minimal $\Sigma^k\wedge\Sigma$ circuit computing f, where $c = \log q$ if $\mathbb{F} = \mathbb{F}_q$ is a finite field, and c equals the maximum bit complexity of any coefficient of f if $\mathbb{F}$ is infinite..*

Remark 3.3. *In Theorem 4.1 of [BSV21] it is assumed that $f = \sum_{i=1}^{k} \ell_i^d$ whereas we allow each power of linear function to be multiplied by a field constant. By following the proof of [BSV21] it is immediate to see that their algorithm works for this slightly more general model. Further, since we can always easily modify the output of the algorithm to be a minimal $\Sigma^k\wedge\Sigma$ circuit, we might as well assume that the output of the algorithm is minimal.*

Another point is that the statement of Theorem 4.1 in [BSV21] suppresses the dependence on the bit-complexity (c in the statement above). We choose to explicitly state it so that the comparison to our result is clearer. ♢

Definition 3.4. *Let $\alpha \in \mathbb{F}$. We denote by $\mathbf{p}_\alpha$ the vector $(\alpha, \alpha^2, \ldots, \alpha^n) \in \mathbb{F}^n$ (where the length n is understood from the context).* ♢

We record the following basic claim.

Claim 3.5. *Let $0 \neq \ell \in \mathbb{F}[x_1, \ldots, x_n]$ be a polynomial of total degree one. Then there are at most n elements $\alpha \in \mathbb{F}$ such that $\ell(\mathbf{p}_\alpha) = 0$.*

Proof. Denote $\ell = a_0 + \sum_{i=1}^n a_i x_i$ where $a_0, a_1, \ldots, a_n \in \mathbb{F}$. For every $\alpha \in \mathbb{F}$, if $\ell(\mathbf{p}_\alpha) = 0$ then $a_0 + \sum_{i=1}^n a_i \alpha^i = 0$. It implies that α is a zero of the univariate polynomial $g(Z) = a_0 + \sum_{i=1}^n a_i Z^i$ of degree n. Since $\ell \neq 0$, it holds that $g \neq 0$, and thus there are at most n values $\alpha \in \mathbb{F}$ such that $g(\alpha) = 0$. □

The following is an immediate corollary of Claim 3.5.

Claim 3.6. *Let $\mathbb{F}$ be a field of size larger than kn. Let $0 \neq \ell_1, \ldots, \ell_k \in \mathbb{F}[x_1, \ldots, x_n]$ be polynomials of total degree one. Then, for every $S \subseteq \mathbb{F}$ of size $|S| > kn$, there is $\alpha \in S$ such that $\ell_i(\mathbf{p}_\alpha) \neq 0$ for every $1 \leq i \leq k$.*

The next lemma shows that when the degree d is larger than the top fan-in k, minimal depth-3 powering circuits cannot compute the zero polynomial.

Lemma 3.7. *Let $f = \sum_{i=1}^k c_i \ell_i^d$. Assume that the representation is minimal and that $d, |\mathbb{F}| \geq k+1$. Then $f \neq 0$.*

Before proving Lemma 3.7 we state and prove the following technical claim.

Claim 3.8. *Let $\ell_1, \ldots, \ell_k$ be linear functions such that for every $i \neq j$, $\ell_i \not\sim \ell_j$. Then, there exist $\mathbf{u}_1, \ldots, \mathbf{u}_k \in \mathbb{F}^n$ such that*

$$\ell_i(\mathbf{u}_j) = 0 \iff i = j.$$

Proof. For every i, let A_i denote the set of zeros of ℓ_i, which is an affine subspace of dimension $n-1$. We want to show that the set $A_i \cap \bigcap_{i \neq j} \overline{A_j}$ is non-empty so that we can pick $\mathbf{u}_i$ in that set. If it is empty, then

$$A_i \subseteq \bigcup_{i \neq j} A_j.$$

Intersecting both sides with A_i we have that $A_i \subseteq \bigcup_{i \neq j}(A_i \cap A_j)$. By assumption, each $A_i \cap A_j$ is either empty or an affine subspace of dimension $n-2$. As $|\mathbb{F}| \geq k+1$, it is not possible to cover an $n-1$ dimensional affine subspace with k affine subspace of dimension $n-2$, and hence the intersection is not empty. □

Proof of Lemma 3.7. Let $\mathbf{u}_1, \ldots, \mathbf{u}_k$ as promised by Claim 3.8. Consider the directional derivative $\frac{\partial f}{\partial \mathbf{u}_1 \cdots \partial \mathbf{u}_{k-1}}$. Since $d > k$, we have that

$$\frac{\partial f}{\partial \mathbf{u}_1 \cdots \partial \mathbf{u}_{k-1}} = \sum_{i=1}^{k} (d(d-1)\cdots(d-k+2)) c_i \ell_i(\mathbf{u}_1) \cdots \ell_i(\mathbf{u}_{k-1}) \ell_i^{d-k+1}.$$

Further, since for $i \leq k-1$ we have that $\ell_i(\mathbf{u}_i) = 0$. It follows that

$$\frac{\partial f}{\partial \mathbf{u}_1 \cdots \partial \mathbf{u}_{k-1}} = (d(d-1)\cdots(d-k+2)) c_k \ell_k(\mathbf{u}_1) \cdots \ell_k(\mathbf{u}_{k-1}) \ell_k^{d-k+1} \neq 0.$$

Therefore, $f \neq 0$. □

We now observe that Lemma 3.7 implies a uniqueness of representation result in the model of depth-3 powering circuits with bounded top fan-in.

Lemma 3.9. *Let $\mathbb{F}$ be a field such that $|\mathbb{F}| \geq 2k+1$, and let $d \geq 2k+1$. Suppose that $f = \sum_{i=1}^{k} c_i \ell_i^d$ and that the representation is minimal. Suppose that there exist another minimal representation of f as $f = \sum_{i=1}^{k} c_i' a_i^d$. Then, up to a permutation of the indices, $\ell_i \sim a_i$.*

Proof. By assumption, $\sum_{i=1}^{k} c_i \ell_i^d - \sum_{i=1}^{k} c_i' a_i^d = 0$. By Lemma 3.7, this representation is *not* minimal. Therefore, it must be that, without loss of generality, $\ell_1 \sim a_1$. Thus, $\alpha_1 \ell_1^d + \left(\sum_{i=2}^{k} c_i \ell_i^d - \sum_{i=2}^{k} c_i' a_i^d \right) = 0$ for some $\alpha_1 \in \mathbb{F}$. Inductively, we can deduce that for every i, $a_i \sim \ell_i$ as we wanted. □

Lemma 3.9 motivates the following definition.

Definition 3.10. *Let $g \in \Sigma^k \wedge \Sigma$ be a degree d polynomial where $d \geq 2k+1$. Define $L(g)$ to be the number of linear functions in the minimal representation of g as a $\Sigma^k \wedge \Sigma$ circuit.* ♢

(Indeed, Lemma 3.9 implies that $L(g)$ is well defined).

We now present an algorithm for reconstructing $\Sigma^k \wedge \Sigma$ circuits.

Algorithm 1 : Reconstruction of $\Sigma^k \wedge \Sigma$ circuits

Input: Black box access to a degree d, n-variate polynomial f computed by a $\Sigma^k \wedge \Sigma$ circuit.

```
if $d \leq 2k+1$ then
    Return the output of Algorithm 4.1 of [BSV21] on $f$.
Set $\mathsf{max}_k = 0$, $g = 0$, $\tilde{\alpha} = 0$ and pick distinct $\alpha_1, \ldots, \alpha_{kn+1} \in \mathbb{F}$
for $\alpha \in \{\alpha_1, \ldots \alpha_{kn+1}\}$ do
    Using Lemma 2.5, run Algorithm 4.1 of [BSV21] on $\frac{\partial^{d-(2k+1)} f}{\partial(\mathbf{p}_\alpha)^{d-(2k+1)}}$ and let $g_\alpha$ denote the output. Compute the minimal representation of $g_\alpha$.
    if $L(g_\alpha) > \mathsf{max}_k$ then
        Set $g = g_\alpha$, $\mathsf{max}_k = L(g_\alpha)$, $\tilde{\alpha} = \alpha$.
Denote $g = \sum_{i=1}^{\mathsf{max}_k} c_i' \ell_i^{2k+1}$. Return $f = \sum_{i=1}^{\mathsf{max}_k} c_i \ell_i^d$, where $c_i = \frac{c_i'}{d(d-1)\cdot\ldots\cdot(2k+2)\ell_i(\mathbf{p}_{\tilde{\alpha}})^{d-(2k+1)}}$.
```

We now state the main theorem of this section.

Theorem 3.11. *Suppose $|\mathbb{F}| > kn+1$ and $\mathrm{char}(\mathbb{F}) > d$ or $\mathrm{char}(\mathbb{F}) = 0$. Given a black box access to a polynomial f computed by a $\Sigma^k \wedge \Sigma$ circuit, Algorithm 1 reconstructs f and runs in time $\mathsf{poly}(n,d,c) \cdot \mathsf{poly}(k)^{k^{k^{10}}}$, where $c = \log q$ if $\mathbb{F} = \mathbb{F}_q$ is a finite field, and c equals the maximum bit complexity of any coefficient of f if $\mathbb{F}$ is infinite.*

We divide the proof of Theorem 3.11 into two lemmas.

Lemma 3.12. *The running time of Algorithm 1 is $\mathsf{poly}(n,d,c) \cdot \mathsf{poly}(k)^{k^{k^{10}}}$.*

Proof. If $d \leq 2k+1$ then by Theorem 3.2 the running time of Algorithm 1 is

$$\mathsf{poly}(n,c,(dk)^{k^{k^{10}}}) = \mathsf{poly}(n,c,k^{k^{k^{10}}}).$$

If $d > 2k+1$, then by Lemma 2.5 each access to a derivative of f, uses $O(d)$ queries to the black box computing f, along with some $\mathsf{poly}(n)$ time computation. Therefore, the running time of each call to Algorithm 4.1 of [BSV21] is $\mathsf{poly}(n,d,c) \cdot \mathsf{poly}(k)^{k^{k^{10}}}$. Given a representation of g_α calculating a minimal representation can be done in polynomial time. The loop in line 4 is being preformed $kn+1$ times, and the lest step can be performed in linear time, thus the entire run time of the algorithm is $\mathsf{poly}(n,d,c) \cdot \mathsf{poly}(k)^{k^{k^{10}}}$. □

Lemma 3.13. *The output of Algorithm 1 is a correct representation of f as a $\Sigma^k \wedge \Sigma$ circuit.*

Proof. If $d \leq 2k+1$ the correctness of our algorithm follows from the correctness of [BSV21].

If $d > 2k+1$, Lemma 3.9 promises us that there is a unique minimal representation of f, and the same is true for each directional derivative $\frac{\partial^{d-(2k+1)} f}{\partial \mathbf{p}_\alpha^{d-(2k+1)}}$ for every $\alpha \in \left\{\alpha_1, \ldots, \alpha_{n(k-1)}\right\}$ as picked in Algorithm 1. Assume $f = \sum_{i=1}^{k'} c_i \ell_i^d$ for some $k' \leq k$, $c_i \in \mathbb{F}$ and affine functions ℓ_i for $i \in [k']$.

Since $L(f)$ cannot increase when taking partial derivatives, it holds that

$$\max\left\{ L\left(\frac{\partial^{d-(2k+1)} f}{\partial \mathbf{p}_{\alpha_j}^{d-(2k+1)}}\right) : 1 \leq j \leq kn+1 \right\} \leq k'. \tag{3.14}$$

By Claim 3.6, there exists $j_0 \in [kn+1]$ such that $\ell_i(\mathbf{p}_{\alpha_{j_0}}) \neq 0$ for all $i \in [k']$. For such j_0, $L\left(\frac{\partial f}{\partial \mathbf{p}_{\alpha_{j_0}}^{d-(2k+1)}}\right) = k'$, so that (3.14) is in fact an equality. Thus, we exit the main loop with a correct computation

$$g_\alpha = \frac{\partial^{d-(2k+)1} f}{\partial \mathbf{p}_{\alpha_{j_0}}^{d-(2k+1)}} = \sum_{i=1}^{k'} (d(d-1) \cdot \ldots \cdot (2k+2)) c_i \ell_i(\mathbf{p}_{\alpha_{j_0}})^{d-(2k+1)} \ell_i^{2k+1},$$

where $\ell_i(\mathbf{p}_{\alpha_{j_0}}) \neq 0$ for every $i \in [k']$. Thus our algorithm outputs a (unique, up to reordering the linear functions) minimal representation of f. □

Proof of Theorem 3.11. Immediate from Lemma 3.12 and Lemma 3.13. □

Remark 3.15. *Our algorithm only uses randomization when running the algorithm from Theorem 4.1 of [BSV21] as a subroutine. As mentioned in Section 4 of [BSV21], the sources of randomness in their algorithm comes from solving a system of polynomial equations, and from Lemma 2.6.*

Over $\mathbb{R}$ or $\mathbb{C}$, solving a system of polynomial equations can be done deterministically. Further, since we only use their algorithm with $d = O(k)$, by combining Lemma 2.8 and Corollary 2.12, and using the fact that $\Sigma^k(\Sigma^k \wedge \Sigma) = \Sigma^{k^2} \wedge \Sigma$, we can derandomize the application of Lemma 2.6 as well, and obtain a deterministic algorithm. ◊

We stress again that the main advantage of Theorem 3.11 over Theorem 1.4 of [BSV21] is that the algorithm in Theorem 3.11 runs in polynomial time even for small, but super-constant values of k, e.g., $k = \left(\frac{\log\log n}{\log\log\log n} - O(1)\right)^{1/10}$.

4 Syntactic Rank of Depth-3 Circuits

In the following two sections, we define syntactic and semantic notions of ranks of polynomials computed by $\Sigma^k\Pi\Sigma$ circuits. Note that *syntactic* ranks are inherently tied to *circuits* computing the polynomials, whereas semantic ranks are independent of the representation or computation of the polynomials.

For a circuit C we denote by $[C]$ the polynomial computed by C. For two $\Sigma^k\Pi\Sigma$ circuits C, C', we define their *syntactic sum*, $C + C'$, to be the depth-3 circuit whose top gate sums all multiplication gates in C and C'. Observe that $C + C'$ is a $\Sigma^{2k}\Pi\Sigma$ circuit.

We start by defining *syntactic* notions of rank and distance for $\Sigma^k\Pi\Sigma$ circuits.

Definition 4.1 (Syntactic Rank and Distance)**.** *Let $C = \sum_{i=1}^{k} M_i = \sum_{i=1}^{k} \prod_{j=1}^{d_i} \ell_{i,j}$ be a $\Sigma^k\Pi\Sigma$ circuit. Define the following notions:*

1. $\deg(C) = \max\{\deg[M_i] : 1 \leq i \leq k\}$.
2. $\gcd(C)$ *is the set of linear functions appearing in all of $M_1, \ldots, M_k$ (up to multiplication by a constant). I.e.,* $\gcd(C) = \gcd(M_1, \ldots, M_k)$.
3. *$sim(C) := \frac{C}{\gcd(C)} = \sum_{i=1}^{k} \frac{M_i}{\gcd(C)} \in \Sigma^k\Pi\Sigma$ is called the* simplification *of C. C is called* simple *if* $\gcd(C) = 1$.
4. *We say that C is* minimal *if for every $\emptyset \neq S \subsetneq [k]$, $\sum_{i\in S} M_i \neq 0$.*
5. *Let $\mathcal{L}_i$ be the collection of linear polynomials appearing in $\frac{M_i}{\gcd(C)}$, we define $\Delta_{syn}(C) :=$* $\dim($*span* $\{\mathcal{L}_1, \ldots, \mathcal{L}_k\})$.
6. *Let C' be a $\Sigma^k\Pi\Sigma$ circuit. We define dist$(C, C') = \Delta_{syn}(C + C')$.* ◊

The usefulness of syntactic rank is expressed in the following well known *rank bound* for multilinear depth-3 circuits.

Theorem 4.2 ([DS07, KS09b, SS11, SS13])**.** *There's a monotone function $R(k, d)$ such that any simple and minimal $\Sigma^k\Pi\Sigma$ circuit C that computes the zero polynomial and such that* $\deg(C) \leq d$, *satisfies $\Delta_{syn}(C) \leq R(k, d)$. Further, $R(k, d) \leq 4k^2 \log(2d)$.*

If C is multilinear there's a similar function $R_M(k)$ depending only on k: any simple and minimal, multilinear $\Sigma^k\Pi\Sigma$ circuit C, computing the zero polynomial satisfies $\Delta_{syn}(C) \leq R_M(k)$. One can take $R_M(k) \leq 10k^2 \log k$.

The next lemma will be useful when studying different representations of the same polynomial.

Lemma 4.3. *For $j \in [t]$, let*

$$M_j = \sum_i M_{j,i}, \quad T_j = \sum_i T_{j,i}$$

be $\Sigma^k\Pi\Sigma$ circuits (with $M_{j,i}, T_{j,i}$ denoting multiplication gates).

Suppose that for every $j \in [t]$, $M_j - T_j$ are minimal circuits with $\Delta_{syn}(M_j - T_j) \leq s$. Further, assume that $\Delta_{syn}(\sum_j T_j) \leq r$. Then, $\Delta_{syn}(\sum_j M_j) \leq t(r + 2s)$.

Proof. By factoring out the gcd of the circuits we obtain

$$M_j - T_j = \left(\prod_i a_i^j\right)(M_j' + T_j').$$

where $\Delta_{\text{syn}}(M_j' + T_j') \leq s$. Further, by the assumption that $\Delta_{\text{syn}}(\sum_j T_j) \leq r$, we have

$$\sum_j T_j = \left(\prod_i \ell_i\right)(\sum_j \tilde{T}_j).$$

where $\Delta_{\text{syn}}(\sum_j \tilde{T}_j) \leq r$. Thus, for every $j \in [t]$,

$$T_j = \left(\prod a_i^j\right) T_j' = \left(\prod \ell_i\right) \tilde{T}_j$$

with $\Delta_{\text{syn}}(T_j') \leq s$ and $\Delta_{\text{syn}}(\tilde{T}_j) \leq r$. Multilinearity implies $\deg(T_j') \leq s$ and $\deg(\tilde{T}_j) \leq r$. Hence, $|\{\ell_i\} \setminus \left\{a_i^j\right\}| \leq s$. Thus, $\{\ell_i\} \cap \bigcap_{j=1}^t \left\{a_i^j\right\} = \left\{\ell_1', \ldots, \ell_q'\right\}$, where $q \geq |\{\ell_i\}| - ts$. Denote $\{\tilde{\ell}_1, \ldots, \tilde{\ell}_u\} = \{\ell_i\} \setminus \left\{\ell_1', \ldots, \ell_q'\right\}$, for $u = |\{\ell_i\}| - q \leq ts$. Hence,

$$\sum_{j=1}^t M_j = \left(\prod_i \ell_i'\right) \cdot \sum_{j=1}^t \left(\frac{\prod a_i^j}{\prod \ell_i'} \cdot \left(M_j'\right)\right).$$

To bound the syntactic rank of $\sum_{j=1}^t \left(\frac{\prod a_i^j}{\prod \ell_i'} \cdot \left(M_j'\right)\right)$ we observe that for every j, it holds that $\left\{a_i^j\right\} \setminus \{\ell_i'\} \subseteq \left(\left\{a_i^j\right\} \setminus \{\ell_i\}\right) \cup \{\tilde{\ell}_1, \ldots, \tilde{\ell}_u\}$. Further, since $\Delta_{\text{syn}}(M_j')$ is at most s, and $|\left\{a_i^j\right\} \setminus \{\ell_i\}| \leq r$, we get that

$$\Delta_{\text{syn}}\left(\sum_{j=1}^t M_j\right) = \Delta_{\text{syn}}\left(\sum_{j=1}^t \left(\frac{\prod a_i^j}{\prod \ell_i'} \cdot \left(M_j'\right)\right)\right) \leq tr + ts + u \leq t(r + 2s). \qquad \square$$

4.1 Syntactic Partitions of $\Sigma^k\Pi\Sigma$ Circuits

In this section we study syntactic partitions of $\Sigma^k\Pi\Sigma$ circuits. In Section 5.2 we shall discuss *semantic* partitions and compare the two notions.

Definition 4.4 (Syntactic Partition, Definition 3.3 of [KS09a]). *Let $C = \sum_{i=1}^{k} \prod_{j=1}^{d_i} \ell_{i,j} = \sum_{i=1}^{k} M_i$ be a $\Sigma^k\Pi\Sigma$ circuit. Let $I = \{A_1, \ldots, A_s\}$ be a partition of $[k]$. For each $i \in [s]$ let $C_i = \sum_{j \in A_i} M_j$. We say that $\{C_i\}_{i\in[s]}$ is a (τ, r)-syntactic partition of C if:*

- *For every $i \in [s]$, $\Delta_{syn}(C_i) \leq r$.*
- *For every $i \neq j \in [s]$, $dist(C_i, C_j) \geq \tau r$.* $\Diamond$

The following lemma captures an important property of the definition.

Lemma 4.5. *Let C be a $\Sigma^k\Pi\Sigma$ circuit. Let $(C_1, \ldots, C_s)$ be a (τ, r)-syntactic partition of C with $\tau \geq 10$. Let M, T be two multiplication gates in C that belong to different clusters. Then,*

$$dist(M, T) > \tau r/10.$$

Proof. Assume without loss of generality that M belongs to C_1 and T belongs to C_2, i.e., $C_1 = M + \tilde{M}$ and $C_2 = T + \tilde{T}$. By pulling out the linear factors from each cluster, we write:

$$C_1 = \left(\prod_{i=1}^{m_a} a_i\right)(M' + \tilde{M}'), \quad C_2 = \left(\prod_{i=1}^{m_b} b_i\right)(T' + \tilde{T}').$$

Further, assume towards contradiction that $\text{dist}(M, T) \leq \tau r/10$, and write

$$M + T = \left(\prod_{i=1}^{m_\ell} \ell_i\right)(\widehat{M} + \widehat{T}).$$

Then, we have

$$M = \left(\prod a_i\right) M' = \left(\prod \ell_i\right) \widehat{M}$$
$$T = \left(\prod b_i\right) T' = \left(\prod \ell_i\right) \widehat{T}$$

Since $\Delta_{\text{syn}}(C_1) \leq r$, $\deg(M') \leq r$ and similarly $\deg(T') \leq r$. Further, by the assumption that $\Delta_{\text{syn}}(M + T) \leq \tau r/10$, we get that $\deg(\widehat{M}) \leq \tau r/10$ and similarly $\deg(\widehat{T}) \leq \tau r/10$.

Recall that $m_a = |\{a_i\}|$, $m_b = |\{b_i\}|$ and $m_\ell = |\{\ell_i\}|$. The above inequalities imply that $|\{\ell_i\} \cap \{a_i\}| \geq m_a - \tau r/10$ and $|\{\ell_i\} \setminus \{a_i\}| \leq r$. Similar inequalities holds for $\{b_i\}$.

Denote $\{a_i\} \cap \{b_i\} = \left\{\ell'_1, \ldots, \ell'_q\right\}$. We have that

$$q \geq |\{a_i\} \cap \{b_i\} \cap \{\ell_i\}| \geq |\{a_i\} \cap \{\ell_i\}| - |\{\ell_i\} \setminus \{b_i\}| \geq m_a - \tau r/10 - r.$$

Similarly, $q \geq m_b - \tau r/10 - r$. Hence,

$$C_1 + C_2 = \left(\prod_i \ell'_i\right) \cdot \left(\frac{\prod a_i}{\prod \ell'_i} \cdot (M' + \tilde{M}') + \frac{\prod b_i}{\prod \ell'_i}(T' + \tilde{T}')\right).$$

The number of linearly independent linear forms in $\frac{\prod a_i}{\prod \ell'_i}$ is at most $\tau r/10 + r$, and similarly for $\frac{\prod b_i}{\prod \ell'_i}$. Further, since $M', \tilde{M}'$ are in the simplification of a cluster, their total rank is at most r, and similarly for $T', \tilde{T}'$. All of which goes to show that

$$\tau r \leq \Delta_{\text{syn}}(C_1, C_2) \leq 2\tau r/10 + 4r \leq 6\tau r/10,$$

where in the first inequality we used that assumption that C_1 and C_2 are a part of a (τ, r) partition, and in the second inequality we used the assumption that $\tau \geq 10$. This is a contradiction. □

We next study different syntactic partitions of a circuit C.

Claim 4.6 (Lower rank implies finer partition)**.** *Let $\tau \geq 10$. Let C be a minimal multilinear $\Sigma^k\Pi\Sigma$ circuit. Let $(C_1, \ldots, C_s)$ be a (τ, r_C)-syntactic partition of the multiplication gates in C. Let $(D_1, \ldots, D_{s'})$ be another (τ, r_D) partition of the gates of C.*

Assume $r_C \geq r_D$. Then, for every $i \in [s]$ there is a subset $S_i \subseteq [s']$ such that

$$C_i = \sum_{j \in S_i} D_j$$

and the subsets $S_1, \ldots, S_s$ form a partition of $[s']$.

Proof. The claim would follow if we prove that if D_i contains a multiplication gate from C_j then it contains no gate from $C_{j'}$ for $j \neq j'$. This will show that D_i is "contained" in C_j. Indeed, if this was the case then Lemma 4.5 would imply that

$$r_C \leq \tau r_C/10 < \Delta_{\text{syn}}(M_j + M_{j'}) \leq \Delta_{\text{syn}}(D_i) \leq r_D$$

in contradiction. □

Corollary 4.7. *Let $\tau \geq 10$. Let C be a minimal multilinear $\Sigma^k\Pi\Sigma$ circuit. Let $(C_1, \ldots, C_s)$ be a (τ, r_C)-syntactic partition of the multiplication gates in C, that has the largest number of clusters among all τ-syntactic partition of C. Then, for every other τ partition of C, $(D_1, \ldots, D_{s'})$, we have that $r_C \leq r_D$.*

Proof. If it was the other case then Claim 4.6 would give that there must be more clusters in $\{D_i\}$. □

Corollary 4.8 (Uniqueness of syntactic partitions with the same number of clusters)**.** *Let $\tau \geq 10$. Let C be a minimal multilinear $\Sigma^k\Pi\Sigma$ circuit. Let $(C_1, \ldots, C_s)$ and $(D_1, \ldots, D_s)$ be (τ, r_C) and (τ, r_D)-syntactic partitions of the multiplication gates in C, respectively. Then, there is a permutation π on $[s]$ such that for every $i \in [s]$, $C_i = D_{\pi(i)}$.*

Proof. Without loss of generality suppose that $r_C \geq r_D$. By Claim 4.6, $(D_1, \ldots, D_s)$ is a refinement of $(C_1, \ldots, C_s)$. However, they have the same number of clusters, so they must be the same partition. □

4.1.1 Algorithms for Computing Partitions

An algorithm for computing (τ, r)-syntactic partitions was provided by Karnin and Shpilka [KS09a].

Lemma 4.9 (Syntactic Clustering Algorithm; See Algorithm 1 and Lemma 5.1 of [KS09a])**.** *Let $n, k, r_{init}, \tau \in \mathbb{N}$. There exists an algorithm that given τ and an n-variate multilinear $\Sigma^k\Pi\Sigma$ circuit C as input, outputs $r \in \mathbb{N}$ such that*

$$R_M(2k) \leq r \leq k^{(k-2)\cdot\lceil \log_k(\tau)\rceil} \cdot R_M(2k) \leq (k\tau)^{k-2} \cdot R_M(2k)$$

and a (τ, r)-syntactic partition of $[k]$, in time $O(\log(\tau) \cdot n^3 k^4)$. Further, with an additional running time of $2^{O(k^2)} \cdot \mathsf{poly}(n)$, we can guarantee that this syntactic partition has the lowest value of r among all τ syntactic partitions of C.

We remark that the "further" part isn't explicitly stated in [KS09a]. However, it is easy to modify their algorithm in order to guarantee this property. For example, after running their algorithm one can run a brute force search over all partitions of $[k]$ and search for a τ-partition with a lower value of r. In the applications of Lemma 4.9, the additional running time incurred by this step is either irrelevant or anyway subsumed by larger factors of k originating from other elements in the proof.

4.2 Existence of a Unique Syntactic Partition

In this section we prove that for every multilinear polynomial $f \in \Sigma^k\Pi\Sigma$ there is a parameter τ, which is bounded by some function of k, such that any two τ partitions of any two $\Sigma^k\Pi\Sigma$ circuits computing f define, up to a permutation, the same clusters.

We start with the following claim that shows the existence of a (τ_1, r) partition with the special property that its rank is bounded as function of τ_0 that is much smaller than τ_1.

Claim 4.10. *For every function $\varphi : \mathbb{N} \to \mathbb{N}$, parameter τ_{min} and multilinear $f \in \Sigma^k\Pi\Sigma$ there is $\tau_{min} \leq \tau_0 \leq {\tau_{min}}^{\varphi(k)^k}$ and a (τ_1, r)-syntactic partition of f with:*

- $\tau_1 = \tau_0^{\varphi(k)}$.
- $r \leq R_M(2k) \cdot (k\tau_0)^{k-2}$.

Proof. Let C be any $\Sigma^k\Pi\Sigma$ circuit computing f. Using the algorithm promised in Lemma 4.9, find a $\kappa_0 = \tau_{\mathrm{min}}$-syntactic partition of C. Lemma 4.9 guarantees that this (κ_0, r_0) partition satisfies $r_0 \leq R_M(2k) \cdot (k\kappa_0)^{k-2}$. If this partition is also a (κ_1, r_0) partition for $\kappa_1 = \kappa_0^{\varphi(k)}$, then we are done by setting $\tau_0 = \kappa_0$ and $\tau_1 = \kappa_1$.

Otherwise, we continue in the same manner with κ_1 instead of κ_0 (i.e., we consider a syntactic partition with parameter κ_1) etc.

We claim that this process terminates after at most k iterations and finds the desired partition. It suffices to show that at every step the number of clusters in the partition decreases. At the i-th iteration of this algorithm we have a (κ_i, r_i) partition and similarly at the $(i+1)$-th step, a (κ_{i+1}, r_{i+1}) partition. Note that $r_{i+1} \geq r_i$, as otherwise, the $(i+1)$-th partition would also be a κ_i partition with a lower rank than the i-th partition, and would have been found by the algorithm in Lemma 4.9.

Both the i-th and the $(i+1)$-th partitions are $\tau_{\min}$ partitions. By Claim 4.6, the i-th partition is a refinement of the $(i+1)$-th partition, and in particular, since they are not the same partition (as otherwise the algorithm terminates), the number of clusters decreases. □

The next claim proves that for every multilinear polynomial $f \in \Sigma^k\Pi\Sigma$ there is $\tau = O(k^{k+2})^{k^{2k+1}}$ such that all τ partitions of f are equivalent. This result fixes the aforementioned error in [KS09a] and makes the argument in [BSV21] work.

Theorem 4.11. *For every multilinear polynomial $f \in \Sigma^k\Pi\Sigma$ there is $\tau = O(k^{k+2})^{k^{2k+1}}$ such that the following holds: Let C, D be any two $\Sigma^k\Pi\Sigma$ circuits computing f. Let $C = \sum_{i=1}^{s} C_i$ and $D = \sum_{i=1}^{s'} D_i$ be the τ-partitions of C and D, respectively, that Lemma 4.9 guarantees. Then $s = s'$ and there is a permutation $\pi : [s] \to [s]$ such that $[C_i] = [D_{\pi(i)}]$. Furthermore, for every i, $\Delta_{syn}(C_i)/k - 2R_M(2k) \leq \Delta_{syn}(D_{\pi(i)}) \leq k \cdot \Delta_{syn}(C_i) + 2kR_M(2k)$.*

Proof. For every τ we denote by

$$r(\tau) := R_M(2k) \cdot (k\tau)^{k-2}$$

the rank bound given in Lemma 4.9 for the rank of clusters in a τ partition.

Let C be any $\Sigma^k\Pi\Sigma$ circuit computing f. Apply Claim 4.10 on C with $\varphi(k) = k^2$ and $\tau_{\min} = 10R_M(2k)k^k$ and let τ_0, τ_1 be the parameters of the claimed partition. Denote this partition as $C = \sum_{i=1}^{s} C_i$. Set $\tau = \tau_0^k$. Calculating we see that

$$\tau = \tau_0^k \leq \left(\tau_{\min}^{\varphi(k)^k}\right)^k = \tau_{\min}^{k^{2k+1}} = (10R_M(2k)k^{k-2})^{k^{2k+1}} = O(k^{k+2})^{k^{2k+1}}.$$

Let D be any $\Sigma^k\Pi\Sigma$ computing f such that $D = \sum_{i=1}^{s'} D_i$ is a (τ, r_D) partition of D, as promised by Lemma 4.9.

Consider the circuit $C - D$ and a minimal subcircuit in it $E = \sum M_i - \sum T_j$ where the M_i's are multiplication gates in C and the T_j's are from D. We claim that there cannot be T_1 and T_2 from different D_i's. Assume for a contradiction that $T_1 \in D_1$ and that $T_2 \in D_2$. In this case Lemma 4.5 gives

$${\tau_{\min}}^k/10 \leq \tau_0^k/10 = \tau/10 \leq \tau r_D/10 < \Delta_{\text{syn}}(T_1 + T_2) \leq \Delta_{\text{syn}}(E) \leq R_M(2k)$$

in contradiction to the choice of $\tau_{\min}$. A similar argument would show that all the M_i's belong to the same cluster. Thus, every minimal circuit E contains gates from a single C_i and a single D_j. We partition each cluster C_i and each D_i according to the subsets of multiplication gates appearing in each minimal circuit. That is, we write $C_i = \sum_j C_{i,j}$ and $D_i = \sum_j D_{i,j}$ so that each such minimal circuit E is of the form $C_{i,j} - D_{i',j'}$.

As each such minimal E contain multiplication gates from a single D_i, we can represent every D_i as $D_i = \sum_j \sum_p C_{j,p}$. We wish to show that all $C_{j,p}$ come from a single C_j. Indeed, assume towards a contradiction that $C_{1,1} \in C_1$ and $C_{2,1} \in C_2$ (this is without loss of generality) and that $E_1 = D_{i,1} - C_{1,1}$ and $E_2 = D_{i,2} - C_{2,1}$ are minimal circuits computing the zero polynomial. As $\Delta_{\mathrm{syn}}(D_{i,1}+D_{i,2}) \le \Delta_{\mathrm{syn}}(D_i) \le r_D$ and $\Delta_{\mathrm{syn}}(E_1), \Delta_{\mathrm{syn}}(E_2) \le R_M(2k)$, we get from Lemma 4.3 (for $t = 2$) and Theorem 4.2 that

$$\begin{aligned}\tau_1/10 \le \tau_1 r_C/10 < \Delta_{\mathrm{syn}}(C_{1,1}+C_{2,1}) &\le 2\Delta_{\mathrm{syn}}(D_i) + 4R_M(2k)\\ &\le 2R_M(2k)\cdot(k\tau)^{k-2} + 4R_M(2k) \qquad (4.12)\end{aligned}$$

(where the lower bound follows by Lemma 4.5). This again contradicts the choice of $\tau_{\min}$ and the fact that $\tau_1 = \tau^k$.

This implies that for every $i \in [s]$, we can sum over the different minimal circuits in which the different "parts" of C_i appear, and get a subset $S_i \subseteq [s']$ such that $C_i = \sum_{j\in S_i} D_j$. This implies that $s' \ge s$, and observe that if $s = s'$ then this guarantees the existence of the claimed permutation π. So assume that $s' > s$ and that $|S_i| > 1$. Consider the $\Sigma^{2k}\Pi\Sigma$ circuit $C_i - \sum_{j\in S_i} D_j$. Using the same notation as before we write $\sum_j C_{i,j} = C_i = \sum_{j\in S_i}\sum_p D_{j,p}$, such that there is a one to one and onto map, that maps, for every j, a pair j', p' such that $E_j = C_{i,j} - D_{j',p'}$ is a minimal circuit computing the zero polynomial. Applying Lemma 4.3 again we get that

$$\Delta_{\mathrm{syn}}(\sum_{j\in S_i}\sum_p D_{j,p}) = \Delta_{\mathrm{syn}}(\sum_j C_{i,j}) \le tr_C + 2tR_M(2k) \le kr_C + 2kR_M(2k). \qquad (4.13)$$

On the other hand, our assumption that $|S_i| > 1$ implies that there are $j \ne j' \in S_i$. We now get that

$$\tau/10 \le \tau r_D/10 < \Delta_{\mathrm{syn}}(D_{j,1} + D_{j',1}) \le \Delta_{\mathrm{syn}}(\sum_{j\in S_i}\sum_p D_{j,p}). \qquad (4.14)$$

Observe that

$$\begin{aligned}kr_C + 2kR_M(2k) &\le kr(\tau_0) + 2kR_M(2k) = kR_M(2k)\cdot(k\tau_0)^{k-2} + 2kR_M(2k)\\ &\le R_M(2k)k^{k-1}\tau_0^{k-2} + 2kR_M(2k) < 3R_M(2k)k^k\tau_0^{k-2}\\ &< \tau_0^k/10 = \tau/10,\end{aligned}$$

where the last inequality follows from definitions of $\tau_{\min}$ and the fact that $\tau_{\min} \le \tau_0$. Combining the last calculation with (4.13) and (4.14) we get a contradiction. Hence $s = s'$ and there is a matching π such that $C_i = D_{\pi(i)}$.

The claim regarding the relation between $\Delta_{\mathrm{syn}}(C_i)$ and $\Delta_{\mathrm{syn}}(D_{\pi(i)})$ follows from the same argument as the one leading to (4.13). □

5 Semantic Rank of Depth-3 Circuits

In the following section, we define the semantic rank of polynomials computed by $\Sigma^k\Pi\Sigma$ circuits. Note that while *syntactic* ranks is inherently tied to a *circuit* C computing the polynomial, the *semantic* rank is independent of the representation or computation of the polynomial.

We say that a polynomial $g \in \mathbb{F}[x_1, \ldots, x_n]$ depends on r linear functions if there exist r linear functions $\ell_1, \ldots, \ell_r$ and a polynomial $h \in \mathbb{F}[y_1, \ldots, y_r]$ such that $g(\mathbf{x}) = h(\ell_1(\mathbf{x}), \ldots, \ell_r(\mathbf{x}))$.

Definition 5.1 (Semantic Rank)**.** *Let $f \in \mathbb{F}[x_1, \ldots, x_n]$ be a polynomial. Define $Lin(f)$ to be the product of the linear factors of f. Let $r \in \mathbb{N}$ be the minimal integer such that $f / Lin(f)$ is a polynomial of exactly r linear functions. We define $\Delta_{sem}(f) = r$.* $\Diamond$

Recall that the number of linear functions that a polynomial depends on equals the rank of its *partial derivative matrix*.

Definition 5.2. *Let $f \in \mathbb{F}[x_1, \ldots, x_n]$ be a polynomial. Define M_f to be a matrix whose i-th row contains the coefficients of $\partial f / \partial x_i$.* $\Diamond$

Note that if f depends on exactly r linear functions and $\mathrm{char}(\mathbb{F}) = 0$ or $\mathrm{char}(\mathbb{F}) > \deg(f)$, then $\mathrm{rank}(M_f) = r$.

Remark 5.3. *Note that under the definition above, it may be the case that f is non-zero and yet $\Delta_{sem}(f)$ equals 0. This happens when f is a product of linear functions. In what follows we will often implicitly assume that $\Delta_{sem}(f) \geq 1$. This doesn't affect our results but somewhat simplifies the presentation. One may also arbitrarily define the semantic rank of f to be 1 when f is a non-zero product of linear functions.* $\Diamond$

5.1 Semantic vs. Syntactic Rank

We now prove several claims that relate the syntactic and semantic notions of rank for polynomials computed by multilinear $\Sigma^k\Pi\Sigma$ circuits. We start by observing that the semantic rank is at most the syntactic rank.

Observation 5.4. *Suppose f is a polynomial in multilinear $\Sigma^k\Pi\Sigma$. Then, every multilinear $\Sigma^k\Pi\Sigma$ circuit C computing f satisfies $\Delta_{syn}(C) \geq \Delta_{sem}(f)$.*

We will also need the following definition from [KS09a].

Definition 5.5 ($\Sigma\Pi\Sigma(k, d, \rho)$ circuits, [KS09a])**.** *A generalized depth-3 circuit with parameters (k, d, ρ) is a depth-3 circuit with top fan-in k such that every multiplication gate can also multiply a polynomial that depends on at most ρ linear functions. That is, a polynomial $f(\mathbf{x})$ that is computed by a $\Sigma\Pi\Sigma(k, d, \rho)$ circuit has the form*

$$f(\mathbf{x}) = \sum_{i=1}^{k} \left(\prod_{j=1}^{d_i} \ell_{i,j}(\mathbf{x}) \right) \cdot h_i(\ell'_{i,1}(\mathbf{x}), \ldots, \ell'_{i,\rho_i}(\mathbf{x}))$$

where $d_i \leq d$ and $\rho_i \leq \rho$ for all $i \in [k]$. $\Diamond$

The measure Δ_{syn} is extended to $\Sigma\Pi\Sigma(k, d, \rho)$ circuits in Definition 2.1 of [KS08] (where it is simply called "rank").

The following observation follows immediately from the definitions.

Observation 5.6. *Suppose f is a polynomial computed by a multilinear $\Sigma^k\Pi\Sigma$ circuit C. Let $r = \Delta_{sem}(f)$. Then, there is a multilinear $\Sigma\Pi\Sigma(1, d, r)$ circuit C_f computing f such that $\Delta_{syn}(C_f) = \Delta_{sem}(f)$.*

An analog of Observation 5.4 also holds for $\Sigma\Pi\Sigma(k, d, \rho)$ circuits.

Lemma 5.7 (Lemma 2.20 in [KS09a])**.** *Suppose f is a polynomial computed by a multilinear $\Sigma\Pi\Sigma(k, d, \rho)$ circuit C. Then, every multilinear $\Sigma\Pi\Sigma(k, d, \rho)$ circuit C' computing f satisfies $\Delta_{syn}(C') \geq \Delta_{sem}(f)$.*

We now want to upper bound the syntactic rank as a function of the semantic rank (naturally, this only makes sense for *minimal* circuits, as other circuits can have artificially large syntactic rank). Our argument is essentially identical to Lemma 2.20 of [KS09a]. However since our notation is different and since we observe that we can slightly improve the bound in the multilinear case (the bound in [KS09a] depends on the degree of the polynomial being computed), we repeat their short argument.

We start with the rank bound for $\Sigma\Pi\Sigma(k,d,\rho)$ circuit proved in [KS08].

Lemma 5.8 (Lemma 4.2 of [KS08])**.** *Let C be a simple and minimal $\Sigma\Pi\Sigma(k,d,\rho)$ circuit computing the zero polynomial. Suppose*

$$C = \sum_{i=1}^{k} \left(\prod_{j=1}^{d_i} \ell_{i,j} \right) \cdot h_i(\tilde{\ell}_{i_1}, \ldots, \tilde{\ell}_{i,\rho_i})$$

and let $\tilde{R} = \sum_{i=1}^{k} \rho_i$. Then $\Delta_{syn}(C) \leq R(k,d) + \tilde{R}$.

Here $R(k,d) = 4k^2 \log(2d)$ is the rank bound for (not necessarily multilinear) $\Sigma^k\Pi\Sigma$ circuits (recall Theorem 4.2). Note that trivially $\tilde{R} \leq k\rho$, but in Lemma 5.10 we shall use the stricter upper bound stated in the lemma.

The proof of Lemma 5.8 in [KS08] is also not very complicated. Given a $\Sigma\Pi\Sigma(k,d,\rho)$ circuit C as in the statement of the lemma, one fixes randomly the linear functions $\tilde{\ell}_{i_1}, \ldots, \tilde{\ell}_{i,\rho_i}$, for $i \in [k]$, to obtain a simple and minimal $\Sigma^k\Pi\Sigma$ circuit of degree at most d, and applies the rank bound for such circuits. Note that fixing those linear functions might make the circuit non-multilinear even if the original circuit was multilinear, which means we have to use the rank bound $R(k,d)$ for non-multilinear $\Sigma^k\Pi\Sigma$ circuits. This incurs a dependence on the degree d. However, it is also convenient to have a form of Lemma 5.8 with no dependence on d. This is possible since C is multilinear. A similar observation was made by Dvir and Shpilka [DS07] for $\Sigma^k\Pi\Sigma$ circuits. Since C is multilinear, all linear functions appearing in each multiplication gate are variable disjoint, and hence linearly independent, which implies that $\Delta_{\mathrm{syn}}(C) \geq d$. Together with the upper bound in Lemma 5.8, this implies the following corollary.

Corollary 5.9 (Rank bound for multilinear $\Sigma\Pi\Sigma(k,d,\rho)$ circuits with no dependence on d)**.** *Let C be a simple and minimal $\Sigma\Pi\Sigma(k,d,\rho)$ circuit computing the zero polynomial. Then,*

$$\Delta_{syn}(C) \leq 40 \cdot (k^2 \log k + k^2\rho).$$

Proof. Since C is multilinear and by Lemma 5.8, we have that

$$d \leq \Delta_{\mathrm{syn}}(C) \leq R(k,d) + k\rho \leq 4k^2 \log(2d) + k\rho.$$

We claim that inequality holds only if $d \leq 40 \cdot (k^2 \log k + k^2\rho)$. Indeed, if $d \leq 40k^2\rho$ then the bound clearly holds. Otherwise we get

$$\begin{aligned} d &\leq 4k^2 \log(2d) + k\rho < 4k^2 \log(2d) + d/40k \\ d &\leq \big((160k^3)/(40k-1)\big) \log(2d) \end{aligned}$$

and a simple calculation shows that this does not hold if $d > 40 \cdot k^2 \log k$. This implies the claimed upper bound on $\Delta_{\mathrm{syn}}(C)$. □

The following lemma uses the notation of Theorem 4.2.

Lemma 5.10 (Small semantic-rank implies small syntactic-rank, similar to Lemma 2.20 in [KS09a])**.** *Let C be a minimal multilinear $\Sigma^k\Pi\Sigma$ circuit computing a polynomial f. Suppose that $\Delta_{sem}(f) \leq r$. Then $\Delta_{syn}(C) \leq r + R(k+1, \Delta_{syn}(C))$. In particular, $\Delta_{syn}(C) \leq 2^7 rk^2 \log k$.*

Proof. Denote $C = \sum_{i=1}^{k} M_i$ where each M_i is a multiplication gate, and denote $r_C = \Delta_{\mathrm{syn}}(C)$.

Let C_f be a $\Sigma\Pi\Sigma(1, d, r)$ computing f, so that $C_f = (\prod_i \ell_i) \cdot h(\tilde{\ell}_1, \ldots, \tilde{\ell}_r)$ where h has no linear factors.

Consider the circuit $C - C_f$, which computes the zero polynomial. We factor out the gcd of this circuit, which is the common linear factor of $M_1, \ldots, M_k$ and $(\prod_i \ell_i)$. Note, however, that if a linear function divides all of $M_1, \ldots, M_k$, then it divides f (since C computes f), and thus it is one of the ℓ_i's. Hence, the gcd of the circuit $C - C_f$ equals $\gcd(M_1, \ldots, M_k)$.

Consequently, we can write $C - C_f$ as

$$C - C_f = \gcd(M_1, \ldots, M_k) \cdot \left(\sum_{i=1}^{k} \tilde{M}_i - \left(\prod_{i \in A} \ell_i \right) \cdot h(\tilde{\ell}_1, \ldots, \tilde{\ell}_r) \right),$$

where A is some subset of the (indices of the) original linear functions appearing in C_f. Let $D = \left(\sum_{i=1}^{k} \tilde{M}_i - (\prod_{i \in A} \ell_i) \cdot h(\tilde{\ell}_1, \ldots, \tilde{\ell}_r) \right)$. Since $C - C_f$ computes the zero polynomial, D is also a circuit computing the zero polynomial

By construction, D is simple, and by minimality of C it is also minimal: indeed, no subcircuit of C (nor C itself) computes the zero polynomial, and no subset $S \subsetneq [k]$ of the $\tilde{M}_i$'s can equal $(\prod_{i \in A} \ell_i) \cdot h$, as this would imply that $[\sum_{i \in S} M_i] = f$, which would again contradict the minimality of C.

Finally, note that the degree of D is at most r_C. Indeed, by multilinearity, each $\tilde{M}_i$ is a product of variable-disjoint (and hence linearly independent) linear functions, which means that their number can be at most r_C. As D computes the zero polynomial, the degree of $(\prod_{i \in A} \ell_i) \cdot h(\tilde{\ell}_1, \ldots, \tilde{\ell}_r)$ is also at most r_C.

Thus, we have concluded that D is a simple and minimal $\Sigma\Pi\Sigma(k+1, r_C, r)$ circuit computing the zero polynomial. By Lemma 5.8,

$$\Delta_{\mathrm{syn}}(D) \leq R(k+1, r_C) + r.$$

Finally note that r_C, which is the dimension of the span of the linear functions in the $\tilde{M}_i$'s, is at most $\Delta_{\mathrm{syn}}(D)$. Thus, we get the inequality

$$r_C \leq \Delta_{\mathrm{syn}}(D) \leq R(k+1, r_C) + r \leq 4(k+1)^2 \log(2r_C) + r.$$

For the "in particular" part, as before, we consider two cases. If $r_C \leq ((k+1)^2 + 1)r$, then the bound clearly holds. Otherwise, $r_C \leq 4((k+1)^2 + 1) \log(2r_C)$ and a simple calculation shows that this implies $r_C \leq 2^7 k^2 \log k$. The statement follows from a combination of the two cases. □

5.2 Semantic Partitions of $\Sigma^k\Pi\Sigma$ Circuits

We next define semantic partitions of $\Sigma^k\Pi\Sigma$ circuits, that correspond to semantic rank in the same manner that syntactic partitions correspond to syntactic rank (recall Definition 4.4).

Definition 5.11 (Semantic Partition)**.** *Let f be a multilinear polynomial. We say that $(f_1, \ldots, f_s)$ is a (τ, r)* semantic *partition of f if $f = \sum_{i=1}^{s} f_i$, and*

- *For every $i \in [s]$, $\Delta_{sem}(f_i) \le r$.*
- *For every $i \neq j \in [s]$, $\Delta_{sem}(f_i + f_j) \ge \tau r$.*

We further say that the partition is realizable *if there exists a $\Sigma^k\Pi\Sigma$ circuit C computing f and a partition of its multiplication gates $(C_1, \ldots, C_s)$ such that $[C_i] = f_i$. From now on, we only consider realizable partitions.* ♢

We also often use the term "τ-partition" (either syntactic or semantic) where it is implied that the partition is a (τ, r)-partition for some value of r.

Corollary 5.12. *Let C be a minimal multilinear $\Sigma^k\Pi\Sigma$ circuit. Every (τ, r)-semantic partition of $[C]$ is also a (τ', r')-syntactic partition of C with $r' = 2^7 k^2 \log k \cdot r$ and $\tau' = \tau/(2^7 k^2 \log k)$.*

Proof. Let $[C] = \sum_i [C_i]$ be the assumed semantic partition (recall that we always assume that the partition is realizable). From Lemma 5.10 we get $\Delta_{\mathrm{syn}}(C_i) \le 2^7 \Delta_{\mathrm{sem}}([C_i]) k^2 \log k \le 2^7 r k^2 \log k = r'$. Furthermore, Observation 5.4 implies that $\Delta_{\mathrm{syn}}(C_i' + C_j') \ge \Delta_{\mathrm{sem}}([C_i'] + [C_j']) \ge \tau r \ge \tau' \cdot r'$, as claimed. □

Corollary 5.13. *Let C be a minimal multilinear $\Sigma^k\Pi\Sigma$ circuit. Let $(C_1, \ldots, C_s)$ be a (τ, r)-syntactic partition of C. Then $([C_1], \ldots, [C_s])$ is a (τ', r)-semantic partition of $[C]$ with $\tau' = \tau/(2^7 k^2 \log k)$.*

Proof. By Observation 5.4, for every $i \in [s]$ we have that $\Delta_{\mathrm{sem}}([C_i]) \le \Delta_{\mathrm{syn}}(C_i) \le r$. Furthermore, Lemma 5.10 implies that for every $i \neq j$,

$$\Delta_{\mathrm{sem}}([C_i] + [C_j]) \ge \Delta_{\mathrm{syn}}(C_i + C_j)/(2^7 k^2 \log k) \ge \tau r/(2^7 k^2 \log k) = \tau' r. \qquad \square$$

5.2.1 Uniqueness Properties of Semantic Partitions

We next state and prove an analogous claim to Claim 4.6.

Claim 5.14 (Lower rank implies finer partition)**.** *Let $\tau > 40k^2 \log k + k^2$. Let C, D be two minimal multilinear $\Sigma^k\Pi\Sigma$ circuits computing the same polynomial f. Let $C = \sum_{i=1}^{s} C_i$ be a partition of the multiplication gates in C and similarly $D = \sum_{i=1}^{s'} D_i$ a partition of the gates in D. Let $f_i = [C_i]$ and suppose that $(f_1, \ldots, f_s)$ is a (τ, r_1)-semantic partition of f. Similarly, let $g_i = [D_i]$ and suppose that $(g_1, \ldots, g_{s'})$ is a (τ, r_2)-semantic partition of f. Assume $r_1 \ge r_2$.*

Then, for every $i \in [s]$ there is a subset $S_i \subseteq [s']$ such that

$$f_i = \sum_{j \in S_i} g_j$$

and the subsets $S_1, \ldots, S_s$ form a partition of $[s']$.

Proof. The proof is similar to the proof of Claim 4.6. Consider the following $\Sigma\Pi\Sigma(2k, d, r_1)$ circuit E computing the zero polynomial:

$$\sum_{i=1}^{s} C_{f_i} - \sum_{i=1}^{s'} C_{g_i}$$

(where C_{f_i} is a $\Sigma\Pi\Sigma(1, \deg(f_i), r_1)$ gate computing f_i, and similarly C_{g_i} computes g_i). Consider a minimal subcircuit E' of E,

$$\sum_{i \in I} C_{f_i} - \sum_{j \in J} C_{g_j}$$

(note that $I, J \neq \emptyset$ since both C, D are minimal circuits).

Assume towards contradiction that there exist $i, i' \in I$ with $i \neq i'$. Then

$$\Delta_{\text{sem}}(f_i + f_{i'}) \leq \Delta_{\text{syn}}(C_i + C_{i'}) \leq \Delta_{\text{syn}}(E') \leq 40(k^2 \log k + k^2 r_1).$$

The first inequality follows from Lemma 5.7. The second inequality is immediate from the definition of syntactic rank, and the last inequality is the rank bound of Corollary 5.9.

From the assumption, $\Delta_{\text{sem}}(f_i + f_{i'}) \geq \tau r_1$, which contradicts the choice of τ. □

Corollary 5.15. *Let C and D be as in Claim 5.14. If $s = s'$ then there is a permutation π of $[s]$ such that $f_i = g_{\pi(i)}$.*

Proof. Indeed, in the notation of Claim 5.14, we will get that the sets S_i must be of size 1, as otherwise we would have $s' > s$, in contradiction to the assumption. As $|S_i| = 1$ it follows that $f_i = g_j$ where $S_i = \{j\}$. □

The next lemma considers two different semantic partitions. In the case of syntactic partitions, Corollary 4.8 argues about partitions of the same circuit, in contrast, in the case of semantic partitions, we can compare partitions of different circuits computing f. This is one of the advantages of semantic rank over syntactic rank.

Lemma 5.16 (Relation between different partitions)**.** *Let $\tau > R_M(2k) + 2^8 k^2 \log k$. Let C, D be two minimal multilinear $\Sigma^k\Pi\Sigma$ circuits computing the same polynomial f. Let $C = \sum_{i=1}^{s} C_i$ be a partition of the multiplication gates in C and similarly $D = \sum_{i=1}^{s'} D_i$ a partition of the gates in D.*

Let $f_i = [C_i]$ and suppose that $(f_1, \ldots, f_s)$ is a (τ, r_1)-semantic partition of f. Similarly, let $g_i = [D_i]$ and suppose that $(g_1, \ldots, g_{s'})$ is a (τ, r_2)-semantic partition of f. Assume $r_1 \geq r_2$.

Then, there's a refinement of $(C_1, \ldots, C_s)$, denoted $(C'_1, \ldots, C'_{s'})$, such that up to reordering, $[C'_i] = g_i$.

Proof. By Claim 5.14, for every $i \in [s]$ there's a subset $S_i \subseteq [s']$ such that

$$f_i = \sum_{j \in S_i} g_j \tag{5.17}$$

and the subsets $S_1, \dots, S_s$ form a partition of $[s']$. Equation (5.17) implies $[C_i] = [\sum_{j \in S_i} g_j]$. Thus,

$$C_i - \sum_{j \in S_i} D_j$$

is a $\Sigma^{2k}\Pi\Sigma$ circuit E computing the zero polynomial.

Let E' be a minimal subcircuit of E. Write

$$E' = \sum_t M_t - \sum_j \sum_p T_{j,p}$$

where $M_t \in C_i$ and $T_{j,p} \in D_j$. In particular,

$$\Delta_{\mathrm{syn}}(\sum_j \sum_p T_{j,p}) \leq \Delta_{\mathrm{syn}}(E') \leq R_M(2k).$$

Suppose $j_1 \neq j_2 \in [s']$ appear in the sum $\sum_j \sum_p T_{j,p}$. Let $T_{j_1,p_1} \in D_{j_1}$ and $T_{j_2,p_2} \in D_{j_2}$. As

$$\mathrm{dist}(T_{j_1,p_1}, T_{j_2,p_2}) \leq R_M(2k),$$

we get from Lemma 2.17 in [KS09a] (the triangle inequality for syntactic rank) and Lemma 5.10 that

$$\begin{aligned} \mathrm{dist}(D_{j_1}, D_{j_2}) &\leq \mathrm{dist}(D_{j_1}, T_{j_1,p_1}) + \mathrm{dist}(T_{j_1,p_1}, D_{j_2}) \\ &\leq \mathrm{dist}(D_{j_1}, T_{j_1,p_1}) + \mathrm{dist}(T_{j_1,p_1}, T_{j_2,p_2}) + \mathrm{dist}(T_{j_2,p_2}, D_{j_2}) \\ &\leq \Delta_{\mathrm{syn}}(D_{j_1}) + R_M(2k) + \Delta_{\mathrm{syn}}(D_{j_2}) \\ &\leq 2^8 k^2 \log k \cdot r_2 + R_M(2k) \ . \end{aligned}$$

Hence, $\tau r_2 \leq \Delta_{\mathrm{sem}}(g_{j_1} + g_{j_2}) \leq \mathrm{dist}(D_{j_1}, D_{j_2}) \leq 2^8 k^2 \log k r_2 + R_M(2k)$ in contradiction. Thus, the sum $\sum_j \sum_p T_{j,p}$ contains multiplication gates from a single D_j. This implies the claim. □

Corollary 5.18. *Let $\tau > R_M(2k) + 2^8 k^2 \log k$. If $f = \sum f_i$ is a (τ, r_1)-semantic partition that has the largest number of clusters among all τ-semantic partitions of f, then for every other (τ, r_2)-semantic partition $f = \sum g_i$ we have that $r_1 \leq r_2$.*

Proof. If it was the other case then Lemma 5.16 would give that there must be more clusters in $\{g_i\}$. □

Corollary 5.19 (Uniqueness of maximal partition regardless of representation)**.** *Let $\tau > R_M(2k) + 2^8 k^2 \log k$. Let C, D be two minimal multilinear $\Sigma^k\Pi\Sigma$ circuits computing the same polynomial f. Let $(C_1, \dots, C_s)$ be a τ-semantic partition of the multiplication gates in C of minimal semantic rank. Similarly let $(D_1, \dots, D_{s'})$ be a τ-semantic partition of D of minimal semantic rank. Then $s = s'$ and there is a permutation π such that for every $i \in [s]$, $[C_i] = [D_{\pi(i)}]$.*

Proof. The claim follows immediately from Corollary 5.18 and Lemma 5.16. □

5.2.2 An Algorithm for Computing Partitions

In this section we provide an algorithm for constructing (τ, r) partitions (either syntactic or semantic).

In Lemma 4.9 we saw an algorithm, by Karnin and Shpilka [KS09a], for computing (τ, r)-syntactic partitions. We now provide an algorithm for constructing (τ, r)-semantic partitions. Our algorithm is quite straightforward, it simply checks all possible partitions and picks the best one. The guarantee on the output of the algorithm follow from Lemma 4.9 and the relations between syntactic and semantic rank.

We first note that computing the semantic rank of a polynomial f in $\Sigma^k\Pi\Sigma$ can be done in randomized polynomial time given black box access to f.

Lemma 5.20. *There exists a randomized polynomial time algorithm that, given black box access to a polynomial $f \in \Sigma^k\Pi\Sigma$ computes $\Delta_{sem}(f)$.*

Proof. We start by using the Kaltofen-Trager black box factorization algorithm [KT90] in order to factor out the linear factor of f. Write $f = P \cdot g$ where P is the product of the linear factors of f. Then $\Delta_{\mathrm{sem}}(f) = \mathrm{rank}(M_g)$ (recall Definition 5.2). Thus it remains to compute the rank of the matrix M_g, which is the dimension of the set of first order partial derivatives of g. This can be done in randomized polynomial time as in (for example) Lemma 4.1 of [Kay11]. □

We now describe the semantic clustering algorithm.

Algorithm 2 : Semantic clustering algorithm

```
Input: White box access to a $\Sigma^k\Pi\Sigma$ circuit, $C$, and a parameter $\tau$
$r = \infty$
for every partition $(C_1, \ldots, C_s)$ of the $k$ multiplication gates of $C$ do
    Compute the semantic ranks $r_i$ of $[C_i]$ and the distances $d_{i,j} = \Delta_{\mathrm{sem}}([C_i], [C_j])$
    Set $r' = \max_{i \in [s]} r_i$
    if for all $i, j$, $d_{i,j} \ge \tau r'$ and $r' < r$ then
        Set $r := r'$ and save the partition
Return $r$ and the saved partition
```

Recall that Lemma 4.9 implies that the Karnin-Shpilka *syntactic* clustering algorithm returns *syntactic* clusters. This allows us to obtain some guarantees on the output of Algorithm 2.

Claim 5.21. *For every τ, Algorithm 2 runs in time at most $2^{k^2} \cdot \mathrm{poly}(n)$ and outputs a (τ, r) partition where*

$$r \le R_M(2k) \cdot k^{\lceil \log_k(\tau \cdot 2^7 k^2 \log k) \rceil \cdot (k-2)} \le R_M(2k) \cdot 2^{7k} k^{4k} \tau^{k-2}.$$

Proof. By Lemma 4.9 with $\kappa = \tau \cdot 2^7 k^2 \log k$, there exist a (κ, r) *syntactic* partition $(C_1, \ldots, C_s)$ where $r \le R_M(2k) \cdot k^{\lceil \log_k(\kappa) \rceil \cdot (k-2)} \le (k\kappa)^{k-2} \cdot R_M(2k)$. Corollary 5.13 implies that it is also a (τ, r) *semantic*-partition. Thus, the algorithm will find at least one τ-semantic partition and it clearly outputs the one with the minimal rank. The statement on the running time follows from Lemma 5.20. $\square$

Remark 5.22. *Note that Corollary 5.19 shows that the semantic partition with the minimal semantic rank is unique regardless of the representation. Hence the output of Algorithm 2 does not depend on the circuit C but only on the polynomial f it computes.* $\diamondsuit$

5.2.3 Semantic Partitions under Restrictions

Corollary 5.19 proves that any maximal semantic partition is unique. However, in our reconstruction algorithm we shall consider restrictions of the unknown polynomial to subsets of the variables. Hence, we will need a stronger property that is analogous to the one proved in Theorem 4.11. We start with the obvious fact that restrictions can't increase the semantic rank (recall Definition 2.1).

Claim 5.23. *If $\Delta_{sem}(f|_{B,\mathbf{a}}) \geq t$ then $\Delta_{sem}(f) \geq t$.*

The next claim shows that for every multilinear polynomial, $f \in \Sigma^k\Pi\Sigma$, there exists a (τ_1, r)-semantic partition with the special property that its rank bound, r, is upper bounded as a function of τ_0, which is much smaller than τ_1. The claim and its prove are completely analogous to Claim 4.10 and so we omit the proof.

Claim 5.24. *For every function $\varphi : \mathbb{N} \to \mathbb{N}$, every $\tau_{min} \in \mathbb{N}$ and for every multilinear $f \in \Sigma^k\Pi\Sigma$ there is $\tau_{min} \leq \tau_0 \leq \tau(k) = R_M(2k)^{\varphi(k)^k}$ such that there is a (τ_1, r)-semantic partition of f with:*

- $\tau_1 = \tau_0^{\varphi(k)}$.
- $r \leq R_M(2k)2^{7k}k^{4k}\tau_0^{k-2}$.

Claim 5.24 implies a useful inclusion property of clusters in any partition of a restriction of f.

Claim 5.25. *Let $f \in \Sigma^k\Pi\Sigma$ be a multilinear polynomial. Let $f = \sum_{i=1}^{s} f_i$ be the semantic partition guaranteed in Claim 5.24 for $\varphi(k) \geq 1$ and $\tau_{min} = R_M(2k) \cdot 2^{7k+12} \cdot k^{4k+3}$, with parameters (τ_0, τ_1) and rank r. Set $\tau = \tau_0^k$ as in Theorem 4.11.*

Let $B \subseteq [n]$ and $\mathbf{a} \in \mathbb{F}^n$. Let $f|_{B,\mathbf{a}} = \sum_{i=1}^{s'} g_i$ be a maximal τ-semantic partition of $f|_{B,\mathbf{a}}$, with rank r_B and let D be a minimal circuit computing $f|_{B,\alpha}$ with $(D_1, \ldots, D_{s'})$ its corresponding partition.

Then, $s' \leq s$ and for every $i \in [s']$ there's a subset $S_i \subseteq [s]$ such that $D_i = \sum_{j \in S_i} (C_j)|_{B,\mathbf{a}}$.

Proof. Let $C = \sum_{i=1}^{s} C_i$ be a $\Sigma^k\Pi\Sigma$ realization of the partition $f = \sum_{i=1}^{s} f_i$, where $[C_i] = f_i$. Remark 5.22 and Claim 5.24 imply that $r \leq R_M(2k)2^{7k}k^{4k}\tau_0^{k-2}$.

As $C|_{B,\mathbf{a}}$ computes $f|_{B,\mathbf{a}}$, Corollary 5.18 and Lemma 5.16 imply that there is a partition $C|_{B,\mathbf{a}} = \sum_{i=1}^{s'} D_i$ such that $[D_i] = g_i$. Corollary 5.12 implies that this is also a $(\tau_{\text{syn}}, r_{B,\text{syn}})$-syntactic partition for $\tau_{\text{syn}} := \tau/(2^7k^2 \log k)$ and $r_{B,\text{syn}} := r_B \cdot 2^7k^2 \log k$.

Let M_1, M_2 be two multiplication gates appearing in the same cluster C_1. Assume towards contradiction that their restrictions to $(B, \mathbf{a})$ appear in different clusters D_1, D_2 (without loss of generality). Lemma 4.5 and Lemma 5.10 imply that

$$\begin{aligned}\tau/10 &\le \tau r_B/10 = \tau_{\mathrm{syn}} r_{B,\mathrm{syn}}/10 \le \Delta_{\mathrm{syn}}((M_1)|_{B,\mathbf{a}} + (M_2)|_{B,\mathbf{a}}) \le \Delta_{\mathrm{syn}}((C_1)|_{B,\mathbf{a}}) \\ &\le \Delta_{\mathrm{syn}}(C_1) \le 2^7 k^2 \log k \Delta_{\mathrm{sem}}(C_1) \le 2^7 k^2 \log k \cdot r \\ &\le 2^7 k^2 \log k \cdot R_M(2k) \cdot 2^{7k} k^{4k} \tau_0^{k-2} \le R_M(2k) \cdot 2^{7k+7} k^{4k+3} \tau_0^{k-2}.\end{aligned}$$

This contradicts the choice of $\tau_{\min}$ and τ. □

6 Learning Low Degree Polynomials

As in [BSV21], we start by providing an algorithm, which is efficient when the degree d is very small.

6.1 Reconstruction Algorithm for Low Degree Multilinear $\Sigma^k\Pi\Sigma$ circuits

Bhargava, Saraf and Volkovich proved the following lemma.

Lemma 6.1 (Lemma 6.4 in [BSV21]). *Let $f \in \mathbb{F}[x_1, \ldots, x_n]$ be a polynomial computed by a degree d, multilinear $\Sigma^k\Pi\Sigma$ circuit C_f of the form*

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

Then, there is a randomized algorithm that given k, d and black-box access to f outputs a multilinear $\Sigma^k\Pi\Sigma$ circuit computing f, in time $\mathsf{poly}(c) \cdot (dkn)^{O(d^2k^3)^{d^2k^2}}$, where $c = \log q$ if $\mathbb{F} = \mathbb{F}_q$ is a finite field, and c equals the maximum bit complexity of any coefficient of f if $\mathbb{F}$ is infinite.

As before, we'd like make the dependence on n polynomial, regardless of d and k. To explain the required changes, we start by sketching the proof of Lemma 6.1.

Proof sketch of Lemma 6.1. The proof follows the following steps.

1. Variable reduction: Denote by m the number of essential variables of f. As there are at most kd linear functions appearing in the circuit, it holds that $m \le kd$.

   By Lemma 2.6, 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 $f(A\mathbf{x})$ depends only on the first m variables. Apply this algorithm to obtain A, and denote $g := f(A\mathbf{x})$.

2. Learn g. As g has $m \le kd$ variables and degree at most d, we can interpolate in order to find a representation $g(x) = \sum_{\mathbf{e}:|\mathbf{e}|\le d} c_{\mathbf{e}} \cdot \mathbf{x}^{\mathbf{e}}$. This is done in time $\mathsf{poly}(\binom{m+d}{d}) \le \mathsf{poly}(\binom{kd+d}{d})$.

3. Add the requirement that g has a $\Sigma^k\Pi\Sigma$ representation. To do so, consider the representation

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

   We view this as a set of at most $\binom{kd+d}{d}$ polynomial equations in $kd(m+1) \leq 2k^2d^2$ variables, $\{a_{j,t}^{(i)} \mid i \in [k], j \in [d], t \in [m+1]\}$.

4. Make sure that $f = g(A^{-1}x)$ is a multilinear polynomial ("lifting"). Recall that we have A, thus we can compute A^{-1}. Let $L_t = \langle R_t, \mathbf{x}\rangle$ where R_t is the t-th row of A^{-1}. L_t is a linear function in $\mathbf{x}$, i.e., $L_t = \sum_{p\in[n]} \alpha_p^t x_p$ for some coefficients α_p^t. For every $p \in [n], j \in [k]$, make sure that the degree of x_p in the j-th product gate is at most one. We can do this by adding a set of polynomial equations that guarantee that the product of coefficients of x_p in any two linear forms appearing in the j-th product gate is 0. This adds $k \cdot n \cdot d^2$ equations in the variables $\{a_{j,t}^{(i)} \mid i \in [k], j \in [d], t \in [m+1]\}$.

5. Solve the polynomial equations. In total we have $k \cdot n \cdot d^2 + \binom{kd+d}{d}$ many equations in $2k^2d^2$ variables. As mentioned in Section 2.4, this is solvable in time

$$\mathsf{poly}\left(n, \mathrm{Sys}(2d^2k^2, kd^2n + \binom{dk+d}{d}, d)\right) \leq (dkn)^{O(d^2k^3)^{2d^2k^2}}. \qquad \square$$

The following lemma improves the time complexity of Lemma 6.1.

Lemma 6.2. *Let $f \in \mathbb{F}[x_1, \ldots, x_n]$ be a polynomial computed by a degree d, multilinear $\Sigma^k\Pi\Sigma$ circuit C of the form*

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

Then, there is a randomized algorithm that given k, d and black-box access to f outputs a multilinear $\Sigma^k\Pi\Sigma$ circuit computing f, in time $\mathsf{poly}\left(n, c, (dk)^{O(d^3k^2)^{2d^2k^2}}\right)$*, where $c = \log q$ if $\mathbb{F} = \mathbb{F}_q$ is a finite field and c is the maximal bit complexity of the coefficients of f is $\mathbb{F}$ is infinite.*

As we only change one step in the algorithm of [BSV21], we will describe the change and its effect on the time complexity.

Proof. We only change the behavior of Step 4 in Lemma 6.1. In this step we make sure that the "lifting" of the polynomial g is indeed a multilinear polynomial. This is done by adding a set of $k \cdot d^2 \cdot n$ polynomial equations that make sure that the degree of each variable in each product gate is at most one. To ease notation let us assume that we want to enforce that the individual degree of x_1 is 1. For $t \in [m]$, denote the coefficient of x_1 in L_t by α_1^t (recall that L_t is the linear function corresponding to the t-th row of A^{-1}). Note that the algorithm knows $\alpha_1^1, \ldots, \alpha_1^m$. Therefore, when substituting $A^{-1}\mathbf{x}$

in the circuit, the coefficient of x_1 in the j-th linear function of the i-th multiplication gate, $\ell_{i,j}$, is $\sum_t a^{(i)}_{j,t}\alpha_1^t$. As we need x_1 to appear in at most one of the linear functions $\ell_{i,1},\ldots,\ell_{i,d_i}$, it is enough to require that for every $j_1 \neq j_2 \in [d_i]$,

$$(\sum_{t=1}^{m} a^{(i)}_{j_1,t}\alpha_1^t)\cdot(\sum_{t=1}^{m} a^{(i)}_{j_2,t}\alpha_1^t) = 0.$$

This is a quadratic equation in the coefficients $\left\{a^{(i)}_{j,t}\right\}$. As the set of quadratic polynomials in $mkd = k^2d^2$ variables has dimension at most k^4d^4, we can find a basis to the set of equations (in time $\mathsf{poly}(n,k,d)$) and add only the equations in the basis to the set of our polynomial constrains, thus adding at most k^4d^4 polynomial equations.

Observe that any solution to the new system will have the property that the lift will be multilinear. Moreover, this system is solvable: as noted by [BSV21], this follows since the "natural" circuit C_g for g, which is obtained from the original multilinear circuit for f by applying $A\mathbf{x}$ to the inputs, has the property that if we replace (in C_g) the input $\mathbf{x}$ by $A^{-1}\mathbf{x}$, then we get a multilinear depth-3 circuit for f.

Hence, the system is solvable in time $\mathrm{Sys}(mdk, k^4d^4 + \binom{dk+d}{d}, d)$, and the overall time complexity of the algorithm is bounded by $\mathsf{poly}\Big(n,c,\mathrm{Sys}(d^2k^2,k^4d^4+\binom{dk+d}{d},d)\Big) \le \mathsf{poly}\left(n,c,(dk)^{O(d^3k^2)^{d^2k^2}}\right)$. □

Similarly to Lemma 6.5 in [BSV21], we can use our improved running time in order to learn circuits with low *syntactic* rank.

Lemma 6.3 (Similar to Lemma 6.5 in [BSV21])**.** *Let $f \in \mathbb{F}[x_1,\ldots,x_n]$ be a polynomial computed by multilinear $\Sigma^k\Pi\Sigma$ circuit C with $\Delta_{syn}(C) \le r$. Then, there is a randomized algorithm that given k,r and black-box access to f, outputs a multilinear $\Sigma^{k'}\Pi\Sigma$ circuit computing f, where $k' \le k$ is the smallest possible fan-in, in time* $\mathsf{poly}(n,c,(rk)^{O(r^3k^2)^{r^2k^2}})$.

This lemma also enables us to learn circuits computing polynomials of low *semantic* rank.

Lemma 6.4. *Let $f \in \mathbb{F}[x_1,\ldots,x_n]$ be a polynomial computed by multilinear $\Sigma^k\Pi\Sigma$ circuit C with $\Delta_{sem}(f) \le r$. Then there is a randomized algorithm that given k,r and black-box access to f outputs a multilinear $\Sigma^{k'}\Pi\Sigma$ circuit computing f, where $k' \le k$ is the smallest possible fan-in, in time* $\mathsf{poly}(n,c,(rk)^{(rk)^{\mathsf{poly}(r,k)}})$.

Proof. By Lemma 5.10, there is a circuit C computing f with $\Delta_{\mathrm{syn}}(C) \le 2^7k^2\log k \cdot r$. We now apply Lemma 6.3. Note that since the algorithm is black box we don't need to actually know C. □

6.2 Reconstruction of Low-Degree Depth-3 Set-Multilinear Circuits

Since set-multilinear depth-3 circuits are a special case of depth-3 multilinear circuits, it's clear that given a black box access to a set-multilinear polynomials $f(\mathbf{x}_1,\ldots,\mathbf{x}_d)$

computed by a depth-3 set-multilinear circuit of top fan-in k, the algorithm in Section 6.1 is able to reconstruct f. However, as stated it's possible that the algorithm outputs a multilinear circuit rather than a set-multilinear circuit, and thus doesn't give a proper learning algorithm. In this section we explain how to modify the proof of Lemma 6.1 to output a set-multilinear circuit.

Let $\mathbf{x} = \mathbf{x}_1 \bigcup \mathbf{x}_2 \cdots \mathbf{x}_d$ denote the set of variables and let $n_i = |\mathbf{x}_i|$.

We first observe that in item 1 we can find a matrix A that "respects" the partition $\mathbf{x}_1, \ldots, \mathbf{x}_d$: that is, A is a direct sum of matrices $A_1, \ldots, A_d$ such that A_i operates on $\mathbf{x}_i$ and $f(A\mathbf{x})$ depends on at most kd variables. As noted in Lemma 2.8, finding, e.g., A_1 amounts to finding a basis to the vector space

$$\left\{ \mathbf{a} \in \mathbb{F}^{n_1} : \sum_{i=1}^{n_1} \frac{f}{\partial x_{1,i}}(\mathbf{x}) = 0 \right\}.$$

By Corollary 2.10 we can find such A_1 deterministically in polynomial time (for small, super-constant k), so that now the polynomial depends on at most $m_1 \leq k$ variables from $\mathbf{x}_1$. We then find $A_2, A_3, \ldots, A_d$.

The next step, item 2 is done as in Lemma 6.1.

We change item 3 to require that f has a depth-3 set-multilinear representation of top fan-in k. This is simply done by changing the system of polynomial equations to

$$\sum_{i=1}^{k} \prod_{j=1}^{d} (a_{j,1}^{(i)} x_{j,1} + \cdots + a_{j,n_j}^{(i)} x_{j,n_j} + a_{j,m_j+1}) = g = \sum_{\mathbf{e}} c_{\mathbf{e}} \cdot \mathbf{x}^{\mathbf{e}}.$$

For $m = \max_j m_j \leq k$, this is a set of at most $\binom{kd+d}{d}$ polynomial equations in $kd(m+1) \leq 2k^2d^2$ variables. Solving the system can be done in time $\mathrm{Sys}(d^2k^2, \binom{dk+d}{d}, d)$.

Note that item 4 and item 5 are now redundant. By the structure of A, we can simply apply A^{-1} (which replaces every variable $x_{i,j} \in \mathbf{x}_i$ by a linear function in $\mathbf{x}_i$). This is because the set multilinear circuits only multiplies linear functions in distinct sets in the partition $\mathbf{x}_1, \ldots, \mathbf{x}_d$, so that $f(A^{-1}\mathbf{x})$ is set multilinear.

As a result of this discussion we obtain the corollary.

Corollary 6.5. *Let $f(\mathbf{x}) \in \mathbb{F}[\mathbf{x}]$ be a set-multilinear polynomial computed by a degree d, set-multilinear depth-3 circuit. Suppose $\mathbf{x} = \mathbf{x}_1 \cup \cdots \cup \mathbf{x}_d$ and $|\mathbf{x}_i| \leq n$ for all i. Then, there is a randomized algorithm that given n, k, d and black-box access to f outputs a set-multilinear depth-3 circuit with top fan-in k that computes f, in time* $\mathsf{poly}\left(n, c, (dk)^{O(d^2k^3)^{d^2k^2}}\right)$.

The following is analogous to Lemma 6.3.

Lemma 6.6. *Let $f \in \mathbb{F}[x_1, \ldots, x_n]$ be a polynomial computed by set-multilinear $\Sigma^k\Pi\Sigma$ circuit C with $\Delta_{sem}(C) \leq r$. Then, there is a randomized algorithm that given k, r and black-box access to f outputs a set-multilinear $\Sigma^k\Pi\Sigma$ circuit computing f, where $k' \leq k$ is the smallest possible fan-in, in time* $\mathsf{poly}\left(n, c, (rk)^{O(r^3k^2)^{r^2k^2}}\right)$.

7 Efficient Construction of Cluster Preserving Sets

In order to reconstruct general multilinear $\Sigma^k\Pi\Sigma$ circuits, we would again like to follow the steps of [BSV21]. However, as some of our definitions are different, and we replace some brute force steps with algorithmically efficient steps, we're required to make substantial changes in the algorithm. In particular we replace their use of the notion of "rank preserving subspaces" with an explicit construction of a subset B of the variables that, in some sense, preserves the structure of semantic clusters of f.

The following algorithm attempts to construct a set B together with a vector $\mathbf{a}$ such that the clusters of $f|_{B,\mathbf{a}}$ (with respect to a certain semantic τ-partition), found by the algorithm of Lemma 5.20, are in one-to-one correspondence with the clusters that the same algorithm would have outputted on f. Our algorithm receives τ as a parameter (τ will be related to the parameters from Claim 5.24).

Algorithm 3 : Deterministic construction of a cluster-preserving set

```
Input: Black box access to a degree $d$, $n$-variate polynomial $f$, computed by a $\Sigma^k\Pi\Sigma$ circuit, $C$, and a parameter $\tau$
1: Let $S \subseteq \mathbb{F}$ an arbitrary set of size $n^{k^{k^{O(k)}}}$
2: Set $B = \emptyset$, $s = 0$, $f_{curr} = \sum_{j=1}^{s} C_j = 0$
3: Pick at random $\mathbf{a} \in S^n$.
4: for every $I \subseteq \{x_1, \ldots, x_n\} \setminus B$ of size at most 4 do
5:     for $k' \in [k]$ do
6:         Using Lemma 6.4 with $k = k'$, learn $f_I = f|_{B\cup I,\mathbf{a}}$. Using randomized polynomial identity testing, make sure that indeed $f_I \equiv f|_{B\cup I,\mathbf{a}}$, and if so, go to the next line
7:     Run Algorithm 2 on $f_I$ with parameter $\tau$ to get $f_I = \sum_{j=1}^{s_I} C'_j$
8:     if $s_I \neq s$ then
9:         Set $B = B \cup I$, $s = s_I$, $f_{curr} = f_I$, and save $(C'_1, \ldots, C'_{s_I})$. Restart the main loop in line 4
10:    for $j \in [s]$ do
11:        Find $\sigma(j) \in [s]$ s.t. $C'_j|_{x_I=\mathbf{a}_I} = C_{\sigma(i)}$ using a randomized PIT algorithm
12:        Calculate $r'_j = \Delta_{\mathrm{sem}}(C'_j), r_j = \Delta_{\mathrm{sem}}(C_{\sigma(j)})$
13:        if $r'_j > r_j$ then
14:            Set $B = B \cup I$, $s = s_I$, $f_{curr} = f_I$, and save $(C'_1, \ldots, C'_{s_I})$. Restart the main loop in line 4
15:            Abort in case any subprocedure failed during the execution of the algorithm
```

We now explain what guarantees we get on the outputs $(B, \mathbf{a})$ of Algorithm 3. We first bound the running time of the algorithm.

Claim 7.1. *Algorithm 3, when given the parameter τ guaranteed in Claim 7.2 as input, runs in time* $\mathsf{poly}(n) \cdot k^{k^{k^{\mathsf{poly}(k)}}}$ *and returns a set B of size at most $k^{k^{O(k)}}$.*

The following important claim shows that if $(B, \mathbf{a})$ is the output of Algorithm 3, then for the "correct" choice of τ, $f|_{B,\mathbf{a}}$ preserves the clusters (with respect to a τ-semantic partition) of f.

Claim 7.2. *Let $f \in \Sigma^k\Pi\Sigma$ be a multilinear polynomial and let C be a minimal multilinear $\Sigma^k\Pi\Sigma$ computing f. There exists a non-zero polynomial Γ_C of degree at most $n^{k^{k^{O(k)}}}$ such that if $B, \mathbf{a}$ are the outputs of Algorithm 3 on f, and $\Gamma_C(\mathbf{a}) \neq 0$, then the following holds: Consider the semantic partition of f, $f = \sum_{i=1}^{s} f_i$, given by Claim 5.24 with $\varphi(k) = k^2$ and $\tau_{min} = R_M(2k) \cdot 2^{7k+20} \cdot k^{4k+4}$. Let τ_0, τ_1, r be its parameters as promised by the claim and let $\tau = \tau_0^k$. Let D be a minimal multilinear $\Sigma^k\Pi\Sigma$ circuit computing $f|_{B,\mathbf{a}}$. Then, the output of Algorithm 2 on D with parameter τ, denoted by $[D] = \sum_{i=1}^{s'} g_i$, satisfies:*

1. *$s' = s$.*
2. *There is a permutation π on $[s]$ such that $g_{\pi(i)} = (f_i)|_{B,\mathbf{a}}$.*
3. *$\Delta_{sem}(g_{\pi(i)}) = \Delta_{sem}(f_i)$.*

In particular, the g_i's also form a (τ, r) partition.

The rest of this section is devoted to proving Claim 7.1 and Claim 7.2.

Remark 7.3. *The fact that $\deg(\Gamma_C) = n^{k^{k^{O(k)}}}$ is the bottleneck in derandomizing our algorithms. For derandomization we shall have to find $\mathbf{a}$ such that $\Gamma_C(\mathbf{a}) \neq 0$, and it is not clear how to achieve this in time $\mathsf{poly}(n, F(k))$ (i.e. not have k in the exponent of n).* ♢

7.1 Proof of Claim 7.1

Let f be the multilinear polynomial in question. Let C be a multilinear $\Sigma^k\Pi\Sigma$ circuit computing f. Consider a partition of f, $f = \sum_{i=1}^{s} f_i$, as given by Claim 5.24 with $\varphi(k) = k^2$. Let $f = \sum_{i=1}^{s} f_i$ the $\tau = \tau_0^k$-semantic partition of minimal rank and let $C = \sum_{i=1}^{s} C_i$ the corresponding partition of the multiplication gates in C.

Claim 7.4. *Let $B \subseteq [n]$ and $\mathbf{a} \in \mathbb{F}^n$. Consider a τ-semantic partition of $g = f|_{B,\mathbf{a}}$, $g = \sum_{i=1}^{s'} g_i$. Observe that by Claim 5.25, there is I such that $g_1 = \sum_{i \in I} (f_i)|_{B,\mathbf{a}}$. Then, for every $J \subseteq I$ it holds that*

$$\Delta_{sem}(\sum_{j \in J} (f_j)|_{B,\mathbf{a}}) \leq 2^7 k^2 \log k \cdot \Delta_{sem}(g_1).$$

Proof. Let $D = C|_{B,\mathbf{a}}$ be the restriction of C to $(B, \mathbf{a})$, which is a multilinear $\Sigma^k\Pi\Sigma$ circuit computing g. By Remark 5.22, there is a subcircuit D_1 such that $[D_1] = g_1$. Note that

$$\begin{aligned}\Delta_{\text{sem}}\left(\sum_{j \in J} (f_j)|_{B,\mathbf{a}}\right) &\leq \Delta_{\text{syn}}\left(\sum_{j \in J} (C_j)|_{B,\mathbf{a}}\right) \leq \Delta_{\text{syn}}\left(\sum_{i \in I} (C_i)|_{B,\mathbf{a}}\right) \\ &= \Delta_{\text{syn}}(D_1) \leq 2^7 k^2 \log k \cdot \Delta_{\text{sem}}(g_1).\end{aligned}$$

where the first inequality follows from Observation 5.4, the second from monotonicity of the syntactic rank, and the third inequality follows from Lemma 5.10. □

Denote by B_i the set obtained in the i-th iteration of the main loop in Algorithm 3. Associate with B_i a partition $\pi(B_i)$ of $[k]$, $[k] = S_1 \sqcup S_2 \ldots \sqcup S_{s_i}$ such that if we denote $g_{i,j} = \sum_{e \in S_j} f_e|_{B_i,\mathbf{a}}$, then $g_i = \sum_{j=1}^{s_i} g_{i,j}$ is a τ-semantic partition of $g_i = f|_{B_i,\mathbf{a}}$. We denote by r_i the rank of this partition.

The following claim provides a "progress measure" that bounds of running time of Algorithm 3.

Claim 7.5. *Let $t < t'$ such that $\pi(B_t) = \pi(B_{t'})$. Then, there exists $i \in [s_t]$ and $J \subseteq S_i$ such that*

$$\Delta_{sem}\left(\sum_{j \in J}(f_j)|_{B_t,\mathbf{a}}\right) < \Delta_{sem}\left(\sum_{j \in J}(f_j)|_{B_{t'},\mathbf{a}}\right).$$

Proof. Observe that since $\pi(B_t) = \pi(B_{t'})$ then for every $i \in [s_t] = [s_{t'}]$ it holds that $g_{t,i} = g_{t',i}|_{B_t,\mathbf{a}}$. For the rest of the proof we shall denote $g_i = g_{t',i}$ and $s' = s_t = s_{t'}$.

We split the proof into three cases according to the relation between the rank parameters of the partitions obtained in various stages of the algorithm.

We first observe that if $(g_1)|_{B_{t+1},\mathbf{a}}, \ldots, (g_{s'})|_{B_{t+1},\mathbf{a}}$ is not a (τ, r_t) partition then the claim holds. Indeed, since for every $i \neq j$

$$\Delta_{\text{sem}}((g_i)|_{B_{t+1},\mathbf{a}}, (g_j)|_{B_{t+1},\mathbf{a}}) \geq \Delta_{\text{sem}}((g_i)|_{B_t,\mathbf{a}}, (g_j)|_{B_t,\mathbf{a}}) \geq \tau \cdot r_t,$$

for it not to be a (τ, r_t) partition it must be the case that there's some $i \in [s']$ such that $\Delta_{\text{sem}}((g_i)|_{B_{t+1},\mathbf{a}}) > r_t$ and thus

$$\Delta_{\text{sem}}((g_i)|_{B_{t'},\mathbf{a}}) \geq \Delta_{\text{sem}}((g_i)|_{B_{t+1},\mathbf{a}}) > r_t, \tag{7.6}$$

which implies the claim. Hence, we assume from now on that $(g_1)|_{B_{t+1},\mathbf{a}}, \ldots, (g_{s'})|_{B_{t+1},\mathbf{a}}$ is a (τ, r_t) partition and that $r_{t+1} \leq r_t$.

If $r_{t+1} = r_t$, then Corollary 5.19 implies that the polynomials $(g_i)|_{B_{t+1},\mathbf{a}}$ are the unique partition. As the number of clusters did not change and since we added variables to B_t, the description of the algorithm implies that the rank of one of the clusters increased, which proves the claim.

Finally, suppose $r_{t+1} < r_t$. Consider the output of Algorithm 2 at the $(t+1)$-th step. This corresponds to a partition $\pi(B_{t+1})$ of the clusters of f. As we showed that $(g_1)|_{B_{t+1},\mathbf{a}}, \ldots, (g_{s'})|_{B_{t+1},\mathbf{a}}$ are a (τ, r_t) partition, the assumption that $r_{t+1} < r_t$ and Lemma 5.16 imply that $\pi(B_{t+1})$ is a refinement of $\pi(B_t) = \pi(B_{t'})$.

To ease notation let us denote with $h_1, \ldots, h_{s_{t+1}}$ the polynomials corresponding to the clusters of the partition B_{t+1}. I.e. $h_i = \sum_{j \in R_i} f_j|_{B_{t+1},\mathbf{a}}$, where $R_1, \ldots, R_{s_{t+1}}$ are the sets in the partition $\pi(B_{t+1})$. In the t'-th step, the polynomials $\{h_i|_{B_{t'},\mathbf{a}} := \sum_{j \in R_i} f_j|_{B_{t'},\mathbf{a}}\}$ no longer form a (τ, r_{t+1}) partition (here we use the fact that $\pi(B_{t'}) = \pi(B_t) \neq \pi(B_{t+1})$), as by Claim 5.14 a finer partition with the same τ implies lower rank, which means that the output of Algorithm 2 at the t'-th step would not have been $g_1, \ldots, g_{s'}$. Similarly to before, this implies that for some $i \in [s_{t+1}]$,

$$\Delta_{\text{sem}}(h_i|_{B_{t'}}) > \Delta_{\text{sem}}(h_i) \geq \Delta_{\text{sem}}(h_i|_{B_t}).$$

As $\pi(B_{t+1})$ is a refinement of $\pi(B_t)$, there is some j such that $R_i \subseteq S_j$, and the claim holds. This concludes the proof. □

Proof of Claim 7.1. Suppose that the algorithm makes T iterations of additions of variables to B (each addition adds at most 4 variables). Recall that at each such iteration the algorithm holds a partition π of the clusters of f. As there are at most 2^{k^2} possible partitions of the clusters (since there are most k clusters), there exists a partition π that is obtained at least $T/2^{k^2}$ times.

By Claim 7.5 and by another application of the pigeonhole principle, there exist a cluster i and a set $J \subseteq S_i$, such that for at least $T/(2^{k^2} \cdot 2^k)$ values of t, $\Delta_{\mathrm{sem}}(\sum_{j \in J}(f_j)|_{B_t,\mathbf{a}})$ increases. In particular after that many steps,

$$\Delta_{\mathrm{sem}}(\sum_{j \in J}(f_j)|_{B_t,\mathbf{a}}) > \frac{T}{2^{k^2+k}}.$$

On the other hand, Claim 7.4 and Claim 5.24 promise that

$$\Delta_{\mathrm{sem}}(\sum_{j \in J}(f_j)|_{B_t,\mathbf{a}}) \leq 2^7 k^2 \log k \cdot R_M(2k) \cdot 2^{7k} \cdot k^{4k} \tau^{k-2} \leq R_M(2k) \cdot 2^{7k+7} k^{4k+3} \tau^{k-2}.$$

Hence,

$$|B| = 4T \leq 4 \cdot 2^{k^2+k} \cdot \Delta_{\mathrm{sem}}(\sum_{j \in J}(f_j)|_{B_t,\mathbf{a}}) < R_M(2k) \cdot 2^{(k+4)^2} k^{4k+3} \tau^{k-2}. \tag{7.7}$$

To complete the proof we recall that by Claim 5.24,

$$\tau = \tau_0^k \leq \left(R_M(2k)^{(k^2)^k}\right)^k = k^{k^{O(k)}}.$$

Together with (7.7), this implies $|B| \leq k^{k^{O(k)}}$.

We also note that by Claim 5.21 at each step

$$r_i \leq R_M(2k) \cdot 2^{7k} k^{4k} \tau^{k-2} \leq k^{k^{O(k)}}.$$

Plugging in the upper bounds on r and $|B|$ to Lemma 6.4 shows that every iteration of Step 6 of Algorithm 3 takes at most

$$\mathsf{poly}\left(|B|, (rk)^{(rk)^{\mathsf{poly}(r,k)}}\right) = k^{k^{k^{O(k)}}}.$$

The other steps in the algorithm run in time polynomial in n and in smaller factors of k, and thus the claim follows. □

Remark 7.8. *We have bounded the size of B and the running time only for a certain "correct" choice of the parameter τ in Algorithm 3. In our reconstruction algorithm we run Algorithm 3 with many possible choices of τ, since we don't know a-priori which is the correct one. However, it is easy to modify the algorithm so that the upper bounds on the size of B and on the running time always hold, by simply terminating the algorithm if B gets too large or if it runs for too many steps.* ◊

7.2 Proof of Claim 7.2

The following lemma shows that for a "good" choice of $\mathbf{a}$, if the semantic rank of $f|_{B,\mathbf{a}}$ is smaller than the semantic rank of f, then one can add a small number of variables to B so that the semantic rank increases. Note that such a statement is fairly easy to prove for *syntactic* rank. Proving it for semantic rank, however, takes a considerable amount of work and is deferred to Section 7.2.1. In Section 7.3 we explain why we had to work with semantic rank rather than with syntactic rank.

Claim 7.9. *Let $f \in \mathbb{F}[x_1, \ldots, x_n]$ and let $B \subseteq \{x_1, \ldots, x_n\}$. There exists a non-zero polynomial Ψ_B of degree at most $4n^6$ with the following property: For every $\mathbf{a} \in \mathbb{F}^n$ such that $\Psi_B(\mathbf{a}) \neq 0$, if $\Delta_{sem}(f|_{B,\mathbf{a}}) < \Delta_{sem}(f)$ then there exist variables $x, y, z, w \in \overline{B}$ such that $\Delta_{sem}(f|_{B,\mathbf{a}}) < \Delta_{sem}(f|_{B\cup\{x,y,z,w\},\mathbf{a}})$.*

We stress that Claim 7.9 holds for *every* set B and the polynomial Ψ_B doesn't depend on $\mathbf{a}$. In Algorithm 3, the set B that the algorithm constructs *does* potentially depend on $\mathbf{a}$. However, we will later assert that since $\mathbf{a}$ is chosen randomly, it is (among other things) not a zero of Ψ_B for *any* set B.

We continue with the proof of Claim 7.2. It turns out that it is easier to argue about how *syntactic* rank behaves with respect to restrictions. We next state some lemmas showing that there are simple polynomial conditions such that if $\mathbf{a}$ is not a zero of any of them then fixing some variables according to $\mathbf{a}$ preserves the structure of the circuit.

We start with a well known lemma (see, e.g., Observation 2.1 in [KMSV13]) that constructs a polynomial that preserves pairwise linear independence between linear functions.

Lemma 7.10. *Let $L(\mathbf{x}) = \sum_{i=1}^{n} a_i x_i + a_0$ and $R(\mathbf{x}) = \sum_{i=1}^{n} b_i x_i + b_0$ be two linearly independent linear functions. Denote $S = \{i \mid a_i \neq 0 \text{ or } b_i \neq 0\}$. Let*

$$D(L, R) := \prod_{i\in S} \left(a_i R(\mathbf{x}) - b_i L(\mathbf{x})\right).$$

Assume that $\mathbf{a}$ satisfies $D(L(\mathbf{a}), R(\mathbf{a})) \neq 0$. Then, for every $I \subsetneq [n]$, such that $S \not\subseteq I$, it holds that $L|_{\mathbf{x}_I=\mathbf{a}_I}$ and $R|_{\mathbf{x}_I=\mathbf{a}_I}$ are linearly independent.

We next define another polynomial condition that allows us to claim that a certain fixing does not hurt the rank too much.

Claim 7.11. *Let C, be a multilinear $\Sigma^k\Pi\Sigma$ circuit. There exists a non-zero n-variate polynomial Φ_C of degree at most n^3k^3, such that for every $\mathbf{a} \in \mathbb{F}^n$, if $\Phi_C(\mathbf{b}) \neq 0$, then:*

1. *For every $I \subseteq [n]$ and every subcircuit C' of C, it holds that $\Delta_{syn}(C'|_{x_I=\mathbf{a}_I}) \geq \Delta_{syn}(C') - |I|$.*
2. *For every set $I \subseteq \{x_1, \ldots, x_n\}$ it holds that $(C_1)|_{\mathbf{x}_I=\mathbf{b}_I}, \ldots, (C_r)|_{\mathbf{x}_I=\mathbf{a}_I}$ is a $(\tau - |I|, r)$-syntactic partition of $C|_{\mathbf{x}_I=\mathbf{a}_I}$.*

Proof. Let $C = \sum_{i=1}^{k} T_i$, where $T_i = \prod_j \ell_{i,j}(\mathbf{x})$ are multilinear multiplication gates, and let $C' = \sum_{i\in S} T_i$ be a subcircuit of C, where $S \subseteq [k]$. Assume $\Delta_{\text{syn}}(C') = r$. As in the proof of Lemma 6.14 of [BSV21] we define the polynomial

$$\Phi_C(\mathbf{x}) = \prod_{i=1}^{k} T_i \cdot \prod_{(i,j)\neq(i',j')} D(\ell_{i,j}, \ell_{i',j'}).$$

Observe that $\deg(\Phi_C) \leq kn \cdot \binom{kn}{2} < n^3k^3$.

We first claim that if $\Phi_C(\mathbf{a}) \neq 0$, then for every $I \subseteq [n]$, $\gcd(C'|_{\mathbf{x}_I=\mathbf{a}_I}) \sim \gcd(C')|_{\mathbf{x}_I=\mathbf{a}_I}$. Indeed, it's clear that for every $\ell \in \gcd(C')$, $\ell|_{\mathbf{x}_I=\mathbf{a}_I}$ is also in $\gcd(C')|_{\mathbf{x}_I=\mathbf{a}_I}$. In the other direction, suppose that $\ell|_{\mathbf{x}_I=\mathbf{a}_I}$ is in $\gcd(C')|_{\mathbf{x}_I=\mathbf{a}_I}$ but ℓ isn't in $\gcd(C')$. This implies that there is a multiplication gate T in C' such that ℓ doesn't divide T. If ℓ is not supported on I, then Lemma 7.10 implies that $\ell|_{\mathbf{x}_I=\mathbf{a}_I}$ does *not* divide $T|_{\mathbf{x}_j=a_j}$, which contradicts the assumption that $\ell|_{\mathbf{x}_I=\mathbf{a}_I}$ is in $\gcd(C')|_{\mathbf{x}_I=\mathbf{a}_I}$. If ℓ is supported on I then, as $\prod_j T_j(\mathbf{a}) \neq 0$, $\ell_{\mathbf{x}_I=\mathbf{a}_I} \neq 0$. Thus, ℓ is restricted to a nonzero constant and does not affect the gcd.

Having shown that, let $\tilde{C}'$ be the simplification of C'. The argument above shows that the only linear functions that "disappear" from $\tilde{C}'$ when restricted to $\mathbf{x}_I = \mathbf{a}_I$, are those that are supported on the variables in I. Let $\ell_1, \ldots, \ell_r$ be linear functions that span the linear functions in $\tilde{C}'$. Thus, $(\ell_1)|_{\mathbf{x}_I=\mathbf{a}_I}, \ldots, (\ell_r)_{\mathbf{x}_I=\mathbf{a}_I}$ span the simplification of $C'|_{\mathbf{x}_I=\mathbf{a}_I}$. Finally, note that the dimension of $\{(\ell_1)|_{\mathbf{x}_I=\mathbf{a}_I}, \ldots, (\ell_r)_{\mathbf{x}_I=\mathbf{a}_I}\}$ is at least $r - |I|$, as it is the intersection of an r dimensional space with a subspace of codimension $|I|$.

To prove the second item, we note that for every $i \in [s]$, $\Delta_{\text{syn}}((C_i)_{\mathbf{x}_I=\mathbf{a}_I}) \leq \Delta_{\text{syn}}(C_i) \leq r$. Further, for every $i \neq i'$, the argument above applied to $C' = C_i + C_{i'}$, implies that

$$\begin{aligned}\operatorname{dist}(C_i|_{\mathbf{x}_I=\mathbf{a}_I}, C_{i'}|_{\mathbf{x}_I=\mathbf{a}_I}) &= \Delta_{\text{syn}}(C_i|_{\mathbf{x}_I=\mathbf{a}_I} + C_{i'}|_{\mathbf{x}_I=\mathbf{a}_I}) = \Delta_{\text{syn}}(C'|_{\mathbf{x}_I=\mathbf{a}_I}) \\ &\geq \Delta_{\text{syn}}(C') - |I| \geq \tau r - |I| \geq (\tau - |I|)r. \qquad \square\end{aligned}$$

Claim 7.12. *Let $C = \sum_{i=1}^{s} C_i$ be a multilinear $\Sigma^k\Pi\Sigma$ circuit, where each C_i is a sum of one or more multiplication gates. Let Φ_C be as in Claim 7.11 and $\mathbf{a} \in \mathbb{F}^n$ be such that $\Phi_C(\mathbf{a}) \neq 0$. Then, for every subset $I \subseteq \{x_1, \ldots, x_n\}$ of at most 4 variables, if $C|_{\mathbf{x}_I=\mathbf{a}_I} = \sum_{i=1}^{s} C_i|_{\mathbf{x}_I=\mathbf{a}_I}$ is a (τ, r)-syntactic partition, then it holds that $\{C_1, \ldots, C_s\}$ is a $(\tau/5, r+4)$-syntactic partition of C.*

Proof. Claim 7.11 gives $\Delta_{\text{syn}}(C_i) \leq \Delta_{\text{syn}}((C_i)|_{\mathbf{x}_I=\mathbf{a}_I}) + 4 \leq r + 4$. Further, for $i \neq i'$,

$$\begin{aligned}\operatorname{dist}(C_i, C_{i'}) &= \Delta_{\text{syn}}(C_i + C_{i'}) \\ &\geq \Delta_{\text{syn}}((C_i)|_{\mathbf{x}_I=\mathbf{a}_I} + (C_{i'})|_{\mathbf{x}_I=\mathbf{a}_I}) \geq \tau r \geq (\tau/5)(r+4). \qquad \square\end{aligned}$$

Finally, we need a simple lemma regarding restrictions that preserve minimality of circuits.

Lemma 7.13. *Let $C = \sum_{i=1}^{k} T_i$ be a minimal multilinear $\Sigma^k\Pi\Sigma$ circuit, where the T_is are multiplication gates. Assume that k is the minimal integer such that $[C] \in \Sigma^k\Pi\Sigma$. Let B a subset of the variables. Then, there's a polynomial $\Upsilon_{B,C}$ of degree at most $n \cdot 2^{3k}$ with the following property: for every $\mathbf{a} \in \mathbb{F}^n$, if $\Upsilon_{B,C}(\mathbf{a}) \neq 0$ then $f|_{B,\mathbf{a}}$ is minimal. Furthermore, for every $S \neq S'$, $\sum_{i \in S} T_i|_{B,\mathbf{a}} - \sum_{i \in S'}^{k} T_i|_{B,\mathbf{a}} \neq 0$.*

Proof. Consider the symbolic restriction $C|_{B,\mathbf{z}}$ over $\mathbb{F}(\mathbf{z})$. This is still a minimal circuit since this restriction just amounts to renaming some of the $\mathbf{x}$ variables to $\mathbf{z}$. For every non-empty $S \subseteq [k]$, let C_S be the subcircuit of $C|_{B,\mathbf{z}}$ consisting of the multiplication gates in S. C_S computes a non-zero multilinear polynomial in $\mathbf{x}, \mathbf{z}$. Similarly, by the assumption on k, we have that $C_S - C_{S'} \neq 0$. Thus, for every such S and S' there are polynomials Υ_S and $\Upsilon_{S,S'}$, in the $\mathbf{z}$ variables, of degree at most n, such that if $\Upsilon_S(\mathbf{a}) \cdot \Upsilon_{S,S'}(\mathbf{a}) \neq 0$ then the restriction of $\mathbf{z}$ to $\mathbf{a}$ preserves the non-zeroness of both C_S and $C_S - C_{S'}$. Finally take $\Upsilon_{B,C} = \prod_S \Upsilon_S \cdot \prod_{S \neq S'} \Upsilon_{S,S'}$. □

We are now ready to prove Claim 7.2.

Proof. Let $C \in \Sigma^k\Pi\Sigma$ be the minimal circuit in the statement of the claim such that $[C] = f$ and let C_i be the subcircuit of C computing f_i. Thus, $[C|_{B,\mathbf{a}}] = f|_{B,\mathbf{a}}$. Let

$$\Gamma_C = \Phi_C \cdot \left(\prod_B \Upsilon_{B,C} \cdot \left(\prod_{S \subseteq [s]} \Psi_{B,S} \right) \right) \cdot \Xi(\mathbf{x})$$

where the product is over all sets B of size at most $k^{k^{O(k)}}$, Φ_C is as defined in Claim 7.11, $\Upsilon_{B,C}$ is as defined in Lemma 7.13, $\Psi_{B,S}$ is the polynomial Ψ_B from Claim 7.9 applied to the polynomial $\sum_{i \in S} f_i$, and $\Xi(\mathbf{x}) := \prod_{i=1}^{s} f_i \cdot \prod_{i \neq j \in [s]} (f_i + f_j)$ (the fact that $\mathbf{a}$ is not a root of $\Xi(\mathbf{x})$ will not be used in this proof but only later, in the proof of Claim 8.2).

Note that by Claim 7.1 we may assume that $|B| \leq k^{k^{O(k)}}$, which implies the upper bound on the degree of Γ_C.

Let $\mathbf{a}$ be such that $\Gamma_C(\mathbf{a}) \neq 0$. Then in particular $C|_{B,\mathbf{a}}$ is also minimal.

Let $[D] = \sum_{i=1}^{s'} g_i$ be the output of Algorithm 2 on D with parameter τ, where D is a minimal multilinear $\Sigma^k\Pi\Sigma$ circuit computing $f|_{B,\mathbf{a}}$, as in the statement of the claim.

By Claim 5.25, $s' \leq s$ and for every $i \in [s']$ there's a subset $S_i \subseteq [s]$ such that $g_i = \sum_{j \in S_j} (f_j)|_{B,\mathbf{a}}$. Note that if $s' = s$ then the second item of Claim 7.2 follows immediately.

Suppose towards contradiction that $s' < s$. Without loss of generality, suppose that $g_1 = \sum_{j \in S_1} (f_j)|_{B,\mathbf{a}}$ where $|S_1| > 1$. To reach a contradiction we will show that if this was the case then the algorithm would have continued running and in particular would have added more variables to B.

We start by proving that

$$\Delta_{\text{sem}} \left(\sum_{j \in S_1} f_j \right) > \Delta_{\text{sem}} \left(\sum_{j \in S_1} (f_j)|_{B,\mathbf{a}} \right) = \Delta_{\text{sem}}(g_1). \tag{7.14}$$

This is a consequence of the following set of inequalities. First,

$$\Delta_{\text{sem}} \left(\sum_{j \in S_1} f_j \right) \geq \Delta_{\text{syn}} \left(\sum_{j \in S_1} C_j \right) / (2^7 k^2 \log k) \geq \tau_1 r / (2^{11} k^2 \log k), \tag{7.15}$$

where the first inequality follows from Lemma 5.10, and the second from the fact that $|S_1| > 1$, the assumption in the statement of Claim 7.2 and Lemma 4.5. On the other hand, Claim 5.21 promises that the output of Algorithm 2 on D, with parameter τ, satisfies $\Delta_{\text{sem}}(g_1) \leq R_M(2k) 2^{7k} k^{4k} \tau^{k-2}$, which, by (7.15) and our choice of parameters, is indeed smaller than $\Delta_{\text{sem}} \left(\sum_{j \in S_1} f_j \right)$. Thus by Claim 7.9 there's a set I of at most 4 variables such that

$$\Delta_{\text{sem}} \left(\sum_{j \in S_1} (f_j)|_{B \cup I,\mathbf{a}} \right) > \Delta_{\text{sem}} \left(\sum_{j \in S_1} (f_j)|_{B,\mathbf{a}} \right). \tag{7.16}$$

We wish to show that when considering the set I, Algorithm 3 will add it to B in contradiction to the fact that it returned B when run on f.

Denote with $h = \sum_{i=1}^{s_I} h_i$, the output of Algorithm 2 on $f|_{B\cup I,\mathbf{a}}$. If $s_I \neq s'$ then Algorithm 3 would have added I to B. So assume that $s_I = s'$.

The h_i's form a τ-semantic partition of $f|_{B\cup I,\mathbf{a}}$, and thus, from Corollary 5.12, they are also κ-syntactic partition for $\kappa = \tau/(2^7k^2\log k)$. Claim 7.11 shows that the $h_i|_{\mathbf{x}_I=\mathbf{a}_I}$'s are a $(\kappa-4)$-syntactic partition of $f|_{B,\mathbf{a}}$ as well. Finally, Corollary 5.13 shows that they are a $(\kappa-4)/(2^7k^2\log k)$-semantic partition of $f|_{B,\mathbf{a}}$. As the g_i's are also a $(\kappa-4)/(2^7k^2\log k)$-semantic partition of $f|_{B,\mathbf{a}}$, and $s' = s_I$, Corollary 5.15 shows the existence of the a matching between them. That is, there is a permutation π of $[s']$ such that $h_i|_{\mathbf{x}_I=\mathbf{a}_I} = g_{\pi(i)}$.

For simplicity assume that π is the identity permutation. I.e., $h_i|_{\mathbf{x}_I=\mathbf{a}_I} = g_i = \sum_{j\in S_j}(f_j)|_{B,\mathbf{a}}$. Going back to Algorithm 3 we see that the algorithm can find the required map σ (this is the π that we found).

We now wish to show that $h_1 = \sum_{j\in S_1}(f_j)|_{B\cup I,\mathbf{a}}$. If this is not the case then, as before, there is a set $S_1' \neq S_1$ such that $h_1 = \sum_{j\in S_1'}(f_j)|_{B\cup I,\mathbf{a}}$. Hence

$$\sum_{j\in S_1'}(f_j)|_{B,\mathbf{a}} = h_1|_{B,\mathbf{a}} = \sum_{j\in S_1}(f_j)|_{B,\mathbf{a}}.$$

Therefore,

$$\sum_{j\in S_1'}(C_j)|_{B,\mathbf{a}} - \sum_{j\in S_1}(C_j)|_{B,\mathbf{a}} = \sum_{j\in S_1'}(f_j)|_{B,\mathbf{a}} - \sum_{j\in S_1}(f_j)|_{B,\mathbf{a}} = 0$$

As $S_1' \neq S_1$, this contradicts the fact that $\Upsilon_{S_1,S_1'}(\mathbf{a}) \neq 0$ (recall Lemma 7.13).

Concluding, the algorithm found the partition $h = \sum_{i=1}^{s_I} h_i$ such that $s_I = s'$, a matching between the h_i's and g_i's, and it holds that $h_1 = \sum_{j\in S_1}(f_j)|_{B\cup I,\mathbf{a}}$. This means that, when considering the set I, the algorithm would add it to B, due to (7.16), in contradiction.

We have therefore shown that it must be the case that $s = s'$. Corollary 5.15 implies the second item in the claim, which also implies the third item. □

7.2.1 Proof of Claim 7.9

We first prove some preliminary lemmas. We start with an obvious observation.

Lemma 7.17. *Let $f \in \mathbb{F}[\mathbf{x}]$ be a polynomial and $\mathbf{a} \in \mathbb{F}^n$. Let $g(\mathbf{x}) = f(\mathbf{x}+\mathbf{a})$. Then, $rank(M_f) = rank(M_g)$.*

The next claim shows that we can add two variables to B and increase the rank of the partial derivative matrix (note that this doesn't imply Claim 7.9, as the rank of a polynomial is the rank of this matrix after pulling out the linear factors).

Claim 7.18. *Let f be a multilinear polynomial and B a subset of the variables. Suppose that $rank(M_f) = r$. There exists a polynomial Λ_B of degree at most n^2 with the following property: for every $\mathbf{a} \in \mathbb{F}^n$ such that $\Lambda_B(\mathbf{a}) \neq 0$, if $rank(M_{f|_{B,\mathbf{a}}}) = t < r$, then there are two variables $x_i, w \notin B$ such that*

$$rank(M_{f|_{B\cup\{x_i,w\},\mathbf{a}}}) > t.$$

Some preliminary work is required before proving Claim 7.18. A polynomial $P(x) = \sum_m c_m \cdot m \in \mathbb{F}^t[\mathbf{x}]$ is called a vector polynomial. I.e., it is a polynomial whose coefficients $c_m \in \mathbb{F}^t$. We denote by $V(P)$ the vector space spanned by its coefficients.

For a multilinear polynomial f we define P_f the polynomial whose coefficients are the corresponding columns of the matrix M_f. In particular, as f is multilinear, the coefficient $c_M \in \mathbb{F}^n$ of a monomial M satisfies: $(c_M)_i$ is the coefficient of $M \cdot x_i$ in f.

The following remark can be verified by direct inspection:

Remark 7.19. *Let $h(\mathbf{x}) = f(\mathbf{x}+\mathbf{z})$. Then $P_h(\mathbf{x}) = P_f(\mathbf{x}+\mathbf{z})$.* ♢

The next key claim follows from a work of Forbes-Ghosh-Saxena [FGS18].

Claim 7.20. *Let f be a multilinear polynomial and B a subset of the variables. Suppose $V(P_f)$ has a basis $\mathcal{B}$ such that the monomials corresponding to all but at most one of the basis elements are supported only on variables of B.*

Then, for some $x_i \notin B$, $V(P_{f(\mathbf{x}+\mathbf{z}')})$ (which is a vector space over $\mathbb{F}(\mathbf{z}')$) has a basis[1] $\mathcal{A}$ such that the monomials corresponding to all *the basis elements are supported only on variables of $B \cup \{x_i\}$.*

Proof. Order the variables so that the variables in B appear first, and then the variables outside of B.

The claim follows from closely looking at Algorithm 1 and Theorem 2 of [FGS18]. We use $\mathcal{B}$ and $\mathcal{A}$, respectively, to denote the sets called B and A in their algorithm. The fact the $\mathcal{A}$ is a basis (and is in fact cone-closed) follows from Theorem 2 of [FGS18]. The fact that $\mathcal{A}$ is supported on $B \cup \{x_i\}$ for some variable x_i follows by inspecting their Algorithm 1: Indeed, note that, starting from $\mathcal{B}$, as long as the projection map π (defined in their algorithm) is one-to-one (i.e., in the notation of that algorithm $\ell = 1$) their algorithm "erases" the last variable and continues recursively. In the first time that $\ell = 2$, since we started with a basis $\mathcal{B}$ such that the monomials corresponding to all but at most one of the basis elements are supported on only variables of B, it must be the case that the current set held by the algorithm is supported on only variables of $B \cup \{x_i\}$. This fact doesn't change until the completion of the algorithm. □

Claim 7.21. *Let f be as in Claim 7.18. There exists a non-zero n-variate polynomial $\Lambda_B(\mathbf{y})$, of degree at most n^2, such that if $\Lambda_B(\mathbf{a}) \neq 0$, then, $g(\mathbf{x}) = f(\mathbf{x}+\mathbf{a})$ satisfies rank$(M_{g|_{B,0}}) = t$, and there exist $x_i, w \notin B$ such that for $B' = B \cup \{x_i, w\}$, rank$(M_{g|_{B',0}}) > t$.*

Proof. Consider the symbolic shift $g(\mathbf{x}) = f(\mathbf{x}+\mathbf{z})$ where $\mathbf{z}$ is a new set of variables and $g \in \mathbb{F}(\mathbf{z})[\mathbf{x}]$. By Lemma 7.17, $\operatorname{rank}(M_g) = \operatorname{rank}(M_f) = r$ (note that M_g is defined over $\mathbb{F}(\mathbf{z})$). For simplicity of notation, suppose that $B = \{x_1, \ldots, x_b\}$ for some $1 \leq b \leq n$. Observe that

$$\begin{aligned} f|_{B,\mathbf{z}}(\mathbf{x}) &= f(x_1, \ldots, x_b, z_{b+1}, \ldots, z_n) \\ &= f((x_1+z_1)-z_1, \ldots, (x_b+z_b)-z_b, z_{b+1}, \ldots, z_n) \\ &= g(x_1-z_1, \ldots x_b-z_b, 0, \ldots, 0) = g|_{B,0}(\mathbf{x}-\mathbf{z}). \end{aligned}$$

[1] The proof of [FGS18] gives a stronger property – that the basis $\mathcal{A}$ is *cone closed*. I.e. that it is closed under taking submonomials. In other words, if the coefficient if a monomial M is in the basis and N divides M then so is the coefficient of N. We do not need this stronger property.

Therefore,

$$t = \operatorname{rank}(M_{f|_{B,\mathbf{z}}}) = \operatorname{rank}(M_{g|_{B,0}(\mathbf{x}-\mathbf{z})}).$$

By Lemma 7.17, it now follows that

$$t = \operatorname{rank}(M_{g|_{B,0}(\mathbf{x})}).$$

Suppose the partial derivatives corresponding to $x_1, \ldots, x_t$ are a row basis for $M_{g|_{B,0}}$. There exists a corresponding set of t columns, i.e., monomials in the variables of B, such that the submatrix has rank t. Since $M_{g|_{B,0}}$ is a submatrix of M_g, these rows and columns are also linearly independent in M_g. Since $t < r$, we can pick a variable $w \notin B$ such that $\partial g/\partial w$ is linearly independent of $\partial g/\partial x_1, \ldots, \partial g/\partial x_t$. We can similarly add another monomial (which would necessarily contain a variable not in B) to get a $(t+1) \times (t+1)$ submatrix of M_g that has full rank.

Our next goal is to restrict only to columns corresponding to monomials in B, w and perhaps another variable $x_i \notin B$ while maintaining the linear independence of the $t+1$ rows.

By Claim 7.20, if we look at $g(\mathbf{x}+\mathbf{z}') = f(\mathbf{x}+\mathbf{z}+\mathbf{z}')$, where $\mathbf{z}'$ is yet again another symbolic shift, then the matrix $M_{f(\mathbf{x}+\mathbf{z}+\mathbf{z}')}$, restricted to the rows of $B \cup \{w\}$, has such a basis. In particular there's some $(t+1) \times (t+1)$ minor of this matrix whose determinant is non-zero in $\mathbb{F}(\mathbf{z}+\mathbf{z}')$. As in $M_{f(\mathbf{x}+\mathbf{z}+\mathbf{z}')}$ every entry is a polynomial in $\mathbf{z}+\mathbf{z}'$, this implies that when doing the "symbolic" shift $f(\mathbf{x}+\mathbf{z}+\mathbf{z}')$, and looking at the same minor, its determinant is a non-zero polynomial Λ_B in $\mathbf{z}+\mathbf{z}'$ of degree at most $(t+1)n$. By doing the change of variables $\mathbf{y} = \mathbf{z}+\mathbf{z}'$ we can think of Λ_B also as a polynomial in a new set of variables $\mathbf{y}$.

Thus, we have found a non-zero polynomial $\Lambda_B(\mathbf{y})$ such that whenever $\Lambda_B(\mathbf{a}) \neq 0$, it holds, for $g(\mathbf{x}) = f(\mathbf{x}+\mathbf{a})$ and $B' = B \cup \{x_i, w\}$, that $\operatorname{rank}(M_{g|_{B',0}}) \geq t+1$. □

Proof of Claim 7.18. Let $\mathbf{a}$ be such that $\Lambda_B(\mathbf{a}) \neq 0$, where Λ_B is as in Claim 7.21. Let $g(\mathbf{x}) = f(\mathbf{x}+\mathbf{a})$ and $B' = B \cup \{w, x_i\}$ where x_i, w are as guaranteed by Claim 7.21. By Lemma 7.17 it follows that

$$\operatorname{rank}(M_{g|_{B',0}(\mathbf{x}-\mathbf{a})}) = \operatorname{rank}(M_{g|_{B',0}}) \geq t+1.$$

Since $g|_{B',0}(\mathbf{x}-\mathbf{a}) = f|_{B',\mathbf{a}}(\mathbf{x})$, Claim 7.18 follows. □

Having proved Claim 7.18, we now move on to handle the linear factors.

Claim 7.22. *Let P be an irreducible multilinear polynomial of degree at least 2 and $B \subseteq [n]$. Then, there exists a polynomial $\Lambda'_{B,P}$ of degree at most $2n^3$ with the following property: for every $\mathbf{a} \in \mathbb{F}^n$, if $\Lambda'_{B,P}(\mathbf{a}) \neq 0$ then there are $y_1, y_2 \notin B$ such that $P|_{B\cup\{y_1,y_2\},\mathbf{a}}$ has no linear factors.*

Proof. Without loss of generality, suppose $B = \{x_1, \ldots, x_b\}$. Suppose further that $P|_{B,\mathbf{a}}$ has a linear factor. Write $P|_{B,\mathbf{a}} = \ell \cdot P'$ where, without loss of generality, $\ell = \sum_{i=1}^{t} \beta_i x_i$ with $\beta_i \neq 0$ and P' depends on $x_{t+1}, \ldots, x_b$.

Assume first that $\deg(P') \geq 1$. For every $i \in [t]$ we can write $P = x_i \cdot h_{i,1} + h_{i,0}$ with $h_{i,0} \neq 0$ (since P is irreducible).

It holds that $P|_{B,\mathbf{a}} = x_i \cdot h_{i,1}|_{B,\mathbf{a}} + h_{i,0}|_{B,\mathbf{a}}$. Since $P|_{B,\mathbf{a}}$ is divisible by ℓ, $\mathrm{Res}_{x_i}(P|_{B,\mathbf{a}}, \ell) = 0$ (recall Section 2.5). Note that $\mathrm{Res}_{x_i}(P|_{B,\mathbf{a}}, \ell) = \beta_1 \cdot h_{i,0}|_{B,\mathbf{a}} - h_{i,1}|_{B,\mathbf{a}} \cdot (\ell - \beta_1 x_1)$, which implies that $h_{i,1}|_{B,\mathbf{a}}$ divides $h_{i,0}|_{B,\mathbf{a}}$.

We claim that for all $i \in [t]$, $h_{i,1}$ depends on at least one variable in B. Indeed, write $P = \sum_{j=b+1}^{n}(x_j - \mathbf{a}_j)A_j + \ell P'$. The variable x_i appears in ℓ and it is multiplied by P', which is a nonconstant polynomial in the B variables. All of its other occurrences are multiplied by $(x_j - \mathbf{a}_j)$ for some $j \geq b+1$, which implies that there's at least one coefficient with a variable $x_{j_i} \in B$ that appears in $h_{i,1}$.

Since P is irreducible, $\gcd(h_{i,1}, h_{i,0}) = 1$ and therefore $\mathrm{Res}_{x_{j_i}}(h_{i,1}, h_{i,0}) \neq 0$.

Let $c_1(\mathbf{x})$ denote the coefficient of x_{j_i} in $h_{i,1}$ and similarly $c_0(\mathbf{x})$ the coefficient of x_{j_i} in $h_{i,0}$ (these are polynomials that may depend on all variables except x_i and x_{j_i} and $c_1(\mathbf{x}) \neq 0$).

If $(c_1)|_{B,\mathbf{a}} \neq 0$ and $(c_0)|_{B,\mathbf{a}} \neq 0$, then it holds that $\mathrm{Res}_{x_{j_i}}(h_{i,1}|_{B,\mathbf{a}}, h_{i,0}|_{B,\mathbf{a}}) = \mathrm{Res}_{x_{j_i}}(h_{i,1}, h_{i,0})|_{B,\mathbf{a}}$ (see, e.g., Proposition 6 in Chapter 3, Section 6 of [CLO07]).

Since $\mathrm{Res}_{x_{j_i}}(h_{i,1}, h_{i,0})$ is a polynomial of degree at most $2n$ and c_0, c_1 are polynomials of degree at most n, by requiring that their restriction to B and $\mathbf{a}$ are non-zero we obtain a polynomial $\Lambda'_{B,i,j_i} = c_1 \cdot c_0 \cdot \mathrm{Res}_{x_{j_i}}(h_{i,1}, h_{i,0})$ of degree at most $4n$ such that if $\Lambda'_{B,i,j_i}(\mathbf{a})$ is non-zero then $\mathrm{Res}_{x_{j_i}}(h_{i,1}, h_{i,0})|_{B,\mathbf{a}}$ is non-zero, which implies that $\mathrm{Res}_{x_{j_i}}(h_{i,1}|_{B,\mathbf{a}}, h_{i,0}|_{B,\mathbf{a}})$ is non-zero, in contradiction to the assumption that ℓ divides $P|_{B,\mathbf{a}}$. We define $\Lambda'_{B,P}$ to be the product of all non-zero polynomials $\Lambda'_{B,i,j}$. Thus, if $\Lambda'_{B,P}(\mathbf{a}) \neq 0$, it can't be the case that $P|_{B,\mathbf{a}} = \ell \cdot P'$ for $\deg(P') \geq 1$. Observe that in this case no variables were added to B.

Suppose then that $P|_{B,\mathbf{a}} = \ell$ for a linear function ℓ. Then, we can add at most two variables y_1, y_2 to B so that $P|_{B \cup \{y_1,y_2\},\mathbf{a}}$ is not a linear function. Indeed, since P is not a linear function, it contains a monomial m of degree at least 2. This monomial must contain at least one variable not in B (since otherwise $P|_{B,\mathbf{a}}$ wouldn't have been linear). Thus, there exist at least one and at most two variables y_1, y_2 not in B such that adding them to B would give a polynomial of degree 2 under the restriction. Having done that, we return to the previous case and similarly obtain $\Lambda'_{B,P}$. □

Corollary 7.23. *Let P be an irreducible multilinear polynomial with no linear factors and B a subset of the variables. Suppose there exists $\mathbf{a} \in \mathbb{F}^n$ such that $P|_{B,\mathbf{a}}$ is non-constant and has no linear factors. Then, there exists a non-zero polynomial Λ'_B of degree at most $2n^3$ such that $\Lambda'_B(\mathbf{a}) \neq 0$ and for all $\mathbf{b} \in \mathbb{F}^n$, if $\Lambda'_B(\mathbf{b}) \neq 0$ then $P|_{B,\mathbf{b}}$ has no linear factors.*

Proof. Observe that in the proof of Claim 7.22, when $\deg(P') \geq 1$, the polynomial $\Lambda'_{B,P}$ has the claimed property. □

We are now ready to prove Claim 7.9.

Proof of Claim 7.9. Write $f = \prod_{i=1}^{a} \ell_i \cdot \prod_{i=1}^{b} P_i$ where the ℓ_i's are linear functions and the P_i's are irreducible non-linear polynomials. Denote $P = \prod_{i=1}^{b} P_i$ to be the non-linear part of f. By definition, it holds that $\Delta_{\mathrm{sem}}(f) = \mathrm{rank}(M_P)$.

Similarly, write $f|_{B,\mathbf{a}} = \prod_{i=1}^{a}(\ell_i)|_{B,\mathbf{a}} \prod_{i=1}^{b}(P_i)|_{B,\mathbf{a}}$.

Set $\Psi_B = \Lambda_B \cdot \Lambda'_B$ where Λ_B is as in Claim 7.18 and $\Lambda'_B = \prod_{i=1}^{b} \Lambda'_{B,P_i}$, where Λ'_{B,P_i} is as defined in Claim 7.22. Ψ_B is a polynomial of degree at most $4n^6$. Pick $\mathbf{a}$ such that $\Psi_B(\mathbf{a}) \neq 0$.

We split the proof into two cases.

1. No $(P_i)|_{B,\mathbf{a}}$ has linear factors. In this case, $\Delta_{\text{sem}}(f|_{B,\mathbf{a}}) = \text{rank}(M_{P|_{B,\mathbf{a}}})$. By Claim 7.18, there exist two variables x, w not in B such that for $B' = B \cup \{x, w\}$, $\text{rank}(M_{P|_{B',\mathbf{a}}}) > \text{rank}(M_{P|_{B,\mathbf{a}}})$.

   We split this case further into two subcases.

   (a) No $(P_i)|_{B',\mathbf{a}}$ has linear factors. In this case, $\Delta_{\text{sem}}(f|_{B',\mathbf{a}}) = \text{rank}(M_{P|_{B',\mathbf{a}}})$ and the proof is completed.

   (b) There exist i such that $(P_i)|_{B',\mathbf{a}}$ has linear factors. This case is handled similarly to case 2 below by setting $B = B'$.

2. There exists $i \in [b]$ such that $(P_i)|_{B,\mathbf{a}}$ has linear factors. Write

$$f|_{B,\mathbf{a}} = \prod_{i=1}^{a} (\ell_i)_{B,\mathbf{a}} \cdot \prod_{i=1}^{b} \left(\prod_{j=1}^{b_i} (\tilde{\ell}_{i,j}) \right) \cdot P_{i,B,\mathbf{a}},$$

   where the $\tilde{\ell}_{i,j}$'s denote the linear factors added in the restriction, and $P_{i,B,\mathbf{a}}$ has no linear factors. Let $P_{B,\mathbf{a}} = \prod_{i=1}^{b} P_{i,B,\mathbf{a}}$ so that $\Delta_{\text{sem}}(f|_{B,\mathbf{a}}) = \text{rank}(M_{P_{B,\mathbf{a}}})$.

   By the assumption there exists i such that $b_i \geq 1$. Without loss of generality suppose $i = 1$. I.e., P_1 is an irreducible polynomial of degree at least 2 that under the restriction has linear factors. By Claim 7.22, we can add two new variables to B to obtain B' such that $(P_1)|_{B',\mathbf{a}}$ has no linear factors. We now claim that $\Delta_{\text{sem}}(f|_{B',\mathbf{a}}) > \Delta_{\text{sem}}(f|_{B,\mathbf{a}})$.

   We start by similarly splitting $f|_{B',\mathbf{a}}$ into a product of linear functions times a polynomial $P_{B',\mathbf{a}}$ such that $\Delta_{\text{sem}}(f|_{B',\mathbf{a}})$ is the number of linearly independent linear functions that $P_{B',\mathbf{a}}$ depends on. First, we note that

$$\Delta_{\text{sem}}(f|_{B',\mathbf{a}}) = \Delta_{\text{sem}}(P_{B',\mathbf{a}}) \geq \Delta_{\text{sem}}(P_{B',\mathbf{a}}|_{B,\mathbf{a}}),$$

   so it suffices to prove that $\Delta_{\text{sem}}(P_{B',\mathbf{a}}|_{B,\mathbf{a}}) > \Delta_{\text{sem}}(P_{B,\mathbf{a}})$. We first claim that $P_{B,\mathbf{a}}$ divides $P_{B',\mathbf{a}}|_{B,\mathbf{a}}$: consider the process of obtaining $f|_{B,\mathbf{a}}$ from $f|_{B',\mathbf{a}}$ by restricting the variables in $B' \setminus B$: some irreducible factors of $f|_{B',\mathbf{a}}$ become linear; Others remain non-linear and appear in the product defining $P_{B,\mathbf{a}}$, and those must come from restrictions to B of non-linear irreducible factors that appear in $P_{B',\mathbf{a}}$. In particular, their restrictions to B appear in $P_{B',\mathbf{a}}|_{B,\mathbf{a}}$, which implies that $P_{B,\mathbf{a}}$ divides $P_{B',\mathbf{a}}|_{B,\mathbf{a}}$. Further, we claim that $\tilde{\ell}_{1,1}$ divides $P_{B',\mathbf{a}}|_{B,\mathbf{a}}$. Indeed, $(P_1)|_{B',\mathbf{a}}$ has no linear factors. Thus by definition of $P_{B',\mathbf{a}}$,

$$(P_1)|_{B',\mathbf{a}} \mid P_{B',\mathbf{a}}.$$

   Restricting both sides further to B, we get on the left hand side a product involving $\tilde{\ell}_{1,1}$ and on the right hand side $P_{B',\mathbf{a}}|_{B,\mathbf{a}}$.

   Thus, $P_{B',\mathbf{a}}|_{B,\mathbf{a}}$ depends on at least $\Delta_{\text{sem}}(P_{B,\mathbf{a}}) + 1$ linear functions: the $\Delta_{\text{sem}}(P_{B,\mathbf{a}})$ linear functions that $P_{B,\mathbf{a}}$ depend on, plus $\tilde{\ell}_{1,1}$, which is variable disjoint and hence linearly independent of all of them. □

7.3 Why Semantic Rank

In this section we add some more technical details to the explanation in Section 1.3.4. Recall that Theorem 4.11 proved that there are certain parameters that guarantee uniqueness of clusters in a syntactic partition. This result is enough to fix the relevant theorems in [KS09a, BSV21]. As explained, one main contribution of this work is an algorithmic construction of a set B such that restricting (a random shift of) the black-box polynomial f to the variables in B, maintains the cluster structure of any $\Sigma^k\Pi\Sigma$ circuit for f. I.e., the new clusters are (up to a permutation) equal to the restrictions of the clusters of f to B.

We constructed B gradually (Algorithm 3). At each step we checked whether adding a small set of variables increases either the number of clusters of $f|_B$ or the rank of at least one cluster. To prove that this process finds B that preserves the cluster structure, we had to prove that if this was not the case then we could have found variables to add to B (Claim 7.9). We then needed to prove that if we guessed the right set of variables to add then we can *verify* that they indeed contributed to the rank (Step 13 of Algorithm 3). All of this works when using *semantic* rank. However, when using *syntactic* rank this is not quite the case. The problem is that when we learn f_I (Step 7 in Algorithm 3) then it may be the case that $C'_j|_{x_I=\mathbf{a}_I} = C_j$, yet $\Delta_{\text{syn}}(C'_j) < \Delta_{\text{syn}}(C_j)$, something that cannot happen if we work with semantic rank. Indeed, the simple example (where ω is a primitive root of unity of order d)

$$\sum_{i=1}^{k}\prod_{j=1}^{d}(x_i - \omega^j x_{i+1}) = x_1^d - x_{k+1}^d = \prod_{j=1}^{d}(x_1 - \omega^j x_{k+1})$$

shows that the left hand side has syntactic rank $k+1$ while the middle expression has syntactic rank 2. Both sides have semantic rank 0 (although recall that by Remark 5.3, it's sometimes convenient to define this rank as 1). We can also cook up such examples where the number of multiplication gates is the same in both circuits, and where the circuits are multilinear and have semantic rank at least 1. Thus, we were able to prove the correctness of Algorithm 3 relying on semantic rank and not on syntactic rank.

8 Reconstruction Algorithm for Multilinear $\Sigma^k\Pi\Sigma$ Circuits

In this section we provide our algorithm for learning multilinear $\Sigma^k\Pi\Sigma$ circuits. Our proof is similar in structure to the proof of Bhargava, Saraf and Volkovich [BSV21]. However, since many of our definitions are different we are required to prove several basic claims. We start by explaining how the results of the previous sections imply that we can get black box access to the clusters on arbitrary points. In the previous section, we picked a random $\mathbf{a}$ in Algorithm 3 and argued about the clusters of $f|_{B,\mathbf{a}}$. We'd like to obtain similar claims about the clusters of $f|_{B,\mathbf{b}}$, assuming $\mathbf{b}$ doesn't satisfy certain degeneracy conditions.

Lemma 8.1. *Let P be a multilinear polynomial and B a subset of the variables. Suppose that $\Delta_{sem}(P|_{B,\mathbf{a}}) \geq t$ and that $P(\mathbf{a}) \cdot \Lambda'_B(\mathbf{a}) \neq 0$ (where Λ'_B is as defined in Corollary 7.23). Then, there exists a non-zero polynomial Θ_B of degree at most $2n^5$ such that $\Theta_B(\mathbf{a}) \neq 0$ and, if $\Theta_B(\mathbf{b}) \neq 0$, then $\Delta_{sem}(P|_{B,\mathbf{b}}) \geq t$.*

Proof. Suppose without loss of generality that P is irreducible (as otherwise argue separately on each irreducible factor). We may also assume that $P|_{B,\mathbf{a}}$ is non-constant as otherwise the statement is trivial. If $P|_{B,\mathbf{a}}$ has no linear factors, then

$$\Delta_{\text{sem}}(P|_{B,\mathbf{a}}) = \text{rank}(M_{P|_{B,\mathbf{a}}}).$$

Since by assumption $\text{rank}(M_{P|_{B,\mathbf{a}}}) \geq t$, there's a $t \times t$ minor whose determinant is non-zero. This implies that, in the matrix $M_{P|_{B,\mathbf{z}}}$, the same $t \times t$ minor has non-zero determinant, as a polynomial in $\mathbf{z}$. Denote this determinant with $\text{Det}(\mathbf{z})$, and observe that $\text{Det}(\mathbf{a}) \neq 0$. Furthermore, as each coefficient in $P|_{B,\mathbf{z}}$ as degree at most n as a polynomial in $\mathbf{z}$, and $t \leq n$ we have that $\deg(\text{Det}) \leq n^2$.

Define $\Theta_B := \Lambda'_B \cdot \text{Det}$ where Λ'_B is as in Corollary 7.23. As $\deg(\Lambda'_B) \leq 2n^3$ we get that $\deg(\Theta_B) \leq 2n^5$ and that $\Theta_B(\mathbf{a}) \neq 0$.

Let $\mathbf{b}$ be such that $\Theta_B(\mathbf{b}) \neq 0$. Corollary 7.23 guarantees that $P|_{B,\mathbf{b}}$ has no linear factors, and since $\text{Det}(\mathbf{b}) \neq 0$, $\text{rank}(M_{P|_{B,\mathbf{b}}}) \geq t$. This implies that $\Delta_{\text{sem}}(P|_{B,\mathbf{b}}) \geq t$.

Now suppose $P|_{B,\mathbf{a}}$ has linear factors and can be written as $L \cdot P'$ where L is a product of linear factors and P' has no linear factors, so that $\Delta_{\text{sem}}(P|_{B,\mathbf{a}}) = \text{rank}(M_{P'})$. Let B' be the set of variables that appears in P' (and is disjoint from the set of variables that appear in L). Note that since $P(\mathbf{a}) \neq 0$ it must be the case that $P|_{B',\mathbf{a}}$ is non-zero and hence $L|_{B',\mathbf{a}}$ is a non-zero constant, i.e., $P' = cP|_{B',\mathbf{a}}$ for some $c \neq 0$.

We now apply the same reasoning as before to P', which has no linear factors, to deduce that for some non-zero polynomial Θ_B, if $\Theta_B(\mathbf{b}) \neq 0$ then $\Delta_{\text{sem}}(P|_{B',\mathbf{b}}) \geq \Delta_{\text{sem}}(P|_{B',\mathbf{a}}) = \Delta_{\text{sem}}(P') = t$. Finally, note that as $B' \subseteq B$, $\Delta_{\text{sem}}(P|_{B,\mathbf{b}}) \geq \Delta_{\text{sem}}(P|_{B',\mathbf{b}}) \geq t$. □

We say that an output $(B, \mathbf{a})$ of Algorithm 3 is *good* if it satisfies $\Gamma_C(\mathbf{a}) \neq 0$, where Γ_C is the polynomial defined in Claim 7.2.

Claim 8.2. *Let $f \in \Sigma^k\Pi\Sigma$ be a multilinear polynomial and let C be a minimal $\Sigma^k\Pi\Sigma$ computing f. Let $(B, \mathbf{a})$ be a good output of Algorithm 3 on f.*

Consider a partition of f, $f = \sum_{i=1}^{s} f_i$, as given by Claim 5.24 with $\varphi(k) = k^2$ and τ_{min} as in Claim 7.2. Let τ_0, τ_1, r be its parameters as promised by the claim. Denote $\tau = \tau_0^k$.

Then, there exists a polynomial $\Theta_{B,C}$ of degree at most $2n^7$ such that $\Theta_{B,C}(\mathbf{a}) \neq 0$, and the following property holds: for every $\mathbf{b} \in \mathbb{F}^n$ such that $\Theta_{B,C}(\mathbf{b}) \neq 0$ and circuit D computing $f|_{B,\mathbf{b}}$, it holds that the output of Algorithm 2, when given D and τ as input, which we denote $[D] = \sum_{i=1}^{s'} g_i$, satisfies:

1. *$s' = s$,*
2. *$g_i = (f_i)|_{B,\mathbf{b}}$, up to reordering of the indices,*
3. *$\Delta_{sem}(g_i) = \Delta_{sem}(f_i)$.*

In particular, the g_i's also form a (τ, r) partition.

Proof. By Claim 7.2, we know that there exists a choice of $\mathbf{b}$ (that is, $\mathbf{b} = \mathbf{a}$) that satisfies the required properties. We first show that there is a non-zero polynomial $\Theta_{B,C}$ of degree at most $2n^7$ such that if $\Theta_{B,C}(\mathbf{b}) \neq 0$ then

1. for every $i \in [s]$, $\Delta_{\text{sem}}(f_i|_{B,\mathbf{b}}) = \Delta_{\text{sem}}(f_i)$,
2. for every $i \neq j \in [s]$, $\Delta_{\text{sem}}(f_i|_{B,\mathbf{b}}, f_j|_{B,\mathbf{b}}) \geq \tau r$.

For the first item, denote by Θ_i the polynomial promised by Lemma 8.1 applied to f_i and B. Note that since $(B, \mathbf{a})$ are good and $\Gamma_C(\mathbf{a}) \neq 0$ (where Γ_C is as defined in the proof of Claim 7.2) and the assumptions on $\mathbf{a}$ in the statement of Lemma 8.1 indeed hold. In particular, $\Theta_i(\mathbf{a}) \neq 0$. Thus, if $\Theta_i(\mathbf{b}) \neq 0$ then

$$\Delta_{\text{sem}}((f_i)|_{B,\mathbf{b}}) \geq \Delta_{\text{sem}}((f_i)|_{B,\mathbf{a}}) = \Delta_{\text{sem}}(f_i),$$

and the reverse inequality is clear.

For the second item, we similarly let $\Theta_{i,j}$ be the polynomial promised by Lemma 8.1 applied to $f_i + f_j$ and B. The assumptions on $\mathbf{a}$ in the statement of Lemma 8.1 again hold because $(B, \mathbf{a})$ is good.

Finally, set $\Theta_{B,C} = (\prod_{i=1}^{s} \Theta_i) \cdot (\prod_{i \neq j \in [s]} \Theta_{i,j})$. Note that $\Theta_{B,C}(\mathbf{a}) \neq 0$ as each factor is non-zero. It is also clear that $\deg(\Theta_{B,C}) \leq 2n^5(n + \binom{n}{2}) < 2n^7$.

Let $\mathbf{b}$ be such that $\Theta_{B,C}(\mathbf{b}) \neq 0$. Consider Algorithm 2 run on $f|_{B,\mathbf{b}}$ with parameter τ. Denote its output by $f|_{B,\mathbf{b}} = \sum_{i=1}^{s'} g_i$. By Claim 5.25, we have that $s' \leq s$. On the other hand, as $\Theta_{B,C}(\mathbf{b}) \neq 0$, Lemma 8.1 implies that the set $\{(f_i)|_{B,\mathbf{b}}\}$ for $i \in [s]$ is a τ partition with s clusters. Hence, by Corollary 5.18, and the fact that Algorithm 2 returns the partition of minimal rank (and in particular with maximal number of clusters), the output will be $\{(f_i)|_{B,\mathbf{b}}\}$, as we wanted to show. □

Claim 8.3. *Let $f \in \Sigma^k\Pi\Sigma$ be a multilinear polynomial and let C be a $\Sigma^k\Pi\Sigma$ circuit computing f. Let $(B, \mathbf{a})$ be good outputs of Algorithm 3 on f. Let $B' \supseteq B$.*

For every $\mathbf{b} \in \mathbb{F}^n$ such that $\Theta_{B,C}(\mathbf{b}) \neq 0$ the following holds: Denote by $f|_{B,\mathbf{b}} = \sum_{i=1}^{s}(f_i)|_{B,\mathbf{b}}$ the output of Algorithm 2 on $f|_{B,\mathbf{b}}$ with parameter τ (as promised by Claim 8.2). Let $f|_{B',\mathbf{b}} = \sum_{i=1}^{s'} g_i$ be the output of Algorithm 2 on $f|_{B',\mathbf{b}}$ with parameter τ. Then $s = s'$ and up to permutation of the indices, $g_i = (f_i)|_{B',\mathbf{b}}$.

Proof. By Claim 8.2, it holds that:

1. for every $i \in [s]$, $\Delta_{\text{sem}}((f_i)|_{B,\mathbf{b}}) = \Delta_{\text{sem}}(f_i)$.
2. for every $i \neq j \in [s]$, $\Delta_{\text{sem}}((f_i)|_{B,\mathbf{b}}, (f_j)|_{B,\mathbf{b}}) \geq \tau r$.

Thus, as $B' \supseteq B$,

1. for every $i \in [s]$, $\Delta_{\text{sem}}(f_i) = \Delta_{\text{sem}}((f_i)|_{B,\mathbf{b}}) \leq \Delta_{\text{sem}}((f_i)|_{B',\mathbf{b}}) \leq \Delta_{\text{sem}}(f_i)$.
2. for every $i \neq j \in [s]$, $\Delta_{\text{sem}}((f_i)|_{B',\mathbf{b}}, (f_j)|_{B',\mathbf{b}}) \geq \tau r$.

Thus, $(f_i)|_{B',\mathbf{b}}$ is a τ partition with s clusters. Claim 5.25 and Corollary 5.18 show that $s' = s$ and up to permutation of the indices, $g_i = (f_i)|_{B',\mathbf{b}}$. □

8.1 Cluster Evaluation

In this section we explain how to evaluate the clusters at arbitrary points. Recall that what we have is access to the clusters $f_i|_{B,\mathbf{a}}$ so we'd like to replace $\mathbf{a}$ by an arbitrary point $\mathbf{b} \in \mathbb{F}^n$. We do it in several stages as in [BSV21]. Basically, we replace all uses of their Lemma 6.14 by our Claim 8.2. We briefly explain how this is done.

The first step shows how to evaluate the clusters of $f|_{B,\mathbf{b}'}$ if the Hamming distance of $\mathbf{b}$ and $\mathbf{b}'$ is 1.

Lemma 8.4 (Similar to Lemma 6.17 in [BSV21])**.** *Let $f \in \Sigma^k\Pi\Sigma$ be a multilinear polynomial and let C be a minimal multilinear $\Sigma^k\Pi\Sigma$ circuit computing f. Let $(B, \mathbf{a})$ a good output of Algorithm 3 on f. Let $\mathbf{b}$ such that $\Theta_{B,C}(\mathbf{b}) \neq 0$, where $\Theta_{B,C}$ is as given in Claim 8.2, and $f|_{B,\mathbf{b}} = \sum_{i=1}^{s} f_i|_{B,\mathbf{b}}$ the output of Algorithm 2 with parameter τ on $f|_{B,\mathbf{b}}$.*

Let $\mathbf{b}' \in \mathbb{F}^n$ be of Hamming distance 1 from $\mathbf{b}$. Then, there exists an algorithm that runs in time $2^{k^2} \cdot \mathsf{poly}(n)$ and outputs $(f_1|_{B,\mathbf{b}'}, \ldots, f_s|_{B,\mathbf{b}'})$.

Proof. Let j be the coordinate on which $\mathbf{b}'$ differs from $\mathbf{b}$. We may assume $j \notin B$ as otherwise the statement is immediate. Let $B' = B \cup \{x_j\}$ and run Algorithm 2 on $f|_{B',\mathbf{b}}$, and denote its output by $\sum_{i=1}^{s} g_i$. By Claim 8.3, $g_i = f_i|_{B',\mathbf{b}}$ up to permutation of the indices. It is easy to find the permutation by running a randomized PIT algorithm between $g_i|_{B,\mathbf{b}}$ and $f_{i'}|_{B,\mathbf{b}}$ for all $i, i' \in [s]$. Thus by fixing the j-th coordinate appropriately we get $f_i|_{B,\mathbf{b}'}$. □

Note that in the above lemma, as we move from $\mathbf{b}$ to $\mathbf{b}'$ we can match the clusters of $f|_{B,\mathbf{b}}$ to the corresponding clusters of $f|_{B,\mathbf{b}'}$. In what follows, we'll implicitly use the fact that we can find this matching.

The next step shows how to evaluate the clusters of $f|_{B,\mathbf{b}}$ for $\mathbf{b}$ that is arbitrarily far from $\mathbf{a}$ but satisfies a technical condition. For $0 \leq i \leq n$, let $\gamma_i(\mathbf{a}, \mathbf{b})$ denote the i-th "hybrid" between $\mathbf{a}$ and $\mathbf{b}$, i.e., a vector whose first $(n-i)$ coordinates are the first $(n-i)$ coordinates of $\mathbf{a}$ and the whose last i coordinates are the last i coordinates of $\mathbf{b}$, so that $\gamma_0(\mathbf{a}, \mathbf{b}) = \mathbf{a}$, $\gamma_n(\mathbf{a}, \mathbf{b}) = \mathbf{b}$ and the Hamming distance between $\gamma_i(\mathbf{a}, \mathbf{b})$ and $\gamma_{i+1}(\mathbf{a}, \mathbf{b})$ is 1.

Lemma 8.5 (Similar to Corollary 6.18 in [BSV21])**.** *Let $f \in \Sigma^k\Pi\Sigma$ be a multilinear polynomial and let C be a minimal multilinear $\Sigma^k\Pi\Sigma$ circuit computing f. Let $(B, \mathbf{a})$ be the output of Algorithm 3 on f. Let $\mathbf{b} \in \mathbb{F}^n$ such that $\Theta_{B,C}(\gamma_i(\mathbf{a}, \mathbf{b})) \neq 0$ for all $0 \leq i \leq n-1$, and let $f|_{B,\mathbf{a}} = \sum_{i=1}^{s} f_i|_{B,\mathbf{a}}$ be the output of Algorithm 2 on $f|_{B,\mathbf{a}}$.*

Then, there exists an algorithm that runs in time $2^{k^2} \cdot \mathsf{poly}(n)$ and outputs $(f_1|_{B,\mathbf{b}}, \ldots, f_s|_{B,\mathbf{b}})$.

Proof. We apply Lemma 8.4 repeatedly on $\mathbf{a} = \gamma_0(\mathbf{a}, \mathbf{b}), \gamma_1(\mathbf{a}, \mathbf{b}), \ldots, \gamma_n(\mathbf{a}, \mathbf{b}) = \mathbf{b}$. □

Finally, we show how to evaluate the clusters on *arbitrary* $\mathbf{b} \in \mathbb{F}^n$. To do this, we consider, as in [BSV21], the line $\ell_{\mathbf{a},\mathbf{b}}(t)$ through $\mathbf{a}$ and $\mathbf{b}$ and show that most points on the line are non-zeros of the polynomial $\Phi_{B,C}$. For each such "good" point we can recover the clusters, and then apply the Berlekamp-Welch algorithm in order to recover the clusters of $f|_{B,\ell_{\mathbf{a},\mathbf{b}}(t)}$, for every t.

Lemma 8.6 (Similar to Lemma 6.19 in [BSV21]). *Let $f \in \Sigma^k\Pi\Sigma$ be a multilinear polynomial and let C be a $\Sigma^k\Pi\Sigma$ circuit computing f. Let $(B, \mathbf{a})$ be good outputs of Algorithm 3 on f. Let $f|_{B,\mathbf{a}} = \sum_{i=1}^{s} f_i|_{B,\mathbf{a}}$ be the output of Algorithm 2 on $f|_{B,\mathbf{a}}$.*

Then, there exists an algorithm that, given any $\mathbf{b} \in \mathbb{F}^n$, runs in time $2^{k^2} \cdot \mathsf{poly}(n)$ and outputs $(f_1|_{B,\mathbf{b}}, \ldots, f_s|_{B,\mathbf{b}})$.

Proof. Let $\ell_{\mathbf{a},\mathbf{b}}(t)$ be the line through $\mathbf{a}$ and $\mathbf{b}$ so that $\ell_{\mathbf{a},\mathbf{b}}(0) = \mathbf{a}$ and $\ell_{\mathbf{a},\mathbf{b}}(1) = \mathbf{b}$. Let $W \subseteq \mathbb{F}$ be a set of size $10n^9$.

For each $u \in W$, let $\mathbf{b}_u = \ell_{\mathbf{a},\mathbf{b}}(u)$. Use the algorithm in Lemma 8.5 to learn $(f_1|_{B,\mathbf{b}_u}, \ldots, f_s|_{B,\mathbf{b}_u})$. Note that $f_i(\mathbf{b}_u) = f_i|_{B,\mathbf{b}_u}(\mathbf{b}_u) = f_i(\ell_{\mathbf{a},\mathbf{b}}(u))$.

For each $i \in [s]$ we apply the Berlekamp-Welch algorithm on the points $f_i(\ell_{\mathbf{a},\mathbf{b}}(u))$, for $u \in W$, to recover the univariate polynomial $f_i(\ell_{\mathbf{a},\mathbf{b}}(t))$, and output the value $f_i(\ell_{\mathbf{a},\mathbf{b}}(1))$, which equals $f_i|_{B,\mathbf{b}}(\mathbf{b}) = f_i(\mathbf{b})$.

To prove that this indeed works, we need to show that there are many values of u for which the conditions of Lemma 8.5 hold and thus the computation is correct, so in particular the Berlekamp-Welch algorithm returns the correct polynomial.

Let $Q(t) = \prod_{i=1}^{n-1} \Theta_{B,C}(\gamma_i(\mathbf{a}, \ell_{\mathbf{a},\mathbf{b}}(t)))$. Q is a non-zero polynomial since $Q(0) = \Theta_{B,C}(\mathbf{a})^n \neq 0$. Further, Q has degree at most $n \cdot \deg(\Theta_{B,C}) \leq 2n^8$ and if $Q(u) \neq 0$, then $\mathbf{b}_u = \ell_{\mathbf{a},\mathbf{b}}(u)$ satisfies the condition of Lemma 8.5. The number of roots of Q is thus at most $2n^8$, which bounds the number of errors for the Berlekamp-Welch algorithm. Thus, since the number of evaluations $|W| > 4n^8 + \deg(f_i) + 1$, we can indeed recover the polynomial $f_i(\ell_{\mathbf{a},\mathbf{b}}(t))$ correctly. □

8.2 The Reconstruction Algorithm

We now give our reconstruction algorithm for multilinear $\Sigma^k\Pi\Sigma$ circuits.

Algorithm 4 : Reconstruction of $\Sigma^k\Pi\Sigma$ circuits

```
Input: Black box access to a degree $d$, $n$-variate multilinear polynomial $f$ computed by a
    minimal multilinear $\Sigma^k\Pi\Sigma$ circuit, $C$.
for every $R_M(2k) \leq \tau_0 \leq R_M(2k)^{k^{2k}}$ do
    Run Algorithm 3 with parameter $\tau = \tau_0^k$ on $f$ to obtain a cluster preserving pair
    $(B, \mathbf{a})$ where $|B| \leq k^{k^{O(k)}}$
    Denote by $(f_1|_{B,\mathbf{a}}, \ldots, f_s|_{B,\mathbf{a}})$ the output of Algorithm 2 on $f|_{B,\mathbf{a}}$ with parameter $\tau$
    for $i \in [s]$ do
        Use Lemma 6.4, along with Lemma 8.6 to simulate black-box access to $f_i|_{B,\mathbf{a}}$, to
    learn a circuit $\tilde{C}_i$.
    Let $\tilde{C} = \tilde{C}_1 + \ldots + \tilde{C}_{s'}$.
    if $[\tilde{C}] \equiv [C]$ (which can be checked using a randomized PIT algorithm) then
        Output $\tilde{C}$.
```

Theorem 8.7. *Suppose $|\mathbb{F}| > n^{k^{k^{O(k)}}}$. Let f be a polynomial computed by a multilinear $\Sigma^k\Pi\Sigma$ circuit. Then, with high probability, Algorithm 4 returns a multilinear $\Sigma^k\Pi\Sigma$ circuit*

$\tilde{C}$ *computing* f *in time* $\mathsf{poly}(n) \cdot k^{k^{k^{k^{\mathsf{poly}(k)}}}}$.

Proof. Let f be as given in the theorem. Consider a partition of f, $f = \sum_{i=1}^{s} f_i$, as given by Claim 5.24 (for $\varphi(k) = k^2$) with parameters τ_0, τ_1. As the algorithm scans all possible values in Line 1, it will find the "correct" value of τ_0.

Focusing on this τ_0, the assumption on the field size and Claim 7.2 guarantee that with high probability $(B, \mathbf{a})$ will be a good output of Algorithm 3. Claim 7.1 promises that Algorithm 3 indeed outputs B of the required size such that the output of Algorithm 2 on $f|_{B,\mathbf{a}}$ are the clusters $(f_1|_{B,\mathbf{a}}, \ldots, f_s|_{B,\mathbf{a}})$.

Lemma 8.6 now guarantees that we can evaluate each f_i correctly on each $\mathbf{b} \in \mathbb{F}^n$ so that we can use the algorithm from Lemma 6.4 to learn circuits computing each f_i. Thus, we are able to reconstruct a circuit computing f and pass the check in Line 7.

The bound on the running time follows from the bound on the running time of Algorithm 3 given in Claim 7.1 and from the running time given in Lemma 6.4 (see also Remark 7.8 for a remark on the running time of Algorithm 3). □

8.3 Proper Learning of Depth-3 Set-Multilinear Circuits

We now explain the changes required in Algorithm 4 in order to prove Theorem 1.3. The main change is to replace each application of Lemma 6.4 with Lemma 6.6. In particular, this makes sure that in Step 6 of Algorithm 3 we always find subcircuits of C that are *set-multilinear*, and not merely multilinear (note that by fixing the variables in the original circuit, we know that the restricted polynomial has a small set-multilinear circuit, and thus the algorithm can find it).

References

[AGKS15] Manindra Agrawal, Rohit Gurjar, Arpita Korwar, and Nitin Saxena. Hitting-Sets for ROABP and Sum of Set-Multilinear Circuits. *SIAM Journal of Computing*, 44(3):669–697, 2015. Pre-print available at arXiv:1406.7535.

[AV08] Manindra 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 2008)*, pages 67–75, 2008. Pre-print available at eccc:TR08-062.

[BIJL18] Markus Bläser, Christian Ikenmeyer, Gorav Jindal, and Vladimir Lysikov. Generalized matrix completion and algebraic natural proofs. In *Proceedings of the 50th Annual ACM Symposium on Theory of Computing (STOC 2018)*, pages 1193–1206. ACM, 2018.

[Bsh13] Nader H. Bshouty. Exact Learning from Membership Queries: Some Techniques, Results and New Directions. In *Algorithmic Learning Theory - 24th International Conference, ALT 2013*, volume 8139 of *Lecture Notes in Computer Science*, pages 33–52. Springer, 2013.

[BSV20] Vishwas Bhargava, Shubhangi Saraf, and Ilya Volkovich. Reconstruction of Depth-4 Multilinear Circuits. In *Proceedings of the 31st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2020)*, pages 2144–2160. SIAM, 2020.

[BSV21] Vishwas Bhargava, Shubhangi Saraf, and Ilya Volkovich. Reconstruction algorithms for low-rank tensors and depth-3 multilinear circuits. In *Proceedings of the 53rd Annual ACM Symposium on Theory of Computing (STOC 2021)*, pages 809–822. ACM, 2021.

[BT88] Michael Ben-Or and Prasoon Tiwari. A Deterministic Algorithm for Sparse Multivariate Polynominal Interpolation (Extended Abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC 1988)*, pages 301–309. ACM, 1988.

[Car06] Enrico Carlini. Reducing the number of variables of a polynomial. In *Algebraic Geometry and Geometric Modeling*, pages 237–247, 2006.

[CLO07] David A. Cox, John B. Little, and Donal O'Shea. *Ideals, Varieties and Algorithms*. Undergraduate texts in mathematics. Springer, 2007.

[DS07] Zeev Dvir and Amir Shpilka. Locally Decodable Codes with Two Queries and Polynomial Identity Testing for Depth 3 Circuits. *SIAM J. Comput.*, 36(5):1404–1434, 2007. Preliminary version in the *37th Annual ACM Symposium on Theory of Computing (STOC 2005)*.

[FGS18] Michael A. Forbes, Sumanta Ghosh, and Nitin Saxena. Towards Blackbox Identity Testing of Log-Variate Circuits. In *Preliminary version in the* 45th International Colloquium on Automata, Languages and Programming (ICALP 2018), volume 107 of *LIPIcs*, pages 54:1–54:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[FS13] Michael A. Forbes and Amir Shpilka. Quasipolynomial-Time Identity Testing of Non-commutative and Read-Once Oblivious Algebraic Branching Programs. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2013)*, pages 243–252, 2013. Full version at `arXiv:1209.2408`.

[FSS14] Michael A. Forbes, Ramprasad Saptharishi, and Amir Shpilka. Hitting sets for multilinear read-once algebraic branching programs, in any order. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC 2014)*, pages 867–875, 2014.

[GG20] Zeyu Guo and Rohit Gurjar. Improved Explicit Hitting-Sets for ROABPs. In *Proceedings of the 24th International Workshop on Randomization and Computation (RANDOM 2020)*, volume 176 of *LIPIcs*, pages 4:1–4:16, 2020.

[GKKS16] Ankit Gupta, Pritish Kamath, Neeraj Kayal, and Ramprasad Saptharishi. Arithmetic Circuits: A Chasm at Depth 3. *SIAM J. Comput.*, 45(3):1064–1079, 2016.

[GKL11] Ankit Gupta, Neeraj Kayal, and Satyanarayana V. Lokam. Efficient Reconstruction of Random Multilinear Formulas. In *Proceedings of the 52nd*

Annual IEEE Symposium on Foundations of Computer Science (FOCS 2011), pages 778–787. IEEE Computer Society, 2011.

[GKL12] Ankit Gupta, Neeraj Kayal, and Satyanarayana V. Lokam. Reconstruction of depth-4 multilinear circuits with top fan-in 2. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC 2012)*, pages 625–642. ACM, 2012.

[GKS20] Ankit Garg, Neeraj Kayal, and Chandan Saha. Learning sums of powers of low-degree polynomials in the non-degenerate case. In *Proceedings of the 61st Annual IEEE Symposium on Foundations of Computer Science (FOCS 2020)*, pages 889–899. IEEE, 2020.

[Hås90] Johan Håstad. Tensor Rank is NP-Complete. *J. Algorithms*, 11(4):644–654, 1990.

[Kay11] Neeraj Kayal. Efficient algorithms for some special cases of the polynomial equivalence problem. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2011)*, pages 1409–1421. SIAM, 2011.

[KMSV13] Zohar Shay Karnin, Partha Mukhopadhyay, Amir Shpilka, and Ilya Volkovich. Deterministic Identity Testing of Depth-4 Multilinear Circuits with Bounded Top Fan-in. *SIAM J. Comput.*, 42(6):2114–2131, 2013.

[KNS19] Neeraj Kayal, Vineet Nair, and Chandan Saha. Average-case linear matrix factorization and reconstruction of low width algebraic branching programs. *Comput. Complex.*, 28(4):749–828, 2019.

[Koi12] Pascal Koiran. Arithmetic Circuits: The Chasm at Depth Four Gets Wider. *Theoretical Computer Science*, 448:56–65, 2012. Pre-print available at `arXiv:1006.4700`.

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

[KS08] Zohar Shay Karnin and Amir Shpilka. Black Box Polynomial Identity Testing of Generalized Depth-3 Arithmetic Circuits with Bounded Top Fan-In. In *Proceedings of the 23rd Annual IEEE Conference on Computational Complexity, CCC 2008, 23-26 June 2008, College Park, Maryland, USA*, pages 280–291. IEEE Computer Society, 2008.

[KS09a] Zohar Shay Karnin and Amir 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 2009)*, pages 274–285. IEEE Computer Society, 2009.

[KS09b] Neeraj Kayal and Shubhangi Saraf. Blackbox polynomial identity testing for depth-3 circuits. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2009)*, 2009.

[KT90] Erich Kaltofen and Barry M. Trager. Computing with Polynomials Given By Black Boxes for Their Evaluations: Greatest Common Divisors, Factorization, Separation of Numerators and Denominators. *J. Symb. Comput.*, 9(3):301–320, 1990.

[Shi16] Yaroslav Shitov. How hard is the tensor rank? *arXiv preprint arXiv:1611.01559*, 2016.

[Sin16] Gaurav Sinha. Reconstruction of Real Depth-3 Circuits with Top Fan-In 2. In *Proceedings of the 31st Annual Computational Complexity Conference (CCC 2016)*, volume 50 of *LIPIcs*, pages 31:1–31:53, 2016.

[Sin22] Gaurav Sinha. Efficient Reconstruction of Depth Three Arithmetic Circuits with Top Fan-In Two. In *Proceedings of the 13th Innovations in Theoretical Computer Science Conference (ICTS 2022)*, volume 215 of *LIPIcs*, pages 118:1–118:33. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[SS11] Nitin Saxena and C. Seshadhri. An Almost Optimal Rank Bound for Depth-3 Identities. *SIAM J. Comput.*, 40(1):200–224, 2011.

[SS12] Nitin 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. Preliminary version in the *43rd Annual ACM Symposium on Theory of Computing (STOC 2011)*.

[SS13] Nitin Saxena and C. Seshadhri. From sylvester-gallai configurations to rank bounds: Improved blackbox identity test for depth-3 circuits. *J. ACM*, 60(5):33:1–33:33, 2013.

[Swe18] Joseph Swernofsky. Tensor Rank is Hard to Approximate. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2018*, volume 116 of *LIPIcs*, pages 26:1–26:9, 2018.

[SY10] Amir Shpilka and Amir Yehudayoff. Arithmetic Circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5:207–388, March 2010.

[Tav15] Sébastien Tavenas. Improved bounds for reduction to depth 4 and depth 3. *Inf. Comput.*, 240:2–11, 2015. Preliminary version in the *38th International Symposium on the Mathematical Foundations of Computer Science (MFCS 2013)*.