Reconstruction of Generalized Depth-3 Arithmetic Circuits with Bounded Top Fan-in

Zohar S. Karnin* Amir Shpilka*

Abstract

In this paper we give reconstruction algorithms for depth-3 arithmetic circuits with k multiplication gates (also known as $\Sigma\Pi\Sigma(k)$ circuits), where $k = O(1)$. Namely, we give an algorithm that when given a black box holding a $\Sigma\Pi\Sigma(k)$ circuit C over a field $\mathbb{F}$ as input, makes queries to the black box (possibly over a polynomial sized extension field of $\mathbb{F}$) and outputs a circuit C' computing the same polynomial as C. In particular we obtain the following results.

1. When C is a *multilinear* $\Sigma\Pi\Sigma(k)$ circuit (i.e. each of its multiplication gates computes a multilinear polynomial) then our algorithm runs in polynomial time (when k is a constant) and outputs a multilinear $\Sigma\Pi\Sigma(k)$ circuits computing the same polynomial.
2. In the general case, our algorithm runs in quasi-polynomial time and outputs a *generalized* depth-3 circuit (a notion that is defined in the paper) with k multiplication gates. For example, the polynomials computed by generalized depth-3 circuits can be computed by quasi-polynomial sized depth-3 circuits. In fact, our algorithm works in the slightly more general case where the black box holds a generalized depth-3 circuits.

Prior to this work there were reconstruction algorithms for several different models of bounded depth circuits: the well studied class of depth-2 arithmetic circuits (that compute sparse polynomials) and its close by model of depth-3 *set-multilinear* circuits. For the class of depth-3 circuits only the case of $k = 2$ (i.e. $\Sigma\Pi\Sigma(2)$ circuits) was known.

Our proof technique combines ideas from [Shp] and [KS08] with some new ideas. Our most notable new ideas are: We prove the existence of a unique *canonical representation* of depth-3 circuits. This enables us to work with a specific representation in mind. Another technical contribution is an *isolation lemma* for depth-3 circuits that enables us to reconstruct a single multiplication gate of the circuit.

*Faculty of Computer Science, Technion, Haifa 32000, Israel. Email: {zkarnin,shpilka}@cs.technion.ac.il. Research supported by the Israel Science Foundation (grant number 439/06).

Contents

List of Algorithms

1 Introduction

In this work we consider the problem of reconstruction an arithmetic circuit for which we have oracle access: We are given a black box holding an arithmetic circuit C belonging to some box holding an arithmetic circuit C belonging to some predetermined class of circuits, and we wish to construct a circuit C' that computes the same polynomial as C. Our only access to the polynomial computed by C is via black-box queries. That is, we are allowed to pick inputs (adaptively) and query the black box for the value of C on those inputs. The problem of reconstructing arithmetic circuits using queries is an important question in algebraic complexity that received a lot of attention. As the problem is notoriously difficult, research focused on restricted models such as the model of depth-2 arithmetic circuits (circuits computing sparse polynomials) that was studied over various fields (see e.g. [BOT88, KS01] and the references within), the model of read-once arithmetic formulas [HH91, BHH95, BB98, SV08, SV09] and the model of *set-multilinear* depth-3 circuits [BBB$^+$00, KS06a].

The focus of this work is on the class of depth-3 arithmetic circuits that have a bounded number of multiplication gates, over some finite field $\mathbb{F}$. We give two different algorithms for this model. The first is a polynomial time reconstruction algorithm for the case that the circuits is a *multilinear* circuit. In the general case, without the multilinearity assumption, we give a quasi-polynomial time reconstruction algorithm. These results extend and generalize a previous work by the second author that gave reconstruction algorithms for depth-3 circuit with only 2 multiplication gates [Shp].

1.1 Depth-3 circuits

In this paper we study depth-3 arithmetic circuits. We only consider circuits with a + gate at the top. Notice that for depth-3 circuits with a multiplication gate at the top, reconstruction is possible using known factoring algorithms [Kal85, KT90, Kal95]. A depth-3 circuit with a plus gate at the top is also known as a $\Sigma\Pi\Sigma$ circuit and has the following structure:

$$\mathcal{C} = \sum_{i=1}^{k} M_i = \sum_{i=1}^{k} \prod_{j=1}^{d_i} L_{i,j}(x_1, \ldots, x_n), \tag{1}$$

where the $L_{i,j}$-s are linear functions. The $\{M_i\}_{i=1}^k$ are called the multiplication gates of $\mathcal{C}$ and k is the fan-in of the top plus gate. We call a depth-3 circuit with a top fan-in k a $\Sigma\Pi\Sigma(k)$ circuit.

Although depth-3 circuits seems to be a very restricted model of computation, understanding it is one of the greatest challenges in arithmetic circuit complexity. It is the first model for which lower bounds are difficult to prove. Over finite fields exponential lower bounds were obtained [GK98, GR00] but over fields of characteristic zero only quadratic lower bounds are known [SW01, Shp02]. Moreover, recently Agrawal and Vinay [AV08] showed[1] that exponential lower bounds for depth-4 circuits imply exponential lower bounds for general arithmetic circuits. In addition they showed that a polynomial time black-box polynomial identity testing algorithm for depth-4 circuits implies a (quasi-polynomial time) derandomization of polynomial identity testing of general arithmetic circuits. As for depth-2 circuits most of the questions are not very difficult, we see that depth-3 circuits stand between the relatively easy depth-2 case and the very difficult general case (depth-4). Hence it is an important goal to deeper understand the class of depth-3 circuits.

1.2 Statement of our results

We give two reconstruction algorithms. The first for general $\Sigma\Pi\Sigma(k)$ circuits and the second for multilinear $\Sigma\Pi\Sigma(k)$ circuits. While in the case of multilinear depth-3 circuits our algorithm returns a

[1] This result also follows from the earlier works of [VSBR83, Raz08].

multilinear $\Sigma\Pi\Sigma(k)$ circuit, in the general case we return what we call a generalized depth-3 circuit. We note however, that generalized depth-3 circuits can be presented as depth-3 circuits of quasi-polynomial size.

A polynomial $f(\bar{x})$ that is computable by a generalized depth-3 circuit has the following form

$$f(\bar{x}) = \sum_{i=1}^{k} M_i = \sum_{i=1}^{k} \left(\prod_{j=1}^{d_i} L_{i,j}(\bar{x}) \right) \cdot h_i \left(\tilde{L}_{i,1}(\bar{x}), \ldots, \tilde{L}_{i,\rho_i}(\bar{x}) \right) \tag{2}$$

where the $L_{i,j}$'s and the $\tilde{L}_{i,j}$'s are linear functions in the variables $\bar{x} = (x_1, \ldots, x_n)$, over $\mathbb{F}$. Every h_i is a polynomial in $\rho_i \leq \rho$ variables, and the functions $\{\tilde{L}_{i,j}\}_{j=1}^{\rho_i}$ are linearly independent. We shall assume, w.l.o.g., that each h_i depends on all its ρ_i variables. We call $M_1, \ldots, M_k$ the multiplication gates of the circuit ($M_i = \prod_{j=1}^{d_i} L_{i,j} \cdot h_i$). The degree of the circuit, $d = \deg(C)$, is defined as the maximal degree of its multiplication gates (i.e. $d = \max_{i=1\ldots k}\{\deg(M_i)\}$). A degree d generalized depth-3 circuit with k multiplication gates is called a $\Sigma\Pi\Sigma(k, d, \rho)$ circuit. The size of a $\Sigma\Pi\Sigma(k, d, \rho)$ circuit is defined as the sum of degrees of the multiplication gates of the circuit (thus, the size of the circuit given by Equation (2) is $\sum_{i=1}^{k} \deg(M_i) \leq dk$). We denote the size of a circuit C by size(C). When $\rho = 0$ (i.e. each h_i is a constant function) we get the class of depth-3 circuits with k multiplication gates and degree d, also known as $\Sigma\Pi\Sigma(k, d)$ circuits (similarly we have the class $\Sigma\Pi\Sigma(k)$ circuits of depth-3 circuits with k multiplication gates). When k and d are arbitrary we get the class of depth-3 circuits that we denote with $\Sigma\Pi\Sigma$.

Notice that the difference between generalized $\Sigma\Pi\Sigma(k)$ circuits and $\Sigma\Pi\Sigma(k)$ circuits is the non linear part h_i of the multiplication gates. Also note that $\Sigma\Pi\Sigma(k, d, \rho)$ circuits can be simulated by depth-3 circuits with $k \cdot d^{\rho}$ multiplication gates. Thus, when the ρ_i-s are polylogarithmic in n (as in this paper) then a $\Sigma\Pi\Sigma(k, d, \rho)$ circuit can be simulated by a depth-3 circuit with quasi-polynomially many gates.

Our result for reconstructing general $\Sigma\Pi\Sigma(k)$ circuits actually holds for generalized depth-3 circuits. Namely, our reconstruction algorithm, when given black-box access to a $\Sigma\Pi\Sigma(k, d, \rho)$ circuit, returns a $\Sigma\Pi\Sigma(k, d, \rho')$ circuit computing the same polynomial. Therefore, instead of stating our result for $\Sigma\Pi\Sigma(k)$ circuits we state them in their most general form.

Theorem 1. *Let f be an n-variate polynomial computed by a $\Sigma\Pi\Sigma(k, d, \rho)$, over a field $\mathbb{F}$. Then there is a reconstruction algorithm for f, that runs in time* $\mathrm{poly}(n) \cdot \exp(\log(|\mathbb{F}|) \cdot \log(d)^{O(k^3)} \cdot \rho^{O(k)})$*, and outputs a $\Sigma\Pi\Sigma(k, d, \rho')$ circuit for f (ρ' is equal to ρ up to an additive polylogarithmic factor). When $|\mathbb{F}| = O(d^5)$ the algorithm is allowed to make queries to f from a polynomial size algebraic extension field of $\mathbb{F}$.*

Our second result deals with the case of multilinear $\Sigma\Pi\Sigma(k)$ circuits. A multilinear circuit is a circuit in which every multiplication gate computes a multilinear polynomial.

Theorem 2. *Let f be a multilinear polynomial in n variables that is computed by a multilinear $\Sigma\Pi\Sigma(k)$ circuit, over a field $\mathbb{F}$. Then there is a reconstruction algorithm for f that runs in time $(n+|\mathbb{F}|)^{2^{O(k \log k)}}$ and outputs a multilinear $\Sigma\Pi\Sigma(k)$ circuits computing f. When $|\mathbb{F}| < n^5$ The algorithm is allowed to make queries to f from a polynomial size algebraic extension field of $\mathbb{F}$.*

Thus, for the multilinear case we give a reconstruction algorithm that also returns a multilinear circuit of the same complexity.

1.3 Related works

We are aware of two reconstruction algorithms for restricted depth-3 circuits. The first by [BBB+00, KS06a] gives a reconstruction algorithm for *set-multilinear* depth-3 circuits. Although being depth-3 circuits, set-multilinear circuits are closer in nature to depth-2 circuits and in fact it is better to think of them as a generalization of depth-2 circuits. In particular the techniques used to reconstruct set multilinear circuits also work when the input is a depth-2 circuit but completely fail even for the case of $\Sigma\Pi\Sigma(2)$ multilinear circuits (see [Shp] for an explanation). The second related result is a reconstruction algorithm for $\Sigma\Pi\Sigma(2)$ circuits [Shp]. Our work uses ideas from this earlier work together with some new techniques to get the more general result.

Another line of results that is related to this paper is works on algorithms for polynomial identity testing (PIT for short) of depth-3 circuits. In [DS06] a quasi-polynomial time non black-box PIT algorithm was given to $\Sigma\Pi\Sigma(k)$ circuits (for a constant k). This result was later improved by [KS07] that gave a polynomial time algorithm for $\Sigma\Pi\Sigma(k)$ circuits (again for a constant k). In [KS08] we managed to generalize the results of [DS06] to the case of *black-box* algorithms. Recently, [SS08] managed to improve a theorem of [DS06] regarding the rank of identically zero depth-3 circuits, which immediately improves the previous black-box PIT algorithm for the problem. Unfortunately, even after this improvement, the running time of the algorithm is quasi-polynomial in n. Thus, no polynomial time algorithm is known in the black-box case. We note that black-box algorithms for the polynomial identity problem are closely related to reconstruction algorithms as they give a set of inputs such that the value of the circuit on those inputs completely determine the circuit and thus it gives sufficient information for reconstructing the circuit. The main problem of course is coming up with an efficient algorithm that will use this information to reconstruct the circuit. Such algorithms are known for depth-2 circuits where several PIT algorithms were used for reconstruction (see e.g. [KS01] and the references within). We are unaware of any generic way of doing so (namely, moving from PIT to reconstruction). This is exactly the reason that the black-box PIT algorithms of [KS01] for $\Sigma\Pi\Sigma(2)$ circuits and of [KS08] for $\Sigma\Pi\Sigma(k)$ circuits do not immediately imply reconstruction algorithms.

The problem of reconstructing arithmetic circuits is also closely related to the problem of learning arithmetic circuits. In the learning scenario the focus is on learning arithmetic circuits over small fields (usually over $\mathbb{F}_2$). Moreover, we are usually satisfied with outputting an approximation to the unknown arithmetic circuit. For example, it is not difficult to see that the learning algorithm of Kushilevitz and Mansour [KM93] can learn $\Sigma\Pi\Sigma(k)$ circuits over constant sized fields, for $k = O(\log n)$, in polynomial time. However, the model that we are considering is different. We are interested in larger fields, in particular we allow queries from extension fields, and we want to output the exact same polynomial. This difference is prominent when considering interpolation (i.e. reconstruction) algorithms, see e.g. the discussion in [KS01].

Other related works include hardness results for learning arithmetic circuits. In [FK06] Fortnow and Klivans showed that polynomial time reconstruction of arithmetic circuits implies a lower bound for the class $\mathrm{ZPEXP}^{\mathrm{RP}}$. However, for $\Sigma\Pi\Sigma(k)$ circuit this does not give much as it is easy to think of polynomials that are not computed in this model. Another related result was given in [KS06b] where Klivans and Sherstov showed hardness result for PAC learning depth-3 arithmetic circuits. Namely, they show that there is no efficient PAC learning algorithm for depth-3 arithmetic circuits that works for every distribution. It is unclear however whether this result can be extended to show a hardness result for a membership-query algorithm over the uniform distribution (like the model under consideration here). Indeed, for the reconstruction problem of arithmetic circuits the uniform distribution is the most relevant model. Thus, although we believe that the problem of reconstructing general depth-3 circuits is difficult we do not have sufficiently strong hardness results to support that feeling. It is an interesting question to prove (or disprove) it.

1.4 Our Techniques

Our algorithms use the intuition behind the construction of the test-sets of [KS08] (that give a black-box PIT algorithm) although here we need to evaluate the circuit at a much larger set of points. The scheme of the algorithm is similar in nature to the algorithm of [Shp], however it is much more complicated and requires several new ideas that we now outline. Conceptually we have two main new ideas. The first is the notion of a canonical circuit for depth-3 circuits and the second is an isolation lemma. We now shortly discuss each of the ideas.

Canonical κ-distant circuits: An important ingredient of our proof is the definition of a canonical κ-distant circuit for a polynomial f computed by a (generalized) circuit. The canonical circuit is also a (generalized) depth-3 circuit computing f, and it is defined uniquely (see Corollary 3.7). The advantage in this definition is that as we know that the canonical circuit is unique then if we manage to compute its restriction to some low dimensional subspace then we can "lift" each multiplication gate separately to the whole space and get the canonical circuit for f. In other words, if $C = M_1 + \ldots + M_k$ is the canonical circuit for f, then $M_1|_V + \ldots + M_k|_V$ is the canonical circuit for $f|_V$, when V is a *rank-preserving* subspace (as defined in [KS08], see Section 2.2). An important step towards the definition of a canonical circuit is the definition of a distance function between multiplication gates (see Section 2.3). We use the distance function in order to cluster multiplication gates together to form a new (generalized) multiplication gate. As a result any two new multiplication gates are far from each other. This robust structure allows us to deal with each multiplication gate separately.

Isolation Lemma: The second new ingredient in our proof is a new isolation lemma that basically shows that for any canonical circuit $C = M_1 + \ldots + M_k$ there exists an index $1 \leq m \leq k$ and a set of subspaces $\mathcal{V} = \{U_i\}$ such such that $M_j|_{U_i} = 0$, for every $j \neq m$ and subspace U_i. Moreover, it is possible to reconstruct M_m from the set of circuits $\{M_m|_{U_i}\}$ (we discuss this in more details in section 4). The proof of the existence of such a structure is very technical and it relies on a generalization of a technical lemma of [DS06].

Finally we also require an idea that appeared in our previous work [KS08]. There we defined the notion of rank-preserving subspaces and used it to give a deterministic sub-exponential black-box PIT algorithm for $\Sigma\Pi\Sigma(k, d, \rho)$ circuit. Here we show that rank-preserving subspaces can be used to derandomize the reconstruction algorithm of [Shp]. In particular this makes our algorithm deterministic whereas the algorithm of [Shp] was randomized.

Given these tools we manage to follow the algorithmic scheme laid out in [Shp]. The sketch of the scheme is: First we restrict our inputs to a rank-preserving subspace V. Then, using the isolation lemma we reconstruct the multiplication gates of the canonical circuit of $f|_V$. After that we further reconstruct $f|_{V_i}$ (for $i \in [n]$), where V is of co-dimension 1 inside each V_i and $\text{span}\,(\cup_{i=1}^n V_i) = \mathbb{F}^n$. Then we use the uniqueness of the canonical circuit for f to "glue" the different circuits $\{f|_{V_i}\}$ to get a single circuit for f.

While this is the general scheme there are a few differences between the multilinear and the general case. In the multilinear case the difficult part is lifting the circuit. It turns out that, unlike the general case, uniqueness is not guaranteed however we show that if the rank-preserving subspace V has several additional properties, then basically any lift is good. In contrary, for the general case the bottleneck lies in reconstructing the circuit $C|_V$. This is the place where we need to apply the isolation lemma.

A more detailed overview of the algorithm for the different cases can be found in Section 4.

1.5 Organization

The paper is organized as follows. In Section 2 We give some definitions and discuss some easy properties of restrictions of linear functions to affine subspaces. We then describe the results of [DS06] and [SS08] regarding identically zero depth-3 circuits. We also discuss the results of [KS08] regarding a deterministic PIT algorithm for depth-3 circuits. Specifically we present the notion of rank-preserving subspaces as given in [KS08]. In Section 3 we define the notion of κ-distant $\Sigma\Pi\Sigma(k,d,r)$ circuit and prove the existence and uniqueness of such a circuit. In Section 4 we give the proof of Theorem 1 and in Section 5 we give the proof of Theorem 2.

2 Preliminaries

For a positive integer k we denote $[k] = \{1,\ldots,k\}$. A partition of a set S is a set of nonempty subsets of S such that every element of S is in exactly one of these subsets. Let $\mathbb{F}$ be a field. We denote with $\mathbb{F}^n$ the n'th dimensional vector space over $\mathbb{F}$. We shall use the notation $\bar{x} = (x_1,\ldots,x_n)$ to denote the vector of n indeterminates. For two linear functions L_1, L_2 we write $L_1 \sim L_2$ or alternatively say that L_1 and L_2 are equivalent whenever L_1 and L_2 are linearly dependent (that is, for some $\alpha,\beta \in \mathbb{F}$, $\alpha L_1 + \beta L_2 = 0$).

Let $V = V_0 + v_0 \subseteq \mathbb{F}^n$ be an affine subspace, where $v_0 \in \mathbb{F}^n$ and $V_0 \subseteq \mathbb{F}^n$ is a linear subspace. Let $L(\bar{x})$ be a linear function. We denote with $L|_V$ the restriction of L to V. Assume the dimension of V_0 is t, then $L|_V$ can be viewed as a linear function of t variables in the following way: Let $\{v_i\}_{i\in[t]}$ be a basis for V_0. For $v \in V$ let $v = \sum_{i=1}^{t} x_i \cdot v_i + v_0$ be its representation according to the basis. We get that

$$L(v) = \sum_{i=1}^{t} x_i \cdot L(v_i) + L(v_0) \triangleq L|_V(x_1,\ldots,x_t),$$

in order for $L|_V(x_1,\ldots,x_t)$ to be well defined, $\{v_i\}_{i\in[t]}$ should be chosen in some unique way. We give a definition for a "default" basis later. A linear function L will sometimes be viewed as a vector of $n+1$ entries. Namely, the function $L(x_1,\ldots,x_n) = \sum_{i=1}^{n} \alpha_i \cdot x_i + \alpha_0$ corresponds to the vector of coefficients $(\alpha_0,\alpha_1,\ldots,\alpha_n)$. Accordingly, we define the span of a set of linear functions of n variables as the span of the corresponding vectors (i.e. as a subspace of $\mathbb{F}^{n+1}$). For an affine subspace V of dimension t, the linear function $L|_V$ can be viewed as a vector of $t+1$ entries. Thus, V defines a linear transformation from $\mathbb{F}^{n+1}$ to $\mathbb{F}^{t+1}$. For example, let $V = V_0 + v_0$ be as above, and $\{v_i\}_{i\in[t]}$ be a basis for V_0. Let A be the $t+1 \times n+1$ matrix whose i'th row, for $1 \le i \le t$, is v_i and its $t+1$'th row is v_0. Then A represents the required linear transformation. Let $L_1,\ldots,L_m$ be a set of linear functions. We define the span of these linear functions along with 1 (i.e., the constant function) as $\mathrm{span}_1(L_1,\ldots,L_m)$. For a subspace $\mathcal{L}$ of linear functions, we define a measure of its dimension "modolu 1" as the dimension of the subspace obtained by taking the homogenous part of its linear functions. We denote the "modolu 1" dimension measure by $\dim_1(\mathcal{L})$. For convenience we say that $L_1,\ldots,L_m$ are linearly independent^H to indicate that their homogenous parts are linearly independent.

2.1 Generalized Depth 3 Arithmetic Circuits

We first recall the usual definition of depth-3 circuits. A depth-3 circuit with k multiplication gates of degree d has the following form:

$$C = \sum_{i=1}^{k} M_i = \sum_{i=1}^{k} \prod_{j=1}^{d_i} L_{i,j}(x_1,\ldots,x_n) \tag{3}$$

where each $L_{i,j}$ is a linear function in the input variables and $d = \max_{i=1\ldots k}\{\deg(M_i)\}$. Recall that we defined a $\Sigma\Pi\Sigma(k,d,\rho)$ circuit (see Equation 2) to be a circuit of the form

$$C = \sum_{i=1}^{k} M_i = \sum_{i=1}^{k} \left(\prod_{j=1}^{d_i} L_{i,j}(\bar{x}) \right) \cdot h_i \left(\tilde{L}_{i,1}(\bar{x}), \ldots, \tilde{L}_{i,\rho_i}(\bar{x}) \right) \tag{4}$$

We thus see that in a generalized depth-3 circuit multiplication gates can have an additional term that is a polynomial that depends on at most ρ linear functions. For each M_i (as in Equation (4)), we assume w.l.o.g. that h_i has no linear factors. The following notions will be used throughout this paper.

Definition 2.1. *Let C be a $\Sigma\Pi\Sigma(k,d,\rho)$ arithmetic circuit that computes a polynomial as in Equation* (4)*.*

1. *For every multiplication gate M_i we define $\mathrm{Lin}(M_i) = \prod_{j=1}^{d_i} L_{i,j}(\bar{x})$ (we use the notations of Equation 4). That is, $\mathrm{Lin}(M_i)$ is the product of all the linear factors of M_i (recall that h_i has no linear factors). We call h_i the* non-linear term *of M_i.*

2. *For each $A \subseteq [k]$, we define $C_A(\bar{x})$ to be a sub-circuit of C as follows: $C_A(\bar{x}) = \sum_{i \in A} M_i(\bar{x})$.*

3. *Define $\gcd(C)$ as the product of all the non-constant linear functions that appear as factors in all the multiplication gates. I.e. $\gcd(C) = \gcd(\mathrm{Lin}(M_1), \ldots, \mathrm{Lin}(M_k))$. A circuit will be called simple if $\gcd(C) = 1$.*

4. *The simplification of C, $\mathrm{sim}(C)$, is defined as $\mathrm{sim}(C) \triangleq C/\gcd(C)$.*

5. *We define*
$$\mathrm{Lin}(C) \triangleq \{L_{i,j}\}_{i\in[k], j\in[d_i]} \cup \left(\cup_{i=1}^{k} \mathrm{span}_1 \{\tilde{L}_{i,j}\}_{j\in[\rho_i]} \right).$$
Notice that we take every linear function in the span of each $\{\tilde{L}_{i,j}\}_{j\in[\rho_i]}$ to be in $\mathrm{Lin}(C)$.

6. *We define $\mathrm{rank}(C)$ as the dimension of the span of the linear functions in C. That is,*
$$\mathrm{rank}(C) = \dim_1 \left(\mathrm{span}_1 \left(\{L_{i,j}\}_{i,j} \cup \{\tilde{L}_{i,j}\}_{i,j} \right) \right) = \dim_1(\mathrm{Lin}(C)).$$

A word of clarification is needed regarding the definition of $\mathrm{Lin}(C)$ and $\mathrm{rank}(C)$. Notice that the definition seems to depend on the specific choice of linear functions $\tilde{L}_{i,j}$. That is, it may be the case (and it is indeed the case) that every polynomial $h_i(\tilde{L}_{i,1}, \ldots, \tilde{L}_{i,\rho_i})$ can be represented as a (different) polynomial in some other set of linear functions. However the following lemma from [Shp] shows that the specific representation that we chose does not change the rank nor the set $\mathrm{Lin}(C)$. We say that a polynomial $h)\bar{x})$ is a polynomial in exactly k linear functions if h can be written as a polynomial in k linear functions but not in $k-1$ linear functions.

Lemma 2.2 (Lemma 20 in [Shp])**.** *Let $h(\bar{x})$ be a polynomial in exactly k linear functions. Let*

$$P(\ell'_1, \ldots, \ell'_k) = h = Q(\ell_1, \ldots, \ell_k)$$

be two different representations for h. Then $\mathrm{span}_1(\{\ell'_i\}_{i\in[k]}) = \mathrm{span}_1(\{\ell_i\}_{i\in[k]})$.

We use the notation $C \equiv f$ to denote the fact that a $\Sigma\Pi\Sigma$ circuit[2] C computes the polynomial f. Notice that this is a syntactic definition, we are thinking of the circuit as computing a polynomial and not a function over the field. Let C be a $\Sigma\Pi\Sigma(k)$ circuit. We say that C is minimal if there is no $A \subsetneq [k]$ such that $C_A \equiv 0$. The following theorem, that relies on the new results of [SS08], gives a bound on the rank of identically zero $\Sigma\Pi\Sigma(k, d, \rho)$ circuits:

Theorem 2.3 (Lemma 4.2 of [KS08] combined with Theorem 2 of [SS08])**.** *Let $k \geq 3$, and C be a simple and minimal $\Sigma\Pi\Sigma(k, d, \rho)$ circuit such that $\deg(C) \geq 2$ and $C \equiv 0$. Using the notations of Equation (4), we have that $\operatorname{rank}(C) < O\left(k^3 \log(d)\right) + \sum_{i=1}^{k} \rho_i \leq O\left(k^3 \log(d)\right) + k\rho$.*

For convenience, we define $R(k, d, \rho) \triangleq O\left(k^3 \log(d)\right) + 2k\rho$ to be twice the above bound on the rank. It follows that $R(k, d, \rho)$ is larger than the rank of any identically zero simple and minimal $\Sigma\Pi\Sigma(k, d, \rho)$ circuit. We also define $R(k, d) \triangleq R(k, d, 0)$ as the upper bound for the rank of a simple and minimal identically zero $\Sigma\Pi\Sigma(k, d)$ circuit. The following theorem gives a bound on the rank of multilinear $\Sigma\Pi\Sigma(k)$ circuits that are identically zero.

Theorem 2.4 (Corollary 6.9 of [DS06] combined with Theorem 2 of [SS08])**.** *There exists an integer function $R_M(k) = O(k^3 \log k)$ such that every multilinear $\Sigma\Pi\Sigma(k)$ circuit C that is simple, minimal and equal to zero, satisfies that $\operatorname{rank}(C) < R_M(k)$.*

Specifically, $R_M(k)$ denotes the minimal integer larger than the rank of any identically zero simple and minimal multilinear $\Sigma\Pi\Sigma(k)$ circuit. This theorem will be used in section 5, where we discuss multilinear circuits.

2.2 Rank Preserving Subspaces

Throughout the paper we use subspaces of low dimension that preserve the circuit rank to some extent. Such subspaces were introduced in [KS08] for the purpose of deterministic black-box identity testing of polynomials that are computable by $\Sigma\Pi\Sigma(k, d, \rho)$ circuits. We define these subspaces, state some of their useful properties and give the construction of [KS08]. Most of the lemmas of this section appear in [KS08] and although the rank bound of [SS08] was not known to [KS08], their proofs remain the same. Therefore, we omit the proofs of the following lemmas.

Given a $\Sigma\Pi\Sigma(k, d, \rho)$ circuit $C = \sum_{i=1}^{k} M_i$ and a subspace V we define C_V to be the circuit whose multiplication gates are $\{M_i|_V\}_{i \in [k]}$. Note that this is a syntactic definition, we do not make an attempt to find a "better" representation for $C|_V$.

Definition 2.5. *Let C be a $\Sigma\Pi\Sigma(k, d, \rho)$ circuit and V an affine subspace. We say that V is r-rank-preserving for C if the following properties hold:*

1. *For any two linear functions $L_1, L_2 \in \operatorname{Lin}(C)$ such that $L_1 \nsim L_2$, it holds that $L_1|_V \nsim L_2|_V$.*
2. $\forall A \subseteq [k],\ \operatorname{rank}(\operatorname{sim}(C_A)|_V) \geq \min\{\operatorname{rank}(\operatorname{sim}(C_A)), r\}$.
3. *No multiplication gate[3] $M \in C$ vanishes on V. In other words $M|_V \not\equiv 0$ for every multiplication gate $M \in C$.*
4. $\operatorname{Lin}(M)|_V = \operatorname{Lin}(M|_V)$ *for every multiplication gate M in C (that is, the non-linear term of M has no new linear factors when restricted to V).*

[2] When speaking of $\Sigma\Pi\Sigma$ circuits we refer to generalized depth-3 circuits.

[3] At times we abuse notations and treat a circuit C as a set of multiplication gates.

The following lemma lists some of the useful properties of rank-preserving subspaces.

Lemma 2.6 (Lemma 3.2 and specific cases of Theorem 3.4 of [KS08])**.** *Let C be a (generalized) depth-3 circuit and V be an r-rank-preserving affine subspace for C. Then we have the following:*

1. *For every $\emptyset \neq A \subseteq [k]$, V is r-rank preserving for C_A.*
2. *V is r-rank-preserving for $\mathrm{sim}(C)$.*
3. $\gcd(C)|_V = \gcd(C|_V)$.
4. $\mathrm{sim}(C)|_V = \mathrm{sim}(C|_V)$.
5. *If C is a $\Sigma\Pi\Sigma(k,d,\rho)$ circuit and $r \geq R(k,d,\rho)$ then $C \equiv 0$ if and only if $C|_V \equiv 0$.*
6. *If C is a $\Sigma\Pi\Sigma(k)$ multilinear circuit, $r \geq R_M(k)$ and $C|_V$ is a multilinear circuit then $C \equiv 0$ if and only if $C|_V \equiv 0$.*

Now that we have seen their definition and some of their useful properties, we explain how to construct rank preserving subspaces. We show two construction methods for rank-preserving-subspaces. One method, used for $\Sigma\Pi\Sigma(k,d,\rho)$ circuits, finds an r-rank preserving subspace. The second method, used for multilinear $\Sigma\Pi\Sigma(k)$ circuits, finds an r-rank preserving subspace that preserves the multilinearity of circuits. That is, for a multilinear $\Sigma\Pi\Sigma(k)$ circuit C we find an r-rank preserving subspace V such that $C|_V$ is also a multilinear circuit.

The following definition and lemma explain how to find a rank preserving subspace for general $\Sigma\Pi\Sigma(k,d,\rho)$ circuits.

Definition 2.7. *Let $\alpha \in \mathbb{F}$ and $r \in \mathbb{N}^+$.*

- *For $0 \leq i \leq r$ let $v_{i,\alpha} \in \mathbb{F}^n$ be the following vector*

$$v_{i,\alpha} = (\alpha^{i+1}, \ldots, \alpha^{n(i+1)}).$$

- *Let $P_{\alpha,r}$ be the matrix whose j-th column (for $1 \leq j \leq r$) is $v_{j,\alpha}$. Namely,*

$$P_{\alpha,r} = (v_{1,\alpha}, \ldots, v_{r,\alpha}) = \begin{pmatrix} \alpha^2 & \alpha^3 & \ldots & \alpha^{r+1} \\ \alpha^4 & \alpha^6 & \ldots & \alpha^{2(r+1)} \\ \vdots & \ddots & & \vdots \\ \alpha^{2n} & & \ldots & \alpha^{n(r+1)} \end{pmatrix}.$$

- *Let $V_{0,\alpha,r}$ be the linear subspace spanned by $\{v_{i,\alpha}\}_{i\in[r]}$. Let $V_{\alpha,r} \subseteq \mathbb{F}^n$ be the affine subspace $V_{\alpha,r} = V_{0,\alpha,r} + v_{0,\alpha}$. In other words,*

$$V_{\alpha,r} = \{P_{\alpha,r}\bar{y} + v_{0,\alpha} \ : \ \bar{y} \in \mathbb{F}^r\}.$$

Lemma 2.8 (Corollary 4.9 in [KS08])**.** *Let C be a $\Sigma\Pi\Sigma(k,d,\rho)$ circuit over $\mathbb{F}$ and $r \geq R(k,d,\rho)$. Let $S \subseteq \mathbb{F}$ be a set of $n\left(\binom{dk}{2} + 2^k\right)\binom{r+2}{2}/\epsilon$ different field elements[4]. Then, for every $\Sigma\Pi\Sigma(k,d,\rho)$ circuit C over $\mathbb{F}$ there are at least $(1-\epsilon)|S|$ elements $\alpha \in S$ such that $V_{\alpha,r}$ is an r-rank-preserving subspace for C.*

[4]Recall our assumption that if $|\mathbb{F}|$ is not large enough then we work over an algebraic extension field of $\mathbb{F}$.

2.2.1 Rank Preserving Subspaces that Preserve Multilinearity

As stated before, when dealing with multilinear circuits we require that the rank-preserving subspaces also preserve the multilinearity of the circuit. Actually, for multilinear circuits we need to slightly change Definition 2.5. We first explain why it is necessary to change the definition and then give the modified definition for the multilinear case. Let C be an n-input multilinear $\Sigma\Pi\Sigma(k)$ circuit having a multiplication gate $M = \prod_{i=1}^{n} x_i$. Let V be a subspace of $\mathbb{F}^n$ of co-dimension r. Assume that $C|_V$ is a multilinear circuit. Then at least r linear functions in M have been restricted to constants and are thus linearly dependent. Whenever $r > 1$, this violates Property 1 of Definition 2.5, indicating that C does not have any low dimension rank preserving subspaces. We now give the definition of multilinear-rank-preserving-subspaces[5]:

Definition 2.9. *Let C be a $\Sigma\Pi\Sigma(k)$ circuit and V an affine subspace. We say that V is r-multilinear-rank-preserving for C if the following properties hold:*

1. *For any two linear functions $L_1 \nsim L_2 \in \mathrm{Lin}(C)$, we either have that $L_1|_V \nsim L_2|_V$ or that both $L_1|_V, L_2|_V$ are constant functions.*
2. *$\forall A \subseteq [k],\ \mathrm{rank}(\mathrm{sim}(C_A)|_V) \geq \min\{\mathrm{rank}(\mathrm{sim}(C_A)), r\}$.*
3. *No multiplication gate $M \in C$ vanishes on V. In other words $M|_V \not\equiv 0$ for every multiplication gate $M \in C$.*
4. *The circuit $C|_V$ is a multilinear circuit.*

Despite the modification of Property 1, Lemma 2.6 applies also for r-multilinear-rank-preserving subspaces[6]. The following definition shows how to construct such a subspace. The two lemmas proceeding it prove the correction of the construction:

Definition 2.10 (Definition 5.3 of [KS08])**.** *Let $B \subseteq [n]$ be a non-empty subset of the coordinates and $\alpha \in \mathbb{F}$ be a field element.*

- *Define V_B as the following subspace:*

$$V_B = \mathrm{span}\{e_i :\ i \in B\}$$

  where $e_i \in \{0,1\}^n$ is the vector that has a single nonzero entry in the i'th coordinate.

- *Let $v_{0,\alpha}$ be, as before, the vector*

$$v_{0,\alpha} = \left(\alpha, \alpha^2, \ldots, \alpha^n\right).$$

- *Let $V_{B,\alpha} \triangleq V_B + v_{0,\alpha}$.*

Lemma 2.11 (Theorem 5.4 of [KS08])**.** *Let C be a $\Sigma\Pi\Sigma(k)$ multilinear depth-3 circuit over the field $\mathbb{F}$ and $r \in \mathbb{N}$. There exists a subset $B \subseteq [n]$ such that $|B| = 2^k \cdot r$ and B has the following properties:*

1. *$\forall A \subseteq [k],\ \mathrm{rank}(\mathrm{sim}(C_A)|_{V_B}) \geq \min\{\mathrm{rank}(\mathrm{sim}(C_A)), r\}$.*
2. *For every $\bar{u} \in \mathbb{F}^n$, $C|_{V_B+\bar{u}}$ is a multilinear $\Sigma\Pi\Sigma(k)$ circuit.*

[5] These subspaces are used only for non-generalized depth-3 circuit. Thus, we only define them for such circuits.

[6] As a matter of fact, in the definition of rank preserving subspaces in [KS08], Property 1 was the same as in Definition 2.9.

Lemma 2.12 (Easy modification of Theorem 5.6 of [KS08])**.** *Let C be a $\Sigma\Pi\Sigma(k)$ multilinear n-variate circuit. Let B be the set guaranteed by lemma 2.11 for some integer r. Then there are less than n^3k^2 many $\alpha \in \mathbb{F}$ such that $V_{B,\alpha}$ is not r-multilinear-rank-preserving for C.*

We now state an additional requirement of r-multilinear-rank-preserving subspaces.

Definition 2.13. *Let C be an n-variate $\Sigma\Pi\Sigma(k,d)$ multilinear circuit over the field $\mathbb{F}$. Let $B \subseteq [n]$ and $\alpha \in \mathbb{F}$. We say that $V_{B,\alpha}$ is a liftable r-multilinear-rank-preserving subspace for C if the following hold: For each $B' \supseteq B$, the subspace $V_{B',\alpha}$ is an r-multilinear-rank-preserving subspace for C.*

Clearly, the subspaces of Definition 2.10 might restrict linear functions from the circuit to constants. Hence, these subspaces are not always liftable. For example, take $C(x,y,z) = (x+y+1)+(x+1)z$, V_1 as the subspace of $\mathbb{F}^3$ where $x = y = 0$ and V_2 as the subspace where $y = 0$. Clearly, $V_1 \subseteq V_2$, V_1 is 1-rank preserving and V_2 is not (as $x+y+1|_{V_2} \sim x+1|_{V_2}$). The following lemma shows how to construct an r-multilinear-rank-preserving liftable subspace.

Lemma 2.14. *Let C be a $\Sigma\Pi\Sigma(k)$ multilinear arithmetic circuit over a field $\mathbb{F}$. Let $r \in \mathbb{N}$ and B be the set guaranteed by lemma 2.11 for C and r. Then there are less than n^4k^2 many $\alpha \in \mathbb{F}$ such that for some $B' \supseteq B$, $V_{B',\alpha}$ is not r-multilinear-rank-preserving for C.*

Proof. Let $B \subseteq B'$. It can easily be seen that the only property that might be violated is Property 1 of definition 2.9. Assume that for each $\hat{B} \supseteq B$ of size $|\hat{B}| = |B|+1$, it holds that $V_{\hat{B},\alpha}$ is r-multilinear-rank-preserving for C. Let $L_1, L_2 \in C$ such that $L_1 \nsim L_2$ and $L_1|_{V_{B',\alpha}}$ is not a constant function. Then there exist some $B' \supseteq \hat{B} \supseteq B$, where $|\hat{B}| = |B|+1$, such that $L_1|_{V_{\hat{B},\alpha}}$ is not a constant function. Hence, by our assumption, $L_1|_{V_{\hat{B},\alpha}} \nsim L_2|_{V_{\hat{B},\alpha}}$ and thus $L_1|_{V_{B',\alpha}} \nsim L_2|_{V_{B',\alpha}}$. It follows that $V_{B',\alpha}$ is r-multilinear-rank-preserving for C.

We now notice that by Lemma 2.12, there are at most n^4k^2 many $\alpha \in \mathbb{F}$ such that for some $\hat{B} \supset B$ of size $|\hat{B}| = |B|+1$, it holds that $V_{\hat{B},\alpha}$ is not r-multilinear-rank-preserving for C. This proves the lemma. □

We conclude this section with the following corollary giving the method to find a liftable r-multilinear-rank-preserving subspace.

Corollary 2.15. *Let $r, n, k \in \mathbb{N}$. Let $S \subseteq \mathbb{F}$ be some set of size $|S| > n^4k^2$. Let C be a multilinear $\Sigma\Pi\Sigma(k)$ circuit of n inputs. There exist some $B \subseteq [n]$, such that $|B| = 2^k \cdot r$ and $\alpha \in S$ such that $V_{B,\alpha}$ is r-multilinear-rank-preserving and liftable for C.*

2.3 A "Distance Function" for $\Sigma\Pi\Sigma$ Circuits

In this section we define a "distance function" for $\Sigma\Pi\Sigma$ circuits. Then we discuss some of its properties. The function measures the dimension of the linear functions in the simplification of the sum of the circuits. Finally we prove that the weight of any two circuits[7] computing the same polynomial is equal up to some additive constant.

Definition 2.16. *Let $C_1, \ldots, C_i$ be a collection of $\Sigma\Pi\Sigma$ circuits ($i \geq 1$). Define:*

$$\Delta(C_1, \ldots, C_i) \triangleq \mathrm{rank}(\mathrm{sim}(\sum_{j=1}^{i} C_j)).$$

Note that this is a syntactic definition as this sum might contain a multiplication gate M and the multiplication gate $-M$.

[7]The weight of a circuit is its distance from 0.

The following lemma explains why we refer to Δ as a distance function.

Lemma 2.17. *(triangle inequality) Let C_1, C_2, C_3 be $\Sigma\Pi\Sigma$ circuits. Then*

$$\Delta(C_1, C_3) \leq \Delta(C_1, C_2, C_3) \leq \Delta(C_1, C_2) + \Delta(C_2, C_3).$$

Proof. The first inequality is trivial since $\mathrm{Lin}(\mathrm{sim}(C_1 + C_3)) \subseteq \mathrm{Lin}(\mathrm{sim}(C_1 + C_2 + C_3))$. We now show the second inequality. Let L be a linear function in $\mathrm{Lin}(\mathrm{sim}(C_1 + C_2 + C_3))$. It suffices to prove that either $L \in \mathrm{Lin}((\mathrm{sim}(C_1 + C_2))$ or $L \in \mathrm{Lin}(\mathrm{sim}(C_1 + C_3))$. We do this by simply checking all possibilities.

$$L \in \mathrm{Lin}(\mathrm{sim}(C_1)) \Rightarrow L \in \mathrm{Lin}(\mathrm{sim}(C_1 + C_2)).$$

$$L \in \gcd(C_1) \overset{(1)}{\Rightarrow} w.l.o.g.\ \ L \notin \gcd(C_2) \Rightarrow L \in \mathrm{Lin}(\mathrm{sim}(C_1 + C_2)).$$

$$L \notin \mathrm{Lin}(C_1) \overset{(2)}{\Rightarrow} w.l.o.g.\ \ L \in \mathrm{Lin}(C_2) \Rightarrow L \in \mathrm{Lin}(\mathrm{sim}(C_1 + C_2)).$$

Implication (1) follows from the fact that $L \notin \gcd(C_1) \cap \gcd(C_2) \cap \gcd(C_3)$. Implication (2) holds since $L \in \mathrm{Lin}(C_1) \cup \mathrm{Lin}(C_2) \cup \mathrm{Lin}(C_3)$. □

We now prove that the weight of two minimal circuits computing the same polynomial is roughly the same. To do so we define a default circuit for a polynomial f. We then show that its weight is roughly the same as the weight of any other minimal circuit computing f.

Definition 2.18. *Let U be a linear space of n-input linear functions. Define the* default basis *of U as the Gaussian elimination of some basis of linear functions*[8]

Definition 2.19. *Let $f(\bar{x})$ be an n-variate polynomial of degree d. Define $\mathrm{Lin}(f)$ as the product of the linear factors of f (i.e. for $f(x_1, x_2, x_3) = x_3(x_1 + x_2)^2(x_1 + x_2^2)$, $\mathrm{Lin}(f) = x_3(x_1 + x_2)^2$). Let $r \in \mathbb{N}^+$ be such that $f/\mathrm{Lin}(f)$ is a polynomial of exactly r linear functions (as defined in Lemma 2.2). Let h be an r-variate polynomial and let $\tilde{L}_1, \ldots, \tilde{L}_r$ be r linear functions such that*

- *$f/\mathrm{Lin}(f) = h(\tilde{L}_1, \ldots, \tilde{L}_r)$.*
- *$\tilde{L}_1, \ldots, \tilde{L}_r$ are the default basis of the linear space they span.*

Notice that the existence of h and $\tilde{L}_1, \ldots, \tilde{L}_r$ is guaranteed by the definition of r. Define C_f, the default circuit *of f, as the following $\Sigma\Pi\Sigma(1, d, r)$ circuit:*

$$C_f \triangleq \mathrm{Lin}(f) \cdot h(\tilde{L}_1, \ldots, \tilde{L}_r).$$

In Appendix A.2 we give a brute force algorithm that given a black-box access to a polynomial, constructs its default circuit. The next lemma implies that for a polynomial f, the Δ measure of the circuit C_f is the lowest among all $\Sigma\Pi\Sigma$ circuits computing f. It also shows that the Δ weight of any circuit computing f is close to the Δ measure of C_f, thus showing that the Δ measure of any two circuits computing the same polynomial is close.

Lemma 2.20. *Let C be a minimal $\Sigma\Pi\Sigma(k, d, \rho)$ circuit computing the polynomial f. Then*

$$\Delta(C_f) \leq \Delta(C) < \Delta(C_f) + R(k + 1, d) + k \cdot \rho.$$

[8]The Gaussian elimination of a set of linear functions is done by performing a Gaussian elimination on the matrix whose rows are the coefficients of the linear functions.

Proof. Using the notations of Definition 2.19, it is not hard to see that $f/\mathrm{Lin}(f)$ is a factor of $\mathrm{sim}(C)$ and that $\mathrm{sim}(C_f) \equiv f/\mathrm{Lin}(f)$. Clearly, $\mathrm{sim}(C)$ is a polynomial of at most $\Delta(C) = \mathrm{rank}(\mathrm{sim}(C))$ linear functions. Since, $f/\mathrm{Lin}(f)$ is a polynomial in exactly $\Delta(C_f)$ linear functions we get that

$$\Delta(C_f) = \mathrm{rank}(\mathrm{sim}(C_f)) \leq \mathrm{rank}(\mathrm{sim}(C)) = \Delta(C).$$

We proceed to the second inequality. If $C \equiv 0$ (i.e., $f = 0$), then since it is also minimal, we get that

$$\Delta(C) < R(k, d, \rho) < R(k+1, d) + k \cdot \rho.$$

Assume that C does not compute the zero polynomial. Consider the $\Sigma\Pi\Sigma(k+1, d, \max\{\rho, \Delta(C_f)\})$ circuit $C - C_f$. As no subcircuit of C (nor C itself) compute the zero polynomial, we have that $C - C_f$ is minimal. Since the circuit clearly computes the zero polynomial, Theorem 2.3 implies that

$$\Delta(C) \leq \Delta(C - C_f) < R(k+1, d) + k \cdot \rho + \Delta(C_f).$$

□

Given a $\Sigma\Pi\Sigma(k, d, \rho)$ circuit C we define its *canonical representation* in the following way. Let $C = \sum_{i=1}^{k} M_i$, be the representation of C as sum of multiplication gates. Let f_i be the polynomial computed by M_i. Then the canonical representation of C is

$$C = \sum_{i=1}^{k} C_{f_i}. \tag{5}$$

Note that the only difference from the description $C = \sum_{i=1}^{k} M_i$, is the basis with respect to which we represent $\mathrm{sim}(M_i)$

3 Canonical κ-Distant Circuits

In this section we define the notion of a κ-distant circuit. We prove the existence of such a circuit C' computing f (Theorem 3.2) and prove its uniqueness (Theorem 3.6). We then show that for a subspace V that is rank preserving for C', the restriction $C'|_V$ is the unique κ-distant circuit computing $f|_V$ (Corollary 3.7).

Definition 3.1. *Let C be a $\Sigma\Pi\Sigma(s, d, r)$ circuit computing a polynomial f (in particular, assume that C is not a $\Sigma\Pi\Sigma(s, d, r-1)$ circuit). We say that C is κ-distant if for any two multiplication gates of C, M and M', we have that $\Delta(M, M') \geq \kappa \cdot r$.*

3.1 Proof of Existence

In this section we prove the existence of a κ-distant $\Sigma\Pi\Sigma(s, d, r)$ circuit C' computing f. We would like to have r as small as possible (as a function of k, d, ρ) as it will affect the running time of the reconstruction algorithm. Our results are given in the following theorem.

Theorem 3.2. *[Existence] Let f be a polynomial that can be computed by a $\Sigma\Pi\Sigma(k, d, \rho)$ circuit and let $\kappa \in \mathbb{N}^+$. Let $r_{init} \in \mathbb{N}^+$ be such that $r_{init} \geq R(k+1, d) + k \cdot \rho$. Then there exist $s, r \in \mathbb{N}^+$ and a $\Sigma\Pi\Sigma(s, d, r)$ circuit C' computing f such that $s \leq k$, $r_{init} \leq r \leq r_{init} \cdot k^{\lceil \log_k(\kappa+1) \rceil \cdot (k-2)}$ and C' is κ-distant.*

Proof. The proof is algorithmic. That is, we give an algorithm for constructing a κ-distant circuit C' that computes the same polynomial as C. The idea behind the algorithm is to cluster the multiplication gates of C, such that any two multiplication gates in the same cluster are close to each other, and any two multiplication gates in different clusters are far away from each other. Then we replace each cluster with the default circuit (recall definition 2.19) for the polynomial that it computes. Before giving the algorithm and its analysis we make the following definition.

Definition 3.3. *Let C be a $\Sigma\Pi\Sigma(k,d,\rho)$ circuit and $I = \{A_1,\ldots,A_s\}$ be some partition of $[k]$. For each $i \in [s]$ define $C_i \triangleq C_{A_i}$. The set $\{C_i\}_{i=1}^s$ is called a partition of C. We say that $\{C_i\}_{i=1}^s$ is (κ',r)-strong when the following conditions hold:*

- $\forall i \in [s],\ \ \Delta(C_i) \le r$.
- $\forall i,j \in [s]$ *such that* $i \ne j$, $\Delta(C_i,C_j) \ge \kappa' \cdot r$.

Input: $n,k,d,\rho,r_{init},\kappa' \in \mathbb{N}$ such that $r_{init} \ge \rho$ and a $\Sigma\Pi\Sigma(k,d,\rho)$ circuit C of n inputs.
Output: An integer $r \ge r_{init}$ and I, a partition of $[k]$.
$\iota \leftarrow \lceil \log_k(\kappa') \rceil$;
$I_1 \leftarrow \{\{1\},\{2\},\ldots,\{k\}\}$;
$r_1 \leftarrow r_{init}$;
while *the partition was changed in any one of the former ι iterations* **do**
  Define j as the number of the current iteration (its initial value is 1);
  Let $G_j(I_j,E_j)$ be a graph where each subset belonging to the partition I_j is a vertex;
  $E_j \leftarrow \emptyset$;
  foreach $A_i \ne A_{i'} \in I_j$ **do**
    if $\Delta(C_{A_i},C_{A_{i'}}) < r_j$ **then**
      $E_j \leftarrow (A_i,A_{i'})$
    end
  end
  $I_{j+1} \leftarrow$ the set of connected components of G_j. That is, every connected component is now a set in the partition;
  $r_{j+1} \leftarrow r_j \cdot k$;
end
Define m as the total number of iterations (that is, in the last iteration we had $j = m$);
$r \leftarrow r_m/k^\iota$;
$I \leftarrow I_m$;

Algorithm 1: Canonical partition of a circuit

Lemma 3.4. *The partition outputted by Algorithm 1 is (κ',r)-strong.*

Proof. Let $I = \{A_1,\ldots,A_s\}$ be the partition found in the algorithm. We shall use the notations of Definition 3.3. First we note that the end of the algorithm, for each $i \ne i' \in [s]$, we have that

$$\Delta(C_i,C_j) \ge r_m = r \cdot k^\iota \ge r \cdot \kappa'.$$

Thus, we only have to prove that for every $i \in [s]$ it holds that $\Delta(C_i) \le r$. Fix some $i \in [s]$. We consider two cases. The first is that C_i is a multiplication gate (i.e., A_i is a singleton). Clearly, its non-linear term is a polynomial of at most ρ linear functions. Hence, $\Delta(C_i) \le \rho \le r_{init} \le r$. In the

second case $C_i = \sum_{l=1}^{s'} C_{i,l}$ where the $C_{i,l}$ were computed at an earlier stage of the algorithm. Let j' be the iteration in which the circuit C_i was computed (that is, the iteration in which the set of indices of the multiplication gates of C_i became a member of the partition). Let $E' \subseteq E_{j'}$ be some a spanning tree of the connected component $C_{i,1}, \ldots, C_{i,s'}$. Then,

$$\Delta(C_i) = \Delta(C_{i,1}, \ldots, C_{i,s'}) \overset{(1)}{\leq} \sum_{(C_{i,l_1}, C_{i,l_2}) \in E'} \Delta(C_{i,l_1}, C_{i,l_2}) < |E'| \cdot r_{j'} \leq k \cdot r_{j'} \overset{(2)}{\leq} r_{m-\iota} = r.$$

Inequality (1) can be reached by repeatedly using the second inequality of Lemma 2.17. To justify inequality (2) notice that the partition did not change in the last ι iterations and thus $j' < m - \iota$. This proves the lemma. □

Now that we are guaranteed that we have a strong partition we prove an upper bound on r. I.e. we show that the weight of each C_i is not too large.

Lemma 3.5. *At the end of Algorithm 1, r_m is at most $r_{init} \cdot k^{\iota \cdot (k-1)}$. Thus, $r \leq r_{init} \cdot k^{\iota \cdot (k-2)}$.*

Proof. In every ι iterations, the number of elements in I is reduced by at least one (otherwise the algorithm terminates). The number of elements in I begins with k and ends with at least 1. Hence, the number of iterations is at most $\iota \cdot (k-1)$ indicating that r_m is at most $r_{init} \cdot k^{\iota \cdot (k-1)}$. Hence, r is bounded from above by $r_{init} \cdot k^{\iota \cdot (k-2)}$. □

We proceed with the proof of Theorem 3.2. We now set $\kappa' = \kappa + 1$ and fix some integer r_{init} such that $r_{init} \geq R(k+1, d) + k \cdot \rho$. Let $\{C_i\}_{i=1}^{s}$ be the partition outputted by Algorithm 1. Let $\{f_i\}_{i=1}^{s}$ be the polynomials computed by the subcircuits of the partition. That is, C_i computes f_i. Define

$$C' \triangleq \sum_{i=1}^{s} C_{f_i}.$$

We now show that C' satisfies the requirements of Theorem 3.2. Since the partition is (κ', r)-strong, Lemma 2.20 implies that $\Delta(C_{f_i}) \leq \Delta(C_i) \leq r$, for each $i \in [s]$. Hence, C' is a $\Sigma\Pi\Sigma(s, d, r)$ circuit. Let $C_i, C_{i'}$ be two subcircuits in the partition of C. Then,

$$\Delta(C_{f_i} + C_{f_{i'}}) \overset{(1)}{\geq} \Delta(C_{f_i + f_{i'}}) \overset{(2)}{>} \Delta(C_i + C_{i'}) - (R(k+1, d) + k \cdot \rho) \overset{(3)}{\geq}$$

$$\kappa' \cdot r - r = \kappa \cdot r$$

Inequalities 1 and 2 stem from lemma 2.20 (we assume w.l.o.g. that C is minimal and thus so is $C_i + C_{i'}$). Inequality 3 holds since $r \geq r_{init} \geq R(k+1, d) + k \cdot \rho$ and $\Delta(C_i + C_{i'}) \geq \kappa' \cdot r$ (by Lemma 3.4). This concludes the proof of Theorem 3.2 □

3.2 Uniqueness

In this section we prove, for a large enough value of κ, the uniqueness of a κ-distant $\Sigma\Pi\Sigma(k, d, r)$ circuit computing a polynomial f. As a corollary we obtain a result showing that if C' is a κ-distant circuit computing f and V is rank preserving for C' then the unique κ-distant circuit computing $f|_V$ is $C'|_V$.

Theorem 3.6. *[Uniqueness] Let f be a polynomial of degree d. Let $k, r, \kappa \in \mathbb{N}^+$ be such that $\kappa \geq R(2k, d, r)/r$. Then there exists at most one canonical minimal κ-distant $\Sigma\Pi\Sigma(s, d, r)$ circuit computing f such that $s \leq k$.*

Proof. Let $s, s' \leq k$ and $C_1 \triangleq \sum_{i=1}^{s} C_{f_i}$ and $C_2 \triangleq \sum_{i=1}^{s'} C_{g_i}$ be two canoncial minimal κ-distant circuits computing f. It suffices to prove that $s = s'$ and that for some reordering of the multiplication gates, $\forall i \in [s],\ C_{f_i} = C_{g_i}$. Consider the $\Sigma\Pi\Sigma(s + s', d, r)$ circuit

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

Clearly, C computes the zero polynomial. We now show that each minimal subcircuit of C is composed of exactly two multiplication gates C_{f_i} and C_{g_j} where $i \leq s$ and $j \leq s'$. This will prove our claim. Let $\tilde{C}$ be some minimal subcircuit of C. Clearly $\tilde{C}$ is a $\Sigma\Pi\Sigma(m, d', r)$ circuit where $2 \leq m \leq s + s'$ and $d' \leq d$. It suffices to prove that $\tilde{C}$ cannot contain two multiplication gates originating from the same C_l ($l \in \{1, 2\}$). If $s = s' = 1$ then both circuit are of the form C_f and are thus clearly equal. So assume w.l.o.g. that $s \geq 2$. Assume for a contradiction and w.l.o.g. that both C_{f_1} and C_{f_2} are multiplication gates in $\tilde{C}$. Then,

$$\kappa \cdot r \overset{(1)}{\leq} \Delta(C_{f_1}, C_{f_2}) \leq \Delta(\tilde{C}) \overset{(2)}{<} R(s + s', d, r) \leq R(2k, d, r).$$

Inequality 1 holds since C_1 is κ-distant, and Inequality 2 follows from the definition of $R(k, d, r)$ right after Theorem 2.3. This contradicts our assumption regarding κ and thus proves the lemma. □

Corollary 3.7. *Let f be an n-variate polynomial of degree d.*

- *Let $k, s, r, \kappa \in \mathbb{N}^+$ be such that $\kappa \geq R(2k, d, r)/r$ and $s \leq k$.*
- *Let C' be a minimal $\Sigma\Pi\Sigma(s, d, r)$ κ-distant circuit computing f.*
- *Let V be an $(r \cdot \kappa)$-rank preserving subspace for C'.*

Then $C'|_V$ is a minimal κ-distant $\Sigma\Pi\Sigma(s, \deg(f|_V), r)$ circuit computing $f|_V$. In addition there is no other κ-distant $\Sigma\Pi\Sigma(k', \deg(f|_V), r)$ minimal circuit computing $f|_V$ for any $k' \leq k$.

Proof. Let M, M' be two different multiplication gates of C'. As V is $(r \cdot \kappa)$-rank preserving for C we get that

$$\Delta((M + M')|_V) \geq \min\{\Delta(M + M'), r \cdot \kappa\} \geq r \cdot \kappa,$$
$$\Delta(M|_V) \leq \Delta(M) \leq r.$$

Hence $C'|_V$ is a κ-distant $\Sigma\Pi\Sigma(s, d', r)$ circuit computing $f|_V$. Since $\kappa \geq R(2k, d, r)/r$, Theorem 3.6 implies the uniqueness of $C'|_V$. □

4 Reconstructing $\Sigma\Pi\Sigma(k, d, \rho)$ Circuits

In this section we give our main reconstruction algorithm (Theorem 1). Recall the general scheme of the algorithm that was described in Section 1.4: we first restrict the inputs to the polynomial f to a low dimensional rank-preserving subspace V, then we reconstruct the unique κ-distant circuit for $f|_V$, and finally we lift this to a circuit over $\mathbb{F}^n$. We now explain the intuition behind this approach. Recall that Corollary 3.7 states that C is a κ-distant circuit for f and V is a *rank-preserving* subspace for C (with the adequate parameters) then $C|_V$ is the unique κ-distant circuit for $f|_V$. Stated differently, if we manage to find a κ-distant circuit C' that computes $f|_V$, then if we lift each multiplication gate of C', separately, to $\mathbb{F}^n$ then we get back the circuit C. Therefore, our goal is to find a κ-distant circuit for $f|_V$. We do this in Section 4.1. This is the main technical part of the algorithm.

We now describe the main idea in the reconstruction of $f|_V$. Let us first start with the simple case that C' $(= C|_V)$ has a single generalized multiplication gate. Then, by factoring $f|_V$ we can find[9] $\gcd(C')$, and then use brute force interpolation to compute $\mathrm{sim}(C')$ (this is possible since $\dim(V)$ is relatively small). This case is not so difficult, however it is not clear what to do if we have more than a single multiplication gate. We handle this by finding a reduction to the case of a circuit with a single multiplication gate. The reduction is based on an *isolation lemma* that roughly says that there exists a set of subspaces that "zero-out" all but one multiplication gate.

Lemma 4.1 (Isolation Lemma (informal))**.** *For every t-variate κ-distant circuit $C = C_{f_1} + \ldots + C_{f_k}$, there exists a polynomial sized set of subspaces $\mathcal{V} = \{U_i\}_i$, where each $U_i \subset \mathbb{F}^t$ has co-dimension at most k, such that the following holds. There exists an index $i_0 \in [k]$ such that for every $U \in \mathcal{V}$ we have that $C_{f_j}|_U = 0$ if and only if $j \neq i_0$. Namely, all gates but $C_{f_{i_0}}$ vanish when restricted to the subspaces in $\mathcal{V}$. Moreover, there is an efficient algorithm for reconstructing $C_{f_{i_0}}$ from the restrictions $\{C_{f_{i_0}}|_U\}_{U \in \mathcal{V}}$.*

The exact version of the lemma is given in Lemma 4.4 (which strongly depends on Lemma 4.6). The "moreover" part of the lemma is proved by gluing the different gates $C_{f_{i_0}}|_U$ to a single $C_{f_{i_0}}$. The way to glue the different restrictions together is given in Section 4.1.2 (Algorithms 2 and 5), and is based on the earlier work of [Shp]. The main question that remain then is how to find the isolating set of subspaces $\mathcal{V}$ (or even how to prove its existence). It turns out that because the dimension of V is relatively low (eventually it will be $\mathrm{polylog}(n)$), then if we know that such a set exists then we can go over all possibilities for it. I.e. we go over all possibilities for the set $\mathcal{V}$, and for each "guess", we try to reconstruct a multiplication gate, that is, a circuit of the form $C_{f_{i_0}}$. The point is that after we reconstruct $C_{f_{i_0}}$ we can continue by recursion and learn all the other multiplication gates. In this way we get many guesses for C', and then we can simply check (by going over all elements of V) which one is a correct representation for it (this is given in Algorithm 4).

In view of the above, we just have to prove the existence of such a set $\mathcal{V}$. The idea is to construct the subspaces step by step. That is, we first find a set of linear functions $\mathcal{L}$ that *splits* the multiplication gates of C' to two sets A and $\bar{A}$ such that for every $\ell \in \mathcal{L}$ all the multiplication gates in $\bar{A}$ vanish when restricted to the subspace defined by $\ell = 0$. On the other hand, none of the multiplication gates in A vanishes on $\ell = 0$ (actually there is a stronger demand that we skip now). The definition of a splitting set and the proof of its existence are given in Section 4.1.1 (Definition 4.3 and Lemma 4.6). Given a splitting set $\mathcal{L}$ and the sets $A, \bar{A}$ we can look for a splitting set $\mathcal{L}'$ for A (that splits it to A' and $A \setminus A'$ for some $\emptyset \subsetneq A' \subsetneq A$). The sets $\mathcal{L}$ and $\mathcal{L}'$ define a set of co-dimension 2 subspaces: for every $\ell \in \mathcal{L}$ and $\ell' \in \mathcal{L}'$ we have the space defined by $\ell = \ell' = 0$. We can continue to do so until we are left with a single multiplication gate that was split from the other multiplication gates by the subspaces that we generated, which are of co-dimension at most s when C is a $\Sigma\Pi\Sigma(s, d, r)$ circuit (actually, this is not a completely accurate description, see Definition 4.2 and Lemma 4.4, that discuss *m-linear function trees* for a more formal treatment). This proves the existence of $\mathcal{V}$. In particular, we reduced the problem of proving the existence of the subspaces $\mathcal{V}$ to the problem of proving the existence of a splitting set. The proof of the existence of splitting sets is the most technical part of the proof. It is based on an extension of the main lemma of [DS06] (Lemma 4.9 in our paper). At the heart of the proof of Lemma 4.9 lies a connection to *locally decodable codes*. Basically, one way to think of the proof regarding the existence of splitting sets is that if there is no splitting set then we can find a too good locally decodable code inside C', and since there are known bounds on the performance of such codes, it must be the case that splitting sets exist. The proof of the lemma is quite technical and we

[9] Specifically, we use a black box factoring algorithm, which produces black boxes to the irreducible components of $f|_V$, and isolate the linear factors.

cannot give further intuition at this stage (except to advise the reader to read the intuition given in [DS06]). To conclude the algorithm for constructing C has the following form (Algorithm 6).

- Find a set of subspaces, of low dimension, that contains a rank-preserving subspace for C. Denote this subspace with V. We do the following for each space in the family but focus on V as we will later verify which of the circuits that we constructed is the correct one.
- Run Algorithm 4, to get $C|_V$. The algorithm uses as a subroutine Algorithm 3 that constructs a single multiplication gate of $\tilde{C}$ using an m-linear function tree (whose existence is based on the existence of splitting sets). In particular we have to try Algorithm 4 for every "guess" of an m-linear function tree.
- Run Algorithm 5 to lift each multiplication gate of the κ-distant circuit $C'|_V$ to $\mathbb{F}^n$.
- Use the PIT algorithm of [KS08] to find the circuit C.

4.1 Finding a κ-Distant Representation in a Low Dimension Subspace

In this section we show how to find a minimal κ-distant $\Sigma\Pi\Sigma(s,d,r)$ circuit computing $f|_V$, given black box oracle access to $f|_V$. The values of s,d,r and κ that we choose are such that the uniqueness of the circuit is guaranteed (according to Theorem 3.6 and Corollary 3.7). Since we only deal with the restriction of f to the subspace V we assume for convenience that f is a polynomial of $t \triangleq \dim(V)$ variables and find the minimal κ-distant $\Sigma\Pi\Sigma(s,d,r)$ circuit $C' \triangleq \sum_{i=1}^{s} C_{f_i}$ computing f (that is, we abuse notations and write f instead of $f|_V$).

In order to find such a circuit we show an algorithm that finds a single multiplication gate of the circuit. Namely, we show how to reconstruct the circuit C_{f_i} for some $i \in [s]$. In the main algorithm we recursively find the other multiplication gates by running the algorithm on $f - f_i$ (which clearly has a $\Sigma\Pi\Sigma(s-1,d,r)$ κ-distant circuit). Given black-box access to the polynomial f_i we can construct the circuit C_{f_i} using brute force interpolation (see Algorithm 11 in section A.2 of the appendix). Obviously, the lack of access to any such polynomial f_i complicates matters. To overcome this problem we find a set of subspaces $\{U_j\}$ such that for some $i \in [s]$, we have that for every U_j,

$$f|_{U_j} = f_i|_{U_j}.$$

Moreover, for every $i \neq i' \in [s]$, it holds that $f_{i'}|_{U_j} = 0$. For each such subspace $U_j \subseteq \mathbb{F}^t$ we use algorithm 11 to reconstruct $C_{f_i|_{U_j}}$. Our choice of subspaces $\{U_j\}$ is such that we are able to construct C_{f_i} given the circuits $\{C_{f_i|_{U_j}}\}$. To conclude, the scheme of our algorithm consists of three steps:

1. Obtain the set of subspaces $\{U_j\}_j$ where for some $i \in [s]$, we have that for every U_j, $f|_{U_j} = f_i|_{U_j}$.
2. Reconstruct the circuits $\{C_{f|_{U_j}}\}$ (that are the same as $\{C_{f_i|_{U_j}}\}$).
3. Glue together the different restrictions $\{C_{f_i|_{U_j}}\}$ to get the circuit C_{f_i}.

Section 4.1.1 deals with the first step. There we present the properties that the set of subspaces $\{U_j\}$ has to satisfy in order for the third step (the "gluing" step) to work. We then construct the set $\{U_j\}$ having the required properties. In the second step we simply apply the brute force algorithm given in the appendix (section A.2). In section 4.1.2 we give the algorithm for the third step. That is, we reconstruct C_{f_i} given the circuits $\{C_{f_i|_{U_j}}\}$. We conclude in section 4.1.3 with algorithm 4 that finds the κ-distant circuit C'.

4.1.1 Step 1: Finding the set of subspaces

In this section we deal with the following problem. Let

$$C' \triangleq \sum_{i=1}^{s} C_{f_i} \tag{6}$$

be a $\Sigma\Pi\Sigma(s,d,r)$ minimal κ-distant t-variate circuit computing a polynomial f. Recall that s is a constant, d is the degree of the polynomial f and t is relatively small (roughly polylog(d)). The values of r and κ will be revealed later. We are searching for a set of (affine) subspaces $\{U_j\}$ with the following properties:

1. $\exists i \in [s]$ s.t. $\forall U_j$, $f|_{U_j} = f_i|_{U_j}$.
2. Given the circuits $\{C_{f_i|_{U_j}}\}$ it is possible to reconstruct the circuit C_{f_i}.

Actually, we are not able to find one set of subspaces realizing the requirements. Instead, we find a family of sets of subspaces that contains at least one "good" set for every $\Sigma\Pi\Sigma(s,d,r)$ circuit C. By gluing together the restrictions of f to each set of subspaces in the family we get a set of possible "guesses" for f_i. For each guess of f_i we recursively search a κ-distant circuit for $f - f_i$ and thus obtain a set of possible circuits computing f. It will be clear that one of the circuits is the needed κ-distant circuit C'. Given the resulting circuits, it is not difficult to find the correct one by using the black-box polynomial identity test of [KS08]. Note that the size of the family of sets of subspaces determines the running time of the algorithm. The size of our family will be exponential in t and in polylog(d).

Most of this section is devoted to proving the existence of a set $\{U_j\}$ containing a low (polylogarithmic) number of subspaces of a small ($< s$) co-dimension. The subspaces $\{U_j\}$ are found recursively. We first find a set of m subspaces $\{U_{1,j}\}_{j\in[m]}$ of co-dimension 1 ($m = \text{polylog}(d)$ and its exact value will be determined later) such that $f|_{U_j}$ has a $\Sigma\Pi\Sigma(s',d,r)$ κ-distant circuit for some $0 < s' < s$. Specifically, in each subspace, at least one of the polynomials $\{f_i\}_{i=1}^{s}$ vanishes while f does not. Within each subspace $U_{1,j}$ we recursively find m co-dimension 1 subspaces in which the restriction of f has a $\Sigma\Pi\Sigma(s'',d,r)$ κ-distant circuit for some $0 < s'' < s'$. We continue this process until we obtain many subspaces such that on any one of them exactly one of the polynomials $\{f_i\}_{i=1}^{s}$ does not vanish. From this large set of subspaces we extract a sufficiently large set of subspaces $\{U_j\}$ on which f_i is the only polynomial that does not vanish, for some fixed i.

The process of finding the set of subspaces $\{U_j\}$ implicitly defines a directed tree. The root corresponds to the entire space of t-variate linear functions. Each edge corresponds to a linear function defining a co-dimension 1 subspace. Each non-root vertex of distance l from the root corresponds to a co-dimension l subspace in a natural way. Hence, what we are after is a family of trees (of small depth) that contains at least one tree whose leaves correspond to a "good" set of subspaces. In fact, we shall prove the existence of a "good" tree of small depth whose non-leaf vertices have exactly m children, and then use brute force search to find this tree. We now define m-linear functions trees whose existence will guarantee the existence of a "good" set of subspaces $\{U_j\}$

Definition 4.2. *An "m-linear function tree" is a tree graph with the following properties.*

- *Each non-leaf vertex has exactly m children.*
- *Each edge e is labelled with a linear function φ_e.*
- *All linear functions labelling the edges from a vertex u to its children are linearly independent^H.*

- *For each vertex u let $\pi_u = \{e_1, \dots, e_l\}$ be the path from the root to u. Define V_u as the co-dimension l subspace in which $\{\varphi_{e_i}\}_{i=1}^{l}$ vanish.*
- *Define the size of the "m-linear function tree" as the number of non-leaf vertices in the tree.*

An "m-linear function tree" is an interpolating tree *for a polynomial f of degree d, if the following hold:*

1. $m \geq \max\{100\log(d), \Delta(C_f) + 2\}$.
2. *For every vertex u in the tree and edge e connecting u to one of its children, $\varphi_e \notin \mathrm{Lin}(C_{f|_{V_u}})$.*

The usefulness of this definition is demonstrated in the following section. There we give an algorithm that receives as input an interpolating tree for f_i and the circuits $C_{f_i|_{V_u}}$, for every leaf u of the tree, and outputs C_{f_i}. In the rest of the section we prove the existence of an m-linear function tree of depth lower than s that is an interpolation tree for some f_i. Moreover, in each leaf u of the tree it holds that $f|_{V_u} = f_i|_{V_u}$. We now explain how to choose the labels of the edges such that the resulting tree will have the required properties.

Definition 4.3. *Let f be a t-variate polynomial of degree d. Let $r, s, k \in \mathbb{N}^+$ and $\xi(k)$ be some increasing positive integer function. Let $C' \triangleq \sum_{i=1}^{s} C_{f_i}$ be a $\xi(k)$-distant $\Sigma\Pi\Sigma(s, d, r)$ circuit computing f. Let $\mathcal{L}$ be a set of linearly independent^H linear functions. We say that $\mathcal{L}$ is a (ξ, r, k)-splitting set for C' when it satisfies the following properties:*

1. $|\mathcal{L}| \geq \max\{100\log(d), r + 2\} \cdot k$.
2. *There exists some $\emptyset \neq A \subsetneq [s]$ such that each $\varphi \in \mathcal{L}$ is a linear factor of f_i if and only if $i \notin A$.*
3. *For every $\varphi \in \mathcal{L}$ and $i \in [s]$, we have that $\varphi \notin \mathrm{Lin}(\mathrm{sim}(C_{f_i}))$.*
4. *For each $i \neq j \in A$ and $\varphi \in \mathcal{L}$, it holds that $\Delta(C_{f_i|_{\varphi=0}}, C_{f_j|_{\varphi=0}}) \geq \xi(k-1) \cdot r$.*

The reason for the name "splitting set" comes from the fact that it splits the multiplication gates of C' into two non-trivial sets (A and $[s]\backslash A$), such that for every $i \in [s] \setminus A$ and every $\varphi \in \mathcal{L}$ we have that $C_{f_i}|_{\varphi=0} = 0$. Before we prove the existence of such a set we give a lemma that demonstrates its usefulness.

Lemma 4.4. *Let $\xi(k)$ be an increasing integer function and $r, d \in \mathbb{N}^+$. Assume that for every $s \leq k \in \mathbb{N}^+$ and every $\xi(k)$-distant $\Sigma\Pi\Sigma(s, d, r)$ circuit $C' \triangleq \sum_{i=1}^{s} C_{f_i}$ computing the polynomial f, there exists a (ξ, r, k)-splitting set. Define $m \triangleq \max\{100\log(d), r + 2\}$. Then for each such circuit, there exists an m-linear function tree T with the following properties*

1. *The depth of T is smaller than s.*
2. *There exists $i' \in [s]$ such that T is an interpolating tree for $f_{i'}$.*
3. *For every subspace V corresponding to a leaf of T we have that $f|_V = f_{i'}|_V$ (recall Definition 4.2).*

Proof. We prove the claim by induction on s. For $s = 1$, the tree is a single node and the claim is trivial. For $s > 1$ let $\mathcal{L}$ be the (ξ, r, k)-splitting set guaranteed. Let $\varphi \in \mathcal{L}$ and consider $f|_{\varphi=0}$. Then, by Definition 4.3, there exists some $\emptyset \neq A \subsetneq [s]$ such that $C_\varphi \triangleq \sum_{i \in A} C_{f_i|_{\varphi=0}}$ is a $\xi(k-1)$-distant $\Sigma\Pi\Sigma(|A|, d, r)$ circuit computing $f|_{\varphi=0}$. Since $|A| < s$ we get by induction that there exists an m-linear function tree T_φ with the following properties:

1. The depth of T_φ is lower than $|A| \leq s-1$.
2. There exists $i_\varphi \in [s]$ such that T_φ is an interpolating tree for f_{i_φ}.
3. For every subspace V corresponding to a leaf of T_φ we have that $f|_V = f_{i_\varphi}|_V$.

In particular, there is some $i' \in [s]$ such that at least $|\mathcal{L}|/s \geq |\mathcal{L}|/k \geq m$ of the trees T_φ satisfy $i_\varphi = i'$. We now construct a new tree T by connecting m of these trees (those with $i_\varphi = i'$) to a new root and labelling each edge from the root to T_φ with the linear function φ. It is not hard to see that T is an m-linear function tree satisfying the required properties (recall that by the definition of $\Delta(C_f)$ we have that $\Delta(C_{f_{i'}}) \leq r$, as $f_{i'}$ is computed by a multiplication gate in a $\Sigma\Pi\Sigma(s,d,r)$ circuit). $\square$

In other words, the lemma above implies that a splitting set guarantees the existence of an interpolating tree. We proceed to proving the existence of splitting sets. That is, we find a function ξ such that for every $s \leq k \in \mathbb{N}^+$ and $\xi(k)$-distant $\Sigma\Pi\Sigma(s,d,r)$ circuit C', there exists a (ξ,r,k)-splitting set $\mathcal{L}$. Our proof will have the following structure. We first define the function ξ and discuss some of its properties. We then find a set of linearly independent^H linear functions for which Property 2 of definition 4.3 holds, $\mathcal{L}$. We then give an upper bound on the number of functions in $\mathcal{L}$ that do not satisfy either Property 3 or Property 4 of Definition 4.3. Finally, we remove the "bad" functions from $\mathcal{L}$ and verify that the number of function remaining is large enough (thus guaranteeing Property 1). The function we work with is the following:

$$\xi_d(k) = \begin{cases} \prod_{j=3}^{k} \binom{j-1}{2} \cdot 2^{\frac{(k+3)(k+4)}{2}} \cdot (\log(d))^k & k > 2 \\ 2^{\frac{(k+3)(k+4)}{2}} \cdot (\log(d))^k & k = 2 \end{cases} . \tag{7}$$

The next lemma gives some properties of ξ_d. Its proof is both trivial and technical and is thus omitted:

Lemma 4.5. *For $k \geq 2$, $d \geq 2$ and $r \geq 50\log(d)$ the following properties hold:*

- *$\xi_d(k)$ is an increasing function.*
- *$\xi_d(k) \geq 2k+1$.*
- *$\xi_d(k) - 2 \geq \xi_d(k) - k \geq \xi_d(k)/2$.*
- *$\frac{\xi_d(k)\cdot r}{2^{k+1}\binom{k-1}{2}} \geq 2k\log(dk) + 2k$ (for $k=2$, replace $\binom{k-1}{2}$ with 1).*
- *$\frac{\xi_d(k)}{2^{k+1}} > 2k\binom{k-1}{2} + 1$ (for $k=2$, replace $\binom{k-1}{2}$ with 1).*

We are now ready to find the linear functions required for the splitting set.

Lemma 4.6. *Let f be a t-variate polynomial of degree $d \geq 2$. Let $C' = \sum_{i=1}^{s} C_{f_i}$ be a $\xi_d(k)$-distant $\Sigma\Pi\Sigma(s,d,r)$ circuit computing f, where $r \geq 50\log(d)$. Then there exists a (ξ_d,r,k)-splitting set for C'.*

Proof. We start with the following lemma that shows that there are many linear functions satisfying Property 2 of Definition 4.3.

Lemma 4.7. *Let $i \neq i' \in [s]$. There exists a set $\mathcal{L}$ of $\frac{\xi_d(k)r}{2^k}$ linearly independent^H linear functions and a partition of $[s]$ to two sets $A, \bar{A} \triangleq [s] \setminus A$ such that*

- *i and i' are not in the same set.*

- *For every $\varphi \in \mathcal{L}$ and $j \in [s]$ we have that φ is a linear factor of f_j if and only if $j \notin A$.*

Proof. We first prove a weaker lemma that shows that for every two different multiplication gates of C' there are many linearly independent^H linear functions that divide one multiplication gate but not the other. We then see that the required set $\mathcal{L}$ can be extracted from those linear functions.

Lemma 4.8. *There are at least $\frac{\xi_d(k)r-2r}{2}$ linearly independent^H linear functions that are either linear factor of f_i and not of $f_{i'}$ or vice versa.*

Proof. To ease the notations we prove the claim for $i = 1$ and $i' = 2$. It is not hard to see that

$$\mathrm{Lin}(\mathrm{sim}(C_{f_1} + C_{f_2})) = \mathrm{Lin}(\mathrm{sim}(C_{f_1})) \cup \mathrm{Lin}(\mathrm{sim}(C_{f_2})) \cup$$

$$(\gcd(C_{f_1}) \setminus \gcd(C_{f_2})) \cup (\gcd(C_{f_2}) \setminus \gcd(C_{f_1})).$$

For each set, we consider the dimension of its span and obtain the following inequality:

$$\Delta(C_{f_1}, C_{f_2}) \leq \Delta(C_{f_1}) + \Delta(C_{f_2}) +$$

$$\dim_1(\gcd(C_{f_1}) \setminus \gcd(C_{f_2})) + \dim_1(\gcd(C_{f_2}) \setminus \gcd(C_{f_1})).$$

Since C_{f_1}, C_{f_2} are multiplication gates of a $\xi_d(k)$-distant $\Sigma\Pi\Sigma(s, d, r)$ circuit, we have that $\xi_d(k) \cdot r \leq \Delta(C_{f_1}, C_{f_2})$ and that $\Delta(C_{f_1}) + \Delta(C_{f_2}) \leq 2r$. Hence,

$$\xi_d(k) \cdot r \leq 2r + \dim_1(\gcd(C_{f_1}) \setminus \gcd(C_{f_2})) +$$

$$\dim_1(\gcd(C_{f_2}) \setminus \gcd(C_{f_1})).$$

Therefore, w.l.o.g. we get that

$$\dim_1(\gcd(C_{f_1}) \setminus \gcd(C_{f_2})) \geq \frac{\xi_d(k) \cdot r - 2r}{2}.$$

According to the definition of a default circuit (Definition 2.19), each linear function in $\gcd(C_{f_1}) \setminus \gcd(C_{f_2})$ is a factor of f_1 and not of f_2. This concludes the proof of Lemma 4.8. □

We return to the proof of lemma 4.7. Assume w.l.o.g. that there are $\frac{\xi_d(k)r-2r}{2}$ linearly independent^H linear functions that divide[10] $f_{i'}$ and do not divide f_i. We initialize $A = \{i\}$, $\bar{A} = \{i'\}$ and $\mathcal{L}$ as the set containing these linear functions. Consider a polynomial f_j where $j \neq i, i'$. If most of the functions in $\mathcal{L}$ divide f_j, then we keep in $\mathcal{L}$ only the functions dividing f_j and add j to $\bar{A}$. Otherwise, we keep in $\mathcal{L}$ only the functions that do not divide f_j and add j to A. We repeat the procedure for all $j \neq i, i'$. In each step we keep at least half of $\mathcal{L}$. As there are $k - 2$ such steps, the size of the set $\mathcal{L}$ that remained at the end of the process is at least

$$\frac{\xi_d(k)r - 2r}{2^{k-1}}.$$

According to lemma 4.5,

$$\frac{\xi_d(k)r - 2r}{2^{k-1}} \geq \frac{\xi_d(k)r}{2^k}.$$

Thus, it can easily be seen that the acquired set $\mathcal{L}$ satisfies the requirements of the lemma. This completes the proof of Lemma 4.7. □

[10] When a φ is a linear factor of f we also say that it divides f.

We continue with the proof of Lemma 4.6. Let $\mathcal{L}$ denote the a set of functions guaranteed by Lemma 4.7. To bound the number of functions that do not satisfy Condition 3 of Definition 4.3, we notice that $\dim_1(\mathrm{Lin}(\mathrm{sim}(C_{f_i}))) \leq r$ for every $i \in [s]$. Hence, there are at most $s \cdot r \leq k \cdot r$ linear functions in $\mathcal{L}$ that belong to $\mathrm{Lin}(\mathrm{sim}(C_{f_i}))$ for some $i \in [s]$ (because the functions of $\mathcal{L}$ are linearly independent^H). To bound the number of functions that do not satisfy Condition 4 we need the following lemma which is a generalization of claim 4.8 of [DS06]. The proof is given in Section B of the appendix.

Lemma 4.9. *Let $k, s, d, r \in \mathbb{N}$ be such that $s \leq k$ and let $C \triangleq \sum_{i=1}^{s} C_{f_i}$ be a $\Sigma\Pi\Sigma(s, d, r)$ circuit. Let $\hat{\mathcal{L}}$ be a set of linearly independent^H linear functions and $A \subseteq [s]$ be a set of size $|A| \geq 2$. Let $\chi, r' \in \mathbb{N}$ be such that $\chi > 1$ and r' satisfies that for every $A' \subseteq A$ of size $|A'| = 2$ we have that $r' \leq \Delta(\sum_{i \in A'} C_{f_i}) - r \cdot k$. Assume that for each $\varphi \in \hat{\mathcal{L}}$, the following holds:*

- *For every $i \in [s]$, φ divides f_i if and only if $i \notin A$.*
- *For every $i \in [s]$, $\varphi \notin \mathrm{Lin}(\mathrm{sim}(C_{f_i}))$.*
- *$\exists i \neq j \in A$, such that $\Delta(C_{f_i|_{\varphi=0}}, C_{f_j|_{\varphi=0}}) < \frac{r'}{2 \cdot \chi \cdot \log(d)}$.*

Then

$$|\hat{\mathcal{L}}| \leq \binom{|A|}{2} \cdot (\max\{2k\log(dk) + 2k, \frac{r'}{\chi}\} + r \cdot k).$$

In other words, the lemma bounds the number of linear functions in $\mathcal{L}$ that satisfy Properties 2 and 3 of Definition 4.3 but not Property 4. We now proceed with the proof of Lemma 4.6. Consider the functions of $\mathcal{L}$ for which property 4 does not hold (but Properties 2 and 3 do hold). For such a function φ, there are $i \neq j \in A$ such that[11]

$$\Delta(C_{f_i|_{\varphi=0}}, C_{f_j|_{\varphi=0}}) < \xi_d(k-1) \cdot r = \frac{\xi_d(k-1)}{\xi_d(k)} \cdot (\xi_d(k) \cdot r) = \frac{\xi_d(k) \cdot r}{2^{k+3}\binom{k-1}{2}\log(d)}.$$

Set $r' = r \cdot (\xi_d(k) - k)$ and $\chi = 2^{k+1} \cdot \binom{k-1}{2}$. It follows that

$$\Delta(C_{f_i|_{\varphi=0}}, C_{f_j|_{\varphi=0}}) < \frac{\xi_d(k) \cdot r}{2^{k+3}\binom{k-1}{2}\log(d)} = \frac{\xi_d(k) \cdot r}{4\chi\log(d)} \overset{(1)}{\leq} \frac{(\xi_d(k) - k) \cdot r}{2\chi\log(d)} = \frac{r'}{2\chi\log(d)},$$

where inequality (1) stems from Lemma 4.5. In addition, for every $i_1 \neq i_2 \in [s]$ we have that

$$\Delta(C_{f_{i_1}}, C_{f_{i_2}}) - r \cdot k \geq (\xi_d(k) - k) \cdot r = r'.$$

Hence, we may apply Lemma 4.9 with the parameters r' and χ, and get that there are at most

$$\binom{k-1}{2} \cdot \left(\max\left\{2k\log(dk) + 2k, \frac{\xi_d(k) \cdot r}{2^{k+1}\binom{k-1}{2}}\right\} + r \cdot k\right) = \frac{\xi_d(k) \cdot r}{2^{k+1}} + r \cdot k \cdot \binom{k-1}{2}$$

linear functions in $\mathcal{L}$ for which condition 4 is the only condition that does not hold (the equality follows from Lemma 4.5). We now remove the functions that do not satisfy Conditions 3 or 4 from $\mathcal{L}$. According to the previous calculations, the number of functions that remain is at least:

$$r\left(\frac{\xi_d(k)}{2^k} - k - \frac{\xi_d(k)}{2^{k+1}} - k\binom{k-1}{2}\right) \geq r\left(\frac{\xi_d(k)}{2^{k+1}} - 2k\binom{k-1}{2}\right) \overset{(1)}{\geq}$$

[11] Notice that Condition 4 is only interesting for $k \geq 3$. When $k = 2$ we have that $|A| = 1$ and the condition is always held.

$$2r \cdot k \overset{(2)}{\geq} \max\{100\log(d), r+2\} \cdot k,$$

where inequality (1) is derived from Lemma 4.5 and inequality (2) holds since $r \geq 50\log(d) \geq 50$. This completes the proof of Lemma 4.6 and shows the existence of a (ξ_d, r, k)-splitting set for C'. $\square$

We now have the following corollary.

Corollary 4.10. *Let $r, d \in \mathbb{N}^+$ be such that $d \geq 2$ and $r \geq 50\log(d)$. Let $m \triangleq \max\{100\log(d), r+2\}$ and $\xi_d(k)$ be as in Equation (7). Let $s \leq k \in \mathbb{N}^+$ and $C' \triangleq \sum_{i=1}^{s} C_{f_i}$ be a minimal $\xi_d(k)$-distant $\Sigma\Pi\Sigma(s,d,r)$ circuit. Then there exists an m-linear function tree T with the following properties:*

- *The depth of T is lower than s.*
- *There exist $i' \in [s]$ such that T is an interpolation tree for $f_{i'}$.*
- *In every subspace V_u corresponding to a leaf u of T, we have that $f|_{V_u} = f_{i'}|_{V_u}$.*

4.1.2 Steps 2 & 3: Gluing the Restrictions Together

In this section we deal with the following problem: Let f be a t-variate polynomial and T be an m-linear function tree that is interpolating for f. Let $\{V_u\}_u$ be the set of subspaces corresponding to the leaves of T. Given the tree T and black-box access to the polynomials $\{f|_{V_u}\}_u$ we would like to reconstruct the circuit C_f (notice that f is actually the polynomial f_i described at the beginning of Section 4.1). We give an algorithm for the problem above. We first describe its general scheme: As a first step we construct, for each leaf u of the tree, the circuit $C_{f|_{V_u}}$ using the brute force algorithm of Appendix A.2. From here on, our methods are "local" when regarding the input tree. Specifically, let v be a vertex in the tree. Denote the children of v as $\{u_j\}_j$. We show how to construct the circuit $C_{f|_{V_v}}$ given the circuits $\{C_{f|_{V_{u_j}}}\}_j$. Using this "local construction method" we gradually construct for each vertex v the circuit $C_{f|_{V_v}}$ until we reach the root (in which we construct C_f). The goal of this section is to prove the following theorem.

Theorem 4.11. *Let f be a t-variate polynomial and T an m-linear function tree that is an interpolating tree for f. Then Algorithm 3, when given f and T as input, runs in time $\mathrm{size}(T) \cdot |\mathbb{F}|^{O(t^2)}$, and outputs C_f.*

Algorithm 2 is the "local" algorithm that in fact deals with an m-linear function tree of depth 1. It works in two stages. First we find the linear functions of $\gcd(C_f)$ (all linear functions dividing f) and then we construct $\mathrm{sim}(C_f)$. In the first stage (finding $\gcd(C_f)$) we find the restrictions $\{\gcd(C_f)|_{V_u}\}_u$ and glue them together using a method presented in [Shp]. In that paper, an algorithm for reconstructing a single (non-generalized) multiplication gate given its restriction to various co-dimension 1 subspaces is devised (see Appendix A.3). In the second stage we glue the different restrictions of $\{\mathrm{sim}(C_f)|_{V_u}\}_u$. For this we use the same gluing algorithm given in Section 4.2. In particular, we quote Theorem 4.18 of Section 4.2 and show that its requirements are satisfied, which guarantees that we can perform the gluing. Since we do not use any of the results of this section when analyzing the gluing algorithm of Section 4.2 this does not affect the correctness of the algorithms.

Lemma 4.12. *Algorithm 2 runs in $d^{O(t)}$ time, for $d = \max\{d_i\}$. If the given tree is interpolating for f and $f_i = f|_{\varphi_i=0}$ then the algorithms outputs C_f.*

Input: A set $\{\varphi_1, \ldots, \varphi_m\}$ of m linearly independent^H t-variate linear functions, over the field $\mathbb{F}$, defining an m-linear function tree of depth-1. In addition, for each $i \in [m]$, the circuit $C_{f_i} \triangleq p_i(L_1^i, \ldots, L_{r_i}^i) \prod_{j=1}^{d_i} l_j^i$.

Output: The circuit C_f in $\mathbb{F}^t$ that has the form $C_f = p(L_1(\bar{x}), \ldots, L_r(\bar{x})) \prod_{j=1}^{d'} l_j(\bar{x})$.

If for some $i, j \in [m]$ we have $r_i \neq r_j$, output "fail". Otherwise, define $r \triangleq r_1$;

Find a set of linear functions $l_1, \ldots, l_{d'}$ such that for each $i \in [m]$, $\prod_{j=1}^{d'} l_j|_{x_i=0} = \prod_{j=1}^{d_i} l_j^i$. If no such functions exist, output "fail";

Output $\gcd(C_f) = \prod_{j=1}^{d'} l_j$;

Find an integer $1 < \hat{i} \leq m$ for which $L_1^1|_{\varphi_{\hat{i}}=0}, \ldots, L_r^1|_{\varphi_{\hat{i}}=0}$ are linearly independent^H. If no such $\hat{i}$ exist, output "fail" ;

Define U_0 as the co-dimension 2 subspace where $\varphi_1 = \varphi_{\hat{i}} = 0$. Let U_1 be the co-dimension 1 subspace where $\varphi_1 = 0$ and U_2 be the co-dimension 1 subspace where $\varphi_{\hat{i}} = 0$;

Let h be the polynomial computed by $\mathrm{sim}(C_f)$. Run Algorithm 5 with input $\{C_{h|_{U_l}}\}_{l=0}^2$. Set $\mathrm{sim}(C_f)$ as the output of the algorithm ;

Algorithm 2: Reconstructing a circuit given a depth-1 m-linear function tree

Proof. Notice that the linear functions of the m-linear function tree given as input in Algorithm 2 are linearly independent^H. Hence, we will hereon assume w.l.o.g. that the functions in hand are $x_1, \ldots, x_m$. In particular, $f_i = f|_{x_i=0}$, for every $i \in [m]$. The proof has the following structure: First we show that in each co-dimension 1 subspace we are able to obtain the restrictions of $\gcd(C_f)$ and $\mathrm{sim}(C_f)$. We then prove that for every $i \in [m]$, we have that $f_i/\mathrm{Lin}(f_i)$ is a polynomial of exactly $\Delta(C_f)$ linear functions. Hence, Step 2 of the algorithm succeeds and $r = \Delta(C_f)$. We proceed to prove that $\gcd(C_f)$ is found correctly by using the results of [Shp], where an algorithm for Step 2 is given. To show that $\mathrm{sim}(C_f)$ is properly constructed we first show that for the polynomial h computed by $\mathrm{sim}(C_f)$, it holds that $C_h = \mathrm{sim}(C_f)$. We then prove that the index $\hat{i}$ exists and so the subspace U_0 can be easily found. Moreover, we show that it is $\Delta(C_h)$-rank preserving for C_h. Note that $C_h|_{U_0} = \mathrm{sim}(C_{f_1})|_{\varphi_{\hat{i}=0}}$, $C_h|_{U_1} = \mathrm{sim}(C_{f_1})$ and $C_h|_{U_2} = \mathrm{sim}(C_{f_{\hat{i}}})$. Given this, the results of section 4.2 guarantee the correctness of step 2 (see Theorem 4.18). We start by proving that Step 2 succeeds.

Lemma 4.13. *For each $i \in [m]$, the codimension 1 subspace defined by the equation $x_i = 0$ is $\Delta(C_f)$-rank preserving for* $\mathrm{sim}(C_f)$.

Proof. Let

$$\mathrm{sim}(C_f) = \tilde{p}(\tilde{L}_1, \ldots, \tilde{L}_{\Delta(C_f)}).$$

Assume for a contradiction that for some $i \in [m]$, the subspace where $x_i = 0$ is not $\Delta(C_f)$-rank preserving for $\mathrm{sim}(C_f)$. Then $\tilde{L}_1|_{x_i=0}, \ldots, \tilde{L}_{\Delta(C_f)}|_{x_i=0}$ must be linearly dependent^H. Hence, there exists a non-trivial combination of these functions that gives a constant function:

$$\sum_{j=1}^{\Delta(C_f)} \alpha_j \cdot \tilde{L}_j|_{x_i=0} = \gamma$$

for some $\alpha_1, \ldots, \alpha_{\Delta(C_f)}, \gamma \in \mathbb{F}$. The same combination of $\tilde{L}_1, \ldots, \tilde{L}_{\Delta(C_f)}$ cannot be constant since

these functions are linearly independent^H. Hence for some $0 \neq \beta \in \mathbb{F}$:

$$\sum_{j=1}^{\Delta(C_f)} \alpha_j \cdot \tilde{L}_j = \beta \cdot x_i + \gamma.$$

Therefore, $x_i \in \mathrm{Lin}(C_f)$, meaning that Condition 2 of Definition 4.2 is violated, contradicting the interpolating properties of the input tree. □

We continue with the proof and show how to obtain, in each co-dimension 1 subspace, the corresponding restrictions of $\gcd(C_f)$ and $\mathrm{sim}(C_f)$.

Lemma 4.14. *Let f be a non-zero t-variate polynomial under $\mathbb{F}$. Let $V \subseteq \mathbb{F}^t$ be a subspace that is $\Delta(C_f)$-rank preserving for $\mathrm{sim}(C_f)$ and $f|_V \neq 0$. Then*

$$\gcd(C_f)|_V = \gcd(C_{f|_V}) \text{ and } \mathrm{sim}(C_f)|_V = \mathrm{sim}(C_{f|_V}).$$

Proof. Let

$$\mathrm{sim}(C_f) = \tilde{p}(\tilde{L}_1, \ldots, \tilde{L}_{\Delta(C_f)}).$$

Let $g(\bar{x})$ be some irreducible factor of $\mathrm{sim}(C_f)$. Obviously, g is not a linear function. Let g' be a $\Delta(C_f)$-variate polynomial such that $g'(\tilde{L}_1, \ldots, \tilde{L}_{\Delta(C_f)}) = g$. Since V is $\Delta(C_f)$-rank preserving for $\mathrm{sim}(C_f)$ we get that

$$\dim_1(\mathrm{span}_1\{\tilde{L}_1|_V, \ldots, \tilde{L}_{\Delta(C_f)}|_V\}) = \Delta(C_f).$$

Thus, we have that $g|_V$ $(= g'(\tilde{L}_1|_V, \ldots, \tilde{L}_{\Delta(C_f)}|_V))$ is also an irreducible non-linear polynomial, meaning that $\mathrm{sim}(C_f)|_V$ does not have any linear factor. Therefore, $\gcd(C_f)|_V$ $(\equiv C_{f|_V}/\mathrm{sim}(C_f)|_V)$ contains all the linear factors of $f|_V$, indicating that

$$\gcd(C_f)|_V = \gcd(C_{f|_V}).$$

It follows that[12]

$$\mathrm{sim}(C_f)|_V = \mathrm{sim}(C_{f|_V}).$$

□

Since the given tree is interpolating for f, we have that $x_i \notin \mathrm{Lin}(C_f)$ for every $i \in [m]$. Therefore, $f|_{x_i=0} \neq 0$. Hence, by Lemmas 4.13 and 4.14 it holds that

$$\gcd(C_{f_i}) = \gcd(C_f)|_{x_i=0}, \quad \mathrm{sim}(C_{f_i}) = \mathrm{sim}(C_f)|_{x_i=0}. \tag{8}$$

Namely, if $\mathrm{sim}(C) = \tilde{p}(\tilde{L}_1, \ldots, \tilde{L}_{\Delta(C_f)})$, then

$$\mathrm{sim}(C_{f_i}) = \tilde{p}(\tilde{L}_1|_{x_i=0}, \ldots, \tilde{L}_{\Delta(C_f)}|_{x_i=0}).$$

Since $\tilde{L}_1|_{x_i=0}, \ldots, \tilde{L}_{\Delta(C_f)}|_{x_i=0}$ are linearly independent^H then for every $i \in [m]$, we have that $r_i = \Delta(C_f)$. Hence, the algorithm does not fail in Step 2 and $r = \Delta(C_f)$. We now prove that we can indeed reconstruct the linear functions of $\gcd(C_f)$. According to equation (8), the set of linear functions in $\gcd(C_{f_i})$ is the set of the projections of the linear functions in $\gcd(C_f)$ to the co-dimension 1 subspace

[12] Actually, we have shown that $\mathrm{sim}(C_{f|_V})$ and $\mathrm{sim}(C_f)|_V$ compute the same polynomial that does not have any linear factor. The representations of this polynomial may not be the same in both circuits. However, it is easy to see that given the "correct" definition of a default basis (used in the definition of default circuits), we can make sure that the representation is equal. We assume w.l.o.g. the equality of these circuits.

defined by $x_i = 0$. In [Shp] an algorithm (Algorithm 6 there) is given for exactly this problem (see Section A.3 in the appendix). Therefore, we can assume that we reconstructed $\gcd(C_f)$. We proceed to the second stage of Algorithm 2. The following lemma proves that there is an index $\hat{i}$ satisfying the requirement of Step 2.

Lemma 4.15. *There exist* $1 < \hat{i} \leq m$ *such that*

$$\dim_1 \left(\operatorname{span}_1 \left\{L_1^1|_{x_{\hat{i}}=0}, \ldots, L_r^1|_{x_{\hat{i}}=0}\right\}\right) = r.$$

Proof. Let $1 < i \leq m$. Assume that

$$\dim_1 \left(\operatorname{span}_1 \left\{L_1^1|_{x_i=0}, \ldots, L_r^1|_{x_i=0}\right\}\right) < r.$$

Then there exists a non-trivial linear combination of $L_1^1|_{x_i=0}, \ldots, L_r^1|_{x_i=0}$ that sums to some constant. Alternatively, for some multiset (containing at least one non-zero element) $\{\alpha_j\}_{j=0}^r$ in $\mathbb{F}$,

$$\alpha_0 + \sum_{j-1}^{r} \alpha_j L_j^1|_{x_i=0} = 0.$$

The same combination of $L_1^1, \ldots, L_r^1$ sums to x_i (up to some multiplicative constant). As $L_1^1, \ldots, L_r^1$ are linearly independent^H, we have that $x_i \in \operatorname{span}_1(\{L_1^1, \ldots, L_r^1\})$. Since $m \geq r+2$, we get by a simple dimension argument that there must exist some $1 < \hat{i} \leq m$ such that $x_{\hat{i}} \notin \operatorname{span}_1(\{L_1^1, \ldots, L_r^1\})$. For this $\hat{i}$ it holds that

$$\dim_1 \left(\operatorname{span}_1 \left\{L_1^1|_{x_{\hat{i}}=0}, \ldots, L_r^1|_{x_{\hat{i}}=0}\right\}\right) = r,$$

as required. □

Thus, the lemma implies that the subspace U_0 exists. We are now close to completing the proof of Lemma 4.12. In Step 2 we run Algorithm 5 of Section 4.2 on the different restrictions of the circuit C_h where h is the polynomial computed by $\operatorname{sim}(C_f)$. As h does not have any linear factors it follows that $C_h = \operatorname{sim}(C_f)$. Thus, it suffices to prove that the output of Algorithm 5 is the circuit C_h, given the restrictions $\{C_{h|_{U_l}}\}_{l=0}^2$. The requirement of Algorithm 5 is that U_0 should be $\Delta(C_h)$-rank preserving for C_h (see Theorem 4.18). Clearly, since $C_h = \operatorname{sim}(C_f)$, it holds that U_0 satisfies this requirement and thus step 2 outputs $C_h = \operatorname{sim}(C_f)$. This proves the algorithm correctness.

We now analyze the running time of the algorithm. By Lemma A.4, finding the linear functions of $\gcd(C_f)$ requires $\operatorname{poly}(d, t)$ time. The last step of the algorithm (constructing $\operatorname{sim}(C)$) requires $t \cdot d^{O(\Delta(C_h))} = d^{O(t)}$ time (this is shown in section 4.2, Theorem 4.18). It can easily be seen that the time required in all other steps is also $\operatorname{poly}(t, d, \operatorname{size}(C_f))$. Hence, the total running time of Algorithm is $d^{O(t)}$ (as $\operatorname{size}(C_f) = d^{O(t)}$). This completes the proof of Lemma 4.12. □

Now that we have analyzed Algorithm 2, we are ready to present Algorithm 3 that handles any m-linear function tree and not only a tree of depth 1.

We now give the analysis of Algorithm 3, thus proving Theorem 4.11.

Proof of Theorem 4.11. The proof of correctness follows by a simple induction. We analyze the running time of the algorithm. In total, Algorithm 2 is invoked $\operatorname{size}(T)$ times (once for every non-leaf vertex). Similarly, we see that Algorithm 11 is called as subroutine at most $\operatorname{size}(T) \cdot m$ times. Hence, according to Lemma A.3 and Lemma 4.12, the running time of Algorithm 3 is

$$d^{O(t)} \cdot \operatorname{size}(T) + |\mathbb{F}|^{O(t^2)} \cdot m \cdot \operatorname{size}(T).$$

Since $m \leq t$ (in a t-dimensional space there are at most t linearly independent^H linear functions) and $d \leq |\mathbb{F}|$ (otherwise we work with an algebraic extension of $\mathbb{F}$ containing more than d elements) the claim follows. □

Input: Tow integers $t, m \in \mathbb{N}^+$ and a t-variate polynomial f. An "m-linear function tree" T. In addition, for every leaf u of T, a black box computing $f|_{V_u}$.
Output: The circuit C_f.
if *T is a single node* **then**
  reconstruct C_f via Algorithm 11 of Section A.2 (brute force interpolation algorithm)
end
else
  For each child u of the root, recursively run with the subtree rooted at u and the black boxes corresponding to its leaves;
  Let $\{u_j\}_j$ be the children of the root. Let $\{\varphi_j\}_j$ be the linear functions labelling the edges connecting the root to its children. Run Algorithm 2 with input $\{\varphi_j\}_j$ and $\{C_{f|_{U_{u_j}}}\}_j$ to obtain C_f;
end

Algorithm 3: Reconstructing a circuit given an m-linear function tree

4.1.3 The Algorithm for finding a κ-distant circuit

We conclude Section 4.1 by giving its main algorithm. The following theorem concludes its analysis and this section

Theorem 4.16. *Let C be a t-variate minimal $\Sigma\Pi\Sigma(k, d, \rho)$ circuit computing a polynomial f. Let $r \in \mathbb{N}$ be such that $r \geq \max\{50\log(d), R(2k, d)\}$. Algorithm 4, given the inputs k, t, d, r and C runs in $|\mathbb{F}|^{O(kt^{k+1})}$ time. It outputs an integer $s \leq k$ and a $\xi_d(k)$-distant $\Sigma\Pi\Sigma(s, d, r)$ circuit computing f.*

Input: $k, t, d, r \in \mathbb{N}$ and oracle access to a $\Sigma\Pi\Sigma(k, d, \rho)$ circuit C in t indeterminates computing a polynomial f.
Output: $s \in \mathbb{N}^+$ such that $s \leq k$ and a $\xi_d(k)$-distant $\Sigma\Pi\Sigma(s, d, r)$ circuit $C' = \sum_{i=1}^{s} C_{f_i}$ computing f.
If $\Delta(C_f) \leq r$, output $s = 1$ and $C' = C_f$;
$m \stackrel{\Delta}{=} \max\{\lceil 100\log(d)\rceil, r + 2\}$;
foreach *"m-linear function tree" T of depth lower than k* **do**
  Run Algorithm 3 with inputs T and f;
  If the algorithm failed, or outputted a circuit C_g such that $\Delta(C_g) > r$ continue to the next tree;
  Recursively construct a $\xi_d(k)$-distant circuit with at most $k - 1$ multiplication gates computing the polynomial $f - f_1$. If it does not exist, continue to the next tree;
  Denote the found circuit as C_1 and the number of its multiplication gates by $\hat{s}$. Check whether $C_1 + C_{f_1}$ is a $\xi_d(k)$-distant $\Sigma\Pi\Sigma(\hat{s} + 1, d, r)$ circuit computing f. If so, output $s = \hat{s} + 1$ and $C' = C_1 + C_{f_1}$. Otherwise, continue to the next tree;
end
If no $\xi_d(k)$-distant circuit was found, output "fail";

Algorithm 4: Finding the $\xi_d(k)$-distant circuit of a polynomial

Proof. (of Theorem 4.16) We first note that there is a unique $\xi_d(k)$-distant $\Sigma\Pi\Sigma(s, d, r)$ circuit computing f. Indeed, as $\xi_d(k) \geq 2k + 1 \geq 2k + R(2k, d)/r = R(2k, d, r)/r$ (see Lemma 4.5), we get by Theorem 3.6 that if such a circuit exists then it is unique.

We start by proving the correctness of the algorithm. Note that before the algorithm outputs any circuit (Step 4) it verifies that it is indeed a $\xi_d(k)$-distant circuit for f so in any case when we output a circuit we are guaranteed to have the unique circuit at hand. So assume that for some $s \leq k$ there exists a $\xi_d(k)$-distant $\Sigma\Pi\Sigma(s,d,r)$ minimal circuit $C' = \sum_{i=1}^{s} C_{f_i}$ for f. We prove that for some $s' \leq k$, a $\xi_d(k)$-distant $\Sigma\Pi\Sigma(s',d,r)$ circuit is outputted by the algorithm, thus proving its correctness. Our proof is by induction on s. When $s = 1$, we clearly output $\{C_f\}$. Assume that $s > 1$. According to Corollary 4.10, for some tree that we check Algorithm 3 will produce, w.l.o.g., C_{f_1}. According to Theorem 3.6, the circuit $\sum_{i=2}^{s} C_{f_i}$ is the unique $\xi_d(k)$-distant $\Sigma\Pi\Sigma(s-1,d,r)$ circuit computing $f - f_i$. Hence, by the induction hypothesis, the recursive call will output $C_1 = \sum_{i=2}^{s} C_{f_i}$. Therefore, the next step of the algorithm will output the circuit C', as we wanted to prove.

We now analyze the running time. Our first action requires checking whether $\Delta(C_f) \leq r$. To do so we reconstruct C_f using Algorithm 11 (brute force algorithm) in $\mathbb{F}^{O(t^2)}$ time.[13] We now analyze the number of iterations in each recursive call. For this we just have to bound the number of m-linear functions tree. Notice that $\xi_d(k) \cdot r$ must be lower than t otherwise we can immediately output "fail" (since then there are no $\xi_d(k)$-distant $\Sigma\Pi\Sigma(s,d,r)$ circuits). Hence, $t = \Omega(\log(d) + r)$. The size of any set of linearly independent^H linear functions is at most t so $m \leq t$. Each tree has at most $O(m^k) = O(t^k)$ edges. For each edge there exists a t-variate linear function. The number of t-variate linear functions over the field $\mathbb{F}$ is at most $|\mathbb{F}|^{t+1}$. Hence, the number of trees we check is bounded by $|\mathbb{F}|^{O(t^{k+1})}$ and so is the number of iterations. In each iteration, besides the recursive call we construct a circuit given an m-linear function tree (Step 4) and preform a PIT test (Step 4). The former requires $|\mathbb{F}|^{O(t^2)}$ time, according to Theorem 4.11. The latter (PIT) may be done deterministically by brute force interpolation in $d^{O(t)}$ time. Hence, The time spent in each iteration not including the recursive call is $|\mathbb{F}|^{O(t^2)}$. Concluding, we get that the total time spent without including recursive calls is $|\mathbb{F}|^{O(t^{k+1})}$. The recursion depth is obviously bounded by k since we reduce its value by one in each call. Thus, the total running time is $|\mathbb{F}|^{O(k \cdot t^{k+1})}$. □

4.2 Gluing Together the Restrictions of a Low Rank Circuit

In this section we deal with the following problem: Let f be an n-variate polynomial of degree d over a field $\mathbb{F}$. Given the circuits $\{C_{f|_{V_i}}\}_i$ for various low dimensional subspaces, we would like to construct the circuit C_f. That is, we show how to "glue" together representations of the restrictions of f to various low dimension subspaces. The set of subspaces $\{V_i\}$ that we work with is such that each subspace V_i is $\Delta(C_f)$-rank preserving for C_f. The algorithm has two parts. First we find the linear functions in $\gcd(C_f)$ and then we find $\mathrm{sim}(C_f)$. Using the properties of rank preserving subspaces we manage to isolate the restriction of each linear function in $\gcd(C_f)$ to every subspace V_i. Having these restrictions, we reconstruc each linear function separately. In the second part (reconstruction of $\mathrm{sim}(C)$) we use a result of [Shp] where an algorithm for gluing together restrictions of a low rank circuit to various low dimensional subspaces, is given.

For convenience we denote $r \triangleq \Delta(C_f)$. We now describe the subspaces in which we receive the restrictions of f. One of the subspaces we have is contained in all other subspaces. We denote it by V. Define $t \triangleq \dim(V)$ and keep in mind that in our main algorithm, both r and t have small values (polylogarithmic in the circuit size). The following definition describes the entire set of subspaces $\{V_i\}$ and contains the notations used in this section:

Definition 4.17. *Let V be an affine subspace of dimension t.*

[13] There are more efficient ways to check whether $\Delta(C_f) \leq r$. However, this does not affect the analysis of the running time.

- *Denote with $\hat{V}$ the homogenous subspace of V and let v_0 be some fixed vector such that $V = \hat{V} + v_0$.*
- *Let $v_1, \ldots, v_n$ be a basis of $\mathbb{F}^n$ such that $v_1, \ldots, v_t$ is the default basis (as in definition 2.18) of $\hat{V}$ and $v_{t+1}, \ldots, v_n$ are the default basis to some complement subspace of $\hat{V}$ (that is, to some subspace of largest possible dimension that has trivial intersection with $\hat{V}$).*
- *For each $0 \leq i \leq n-t$, set $V_i \stackrel{\Delta}{=} \mathrm{span}_1(\hat{V} \cup \{v_{t+i}\}) + v_0$ (note that $V_0 = V$).*
- *Let $v_1^*, \ldots, v_n^*$ be the dual basis of $v_1, \ldots, v_n$. That is, each v_i^* is a linear function and $v_i^*(v_j) = 1$ if and only if $i = j$ (it is zero otherwise).*

The set of subspaces w.r.t. which we receive the restrictions of f is $\{V_i\}_{i=0}^{n-t}$. The following theorem summarizes the properties of Algorithm 5 (the gluing algorithm).

Theorem 4.18. *Let f be an n-variate polynomial over a field $\mathbb{F}$. Let V be an affine t-dimensional subspace of $\mathbb{F}^n$ (equivalently, V is of codimension $n-t$). Algorithm 5, given $\{C_{f|_{V_i}}\}_{i=0}^{n-t}$ as input (as defined in Definition 4.17), runs in time $O(n \cdot d^{\Delta(C_f)})$. If V is $\Delta(C_f)$-rank preserving for C_f then the algorithm outputs C_f.*

Proof of Theorem 4.18. We first note that if $V = V_0$ is r-rank preserving for C_f then for each $i \in [n-t]$ it holds that V_i is also r-rank preserving for C_f (since $V_0 \subseteq V_i$). Thus, according to Lemma 4.14, in each subspace V_i, we have that

$$\gcd(C_{f|_{V_i}}) = \gcd(C_f)|_{V_i} \text{ and } \mathrm{sim}(C_{f|_{V_i}}) = \mathrm{sim}(C_f)|_{V_i}.$$

Hence, the circuits $C_f|_{V_i}$ and $C_{f|_{V_i}}$ are identical. This gives us the method for obtaining the restrictions $\gcd(C_f)|_{V_i}$ and $\mathrm{sim}(C_f)|_{V_i}$ for every $0 \leq i \leq n-t$. We now deal with the different parts of Algorithm 5.
Part 1: Constructing $\gcd(C_f)$**:** Denote the distinct linear factors of $\gcd(C_f)$ by $\ell'_1, \ldots, \ell'_{d'}$ and let their multiplicities be $e_1, \ldots, e_{d'}$, respectively. Namely,

$$\gcd(C_f) = \prod_{j=1}^{d'} (\ell'_j)^{e_j}.$$

We shall think of those linear functions as represented with respect to the basis $v_1^*, \ldots, v_n^*$. That is, for each $i \in [d']$, we define:

$$\ell'_i = \alpha'_{i,0} + \sum_{j=1}^{n} \alpha'_{i,j} v_j^*.$$

The following lemma gives the analysis of part 1 of the algorithm.

Lemma 4.19. *Part 1 of Algorithm 5 runs in time $n \cdot poly(d)$. If V is r-rank preserving for C_f, it outputs* $\gcd(C_f)$.

Proof. The complexity analysis is trivial, so we only prove the correctness of the algorithm. Since all the subspaces are r-rank preserving, we have that $\gcd(C_f)|_{V_i} = \gcd(C_{f|_{V_i}})$ for every subspace V_i (Lemma 4.14). Furthermore, the restrictions of every two non-equivalent linear functions remain non-equivalent (Property 1 of Definition 2.5). It follows that[14] $\deg(\gcd(C_f)|_{V_i}) = \deg(\gcd(C_f))$. Hence, Step 5 of the algorithm does not fail. As no two linearly independent linear functions become linearly

[14] It is possible for exactly one linear function in $\gcd(C_f)$ to be restricted to a constant. This case can be easily dealt with. We assume for convenience and w.l.o.g. that no functions were restricted to constants.

Input: For each $0 \le i \le n-t$, the circuit $C_{f|_{V_i}}$: $p_i(L_1^{(i)}, \ldots, L_{r_i}^{(i)}) \cdot \prod_{j=1}^{d_i} (\ell_j^{(i)})^{e_j}$.

Output: The circuit C_f: $p(L_1, \ldots, L_r) \cdot \prod_{j=1}^{d'} \ell_j^{e_j}$.

`Part 1: Reconstructing` $\gcd(C_f)$

If there exist $0 \le i < i' \le n-t$ such that $d_i \ne d_{i'}$ output "fail". Define $d' = d_1$;

foreach *linear function* $\ell^{(0)}$ *dividing* $f|_V$*, with some multiplicity* e*,* **do**

In each $i \in [n-t]$ find a linear functions $\ell^{(i)}$ such that $\ell^{(i)} \in \gcd(C_{f|_{V_i}})$ and $\ell^{(i)}|_V = \ell^{(0)}$. If for some $i \in [n-t]$ no such linear function $\ell^{(i)}$ exists or is not unique then output "fail";

For each $0 \le i \le n-t$, denote $\ell^{(i)} = \alpha_{i,0} + \sum_{j=1}^{n} \alpha_{i,j} v_j^*$. Define $\ell \triangleq \alpha_{0,0} + \sum_{j=1}^{t} \alpha_{0,j} v_j^* + \sum_{j=1}^{n-t} \alpha_{j,j+t} v_{j+t}^*$;

Insert the linear function ℓ to $\gcd(C_f)$ with multiplicity e;

end

`Part 2: Reconstructing` $\mathrm{sim}(C_f)$

If there exist $0 \le i < i' \le n-t$ such that $r_i \ne r_{i'}$ output "fail". Define $r = r_1$;

foreach $i \in [n-t]$ **do**

find the unique $r \times r$ matrix that transforms the linear functions $L_1^{(i)}|_V, \ldots, L_r^{(i)}|_V$ into $L_1^{(0)}, \ldots, L_r^{(0)}$. If no such matrix exist output "fail";

transform the linear functions $L_1^{(i)}, \ldots, L_r^{(i)}$ and the polynomial p_i according to the found matrix so that for each $j \in [r]$ and $i \in [n-t]$ it will hold that $L_j^{(i)}|_V = L_j^{(0)}$ and $\hat{p} \triangleq p_0 = p_i$;

end

Find the unique set of linear functions $\hat{L}_1, \ldots, \hat{L}_r$ such that for every $j \in [r]$ and $0 \le i \le n-t$, we will have $L_j|_{V_i} = L_j^{(i)}$;

Set $L_1, \ldots, L_r$ as the default basis of $\mathrm{span}_1\{\hat{L}_1, \ldots, \hat{L}_r\}$. Construct the polynomial p such that $p(L_1, \ldots, L_r) = \hat{p}(\hat{L}_1, \ldots, \hat{L}_r)$;

output $C_f \triangleq p(L_1, \ldots, L_r) \cdot \gcd(C_f)$;

Algorithm 5: Gluing together low dimension restrictions of a low rank circuit

dependent when restricted to V we get that indeed there exists a unique $\ell^{(i)}$ such that $\ell^{(i)}|_V = \ell^{(0)}$. We now note that the function ℓ defined in Step 5 satisfy that $\ell|_{V_i} = \ell^{(i)}$ for every $0 \le i \le n-t$. By the structure of the V_i's it is clear that this ℓ is unique. In particular ℓ must belong to $\gcd(C_f)$ with multiplicity e as required. □

Part 2: Gluing the Restrictions of $\mathrm{sim}(C_f)$**:** In [Shp] an algorithm for exactly this task was given (Algorithm 4). Although it is not presented in this form, the Algorithm 4 of [Shp] can be seen as getting representations of the form $\mathrm{sim}(f)|_{V_i} = Q(\ell_1^{(i)}, \ldots, \ell_r^{(i)})$, where for each i and j, $\ell_j^{(i)}|_V = \ell_j$, and then computing $\tilde{\ell}_j$ such that for every i it holds that $\tilde{\ell}_j|_{V_i} = \ell_j^{(i)}$. This is exactly what we wish to achieve in Part 2 of our algorithm, and indeed the algorithm we give is exactly the algorithm (presented implicitly) of [Shp]. Thus correctness follow from the following lemma.

Lemma 4.20. *(implicit in [Shp]) Let h be a non-zero n-variate polynomial of degree d. Let $r \in \mathbb{N}^+$ be such that h is a polynomial of exactly r linear functions. Let V_0 be a subspace of $\mathbb{F}^n$ such that $h|_{V_0}$ is a polynomial of exactly r linear functions. Algorithm 4 of [Shp], given the input $\{h|_{V_i}\}_{i=0}^{n-t}$ outputs a representation of h as a polynomial of r linear functions in $O(n \cdot d^r)$ time.*

By setting h as $f/\mathrm{Lin}(f)$, we clearly satisfy the requirements of Lemma 4.20. Hence, part 2 of Algorithm 5 (that acts exactly as Algorithm 4 of [Shp]) outputs the circuit $\mathrm{sim}(C_f)$ in $O(n \cdot d^r)$ time. Theorem 4.18 follows from this and from Lemma 4.19. □

4.3 The Reconstruction Algorithm

We are now ready to summarize our findings and present the learning algorithm. Algorithm 6, given the inputs $k, n, d \in \mathbb{N}$ and black box access to an n-input $\Sigma\Pi\Sigma(k, d, \rho)$ circuit C, outputs a $\Sigma\Pi\Sigma(s, d, r)$ circuit C' computing the same polynomial as C. Its running time is quasi-polynomial in $n, d, |\mathbb{F}|$. The following lemma gives the summary of the algorithm analysis. Theorem 1 is immediately implied by it

Lemma 4.21. *Let $k, n, d, \rho \in \mathbb{N}$ and let C be an n-variate $\Sigma\Pi\Sigma(k, d, \rho)$ circuit over the field $|\mathbb{F}|$. Then Algorithm 6, given a black box computing C as input, outputs for some $r \le \max\{50\log(d), R(2k, d), R(k+1, d) + k \cdot \rho\} \cdot (\xi_d(k) \cdot k)^k$ and $s \le k$, a $\Sigma\Pi\Sigma(s, d, r)$ circuit C' such that $C' \equiv C$. The running time of the algorithm is*

$$n^2 \cdot |\mathbb{F}|^{O\left(\left(\max\{50\log(d), R(2k,d), R(k+1,d)+k\cdot\rho\}\cdot((\xi_d(k)+1)\cdot k)^{k-2}\cdot\xi_d(k)\right)^{k+1}\right)} =$$

$$poly(n) \cdot \exp(\log(|\mathbb{F}|) \cdot \log(d)^{O(k^3)} \cdot \rho^{O(k)}).$$

Proof. (of Lemma 4.21) We first prove the correctness of the algorithm. Notice that before we output any circuit we verify that it computes the correct polynomial. Hence, it suffices to prove that there exists at least one pair of (r, α) for which a circuit is outputted.

Let f be the polynomial computed by the input circuit C. In the main iteration, the integer r takes each value between r_{init} and $r_{init} \cdot k^{\lceil \log_k(\xi_d(k)+1)\rceil \cdot (k-2)}$. According to Theorem 3.2, there exist at least one such value r for which there exist an integer $s' \le k$ and a minimal $\xi_d(k)$-distant $\Sigma\Pi\Sigma(s, d, r)$ circuit C' computing f. We focus on the iterations in which r takes a minimal such value.

According to Lemma 2.8, there exist $\alpha \in S$ for which the chosen subspace $V_{\alpha,t}$ is t-rank preserving for C'. We focus on the iteration where such an element α is chosen along with the specified value of

Input: $k, n, d, \rho \in \mathbb{N}$ and a black box holding an n-variate $\Sigma\Pi\Sigma(k, d, \rho)$ circuit C.
Output: A $\Sigma\Pi\Sigma(s, d, r)$ circuit C' computing the same polynomial as C.
$r_{init} \triangleq \max\{50\log(d), R(2k, d), R(k+1, d) + k \cdot \rho\}$;
Let $S \subseteq \mathbb{F}$ be such that $|S| = n\left(\binom{dk}{2} + 2^k\right)\binom{r_{init} \cdot k^{\lceil \log_k(\xi_d(k)+1)\rceil \cdot k} + 2}{2} + 1$;
foreach $r_{init} \leq r \leq r_{init} \cdot k^{\lceil \log_k(\xi_d(k)+1)\rceil \cdot (k-2)}$ *and* $\alpha \in S$ **do**

- Set $t \triangleq r \cdot \xi_d(k)$;
- Define $V \triangleq V_{\alpha,t}$. Let $\hat{V}$ be the homogenous part of V, and $v_1, \ldots, v_t$ be the default basis of $\hat{V}$. Also let v_0 be such that $V = \hat{V} + v_0$. Let $v_{t+1}, \ldots, v_n$ be the default basis for some V' satisfying $V' \oplus \hat{V} = \mathbb{F}^n$;
- For each $j \in [n-t]$, define $V_j \triangleq \mathrm{span}(\hat{V} \cup \{v_{t+j}\}) + v_0$;
- For each $0 \leq j \leq n-t$ run Algorithm 4 on the circuit $C|_{V_j}$. If the algorithm failed in any of the restrictions then proceed to the next pair of (r, α), otherwise, for every j, define the outputted circuit as $\sum_{i=1}^{s_j} C_{f_i^j}$;
- If for any $0 \leq j_1 < j_2 \leq n-t$, $s_{j_1} \neq s_{j_2}$ then proceed to the next pair of (r, α). Otherwise define $s \triangleq s_1$;
- Reorder the numbering of the multiplication gates so that for every $i \in [s]$ and $j \in [n-t]$ it holds that $f_i^j|_V = f_i^0$. If this is not possible then output "fail" ;
- For every $i \in [s]$, run Algorithm 5 with input $\{C_{f_i^j}\}_{j=0}^{n-t}$. If the algorithm failed for any $i \in [s]$ then proceed to the next pair of (r, α). Otherwise, for each $i \in [s]$ set C_{f_i} to be the circuit found by Algorithm 5, given i;
- If $\sum_{i=1}^{s} C_{f_i} \equiv C$, output $\sum_{i=1}^{s} C_{f_i}$. Otherwise, proceed to the next pair of (r, α);

end

Algorithm 6: Learning a $\Sigma\Pi\Sigma(k, d, \rho)$ circuit

r. Let $C' \triangleq \sum_{i=1}^{s'} C_{f_i'}$ be a minimal $\xi_d(k)$-distant $\Sigma\Pi\Sigma(s,d,r)$ circuit computing f whose existence stems from the choice of r. Notice that,

$$R(2k,d,r)/r = R(2k,d)/r + 2k \overset{(1)}{\leq} 2k+1 \overset{(2)}{\leq} \xi_d(k)$$

Inequality 1 holds since $r \geq R(2k,d)$. Inequality 2 follows from Lemma 4.5. Hence, by Corollary 3.7, the circuit $C'|_V$ is a $\xi_d(k)$-distant $\Sigma\Pi\Sigma(s,\deg(f|_V),r)$ minimal circuit computing $f|_V$. In addition, by the same corollary, there is no other $\xi_d(k)$-distant $\Sigma\Pi\Sigma(k',\deg(f|_V),r)$ minimal circuit computing $f|_V$ for any $k' \leq k$. Therefore, for each subspace V_j, the circuit reconstructed in Step 6 is in fact $C'|_{V_j}$. In particular, all the s_j's are the same (and equal to s as defined in Step 6) and for each $i \in [s]$ and $0 \leq j \leq n-t$ it holds, after Step 6, that $C_{f_i^j} = C_{f_i'}|_{V_j}$. Since each V_j is r-rank preserving for C' we have by Lemma 4.14 that $C_{f_i'}|_{V_j} = C_{f_i'|_{V_j}}$. From this and from V being t-rank preserving for each $C_{f_i'}$, it follows by Theorem 4.18 that in Step 6, for every $i \in [s]$ Algorithm 5 outputs the circuit $C_{f_i'}$ (that is, for every $i \in [s]$, $f_i = f_i'$).

We have shown so far that for some pair of (r,α), in Step 6 we reconstruct the circuits $\{C_{f_i'}\}_{i=1}^{s'}$. Hence, the circuit we check in the following step is C'. As $C' \equiv f$, the algorithm outputs it as the required circuit. This proves the correctness of Algorithm 6.

We now analyze the complexity of the algorithm. The number of iterations we have is

$$|S| \cdot 2^{O(k^3)} \cdot (\log(d))^{O(k^2)} \cdot \rho = n \cdot \mathrm{poly}(d,\rho) \cdot 2^{O(k^3)} \cdot (\log(d))^{O(k^2)}$$

The only steps whose running time analysis are neither trivial nor already analyzed (in previous sections) are steps 6 and 6. In Step 6, by performing a brute force PIT we reach a running time bound of

$$n \cdot s^2 \cdot d^{O(t)} \leq n \cdot |\mathbb{F}|^{O(t^{k+1})}.$$

In Step 6, we would like to deterministically check whether the $\Sigma\Pi\Sigma(s+k,d,r)$ circuit $C-C'$ computes the zero polynomial given only a black box computing the circuit. The results of [KS08], combined with [SS08], give such an algorithm whose running time

$$n \cdot \exp\left(k^3 \cdot (\log d) + kr \log d\right)$$

(see Lemma A.5 of Section A.4). Recall that

$$t^{k+1} = (\xi_d(k) \cdot r)^{k+1} = \Omega\left(k^3 \cdot (\log d) + kr \log d\right).$$

Therefore, assuming that $d \leq |\mathbb{F}|$, the running time of is Step 6 is in fact

$$n \cdot |\mathbb{F}|^{O(t^{k+1})}.$$

We have that the total time of each iteration, according to Theorems 4.16 and 4.18 is

$$n \cdot |\mathbb{F}|^{O(t^{k+1})} + n \cdot d^{O(r)} = n \cdot |\mathbb{F}|^{O(t^{k+1})}.$$

Hence, the total running time of the algorithm is

$$\left(n \cdot \mathrm{poly}(d,\rho) \cdot 2^{O(k^3)} \cdot (\log(d))^{O(k^2)}\right) \cdot \left(n \cdot |\mathbb{F}|^{O(t^{k+1})}\right) = n^2 \cdot |\mathbb{F}|^{O(t^{k+1})}$$

This proves Lemma 4.21. □

5 Reconstructing Multilinear $\Sigma\Pi\Sigma(k)$ circuits

When reconstructing multilinear $\Sigma\Pi\Sigma(k)$ circuits we obtain much better results. We present an algorithm that given black box access to some n-variate multilinear $\Sigma\Pi\Sigma(k)$ circuit C, deterministically outputs, in $\text{poly}(n, |\mathbb{F}|)$ time, a multilinear $\Sigma\Pi\Sigma(k)$ circuit C' computing the same polynomial as C.

The main outline of the reconstruction algorithm is the same as in $\Sigma\Pi\Sigma(k, d, \rho)$ circuits. We first reconstruct the restriction of the circuit to several low dimensional subspaces and then glue together the different restricted circuits. As before, the representation of the circuit (i.e., a *canonical κ-distant circuit*) over the low dimensional subspaces (detailed in Section 5.1) ensure a small number of possible "lifts" of the circuit.

Specifically, we reduce the case of lifting a multilinear $\Sigma\Pi\Sigma(k)$ circuit to the case of lifting a $\Sigma\Pi\Sigma(k')$ circuit (for some $k' \leq k$) of low Δ measure. This gives us a polynomial size list of circuits (the way to obtain this list is given in Section 5.2) containing at least one circuit computing the same polynomial as C. Using a deterministic *Polynomial Identity Test* [SV09] we find the desired circuit.

There are two main differences from the general case. The first is that we need to make sure that the circuit $C|_V$ is multilinear. We achieve this by defining a different set of rank-preserving subspaces for multilinear circuits. The second difference, which is the main reason that the algorithm is more efficient now, is is the main reason that the algorithm is more efficient now, is rank-preserving subspace can be a constant. Thus, we shall restrict our multilinear circuit to a constant dimensional subspace. As C is multilinear it follows that $C|_V$ is of constant size and thus the time required to go over all possible multilinear circuits equivalent to $C|_V$ is polynomial in $|\mathbb{F}|$. Hence, given the existence of some "liftable" representation of $C|_V$, we can simply try to lift each possible representation until we succeed.

The questions remaining is what representation do we use and how do we lift it? Assume first that the Δ measure of C (i.e., the rank of the simplified circuit) is low. Notice that this case is somewhat analogous to the case of only one multiplication gate when dealing with $\Sigma\Pi\Sigma(k, d, \rho)$ circuits. Since we are looking for a $\Sigma\Pi\Sigma(k)$ circuit and not a single multiplication gate, this task is not as simple as before. The main problem arising is that many different lifts might compute the same polynomial. Hence, we must make sure that the number of (lifted) circuits computing the same polynomial is small. In Section 5.2 we give an algorithm that reconstructs a low Δ measured multilinear $\Sigma\Pi\Sigma(k)$ circuit given black boxes computing its restrictions in several low dimension subspaces. As in the previous section, the subspaces we work with contain a low dimension subspace V and several other subspaces containing it. For each possible multilinear $\Sigma\Pi\Sigma(k)$ circuit C' equivalent to $C|_V$, we obtain, by using the fact that C is a low Δ measured circuit, a single "guess" of an n-variate $\Sigma\Pi\Sigma(k)$ circuit. We prove that at least one of our guesses is correct (i.e., is equivalent to C). Since there is only a polynomial number of circuits equivalent to $C|_V$ in V, the total number of "guesses" is of polynomial size.

When the Δ measure of C is high, we reduce our problem the low Δ measure problem using a unique partition of the circuit into low Δ measured subcircuit. In Section 5.1, we present a method to find integers κ and r that ensure the existence of the a (κ, r)-*strong partition* (as defined in Section 3) $C_1, \ldots, C_s$ of C, with the following property: For each multilinear-rank preserving subspace V, and multilinear circuit C' s.t. $C' \equiv C|_V$, there is a unique (κ, r)-strong partition $C'_1, \ldots, C'_{s'}$ of C' that is equivalent to the restriction of the partition of C. Namely, $s = s'$ and for each $1 \leq i \leq s$, $C_i|_V \equiv C'_i$. Hence, given a circuit $C' \equiv C|_V$, by using Algorithm 1 of Section 3, we obtain some equivalents of $C_1|_V, \ldots, C_s|_V$ as required.

In the final section we gather our findings and present the main algorithm for reconstructing a multilinear $\Sigma\Pi\Sigma(k)$ circuit.

5.1 Finding a Partition of the Circuit in a Low Dimension Subspace

This section includes results analogous to sections 3 and 4.1. Given a multilinear $\Sigma\Pi\Sigma(k)$ circuit C and a low dimensional subspace V we show how to find a (κ, r)-strong partition (recall Definition 3.3 of Section 3.1) of $C|_V$ to subcircuits. We then prove that the partition we find is equal to the restriction of some specific (predetermined) (κ, r)-strong partition of C (namely, each subcircuit in the partition that we found is the restriction of some subcircuit from the predetermined partition).

The way we find the partition of $C|_V$ is very simple. In the multilinear case V is of dimension $O(1)$ and as the degree of a multilinear circuit is bounded by its rank, we get that $C|_V$ is computable by a multilinear $\Sigma\Pi\Sigma(k)$ circuit of a constant size. Hence, by using brute force techniques, we can construct a multilinear $\Sigma\Pi\Sigma(k)$ circuit computing the same polynomial as $C|_V$ in time poly($|\mathbb{F}|$). We then run Algorithm 1 (see Section 3.1) to find the required partition. The main result of this section shows that the partition found above is equivalent to the restriction of the partition that Algorithm 1 finds when given as input the circuit C. Specifically, we show the following: Assume that given C, κ and r_{init} as input Algorithm 1 outputs the (κ, r)-strong partition $C_1, \ldots, C_s$ of C. Then for any $\Sigma\Pi\Sigma(k)$ circuit C' such that $C' \equiv C|_V$, algorithm 1, given the input C', κ and r_{init}, outputs a (κ, r)-strong partition $C'_1, \ldots, C'_s$ such that there exist a reordering of $C'_1, \ldots, C'_s$ where for each $i \in [s]$, $C'_i \equiv C_i|_V$.

To prove the result we shall need to analyze the running time of Algorithm 1 and discuss some of its properties. Note that in Section 4 we only used the existence of a partition and did not run Algorithm 1 so we didn't need to analyze its running time there. For convenience, we repeat the notations and analysis of Algorithm 1 given in Section 3.1:

- Let $n, k, r_{init}, \kappa \in \mathbb{N}$, and C be a n-variate multilinear $\Sigma\Pi\Sigma(k, d)$ circuit.
- Algorithm 1 outputs an integer $r_{init} \leq r \leq k^{(k-2)\cdot\lceil\log_k(\kappa)\rceil} \cdot r_{init}$ and a partition I of $[k]$ corresponding to a (κ, r)-strong partition of C.
- Denote by m the total number of iterations preformed by the algorithm.
- $\iota \triangleq \lceil\log(\kappa)\rceil$.

Since we are about to use the algorithm it is essential to discuss its running time:

Lemma 5.1. *Algorithm 1 finds a (κ, r) strong partition of $[k]$ in time $O(\log(\kappa) \cdot n^3 k^4)$.*

Proof. In each iteration of the main loop, all actions except the inner loop require time $O(k^2)$. The inner loop has $O(k^2)$ iterations in which we check the Δ measure of a circuit having at most nk linear functions. To check the rank of a circuit we preform a gaussian elimination to the matrix of coefficients of the linear functions in it. This requires $O(n^3 k)$ time (there are faster algorithms of course but it this will not affect the total running time). Hence, the total running time of a single iteration in the main loop is $O(n^3 k^3)$.

In every ι iterations, the number of elements in I is reduced by at least one. The number of elements in I begins with k and ends with at least 1. Hence, the number of iterations is at most $\iota \cdot (k-1)$. Thus, the total running time is $O(\iota \cdot n^3 k^4) = O(\log(\kappa) \cdot n^3 k^4)$. □

To prove the uniqueness of the partition we first analyze the Δ measure of the subcircuits and their multiplication gate. The following lemma uses the notations of Algorithm 1 (most importantly, m and r_m were defined there).

Lemma 5.2. *Let $C_1, \ldots, C_s$ denote the subcircuits of C induced by the partition found in Algorithm 1. If $\iota \geq 2$ then we have the following: For every $i \neq i' \in [s]$ and any two multiplication gates M, M' in the circuits $C_i, C_{i'}$ (respectively), it holds that $\Delta(M, M') \geq r_{m-1}$.*

Proof. Let $C_i = M_1 + \ldots + M_l$ (where each M_j is a multiplication gate) and $C_{i'} = M'_1 + \ldots + M'_{l'}$ be two different subcircuits in the partition. Assume w.l.o.g. and for a contradiction that $\Delta(M_1, M'_1) < r_{m-1}$. Then,

$$\forall j_1, j_2 \in [l], \;\; \Delta(M_{j_1}, M_{j_2}) \overset{(1)}{\leq} \Delta(C_i) \overset{(2)}{\leq} r = r_{m-\iota} \leq r_{m-2}.$$

Inequality 1 holds according to Lemma 2.17. Since the partition is (κ, r)-strong (Lemma 3.4), inequality 2 holds as well. Analogically, we have that

$$\forall j'_1, j'_2 \in [l'], \;\; \Delta(M'_{j'_1}, M'_{j'_2}) \leq r_{m-2}.$$

Hence,

$$r_m \overset{(1)}{\leq} \Delta(C_i, C_{i'}) = \Delta\left(M_l, M_{l-1}, \ldots, M_1, M'_1, M'_2, \ldots, M'_{l'}\right) \overset{(2)}{\leq}$$

$$\sum_{j=2}^{l} \Delta(M_j, M_{j-1}) + \sum_{j=2}^{m'} \Delta(M'_j, M'_{j-1}) + \Delta(M_1, M'_1) \leq$$

$$r_{m-1} + (k-1) \cdot r_{m-2} < 2 \cdot r_{m-1} \leq k \cdot r_{m-1} = r_m$$

Note that inequality 1 holds as otherwise the m'th iteration would not have been the last one (contradicting the definition of m). Inequality 2 follows from Lemma 2.17. This proves the claim. □

Theorem 5.3. *Let $r_{init}, \kappa \in \mathbb{N}^+$ and let C and C' be two minimal multilinear $\Sigma\Pi\Sigma(k)$ circuits computing the same non-zero polynomial.*

- *Let $C_1, \ldots, C_s$ and $C'_1, \ldots, C'_{s'}$ ($s' \geq s$) be the partitions of C and C' found by Algorithm 1 when given κ, r and C (or C') as input.*
- *Assume that $r_{init} \geq R_M(2k)$ and that $\kappa > k^2$ (that is, $\iota \geq 3$).*

Then, $s = s'$ and there exists a reordering of the subcircuits such that for each $i \in [s]$, it holds that $C_i \equiv C'_i$.

Proof. Consider the circuit $C' - C$. Clearly it computes the identically zero polynomial. Let M be a multiplication gate in C_1. M also appears in some minimal zero subcircuit of $C' - C$. Denote this minimal zero subcircuit with $\tilde{C}$. Note that $\tilde{C}$ must contain at least one multiplication gate from C' as otherwise we will get that C is not minimal, in contradiction. Assume w.l.o.g. that there exists a multiplication gate M' in $\tilde{C}$ that was "originally" a multiplication gate of C'_1. Then for each[15] $\hat{M} \in \tilde{C}$, we have that

$$\Delta(\hat{M}, M) \leq^* \Delta(\tilde{C}) < R_M(2k) \leq r_{init} \leq r_{m-1}, \quad \Delta(\hat{M}, M') \leq \Delta(\tilde{C}) < r_{m-1},$$

where equality $*$ holds as $\tilde{C}$ computes the zero polynomial. Therefore, by Lemma 5.2 either $\hat{M} \in C_1$ or $\hat{M} \in C'_1$. It follows that each minimal identically zero subcircuit of $C' - C$ is contained in the union of exactly two subcircuits, one from C and one from C'. In other words, we have shown that for a minimal subcircuit $\tilde{C}$ of $C' - C$, there exist $i \in [s]$ and $i' \in [s']$ such that $\tilde{C} \cap C \subseteq C_i$ and $\tilde{C} \cap C' \subseteq C'_{i'}$.

It is now left to prove that each two minimal zero circuits of $C' - C$ are either composed of multiplication gates of the same pair of subcircuits (of C and C') or from distinct pairs of subcircuits. Formally, we must prove the following claim: Let $\tilde{C}, \hat{C}$ be two different minimal zero circuits of $C' - C$.

[15] For convenience we abuse notations and sometimes refer to a circuit as a set of multiplication gates.

Assume that $\tilde{C} \cap C_1 \neq \emptyset$, $\hat{C} \cap C_1 \neq \emptyset$ and that $\tilde{C} \cap C_1' \neq \emptyset$. Then $\hat{C} \cap C_1' \neq \emptyset$. Proving this will show that $C_1 - C_1'$ is a sum of identically zero circuits.

Let $N \in \hat{C} \cap C_1$, $M \in \tilde{C} \cap C_1$, $M' \in \tilde{C} \cap C_1'$ and $N' \in \hat{C} \cap C'$. We will show that $N' \in \hat{C} \cap C_1'$. The following inequalities hold:

$$\Delta(M', M) \leq \Delta(\tilde{C}) < R_M(2k) \leq r_{init} \leq r_{m-\iota} \leq r_{m-3},$$

$$\Delta(M, N) \leq \Delta(C_1) \leq r_{m-\iota} \leq r_{m-3}, \quad \Delta(N, N') \leq \Delta(\hat{C}) < R_M(2k) \leq r_{m-3}.$$

Hence, by Lemma 2.17 we get that

$$\Delta(M', N') \leq \Delta(M', M) + \Delta(M, N) + \Delta(N, N') < 3 \cdot r_{m-3} \leq k^2 \cdot r_{m-3} = r_{m-1}.$$

Since $M' \in C_1'$, we have by Lemma 5.2 that $N' \in C_1'$. It follows that $C_1 - C_1'$ is a sum of identically zero circuits, i.e., $C_1 \equiv C_1'$. Analogically, we have w.l.o.g. that $C_i \equiv C_i'$ for all $i \in [s]$. Furthermore, if $s < s'$ then C' is not minimal, meaning that $s = s'$. □

Corollary 5.4. *Let C be a non-zero minimal multilinear $\Sigma\Pi\Sigma(k)$ circuit:*

- *Let $r_{init}, \kappa \in \mathbb{N}^+$ be such that $r_{init} \geq R_M(2k)$ and $\kappa > k^2$.*
- *Let $C_1, \ldots, C_s$ be the partition of C outputted by Algorithm 1 when given C, r_{init} and κ as input.*
- *Let V be an $(r_{init} \cdot k^{(k-1)\cdot\lceil\log_k(\kappa)\rceil})$-multilinear-rank-preserving subspace for C.*
- *Let C' be a $\Sigma\Pi\Sigma(k)$ circuit in $\dim(V)$ indeterminates such that $C' \equiv C|_V$.*
- *Let $C_1', \ldots, C_{s'}'$ be the partition of C' outputted by Algorithm 1 when given C', r_{init} and κ as input.*

Then, $s = s'$ and there exists a reordering of the subcircuits such that for each $i \in [s]$, it holds that $C_i|_V \equiv C_i'$.

Proof. According to Theorem 5.3, it suffices to show that Algorithm 1, given $C|_V$, r_{init} and κ as input outputs $C_1|_V, \ldots, C_s|_V$. Let r be the integer outputted by Algorithm 1 when given C, r_{init} and κ as input. We have that $r \leq r_{init} \cdot k^{(k-2)\cdot\lceil\log_k(\kappa)\rceil}$. Hence V is $(r \cdot k^{\lceil\log_k(\kappa)\rceil})$-multilinear-rank-preserving for C. It follows that for every subcircuit $\tilde{C}$ of C and $\tilde{r} \leq k^{\lceil\log_k(\kappa)\rceil} \cdot r$, it holds that

$$\Delta(\tilde{C}) \leq \tilde{r} \Leftrightarrow \Delta(\tilde{C}|_V) \leq \tilde{r}.$$

Hence, by uniqueness of the partition found by Algorithm 1 (Theorem 3.6),[16] we get that the two partitions are equal (up to reordering of the indices). This proves the claim. □

5.2 Lifting a Low Rank Multilinear $\Sigma\Pi\Sigma(k)$ Circuit

In this section we give an algorithm that given restrictions, of a low Δ measured circuit, to various low dimension subspaces, glues these representations to one n-variate multilinear $\Sigma\Pi\Sigma(k)$ circuit computing the same polynomial as the original circuit. As in the previous section we saw how to find a strong partition of the circuit in a low dimensional subspace, we can combine the two results to get an algorithm for reconstructing the original circuit.

[16] We proved uniqueness in the case that $\kappa \geq R(2k, d, r)/r$, but in the case of multilinear circuit it is not difficult to see that $\kappa \geq R_M(2k)/r$ is sufficient to guarantee uniqueness.

More formally, the problem we solve is the following: Let $k, r, n \in \mathbb{N}$. Let C be an n-variate multilinear $\Sigma\Pi\Sigma(k)$ circuit computing a polynomial f such that $\Delta(C) \leq r$. Let $B \subseteq [n]$ and $\alpha \in \mathbb{F}$ be such that $V_{B,\alpha}$ is a liftable r-multilinear-rank-preserving t dimensional subspace for C (recall Definition 2.13). Given black boxes computing the polynomial $f|_{V_{B',\alpha}}$ for each $B' \supseteq B$ of size $|B'| = |B| + 2$, we would like to construct an n-variate $\Sigma\Pi\Sigma(k)$ multilinear circuit C' that computes the polynomial f. For convenience, we set $V \triangleq V_{B,\alpha}$. For each $A \subseteq [n]$ let

$$V^A = V^A_{B,\alpha} \triangleq V_{B \cup A,\alpha}. \tag{9}$$

We assume w.l.o.g. that the set B is in fact the set $\{n-t+1, n-t+2, \ldots, n\}$ and that the shifting vector $v_{\alpha,0} = \{0, 0, \ldots, 0\}$ (that is $\alpha = 0$).

As a first step towards computing a $\Sigma\Pi\Sigma(k)$ circuit for f we first construct a $\Sigma\Pi\Sigma(k)$ circuit computing a restriction of f to V. Notice that since $C|_V$ is a multilinear $\Sigma\Pi\Sigma(k)$ circuit in t variables, its size is bounded from above by some integer function of t and k. Hence, by going over all circuits bounded by that size restriction we will at some point find $C|_V$. This "pool of circuits" is of size $\text{poly}(|\mathbb{F}|)$, assuming that k and t are constants. Therefore, in the first model we work on, the circuit $C|_V$ is given to us as input while in practice we try to lift the circuit for each guess of $C|_V$.

The lifting process consists of two phases. First we find the linear functions of $\gcd(C)$ and then, having access to $\text{sim}(C)$ we find a simple circuit C' such that $C' \equiv \text{sim}(C)$. Algorithm 7 constructs $\gcd(C)$.

Input: The circuit $C|_V$, where V is a liftable r-multilinear-rank-preserving t dimensional subspace for C. Black boxes computing the circuits $C|_{V^{\{i,j\}}}$ for each $i \neq j \in [n-t]$.
Output: $\gcd(C)$.
$S \leftarrow [n-t]$;
$\mathcal{L} \leftarrow \gcd(C|_V)$;
while $S \neq \emptyset$ **do**
  Pick some $i \in S$ and remove it from S;
  In a brute force manner, look for a linear function ℓ such that x_i appears in ℓ and ℓ is a factor of $(C|_{V^{\{i\}}})$. If no such function exists or ℓ contains a variable x_j that appears in $\text{sim}(C|_V)$ then continue to the next iteration;
  Assuming that a linear function ℓ was found, find, for each $j \in S$, a linear function ℓ^j dividing $C|_{V^{\{i,j\}}}$ such that $\ell^j|_{V^{\{i\}}} = \ell$. Denote by α_j the coefficient of x_j in ℓ^j. If $\alpha_j \neq 0$, then remove j from S;
  Add the linear function $\ell + \sum \alpha_j x_j$ to $gcd(C)$;
  $\mathcal{L} \leftarrow \mathcal{L} \backslash \{\ell|_V\}$;
end
Add all functions left in $\mathcal{L}$ to $\gcd(C)$;

Algorithm 7: Lifting the g.c.d of a low Δ measured circuit

Lemma 5.5. *Algorithm 7 outputs* $\gcd(C)$ *in* $n^2 \cdot |\mathbb{F}|^{O(t)}$ *time.*

Proof. To prove the correctness of the algorithm we show that each linear function in $\gcd(C)$ is found exactly once. Clearly, each linear function $\ell \in \gcd(C)$ such that $\ell|_V$ is not constant either belongs to $\mathcal{L}$ or is found in the main iteration. We prove that each linear function that is constant on V is found in the main iteration and that no unnecessary function is added to $\gcd(C)$.

Step 7 is meant to find whether x_i appears in $\gcd(C)$. The other steps are reached only when x_i appears in $\gcd(C)$. In these steps we find the linear function containing x_i (there is only one such

function since the circuit is multilinear). To prove the correctness of step 7, we use the following two lemmas stating sufficient and necessary conditions for the appearance of a variable and of a linear function in $\gcd(C)$:

Lemma 5.6. *Let $i \in [n-t]$. Then x_i appears in $\gcd(C)$ if and only if x_i appears in $\gcd(C|_{V^{\{i\}}})$*

Proof. Notice that $V^{\{i\}}$ is multilinear-rank-preserving subspace for C. Hence, Lemma 2.6 implies that

$$\gcd(C|_{V^{\{i\}}}) = \gcd(C)|_{V^{\{i\}}} \text{ and } \operatorname{sim}(C|_{V^{\{i\}}}) = \operatorname{sim}(C)|_{V^{\{i\}}}.$$

Now, if x_i appears in $\gcd(C)$ then x_i appears in $\gcd(C)|_{V^{\{i\}}}$. Hence x_i appears in $\gcd(C|_{V^{\{i\}}})$. If, on the other hand, x_i does not appear in $\gcd(C)$ then it appears[17] in $\operatorname{sim}(C)$ and in a similar manner we get that x_i appears in $\operatorname{sim}(C|_{V^{\{i\}}})$. Since the circuit $C|_{V^{\{i\}}}$ is multilinear, x_i does not appear in $\gcd(C|_{V^{\{i\}}})$. □

Lemma 5.7. *The following holds for every linear function l appearing in the circuit C: It holds that $\ell \in \gcd(C)$ if and only if ℓ divides C and each x_i appearing in ℓ does not appear in $\operatorname{sim}(C|_V)$.*

Proof. Assume that $\ell \in \gcd(C)$. Clearly ℓ divides C. Since C is a multilinear circuit, any x_i appearing in ℓ cannot appear in $\operatorname{sim}(C)$. Hence, it cannot appear in $\operatorname{sim}(C|_V)$. This proves the first direction of the claim.

Let ℓ be a linear function dividing C such that $\ell \notin \gcd(C)$. To prove the second direction it is enough to show that some variable appearing in ℓ also appears in $\operatorname{sim}(C|_V)$. Indeed, as $\ell \notin \gcd(C)$, it follows that ℓ divides $\operatorname{sim}(C)$. Hence, ℓ is equal to some non-trivial linear combination of the linear functions in the default basis of $\operatorname{span}_1(\operatorname{Lin}(\operatorname{sim}(C)))$. Let $L_1, \ldots, L_r$ be this basis. Since V is r-multilinear-rank-preserving for C, we have that $L_1|_V, \ldots, L_r|_V$ are linearly independent^H linear functions and thus $\ell|_V$ is not a constant function. Let x_i be some variable appearing in $\ell|_V$. Obviously, x_i appears in ℓ (and hence in $\operatorname{sim}(C)$). By Lemma 2.6 we get that $\operatorname{sim}(C|_V) = \operatorname{sim}(C)|_V$, and therefore, x_i appears in $\operatorname{sim}(C|_V)$. This completes the proof of the lemma. □

We continue the proof of Lemma 5.5. The two previous lemmas show that in order to check whether x_i appears in $\gcd(C)$ it is enough to check that there exists a function ℓ, containing x_i, such that ℓ divides $(C|_{V^{\{i\}}})$ and each other variable x_j that appears in ℓ does not appear in $\operatorname{sim}(C|_V)$. This proves that if we found a "good" ℓ in Step 7 then $x_i \in \gcd(C)$.

Lemma 5.8. *Let $i \in [n]$ and let $l \in \gcd(C|_{V^{\{i\}}})$ be the linear function containing x_i. There exists a unique function $L \in \gcd(C)$ such that $L|_{V^{\{i\}}} = l$*

Proof. Since l divides $C|_{V^{\{i\}}}$, the function l must originate from some irreducible polynomial g dividing C (that is, l divides $(g|_{V^{\{i\}}})$). If g is not a linear function then we must have that g divides $\operatorname{sim}(C)$ meaning that x_i must appear in $\operatorname{sim}(C)$ which is a contradiction. Hence, there exists a linear function $L \in \gcd(C)$ such that $L|_{V^{\{i\}}} = l$. Assume that the function L is not unique. That is, there exists a function $L' \in \gcd(C)$ such that $L' \neq L$ and $L'|_{V^{\{i\}}} = l$. In such a case, x_i appears in two linear functions of $\gcd(C)$ and the circuit is not multilinear. □

We now finish the proof of Lemma 5.5. So far we have that there exists a single linear function $L \in C$ such that $L|_{V^{\{i\}}} = l$. Hence, for each $j \in [n-t]$ chosen in Step 7, the linear function l^j is unique and $L|_{V^{\{i,j\}}} = l^j$. This proves the algorithm correctness.

[17] We ignore the possibility that x_i does not appear in the circuit at all. In that case, x_i obviously does not appear in any restriction of the circuit.

We now analyze the complexity of the algorithm. We require, for each circuit $C|_{V^{\{i,j\}}}$, the linear functions dividing it. According to Lemma A.2 of Section A.1 there is a deterministic algorithm that finds the linear functions dividing a polynomial given black box access to it in $|\mathbb{F}|^{O(t)}$ time. Hence, to find the linear functions dividing each circuit $C|_{V^{\{i,j\}}}$ we require $n^2 \cdot |\mathbb{F}|^{O(t)}$ running time. Having these linear functions, it is clear that the algorithm requires an additional $n^2 \cdot \text{poly}(t)$ running time. Hence, the total time required for the algorithm is $n^2 \cdot |\mathbb{F}|^{O(t)}$. □

After constructing $\gcd(C)$ we obtain black boxes computing the different restrictions of $C/\gcd(C) = \text{sim}(C)$. We now present an algorithm that is given $\text{sim}(C)|_V$ and black boxes computing $\text{sim}(C)|_{V^{\{i\}}}$ for each $i \in [n-t]$ as input and outputs a circuit C' computing the same polynomial as $\text{sim}(C)$. We abuse notations for convenience and refer to the circuit $\text{sim}(C)$ as C.

Input: The circuit $C|_V$. Black box access to the polynomials computed by $C|_{V^{\{i\}}}$ for each $i \in [n-t]$.
Output: A circuit C' of n inputs such that $C' \equiv C$.
Let $L_1, \ldots, L_r$ be the default basis of $\text{span}_1(\text{Lin}(C|_V))$. Find a circuit $\bar{C}$ of r variables such that $\bar{C}(L_1, \ldots, L_r) = C|_V$. Note that the equality is between the circuits and not just the polynomials they compute;
For each $i \in [n-t]$, find a (super) set of elements of $\mathbb{F}$, $\{\beta_{i,j}\}_{j\in[r]}$ for which $C^i \triangleq \bar{C}(L_1 + \beta_{i,1}x_i, \ldots, L_r + \beta_{i,r}x_i) \equiv C|_{V^{\{i\}}}$ and C^i is a multilinear circuit. If such a set does not exist, output "fail";
output $C' \triangleq \bar{C}(L_1 + \sum_{i\in[n-t]} \beta_{i,1}x_i, \ldots, L_r + \sum_{i\in[n-t]} \beta_{i,r}x_i)$;

Algorithm 8: Lifting a low rank multilinear circuit to $\mathbb{F}^n$

Lemma 5.9. *Let $V \subseteq \mathbb{F}^n$ be an affine subspace of dimension t. Let C be an n-variate $\Sigma\Pi\Sigma(k)$ multilinear circuit over $\mathbb{F}$. Algorithm 8, given the corresponding inputs, runs in $n \cdot |\mathbb{F}|^{O(t)}$ time. If V is a liftable $\Delta(C)$-multilinear-rank preserving subspace then it outputs a multilinear $\Sigma\Pi\Sigma(k)$ circuit equivalent to C.*

Proof. We first analyze the time complexity. In Step 8, in order to find a circuit of the wanted form, we simply go over all possibilities for $\{\beta_{i,j}\}_{j\in[r]}$. There are $|\mathbb{F}|^r$ such possible sets. For each option we preform a PIT test to check if the found circuit computes the needed polynomial. We preform a deterministic PIT test in $2^{O(t)}$ time since C is multilinear (see Schwartz-Zippel lemma). We have that for each $i \in [n-t]$ we "spend" $|\mathbb{F}|^r \cdot 2^{O(t)} = |\mathbb{F}|^{O(t)}$ time. Hence, the total running time of Step 8 is at most $n \cdot |\mathbb{F}|^{O(t)}$. Clearly, all other steps require $\text{poly}(t, k)$ time and thus, since $t \geq k$, the total running time is $n \cdot |\mathbb{F}|^{O(t)}$.

We now prove the correctness of the algorithm. We first prove that in Step 8, there exists a circuit of the form C^i for each $i \in [n-t]$. Since V is $\text{rank}(C)$-multilinear-rank-preserving for C, then we have that $\text{rank}(C|_V) = \text{rank}(C) = r$. Let $\{\bar{L}_j\}_{j=1}^r$ be a basis of $\text{span}_1(\text{Lin}(\text{sim}(C)))$. Since $\text{rank}(C|_V) = \text{rank}(C)$, we have that the linear functions of $\{\bar{L}_j|_V\}_{j=1}^r$ are linearly independent^H. We define $\{\bar{L}_j\}_{j=1}^r$ as the basis of $\text{span}_1(\text{Lin}(\text{sim}(C)))$ such that the linear functions $\{\bar{L}_j|_V\}_{j=1}^r$ are the default basis of the subspace they span (that is, $\bar{L}_j|_V = L_j$ for each $j \in [r]$). Let $\tilde{C}$ be the r-variate circuit such that $\tilde{C}(\bar{L}_1, \ldots, \bar{L}_r) = C$. Then

$$\tilde{C}(\bar{L}_1|_V, \ldots, \bar{L}_r|_V) = C|_V = \bar{C}(L_1, \ldots, L_r).$$

It follows that $\tilde{C} = \bar{C}$. Therefore, for each $i \in [n-t]$, the circuit $C|_{V^{\{i\}}}$ is a circuit of the form

$$\bar{C}(L_1 + \alpha_{i,1}x_i, \ldots, L_r + \alpha_{i,r}x_i)$$

This means that in Step 8, we are guaranteed to find a circuit C^i for each $i \in [n-t]$.

In the following lemma we prove that $C' \equiv C$ thus proving the algorithm correctness.

Lemma 5.10. *Let $\{\beta_{i,j}\}_{i\in[n-t],j\in[r]}$ be a (super) set of field elements such that for each $i \in [n-t]$:*

$$\bar{C}(L_1 + \beta_{i,1}x_i, \ldots, L_r + \beta_{i,r}x_i) \equiv C|_{V^{\{i\}}}$$

Let $\{\alpha_{i,j}\}_{i\in[n-t],j\in[r]}$ be a (super) set of field elements such that

$$C = \bar{C}(L_1 + \sum_{i\in[n-t]} \alpha_{i,1}x_i, \ldots, L_r + \sum_{i\in[n-t]} \alpha_{i,r}x_i)$$

Let $A_1, A_2 \subseteq [n-t]$ such that $A_1 \cap A_2 = \emptyset$. Then

$$\bar{C}(L_1 + \sum_{i\in A_1} \beta_{i,1}x_i + \sum_{i\in A_2} \alpha_{i,1}x_i, \ldots, L_r + \sum_{i\in A_1} \beta_{i,r}x_i + \sum_{i\in A_2} \alpha_{i,r}x_i) \equiv \mathrm{sim}(C)|_{V^{A_1\cup A_2}}$$

Specifically,

$$\bar{C}(L_1 + \sum_{i\in[n-t]} \beta_{i,1}x_i, \ldots, L_r + \sum_{i\in[n-t]} \beta_{i,r}x_i) \equiv C$$

Proof. By induction on $|A_1|$: The base case where $A_1 = \emptyset$ is trivial. To prove the induction step we present a few definitions: For each $i \in [n-t]$ Let $p_i(y_1, \ldots, y_r, x)$ be the polynomial computed by $\bar{C}(y_1+\alpha_{i,1}x, \ldots, y_r+\alpha_{i,r}x)$. Let $q_i(y_1, \ldots, y_r, x)$ be the polynomial computed by $\bar{C}(y_1+\beta_{i,1}x, \ldots, y_r+\beta_{i,r}x)$.

Since for each $i \in [n-t]$, we have that $L_1, \ldots, L_r, x_i$ are linearly independent^H linear functions, it follows that for each $r+1$ indeterminates $y_1, \ldots, y_r, x$, it holds that

$$q_i(y_1, \ldots, y_r, x) = p_i(y_1, \ldots, y_r, x).$$

Let $m \in A_1$ be some integer. For each $j \in [r]$ define:

$$\hat{L}_j \triangleq L_j + \sum_{i\in A_2} \alpha_{i,j}x_i + \sum_{i\in A_1\setminus\{m\}} \beta_{i,j}x_i.$$

Then,

$$\bar{C}(L_1 + \sum_{i\in A_1} \beta_{i,1}x_i + \sum_{i\in A_2} \alpha_{i,1}x_i, \ldots, L_r + \sum_{i\in A_1} \beta_{i,r}x_i + \sum_{i\in A_2} \alpha_{i,r}x_i) =$$

$$\bar{C}(\hat{L}_1 + \beta_{m,1}x_m, \ldots, \hat{L}_r + \beta_{m,r}x_m) \equiv q_m(\hat{L}_1, \ldots, \hat{L}_r, x_m) =$$

$$p_m(\hat{L}_1, \ldots, \hat{L}_r, x_m) \equiv \bar{C}(\hat{L}_1 + \alpha_{m,1}x_m, \ldots, \hat{L}_r + \alpha_{m,r}x_m) \equiv \mathrm{sim}(C)|_{V^{A_1\cup A_2}}.$$

This proves the Lemma. □

By setting $A_1 = [n-t], A_2 = \emptyset$ in the former lemma, we get the proof of Lemma 5.9. □

We conclude this section with Algorithm 9 combining the section results. The algorithm works in the more general model where we are only given black boxes computing the restrictions of C.

Theorem 5.11. *Let $k, n \in \mathbb{N}$. Let C be an n-variate $\Sigma\Pi\Sigma(k)$ multilinear circuit. Let $B \subseteq [n]$, $\alpha \in \mathbb{F}$ and the corresponding restrictions of C be the input of Algorithm 9. Let $t = |B|$. Then Algorithm 9 runs in $n^2 \cdot |\mathbb{F}|^{O(k\cdot t)}$ time and outputs a set of size $|\mathbb{F}|^{O(k\cdot t)}$. If $V_{B,\alpha}$ is a liftable $\Delta(C)$-multilinear-rank-preserving subspace for C then there exists at least one circuit $C' \in \mathcal{C}$ such that $C' \equiv C$.*

Input: $k, n \in \mathbb{N}$, $B \subseteq [n]$, $\alpha \in \mathbb{F}$. For some $\Sigma\Pi\Sigma(k)$ multilinear n-variate circuit C we receive black box access to the polynomials computed by $C|_{V^A_{B,\alpha}}$ for each $A \subseteq [n]$ of size $|A| \leq 2$.

Output: A set $\mathcal{C}$ of n-variate $\Sigma\Pi\Sigma(k)$ multilinear circuits.

Define $V \triangleq V_{B,\alpha}$ and $t \triangleq \dim(V)$;

For each $\Sigma\Pi\Sigma(k)$ multilinear circuit $\hat{C}$ of t inputs such that $\hat{C} \equiv C|_V$, run Algorithms 7 and 8 where $\hat{C}$ is given as $C|_V$;

For each $\hat{C}$ where Algorithms 7 and 8 did not fail, add their output to $\mathcal{C}$;

Algorithm 9: Lifting a multilinear circuit to $\mathbb{F}^n$

Proof. Clearly, if $V_{B,\alpha}$ is a liftable $\Delta(C)$-multilinear-rank-preserving subspace for C, then when we run Algorithms 7 and 8 on $C|_V$ then we get a circuit C' computing the same polynomial as C. This proves the correctness of the algorithm.

We now analyze the time complexity and the size of $\mathcal{C}$. For each "guess" of $C|_V$, according to Lemmas 5.9 and 5.5, we require running time of $n^2 \cdot |\mathbb{F}|^{O(t)}$. For each such guess we increase the size of $\mathcal{C}$ by at most one. The question remaining is how many guesses for $C|_V$ do we have. Using combinatorial methods[18], it can be shown that the number of multilinear multiplication gates is at most $|\mathbb{F}|^{2t} \cdot \exp(t)$. Since we have at most k multiplication gates, the number of possible $\Sigma\Pi\Sigma(k)$ multilinear circuits is at most $|\mathbb{F}|^{O(k \cdot t)}$. Hence, the total running time of the algorithm is

$$|\mathbb{F}|^{O(k \cdot t)} \cdot |\mathbb{F}|^{O(t)} \cdot n^2 = n^2 \cdot |\mathbb{F}|^{O(k \cdot t)}.$$

The maximum size of $\mathcal{C}$ is $|\mathbb{F}|^{O(k \cdot t)}$. This proves the theorem. □

5.3 The Algorithm

We are now ready to present the main reconstruction algorithm for $\Sigma\Pi\Sigma(k)$ multilinear circuits. Algorithm 10 receives a black box computing an n-variate multilinear $\Sigma\Pi\Sigma(k)$ circuit over a field $\mathbb{F}$ and outputs a multilinear $\Sigma\Pi\Sigma(k)$ circuit computing the same polynomial. We give a lemma analyzing the algorithm that leads to Theorem 2.

Lemma 5.12. *Algorithm 10, given as input k, n and a black box computing an n-variate $\Sigma\Pi\Sigma(k)$ multilinear circuit C, outputs a circuit C' computing the same polynomial as C in $(n + |\mathbb{F}|)^{2^{O(k \log k)}}$ time.*

Proof. We first prove the algorithm correctness. Before the algorithm outputs a circuit, it verifies the output is legal (that is, we verify that $C' \equiv C$). Hence, it suffices to prove that in some iteration, the algorithm outputs a circuit.

According to Corollary 2.15, there exists a set $B \subseteq [n]$ and a field element $\alpha \in S$ such that $V_{B,\alpha}$ is a liftable r'-multilinear-rank-preserving subspace for C. We focus on the iterations where such B, α are chosen. Let $C_1, \ldots, C_s$ and r be a the circuits of the partition and the integer outputted by Algorithm 1 when its input is the circuit C, $r_{init} = R_M(2k)$ and $\kappa = k^3$. Let $k'_1, \ldots, k'_s$ the number of multiplication gates in $C_1, \ldots, C_s$ (respectively).

Recall that the subspace $V_{B,\alpha}$ is a liftable r'-multilinear-rank-preserving subspace for C. Also, $r_{init} \geq R_M(2k)$, $\kappa > k^2$ and $r' = r_{init} \cdot k^{(k-1) \cdot \lceil \log_k(\kappa) \rceil}$. Hence, by Corollary 5.4 we have that for each

[18] We bound the number of coefficients in a multiplication gate by 2t, this gives us $|\mathbb{F}|^{2t}$ options. To determine which indeterminates appear in the same linear functions, we require a partition of [t]. There are $\exp(t)$ possible partitions of $[t]$.

Input: $k, n \in \mathbb{N}$. A black box computing an n-variate $\Sigma\Pi\Sigma(k)$ multilinear circuit C.
Output: A multilinear $\Sigma\Pi\Sigma(k)$ circuit C' such that $C' \equiv C$.
Let $r_{init} \triangleq R_M(2k)$, $\kappa \triangleq k^3$, $r' \triangleq r_{init} \cdot k^{3(k-2)}$ and $S \subseteq \mathbb{F}$ such that $|S| = n^4k^2 + 1$;
foreach $\alpha \in S$ *and* $B \subseteq [n]$ *of size* $|B| = r' \cdot 2^k$ **do**
  For each $A \subseteq [n]$ define $V^A_{B,\alpha}$ as the affine subspace: $\text{span}_1 \{x_i | i \in B \cup A\} + (\alpha, \alpha^2, \ldots, \alpha^n)$;
  Define $\mathcal{A} = \{A \mid A \subseteq [n] \backslash B, |A| \leq 2\}$. For each $A \in \mathcal{A}$ find a multilinear $\Sigma\Pi\Sigma(k)$ circuit C^A such that $C^A \equiv C|_{V^A_{B,\alpha}}$;
  For each such circuit run Algorithm 1 with inputs n, k, r_{init}, κ, and C^A. Denote the polynomials computed by the partition circuits as $f^A_1, \ldots, f^A_{s_A}$;
  If there exist two such sets $A_1, A_2 \in \mathcal{A}$ such that $s_{A_1} \neq s_{A_2}$, proceed to the next pair of B, α. Otherwise, denote s as the number of the polynomials in each partition. Also, reorder the polynomials so that for each $A_1, A_2 \in \mathcal{A}$ and each $j \in [s]$ it will hold that $f^{A_1}_j|_{V_{B,\alpha}} = f^{A_2}_j|_{V_{B,\alpha}}$;
  foreach $k_1, \ldots, k_s \in \mathbb{N}^+$ *such that* $\sum_{i=1}^s k_i = k$ **do**
    For each $j \in [s]$, activate Algorithm 9 with inputs k_j, B, α and $\{f^A_j\}_{A \in \mathcal{A}}$. Denote the outputted set as $\mathcal{C}_{B,\alpha,j}$;
    For each $(\hat{C}_1, \ldots, \hat{C}_s) \in \mathcal{C}_{B,\alpha,1} \times \ldots \times \mathcal{C}_{B,\alpha,s}$ check whether $\sum_{j=1}^s \hat{C}_j \equiv C$. If so, output $\sum_{j=1}^s \hat{C}_j$. If no such circuits exist, proceed to the next iteration;
  end
end

Algorithm 10: Reconstruction of a multilinear $\Sigma\Pi\Sigma(k)$ circuit

$A \in \mathcal{A}$ it holds that $s_A = s$ and that for each set A there exist some reordering of the polynomial $\{f^A_j\}_j$ such that $f^A_j \equiv C_j|_{V^A_{B,\alpha}}$. In particular, this shows that Step 10 does not fail the algorithm. In Step 10 we reorder the polynomials so that w.l.o.g. it holds that $f^A_j \equiv C_j|_{V^A_{B,\alpha}}$ for each $j \in [s]$.

In the inner loop, we focus on the iteration where $k'_1, \ldots, k'_s$ are chosen as $k_1, \ldots, k_s$. By Theorem 5.11 we have that for each j, the set $\mathcal{C}_{B,\alpha,j}$ contains a circuit C'_j such that $C_j \equiv C'_j$. Hence, one of the circuits we check is the circuit $\sum_{j=1} C'_j$. Clearly, $\sum_{j=1} C'_j \equiv C$. This proves the correctness of the algorithm.

We now analyze the time complexity of Algorithm 10. In Step 10, we find several circuits via brute force techniques. As detailed in the proof of Theorem 5.11, the number of $\Sigma\Pi\Sigma(k)$ multilinear $(t+2)$-variate circuits over $\mathbb{F}$ is $|\mathbb{F}|^{O(k \cdot t)}$. For each such circuit, we perform a PIT check with $|\mathcal{A}|$ different multilinear $O(t)$-variate circuits. The time required for the PIT checks for each circuit is $2^{O(t)} \cdot |\mathcal{A}| = 2^{O(t)} \cdot n^2$. Hence, the total amount of time needed in Step 10 is $n^2 \cdot |\mathbb{F}|^{O(kt)}$.

Step 10 requires, according to Lemma 5.1, $O(\log(\kappa) \cdot t^3k^3 \cdot |\mathcal{A}|) = n^2 \cdot \text{poly}(t, k)$ time. Step 10 requires $O(n^4k^2)$ PIT checks of t-variate multilinear polynomials. Hence, it requires $n^4 \cdot 2^{O(t)}$ time (since $t \geq 2^k$).

The inner loop has $\exp(k)$ iterations. We analyze the running time of each iteration. Step 10 contains $s \leq k$ calls to Algorithm 9. This requires, according to Theorem 5.11, $n^2 \cdot |\mathbb{F}|^{O(k \cdot t)}$ running time. In Step 10 we preform, according to Theorem 5.11, $(|\mathbb{F}|^{O(kt)})^s = |\mathbb{F}|^{O(k^2t)}$ PIT tests. Each such test is between two n-variate multilinear $\Sigma\Pi\Sigma(k)$ circuits. To do so deterministically we use a deterministic algorithm given in [SV09], that runs in $n^{2^{O(k)}}$ time (see Lemma A.6 in Section A.4). Hence, Step 10 requires $|\mathbb{F}|^{O(k^2t)} \cdot n^{O(k)}$ time. In total, each iteration of the main loop requires $|\mathbb{F}|^{O(k^2t)} \cdot n^{O(k)}$ time. The number of such iterations is at most $|S| \cdot n^{|B|} = |S| \cdot n^t$. Hence, the total

running time of the algorithm is:

$$(n^4k^2+1)\cdot n^{k^{3(k-1)}R_M(2k)2^k}\cdot |\mathbb{F}|^{O(k^{3(k-1)+2}R_M(2k)2^k)}\cdot n^{O(k)} = (n+|\mathbb{F}|)^{2^{O(k\log k)}}.$$

This proves Lemma 5.12. $\square$

References

[AV08] M. Agrawal and V. Vinay. Arithmetic circuits: A chasm at depth four. In *Proceedings of the fourty ninth Annual FOCS*, pages 67–75, 2008.

[BB98] D. Bshouty and N. H. Bshouty. On interpolating arithmetic read-once formulas with exponentiation. *J. of Computer and System Sciences*, 56(1):112–124, 1998.

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

[BHH95] N. H. Bshouty, T. R. Hancock, and L. Hellerstein. Learning arithmetic read-once formulas. *SIAM J. on Computing*, 24(4):706–735, 1995.

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

[DS06] Z. Dvir and A. Shpilka. Locally decodable codes with 2 queries and polynomial identity testing for depth 3 circuits. *SIAM J. on Computing*, 36(5):1404–1434, 2006.

[FK06] L. Fortnow and A. R. Klivans. Efficient learning algorithms yield circuit lower bounds. In *Proceeding of the 19th annual COLT*, pages 350–363, 2006.

[GK98] D. Grigoriev and M. Karpinski. An exponential lower bound for depth 3 arithmetic circuits. In *Proceedings of the 30th Annual STOC*, pages 577–582, 1998.

[GR00] D. Grigoriev and A. A. Razborov. Exponential complexity lower bounds for depth 3 arithmetic circuits in algebras of functions over finite fields. *Applicable Algebra in Engineering, Communication and Computing*, 10(6):465–487, 2000.

[HH91] T. R. Hancock and L. Hellerstein. Learning read-once formulas over fields and extended bases. In *Proceedings of the 4th Annual COLT*, pages 326–336, 1991.

[Kal85] E. Kaltofen. Polynomial-time reductions from multivariate to bi- and univariate integral polynomial factorization. *SIAM J. on computing*, 14(2):469–489, 1985.

[Kal95] E. Kaltofen. Effective Noether irreducibility forms and applications. *J. of Computer and System Sciences*, 50(2):274–295, 1995.

[KM93] E. Kushilevitz and Y. Mansour. Learning decision trees using the fourier spectrum. *SIAM J. Comput.*, 22(6):1331–1348, 1993.

[KS01] A. Klivans and D. Spielman. Randomness efficient identity testing of multivariate polynomials. In *Proceedings of the 33rd Annual STOC*, pages 216–223, 2001.

[KS06a] A. Klivans and A. Shpilka. Learning restricted models of arithmetic circuits. *Theory of computing*, 2(10):185–206, 2006.

[KS06b] A. R. Klivans and A. A. Sherstov. Cryptographic hardness for learning intersections of half-spaces. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 553–562, 2006.

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

[KS08] Z. S. Karnin and A. Shpilka. Deterministic black box polynomial identity testing of depth-3 arithmetic circuits with bounded top fan-in. In *Proceedings of the 23rd Annual CCC*, pages 280–291, 2008.

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

[Raz08] R. Raz. Elusive functions and lower bounds for arithmetic circuits. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC)*, pages 711–720, 2008.

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

[Shp]

[Shp02] A. Shpilka. Affine projections of symmetric polynomials. *J. of Computer and System Sciences*, 65(4):639–659, 2002.

[SS08] N. Saxena and C. Seshadhri. An almost optimal rank bound for depth-3 identities. *CoRR*, abs/0811.3161, 2008.

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

[SV09] A. Shpilka and I. Volkovich. Improved polynomial identity testing for read-once formulas. Full version available at http://www.cs.technion.ac.il/ shpilka/publications/PROF.pdf. Submitted to RANDOM, 2009.

[SW01] A. Shpilka and A. Wigderson. Depth-3 arithmetic circuits over fields of characteristic zero. *Computational Complexity*, 10(1):1–27, 2001.

[VSBR83] L. G. Valiant, S. Skyum, S. Berkowitz, and C. Rackoff. Fast parallel computation of polynomials using few processors. *SIAM J. on Computing*, 12(4):641–644, November 1983.

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

A Toolbox

In this section we present several algorithms that were used throughout the paper. These algorithms are not directly related to the paper's topic and were thus chosen to appear in the appendix.

A.1 Finding Linear Factors

In several sections of the paper we encounter the following problem: Given black box oracle access to a circuit computing a t-variate polynomial f of degree d over a field $\mathbb{F}$, we would like to find all the linear factors of f. To solve this problem we use an algorithm for black box factoring of a multivariate polynomial devised in [Kal85, KT90, Kal95]. The algorithm outputs black boxes to the irreducible factors of the polynomial and their multiplicities. It requires that the field we are working with is of a large enough size ($\Omega(d^5)$). We assume that if the field is not large enough, we are allowed to make queries from an extension field. The following lemma gives the requirements and results of the algorithm:

Lemma A.1. *Let d, t be integers and f be a polynomial of t variables and of degree d over the field $\mathbb{F}$. Assume that $|\mathbb{F}| = \Omega(d^5)$. Then there is a randomized algorithm that gets as input a black box access to f, and the parameters t and d, and outputs, in $\mathrm{poly}(t, d, \log|\mathbb{F}|)$ time, with probability $1 - exp(-t)$ black boxes to all the linear factors, with their multiplicities*[19]*, of f. The algorithm uses $O(t \log|\mathbb{F}|)$ random bits.*

In order to find the linear factors of a given black box polynomial, we check which factors are linear functions and reconstruct them. Namely, we interpolate each factor as a linear function (To interpolate a linear function of t indeterminates we must query the polynomial in $t+1$ different points). We then use a polynomial identity test to check whether the factor is identical to the linear function. The PIT test we use is the "classic" randomized PIT test based on the Schwartz-Zippel lemma [Sch80, Zip79] requiring $O(t \log(d))$ random bits and one query. In our setting t is relatively small so we can go over all choices for random bits to get a deterministic algorithm.

Lemma A.2. *Let d, n be integers and f be a polynomial of t variables and of degree d. Let $\mathbb{F}$ be a finite field such that $|\mathbb{F}| = \Omega(d^5)$. Then there is a deterministic algorithm that gets as input a black box access to f, and the parameters t and d, and outputs, in $|\mathbb{F}|^{O(t)}$ time all the linear factors, with their multiplicities, of f.*

We note that while the original algorithm of Lemma A.1 does not necessarily output all the correct multiplicities and all the correct linear functions (in case that the characteristic of the field is a factor of the multiplicity), there is an easy way of taking care of that when the linear factors is all that we care about. Indeed, there is a simple algorithm that given black box access to g^{p^i} outputs both g and p^i, when g is a linear function, using queries from a polynomial size extension field (the algorithm is randomized, but when t is small we can use its brute force version).

A.2 Brute Force Interpolation

Throughout the paper we construct circuits computing polynomials for which we have black box access. In definition 2.19 we presented the notion of the default circuit for a polynomial f. Algorithm 11 shows how to construct this circuit. Namely, we solve the following problem: Let f be a t-variate polynomial over the field $\mathbb{F}$ of degree d. Given a black box computing f, we would like to construct the circuit C_f. Algorithm 11 solves the stated problem.

Since we use brute force techniques the running time of the algorithm is exponential in t. However, for our applications, the number of indeterminates is relatively small. That is, t is substantially smaller than d and $|\mathbb{F}|$. The methods we use take these relations into consideration (that is, the running time is exponential in t but not in $|\mathbb{F}|$ nor d). We start by finding the linear factors of f using the methods described in section A.1. We then construct $\mathrm{sim}(C_f)$ via brute force interpolation of $f/\mathrm{Lin}(f)$ according to all possible bases for $\mathbb{F}^t$.

Lemma A.3. *Let f be a t-variate polynomial over a field $\mathbb{F}$. Algorithm 11, given a black box holding f as input (as well as t and $\deg(f)$), outputs the circuit C_f in $|\mathbb{F}|^{O(t^2)}$ time.*

Proof. We first prove the correctness of the algorithm. The linear functions we find in Step 1 are the linear functions of $\mathrm{Lin}(f)$. By definition, $\gcd(C_f) = \mathrm{Lin}(f)$ so the output of $\gcd(C_f)$ is correct. By the choice of r it follows that $\mathrm{sim}(c_f)$ is a polynomial in exactly r liner functions, and clearly the

[19] In fact, the basic algorithm only guarantees that if p is the characteristic of the field and $g^{p^i \cdot e}$ is a factor of f (where e is not divisible by p), then the black-box corresponding to g that is outputted by the algorithm, holds g^{p^i} and its multiplicity is e.

Input: $t, d \in \mathbb{N}$ and a black box holding a t-variate polynomial f of degree bounded by d.
Output: The circuit $C_f = \gcd(C_f) \cdot \mathrm{sim}(C_f)$.
Find the linear factors of f. Output their product as $\gcd(C_f)$;
foreach *t linearly independent^H linear functions $\{L_i\}_{i\in[t]}$, in the variables $\bar{x} = (x_1, \ldots, x_t)$* **do**
  Represent $f/\mathrm{Lin}(f)$ as a polynomial in the L_i's That is, look for a polynomial $\tilde{h}$ such that $\tilde{h}(L_1, \ldots, L_r) = f$, for some $r \leq t$;
end
For some set $\{L_i\}$ for which the polynomial depends on a minimal number of linear functions[20] (any such basis will do), denote the polynomial as $\hat{h}$ and the r linear functions on which it depends on as $\hat{L}_1, \ldots, \hat{L}_r$;
Let $L_1, \ldots, L_r$ be the linear functions forming the default basis of $\mathrm{span}_1\{\hat{L}_1, \ldots, \hat{L}_r\}$. Let h be the polynomial such that $\hat{h}(\hat{L}_1, \ldots, \hat{L}_r) = h(L_1, \ldots, L_r)$;
Set $\mathrm{sim}(C)$ as $h(L_1, \ldots, L_r)$;

Algorithm 11: Brute force interpolation

polynomial $h(L_1, \ldots, L_r)$ equals to $\mathrm{sim}(C_f)$. Since $L_1, \ldots, L_r$ are the default basis, it follows that $h(L_1, \ldots, L_r) \equiv \mathrm{sim}(C_f)$.

We now analyze the running time of the algorithm: The first step finds all the linear factors of f. Lemma A.2 states that this can be done deterministically in $|\mathbb{F}|^{O(t)}$ time. In Step 11, at each iteration we interpolate a polynomial of degree bounded by d with t inputs. This requires[21] $d^{O(t)}$ time. The number of iterations is the number of bases for $\mathbb{F}^t$. It can easily be shown that this number is $|\mathbb{F}|^{O(t^2)}$. The time required by Step 11 is linear in the size of the description of the polynomial $\hat{h}$. Hence, it requires $d^{O(t)}$ time. To Conclude, by assuming that $|\mathbb{F}| \geq d$ we have that the total running time is $|\mathbb{F}|^{O(t^2)}$. □

A.3 Reconstructing Linear Functions

In [Shp] a method for efficiently reconstructing $\Sigma\Pi\Sigma(2)$ circuits was given. One of the algorithms presented there reconstructs a multiplication gate when given its restrictions to several co-dimension 1 subspaces on which it does not vanish. We use this algorithm in our paper for a similar purpose (we reconstruct the linear factors of a generalized multiplication gate). Parts 2 and 3 of algorithm 6 in [Shp] reconstruct a set of linear functions given their restrictions to several co-dimension 1 subspaces. The following lemma summarizes the results of the algorithm needed for our paper:

Lemma A.4. *(implicit in [Shp]) Let $\mathcal{L}$ be a (multi) set containing d linear functions in n indeterminates. Let $\{\varphi_1, \ldots, \varphi_m\}$ be a set of linearly independent^H linear functions such that $m \geq 100\log(d)$. For each $j \in [m]$ define the (multi) set*

$$\mathcal{L}_j \triangleq \left\{ \ell|_{\varphi_j=0} \;\middle|\; \ell \in \mathcal{L} \right\}.$$

Then there exists a deterministic algorithm that given $\{\mathcal{L}_j\}_{j=1}^m$ outputs $\mathcal{L}$ in $\mathrm{poly}(n, d)$ time.

A.4 Deterministic Polynomial Identity Testing Algorithms for Depth-3 Circuits

In [KS08] a deterministic black-box PIT algorithm for $\Sigma\Pi\Sigma(k, d, \rho)$ circuits was given. That is, [KS08] give a deterministic algorithm (Algorithm 1 in [KS08]) that verifies, in quasi-polynomial time, whether

[21] In order to use a simple interpolation of a polynomial of t inputs and degree d we may use Lagrange's formula which gives a description of size $d^{O(t)}$ of the polynomial.

a black-box $\Sigma\Pi\Sigma(k,d,\rho)$ circuit C computes the zero polynomial. Using the new rank bounds of [SS08] the following result is obtained.

Lemma A.5 (Lemma 4.10 of [KS08] combined with Theorem 2 of [SS08])**.** *Let C be an n-variate $\Sigma\Pi\Sigma(k,d,\rho)$ circuit. Then there exist a deterministic algorithm that, given black box access to C, verifies whether C computes the zero polynomial. The running time of the algorithm is*

$$\left(n\left(\binom{kd}{2}+2^k\right)\binom{R(k,d,\rho)+2}{2}+1\right)\cdot(d+1)^{R(k,d,\rho)}=$$

$$n\cdot\exp\left(2^{O(k^2)}\cdot(\log d)^{k-1}+k\rho\log d\right).$$

When C is a multilinear $\Sigma\Pi\Sigma(k)$ circuit a much better result was recently proved in [SV09].

Lemma A.6 (Theorem of [SV09])**.** *Let C be an n-variate $\Sigma\Pi\Sigma(k)$ multilinear circuit. Then there exist a deterministic algorithm that, given black box access to C, verifies whether C computes the zero polynomial. The running time of the algorithm is $n^{O(k)}$.*

B Proof of Lemma 4.9

In this section we give the proof of Lemma 4.9. The proof relies on the following lemma, which is a small generalization of claim 4.8 of [DS06], that regards the rank of $\Sigma\Pi\Sigma(k,d)$ circuits when restricted to many co-dimension 1 subspaces. Intuitively, the lemma shows that if $\mathcal{L}$ is a set of linear functions such that for every $\varphi\in\mathcal{L}$ we have that $\Delta(C)$ drops when restricted to the codimension 1 subspace defined by $\varphi=0$, then the set $\mathcal{L}$ is not too large.

Lemma B.1. *Let $C=\sum_{i=1}^{k}M_i$ be a simple and minimal $\Sigma\Pi\Sigma(k,d)$ circuit such that* [22] *$k\geq 3, d\geq 4$. Let $\chi\in\mathbb{N}^+$ and $\mathcal{L}$ be a set of m linearly independent^H linear functions such that for some $A\subseteq[k]$, the following holds for every $\varphi\in\mathcal{L}$:*

- *For every $i\in[k]$, it holds that $i\in A$ if and only if $\varphi\notin M_i$ (in particular, φ is a linear factor of $\gcd(C_{[k]\setminus A})$).*
- $\Delta(C_A|_{\varphi=0})<\frac{\Delta(C_A)}{2\chi\cdot\log(d)}$.

Assume that $\Delta(C_A)\geq 4k$. Then

$$|\mathcal{L}|=m\leq\max\{2k\log(dk)+2k,\frac{\Delta(C_A)}{\chi}\}.$$

The proof is almost identical to the proof of claim 4.8 in [DS06] and is postponed to Appendix B.1. We continue with generalizations of the claim that is needed for the proof of Lemma 4.9.

Lemma B.2. *Let C be an n-variate $\Sigma\Pi\Sigma(k,d)$ circuit over the field $\mathbb{F}$ such that $k\geq 3, d\geq 4$. Let $A\subseteq[k]$ and $\chi,r'\in\mathbb{N}^+$ be such that $\chi>1$ and $4k\leq r'\leq\Delta(C_A)$. Let $\mathcal{L}$ be a set of m linearly independent^H linear functions. Assume that the following holds for every $\varphi\in\mathcal{L}$:*

- *For every $i\in[k]$, it holds that $i\in A$ if and only if $\varphi\notin M_i$ (in particular, φ is a linear factor of $\gcd(C_{[k]\setminus A})$).*

[22] If $d<4$ then the reconstruction problem is trivial using simple interpolation so we assume that $d\geq 4$. If $k<3$ then the lemma is trivial as it concerns the Δ function of a non-empty subcircuit of C which must consist of a single multiplication gate.

- $\Delta(C_A|_{\varphi=0}) < \frac{r'}{2\chi\cdot\log(d)}$.

Then

$$|\mathcal{L}| = m \leq \max\{2k\log(dk) + 2k, \frac{r'}{\chi}\}.$$

Proof. We prove the claim by induction on $\Delta(C_A) - r'$. The basis of the induction is when $\Delta(C_A) - r' = 0$, or $r' = \Delta(C_A)$. In this case the claim follows directly from Lemma B.1. So assume that $\Delta(C_A) - r' > 0$. For a contradiction we assume that $m > \max\{2k\log(dk) + 2k, \frac{r'}{\chi}\}$. Note that as $\Delta(C_A) - r' > \Delta(C_A) - (r'+1) \geq 0$ the induction hypothesis holds for $\Delta(C_A) - (r'+1)$ and so $m \leq \max\{2k\log(dk) + 2k, \frac{r'+1}{\chi}\}$. In particular this implies that $m = \frac{r'}{\chi} + 1$. As $m < r' \leq \Delta(C_A)$ there is some linear function in $\mathrm{Lin}(\mathrm{sim}(C_A))$ that is not in $\mathrm{span}_1(\mathcal{L})$. Assume w.l.o.g. that this linear function is x_1. We now would like to apply the induction hypothesis on the circuit $C|_{x_1=\alpha}$ for some $\alpha \in \mathbb{F}$ (or for α from an extension field of $\mathbb{F}$). Assume that for some $\alpha \in \mathbb{F}$, the following properties hold.

- The linear function $x_1 - \alpha$ does not appear in the circuit C. That is, $x_1 - \alpha \notin \mathrm{Lin}(C)$.
- For every $\varphi \in \mathcal{L}$, the linear function $\varphi|_{x_1=\alpha}$ does not appear in any multiplication gate of $C_A|_{x_1=\alpha}$. I.e. $\varphi|_{x_1=\alpha} \notin \mathrm{Lin}(C_A|_{x_1=\alpha})$.
- The circuit $\mathrm{sim}(C_A)|_{x_1=\alpha}$ is simple.

We now continue the proof assuming that we have an α that satisfies the above properties, and later we shall prove that such an α indeed exists (it will be a corollary of Lemma B.3). In order to apply the induction hypothesis on the circuit $C|_{x_1=\alpha}$ we now show that the conditions of the lemma are satisfied. Let $C' \triangleq C|_{x_1=\alpha}$. As $x_1 - \alpha \notin \mathrm{Lin}(C)$, we have that C' is a $\Sigma\Pi\Sigma(k)$ circuit (and not a $\Sigma\Pi\Sigma(k-1)$ circuit, since no multiplication gate vanished). By the assumptions that $\mathrm{sim}(C_A)|_{x_1=\alpha}$ is simple and that $x_1 \in \mathrm{Lin}(\mathrm{sim}(C_A))$ it follows that $\Delta(C_A|_{x_1=\alpha}) = \Delta(C_A) - 1$. Let $\mathcal{L}' = \{\varphi|_{x_1=\alpha} \mid \varphi \in \mathcal{L}\}$. From the assumption that $x_1 \notin \mathrm{span}_1(\mathcal{L})$ we get that linear functions in $\mathcal{L}'$ are linearly independent^H. Consider some $\varphi \in \mathcal{L}$. By the choice of α we have that $\varphi|_{x_1=\alpha} \notin M_i|_{x_1=\alpha}$ for any $i \in A$. In addition, as $\varphi \in M_i$, for $i \notin A$, we have that $\varphi|_{x_1=\alpha} \in M_i|_{x_1=\alpha}$ for all $i \notin A$. Hence, the following holds for every $\varphi' \in \mathcal{L}'$:

- For each $i \in [k]$, it holds that $i \in A$ if and only if $\varphi' \notin M'_i$ (where M'_i is the i'th multiplication gate of $C|_{x_1=\alpha}$).
- Let $\varphi \in \mathcal{L}$ be the linear function such that $\varphi' = \varphi|_{x_1=\alpha}$. Then, $\Delta((C|_{x_1=\alpha})_A|_{\varphi'=0}) = \Delta(C_A|_{\varphi=0,x_1=\alpha}) \leq \Delta(C_A|_{\varphi=0}) < \frac{r'}{2\chi\cdot\log(d)}$.

Since $r' \leq \Delta(C_A) - 1 = \Delta(C_A|_{x_1=\alpha})$ we may apply the induction hypothesis on $C|_{x_1=\alpha}$, $\mathcal{L}'$, A, r' and χ to conclude[23] that

$$m = |\mathcal{L}| = |\mathcal{L}'| \leq \max\{2k\log(dk) + 2k, \frac{r'}{\chi}\}.$$

However, this contradicts the (contradiction) assumption that $m = \frac{r'}{\chi} + 1$. Therefore in order to conclude the proof of Lemma B.2 we only need to prove the existence of an appropriate $\alpha \in \mathbb{F}$ (or is some small extension field of $\mathbb{F}$). The following lemma is stronger than what we currently need, however, we state it in its stronger form as it will later be needed.

[23] We did not make sure that $\deg(C|_{x_1=\alpha}) \geq 4$. However, if this is not the case, we may add a new indeterminate y and work with the circuit $C|_{x_1=\alpha} \cdot y^4$ (that is, multiply every multiplication gate by y^4).

Lemma B.3. *Let $C \triangleq \sum_{i=1}^{k} C_{f_i}$ be an n-variate $\Sigma\Pi\Sigma(k,d,\rho)$ circuit over a field $\mathbb{F}$. Let $\hat{C} \triangleq \sum_{i=1}^{k} \gcd(C_{f_i})$. Let $V_0 \subseteq \mathbb{F}^n$ be a subspace of co-dimension $\tilde{r}$. Let $A \subseteq [k]$ and let $\mathcal{L}$ be a set of linearly independent^H linear functions with the following properties.*

- *For every $\varphi \in \mathcal{L}$, we have that $\varphi \notin \mathrm{span}_1\left(\cup_{i=1}^{k} \mathrm{Lin}\left(\mathrm{sim}(C_{f_i})\right)\right)$.*
- *For every $\varphi \in \mathcal{L}$ and $i \in [k]$, it holds that $i \in A$ if and only if φ is not a factor of C_{f_i} (i.e. φ is a factor of $\gcd(\sum_{i \notin A} C_{f_i})$).*

Then there exists an affine subspace $V = V_0 + v_0$ (V is a shift of V_0) with the following properties.

1. *No linear function $\ell \in \mathrm{Lin}(\hat{C})$ vanishes when restricted to V.*
2. *For every $\varphi \in \mathcal{L}$ and $\ell \in \cup_{i \in A} \gcd(C_{f_i})$ it holds that either $\varphi|_V \nsim \ell|_V$ or $\varphi|_V$ is a constant function.*
3. *For every pair of linear function $\ell_1 \nsim \ell_2 \in \mathrm{Lin}(\hat{C})$, we have that either $\ell_1|_V$ and $\ell_2|_V$ are constants or that $\ell_1|_V \nsim \ell_2|_V$.*

Proof. Let V' be a subspace of dimension $\tilde{r}$ such that $\mathbb{F}^n = V_0 \oplus V'$. Let $\{\psi_i\}_{i \in [r']}$ be a basis for V'. For every $\bar{u} = (u_1, \ldots, u_{\tilde{r}}) \in \mathbb{F}^{\tilde{r}}$ let $V_{\bar{u}} = V_0 + \sum_{i=1}^{\tilde{r}} u_i \cdot \psi_i$, be the affine shift of V_0 corresponding to $\bar{u}$. We now show that there exist an $\tilde{r}$-variate polynomial p, such that if $V_{\bar{u}}$ does not satisfy the conclusion of the lemma then $p(\bar{u}) = 0$. The following lemma of [KS08] shows the existence of an $\tilde{r}$-variate polynomial possessing similar properties.

Lemma B.4. *(implicit in the proof of Lemma 4.3 of [KS08]) Let $\tilde{\mathcal{L}}$ be a set of n-variate linear functions over the field $\mathbb{F}$. Let $V \subseteq \mathbb{F}^n$ be a subspace of co-dimension $\tilde{r}$. Let $\{V_{\bar{u}}\}_{\bar{u} \in \mathbb{F}^{\tilde{r}}}$ be the different shifts of V. Then there exists an $\tilde{r}$-variate non-zero polynomial h, of degree at most $\binom{|\tilde{\mathcal{L}}|}{2} + |\tilde{\mathcal{L}}|$, such that for every $\bar{u} \in \mathbb{F}^{\tilde{r}}$ it holds that if $h(\bar{u}) \neq 0$ then*

1. *For any two linear functions $\ell \nsim \ell' \in \tilde{\mathcal{L}}$ we have that either $\ell|_{V_{\bar{u}}} \nsim \ell|'_{V_{\bar{u}}}$ or both functions become constant when restricted to $V_{\bar{u}}$.*
2. *For any $\ell \in \tilde{\mathcal{L}}$ we have that $\ell|_{V_{\bar{u}}} \neq 0$.*

We proceed with the proof of Lemma B.3. Let $\tilde{h}$ be the $\tilde{r}$-variate polynomial guaranteed by Lemma B.4, for $\tilde{\mathcal{L}} \triangleq \mathrm{Lin}(\hat{C}) \cup \mathcal{L}$. Let $h = \tilde{h} \cdot \prod_{i=1}^{k} \mathrm{sim}(C_{f_i})$.

Let $\bar{u} \in \mathbb{F}^{\tilde{r}}$ be a vector such that $h(\bar{u}) \neq 0$ (note that if $|\mathbb{F}| > \binom{|\tilde{\mathcal{L}}|}{2} \geq \deg(h)$ then such $\bar{u}$ exists, otherwise we move to a slightly larger extension field). We now show that for this $\bar{u}$, the affine space $V_{\bar{u}}$ has the required properties. It is clear that $V_{\bar{u}}$ satisfies Properties 1 and 3 of the lemma. In order to show that the Property 2 is also satisfied we assume w.l.o.g.[24] that there is at least one linear function in $\mathcal{L}$ that is not constant on V_0. ¿From the assumption of the lemma we get that every $\varphi \in \mathcal{L}$ and every $\ell \in \cup_{i \in A} \gcd(C_i)$ satisfy $\ell \nsim \varphi$. Therefore, by the choice of $\bar{u}$ we have that Property 2 also holds. This completes the proof of Lemma B.3. □

To complete the proof of Lemma B.2, set $\tilde{r} = 1$ and let V_0 be the co-dimension 1 subspace on which $x_1 = 0$. Thus, there exist a field element α such that $V = V_0 + \alpha \cdot (1, 0, \ldots, 0)$ is the affine shift guaranteed by Lemma B.3. It is clear that restricting C to V amounts to substituting $x_1 = \alpha$. We now notice that the three properties of Lemma B.3, that are satisfied by V, imply the corresponding three properties that α should satisfy. This completes the proof of Lemma B.2. □

[24] Notice that if it is not the case then any $V_{\bar{u}}$ satisfies the second requirement of the lemma.

The next lemma is a generalization of Lemma B.2 for $\Sigma\Pi\Sigma(s,d,r)$ circuits.

Lemma B.5. *Let $s,d,r \in \mathbb{N}^+$ and let $C \triangleq \sum_{i=1}^{s} C_{f_i}$ be an n-variate $\Sigma\Pi\Sigma(s,d,r)$ circuit of degree d over a field $\mathbb{F}$. Let $A \subseteq [s]$ and $r', \chi \in \mathbb{N}^+$ such that $\chi > 1$ and and $4s \leq r' \leq \Delta(C_A) - s \cdot r$. Let $\mathcal{L}$ be a set of m linearly independent^H linear functions. Assume that the following holds for every $\varphi \in \mathcal{L}$:*

- *For every $i \in [s]$, we have that $i \notin A$ if and only if φ divides f_i.*
- $\varphi \notin \mathrm{span}_1\left(\cup_{i=1}^{s} \mathrm{Lin}\left(\mathrm{sim}(C_{f_i})\right)\right)$.
- $\Delta(C_A|_{\varphi=0}) < \frac{r'}{2\chi \cdot \log(d)}$.

Then

$$m \leq \max\{2s\log(ds) + 2s, \frac{r'}{\chi}\} + r \cdot s.$$

Proof. The proof is by a reduction to the proof of Lemma B.2. The idea is to first restrict the circuit to a subspace on which all the non linear terms of the multiplication gates are set to constants. That is, after the restriction we are left with a $\Sigma\Pi\Sigma(s,d)$ circuit. Then we use Lemma B.2 to get the required result. Let $\hat{C} = \sum_{i=1}^{s} \gcd(C_{f_i})$. Set

$$\tilde{r} \triangleq \dim_1\left(\mathrm{span}_1\left(\cup_{i=1}^{s} \mathrm{Lin}\left(\mathrm{sim}(C_{f_i})\right)\right)\right) \leq s \cdot r.$$

Assume w.l.o.g. that the liner functions in $\mathrm{sim}(C_{f_i})$ are homogeneous, i.e. have no constant term. Let V_0 be the co-dimension $\tilde{r}$ subspace in which the linear functions of $\mathrm{span}_1\left(\cup_{i=1}^{s} \mathrm{Lin}\left(\mathrm{sim}(C_{f_i})\right)\right)$ vanish. Similarly to the proof of Lemma B.3 we take V' be a subspace of dimension $\tilde{r}$ such that $\mathbb{F}^n = V_0 \oplus V'$. Let $\{\psi_i\}_{i\in[r']}$ be a basis for V'. For every $\bar{u} = (u_1, \ldots, u_{\tilde{r}}) \in \mathbb{F}^{\tilde{r}}$ let $V_{\bar{u}} = V_0 + \sum_{i=1}^{\tilde{r}} u_i \cdot \psi_i$, be the affine shift of V_0 corresponding to $\bar{u}$. We now show that there exists an affine subspace V, that is a shift of V_0, having the following properties:

- No linear function $\ell \in \mathrm{Lin}(\hat{C})$ vanishes when restricted to V.
- For every $\varphi \in \mathcal{L}$ and $\ell \in \hat{C}_A$ it holds that either $\varphi|_V \nsim \ell|_V$ or $\varphi|_V$ is a constant function.
- For every pair of linear function $\ell_1 \nsim \ell_2 \in \mathrm{Lin}(\hat{C})$, we have that either $\ell_1|_V$ and $\ell_2|_V$ are constants or $\ell_1|_V \nsim \ell_2|_V$.
- For every $i \in [s]$ it holds that $\mathrm{sim}(C_{f_i})|_V \neq 0$.

Indeed, in the proof of Lemma B.3 we get that there exists a non zero $\tilde{r}$-variate polynomial h such that if $h(\bar{u}) \neq 0$ then the affine space $V_{\bar{u}}$ satisfies the first three requirements above (we use the notations of the proof of Lemma B.3). Consider the polynomial $h'(\bar{u}) \triangleq \prod_{i=1}^{s} \mathrm{sim}(C_{f_i})(\bar{u})$. Clearly, if $h'(\bar{u}) \neq 0$ then for every $i \in [s]$ it holds that $\mathrm{sim}(C_{f_i})|_{V_{\bar{u}}} \neq 0$. Let $\bar{u}$ be such that $h(\bar{u}) \cdot h(\bar{u}) \neq 0$ (if $|\mathbb{F}| > \deg(h) + \deg(h')$ then such $\bar{u}$ exists, otherwise we pick it from an extension field of $\mathbb{F}$). It follows that $V \triangleq V_{\bar{u}}$ is the required affine shift of V_0. We also note that as $\mathrm{span}_1\left(\cup_{i=1}^{s} \mathrm{Lin}\left(\mathrm{sim}(C_{f_i})\right)\right)$ vanishes on V_0, then for any $\bar{y} \in V_{\bar{u}}$ it holds that $\mathrm{sim}(C_{f_i})(\bar{y}) = \mathrm{sim}(C_{f_i})(\bar{u})$, for every $i \in [s]$. In particular, for every $i \in [s]$ we have that $\mathrm{sim}(C_{f_i})|_V$ is a nonzero constant. Let

$$\mathcal{L}|_V \triangleq \{\varphi|_V \mid \varphi \in \mathcal{L}\}.$$

The linear functions of $\mathcal{L}|_V$ are not necessarily linearly independent^H. However,

$$\dim_1(\mathrm{span}_1(\mathcal{L}|_V)) \geq \dim_1(\mathrm{span}_1(\mathcal{L})) - (n - \dim(V)) = m - \tilde{r} \geq m - s \cdot r.$$

Let $\mathcal{L}' \subseteq \mathcal{L}|_V$ be some set of linearly independent^H linear functions such that $|\mathcal{L}'| \geq m - s \cdot r$. We would like to use the result of Lemma B.2 with regards to the set $\mathcal{L}'$, the circuit $C|_V$, the set A and the integers χ and r'. We now prove that the requirements of Lemma B.2 are satisfied. We first notice that $C|_V$ is indeed an n-variate $\Sigma\Pi\Sigma(s, d, 0) = \Sigma\Pi\Sigma(s, d)$ circuit (indeed each $\text{sim}(C_{f_i})$ is a non zero constant when restricted to V). We now prove that $4s \leq r' \leq \Delta((C|_V)_A)$. Note, that $\text{sim}(C_A)|_V$ is a simple circuit, as otherwise there should exist two linear functions $\ell_1 \nsim \ell_2 \in \text{Lin}(\hat{C})$ such that $\ell_1|_V$ is not a constant function and $\ell_1|_V \sim \ell_2|_V$, which is in contradiction to the choice of V. Therefore we get that $\text{sim}(C_A|_V) = \text{sim}(C_A)|_V$. It follows that

$$\Delta(C_A|_V) = \text{rank}(\text{sim}(C_A|_V)) = \text{rank}(\text{sim}(C_A)|_V) \geq$$

$$\text{rank}(\text{sim}(C_A)) - (n - \dim(V)) = \Delta(C_A) - \tilde{r} \geq \Delta(C_A) - r \cdot s.$$

Hence,

$$4s \leq r' \leq \Delta(C_A) - s \cdot r \leq \Delta((C|_V)_A).$$

We now prove that $\mathcal{L}'$ satisfies the requirements of the lemma . Let φ' be some linear function in $\mathcal{L}'$. Let $\varphi \in \mathcal{L}$ such that $\varphi|_V = \varphi'$. Let $i \in [s]$. If $i \notin A$ then φ is a factor of f_i and thus φ' is a factor of $f_i|_V$. If $i \in A$ then φ is not a factor of f_i. Hence, for every liner function $\ell \in \gcd(C_{f_i})$ we have that $\ell \nsim \varphi$. It follows from the properties of V that since φ' is not constant, then $\varphi' = \varphi|_V \nsim \ell|_V$. Since $\gcd(C_{f_i})|_V = \beta \cdot C_{f_i|_V}$ for some $0 \neq \beta \in \mathbb{F}$ (as the non-linear term of C_{f_i} is restricted to a non-zero constant in V), we have that φ' is not a factor of $f_i|_V$. To show that the second requirement of $\mathcal{L}'$ holds we notice that

$$\Delta((C|_V)_A|_{\varphi'=0}) \leq \Delta(C_A|_{\varphi=0}) < \frac{r'}{2\chi \cdot \log(d)}.$$

We thus have that the requirements of Lemma B.2 are satisfied and thus

$$m - r \cdot s \leq |\mathcal{L}'| \leq \max\{2s\log(ds) + 2s, \frac{r'}{\chi}\}.$$

This completes the proof of Lemma B.5. □

We now give the final lemma before the proof of Lemma 4.9. The difference from Lemma B.5 is that we only require that the rank of some subcircuit of C_A is reduced when restricted to the subspace on which $\varphi = 0$.

Lemma B.6. *Let $d, r, s \in \mathbb{N}^+$ and let $C \triangleq \sum_{i=1}^{s} C_{f_i}$ be an n-variate $\Sigma\Pi\Sigma(s, d, r)$ circuit over a field $\mathbb{F}$. Let $A \subseteq [s]$ such that $|A| \geq 2$. Let $r', \chi \in \mathbb{N}^+$ be such that $\chi > 1$ and for every subset $A' \subseteq A$ of size $|A'| = 2$, we have that $4s \leq r' \leq \Delta(C_{A'}) - s \cdot r$. Let $\mathcal{L}$ be a set of m linearly independent^H linear functions. Assume that the following holds for every $\varphi \in \mathcal{L}$:*

- *For every $i \in [s]$, it holds that φ is a factor of f_i if and only if $i \notin A$.*
- $\varphi \notin \text{span}_1(\cup_{i=1}^{s} \text{Lin}(\text{sim}(C_{f_i})))$.
- *There exists some subset $A' \subseteq A$ of size $|A'| = 2$ such that $\Delta(C_{A'}|_{\varphi=0}) < \frac{r'}{2\chi \cdot \log(d)}$.*

Then

$$m \leq \binom{|A|}{2} \left(\max\{2s\log(ds) + 2s, \frac{r'}{\chi}\} + r \cdot s \right).$$

Proof. For each $A' \subseteq A$ Define $\mathcal{L}_{A'}$ as the set of linear functions in $\mathcal{L}$ for which:

$$\Delta(C_{A'}|_{\varphi=0}) < \frac{r'}{2\chi \cdot \log(d)}.$$

According to Lemma B.5 when applied on the circuit $C_{([s]\setminus A)\cup A'}$, the set of linear functions $\mathcal{L}_{A'}$, the subset A', and the integers χ and r', we have that

$$|\mathcal{L}_{A'}| \leq \max\{2s\log(ds) + 2s, \frac{r'}{\chi}\} + r \cdot s$$

Thus,

$$|\mathcal{L}| \leq \sum_{A' \subseteq A,\, |A'|=2} |\mathcal{L}_{A'}| \leq \binom{|A|}{2}\left(\max\{2s\log(ds) + 2s, \frac{r'}{\chi}\} + r \cdot s\right).$$

□

We now give the proof of Lemma 4.9. For convenience we repeat it:

Lemma. *Let $k, s, d, r \in \mathbb{N}$ be such that $s \leq k$ and let $C \triangleq \sum_{i=1}^{s} C_{f_i}$ be a $\Sigma\Pi\Sigma(s, d, r)$ circuit. Let $\hat{\mathcal{L}}$ be a set of linearly independent^H linear functions and $A \subseteq [s]$ be a set of size $|A| \geq 2$. Let $\chi, r' \in \mathbb{N}$ be such that $\chi > 1$ and r' satisfies that for every $A' \subseteq A$ of size $|A'| = 2$ we have that $r' \leq \Delta(\sum_{i\in A'} C_{f_i}) - r \cdot k$. Assume that for each $\varphi \in \hat{\mathcal{L}}$, the following holds:*

- *For every $i \in [s]$, φ is a factor of f_i if and only if $i \notin A$.*
- *For every $i \in [s]$, $\varphi \notin \mathrm{Lin}(\mathrm{sim}(C_{f_i}))$.*
- *$\exists i \neq j \in A$, such that $\Delta(C_{f_i|_{\varphi=0}}, C_{f_j|_{\varphi=0}}) < \frac{r'}{2\cdot\chi\cdot\log(d)}$.*

Then

$$|\hat{\mathcal{L}}| \leq \binom{|A|}{2} \cdot \left(\max\{2k\log(dk) + 2k, \frac{r'}{\chi}\} + r \cdot k\right).$$

Proof. Note that if for every $i \in A$ and $\varphi \in \mathcal{L}$, it holds that[25] $C_{(f_i|_{\varphi=0})} = (C_{f_i})|_{\varphi=0}$, then the claim follows directly from Lemma B.6. So let $i \in A$ and $\varphi \in \mathcal{L}$. Let V be the co-dimension 1 subspace on which $\varphi = 0$. Since $\varphi \notin \mathrm{Lin}(\mathrm{sim}(C_{f_i}))$, we have that $\mathrm{rank}(\mathrm{sim}(C_{f_i})|_V) = \mathrm{rank}(\mathrm{sim}(C_{f_i}))$. In particular, V is $\Delta(C_{f_i})$-rank preserving for $\mathrm{sim}(C_{f_i})$. Furthermore, as $i \in A$ we have that φ is not a factor of f_i. Namely, $f_i|_V \neq 0$. Hence, by Lemma 4.14 we have that $C_{f_i}|_V = C_{f_i|_V}$. This completes the proof of Lemma 4.9. □

B.1 proof of Lemma B.1

In this section we prove Lemma B.1. The following notations are required for the proof. The proof is almost identical to the proof of Claim 4.8 of [DS06], however in order to make the reading easier we give the complete proof here instead of pointing out the required changes.

[25] The first circuit is the default circuit for $C_{(f_i|_{\varphi=0})}$ and the second circuit is the circuit resulting from the restriction of the default circuit of f_i to the subspace on which $\varphi = 0$.

B.1.1 Notations

For convenience, we refer to $\mathrm{sim}(C_A)$ as $\hat{\mathcal{C}}$ and define $\hat{k} \triangleq |A|$ and $\hat{r} \triangleq \mathrm{rank}(\hat{\mathcal{C}})$. As in [DS06] we assume w.l.o.g.[26] that the linear functions appearing in the circuit are homogenous, and that all the multiplication gates have the same degree, $\hat{d}$. Let

$$\hat{C} = \sum_{i=1}^{\hat{k}} N_i = \sum_{i=1}^{\hat{k}} \prod_{j=1}^{\hat{d}} L_{i,j},$$

where $\hat{N}_1, \ldots, \hat{N}_{\hat{k}}$ are the multiplication gates of $\hat{\mathcal{C}}$. It will be very convenient to think of $\hat{\mathcal{C}}, \hat{N}_1, \ldots, \hat{N}_{\hat{k}}$ as sets of indices. That is, we shall abuse notations and write

$$\hat{\mathcal{C}} = \{(i,j) \,|\, i \in [\hat{k}], j \in [\hat{d}]\},$$

$$\hat{N}_i = \{(i,j) \,|\, j \in [\hat{d}]\}.$$

For a set $H \subseteq \hat{\mathcal{C}}$, we denote with $\mathrm{rank}(H)$ the dimension of the vector space spanned by the linear functions whose indices appear in H. That is

$$\mathrm{rank}(H) \triangleq \dim\left(\mathrm{span}_1\{L_{ij} : (i,j) \in H\}\right).$$

For the rest of the proof we will treat subsets of $\hat{\mathcal{C}}$ interchangeably as sets of indices and as (multi)sets of linear functions. We also assume w.l.o.g. that $\mathcal{L} = \{x_1, \ldots, x_m\}$.

We would next like to define, for each $t \in [m]$, certain subsets of $\hat{\mathcal{C}}$ that capture the structure of $\hat{\mathcal{C}}|_{x_t=0}$. Fix some $t \in [m]$, and consider what happens to $\hat{\mathcal{C}}$ when we set x_t to zero. The resulting circuit $\hat{\mathcal{C}}|_{x_t=0}$ is generally not simple, and can therefore be partitioned into two disjoint sets: A set containing the indices of the linear functions appearing in $\gcd(\hat{\mathcal{C}}|_{x_t=0})$, and a set containing the indices of the remaining linear functions (these are the linear functions appearing in $\mathrm{sim}(\hat{\mathcal{C}}|_{x_t=0})$). To be more precise, denote by δ_t the degree of $\gcd(\hat{\mathcal{C}}|_{x_t=0})$. In every multiplication gate $\hat{N}_i$, there are δ_t linear functions that the restriction of their product to the linear space defined by the equation $x_t = 0$, is equal to $\gcd(\hat{\mathcal{C}}|_{x_t=0})$. In other words, the product of these δ_t linear functions is equal to $\gcd(\hat{\mathcal{C}})$ under the restriction $x_t = 0$. Denote the set of indices of these functions by G_t^i, and let $R_t^i \triangleq \hat{N}_i \backslash G_t^i$ be the set of indices of the remaining linear functions of this multiplication gate. We thus have (for some choice of constants $\{c_i\}_{i=1}^{\hat{k}}$)

$$\mathrm{sim}(\hat{\mathcal{C}}|_{x_t=0}) = \sum_{i=1}^{\hat{k}} c_i \prod_{(i,j)\in R_t^i} (L_{ij}|_{x_t=0}),$$

and,

$$\forall i \in [\hat{k}] \ , \ \gcd(\hat{\mathcal{C}}|_{x_t=0}) = \prod_{(i,j)\in G_t^i} (L_{ij}|_{x_t=0}).$$

We now define, for each $t \in [m]$, the sets $R_t \triangleq \bigcup_{i=1}^{\hat{k}} R_t^i$, and $G_t \triangleq \bigcup_{i=1}^{\hat{k}} G_t^i$. The following claim summarizes some facts that we will later need.

Claim B.7. *For every $t \in [m]$:*

1. $R_t \cap G_t = \emptyset$.

[26] This can be done by adding another indeterminate, y, to the circuit as is done in Lemma 3.5 of [DS06]. This will not have any affect on the claim and proof, beside saving some additional notations.

2. $\hat{\mathcal{C}} = R_t \cup G_t$.

3. $|G_t^i| = |G_t^{i'}|$ *for all* i, i'.

4. $|G_t| = \hat{k} \cdot \deg(\gcd(\hat{\mathcal{C}}|_{x_t=0})) = \hat{k} \cdot \delta_t$.

5. R_t *contains the indices of the linear functions appearing in* $\mathrm{sim}(\hat{\mathcal{C}}|_{x_t=0})$.

6. $r_t = \mathrm{rank}(\mathrm{sim}(\hat{\mathcal{C}}|_{x_t=0})) = \mathrm{rank}(R_t)$.

Proof. Items (1) and (2) follows directly from the definition of R_t and G_t as R_t^i and G_t^i give a partition of the indices in $\hat{N}_i$. Items (3) and (4) hold as the linear factors of $\gcd(\hat{\mathcal{C}}|_{x_t=0})$ belong to all the multiplication gates. By definition, R_t^i is the set of linear functions in $\hat{N}_i$ that belong to $\mathrm{sim}(\hat{\mathcal{C}}|_{x_t=0})$, which implies item (5). Finally, by definition $r_t = \mathrm{rank}(\mathrm{sim}(\hat{\mathcal{C}}|_{x_t=0}))$, and by (5) we have that R_t is the set of linear functions appearing in $\mathrm{sim}(\hat{\mathcal{C}}|_{x_t=0})$. □

B.1.2 The Proof

We finally give the proof of Lemma B.1. For the readers convenience we restate it here, rephrased to fit the new notations.

Lemma Let $C = \sum_{i=1}^{k} M_i$ be a simple and minimal $\Sigma\Pi\Sigma(k, d)$ circuit such that $k \geq 3, d \geq 4$. Let $\chi, m \in \mathbb{N}^+$. Let $A \subseteq [k]$ be such that:

- For each $i \in [k]$ and $t \in [m]$, it holds that $i \in A$ if and only if $x_t \notin M_i$.
- For each $t \in [m]$, it holds that $r_t = \mathrm{rank}(\mathrm{sim}((\hat{\mathcal{C}}|_{x_t=0}))) < \frac{\hat{r}}{2\chi \cdot \log(d)}$.

Assume that $\hat{r} \geq 4k$. Then

$$m \leq \max\{2k\log(dk) + 2k, \frac{\hat{r}}{\chi}\}.$$

As a first step we assume for a contradiction that

$$m > \max\{2k\log(dk) + 2k, \frac{\hat{r}}{\chi}\}.$$

Having defined, for each $t \in [m]$, the sets R_t and G_t, we would now like to show that there exist a 'small' ($\sim \log(d)$) number of sets R_t, such that their union covers almost all of $\hat{\mathcal{C}}$. As $rank(\hat{\mathcal{C}})$ is relatively high, and for each t, $r_t = rank(R_t)$ is (assumed to be) relatively small, we will get a contradiction. We construct the cover step by step, in each step we will find an index $t \in [m]$ such that the set R_t covers at least half of the linear functions not yet covered. This idea is made precise by the following claim.

Claim B.8. *For every integer* $1 \leq q \leq \log(\hat{d})$ *there exist* q *indices* $t_1, \ldots, t_q \in [m]$ *for which*

$$\left|\bigcup_{s=1}^{q} R_{t_s}\right| \geq \hat{k}\hat{d}(1 - 2^{-q}).$$

Proof. by induction on q:
Base case $q = 1$**:** In order to prove the claim for $q = 1$, it is sufficient to show that there exists $t \in [m]$ for which $|R_t| \geq \frac{1}{2}\hat{k}\hat{d}$. Suppose, on the contrary, that for all $t \in [m]$, $|R_t| < \frac{1}{2}\hat{k}\hat{d}$. Claim B.7 implies

that for all $t \in [m]$, $|G_t| \geq \frac{1}{2}\hat{k}\hat{d}$ (as $|G_t| + |R_t| = \hat{k} \cdot \hat{d}$). This in turn implies (by item (3) of Claim B.7) that for all $t \in [m]$

$$|G_t^1| \geq \frac{1}{2}\hat{d}. \tag{10}$$

Lemma B.9 is more general than what is required at this point, however, we will need it in its full generality when we handle $q > 1$.

Lemma B.9. *Let $\mathcal{C}$ be a simple $\Sigma\Pi\Sigma(k, d)$ circuit with n inputs. Let $t \in [n]$, $i_0 \in [k]$. Denote $\delta_t = \deg(gcd(\mathcal{C}|_{x_t=0}))$. Suppose that the linear functions in N_{i_0} are ordered such that*

$$\gcd(\mathcal{C}|_{x_t=0}) = (L_{i_0,1}|_{x_t=0})(x) \cdot (L_{i_0,2}|_{x_t=0})(x) \cdot \ldots \cdot (L_{i_0,\delta_t}|_{x_t=0})(x).$$

Then, there exist a matching, $M = \{\mathcal{P}_1, \ldots, \mathcal{P}_{\delta_t}\} \subseteq \mathcal{C} \times \mathcal{C}$, consisting of δ_t disjoint pairs of linear functions, such that for each $j \in [\delta_t]$:

- *The two linear functions in $\mathcal{P}_j$ span x_t.*
- *The first element of $\mathcal{P}_j$ is $L_{i_0,j}$.*

Proof. We can reorder the linear functions in each gate N_i, $i \neq i_0$, such that

$$\forall j \in [\delta_t] \ : \ L_{1,j}|_{x_t=0} \sim L_{2,j}|_{x_t=0} \sim \ldots \sim L_{i_0,j}|_{x_t=0} \sim \ldots \sim L_{k,j}|_{x_t=0}.$$

As $\mathcal{C}$ is simple, it cannot be the case that, for some j, $L_{i_0,j}$ appears in all the multiplication gates. Therefore, for every $j \in [\delta_t]$ there exists an index $\alpha(j) \in [k]$, such that $L_{i_0,j} \not\sim L_{\alpha(j),j}$. It follows that,

$$\forall j \in [\delta_t] \ : \ x_t \in \mathrm{span}_1\{L_{i_0,j}, L_{\alpha(j),j}\}.$$

For each $j \in [\delta_t]$ let $\mathcal{P}_j = (L_{i_0,j}, L_{\alpha(j),j})$. Set $M = \{\mathcal{P}_1, \ldots, \mathcal{P}_{\delta_t}\}$. It is clear that each $\mathcal{P}_j$ satisfies the two conditions of the lemma, and that the $\mathcal{P}_j$'s are disjoint. □

We continue with the proof of Claim B.8. From Equation 10 and Lemma B.9 we conclude that for each $t \in [m]$, there exists a matching $M_t \subset \mathcal{C} \times \mathcal{C}$, containing at least $\frac{1}{2}\hat{d}$ disjoint pairs of linear functions, such that every pair in M_t spans x_t. We now present a lemma, initially given in [DS06]. It in fact relates to a lower bound on the size of *locally decodable codes of 2 queries*. As the topic of locally decodable codes does not directly relate to this paper we do not elaborate on it any further.

Lemma B.10. *[Corollary 2.9 of [DS06]] Let $\mathbb{F}$ be any field, and let $a_1, \ldots, a_m \in \mathbb{F}^n$. For every $i \in [n]$ let $M_i \subset [m] \times [m]$ be a set of disjoint pairs of indices (j_1, j_2) such that $e_i \in \mathrm{span}_1\{a_{j_1}, a_{j_2}\}$. Then*

$$\sum_{i=1}^{n} |M_i| \leq m \log(m) + m.$$

□

We proceed with the proof of Claim B.8. From Lemma B.10 we get that

$$\frac{1}{2}\hat{d}m \leq \delta_t \cdot m \leq \sum_{t=1}^{m} |M_t| \leq \hat{k}\hat{d}\log(\hat{k}\hat{d}) + \hat{k}\hat{d}$$

which gives

$$m \leq 2\hat{k}\log(\hat{k}\hat{d}) + 2\hat{k} < 2k\log(kd) + 2k.$$

This contradict the assumption that $m > 2k\log(dk) + 2k$.

Induction step: Let us now assume that we have found $q-1$ indices $t_1, \ldots, t_{q-1} \in [m]$ for which

$$\left| \bigcup_{s=1}^{q-1} R_{t_s} \right| \geq \hat{k}\hat{d}(1 - 2^{-(q-1)}).$$

Let

$$R \triangleq \bigcup_{s=1}^{q-1} R_{t_s}, \tag{11}$$

$$S \triangleq \hat{\mathcal{C}} \backslash R. \tag{12}$$

Then, by our assumption

$$|S| \leq \hat{k}\hat{d}2^{-(q-1)}. \tag{13}$$

The proof goes along the same lines as the proof for $q = 1$: we show that there exists an index $t \in [m]$, such that R_t covers at least half of S. We will argue that if such an index does not exists, then a contradiction to our assumption can be derived. Our main tools in doing so are Lemma B.9 and Lemma B.10.

Claim B.11. *There exists $t \in [m]$, such that for all $i \in [\hat{k}]$,*

$$|G_t^i \cap S| < \hat{d}2^{-q}.$$

Roughly, the lemma states that there exists some variable, x_t, such that most of the linear functions in S do not belong to $gcd(\hat{\mathcal{C}}|_{x_t=0})$. In particular it implies that R_t covers a large fraction of S, as needed.

Proof. Assume, on the contrary, that for every $t \in [m]$, there exists $i_t \in [\hat{k}]$, for which

$$|G_t^{i_t} \cap S| \geq \hat{d}2^{-q}.$$

¿From Lemma B.9 we get that, for every $t \in [m]$, there exists a matching M_t, consisting of $\hat{d}2^{-q}$ disjoint pairs of linear functions, such that each pair spans x_t, and that the first element in each pair is in $G_t^{i_t} \cap S$ (from the lemma we actually get that M_t contains $deg(gcd(\hat{\mathcal{C}}|_{x_t=0}))$ many pairs, but we are only interested in the pairs whose first element is in $G_t^{i_t} \cap S$). We would now like to apply Lemma B.10 on the matchings $\{M_t\}_{t\in[m]}$, however, we also need that all the linear functions in all the matchings belong to S. We achieve this by projecting all the functions in R to zero. As the dimension of the linear functions in R is small (by our assumption that each r_{t_s} is small) we can find a linear transformation that sends the linear functions in R to zero but leaves many of the linear functions $\{x_t\}$ linearly independent. This is formalized in the next claim.

Claim B.12. *There exists a subset $B \subset [m]$, of size $|B| = m - \mathrm{rank}(R) \geq \frac{m}{2}$, and a linear transformation $\pi : \mathbb{F}^n \to \mathbb{F}^n$ such that*

- $\ker(\pi) = \mathrm{span}_1(R)$.
- $\forall t \in B$, $\pi(x_t) = x_t$.

Proof. Calculating, we get that

$$\begin{aligned}\operatorname{rank}(R) &= \operatorname{rank}(\bigcup_{s=1}^{q-1} R_{t_s}) \leq \sum_{s=1}^{q-1} \operatorname{rank}(R_{t_s}) = \sum_{s=1}^{q-1} r_{t_s} \\ &\leq (q-1)\frac{\hat{r}}{\log(d)2\chi} \leq \frac{\hat{r}}{2\chi} \leq \frac{m}{2}, \end{aligned} \tag{14}$$

where the second inequality follows from the assumptions in Lemma B.1, the third inequality follows from the fact that $q \leq \log \hat{d} \leq \log d$. The last inequality follows from our (contradiction) assumption. Let $m' = m - \operatorname{rank}(R)$. From Equation 14 we get that $m' \geq m/2$. In particular, there exists a subset $B \subset [m]$, of size $|B| = m'$, such that

$$\operatorname{span}_1(\{x_t | t \in B\}) \cap \operatorname{span}_1(R) = \{0\}.$$

Hence, there exists a linear transformation $\pi : \mathbb{F}^n \to \mathbb{F}^n$ such that

- $\ker(\pi) = \operatorname{span}_1(R)$.
- $\forall t \in B\ ,\ \pi(x_t) = x_t$.

This completes the proof of Claim B.12. □

Let B be the set obtained from the claim above, and π the corresponding linear transformation. We assume, w.l.o.g., that $B = [m']$. From here on, we only consider variables x_t such that $t \in [m']$ (i.e. $t \in B$). Fix such $t \in [m']$, and let $M'_t = \pi(M_t)$. In other words, $M'_t = \{(\pi(L), \pi(L'))\}_{(L,L') \in M_t}$. Clearly,

$$|M'_t| = |M_t| \geq \hat{d}2^{-q}. \tag{15}$$

Note that the pairs in M'_t still span x_t, as for any pair $(L, L') \in M_t$, with $x_t = \alpha L + \beta L'$, we have that

$$x_t = \pi(x_t) = \pi(\alpha L + \beta L') = \alpha\pi(L) + \beta\pi(L').$$

Since all the linear functions appearing in R were projected to zero, we know that all the pairs in each M'_t are contained in the multiset[27] $S' \triangleq \{\pi(L) : L \in S\}$. After this long preparation we apply Lemma B.10 to the matchings M'_t, and derive the following inequality:

$$\sum_{t=1}^{m'} |M'_t| \leq |S'| \log(|S'|) + |S'|. \tag{16}$$

As $|S'| = |S|$ (remember that S' is a multiset), we get by Equation 13 that

$$|S'| \leq \hat{k}\hat{d}2^{-(q-1)}. \tag{17}$$

By Equations 15, 16 and 17, it follows that

$$\begin{aligned} m/2 \cdot (\hat{d}2^{-q}) \leq m' \cdot (\hat{d}2^{-q}) &\leq \sum_{t=1}^{m'} |M'_t| \leq |S'| \log(|S'|) + |S'| \\ &\leq \hat{k}\hat{d}2^{-(q-1)} \log(\hat{k}\hat{d}2^{-(q-1)}) + \hat{k}\hat{d}2^{-(q-1)}. \end{aligned}$$

Hence,

$$m \leq \hat{k} \log(\hat{k}\hat{d}2^{-(q-1)}) + \hat{k} < \hat{k} \log(\hat{k}\hat{d}) + \hat{k}$$

which contradicts our (contradiction) assumption. This completes the proof of Claim B.11. □

[27] Note that we can replace each pair in M'_t, that contains the zero vector, with a singleton.

Let us now proceed with the proof of Claim B.8. Take t_q to be the index guaranteed by Claim B.11. We have that

$$\forall i \in [\hat{k}] \ : \ |G^i_{t_q} \cap S| < \hat{d}2^{-q}.$$

In particular

$$|G_{t_q} \cap S| < \hat{k}\hat{d}2^{-q}.$$

Notice that by equations 11 and 12, and by the fact that R_{t_q} and G_{t_q} give a partition of $\hat{\mathcal{C}}$, we get that the complement of $\bigcup_{s=1}^{q} R_{t_s}$ is exactly $G_{t_q} \cap S$. From this we get that adding R_{t_q} to R gives

$$\left| \bigcup_{s=1}^{q} R_{t_s} \right| \geq \hat{k}\hat{d}(1 - 2^{-q}).$$

This completes the proof of the Claim B.8. □

Having proved Claim B.8, we are now just steps away from completing the proof of Lemma B.1. Taking q to be $\lfloor \log(\hat{d}) \rfloor$ in Claim B.8, we get that there exist indices $t_1, \ldots, t_{\lfloor \log(\hat{d}) \rfloor} \in [m]$, such that

$$\left| \bigcup_{s=1}^{\lfloor \log(\hat{d}) \rfloor} R_{t_s} \right| \geq \hat{k}\hat{d} - 2\hat{k}.$$

Thus,

$$\hat{r} - 2\hat{k} \leq \operatorname{rank}\left(\bigcup_{s=1}^{\lfloor \log(\hat{d}) \rfloor} R_{t_s} \right) \leq \sum_{s=1}^{\lfloor \log(\hat{d}) \rfloor} r_{t_s}.$$

The last inequality tells us that there exists some $t \in [m]$ for which

$$r_t \geq \frac{\hat{r} - 2\hat{k}}{\lfloor \log(\hat{d}) \rfloor} \geq \frac{\hat{r} - 2\hat{k}}{\log(d)}. \tag{18}$$

Since $\hat{r} \geq 4k$ and $\chi \geq 1$, we have that

$$r_t \geq \frac{\hat{r} - 2k}{\log(d)} \geq \frac{\hat{r}}{2\log(d)} \geq \frac{\hat{r}}{2\chi \log(d)}.$$

This is in contradiction to the assumption in the statement of the lemma. This completes the proof of Lemma B.1. □