Electronic Colloquium on Computational Complexity, Report No. 104 (2019)

Reconstruction of Depth-4 Multilinear Circuits

Vishwas Bhargava [*] Shubhangi Saraf [†] Ilya Volkovich [‡]

Abstract

We present a deterministic algorithm for reconstructing multilinear $\Sigma\Pi\Sigma\Pi(k)$ circuits, i.e. multilinear depth-4 circuits with fan-in k at the top $+$ gate. For any fixed k, given black-box access to a polynomial $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ computable by a multilinear $\Sigma\Pi\Sigma\Pi(k)$ circuit of size s, the algorithm runs in time quasi-poly$(n, s, |\mathbb{F}|)$ and outputs a multilinear $\Sigma\Pi\Sigma\Pi(k)$ circuit of size quasi-poly(n, s) that computes f.

Our result solves an open problem posed in [GKL12] (STOC, 2012). Indeed, prior to our work, efficient reconstruction algorithms for multilinear $\Sigma\Pi\Sigma\Pi(k)$ circuits were known only for the case of $k = 2$ [GKL12, Vol17].

1 Introduction

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

Reconstruction algorithms could be deterministic or randomized. As we will discuss later, even with randomness, reconstruction is a highly nontrivial problem. A fundamental problem which is inherently related with reconstruction is the problem of black-box Polynomial Identity Testing (PIT for short). Just like in reconstruction we are given black-box access to a circuit, but here we just have to decide if the polynomial computed by this is identically zero or not. It is easy to see that, a *deterministic* reconstruction algorithm for a circuit class $\mathcal{C}$ will directly yield a black-box PIT for $\mathcal{C}$. Hence deterministic reconstruction is strictly harder than deterministic black-box PIT, and we know that deterministic black-box PIT for any class $\mathcal{C}$ implies strong lower bounds for that class [KI03, Agr05, HS80]. However, unlike reconstruction, we know there is an easy randomized algorithm for black-box PIT, even for general arithmetic circuits [DL78, Zip79, Sch80].

Note that, any efficient (polynomial-time) reconstruction algorithm will naturally also yield an approximation algorithm (up to polynomial factors) of the minimum (arithmetic) circuit size. Thus reconstruction is expected to be an inherently hard problem. Moreover, there are a number

[*]Department of Computer Science, Rutgers University, Piscataway, NJ 08854. Email: `vishwas1384@gmail.com`.

[†]Department of Mathematics & Department of Computer Science, Rutgers University, Piscataway, NJ 08854. Research supported in part by NSF grants CCF-1350572 and CCF-1540634. Email: `shubhangi.saraf@gmail.com`.

[‡]Department of EECS, CSE Division, University of Michigan, Ann Arbor, MI 48109. Email: `ilyavol@umich.edu`.
.

ISSN 1433-8092

of results showing hardness of reconstruction under various complexity-theoretic and cryptographic assumptions [FK09, Hås90, KS09c]. (For more details see Section 1.2).

Despite reconstruction being a very hard problem, there has been a lot of research focused on efficient reconstruction for restricted (yet interesting) models. Many of these models are models for which we already knew good structural results and/or efficient deterministic PIT algorithms. For instance, for bounded-depth circuits we know efficient reconstruction algorithms in following cases: depth-2 ($\Sigma\Pi$) arithmetic circuits (a.k.a sparse polynomials) [BOT88, KS01], depth-3 ($\Sigma\Pi\Sigma$) circuits with bounded top fan-in [Sin16, KS09a], and set-multilinear depth-3 circuits [BBB$^+$00, KS06]. Other models for which we have efficient reconstruction algorithms are read-once arithmetic formulas [HH11, BB98, SV14], read-once oblivious branching programs (ROABP) [RS03, FS12] and multilinear depth-4 ($\Sigma\Pi\Sigma\Pi$) circuits with top fan-in 2 [GKL12].

The focus of this work is on the class of multilinear depth-4 arithmetic $\Sigma\Pi\Sigma\Pi$ circuits. A multilinear polynomial is a polynomial with individual degree of each variable bounded by 1. We say that a circuit $\mathcal{C}$ is multilinear (or syntactically multilinear) if every gate in $\mathcal{C}$ computes a multilinear polynomial. Many of the polynomials of importance/interest are multilinear (for instance Determinant, Permanent, Iterated Matrix Multiplication), and hence multilinearity is a natural restriction to study. Depth-4 arithmetic circuits are already computationally very powerful as was shown by the surprising depth reduction result by Agrawal and Vinay [AV08], and later improved by Koiran [Koi10] and Tavenas [Tav13]) who showed that any general (multilinear) arithmetic circuit of polynomial size and polynomial degree has an equivalent depth-4 (multilinear) circuit of sub-exponential size. Thus, a polynomial-time reconstruction algorithm for even depth-4 (multilinear) circuits will yield a non-trivial (sub-exponential) reconstruction algorithm for general (multilinear) arithmetic circuits.

We will study the model of multilinear depth-4 arithmetic circuits of top fan-in $= k$, which we call multilinear $\Sigma\Pi\Sigma\Pi(k)$ circuits. Formally, a $\Sigma\Pi\Sigma\Pi(k)$ circuit C has four layers of alternating Σ and Π gates and it computes a polynomial of the form

$$C(\bar{x}) = \sum_{i=1}^{k} F_i(\bar{x}) = \sum_{i=1}^{k} \prod_{j=1}^{d_i} P_{ij}(\bar{x})$$

where the $P_{ij}(\bar{x})$-s are polynomials computed by the last two layers of $\Sigma\Pi$ gates of the circuit and are the inputs to the Π gates at the second level. A *multilinear* $\Sigma\Pi\Sigma\Pi(k)$ circuit is a $\Sigma\Pi\Sigma\Pi(k)$ circuit in which each multiplication gate F_i computes a multilinear polynomial.

As we mentioned before, most of the models for which we know efficient reconstruction algorithms are classes for which we know how to do efficient polynomial identity testing, and often times this is because we have some illuminating insight into the structure of circuits from that class. One interesting example of such a structural result is the notion of "rank bounds" for depth-3 circuits of bounded top fan-in ($\Sigma\Pi\Sigma(k)$ circuits) which led to very efficient deterministic PIT results for this class, and then eventually led to efficient reconstruction algorithms. The rank bound results of [DS07, KS07, KS09b, SS09, SS10] essentially showed that under some mild conditions, any $\Sigma\Pi\Sigma(k)$ circuit which computes the identically zero polynomial must essentially be a polynomial that is "low dimensional".

Analogous to the rank bounds results for $\Sigma\Pi\Sigma(k)$ circuits, we also know interesting structural results for multilinear $\Sigma\Pi\Sigma\Pi(k)$ circuit computing the zero polynomial. It was shown in [SV18] that under similar mild conditions, any multilinear $\Sigma\Pi\Sigma\Pi(k)$ circuit computing the zero polynomial must be such that every gate in it computes a sparse polynomial. This sparsity bound was then used to design efficient deterministic black-box PIT algorithms for $\Sigma\Pi\Sigma\Pi(k)$ circuits. It was a very natural question to extend this result to design efficient reconstruction algorithms for the same class. However, despite all the progress on PIT and reconstruction algorithms in the last few years [GKL12, FS12, SV18, ST17, MV18, KS19], such an algorithm remained elusive for any value of k that was greater than 2.

1.1 Our Result

We give an efficient reconstruction algorithm for multilinear $\Sigma\Pi\Sigma\Pi(k)$ circuits for any top fan-in, k, as long as it is bounded (for instance a constant). Our algorithm is deterministic and runs in time quasipoly$(n, s, d, |\mathbb{F}|)$ and outputs a multilinear $\Sigma\Pi\Sigma\Pi(k)$ circuit of size at most quasipoly(n, s, d) as long as k is any constant. Our result solves an open problem posed in [GKL12]. Indeed, prior to our work, reconstruction algorithms for multilinear depth-4 circuits were only known when the top fan-in, k, is at most 2 [GKL12, Vol17]. We now state our main theorem formally below.

Theorem 1. *Given n, s, k and black-box access to a multilinear $\Sigma\Pi\Sigma\Pi(k)$ circuit C of size at most s over a finite field $\mathbb{F}$ in n variables as input, there exist a deterministic algorithm which runs in time $|\mathbb{F}|^{(\log s)^{f(k)}}$ and outputs a multilinear $\Sigma\Pi\Sigma\Pi(k)$ circuit C' of size at most $2^{(\log s)^{f(k)}}$ such that C' computes the same polynomial as C, where $f(k) \triangleq 2^{2^{k+1}}$.*

Remark 1.1. *We will need the field size to be large enough (something like $20kn^3$) for our algorithm to work. If this is not the case, just as is usually done for PIT, we assume that for reconstruction too, we are able to query the circuit on inputs that might come from an extension field. As long as we can query on a field extension, we can do away with this assumption of largeness of field size, size we can just take an extension of our field which is of size at least $20kn^3$ and work over that field. Our output circuit C' however will be guaranteed to be over the same field as C.*

Remark 1.2. *Later, in Theorem 5.1 we restate this theorem with more refined parameters.*

It would be very interesting to get an algorithm whose runtime is only polynomial (or quasi-polynomial) in $\log(|\mathbb{F}|)$. However, as was shown in [Vol16]: any proper learning algorithm for multilinear $\Sigma\Pi\Sigma\Pi(k)$ circuits over $\mathbb{F}$ (even for $k = 2$) will directly lead to an algorithm to compute square roots in $\mathbb{F}$. For a finite field $\mathbb{F}$, computing square roots deterministically in time poly$(\log(|\mathbb{F}|))$ is a long standing open problem with all best known algorithms having polynomial dependence on the field size [AM94, Bur57]. Therefore, any learning algorithm (that does not improve on the time complexity of computing square roots) should either be randomized or run in time at least poly$(|\mathbb{F}|)$. We could still hope to get more efficient run time with randomization.

Another thing we would like to note is that getting an efficient reconstruction algorithm for general multilinear $\Sigma\Pi\Sigma\Pi$ circuits (without the bound on top fan-in) is an enormously interesting and likely a very difficult problem. Via the depth reduction results of [AV08, Koi10, Tav13, KdOS19] which preserve the syntactic multilinear structure of the circuit, any subexponential-time reconstruction algorithm for multilinear $\Sigma\Pi\Sigma\Pi$ circuits would give a subexponential-time reconstruction algorithm for general multilinear circuits.

1.2 Related Work

Reconstruction of multilinear $\Sigma\Pi\Sigma\Pi(k)$ circuits has already received a fair amount of attention. However till our work, we only knew efficient algorithms when the top fan-in, k, was at most 2.

The case of $k = 1$ is equivalent to finding an efficient factorization algorithm for sparse multilinear polynomials and was resolved in the work of Shpilka and Volkovich [SV10]. The case of $k = 2$ is already highly nontrivial and very interesting and thus needed quite a few new ideas. Gupta et. al [GKL12] gave a randomized $\mathrm{poly}(n, s)$ algorithm for multilinear $\Sigma\Pi\Sigma\Pi(2)$ circuits (this works over all fields). This work was later derandomized in [Vol17] over fields of characteristic zero and polynomial sized finite fields. It was open for a while to extend these results to larger values of k, specially since we already know efficient derandomized black-box polynomial identity testing results for multilinear $\Sigma\Pi\Sigma\Pi(k)$ circuits for any constant k [KMSV13, SV18]. Indeed, this question was posed as an open problem in [GKL12].

Another closely related work to ours is that of Karnin and Shpilka [KS09a] which gave efficient reconstruction algorithms for depth-3 ($\Sigma\Pi\Sigma$) circuits with bounded top fan-in. Their algorithm is randomized and runs in time $\mathrm{quasipoly}(n, d, |\mathbb{F}|, s)$. When the input is a *multilinear* $\Sigma\Pi\Sigma(k)$ circuit (a strict subclass of the class of circuits we consider in this paper), they gave a deterministic $\mathrm{poly}(n, |\mathbb{F}|, s)$ time proper learning algorithm for this class. However, for general $\Sigma\Pi\Sigma(k)$ circuits, their algorithm is randomized and their algorithm is not proper - it outputs an equivalent circuit from the class of what they called "generalized" depth-3 circuit. Their algorithm, like ours also has a quasi-polynomial dependence on $|\mathbb{F}|$, and hence just works for finite fields. Our analysis at a very high level has many similar notions to their analysis and we discuss these similarities a bit in the next section. Over field of characteristic 0, the only efficient reconstruction algorithm we know for $\Sigma\Pi\Sigma$ circuits is when the top fan-in is 2 [Sin16] and this algorithm is randomized.

Indeed, in almost all cases for which we know efficient reconstruction algorithms, these were preceded by deterministic PIT algorithms for the same class, and this is hardly surprising. It is easy to see that any deterministic reconstruction algorithm for a class of circuits $\mathcal{C}$ is at least as hard is derandomizing black-box PIT for $\mathcal{C}$. Even randomized reconstruction almost always requires some deep understanding of the structure of the underlying circuit class and in almost every case we know, it seems harder than derandomizing PIT for that class. We list below some instances for which we have efficient reconstruction algorithms, and in almost all cases we also have efficient derandomized black-box PIT algorithms.

The class of circuits for which we understand reconstruction really well is the class of depth-2 ($\Sigma\Pi$) arithmetic circuits (a.k.a sparse polynomials). We can properly learning sparse polynomials (them) in *deterministic* $\mathrm{poly}(s, n, d)$ time over any field [BOT88, KS01]. Another class for which we understand reconstruction *reasonably* well is the class of read-once oblivious branching programs (ROABP). Raz and Shpilka [RS04] gave a randomized reconstruction (proper learning) algorithm for which ran in time $\mathrm{poly}(n, d, s)$. This was later derandomized in [FS12] with time complexity $\mathrm{quasipoly}(n, d, s)$. For depth-3 circuits, reconstruction algorithms for various restricted classes have been studied. For instance, for set-multilinear depth-3 circuits [BBB$^+$00, KS06] gave a randomized poly(n,d,s) (improper) learning algorithm which outputs an ROABP.

Recently, there has been a flurry of activity in average case learning algorithms for various arithmetic circuit classes [KS19, KNST17, KNS18, GKQ14, GKL11]. These results can be thought of as worst case reconstruction, given some non-degeneracy condition holds for some implicit polynomials (which are usually computed by intermediate gates). Interestingly, these results fall under the umbrella of learning from natural lower bounds which is exciting area of research in arithmetic as well as Boolean circuit complexity [CIKK16, KS19].

Hardness Results

We also know various hardness results for polynomial reconstruction and we list below some of those results that are interesting and relevant to this paper. Indeed, we see that polynomial reconstruction even for depth-3 and depth-4 circuits is an extremely interesting and challenging problem. Firstly, as we mentioned before, *deterministic* reconstruction for a circuit class $\mathcal{C}$ will directly yield a black-box PIT for $\mathcal{C}$. This implies that black-box PIT is a prerequisite for deterministic reconstruction. (See also [Vol16] for more details.) Moreover, efficient algorithms for black-box PIT for a class $\mathcal{C}$ are in turn known [HS80, Agr05] to imply super-polynomial lower bounds for circuits in $\mathcal{C}$ computing an explicit polynomial. Thus deterministic reconstruction algorithms for any circuit class immediately gives strong lower bounds for that class.

Another "hardness result", as mentioned before, is that via the depth reduction results of [AV08, Tav13, KdOS19] which preserve the syntactic multilinear structure of the circuit, any sub-exponential-time reconstruction algorithm for (multilinear) $\Sigma\Pi\Sigma\Pi$ circuits would give a sub-exponential time reconstruction algorithm for general (multilinear) circuits. If these algorithms are deterministic, this would imply new circuit lower bounds for general arithmetic circuits.

Even if allow randomization, [FK09] showed that a randomized polynomial-time reconstruction algorithm for an arithmetic circuit class $\mathcal{C}$ implies the existence of a function in BPEXP that does not have polynomial size circuits from $\mathcal{C}$.

We also know some NP-hardness results for reconstruction, which essentially stems from hardness of tensor decomposition. Håstad [Hås90] showed that reconstructing *optimal* size depth-3 set-multilinear circuits is already NP-hard. Finding the smallest depth-3 powering circuit computing a given polynomial is also NP-hard [Shi16]; it amounts to computing the symmetric-tensor rank. It might appear that the additional constraint of finding the optimal/smallest circuit might be a reason for this hardness. However, most of the reconstruction algorithms we know (including this result) satisfy some optimality condition [KS09a, GKL12]. In [KS09d], Klivans and Sherstov showed hardness of reconstruction under cryptographic assumptions. Concretely, they proved that assuming the hardness of the shortest vector problem (SVP) for quantum algorithms, no PAC learning algorithm exists for $\Sigma\Pi\Sigma$ circuits. We would like to remark that assumption of PAC learning is stronger than the learning considered in this article, since input points for PAC learning comes from an unknown distribution.

1.3 Organization

The paper is organized as follows: In Section 3, we recall some algebraic tools and algebraic algorithms that will be useful for us. In Section 4, we formally introduce our model, give some related definition and some structural results. We present and analyze the deterministic learning algorithm in Section 5. We conclude with some open questions in Section 6.

2 Overview of Proof

Our techniques are inspired by the technique of Karnin and Shpilka for the reconstruction of $\Sigma\Pi\Sigma(k)$ circuits [KS09a]. At a high level, their algorithm first computes a projection of the circuit down to a few variables, uses the notion of "rank bounds" for $\Sigma\Pi\Sigma(k)$ circuits to show that the projection will have a unique reconstruction, and it then attempts to "lift" the projection to recover the original circuit. The structure of our algorithm is similar, though there are several new obstacles to deal with and the implementation of the various steps is quite different. We also take a certain projection of the circuit down to a few variables. Instead of using rank bounds, we use "sparsity bounds" proved by Saraf and Volkovich [SV18] in order to prove uniqueness of reconstruction of the projected circuit, and then we lift the projection to recover the original circuit. Our lifting step is inspired by some elements of the list-decoding algorithm of [STV01] and the factorization algorithm of [KT90].

To demonstrate an overview of the proof, we will focus on the case when the top fan-in, i.e. k, equals 2. The case of $k = 1$ has been studied and resolved in the past and it is equivalent to finding an efficient factorization algorithm for sparse multilinear polynomials [SV10]. Recall that for the case of $k = 2$, the work of Gupta et al. [GKL12] already gives an efficient randomized polynomial-time algorithm which works over all fields. The work of Volkovich [Vol17] shows how to derandomize this algorithm over small fields and fields of characteristic 0 [1].

Our algorithm is the first efficient algorithm for larger values of k. It is deterministic and runs in time that is quasi-polynomial in the field size and input circuit size, when k is any constant.

In the overview of proof, we will focus on the case $k = 2$. Restricted to this case, our algorithm is quite different from the other algorithms mentioned before [GKL12, Vol17] and most importantly it generalizes easily to larger values of k. Most of the ideas for handling general k already appear in the case $k = 2$ and we now describe this case.

2.1 The case of top fan-in $= 2$

We are given black-box access to a polynomial $P \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ computed by a multilinear $\Sigma\Pi\Sigma\Pi(2)$ circuit $C(\bar{x}) = F_1(\bar{x}) + F_2(\bar{x})$ of size s, and the goal is to recover a multilinear $\Sigma\Pi\Sigma\Pi(2)$ circuit of a "comparable" size that computes P.

A distance measure Inspired by a similar notion of Karnin and Shpilka [KS09a], we define a distance measure between the two multiplication gates F_1 and F_2. Roughly speaking, the distance between them is the number of factors that appears in only one of the gates. We say that F_1 and F_2 are *m-far* if at least one of F_1, F_2 has at least m factors that do not appear in the other gate. (We use this notion here for ease of exposition - later in the paper we just a slightly different notion, which is very similar in spirit, but is more convenient for calculations).

The plan of proof will be the following. We will first reduce to the case where the two gates are "far" from each other and then show that in this case the circuit can be learnt exactly, i.e. our algorithm will learn F_1 and F_2. In order to show this, the main bulk of the work will be to simulate query access to F_1. Once we have this, then we can invoke the algorithm for the case $k = 1$ to reconstruct F_1 and then subtract off the value at F_1 to obtain query access to F_2 and then learn F_2.

A testing procedure One intermediate procedure that we will use often is a "testing" procedure. Our algorithm will often iterate over some "small" number of choices and hence output a small number of potential $\Sigma\Pi\Sigma\Pi(2)$ circuits. Many of these might be spurious, yet there will be a guarantee that at least one of the outputs is the correct one. Our algorithm will have to test which one is the correct one. For instance, our algorithm will iterate over some "small" number

[1] More generally, the work of [Vol17] shows how to derandomize [GKL12] for all fields, given an oracle for computing square roots.

of choices over H' for a multiplication gate in C, such that only one of these choices might be the right choice. We will have to detect what the right choice is. In order to do this, we run the reconstruction algorithm on the "remaining part" i.e. $C - H'$, for the various choices H' and then let the output be their sum. We will then test if the output circuit computes the same polynomial as C. This can be done by polynomial identity testing for multilinear $\Sigma\Pi\Sigma\Pi(4)$ circuits. Since one of the choices will ultimately work, we will eventually find a suitable reconstruction. The randomized version of PIT for $\Sigma\Pi\Sigma\Pi(4)$ circuits is easy and the deterministic version was given in [KMSV13, SV18].

Reducing to the case of large distance between the gates Suppose that the distance between F_1 and F_2 at most m, where we will set $m = \mathcal{O}(\log^2(s))$. Observe that if C is a circuit of size s, then each irreducible factor of F_1 and F_2 is s-sparse. Therefore, in this case the polynomial resulting after dividing out $\gcd(F_1, F_2)$ from C is a $2s^m$-sparse polynomial. Hence C can be represented as a multilinear $\Sigma\Pi\Sigma\Pi(1)$ circuit of sparsity at most $2s^m$, and therefore can be learnt using the reconstruction algorithm for such circuits in quasi-polynomial-time [SV10].

Thus from now on we will assume WLOG that F_1 and F_2 are at least m-far, where $m = \Theta(\log^2(s))$.

Projecting down to only polylogarithmically-many variables We will now show how to find a suitable projection of the circuit which sets all but m variables to some constants in the underlying field. Let the projections of F_1 and F_2 be $\hat{F_1}$ and $\hat{F_2}$, respectively. Our projection will additionally have the property that one of the two gates, say $\hat{F_1}$ is a product of m univariate affine forms, and the distance between $\hat{F_1}$ and $\hat{F_2}$ is at least m.

We first show that such a projection exists. Since F_1 and F_2 are at least m-far, at least one of them, say F_1, has at least m factors that do not appear in F_2. Moreover, we can assume that each such factor has at least 2 monomials. We will assume WLOG that all factors have at least 2 monomials, because all single monomial factors can be multiplied with each other and then absorbed into some other factor.

From each of these m factors of F_1, we pick any single variable appearing in it to keep "alive" (for more details see the next paragraph). Thus we get a collection of m alive variables. Let us call this set of variables I. Now let $\bar{a} = (a_1, a_2, \ldots, a_n)$ be a suitably chosen (as described later) element of $\mathbb{F}^n$. For each x_i that is not in I, we set $x_i = a_i$. From the projection, we will require the property that $\hat{F_1}$ is a product of m affine forms (so each factor has 2 monomials), and that $\gcd(\hat{F_1}, \hat{F_2})$ is 1. It then will follow that the distance between $\hat{F_1}$ and $\hat{F_2}$ will be at least m.

It is easy to see that if $\bar{a}$ was a uniformly random element of $\mathbb{F}^n$ then the projection has the desired properties (if the field is large enough). Indeed, one can define a polynomial Φ of not too large degree (see definition 4.3) such that any $\bar{a}$ which is a nonzero of Φ will have the desired properties. We do not know what Φ is, yet it turns out that Φ is a product of sparse polynomials and hence we already have efficient black-box hitting sets for this class of circuits by the works of [KS01, SV15, SV18] (See Lemma 3.7 for a formal statement). Hence we can efficiently construct a small set of elements of $\mathbb{F}^n$ such that one of them will result in a good projection. Our algorithm will then iterate over all possible choices of elements from this set and prune out the bad choices at a later stage by the testing step.

Now we show how to pick the set I. Note that since we cannot "see" F_1 or F_2, we do not really know how to pick these m variables that comprise the set I. All we know is that such a set exists. However, this will be enough, since our algorithm will just iterate over all possible $\binom{n}{m}$ subsets of variables of size m. As per the above discussion, at least one of these subsets will be a "good" subset. All the "bad" choices we will be able to prune out by a testing step at a later stage (as discussed earlier). Thus we can essentially assume for the rest of the algorithm that we know the special set I of m variables to keep alive.

Note that we can simulate black-box query access to the projected circuit.

Learning the projected circuit Let $\hat{C} = \hat{F_1} + \hat{F_2}$. We claim that we can efficiently (and exactly) reconstruct $\hat{C}$. By this we mean that we can recover $\hat{F_1}$ and $\hat{F_2}$.

Recall that $\hat{F_1}$ is a product of m affine forms, each in one variable. The algorithm will try to "guess" these affine forms by "guessing" each of the $2m$ coefficients of these affine forms. The way it will do this is by iterating over all possible choices of the guess. There are only $\binom{|\mathbb{F}|}{2m}$ many possible choices, and this is only quasi-polynomial in the circuit size and the field size, so this is a price we can pay. Notice that a crucial reason we wanted F_1 to become a product of affine forms was precisely for this guessing step. In general, a polynomial in m variables can have $2^{\Omega(m)}$ many monomials. Therefore guessing all the coefficients can be too expensive. However, since we have additionally ensured that $\hat{F_1}$ was a product of affine linear forms, we only have to guess $2m$ coefficients[2]. Once we "guessed" the correct value for $\hat{F_1}$ then we can subtract it off from $\hat{C}$ and obtain black-box query access to $\hat{F_2}$ and learn it as well.

One could hope that as before this is enough. That is, one of the guesses is the "correct" guess and the rest can be pruned out by testing if the resulting reconstructed circuit is equivalent to $\hat{C}$. The trouble might be if $\hat{C}$ can be written as a sum of two products of variable disjoint multilinear polynomials in more than one way. Suppose $\hat{C} = \hat{F_1} + \hat{F_2} = \hat{G_1} + \hat{G_2}$ where $\hat{F_1}, \hat{F_2}, \hat{G_1}, \hat{G_2}$ are all distinct multilinear $\Sigma\Pi\Sigma\Pi(1)$ circuits. In this case, because of the ambiguity of representation, it will be very difficult to recover $\hat{F_1}$ given oracle access to $\hat{C}$. Thankfully, this situation does not occur. If $\hat{F_1} + \hat{F_2} = \hat{G_1} + \hat{G_2}$ (where $\hat{G_1}$ and $\hat{G_2}$ have similar properties as $\hat{F_1}$ and $\hat{F_2}$) then $\hat{F_1} + \hat{F_2} - \hat{G_1} - \hat{G_2} \equiv 0$. Since $\gcd(\hat{F_1}, \hat{F_2}) = 1$, the circuit is "simple", i.e. there is no factor appearing in all the gates. If it is also minimal (i.e. no strict subset of the multiplication gates sums to 0), then by the *sparsity bound* proved in [SV18], it will follow that each of the multiplication gates must compute a "somewhat" sparse polynomial. Since $\hat{F_1}$ has at least 2^m monomials, this violates the sparsity bound and implies that the circuit $\hat{F_1} + \hat{F_2} - \hat{G_1} - \hat{G_2}$ is not minimal. In other words, up to renaming of the gates, the two representations of $\hat{C}$ must be equivalent. Therefore since $\hat{C}$ has a unique representation as a small $\Sigma\Pi\Sigma\Pi(2)$ circuit, the algorithm of guessing and pruning out the bad guess will indeed recover $\hat{F_1}$ and $\hat{F_2}$.

There is still one small issue. The algorithm will return the *set* $\left\{\hat{F_1}, \hat{F_2}\right\}$. We may not know which of the two polynomials is $\hat{F_1}$ (in the case when both end up being a product of m affine forms). This will become a bit of an issue later on that we will see how to resolve. Let us call this the "F_1-F_2 ambiguity" issue.

For now, if we are confused between the two gates, we arbitrarily pick one of them and call it $\hat{F_1}$. (Another way of doing this is that we can iterate over both choices and one of them will be correct). Thus for now let us assume that we know what $\hat{F_1}$ is.

[2]we actually end up combining all the leading coefficients into a single one, therefore we only have to guess $m+1$ parameters.

Observe that by setting the m alive variables in $\hat{F}_1$ to the corresponding entries of $\bar{a}$ (that was sampled earlier for the projection step), we recover $F_1(\bar{a})$.

Obtaining query access to F_1 We will now show that for any $\bar{b} \in \mathbb{F}^n$, there is an efficient algorithm that will compute $F_1(\bar{b})$. Once we show this, this means that we get oracle access to F_1 and hence can recover it via the algorithm for $\Sigma\Pi\Sigma\Pi(1)$ circuits.

Pick any $\bar{b} \in \mathbb{F}^n$. Consider the line through $\bar{a}$ and $\bar{b}$ given by $\ell_{\bar{a},\bar{b}}(t) \overset{\Delta}{=} (1-t)\bar{a} + t\bar{b}$. We already know $F_1(\bar{a})$. We want to compute $F_1(\bar{b})$. In order to to do this, we will recover F_1 restricted to $\ell_{\bar{a},\bar{b}}(t)$ to obtain a univariate polynomial in t. Evaluating at $t = 1$ will give us the desired value. This strategy mirrors the list-decoding algorithm of [STV01] and the factorization algorithm of [KT90],

Let $F_1^{(\bar{a},\bar{b})}(t)$ be the univariate polynomial $F_1(\ell_{\bar{a},\bar{b}}(t))$. This is a polynomial in t of degree at most n. Moreover, at $t = 0$, it evaluates to $F_1(\bar{a})$. If we are able to successfully evaluate $F_1^{(\bar{a},\bar{b})}(t)$ at any $n + 1$ distinct points then we can recover $F_1^{(\bar{a},\bar{b})}(t)$ entirely by interpolation and then evaluate it at $t = 1$ to get the value at $\bar{b}$.

If we were happy with a randomized algorithm, we could then pick $n + 1$ random points on the line and show that with high probability we can successfully evaluate $F_1^{(\bar{a},\bar{b})}$ on all these points. To get a deterministic algorithm we will sample several more points, argue that the value at "most" of these points can be accurately recovered and then use noisy polynomial interpolation (for instance the Berlekamp-Welch algorithm, see e.g. [Sud98] for more details) to recover the entire polynomial $F_1^{(\bar{a},\bar{b})}(t)$.

Let t_0 be any element of $\mathbb{F}$. We will try to compute the correct value of $F_1^{(\bar{a},\bar{b})}(t_0)$. In other words, we would like to be able to evaluate F_1 at a point $\bar{v} \overset{\Delta}{=} \ell_{\bar{a},\bar{b}}(t_0) = (1-t_0)\bar{a} + t_0\bar{b}$. Recall that when we evaluated F_1 at $\bar{a}$, all we needed of the point $\bar{a}$ was that it was a nonzero of some polynomial Φ of degree d. Now consider $\Phi(\ell_{\bar{a},\bar{b}}(t))$. Since $\Phi(\ell_{\bar{a},\bar{b}}(0)) = \Phi(\bar{a})$ is nonzero, this polynomial (in t) is not identically zero and hence will evaluate to zero at at most d points. Thus if we query the polynomial $F_1^{(\bar{a},\bar{b})}$ on "sufficiently many" (say $n + 2d + 1$) points on the line, we will be able to recover it by noisy polynomial interpolation.

There is one subtle issue, which is the aforementioned "F_1-F_2 ambiguity" issue. On any input $\bar{v}$ which is a nonzero of Φ, our algorithm will be able to return the value of F_1 at $\bar{v}$ and F_2 at $\bar{v}$ but it will not know which evaluation came from F_1 and which one came from F_2. This is a bit of an annoying issue, but thankfully one that is not too difficult to handle.

In order to resolve this issue, we do a hybrid argument. Instead of running the evaluation procedure just on $\bar{v}$, we run it on a sequence of elements $\gamma^0, \gamma^1, \ldots, \gamma^n$, where $\gamma^0 = \bar{a}$, $\gamma^n = \bar{v}$ and for every i, γ^i and γ^{i+1} differ in just one coordinate. Thus it suffices to do the following: given the evaluation of F_1 at γ^i, compute the correct evaluation of F_1 at γ^{i+1}, assuming both γ^i and γ^{i+1} are nonzeros of Φ. In order to do this, we will actually be asking for a bit more information from our algorithm. We will not just ask for the evaluation of F_1 at γ^i, we will instead ask for F_1 (presented as a product of affine forms) after a partial evaluation, where only the variables outside the special set I have been set to the corresponding coordinates of γ^i. Let us call this polynomial $F_1|_{\bar{x}_{[n]\setminus I}=\gamma^i_{[n]\setminus I}}$. It will suffice to show that given this polynomial we can compute $F_1|_{\bar{x}_{[n]\setminus I}=\gamma^{i+1}_{[n]\setminus I}}$.

Notice that since F_1 is multilinear and γ^i and γ^{i+1} differ in just one coordinate, the two polynomials

$$F_1|_{\bar{x}_{[n]\setminus I}=\gamma^i_{[n]\setminus I}} \text{ and } F_1|_{\bar{x}_{[n]\setminus I}=\gamma^{i+1}_{[n]\setminus I}}$$

differ in at most one factor. Our algorithm will "guess" $F_1|_{\bar{x}_{[n]\setminus I}=\gamma^{i+1}_{[n]\setminus I}}$ (by iterating over all possible factors of $F_1|_{\bar{x}_{[n]\setminus I}=\gamma^{i}_{[n]\setminus I}}$, and all possible expressions that factor could change to) and then try to verify whether that guess is correct. This can be done similar to as was done before when we had to compute $F_1|_{\bar{x}_{[n]\setminus I}=\bar{a}_{[n]\setminus I}}$ (i.e. the projection corresponding to $\bar{a}$). However this time the F_1-F_2 ambiguity issue will not arise since it cannot be that both $F_1|_{\bar{x}_{[n]\setminus I}=\gamma^{i+1}_{[n]\setminus I}}$ and $F_2|_{\bar{x}_{[n]\setminus I}=\gamma^{i+1}_{[n]\setminus I}}$ share most factors with $F_1|_{\bar{x}_{[n]\setminus I}=\gamma^{i}_{[n]\setminus I}}$ because otherwise then they would have have a common GCD.

2.2 The case of general k

At a high level, our argument for how to reconstruct $\Sigma\Pi\Sigma\Pi(k)$ circuits for larger values of k is very similar. We would like to somehow obtain oracle access to one of the gates (say F_1) and hence reconstruct it. We then subtract off that gate and get oracle access to a $\Sigma\Pi\Sigma\Pi(k-1)$ circuit which we assume can be reconstructed by induction on the top fan-in.

Thus the main goal will again be to evaluate F_1 at any point. To this end, we again take a projection so that the projected circuit only depends on polylogarithmic number of variables and the projected F_1 is a product of affine forms which we assume can be "guessed" and then the guess verified. There are a few subtle issues that arise. Earlier (in the case of $k = 2$), we could subtract off the projected F_1, get query access to the projected F_2 and hence learn it too. Knowing the projected F_2 was important in the lifting step, in particular in showing how to get query access to F_1, since we had to disambiguate between F_1 and F_2 when the algorithm wanted to evaluate F_1 at any point. In the case of larger k, after we subtract off the projected F_1, and trying to learn the remaining part of the circuit, we are not able to guarantee that the projected sum of remaining gates has a unique way of recovering it, and the algorithm gets a bit delicate. Yet, despite not having a unique representation, we are still able to show that when we use "minimal" representations while trying to learn the remaining part of the circuit, we are still able to carry out the disambiguation process and correctly evaluate F_1 at any chosen point (we still need to do the hybrid argument). For more details see Lemma 5.2.

3 Preliminaries

Let $\mathbb{F}$ denote a field, finite or otherwise, and let $\overline{\mathbb{F}}$ denote its algebraic closure.

3.1 Polynomials

A polynomial $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ *depends* on a variable x_i if there are two inputs $\bar{\alpha}, \bar{\beta} \in \overline{\mathbb{F}}^n$ differing only in the i^{th} coordinate for which $f(\bar{\alpha}) \neq f(\bar{\beta})$. We denote by $\text{var}(f)$ the set of variables that f depends on. We say that f is g are *similar* and denote by it $f \sim g$ if $f = \alpha g$ for some $\alpha \neq 0 \in \mathbb{F}$.

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

Definition 3.1 (Hybrids & Lines)**.** *Let $\bar{a}, \bar{b} \in \mathbb{F}^n$ and $0 \leq i \leq n$. We define the i-th* hybrid of $\bar{a}, \bar{b}$ *as $\gamma^i(\bar{a}, \bar{b}) \triangleq (b_1, \ldots, b_i, a_{i+1}, \ldots, a_n)$. In particular, $\gamma^0(\bar{a}, \bar{b}) = \bar{a}$ and $\gamma^n(\bar{a}, \bar{b}) = \bar{b}$.*
We define a line *passing through $\bar{a}$ and $\bar{b}$ as $\ell_{\bar{a},\bar{b}} : \mathbb{F} \to \mathbb{F}^n$, $\ell_{\bar{a},\bar{b}}(t) \triangleq (1-t) \cdot \bar{a} + t \cdot \bar{b}$. In particular, $\ell_{\bar{a},\bar{b}}(0) = \bar{a}$ and $\ell_{\bar{a},\bar{b}}(1) = \bar{b}$.*

We state below a well known result by Berlekamp and Welch which gives an efficient algorithm for noisy polynomial interpolation.

Lemma 3.2 (Berlekamp-Welch Algorithm (for a description see [Sud98]))**.** *Let $P(t)$ be a univariate polynomial of degree at most d. There exists a deterministic algorithm that given m evaluations of P with at most e errors outputs P, provided that $m - d > 2e + 1$.*

For two vectors $\bar{a}$ and $\bar{b} \in \mathbb{F}^n$, let $\mathrm{w_H}(\bar{a}, \bar{b})$ denote the Hamming distance between $\bar{a}$ and $\bar{b}$

3.2 Sparse Polynomials

In this section we discuss sparse polynomials, their properties and some related efficient algorithms which leverage these properties.

Definition 3.3 (Sparsity, Split and Maximum Split)**.** *An s-*sparse polynomial *is polynomial with at most s (non-zero) monomials. We denote by $\|f\|$ the* sparsity *of f. A polynomial $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ is s-*sparse-split *if it can be written as a product of (not necessarily irreducible) s-sparse polynomials.*
We define $\lambda(f)$ as the smallest non-negative integer s such that f is s-sparse-split.

It is easy to see that for any multilinear polynomials with sparsity s any factorization is s-sparse-split, and this was studied in [SV10].

Lemma 3.4 ([SV10])**.** *Let f be a multilinear polynomial and $hg = f$ then $\|f\| = \|g\| \cdot \|h\|$.*

We move on to an efficient reconstruction algorithm for sparse polynomials.

Lemma 3.5 ([KS01])**.** *Let $n, s, d \in \mathbb{N}$. There exists a deterministic algorithm that given n, s, d and a black-box access to an s-sparse polynomial $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ of degree d, uses* $\mathrm{poly}(n, s, d, \log|\mathbb{F}|)$ *field operations and outputs f (in its monomial representation).*

In [SV10], this algorithm was extended to multilinear sparse-split polynomials (which may not necessarily themselves be sparse).

Lemma 3.6 ([SV10])**.** *Let $n, s \in \mathbb{N}$. There exists a deterministic algorithm that given n, s and a black-box access to a multilinear s-split polynomial $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$, uses* $\mathrm{poly}(n, s, \log|\mathbb{F}|)$ *field operations and outputs f (as a product of s-sparse polynomials).*

As an easy corollary, Lemma 3.5, establishes the existence of an efficient hitting set for sparse polynomials. The next lemma extends this result and shows the existence of an efficient hitting set for a *product* of sparse polynomials. Indeed it was shown in [SV15] that if there is an efficient hitting set for any class of polynomials, then one can construct an efficient hitting set for a product of few polynomials from that class.

Lemma 3.7 ([KS01, SV15]). *There exists a deterministic algorithm that given $n, s, d, k \in \mathbb{N}$ outputs a set $\mathcal{SP}_{(n,s,d,k)}$ of size* $\mathrm{poly}(n, s, d, k)$ *such that any set of (at most) k non-zero s-sparse polynomials $f_1, \dots, f_k \in \mathbb{F}[x_1, x_2, \dots, x_n]$ of total degree at most d have a common non-zero in $\mathcal{SP}_{(n,s,d,k)}$. In other words, there exists $\bar{a} \in \mathcal{SP}_{(n,s,d,k)}$ such that $\forall i : f_i(\bar{a}) \neq 0$.*

Inspired by a similar notion of [KS09a], we define a distance measure between multiplication gates. This measure will play a crucial role in the analysis of our reconstruction algorithm. Roughly speaking, the log to the base s (i.e. $\log_s$) of the distance between two polynomials, each of them being s-sparse-split, is the number of factors that appear in only one of them.

Definition 3.8 (Distance). *For $f, g \in \mathbb{F}[x_1, x_2, \dots, x_n]$, we defined a* distance *function:*

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

The following observation is immediate given Lemma 3.4:

Observation 3.9. *If f and g are multilinear polynomials, then $\Delta(f, g) = \max\left\{\frac{\|f\|}{\|\gcd(f,g)\|}, \frac{\|g\|}{\|\gcd(f,g)\|}\right\}$.*

The next lemma lists the basic properties of the defined distance measure.

Lemma 3.10 (Properties). *Let f, g, h be multilinear polynomials $\in \mathbb{F}[x_1, x_2, \dots, x_n]$, then we have that,*

1. $\Delta(f, g) = \Delta(g, f)$.
2. $\Delta(f, g) \geq 1$, *moreover* $\Delta(f, g) = 1 \iff f \sim g$.
3. $\Delta(f, g) \leq \Delta(f, g, h) \leq \Delta(f, h) \cdot \Delta(g, h)$, *where* $\Delta(f, g, h) \triangleq \frac{\max\{\|f\|, \|g\|, \|h\|\}}{\|\gcd(f,g,h)\|}$.
4. $\Delta(g - f, f) \leq 2 \cdot \Delta(f, g)$

Proof. 1. Follows directly from the definition.

2. The property follows by noticing that $\gcd(f, g)$ is a factor of both f and g and using Lemma 3.4.

3. The first inequality directly follows from the noticing that $\gcd(f, g, h) \mid \gcd(f, g)$ and Lemma 3.4.

   For the second inequality, we can assume WLOG that $\|f\| \geq \|g\|$. Notice that, $\gcd(f, h) \cdot \gcd(g, h) \mid h \cdot \gcd(f, g, h)$.

Thus,

$$\|\gcd(f,h)\| \cdot \|\gcd(g,h)\| \le \|h\| \cdot \|\gcd(f,g,h)\| \qquad \text{(by Lemma 3.4)}$$

$$\frac{1}{\|\gcd(f,g,h)\|} \le \frac{\max\{\|h\|,\|g\|\}}{\|\gcd(f,h)\| \cdot \|\gcd(g,h)\|} \qquad \text{(By rearranging)}$$

$$\frac{\max\{\|f\|,\|g\|,\|h\|\}}{\|\gcd(f,g,h)\|} \le \frac{\max\{\|h\|,\|g\|\} \cdot \max\{\|f\|,\|g\|,\|h\|\}}{\|\gcd(f,h)\| \cdot \|\gcd(g,h)\|} \qquad \text{(Multiplying by } \max\{\|f\|,\|g\|,\|h\|\}\text{)}$$

$$\frac{\max\{\|f\|,\|g\|,\|h\|\}}{\|\gcd(f,g,h)\|} \le \frac{\max\{\|h\|,\|g\|\} \cdot \max\{\|f\|,\|h\|\}}{\|\gcd(f,h)\| \cdot \|\gcd(g,h)\|} \qquad \text{(Since, } \|f\| \ge \|g\| \text{)}$$

$$\Delta(f,g,h) \le \Delta(f,h) \cdot \Delta(g,h)$$

4. Let $h = \gcd(f,g)$. We can write $f = f'h$ and $g = g'h$. Then

$$\Delta(g-f,f) = \Delta(g'h-f'h,f'h) = \Delta(g'-f',f') \le \max\{\|g'-f'\|,\|f'\|\} \le \|g'\|+\|f'\| \le 2\cdot\Delta(f,g)$$

□

3.3 The Operator D_j

This operator was previously used in [GKL12, KMSV13, SV18]. In this section we formally define the operator and list some properties that immediately follow (and will be used later).

Definition 3.11. *For $j \in [n]$ let $D_j(P,Q)$ be the polynomial defined as follows:*

$$D_j(P,Q)(\bar{x}) \triangleq \left| \begin{pmatrix} P & P|_{x_\ell=0} \\ Q & Q|_{x_\ell=0} \end{pmatrix} \right| (\bar{x}) = (P \cdot Q|_{x_\ell=0} - P|_{x_\ell=0} \cdot Q)(\bar{x}).$$

Note that D_j is a bilinear transformation. The following lemma lists several useful properties of D_j that are easy to verify.

Lemma 3.12 ([GKL12, KMSV13, SV18])**.** *Let $P,Q,R \in \mathbb{F}[x_1,x_2,\ldots,x_n]$ be multilinear polynomials and let $j \in [n]$. Then the following properties hold:*

1. *Let R be such that $j \notin \mathrm{var}(R)$ then $D_j(R \cdot Q, P) = R \cdot D_j(Q,P)$.*
2. *$\|D_j(Q,P)\| \le \|Q\| \cdot \|P\|$.*
3. *Let $I \subseteq [n]$ be such that $I \ne \emptyset$. Then, if $\prod_{j\in[n]} D_j(Q,P)(\bar{a}) \ne 0$ then $\gcd(P,Q)|_{\bar{x}_{[n]\setminus I}=\bar{a}_{[n]\setminus I}} = \gcd(P|_{\bar{x}_{[n]\setminus I}=\bar{a}_{[n]\setminus I}}, Q|_{\bar{x}_{[n]\setminus I}=\bar{a}_{[n]\setminus I}})$.*

Corollary 3.13. *Let $P,Q \in \mathbb{F}[x_1,x_2,\ldots,x_n]$ be multilinear polynomials such that $P = \prod_{i\in[k]} P_i$ and $Q = \prod_{i'\in[k']} Q_i$. If $j \notin \mathrm{var}(\prod_{i=2}^{k} P_i \cdot \prod_{i'=2}^{k'} Q_{i'})$ then $D_j(P,Q) = D_j(P_1,Q_1) \cdot \prod_{i=2}^{k} P_i \cdot \prod_{i'=2}^{k'} Q_{i'}$.*

3.4 Partial Derivatives

The concept of a *partial derivative* of a multivariate polynomial and its properties (for example: P depends on x_i if and only if $\frac{\partial P}{\partial x_i} \not\equiv 0$) are well-known and well-studied for continuous domains (such as $\mathbb{R}$ and $\mathbb{C}$). Here we use a well-known variant for polynomials over arbitrary fields. Namely, the *discrete* partial derivatives. Discrete partial derivatives will play a useful role in the analysis of our algorithms.

Definition 3.14. *Let $P(\bar{x})$ be an n variate polynomial over a field $\mathbb{F}$. We define the* discrete partial derivative *of $P(\bar{x})$ with respect to x_i as $\frac{\partial P}{\partial x_i} = P|_{x_i=1} - P|_{x_i=0}$.*

Notice that if P is a multilinear polynomial then this definition coincides with the "analytical" one when $\mathbb{F} = \mathbb{R}$ or $\mathbb{C}$. The following lemma is easy to verify and we will use it implicitly from now on.

Lemma 3.15 (See e.g. [SV15])**.** *Let $P \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a multilinear polynomial and let $i \in [n]$. Then the following properties hold:*

- *$x_i \in \mathrm{var}(P)$ if and only if $\frac{\partial P}{\partial x_i} \not\equiv 0$.*
- *$\forall j \neq i$ $\frac{\partial P}{\partial x_i}|_{x_j=a} = \frac{\partial}{\partial x_i}\left(P|_{x_j=a}\right)$.*
- *$\|\frac{\partial P}{\partial x_i}\| \leq \|P\|$.*

Since the above properties trivially hold, we will use them implicitly.

4 Depth-4 Multilinear Circuits

In this section, we formally present the model of depth-4 multilinear circuits and some related definitions. Similar definitions were given in [GKL12, KMSV13, SV18].

Definition 4.1. *A depth-4 $\Sigma\Pi\Sigma\Pi(k)$ circuit C has four layers of alternating Σ and Π gates (the top Σ gate is at level one) and it computes a polynomial of the form*

$$C(\bar{x}) = \sum_{i=1}^{k} F_i(\bar{x}) = \sum_{i=1}^{k} \prod_{j=1}^{d_i} P_{ij}(\bar{x})$$

where the $P_{ij}(\bar{x})$-s are polynomials computed by the last two layers of $\Sigma\Pi$ gates of the circuit and are the inputs to the Π gates at the second level. A multilinear $\Sigma\Pi\Sigma\Pi(k)$ *circuit is a $\Sigma\Pi\Sigma\Pi(k)$ circuit in which each multiplication gate F_i computes a multilinear polynomial.*

The requirement that the F_i-s compute multilinear polynomials implies that for each $i \in [n]$ the polynomials $\{P_{ij}\}_{j\in[d_i]}$ are variable-disjoint. Note that if the circuit is of size s then each P_{ij} is s-sparse. Indeed $\lambda(F_i) \leq s$, for each $i \in [k]$. For every $A \subseteq [k]$ we define the subcircuit C_A of C as $C_A \triangleq \sum_{i\in A} F_i$. We define $\gcd(C) \triangleq \gcd(F_1, \ldots, F_k)$. We say that the circuit C is *simple* if $\gcd(C) = 1$. We say that the circuit C is *minimal* if no proper subcircuit of C computes the zero polynomial. That is, for every $\emptyset \subsetneq A \subsetneq [k]$ it holds that $C_A \not\equiv 0$. For two circuits C and C', we say that $C \equiv C'$ if both circuits compute the same polynomial.

Definition 4.2 (Distant Circuits)**.** *Let $C(\bar{x}) = \sum_{i=1}^{k} F_i(\bar{x})$ be a multilinear $\Sigma\Pi\Sigma\Pi(k)$ circuit. We say that F_i and F_j are r-*far*, if $\Delta(F_i, F_j) \geq r$; F_i is r-*distant*, if for all $j \neq i$, F_i and F_j are r-far. Finally we say that C is r-*distant*, if each pair of multiplication gates F_i and F_j is r-far.*

Our first contribution is a structural result for distant circuits. Roughly speaking, we show that we can project a circuit down to a "small" number of variables such that one of the multiplication gates becomes to a depth-3 gate (a product of affine forms), while still being "sufficiently" distant. To this end, we require the following definition.

Definition 4.3. *Let $C(\bar{x}) = \sum_{i=1}^{k} F_i(\bar{x})$ be a multilinear $\Sigma\Pi\Sigma\Pi(k)$ circuit. We define the polynomial Φ_C as follows:*

$$\Phi_C \triangleq \prod_{j,i \,:\, D_j(F_1,F_i)\not\equiv 0} D_j(F_1, F_i) \cdot \prod_{i,j \,:\, \frac{\partial F_i}{\partial x_j}\not\equiv 0} \frac{\partial F_i}{\partial x_j} \cdot \prod_{i,j \,:\, F_i|_{x_j=0}\not\equiv 0} F_i|_{x_j=0}$$

We will need the following observation about Φ_C, the proof of which is immediate from Lemma 3.12 and Corollary 3.13.

Observation 4.4. *Φ_C is non-zero polynomial in $\mathbb{F}[x_1, x_2, \dots, x_n]$ which is a product of at most $3kn^2$, s^2-sparse polynomials of degree at most $2n$.*

We now give the structural result.

Lemma 4.5. *Let C be an r-distant circuit with $\lambda(C) \leq s$. Suppose that F_1 is the gate with maximum sparsity i.e. $\|F_1\| \geq \|F_j\|$ for $j \geq 2$. Then there exists a subset $I \subseteq [n]$ of size $|I| \leq k \log_s(r)$ such that for each $\bar{a} \in \mathbb{F}^n$ satisfying $\Phi_C(\bar{a}) \neq 0$: $F_1|_{\bar{x}_{[n]\setminus I}=\bar{a}_{[n]\setminus I}}$ is a product of affine linear forms and is $r^{1/\log s}$-distant in $C|_{\bar{x}_{[n]\setminus I}=\bar{a}_{[n]\setminus I}}$.*

Proof. Note that for any j, since $\lambda(C) \leq s$, the polynomial $h \triangleq \frac{F_1}{\gcd(F_1,F_j)}$ will have at least $\log_s r$ many factors and sparsity of these factors is at least 2. In particular, F_1 has at least $\log_s r$ factors, each with at least two monomials, which do not divide F_j. Taking the union of this set of factors over each j between 2 and k, we get a set $\mathcal{T}$ of at most $k \log_s r$ many factors of F_1, such that for each $j \neq 1$, there are at least $\log_s r$ factors of sparsity at least 2 from this set that do no divide F_j.

Also, since, F_1 is multilinear its factors will be variable disjoint. We arbitrarily pick one variable from each factor of $\mathcal{T}$ and call that set of variables I. Then clearly, $|I| \leq k \log_s r$. For each $i \notin I$, we will set x_i to a_i.

Since, $\Phi_C(\bar{a}) \neq 0$, Lemma 3.12 implies that,

$$\gcd(F_1, F_j)|_{\bar{x}_{[n]\setminus I}=\bar{a}_{[n]\setminus I}} = \gcd(F_1|_{\bar{x}_{[n]\setminus I}=\bar{a}_{[n]\setminus I}}, F_j|_{\bar{x}_{[n]\setminus I}=\bar{a}_{[n]\setminus I}}) \qquad \forall j \neq 1. \tag{1}$$

What the above statement is saying is that basically that since $\Phi_C(\bar{a}) \neq 0$, after restricting some of the variables to the corresponding elements of $\bar{a}$ we do not introduce any *new* terms in the GCD of two multiplication gates. The proof of this fact just uses the nonzeroness first term in the definition of Φ_C. The nonzeroness of the second term will show that the restricted F_1 still depends on all the 'alive" variables, and the nonzeroness of third term shows that none of the alive variables divides the restricted F_1. This implies that $F_1|_{\bar{x}_{[n]\setminus I}=\bar{a}_{[n]\setminus I}}$ will be a product of affine linear forms, each of sparsity exactly 2.

Once we have these facts it follows quite easily that F_1 will still be distant in the projected circuit, since

$$\begin{aligned}\Delta(F_1|_{\bar{x}_{[n]\setminus I}=\bar{a}_{[n]\setminus I}}, F_j|_{\bar{x}_{[n]\setminus I}=\bar{a}_{[n]\setminus I}}) &\geq \frac{\|F_1|_{\bar{x}_{[n]\setminus I}=\bar{a}_{[n]\setminus I}}\|}{\|\gcd(F_1|_{\bar{x}_{[n]\setminus I}=\bar{a}_{[n]\setminus I}}, F_j|_{\bar{x}_{[n]\setminus I}=\bar{a}_{[n]\setminus I}})\|} \\ &= \frac{\|F_1|_{\bar{x}_{[n]\setminus I}=\bar{a}_{[n]\setminus I}}\|}{\|\gcd(F_1, F_j)|_{\bar{x}_{[n]\setminus I}=\bar{a}_{[n]\setminus I}}\|} && \textit{By}\,(1) \\ &= \left\|\frac{F_1}{\gcd(F_1, F_j)}|_{\bar{x}_{[n]\setminus I}=\bar{a}_{[n]\setminus I}}\right\| && \textit{By}\,(3.4)\end{aligned}$$

Thus,

$$\Delta(F_1|_{\bar{x}_{[n]\setminus I}=\bar{a}_{[n]\setminus I}}, F_j|_{\bar{x}_{[n]\setminus I}=\bar{a}_{[n]\setminus I}}) \geq \left\|\frac{F_1}{\gcd(F_1, F_j)}|_{\bar{x}_{[n]\setminus I}=\bar{a}_{[n]\setminus I}}\right\| \geq 2^{\log_s r} = r^{1/\log s}.$$

□

Next is an important but easy observation that shows that we can decrease the number of multiplication gates, if a circuit is not distant.

Observation 4.6. *Suppose C is not r-distant, then C has an equivalent (i.e. computing the same polynomial) $\Sigma\Pi\Sigma\Pi(k-1)$ circuit C' with $\lambda(C') \leq \max\{s, 2r\}$.*

Proof. By definition, there exists F_i, F_j such that $\Delta(F_i, F_j) < r$. Then replace $F_i + F_j$ with a new single multiplication gate $H' \triangleq F_i + F_j$ which is formed by taking the GCD of the two gates and then expanding out the remaining polynomial (which one would get after diving by the GCD) as a single somewhat sparse multilinear polynomial. □

We conclude this section with two results. The first is a structural result on multilinear $\Sigma\Pi\Sigma\Pi(k)$ circuits that compute the zero polynomial.

Lemma 4.7 (The Sparsity Bound - [SV18])**.** *There exists an non-decreasing function $\varphi(k, s) \leq s^{5k^2}$ such that if $C(\bar{x}) = \sum_{i=1}^{k} F_i(\bar{x})$ is a simple and minimal, multilinear $\Sigma\Pi\Sigma\Pi(k)$ circuit computing the zero polynomial with $\lambda(C) \leq s$, then for each $i \in [k]$ it holds that $\|F_i\| < \varphi(k, s)$.*

The following is immediate.

Corollary 4.8. *Let $C(\bar{x}) = \sum_{i=1}^{k} F_i(\bar{x})$ be a minimal, multilinear $\Sigma\Pi\Sigma\Pi(k)$ circuit computing the zero polynomial with $\lambda(C) \leq s$. Then for all $i \neq j \in [k] : \Delta(F_i, F_j) < \varphi(k, s)$.*

The second result provides a deterministic black-box PIT (polynomial identity testing) algorithm for multilinear $\Sigma\Pi\Sigma\Pi(k)$ circuits.

Lemma 4.9 (Black-Box PIT for $\Sigma\Pi\Sigma\Pi(k)$ circuits - [SV18])**.** *Let k, n, s be integers. There is an explicit set $\mathcal{H}$ of size $n^{\mathcal{O}(k)} \cdot s^{\mathcal{O}(k^3)}$, that can be constructed in time $n^{\mathcal{O}(k)} \cdot s^{\mathcal{O}(k^3)}$, such that the following holds. Let $P \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a non-zero polynomial computed by a multilinear $\Sigma\Pi\Sigma\Pi(k)$ circuit of size s on n variables. Then $P|_{\mathcal{H}} \not\equiv 0$.*

5 The Reconstruction Algorithm

We now prove our main result. We first restate it below with more precise parameters. The result below refers to Algorithm 1, which is our main algorithm. Algorithm 1 as a subroutine will invoke Algorithm 3, which in turn will invoke Algorithm 2. These are all described in the coming pages. At a high level Algorithm 1 gets as input black-box access to a multilinear $\Sigma\Pi\Sigma\Pi(k)$ circuit C and tries to reconstruct it. It first checks if C can be represented as a $\Sigma\Pi\Sigma\Pi(k-1)$ circuit with slightly larger size parameters. If not, then it knows that C must be a distant circuit. Now the algorithm will try to obtain query access to F_1 (and then learn F_1) and for this, it will use Algorithm 3 as a subroutine. In order to use Algorithm 3, Algorithm 1 iterates over all possible restrictions of C to polylogarithmically-many variables. At least of these restrictions will have "nice enough" properties which will allow Algorithm 3 to be effective. All the restrictions that were not good and do not lead to the output being the correct circuit will be pruned out later by a PIT step. Once F_1 is learnt, the algorithm will subtract it off to get query access to a $\Sigma\Pi\Sigma\Pi(k-1)$ circuit which can be then learnt recursively.

```
   Input: $n, k, s$, black-box access to a $\Sigma\Pi\Sigma\Pi(k)$ circuit $C$ with $\lambda(C) \leq s$
   Output: $\Sigma\Pi\Sigma\Pi(k')$ circuit $C'$ such that $C' \equiv C$ with $k' \leq k$ and $\lambda(C') \leq 2^{(\log s)^{\nu(k,k')}}$

 1 if $k = 1$ then
 2 |   Invoke the reconstruction algorithm from Lemma 3.6;
   /* $k \geq 2$                                                          */
 3 $r \leftarrow 2^{(\log s)^{2^{2^k}}-1}$;
 4 Invoke the algorithm recursively on $C$ with $k = k-1$ and $s = 2r$ to obtain $\hat{C}$;
 5 if $C \equiv \hat{C}$ (using the black-box PIT algorithm from Lemma 4.9) then
 6 |   return $\hat{C}$
   /* $C$ is $r$-distant                                                  */
   /* ``guess'' $I$ and $\bar{a}$                                         */
 7 foreach $I \subseteq [n]$ of size $|I| \leq k\log_s(r)$ and $\bar{a} \in \mathcal{SP}_{(n,s^2,2n,3kn^2)}$ do
   |   /* ``guess'' $F_1|_{\bar{x}_{[n]\setminus I}=\bar{a}_{[n]\setminus I}}$          */
 8 |   foreach $(u_0, \bar{u}_I) \in (\mathbb{F}\setminus\{0\}) \times \mathbb{F}^{|I|}$ do
 9 |   |   $H_0 \leftarrow u_0 \cdot \prod_{i\in I}(x_i - u_i)$;
10 |   |   Invoke the reconstruction algorithm from Lemma 3.6 to obtain $\hat{H}$. Simulate each
   |   |     query point $\bar{b}$ of the reconstruction algorithm by invoking Algorithm 3;
11 |   |   Invoke the algorithm recursively on $C - \hat{H}$ with $k = k-1$ and $s = s$ to obtain $\hat{C}$ ;
12 |   |   if $C \equiv \hat{C} + \hat{H}$ (using the black-box PIT algorithm from Lemma 4.9) then
13 |   |   |   return $\hat{C} + \hat{H}$
14 |   return $\perp$
```

Algorithm 1: Reconstruct.

Theorem 5.1. *Given n, s, k and black-box access to a multilinear $\Sigma\Pi\Sigma\Pi(k)$ circuit C over a finite field $\mathbb{F}$ in n variables and with $\lambda(C) \leq s$ as input, Algorithm 1 runs in time $|\mathbb{F}|^{(\log s)^{\nu(k,1)}}$ and outputs a $\Sigma\Pi\Sigma\Pi(k')$ circuit C' such that $C' \equiv C$ with $k' \leq k$ and $\lambda(C') \leq 2^{(\log s)^{\nu(k,k')}}$ where*

$\nu(k,k') \triangleq 2^{(2^{k+1}-2^{k'+1})}$.

Proof. The proof will be by induction on k. For $k=1$ the claim follows from Lemma 3.6. Observe that $2^{(\log s)^{\nu(k,k)}} = s$. Now assume $k \geq 2$. First, observe that due to the PIT tests, the algorithm outputs $\perp$ if and only if it could not find a circuit equivalent to C. We will argue now that the algorithm always finds such a circuit. Consider two cases:

Suppose C is not r-distant. Then by Observation 4.6, it has an equivalent $\Sigma\Pi\Sigma\Pi(k-1)$ circuit C' with $\lambda(C') \leq 2r$. By the induction hypothesis, the algorithm will output a $\Sigma\Pi\Sigma\Pi(\hat{k})$ circuit $\hat{C}$ such that $\hat{C} \equiv C' \equiv C$ with $\hat{k} \leq k-1 \leq k$ and

$$\lambda(\hat{C}) \leq 2^{(\log(2r))^{\nu(k-1,k')}} = \left(2^{(\log s)^{2^{2^k}-1+1}}\right)^{2^{(2^k-2^{k'}+1)}} = 2^{(\log s)^{\nu(k,k')}}.$$

Suppose C is r-distant. Assume WLOG that F_1 is the gate with maximum sparsity in C (otherwise we just relabel the gates so that the gate of maximum sparsity if called F_1). By Observation 4.4 and Lemma 3.7, there exists $\bar{a} \in \mathcal{SP}_{(n,s^2,2n,3kn^2)}$ such that $\Phi_C(\bar{a}) \neq 0$. Furthermore, by Lemma 4.5 there exists a subset $I \subseteq [n]$ of size $|I| \leq k \log_s(r)$ such that $F_1|_{\bar{x}_{[n]\setminus I}=\bar{a}_{[n]\setminus I}}$ is a product of univariates. Consequently, there exists H_0 such that $H_0 = F_1|_{\bar{x}_{[n]\setminus I}=\bar{a}_{[n]\setminus I}}$. Given these, Algorithm 3 will simulate oracle access to F_1 by computing $F_1(\bar{b})$ on any query point $\bar{b} \in \mathbb{F}^n$ (Lemma 5.4), where the induction hypothesis provides the required reconstruction algorithm $\mathcal{A}$.

Given oracle access to F_1, the algorithm in Lemma 3.6 will reconstruct $\hat{H} \equiv F_1$. As $C - F_1$ is a $\Sigma\Pi\Sigma\Pi(k-1)$ circuits with $\lambda \leq s$, by the induction hypothesis, the algorithm will reconstruct $\hat{C} \equiv C - \hat{H}$ and hence return $\hat{C} + \hat{H} = C - \hat{H} + \hat{H} \equiv C$.

Runtime Analysis: Let $T(k,s)$ denote the runtime of the algorithm given k and s, respectively. Based on the algorithm, we obtain that

$$T(k,s) \leq \max\left\{T(k-1,2r) + \mathrm{poly}(r)\ ,\ n^{k\log_s r} \cdot |\mathbb{F}|^{k\log_s r} \cdot n \cdot |\mathbb{F}|^2 \cdot T(k-1,s) \cdot \mathrm{poly}(s^{k^3})\right\}.$$

Solving for k implies the answer.

□

5.1 Evaluation

In this section we will demonstrate how, given black-box access to a $\Sigma\Pi\Sigma\Pi(k)$ circuit, to evaluate a particular multiplication gate at a point of choice. Our algorithm will assume it has access to a reconstruction algorithm for $\Sigma\Pi\Sigma\Pi(k-1)$ circuits.

More formally, given a black-box access to a multilinear $\Sigma\Pi\Sigma\Pi(k)$ circuit C and $\bar{b} \in \mathbb{F}^n$ we would like to compute $F_1(\bar{b})$. To this end, we apply Theorem 5.1 via the inductive hypothesis. We assume that we are given as a subroutine a reconstruction algorithm $\mathcal{A}$ that given a black-box access to a $\Sigma\Pi\Sigma\Pi(m)$ circuit C with $m \leq k-1$ and $\lambda(C) \leq s$, outputs a $\Sigma\Pi\Sigma\Pi(m')$ circuit C' with $m' \leq m$ and $\lambda(C') \leq 2^{(\log s)^{\nu(m,m')}}$. Additionally, we assume that we are given the set I guaranteed by Lemma 4.5, an assignment $\bar{a} \in \mathbb{F}^n$ such that $\Phi_C(\bar{a}) \neq 0$ and the polynomial $H_0 = u_0 \cdot \prod_{i\in I}(x_i - u_i) = F_1|_{\bar{x}_{[n]\setminus I}=\bar{a}_{[n]\setminus I}}$.

Given all this information, we first show how to evaluate F_1 at a point $\bar{b}$ which has the additional properties that $w_H(\bar{a}, \bar{b}) = 1$ (where w_H denotes Hamming distance), and $\Phi(\bar{b}) \neq 0$. This will be accomplished by Algorithm 2, and we prove correctness of this statement in Lemma 5.2. Then in Algorithm 3, it will be shown how to compute $F_1(\bar{b})$ without assuming $w_H(\bar{a}, \bar{b}) = 1$ and $\Phi(\bar{b}) \neq 0$. In order to do this, Algorithm 3 will use Algorithm 2 as a subroutine. Correctness of Algorithm 3 is proved in Lemma 5.4.

```
Input: Reconstruction Algorithm $\mathcal{A}$ for reconstructing $\Sigma\Pi\Sigma\Pi(k-1)$ circuits
$k, n, s \in \mathbb{N}$
Assignments: $\bar{a}, \bar{b} \in \mathbb{F}^n$ such that $w_H(\bar{a}, \bar{b}) = 1$
Subset: $I \subseteq [n]$
Polynomial: $H_0 = u_0 \cdot \prod_{i \in I}(x_i - u_i)$
Black-box access to a $\Sigma\Pi\Sigma\Pi(k)$ circuit $C$
Output: Polynomial: $H = u'_0 \cdot \prod_{i \in I}(x_i - u'_i)$

1  $H \leftarrow \perp, m \leftarrow k$;
2  $C_{\bar{b}} \triangleq C|_{\bar{x}_{[n]\setminus I} = \bar{b}_{[n]\setminus I}}$;
   /* ``guess'' $F_1|_{\bar{x}_{[n]\setminus I} = \bar{b}_{[n]\setminus I}}$ by trying to change one univariate factor of $H_0$ at
      a time */
3  foreach $i' \in I$, $u'_0 \in \mathbb{F} \setminus \{0\}$ and $u' \in \mathbb{F}$ do
4      $H' \leftarrow u'_0 \cdot \prod_{i \in I \setminus \{i'\}}(x_i - u_i) \cdot (x_{i'} - u')$;
       /* Attempt to reconstruct the circuit $C_{\bar{b}} - H'$ as a $\Sigma\Pi\Sigma\Pi(k')$ circuit with
          $k' \leq k-1$ */
5      for $k' \leftarrow k-1$ to $1$ do
6          Invoke Algorithm $\mathcal{A}$ on $C_{\bar{b}} - H'$ with $k = k'$ and $s = s$ to obtain $C'$.
7          Let $m'$ denote the top fan-in of $C'$ ;
8          if $C' \neq \perp$ and $m' < m$ then
9              $H \leftarrow H'$;
10             $m = m'$;
11 return $H$ /* Output $H'$ for which the corresponding $C'$ has the smallest top
      fan-in */
```

Algorithm 2: Evaluate Distance 1

Lemma 5.2. *Let $k, n, s \in \mathbb{N}$ and let $r \triangleq 2^{(\log s)^{2^{2^k}} - 1}$. Then given:*

1. *$k, n, s \in \mathbb{N}$*

2. *Black-box access to a $\Sigma\Pi\Sigma\Pi(k)$ circuit $C = \sum_{i=1}^{k} F_i$ satisfying Lemma 4.5*

3. *A reconstruction algorithm $\mathcal{A}$ that given a black-box access to a $\Sigma\Pi\Sigma\Pi(m)$ circuit C with $m \leq k-1$ and $\lambda(C) \leq s$, outputs a $\Sigma\Pi\Sigma\Pi(m')$ circuit C' with $m' \leq m$ and $\lambda(C') \leq 2^{(\log s)^{\nu(m,m')}}$.*

4. *Assignments: $\bar{a}, \bar{b} \in \mathbb{F}^n$ such that $\mathrm{w_H}(\bar{a}, \bar{b}) = 1$ and $\Phi_C(\bar{a}) \neq 0, \Phi_C(\bar{b}) \neq 0$*

5. *The subset $I \subseteq [n]$ guaranteed by Lemma 4.5*

6. *Polynomial: $H_0 = u_0 \cdot \prod_{i \in I}(x_i - u_i)$, where $H_0 = F_1|_{\bar{x}_{[n]\setminus I} = \bar{a}_{[n]\setminus I}}$*

Algorithm 2 runs in time $|\mathbb{F}|^{(\log s)^{\nu(k,1)}}$ outputs $H = F_1|_{\bar{x}_{[n]\setminus I} = \bar{b}_{[n]\setminus I}}$.

Proof. Let H be the output of the algorithm and let $C' = \sum_{j=1}^{m} G_j$ be the corresponding circuit. In addition, let $H_1 \triangleq F_1|_{\bar{x}_{[n]\setminus I} = \bar{b}_{[n]\setminus I}}$. By Lemma 4.5, both H_0 and H_1 are products of univariates. Moreover, since $\mathrm{w_H}(\bar{a}, \bar{b}) = 1$, these products differ by at most one univariate factor. Therefore, one of the H'-s will be equal to H_1, thus ensuring $H \neq \perp$. Furthermore, observe that $\Delta(H_1, H_0) \leq 2$ and $\Delta(H, H_0) \leq 2$. Therefore,

$$\Delta(H_1 - H, H_1) \leq 2 \cdot \Delta(H, H_1) \leq 2 \cdot \Delta(H, H_0) \cdot \Delta(H_0, H) \leq 8.$$

Let us now assume for contradiction that $H \neq H_1$. By definition:

$$H_1 + \sum_{i=2}^{k} F_i|_{\bar{x}_{[n]\setminus I} = \bar{b}_{[n]\setminus I}} - H = C_{\bar{b}} - H = C' = \sum_{j=1}^{m} G_j \implies \tag{2}$$

$$(H_1 - H) + \sum_{i=2}^{k} F_i|_{\bar{x}_{[n]\setminus I} = \bar{b}_{[n]\setminus I}} - \sum_{j=1}^{m} G_j \equiv 0. \tag{3}$$

Therefore, we have a multilinear $\Sigma\Pi\Sigma\Pi(k+m)$ circuit computing the zero polynomial with $\lambda \leq 2^{(\log s)^{\nu(k-1,m)}}$. Let us consider the minimal component that contains $(H_1 - H)$. By Lemma 4.5, for $i \geq 2$:

$$\Delta\left(H_1, F_i|_{\bar{x}_{[n]\setminus I} = \bar{b}_{[n]\setminus I}}\right) \geq r^{1/\log s}$$

In addition, by Lemma 3.10:

$$\Delta\left(H_1, F_i|_{\bar{x}_{[n]\setminus I} = \bar{b}_{[n]\setminus I}}\right) \leq \Delta(H_1, H_1 - H) \cdot \Delta\left(H_1 - H, F_i|_{\bar{x}_{[n]\setminus I} = \bar{b}_{[n]\setminus I}}\right)$$

and hence

$$\Delta\left(H_1 - H, F_i|_{\bar{x}_{[n]\setminus I} = \bar{b}_{[n]\setminus I}}\right) \geq r^{1/\log s}/8 \geq \left(2^{(\log s)^{2^{2^k}} - 1}\right)^{1/8\log s} \geq 2^{(\log s)^{2^{2^k} - 1}}.$$

On the other hand,

$$\lambda(C') \leq 2^{(\log s)^{\nu(k-1,m)}} \leq 2^{(\log s)^{\nu(k-1,1)}} \leq 2^{(\log s)^{2^{2^k} - 4}}.$$

Therefore, by Corollary 4.8, this component cannot contain any such $F_i|_{\bar{x}_{[n]\setminus I} = \bar{b}_{[n]\setminus I}}$. Consequently, it must be the case that

$$H_1 - H - \sum_{j \in A} G_j \equiv 0$$

for some non-empty $A \subseteq [m]$. Subtracting from Equation 3 we obtain:

$$\sum_{i=2}^{k} F_i|_{\bar{x}_{[n]\setminus I}=\bar{b}_{[n]\setminus I}} = C_{\bar{b}} - H_1 = \sum_{j\in[m]\setminus A} G_j.$$

In other words, the circuit that corresponds to $H' = H_1$ has fan-in m' that is strictly smaller than m. Therefore, the learning algorithm should have picked $H' = H_1$, which leads to a contradiction. $\square$

By applying the lemma iteratively we can extend the algorithm to assignments with arbitrary Hamming distance, yet under some technical conditions. This can be considered as a grass-hopper jump.

Corollary 5.3. *Let $k, n, s \in \mathbb{N}$ and let $r \triangleq 2^{(\log s)^{2^{2^k}}-1}$. Then given:*

1. *$k, n, s \in \mathbb{N}$*
2. *Black-box access to a $\Sigma\Pi\Sigma\Pi(k)$ circuit $C = \sum\limits_{i=1}^{k} F_i$ satisfying Lemma 4.5*
3. *A reconstruction algorithm $\mathcal{A}$ that given a black-box access to a $\Sigma\Pi\Sigma\Pi(m)$ circuit C with $m \leq k-1$ and $\lambda(C) \leq s$, outputs a $\Sigma\Pi\Sigma\Pi(m')$ circuit C' with $m' \leq m$ and $\lambda(C') \leq 2^{(\log s)^{\nu(m,m')}}$.*
4. *Assignments: $\bar{a}, \bar{b} \in \mathbb{F}^n$ such that for all $0 \leq i \leq n$, $\Phi_C(\gamma^i(\bar{a}, \bar{b})) \neq 0$.*
5. *The subset $I \subseteq [n]$ guaranteed by Lemma 4.5*
6. *Polynomial: $H_0 = u_0 \cdot \prod_{i\in I}(x_i - u_i)$, where $H_0 = F_1|_{\bar{x}_{[n]\setminus I}=\bar{a}_{[n]\setminus I}}$*

runs in time $|\mathbb{F}|^{(\log s)^{\nu(k,1)}}$ outputs $H = F_1|_{\bar{x}_{[n]\setminus I}=\bar{b}_{[n]\setminus I}}$.

Proof. Starting with $\bar{a} = \gamma^0(\bar{a}, \bar{b})$, apply Lemma 2 iteratively using the value on $\gamma^i(\bar{a}, \bar{b})$ to compute the value on $\gamma^{i+1}(\bar{a}, \bar{b})$ until $\gamma^n(\bar{a}, \bar{b}) = \bar{b}$. $\square$

Now we show how to compute $F_1(\bar{b})$ without assuming $\Phi(\bar{b}) \neq 0$. In order to do this, we will consider the line $\ell_{\bar{a},\bar{b}}(t)$ through $\bar{a}$ and $\bar{b}$ and show that most points $\bar{v}$ on the line do satisfy the condition that $\Phi(\bar{v}) \neq 0$. Once we have this, by Corollary 5.3, we will show that for most points $\bar{v}$ on the line, $F_1(\bar{v})$ can be computed accurately. We then apply noisy polynomial interpolation (for instance the Berlekamp-Welch algorithm for decoding Reed-Solomon Codes) to recover the entire univariate polynomial which is F_1 restricted to $\ell_{\bar{a},\bar{b}}(t)$, and from this we can recover $F_1(\bar{b})$.

Lemma 5.4. *Let $k, n, s \in \mathbb{N}$ and let $r \triangleq 2^{(\log s)^{2^{2^k}}-1}$. Then given:*

1. *$k, n, s \in \mathbb{N}$*
2. *Black-box access to a $\Sigma\Pi\Sigma\Pi(k)$ circuit $C = \sum\limits_{i=1}^{k} F_i$ satisfying Lemma 4.5*
3. *A reconstruction algorithm $\mathcal{A}$ that given a black-box access to a $\Sigma\Pi\Sigma\Pi(m)$ circuit C with $m \leq k-1$ and $\lambda(C) \leq s$, outputs a $\Sigma\Pi\Sigma\Pi(m')$ circuit C' with $m' \leq m$ and $\lambda(C') \leq 2^{(\log s)^{\nu(m,m')}}$.*

```
Input: Reconstruction Algorithm $\mathcal{A}$ for reconstructing multilinear $\Sigma\Pi\Sigma\Pi(k)$ circuits
$k, n, s \in \mathbb{N}$
Assignments: $\bar{a}, \bar{b} \in \mathbb{F}^n$
Subset: $I \subseteq [n]$
Polynomial: $H_0 = u_0 \cdot \prod_{i \in I}(x_i - u_i)$
Black-box access to a $\Sigma\Pi\Sigma\Pi(k)$ circuit $C$
Output: $h_1 \in \mathbb{F}$

1 Choose a subset $V \subseteq \mathbb{F}$ be of size $|V| = 20kn^3$;
2 foreach $t' \in V$ do
3     Let $H_{t'}$ be the output of the algorithm in Corollary 5.3 for the input point $\ell_{\bar{a},\bar{b}}(t')$ ;
4     $h_{t'} \leftarrow H_{t'}(\bar{b})$ ;
      /* If $\ell_{\bar{a},\bar{b}}(t')$ satisfies the technical conditions of Corollary 5.3, then
         $h_{t'} = F_1(\ell_{\bar{a},\bar{b}}(t'))$; otherwise, $h_{t'}$ can be arbitrary.    */
5 Use noisy polynomial interpolation to recover $\hat{P}(t)$ (Lemma 3.2);
6 return $\hat{P}(1)$ ;
```

Algorithm 3: Evaluate.

4. *Assignments: $\bar{a}, \bar{b} \in \mathbb{F}^n$ such that $\Phi_C(\bar{a}) \neq 0$*

5. *The subset $I \subseteq [n]$ guaranteed by Lemma 4.5*

6. *Polynomial: $H_0 = u_0 \cdot \prod_{i \in I}(x_i - u_i)$, where $H_0 = F_1|_{\bar{x}_{[n]\setminus I} = \bar{a}_{[n]\setminus I}}$*

Algorithm 3 runs in time $|\mathbb{F}|^{(\log s)^{\nu(k,1)}}$ outputs $h_1 = F_1(\bar{b})$.

Proof. Consider the following polynomials:

$$P(t) \triangleq F_1(\ell_{\bar{a},\bar{b}}(t)) \ , \ Q(t) \triangleq \prod_{i=1}^{n} \Phi_C\left(\gamma^i(\bar{a}, \ell_{\bar{a},\bar{b}}(t))\right) .$$

By Corollary 5.3, if $Q(t') \neq 0$ then $h_{t'} = P(t')$. We will now bound the number of roots of $Q(t)$. By Observation 4.4, $Q(t)$ is a univariate polynomial of degree at most $6kn^3$. In addition, observe that $Q(0) = (\Phi_C(\bar{a}))^n \neq 0$. Consequently, $Q(t)$ has at most $6kn^3$ roots. While $P(t)$ is is a univariate polynomial of degree at most n. By Lemma 3.2, $\hat{P}(t) \equiv P(t)$. In particular, $\hat{P}(1) = P(1) = F_1(\bar{b})$. □

6 Open Questions

In this work we present the first efficient reconstruction algorithm for multilinear $\Sigma\Pi\Sigma\Pi(k)$ circuits for arbitrary, yet fixed top fan-in k, thus solving an open problem posed in [GKL12]. The algorithms is deterministic and runs in time quasi-poly$(n, s, |\mathbb{F}|)$. We conclude by discussing some open problems.

- **Improve dependence on the field size:** Perhaps the most immediate and natural question is if we can have a learning algorithm for "larger" fields. In particular, reconstructing $\Sigma\Pi\Sigma\Pi(k)$ circuits over fields of characteristic 0? Or over finite fields with $\text{poly}(\log(|\mathbb{F}|))$ dependence in time complexity? Notice that for our algorithm the $\text{poly}(|\mathbb{F}|)$ dependence essentially stems from the brute force step which is central to our approach. Thus, we have to avoid that using some new techniques. Also, (as was mentioned before) any proper learning algorithm that has $\text{poly}(\log(|\mathbb{F}|))$ dependence on the field size must be randomized, unless we can compute square roots deterministically [Vol16].

- **Handling larger (super-constant) top fan-in:** It would be interesting to generalize our results to multilinear $\Sigma\Pi\Sigma\Pi(k)$ circuits, with $k = \Omega(1)$. Notice that, any learning algorithm with polynomial dependence on k will directly yield a subexponential-time reconstruction algorithm for general multilinear circuits via the depth reduction results of [AV08, Koi10, Tav13, KdOS19]. This would be enormously interesting and very likely a difficult problem. Also, on a technical note, the structural result (Sparsity Bound [SV18]) which is central to the analysis of our algorithm will become will become trivial when $k = \Omega(\sqrt{n/\log s})$.

  Another interesting question is improving the running time of our algorithm. One technical difficulty to overcome is projecting the circuit to constantly many variables and still maintaining the structural properties of the circuit.

Acknowledgements

Part of the research was done when the first author was interning at Microsoft Research, India. The third author would like to thank Alex Samorodnitsky and Rafael Oliveira for many useful conversations.

References

[Agr05] M. Agrawal. Proving lower bounds via pseudo-random generators. In *Proceedings of the 25th FSTTCS*, volume 3821 of *LNCS*, pages 92–105, 2005.

[AM94] L. M. Adleman and K. S. McCurley. Open problems in number theoretic complexity, ii. In *International Algorithmic Number Theory Symposium*, pages 291–322. Springer, 1994.

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

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

[BB98] D. Bshouty and N. H. Bshouty. On interpolating arithmetic read-once formulas with exponentiation. *JCSS*, 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. *J. ACM*, 47(3):506–530, 2000.

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

[Bur57] D. A. Burgess. The distribution of quadratic residues and non-residues. *Mathematika*, 4(2):106–112, 1957.

[CIKK16] M. L. Carmosino, R. Impagliazzo, V. Kabanets, and A. Kolokolova. Learning algorithms from natural proofs. In *Proceedings of the 31st Conference on Computational Complexity, CCC*, pages 1–24, 2016.

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

[DS07] Z. Dvir and A. Shpilka. Locally decodable codes with 2 queries and polynomial identity testing for depth 3 circuits. *SIAM J. on Computing*, 36(5):1404–1434, 2007.

[FK09] L. Fortnow and A. R. Klivans. Efficient learning algorithms yield circuit lower bounds. *J. Comput. Syst. Sci.*, 75(1):27–36, 2009.

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

[GKL11] A. Gupta, N. Kayal, and S. V. Lokam. Efficient reconstruction of random multilinear formulas. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS*, pages 778–787, 2011.

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

[GKQ14] A. Gupta, N. Kayal, and Y. Qiao. Random arithmetic formulas can be reconstructed efficiently. *Computational Complexity*, 23(2):207–303, 2014.

[Hås90] Johan Håstad. Tensor rank is np-complete. *J. Algorithms*, 11(4):644–654, 1990.

[HH11] R. C. Harkins and J. M. Hitchcock. Exact learning algorithms, betting games, and circuit lower bounds. In *Proceedings of the 38th ICALP*, pages 416–423, 2011.

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

[KdOS19] M. Kumar, R. de Oliveira, and R. Saptharishi. Towards optimal depth reductions for syntactically multilinear circuits. *CoRR*, abs/1902.07063, 2019.

[KI03] V. Kabanets and R. Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC)*, pages 355–364, 2003.

[KMSV13] Z. S. Karnin, P. Mukhopadhyay, A. Shpilka, and I. Volkovich. Deterministic identity testing of depth 4 multilinear circuits with bounded top fan-in. *SIAM J. on Computing*, 42(6):2114–2131, 2013.

[KNS18] N. Kayal, V. Nair, and C. Saha. Average-case linear matrix factorization and reconstruction of low width algebraic branching programs. *Electronic Colloquium on Computational Complexity (ECCC)*, 25:29, 2018.

[KNST17] N. Kayal, V. Nair, C. Saha, and S. Tavenas. Reconstruction of full rank algebraic branching programs. In *32nd Computational Complexity Conference, CCC 2017.*, pages 21:1–21:61, 2017.

[Koi10] P. Koiran. Arithmetic circuits: the chasm at depth four gets wider. *CoRR*, abs/1006.4700, 2010.

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

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

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

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

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

[KS09c] A. R. Klivans and A. A. Sherstov. Cryptographic hardness for learning intersections of halfspaces. *J. Comput. Syst. Sci.*, 75(1):2–12, 2009.

[KS09d] A. R. Klivans and A. A. Sherstov. Cryptographic hardness for learning intersections of halfspaces. *J. Comput. Syst. Sci.*, 75(1):2–12, 2009.

[KS19] N. Kayal and C. Saha. Reconstruction of non-degenerate homogeneous depth three circuits. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019.*, pages 413–424, 2019.

[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.

[MV18] D. Minahan and I. Volkovich. Complete derandomization of identity testing and reconstruction of read-once formulas. *TOCT*, 10(3):10:1–10:11, 2018.

[RS03] R. Raz and A. Shpilka. Lower bounds for matrix product, in bounded depth circuits with arbitrary gates. *SIAM J. on Computing*, 32(2):488–513, 2003.

[RS04] R. Raz and A. Shpilka. In *19th Annual IEEE Conference on Computational Complexity*, pages 215–222, 2004.

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

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

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

[SS09] N. Saxena and C. Seshadhri. An almost optimal rank bound for depth-3 identities. In *Proceedings of the 24th Annual IEEE Conference on Computational Complexity (CCC)*, pages 137–148, 2009.

[SS10] N. Saxena and C. Seshadhri. From Sylvester-Gallai Configurations to Rank Bounds: Improved Black-Box Identity Test for Deph-3 Circuits. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 21–30, 2010.

[ST17] R. Saptharishi and A. Tengse. Quasi-polynomial hitting sets for circuits with restricted parse trees. *CoRR*, abs/1709.03068, 2017.

[STV01] M. Sudan, L. Trevisan, and S. P. Vadhan. Pseudorandom generators without the XOR lemma. *J. Comput. Syst. Sci.*, 62(2):236–266, 2001.

[Sud98] M. Sudan. Algebra and computation. http://people.csail.mit.edu/madhu/FT98/course.html, 1998. Lecture notes.

[SV10] A. Shpilka and I. Volkovich. On the relation between polynomial identity testing and finding variable disjoint factors. In *Automata, Languages and Programming, 37th International Colloquium (ICALP)*, pages 408–419, 2010. Full version at https://eccc.weizmann.ac.il/report/2010/036.

[SV14] A. Shpilka and I. Volkovich. On reconstruction and testing of read-once formulas. *Theory of Computing*, 10:465–514, 2014.

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

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

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

[Vol16] I. Volkovich. A guide to learning arithmetic circuits. In *Proceedings of the 29th Conference on Learning Theory, (COLT)*, pages 1540–1561, 2016.

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

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