Electronic Colloquium on Computational Complexity, Report No. 153 (2011)

Reconstruction of Depth-4 Multilinear Circuits with Top Fan-in 2

Ankit Gupta* Neeraj Kayal* Satya Lokam*

Abstract

We present a randomized algorithm for reconstructing multilinear $\Sigma\Pi\Sigma\Pi(2)$ circuits, i.e. multilinear depth-4 circuits with fan-in 2 at the top + gate. The algorithm is given blackbox access to a polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ computable by a multilinear $\Sigma\Pi\Sigma\Pi(2)$ circuit of size s and outputs an equivalent multilinear $\Sigma\Pi\Sigma\Pi(2)$ circuit, runs in time poly(n, s), and works over *any* field $\mathbb{F}$.

This is the first reconstruction result for any model of depth-4 arithmetic circuits. Prior to our work, reconstruction results for bounded depth circuits were known only for depth-2 arithmetic circuits (Klivans & Spielman, STOC 2001), $\Sigma\Pi\Sigma(2)$ circuits (depth-3 arithmetic circuits with top fan-in 2) (Shpilka, STOC 2007), and $\Sigma\Pi\Sigma(k)$ with $k = O(1)$ (Karnin & Shpilka, CCC 2009). Moreover, the running times of these algorithms have a polynomial dependence on $|\mathbb{F}|$ and hence do not work for infinite fields such as $\mathbb{Q}$.

Our techniques are quite different from the previous ones for depth-3 reconstruction and rely on a polynomial operator introduced by Karnin et al. (STOC 2010) and Saraf & Volkovich (STOC 2011) for devising blackbox identity tests for multilinear $\Sigma\Pi\Sigma\Pi(k)$ circuits. Some other ingredients of our algorithm include the classical multivariate blackbox factoring algorithm by Kaltofen & Trager (FOCS 1988) and an *average-case* algorithm for reconstructing $\Sigma\Pi\Sigma(2)$ circuits by Kayal.

1 Introduction

A reconstruction algorithm for a multivariate polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ is given blackbox access to f and must output a (succinct) representation of f. The algorithm can make adaptive queries to the blackbox to evaluate f on inputs of its choice. The running time, and hence number of queries, of the algorithm is required to be polynomial in the size of the representation of f produced by the algorithm. The simplest representation of a polynomial is as a sum of monomials, i.e., as a $\Sigma\Pi$ (depth-2) circuit[1]. The reconstruction problem in this case is referred to as the *interpolation problem* and admits efficient algorithms (cf. [KS01]). Many interesting polynomials, e.g., determinant, however, have exponentially long representations as sums of monomials, whereas as arithmetic formulas or circuits, they can be represented with quasipolynomial, or even polynomial, complexity. In its strongest formulation, the reconstruction problem may demand the output to be (roughly) the smallest arithmetic circuit computing the polynomial f hidden by the blackbox. In such generality, the reconstruction problem is extremely hard. In fact, a polynomial time *deterministic* reconstruction algorithm for a circuit class $\mathcal{C}$ is easily seen to imply a poly-time deterministic blackbox Polynomial Identity Testing (PIT) algorithm for $\mathcal{C}$. Recall blackbox PIT problem for $\mathcal{C}$: determine if a polynomial f computed by a circuit from $\mathcal{C}$ is the identically zero polynomial by making blackbox queries to f. Efficient algorithms for blackbox PIT for a class $\mathcal{C}$ are in turn known [HS80, Agr05] to imply superpolynomial lower bounds for circuits in $\mathcal{C}$ computing an explicit polynomial. This last problem remains a formidable challenge in general models

*Microsoft Research India, {t-ankitg, neeraka, satya}@microsoft.com.

[1]Another simple representation of a polynomial is a as a product of linear forms, i.e. as a $\Pi\Sigma$ (depth-2) circuit. Most interesting polynomials, e.g. determinant, cannot be computed by $\Pi\Sigma$ circuits but in case a polynomial f admits such a representation, it is unique (by uniqueness of polynomial factorization) and can be efficiently found using Kaltofen's factoring algorithm [Kal89, KT90].

ISSN 1433-8092

of arithmetic complexity. We conclude that deterministic reconstruction is at least as hard as proving superpolynomial lower bounds for the corresponding model of arithmetic complexity. Thus progress on the reconstruction problem has been possible only in restricted models of computation such as constant depth circuits.

There's an obvious similarity between the reconstruction problem in arithmetic complexity and the learning problem in Boolean complexity[2]. With this analogy in mind, it seems reasonable to allow *randomized* algorithms for reconstruction. Even when we allow randomization, the reconstruction problem for general models is still quite challenging. For instance, we know randomized algorithms for PIT for any efficiently computable polynomial. But we still do not have reconstruction algorithms for many nontrivial models of computation even with randomization. Another example is the class of depth-3 set-multilinear circuits for which a deterministic PIT algorithm is known [RS05] but the reconstruction problem (finding the smallest set-multilinear circuit for a given set-multilinear polynomial) is known to be NP-hard [Hås90]. These differences indicate that the reconstruction problem may be inherently harder than the PIT problem for a given model of computation.

Efficient reconstruction algorithms are previously known for depth-2 $\Sigma\Pi$ circuits (sparse polynomials) [KS01], *read-once* arithmetic formulas [SV09], *set-multilinear* depth-3 circuits [BBB+00, KS06] [3], non-commutative *arithmetic branching programs* [AMS10], $\Sigma\Pi\Sigma(2)$ circuits, i.e., depth-3 circuits with top + gate of fan-in 2, [Shp09], and $\Sigma\Pi\Sigma(k)$ circuits with $k = O(1)$ [KS09]. For more information on circuit reconstruction, we refer the reader to the excellent survey by Shpilka & Yehudayoff [SY10]. We remark that reconstruction of even constant depth circuits is a highly nontrivial task. For example, a reconstruction algorithm producing optimal size depth-3 *multilinear*[4] circuits can be used to produce the smallest circuit for computing the product of two 3×3 matrices, which may help improve the current best running time of matrix multiplication using Strassen's approach. On the other hand, the result by Håstad [Hås90] mentioned earlier shows that reconstructing optimal size depth-3 *set-multilinear* circuits is already NP-hard. Thus, to be tractable, the reconstruction problem even for constant depth multilinear circuits must impose additional restrictions such as on top fan-in and/or allow the output circuit to be polynomially larger than optimal.

In this work, we study the model of depth-4 multilinear $\Sigma\Pi\Sigma\Pi$ circuits. The importance of this model stems from a surprising result by Agrawal & Vinay [AV08] (see also [Raz10]) who showed that exponential lower bounds for depth-4 circuits are already as hard as proving similar lower bounds for arbitrary depth circuits, i.e., fundamental barriers to proving lower bounds start to appear even at depth-4. Using [AV08], it can also be shown that derandomizing blackbox identity testing of *multilinear* depth-4 circuits implies an exponential lower bound for *general*, i.e., with no depth restriction, *multilinear* circuits. Currently the best known lower bound for multilinear circuits is $\Omega(n^{4/3}/\log^2 n)$, due to Raz et al. [RSY08], and $2^{n^{\Omega(1/d)}}$ for depth-d multilinear circuits due to Raz & Yehudayoff [RY09][5]. Hence, understanding the model of multilinear depth-4 circuits will shed light on the bigger problem of proving lower bounds for multilinear circuits. Recently, Karnin et al. [KMSV10] and Saraf & Volkovich [SV11] made progress in this direction by devising quasi-polynomial and polynomial (respectively) time deterministic blackbox identity tests for multilinear $\Sigma\Pi\Sigma\Pi(k)$ circuits for $k = O(1)$. We go one step further and consider the harder problem of reconstructing depth-4 multilinear circuits. In this paper, we make progress on this problem by devising a reconstruction algorithm for the model of multilinear $\Sigma\Pi\Sigma\Pi(2)$ circuits which runs in time polynomial in the size of the hidden circuit, over *any given field*.

[2]In particular, reconstruction with blackbox queries may be compared to learning with membership and equivalence queries (when we allow randomized reconstruction). On the other hand, for multivariate polynomials exact reconstruction and approximate reconstruction are equivalent (again, allowing randomized reconstruction).

[3]The output of [BBB+00, KS06] is not the hidden set multilinear $\Sigma\Pi\Sigma(2)$ circuit but an algebraic branching program of roughly the same size.

[4]A circuit is said to be *multilinear* if every gate computes a multilinear polynomial.

[5]For multilinear *formulas* superpolynomial lower bounds are known [Raz09] and these bounds can be improved to exponential when the formula is of constant depth [RY09].

Our techniques for depth-4 have a relatively simple specialization to depth-3: whereas we handle *sparse* polynomials in the depth-4 case, we only need to handle *linear* polynomials in the depth-3 case. In fact, our result for the depth-3 case (Section 3) may be viewed as a simpler realization of the program described in Section 1.2 that we use for both depth-3 and depth-4 results. In this way we get an alternative reconstruction algorithm for depth-3 *multilinear* circuits, improving on earlier results due to [Shp09] and [KS09] in the following respect. The previous algorithms run in time polynomial in the size of the field and hence can only be efficient when $\mathbb{F}$ is small. In contrast, our results work over any field, including infinite fields. Perhaps more importantly, the techniques in [Shp09, KS09] seem inherently tied to depth-3 circuits and, as we will explain later, difficult to generalize to depth-4. Our main result for depth-4 case is presented in Section 4.

1.1 Depth-4 multilinear $\Sigma\Pi\Sigma\Pi(2)$ circuits

A depth-4 $\Sigma\Pi\Sigma\Pi$ circuit C is a layered circuit containing 4 alternating layers of addition($+$) and product($\times$) gates and computes a polynomial of the form

$$f(x_1,\ldots,x_n) = \sum_{i=1}^{k} T_i = \sum_{i=1}^{k}\prod_{j=1}^{d_i} P_{ij}, \tag{1}$$

where k is the fan-in of the top Σ gate and d_i's are the fan-ins of the Π gates at the second layer. We define the *min-degree* of C to be $\deg_{\mathrm{m}}(C) := \min\{d_i \mid i \in [k]\}$. As P_{ij}'s are the polynomials in $\mathbb{F}[x_1,\ldots,x_n]$, computed at the Σ layer at the third level of the circuit, the number of monomials in any such polynomial is bounded by circuit size. The number of non-zero monomials in a polynomial is called its *sparsity* and a polynomial with sparsity at most s is said to be s-sparse (else s-dense). Hence if the size of C is s then all the P_{ij}'s are s-sparse. Define $\gcd(C) := \gcd(T_1,\ldots,T_k)$ and C is said to be *simple* if $\gcd(C) = 1$. Define the *simplification* of C to be $\mathrm{sim}(C) := C/\gcd(C)$. Since every s-sparse multilinear polynomial is a product of irreducible s-sparse multilinear polynomials on disjoint sets of variables, from the point of view of reconstruction, we can assume G_i's, P_i's and Q_i's to be irreducible. If C is as in equation (1) then clearly, $\forall i \in [k]\ \exists I_i \subsetneq [d_i]$ such that

$$\mathrm{sim}(C) = \sum_{i=1}^{k} T'_i = \sum_{i=1}^{k} T_i/\gcd(C) = \sum_{i=1}^{k}\prod_{j\in I_i} P_{ij}. \tag{2}$$

A circuit C is said to be *minimal* if, for all $\emptyset \subsetneq A \subsetneq [k]$, the corresponding subcircuit $C_A := \sum_{i\in A} T_i$ of C is non-zero. Let the sparsity of a polynomial f be denoted by $||f||$. For a circuit C, where $\mathrm{sim}(C)$ is as in equation (2), define the sparsity of C to be $||C|| := \max\{||T'_1||,\ldots,||T'_k||\}$. We are interested in the class of multilinear $\Sigma\Pi\Sigma\Pi(2)$ circuits which contains multilinear $\Sigma\Pi\Sigma\Pi$ circuits with top fan-in 2. Hence any multilinear $\Sigma\Pi\Sigma\Pi(2)$ circuit C of size s computes a multilinear polynomial of the form

$$f(x_1,\ldots,x_n) = \gcd(C)\cdot\mathrm{sim}(C) = G\cdot(T_1+T_2) = G_1\cdot\ldots G_r\cdot\left(\prod_{i=1}^{d_1} P_i + \prod_{i=1}^{d_2} Q_i\right), \tag{3}$$

where $\{G_i\}_{i\in[r]}$, $\{P_i\}_{i\in[d_1]}$ and $\{Q_i\}_{i\in[d_2]}$ are sets of variable-disjoint s-sparse multilinear polynomials. Here, $\gcd(T_1,T_2) = 1$ and $||C|| = \max\{||T_1||, ||T_2||\}$.

We are now ready to state our main theorem. In the formal statement below, we will assume that the function f given by the blackbox is computable by a $\Sigma\Pi\Sigma\Pi(2)$ circuit C satisfying certain *nondegeneracy conditions*, namely, $\|C\| > 4s^4$ and $\deg_{\mathrm{m}}(C) \geq 3$. In Section 4.4, we will see that if C does not satisfy these conditions, then there is an easy interpolation-based solution to produce a $\Sigma\Pi\Sigma\Pi(2)$ circuit for f that is only polynomially larger than C. We choose to state the formal theorem with the these mild technical assumptions on C since, in this case, our algorithm actually produces essentially *"the unique"*

C computing f. In other words, except under the degeneracy conditions, our reconstruction algorithm has the stronger guarantee of producing the optimal $\Sigma\Pi\Sigma\Pi(2)$ circuit computing f.

Theorem 1. *Let $f \in \mathbb{F}[x_1, \ldots, x_n]$ be the polynomial computed by a multilinear $\Sigma\Pi\Sigma\Pi(2)$ circuit C of size s with $\|C\| > 4s^4$ and $\deg_{\mathrm{m}}(C) \geq 3$. Then, there is a randomized algorithm which, given blackbox access to f and the parameters n and s, outputs C in time $\mathrm{poly}(n, s, c_{\max})$, where $c_{\max}$ is the maximum bit length of any coefficient appearing in f. When $|\mathbb{F}| < n^5$, the algorithm should be allowed to make queries to f from a suitable polynomial-sized extension[6] field of $\mathbb{F}$.*

1.2 Basic idea and approach

In this section we give an overview of our algorithm. Towards this end we need to introduce some terminology. A polynomial $f \in \mathbb{F}[X_n]$ is said to *depend* on a variable x if the *derivative* of f w.r.t x, denoted by $\frac{\partial f}{\partial x} := f|_{x=1} - f|_{x=0}$, is non-zero. The *variable-set* of f, denoted $\mathsf{var}(f)$ is defined to be the set $\{x \mid x \in X_n,\ f \text{ depends on } x\}$. Observe that the irreducible factors of a multilinear polynomial depend on disjoint sets of variables. We now define a binary operator $D_x : \mathbb{F}[X_n] \times \mathbb{F}[X_n] \mapsto \mathbb{F}[X_n]$ and a unary operator $\Delta_{xy} : \mathbb{F}[X_n] \mapsto \mathbb{F}[X_n]$. For polynomials $P, Q \in \mathbb{F}[X_n]$ and variables $x, y \in X_n$ define [7]

$$D_x(P,Q) := \begin{vmatrix} \frac{\partial P}{\partial x} & P|_{x=0} \\ \frac{\partial Q}{\partial x} & Q|_{x=0} \end{vmatrix} = \frac{\partial P}{\partial x} \cdot Q|_{x=0} - \frac{\partial Q}{\partial x} \cdot P|_{x=0}.$$

while

$$\Delta_{xy}(P) := D_y(\frac{\partial P}{\partial x}, P|_{x=0}) = (P|_{x=0,y=0} \cdot P|_{x=1,y=1}) - (P|_{x=1,y=0} \cdot P|_{x=0,y=1}).$$

These two operators have many nice properties and these are given in Section 2. With this small piece of terminology in hand we are now ready to give an overview of the algorithm.

Suppose we have blackbox access to the output polynomial f of a multilinear $\Sigma\Pi\Sigma\Pi(2)$ circuit C as in equation (3). By querying f at points of our choice, we want to recover C. How can we do this? We first give the basic idea by describing how the algorithm works for a *generic instance* of our problem (this notion is made precise below). There will however be a number of *degenerate/boundary cases*[8] which will be addressed later. We will say that a polynomial f admitting a representation of the form (3), is a generic instance of our problem if the following additional conditions are satisfied.

1. $G = 1$ and $d_1 \geq 3$ and $d_2 \geq 3$.
2. There exist variables $x, y, u, v \in (\mathsf{var}(T_1) \cap \mathsf{var}(T_2)) \subseteq \mathsf{var}(f)$ and factors P_1, P_2, Q_1, Q_2 such that:
   (a) $x \in \mathsf{var}(P_1) \cap \mathsf{var}(Q_1)$ while $y \in \mathsf{var}(P_1) \setminus (\mathsf{var}(Q_1) \cup \mathsf{var}(Q_2))$
   (b) $u \in \mathsf{var}(P_2) \cap \mathsf{var}(Q_2)$ while $v \in \mathsf{var}(Q_2) \setminus (\mathsf{var}(P_1) \cup \mathsf{var}(P_2))$
3. $\mathsf{var}(P_2) \cap \mathsf{var}(Q_1)$ is non-empty.

First note that the assumption $G = 1$ means that the input polynomial f admits a representation of the form

$$f(x_1, \ldots, x_n) = T_1 + T_2 = \left(\prod_{i=1}^{d_1} P_i\right) + \left(\prod_{i=1}^{d_2} Q_i\right).$$

[6] We can assume this w.l.o.g. since if $\mathbb{F}$ is small, then a large enough extension $\mathbb{F}'$ of $\mathbb{F}$ can be found using the deterministic $\mathrm{poly}(\log|\mathbb{F}'|)$-time method of Adleman & Lenstra [AL86].

[7] The D_x operator was defined and used in [KMSV10, SV11] to analyze multilinear $\Sigma\Pi\Sigma\Pi(2)$ circuits.

[8] Indeed a particularly important degenerate case, to be handled later, is when the unknown circuit is set-multilinear, i.e. when the two multiplication gates induce the same partition on the variable set.

Also note that given a candidate solution, i.e. given the P_i's and the Q_i's, we can verify the correctness of the solution in randomized polynomial time by applying the DeMillo-Lipton-Schwartz-Zippel identity testing algorithm to the above equation. Indeed, it suffices to determine the P_i's and Q_i's upto scalar multiples because given polynomials f, T_1 and T_2 we can determine the set of all scalars $\alpha_1, \alpha_2 \in \mathbb{F}$ such that[9]

$$f = \alpha_1 \cdot T_1 + \alpha_2 \cdot T_2.$$

We nondeterministically guess the 4-tuple of variables (x, y, u, v) satisfying the above properties[10]. So given f, how do we determine the P_i's and Q_i's (upto scalar multiples)? The idea is to look at the polynomial $\Delta_{xy}(f)$.

The key observation is that the polynomial P_2 is a factor of $\Delta_{xy}(f)$ [11]. We therefore compute $\Delta_{xy}(f)$ and factor it. We then nondeterministically guess the correct P_2 from among the list of irreducible factors of $\Delta_{xy}(f)$ (there are at most $2n$ of them). In a similar manner, using u, v, we obtain Q_1. We now a pick a variable, say $z \in \mathsf{var}(P_2) \cap \mathsf{var}(Q_1)$ and compute the polynomials $D_z(f, P_2)$ and $D_z(Q_1, P_2)$. It turns out that

$$D_z(f, P_2) = D_z(Q_1, P_2) \cdot (Q_2 \cdot Q_3 \cdot \ldots Q_{d_2})$$

and moreover that $D_z(Q_1, P_2)$ is nonzero. Thus by factoring $(D_z(f, P_2)) / (D_z(Q_1, P_2))$, we obtain $Q_2, Q_3, \ldots, Q_{d_2}$. In a similar manner we obtain $P_1, P_3, \ldots, P_{d_1}$ and therefore the complete circuit for f. This completes our brief description of the algorithm for a generic input. The actual algorithm, which must handle all the degenerate cases including the set-multilinear one, is somewhat lengthier and more involved. This unfortunately also hampers the readability of the present paper - we suggest that some of the more exotic special cases be skipped in afirst reading. We now give a more detailed *program* for reconstruction.

Multilinear $\Sigma\Pi\Sigma\Pi(2)$ Reconstruction Program:

1. **Obtain blackbox access to** $\mathrm{sim}(C)$: This step computes $G = \gcd(C)$ and reduce us to the case when C is simple. Note that every factor G_i of G is an s-sparse irreducible factor of f. We obtain G by showing that the converse holds under some mild technical conditions (Lemma 11).

2. **Determine a factor** R **of either** T_1 **or** T_2 : We show that, under some mild technical conditions, it is enough to determine such an R, i.e., given just such an s-sparse R, we can reconstruct C entirely. Note that, by simplicity, any such R is a factor of *exactly one of* T_1 *or* T_2. We consider the blackbox for $D_x(T_1 + T_2, R)$, where x is any variable with a non-zero coefficient in R, and consider its s-sparse multilinear irreducible factors. This would equal $D_x(T_2, R) = Q_2 \cdot \ldots Q_{d_2} \cdot D_x(Q_1, R)$ where, say, Q_1 depends on x. As Q_i's are s-sparse multilinear, we would have obtained a list of s-sparse polynomials containing these Q_i's. But unfortunately this list also has numerous "spurious" factors contributed by $D_x(Q_1, R)$. Momentarily assume that there exists a Q_i ($i \in [2..d_2]$) such that $\mathsf{var}(R) \cap \mathsf{var}(Q_i)$ is non-empty. In this situation we guess an appropriate Q_i from among the polynomials in this list, and find a variable $z \in \mathsf{var}(R) \cap \mathsf{var}(Q_i)$. By looking at the polynomials

$$D_z(T_1 + T_2, Q_i), D_z(T_1 + T_2, R) \text{ and } D_z(R, Q_i),$$

and proceeding as in the solution to the generic case described above, we can obtain all the P_j's and all the Q_j's. In general however there need not exist not exist any Q_i in the list for which

[9]This is done by solving for α_1 and α_2 in the pair of linear equations obtained by making independent random substitutions of the variables - cf. [Kay11].

[10]Our algorithm implements the nondeterministic guessing of an appropriate element of a list by iterating through the list. We will make only a constant number of nondeterministic guesses and in all such cases, the list will be short enough (of polynomial size) and a correct guess can be verified fast enough (in polynomial time) so that we can implement all the nondeterministic guesses efficiently with only a polynomial overhead in the running time.

[11]This can be verified directly by expanding and simplifying $\Delta_{xy}(f)$.

$\mathsf{var}(R) \cap \mathsf{var}(Q_i)$ is non-empty. With some more work involving some careful analysis of such a situation, we show that all a complete solution can always be obtained given just the one factor R of T_1.

3. **Reduce to the case when T_1 and T_2 have the same variable sets** : If the variable sets of T_1 and T_2 are different, then there exists a variable x (which we nondeterministically guess) such that exactly one of T_1 or T_2 depends on x - say T_1 depends on x. Also x would occur in exactly one of the P_i's, say P_1, and hence the coefficient of x in f would have $P_2, \ldots, P_{d_1}$ as its factors (along with some spurious factors contributed by the coefficient of x in P_1). We nondeterministically guess a correct P_i as before.

4. **Reduce to the case when T_1, T_2 have the same partition** : We show that if there exists a pair of variables x, y (which we nondeterministically guess) such that x, y occur in the same factor T_1 but in different factors of T_2 (or vice versa), then we can determine a P_i (or a Q_i) by looking at the irreducible factors of $\Delta_{xy}(T_1 + T_2)$.

5. **Determine the partition of T_1, T_2** : We show that if the partition given by the factorization of $(T_1 + T_2)$ is different from the one given by T_1 (or T_2), then $\Delta_{xy}(T_1 + T_2)$ yields either a P_i or a Q_j for some pair (x, y) of variables. Thus factoring $(T_1 + T_2)$ and looking at the induced partition on the set of variables gives us the the partition of T_1, T_2.

6. **Reduce to the case of reconstructing set-multilinear $\Sigma\Pi\Sigma(2)$ circuits** : We are now reduced to the case when $T_1 + T_2 = P_1(\bar{x}_1) \cdot \ldots P_d(\bar{x}_d) + Q_1(\bar{x}_1) \cdot \ldots Q_d(\bar{x}_d)$ and we know $\bar{x}_i$'s. We consider each set of this partition at a time, say $\bar{x}_i$, and substitute the rest of the variables to random values over the field $\mathbb{F}$, twice, to obtain $2s$-sparse polynomials $R_i(\bar{x}_i)$ and $S_i(\bar{x}_i)$ such that, for some scalars $a_i, b_i, c_i, d_i \in \mathbb{F}$ such that $a_i d_i \neq b_i c_i$, we have $P_i = a_i R_i(\bar{x}_i) + b_i S_i(\bar{x}_i)$ and $Q_i = c_i R_i(\bar{x}_i) + d_i S_i(\bar{x}_i)$. We now have
$$T_1 + T_2 = \prod_{i \in [d]} (a_i R_i(\bar{x}_i) + b_i S_i(\bar{x}_i)) + \prod_{i \in [d]} (c_i R_i(\bar{x}_i) + d_i S_i(\bar{x}_i)),$$
where we know R_i's and S_i's explicitly and we just have to determine the above scalars. At this point, the problem can be reduced to a special case of the problem of reconstruction of depth three circuits with bounded top fanin. has been examined by Shpilka and Karnin [Shp09, KS09] and Kayal [Kay11]. Using ideas similar to these works we show that the above scalars can indeed be determined efficiently.

2 Preliminaries

Notation: $[n]$ denotes the set $\{1, 2, \ldots, n\}$ and X_n denotes the set of variables $\{x_1, \ldots, x_n\}$. For a polynomial f, the homogenous degree-d part of f is denoted by $f^{[d]}$. Tuples are indicated by placing a bar over a letter, e.g. $\bar{x}$. For a polynomial $f \in \mathbb{F}[X_n]$ and $\alpha \in \mathbb{F}$, $f_{x=\alpha}$ denotes f with the variable x substituted to α. A multilinear polynomial $f \in \mathbb{F}[X_n]$ is said to *depend* on a variable x if the *derivative* of f w.r.t x, denoted by $\frac{\partial f}{\partial x} := f|_{x=1} - f|_{x=0}$, is non-zero. Intuitively, $\frac{\partial f}{\partial x}$ is the coefficient polynomial of x in the sum of monomials representation of f. Let the *variable-set* of f be $\mathsf{var}(f) := \{x \mid x \in X_n,\ f \text{ depends on } x\}$. Observe that the irreducible factors of a multilinear polynomial depend on disjoint sets of variables. If $f_1 \cdot \ldots f_k$ is the factorization of a multilinear polynomial f, then $\mathsf{par}(f) := \{\mathsf{var}(f_i)\}_{i \in [k]}$ is defined to be the *partition* of f. Two polynomials $f, g \in \mathbb{F}[X_n]$ are said to be linearly dependent, abbreviated LD, and denoted by $f \sim g$, if $\exists \alpha, \beta \in \mathbb{F}$ such that $\alpha f = \beta g$; otherwise, they are linearly independent, abbreviated LI.

Blackbox PIT: The following well-known lemma immediately implies a randomized algorithm for Polynomial Identity Testing (PIT).

Lemma 1 (DeMillo-Lipton-Schwartz-Zippel [Sch80, Zip79, DL78])**.** *Let $f \in \mathbb{F}[x_1, \ldots, x_n]$ be a non-zero polynomial of degree $d \geq 0$. Let S be a finite subset of $\mathbb{F}$ and let $r_1, \ldots, r_n$ be selected randomly from S. Then* $\Pr[f(r_1, r_2, \ldots, r_n) = 0] \leq \frac{d}{|S|}$.

Kaltofen's Blackbox Factoring: We state the multivariate blackbox factoring algorithm by Kaltofen & Trager [Kal89, KT90] (we assume $|\mathbb{F}| > n^5$).

Lemma 2 (Kaltofen's Blackbox Factoring [Kal89, KT90])**.** *There is a randomized algorithm which, given blackbox access to a degree-d polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ and the parameters n and d, with probability $1 - 2^{-\Omega(n)}$, outputs blackboxes to all the irreducible factors of f (with their respective multiplicities) in time $K(n, d, c_{\max}) = \mathrm{poly}(n, d, c_{\max})$ where $c_{\max}$ is the maximum bit length of any coefficient appearing in f.*

Interpolation of Sparse Polynomials: There are many interpolation algorithms known for the class of $\Sigma\Pi$ circuits, which are essentially sparse polynomials (see [KS01] and references within).

Lemma 3 (Sparse Interpolation [KS01])**.** *Given blackbox access to a degree-d s-sparse polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$, we can determine the monomials of f with their coefficients in time $I(n, d, s, c_{\max}) = \mathrm{poly}(n, d, s, c_{\max})$ where $c_{\max}$ is the maximum bit length of any coefficient appearing in f. If $|\mathbb{F}| < (nd)^6$, we are allowed to make queries over an appropriate polynomial-sized extension of $\mathbb{F}$.*

Determining the coefficient of a given monomial: Although for general polynomials this problem is #P-complete (over $\mathbb{Q}$), it is easy for multilinear polynomials. For this we would need the following lemma from our previous work.

Lemma 4 ([GKL11])**.** *Let $\mathbb{F}$ be a field with at least $d + 1$ elements and $f \in \mathbb{F}[X_n]$ be a degree-d polynomial. Given blackbox access to f and r, we can simulate blackbox access to $f^{[r]}$ in time* $\mathrm{poly}(n, d)$.

Lemma 5 (Determining a coefficient)**.** *Let $\mathbb{F}$ be a field with at least $n + 1$ elements and $f \in \mathbb{F}[X_n]$ be a multilinear polynomial. Given blackbox access to f and a set $S \subseteq X_n$, we can determine the coefficient of M_S in f in time $\mathrm{poly}(n)$, where M_S denotes the monomial $\prod_{x \in S} x$.*

Proof. In time $\mathrm{poly}(n)$, using Lemma 4, obtain the blackbox for $f^{[|S|]}$. For all $x \in X_n$, substitute x to 1 if $x \in S$ and 0 otherwise. Clearly the value obtained is the coefficient of M_S as every other degree-$|S|$ monomial will have at least one variable from $X_n \backslash S$ and hence would vanish. ∎

2.1 The Operators D_x and Δ_{xy}

Recall the definitions of the operators D_x and Δ_{xy} from Section 1.2, repeated here for convenience.

Definition (The D_x Operator[12])**.** The binary operator $D_x : \mathbb{F}[X_n] \times \mathbb{F}[X_n] \mapsto \mathbb{F}[X_n]$ is defined as follows. For polynomials $P, Q \in \mathbb{F}[X_n]$ and a variable $x \in X_n$,

$$D_x(P, Q) := \begin{vmatrix} \frac{\partial P}{\partial x} & P|_{x=0} \\ \frac{\partial Q}{\partial x} & Q|_{x=0} \end{vmatrix} = \frac{\partial P}{\partial x} \cdot Q|_{x=0} - \frac{\partial Q}{\partial x} \cdot P|_{x=0}.$$

Definition (The Δ_{xy} Operator)**.** The unary operator $\Delta_{xy} : \mathbb{F}[X_n] \mapsto \mathbb{F}[X_n]$ is defined as follows. For a polynomial $P \in \mathbb{F}[X_n]$ and variables $x, y \in X_n$,

$$\Delta_{xy}(P) := D_y(\frac{\partial P}{\partial x}, P|_{x=0}) = (P|_{x=0,y=0} \cdot P|_{x=1,y=1}) - (P|_{x=1,y=0} \cdot P|_{x=0,y=1}).$$

[12]The operator D_x was defined and used in [KMSV10, SV11] to prove structural results about multilinear $\Sigma\Pi\Sigma\Pi(2)$ identities ultimately leading to a blackbox PIT algorithm for multilinear $\Sigma\Pi\Sigma\Pi(k)$ circuits with bounded top fanin.

It is easy to see that D_x is a bilinear operator and satisfies the following useful properties.

Lemma 6 (Properties of D_x operator, [SV11])**.** *Let $P, Q, R \in \mathbb{F}[X_n]$ be multilinear polynomials, $x \in X_n$ and $\alpha, \beta \in \mathbb{F}$. Then the following properties hold:*

1. $D_x(P + R, Q) = D_x(P, Q) + D_x(R, Q)$
2. *If* $x \notin \mathsf{var}(R)$ *then* $D_x(R \cdot P, Q) = R \cdot D_x(P, Q)$
3. $D_x(Q, P) = -D_x(P, Q)$
4. *If* $x \neq y$ *then* $D_x(P|_{y=\alpha}, Q|_{y=\alpha}) = D_x(P, Q)|_{y=\alpha}$
5. *If* P *is irreducible and* $x \in \mathsf{var}(P)$ *then* $D_x(Q, P) \equiv 0$ *iff* $P \mid Q$
6. *If* $x \in \mathsf{var}(P)$ *and* $Q \not\equiv 0$ *then* $D_x(Q, P) \equiv 0$ *iff* $x \in \mathsf{var}(\gcd(Q, P))$

Meanwhile, Δ_{xy} captures the irreducibility of a multilinear polynomial in the following manner.

Corollary 1. *For a multilinear polynomial $P \in \mathbb{F}[X_n]$ and a pair of variables $x, y \in \mathsf{var}(P)$, $\Delta_{xy}(P) \equiv 0$ if and only if the (unique) irreducible factor of P which depends on x is distinct from the irreducible factor which depends on y. In particular, P is reducible iff $\exists x \neq y \in \mathsf{var}(P)$ such that $\Delta_{xy}(P) \equiv 0$.*

In fact, the Δ_{xy} operator has some really neat properties given below (including the above corollary).

Proposition 1 (Properties of Δ_{xy} operator)**.** *Let P, Q, R be multilinear polynomials, x, y, z be variables and $\alpha, \beta \in \mathbb{F}$. Then the following properties hold.*

1. $\Delta_{xy}(P) = -\Delta_{yx}(P)$.
2. *If* $x, y \notin \mathsf{var}(R)$ *then* $\Delta_{xy}(R \cdot P) = R^2 \cdot \Delta_{xy}(P)$. *In particular* $\Delta_{xy}(\alpha \cdot P) = \alpha^2 \cdot \Delta_{xy}(P)$.
3. *If* $z \neq x, y$ *then* $\Delta_{xy}(P)|_{z=\alpha} = \Delta_{xy}(P|_{z=\alpha})$.
4. *If* $x \notin \mathsf{var}(P)$ *then* $\Delta_{xy}(P) = 0$.
5. *If* P *is irreducible and* $x, y \in \mathsf{var}(P)$ *then* $\Delta_{xy}(P) \neq 0$.
6. $\Delta_{xy}(P)$ *is nonzero if and only if* $x, y \in \mathsf{var}(P)$ *and occur in the same irreducible factor of* P. *In particular,* P *is irreducible if and only if* $\Delta_{xy}(P)$ *is nonzero for all* $x, y \in \mathsf{var}(P)$.
7. $$\begin{aligned} \Delta_{xy}(P + Q) &= \Delta_{xy}(P) + D_y(P|_{x=1}, Q|_{x=0}) - D_y(P|_{x=0}, Q|_{x=1}) + \Delta_{xy}(Q) \\ &= \Delta_{xy}(P) + D_x(P|_{y=1}, Q|_{y=0}) - D_x(P|_{y=0}, Q|_{y=1}) + \Delta_{xy}(Q) \end{aligned}$$
8. *If* $\Delta_{xy}(Q) = 0$ *and* $x, y \notin \mathsf{var}(R)$ *then* R *divides* $\Delta_{xy}(R \cdot P + Q)$.

These operators turn out to be very informative and useful in understanding multilinear circuits. We would be heavily using the above properties of the D_x and Δ_{xy} operators to devise our reconstruction algorithms for the class of multilinear $\Sigma\Pi\Sigma\Pi(2)$ circuits. Note that, by definition, the D_x operator is nothing but the *resultant* of two polynomials in $\mathbb{F}(X_n \setminus \{x\})[x]$ that are linear in x. As expected, it inherits all the properties of the resultant operator. Intuitively, the D_x operator can be interpreted as an indirect way of working modulo polynomials. Note that for any field $\mathbb{F}$, although $\mathbb{F}[X_n]$ is a unique factorization domain (UFD), the same is not true for its factor rings, for e.g. $\mathbb{F}[X_n]/(x_1x_2 - x_3x_4)$. As many of the techniques involved in the previous works on reconstruction algorithms (especially for $\Sigma\Pi\Sigma(k)$ circuits) involved working over factor rings modulo affine forms, directly extending these techniques to the depth-4 case seems difficult. While working modulo a set of polynomials, one usually has to fix an admissable monomial ordering and do computations using the reduced Gröbner basis of the ideal generated by these polynomials. For more details, see the book by Cox, Little, and O'Shea [CLO97].

3 Reconstructing Multilinear $\Sigma\Pi\Sigma(2)$ Circuits

In this section, we present a reconstruction algorithm for multilinear $\Sigma\Pi\Sigma(2)$ circuits. Our main theorem for depth-3 is

Theorem 2. *Let $f \in \mathbb{F}[x_1, \ldots, x_n]$ be the polynomial computed by a multilinear $\Sigma\Pi\Sigma(2)$ circuit C. There is a randomized algorithm* `SPSRECON` *which, given blackbox access to f and the parameter n, outputs C in time $\mathrm{poly}(n, c_{\max})$, where $c_{\max}$ is the maximum bit length of any coefficient appearing in f. When $|\mathbb{F}| < n^5$, the algorithm is allowed to make queries to f from a suitable polynomial-sized extension field*[13] *of $\mathbb{F}$.*

As already mentioned, our algorithm will be more or less based on the steps described earlier in Section 1.2.
Recall that the polynomial produced by such a circuit C has the form

$$f(X_n) = G \cdot (T_1 + T_2) = G_1 \cdot \ldots \cdot G_r \cdot \left(\prod_{i=1}^{d_1} L_i(S_i) + \prod_{i=1}^{d_2} M_i(S_i') \right), \tag{4}$$

where $\gcd(T_1, T_2) = 1$, G_i's, L_i's, and M_i's are linear polynomials, and $\mathsf{par}(T_1) = \{S_i\}_{i\in[d_1]}$ and $\mathsf{par}(T_2) = \{S_i'\}_{i\in[d_2]}$ are partitions of $\mathsf{var}(T_1 + T_2)$. We say C is *set-multilinear* if these partitions are same.

3.1 Obtaining blackbox access to $\mathrm{sim}(C)$

The following lemma (first proved in [Shp09]; reproved below for completeness) immediately results in Step 1.

Lemma 7. *Let $f \in \mathbb{F}[X_n]$ be the polynomial computed by a multilinear $\Sigma\Pi\Sigma(2)$ circuit C as given in equation* (4) *such that it is not a product of linear polynomials. Then for any linear polynomial ℓ we have $\ell \mid f \iff \ell \mid G$.*

Proof. Clearly if $\ell \mid G$ then $\ell \mid f$ as $G \mid f$. For the converse, suppose $\ell \nmid G$. Then we have that $\ell \mid T_1 + T_2$ or $T_1 \equiv T_2 \pmod{\ell}$. Now if $\ell \mid T_1$ then $\ell \mid T_2$ and hence $\ell \mid \gcd(T_1, T_2)$, a contradiction. Hence $\ell \nmid T_1, T_2$. Let $x \in \mathsf{var}(\ell)$ with α as its coefficient in ℓ. Let $\ell = \alpha x + \ell'$. Also w.l.o.g. let $x \in S_1, S_1'$. Hence,

$$T_1 \equiv T_2 \pmod{\ell} \iff L_1|_{x=-\ell'/\alpha} \prod_{i=2}^{d_1} L_i(S_i) = M_1|_{x=-\ell'/\alpha} \prod_{i=2}^{d_2} M_i(S_i').$$

As $\gcd(T_1, T_2) = 1$ the above can hold only if $d_1, d_2 < 3$. But if degree of $T_1 + T_2$ is at most 2, then it is of the form ℓm for some linear m and hence f is a product of linear polynomials, a contradiction. ∎

Hence, determining linear factors of f completely determines $\gcd(C)$ and from this it is easy to obtain blackbox access to $\mathrm{sim}(C) = f / \gcd(C)$[14].

[13] We can assume this w.l.o.g. since, if $\mathbb{F}$ is small, then a large enough extension $\mathbb{F}'$ of $\mathbb{F}$ can be found using the deterministic $\mathrm{poly}(\log |\mathbb{F}'|)$-time method of Adleman & Lenstra [AL86].

[14] Since variable sets of $\mathrm{sim}(C), \gcd(C)$ are disjoint, the value of $\mathrm{sim}(C)$ for any assignment to its variable set can be determined by simply choosing a random assignment for the variables in $X_n \backslash \mathsf{var}(\mathrm{sim}(C))$. This is easy to do and we omit the details.

3.2 Reconstructing when a factor of one of the product gates is known

In the following lemma, we prove the claim we made in Step 2 of our program, which essentially says that, for the purpose of reconstruction of a simple circuit C, it is enough to determine a factor of any one of the multiplication gates T_1, T_2.

Lemma 8. *Let $f \in \mathbb{F}[X_n]$ be the polynomial computed by the following* simple *multilinear $\Sigma\Pi\Sigma(2)$ circuit C, where L_i's and M_i's are linear functions and $\{S_i\}_{i\in[d_1]}$ and $\{S'_i\}_{i\in[d_2]}$ are partitions of X_n,*

$$f(X_n) = T_1 + T_2 = \prod_{i=1}^{d_1} L_i(S_i) + \prod_{i=1}^{d_2} M_i(S'_i). \tag{5}$$

Given blackbox access to f and the linear function L_1 explicitly, algorithm `RECONFACTOR` *described below outputs the L_i's and M_i's in time* $\mathrm{poly}(n, c_{\max})$*, where $c_{\max}$ is the maximum bit length of any coefficient in f.*

Algorithm : $\texttt{RECONFACTOR}(n, \mathcal{O}_f, L_1)$.
Input: oracle $\mathcal{O}_f$ for the polynomial $f \in \mathbb{F}[X_n]$ computable by a simple multilinear $\Sigma\Pi\Sigma(2)$ circuit C as given in equation (5) and a linear factor L_1 of T_1.
Output: sets of linear functions $\{L_1, \dots, L_{d_1}\}, \{M_1, \dots, M_{d_2}\}$ s.t. equation (5) holds or `FAIL`.

---

1. *Product of linear functions case* : Using Algorithm 2, obtain blackboxes for the factors of f. For every factor g determine if it is linear as follows. For any $x_i \in X_n$, $g_i = g|_{x_i=1} - g|_{x_i=0}$ is the coefficient polynomial of x_i in g. For all g_i's, using blackbox-PIT on $g_i|_{x_j=1} - g_i|_{x_j=0}$, determine if g_i depends on some x_j. If all non-zero g_i's are independent of X_n, then g is linear. If any factor is linear, by Lemma 7 all the factors are linear, and so simply interpolate them and output a multilinear $\Pi\Sigma$ circuit for f.

2. $\mathsf{var}(L_1) \nsubseteq \mathsf{var}(T_2)$ *case* : By iterating over $x \in \mathsf{var}(L_1)$ guess a variable x (if it exists) such that $x \notin \mathsf{var}(T_2)$. Let $L_1 = \alpha x + L'_1$ and obtain the blackbox for $f|_{x=-L'_1/\alpha}$. Obtain its linear factors $M'_1, \dots, M'_{d_2}$ as described above and check if they are on disjoint sets of variables. If not, proceed to next x, else obtain the blackbox for $f - f|_{x=-L'_1/\alpha}$, factorize it, and check the linearity of its factors. If all the factors $L'_1, \dots, L'_{d_1}$ are linear then simply output the sets $\{M'_1, \dots, M'_{d_2}\}$ and $\{L'_1, \dots, L'_{d_1}\}$.

3. Pick any $x \in \mathsf{var}(L_1)$ and let $L_1 = \alpha x + L'_1$. Obtain the blackbox for $f|_{x=-L'_1/\alpha}$, factorize it and obtain its factors $M'_1, \dots, M'_{d_2}$ (if any factor is non-linear output `FAIL`). If $d_2 \le 2$ proceed to Step 5 that handles the low degree cases. Since exactly one of the M_i's depends on x, exactly one of the obtained M'_i 's is "corrupted".

4. Guess this corrupted factor by iteration, i.e, for every $i \in [d_2]$ do:
   (a) Assume that $\forall j \in [d_2] \setminus \{i\}$, $M_j = M'_j$ and pick such a factor M_j of T_2.
   (b) Repeat all the above steps for M_j as a factor of T_2 to get a list of all the correct L_i's except one. By iterating over this list of L_i's as above guess the corrupted factor.
   (c) Hence w.l.o.g. assuming that we correctly know $L_1, \dots, L_{d_1-1}$ and $M_1, \dots, M_{d_2-1}$, determine L_{d_1} and M_{d_2} as follows.
      i. If $\mathsf{var}(L_i) \cap \mathsf{var}(M_j) \ne \emptyset$ for some i, j, $1 \le i \le d_1-1$ and $1 \le j \le d_2-1$. Let $M_i = \alpha x + M'_i$ then we can obtain the blackbox for $f|_{x=-M'_i/\alpha}$ and its linear factors. One of these factors would be $L_j|_{x=-M'_i/\alpha}$ (after easily taking care of a scalar multiple) and the rest would be $L_1, \dots, L_{j-1}, L_{j+1}, \dots, L_{d_1}$. Hence we know L_i's, from which we can determine M_i's by

factoring $f-\prod_i L_i$. Using blackbox-PIT check if these linear functions form a multilinear $\Sigma\Pi\Sigma(2)$ circuit satisfying equation (5). If they do, output $\{M_1,\ldots,M_{d_2}\}$, $\{L_1,\ldots,L_{d_1}\}$, else proceed.

ii. So, we have $\mathsf{var}(L_i)\cap\mathsf{var}(M_j)=\emptyset$ for all i,j, $1\le i\le d_1-1$ and $1\le j\le d_2-1$. By factoring $f|_{M_1=0}$ and $f|_{M_2=0}$ (and knowing $L_1,\ldots,L_{d_1-1}$), exactly determine $L_{d_1}(\text{mod } M_1)$ and $L_{d_1}(\text{mod } M_2)$.Let $\phi:\mathbb{F}^{n+1}\mapsto\mathbb{F}^{n+1}$ be an invertible linear transformation[15] such that $\phi(M_1)=y_1$ and $\phi(M_2)=y_2$ (as M_1,M_2 are LI). Hence, from above, we exactly know $\phi(L_{d_1})(\text{mod } y_1)$ and $\phi(L_{d_1})(\text{mod } y_2)$ from which we can easily determine $\phi(L_{d_1})$, and then L_{d_1} using ϕ^{-1}. Again check if the obtained candidate polynomials form a multilinear $\Sigma\Pi\Sigma(2)$ circuit satisfying equation (5); else proceed to next $i\in[d_2]$ if it exists or `FAIL`.

5. *Low-degree cases:*

(a) Case $d_1\ge 3, d_2\le 1$. Pick any $x\in\mathsf{var}(L_1)$ and guess any $y\in\mathsf{var}(L_2)$ (also w.l.o.g. let the coefficient of y in L_2 be 1). Factorize the coefficient polynomial of xy in f (scaled by the inverse of the coefficient of x in L_1) to get the factors $L_3,\ldots,L_{d_1}$. Pick any $z\in\mathsf{var}(L_3)$. Factorize the coefficient polynomial of xz in f (scaled by the inverse of the coefficient of z in L_3) to get L_2. Having determined all the L_i's we can determine M_1 from $f-\prod_i L_i$.

(b) Case $d_1\le 1, d_2\ge 3$. As done above, we can pick any $x\in\mathsf{var}(L_1)$ with $L_1=\alpha x+L_1'$, obtain the blackbox for $f|_{x=-L_1'/\alpha}$, factorize it and obtain its linear factors $M_1',\ldots,M_{d_2}'$ and guess the corrupt factor by iteration. As $d_2\ge 3$ this case reduces to the previous case.

(c) Case $d_1\ge 3, d_2=2$. As done above we are reduced to the stage when we know $L_1,\ldots,L_{d_1-1}$ and M_1. We can determine M_2 as in step 4c and after.

(d) Case $d_1=2, d_2\ge 3$. Reduces to the previous case, using the same the argument as in step 5b.

(e) Case $d_1=2, d_2=1$. Compute the degree 2 homogenous part and factor it to get L_2L_1' where L_1' is the degree 1 part of L_1. Take M_1 to be $f-L_2L_1$.

(f) Case $d_1=2, d_2=1$. Reduces to the previous case, using the same the argument as above.

(g) Case $d_1=d_2=2$. As done above, we are reduced to the stage when we know L_1 and M_1. From 4c we are done if they share a common variable. Let $\phi:\mathbb{F}^{n+1}\mapsto\mathbb{F}^{n+1}$ be an invertible linear transformation such that $\phi(L_1)=y_1$ and $\phi(M_1)=y_2$ and only the x_i's in $\mathsf{var}(L_1)$ depend on y_1 and only the x_i's in $\mathsf{var}(M_1)$ depend on y_2. Hence we know $M_2'=\phi(M_2)(\text{mod } y_1)$ and $L_2'=\phi(L_2)(\text{mod } y_2)$. Determine α, the coefficient of y_1y_2 in $\phi(f)$. Then $\phi(f)$ can be represented as $y_1(\alpha y_2+L_2')+y_2M_2'$, from which we can determine L_2,M_2.

Proof. Before we analyze the correctness, let's verify the running time. From the test described in step 1, the linearity of a multilinear polynomial can be tested in $O(n^3)$ time. In step 1, the time to factorize a multilinear polynomial is $K(n,n,c_{\max})$. Checking linearity of at most n factors can be done in $O(n^4)$ time and interpolation in $n.I(n,n,n,c_{\max})$ time. In step 2, for each of at most n variables, we spend at most $K(n,2n,c_{\max})$ time to factorize, $O(n^3)$ time to determine the variable sets of factors and repeat this step for another polynomial. In step 3, to factorize and testing linearity the time spent is again $\mathrm{poly}(K(n,2n,c_{\max}))$. It follows that the substeps (c) and (d) can clearly be done in time $\mathrm{poly}(K(n,2n,c_{\max}),I(n,n,n,c_{\max}))$ which includes inverting a matrix of dimension $n+1$, factoring, identity testing, etc. As we iterate over pairs of corrupted factors in two lists with at most n factors, the time taken is n^2 times the total time taken till now. Each of the low degree cases take time at most the

[15] affine forms on X_n can be viewed as linear forms on $X_n\cup\{x_{n+1}\}$

time required till now and hence the algorithm runs in time poly$(n, c_{\max})$. Blackboxes to homogenous components can be found as in Lemma 4.

Correctness: We will show the above algorithm succeeds in at least one of the steps based on the cases handled by those steps.

The first step handles the degenerate case that C is in fact a $\Pi\Sigma$ circuit in an obvious way.

Suppose first that $\mathsf{var}(L_1) \not\subseteq \mathsf{var}(T_2)$. Let $x \in \mathsf{var}(L_1) \setminus \mathsf{var}(T_2)$ and let $L_1 = \alpha x + L_1'$. Then,

$$f|_{x=-L_1'/\alpha} \;=\; L_1|_{x=-L_1'/\alpha} \cdot \prod_{i=2}^{d_1} L_i(S_i) \;+\; \prod_{i=1}^{d_2} M_i(S_i')|_{x=-L_1'/\alpha} \;=\; \prod_{i=1}^{d_2} M_i(S_i') \;=\; T_2.$$

Hence after factoring $f|_{x=-L_1'/\alpha}$, Step 2 would have the correct M_i's and factoring $f - f|_{x=-L_1'/\alpha}$ would result in the correct L_i's.

Thus we can assume $\mathsf{var}(L_1) \subseteq \mathsf{var}(T_2)$. We will first assume $d_1, d_2 \geq 3$ and handle the remaining possibilities in Step 5 of the algorithm and verify their correctness later. Suppose, for instance, $x \in S_1 \cap S_{d_2}'$ with $L_1 = \alpha x + L_1'$. Then,

$$f|_{x=-L_1'/\alpha} \;=\; L_1|_{x=-L_1'/\alpha} \cdot \prod_{i=2}^{d_1} L_i(S_i) \;+\; M_{d_2}|_{x=-L_1'/\alpha} \cdot \prod_{i=1}^{d_2-1} M_i(S_i') \;=\; M_{d_2}|_{x=-L_1'/\alpha} \cdot \prod_{i=1}^{d_2-1} M_i(S_i').$$

Note that $M_{d_2}|_{x=-L_1'/\alpha} = M_{d_2} \pmod{L_1}$ is a non-zero linear function as C is simple. We call this the "corrupted" linear form of T_2. Moreover, for a given $x \in \mathsf{var}(L_1)$, there is a unique linear form of T_2 that is corrupted. Hence, in general, after factoring $f|_{x=-L_1'/\alpha}$ we have a list of d_2 linear functions such that all of them are factors of T_2 *except* one that is corrupted.

Since there's no direct way to determine which factor of T_2 is corrupted, we guess it *nondeterministically*; in reality, the algorithm implements this guess by iterating through all the factors of $f|_{x=-L_1'/\alpha}$ assuming each of them to be the corrupt one and checking if this guess is correct (note that this checking can be done efficiently). So, we assume, for concreteness, that the corrupted factor is M_{d_2} and analyze the algorithm. At this stage, the algorithm knows a correct factor of T_2, say M_1, w.l.o.g. (recall we are assuming $d_2 \geq 3$). Using M_1 as a factor T_2 repeat the previous steps (which we did for L_1 as a factor of T_1). This will result is determining all but one of the factors of T_1 (or succeeding already in producing the circuit). Again, assume w.l.o.g. that these are $L_1, \ldots, L_{d_1-1}$.

So, we just need to determine L_{d_1} and M_{d_2} at this point. In step 4(c), we check if any M_i and L_j depend on a common variable x and, if they do, then determine T_1 straightaway as follows. W.l.o.g., suppose $x \in \mathsf{var}(M_1) \cap \mathsf{var}(L_1)$ and $M_1 = \alpha x + M_1'$. Then,

$$f|_{x=-M_1'/\alpha} \;=\; L_1|_{x=-M_1'/\alpha} \cdot \prod_{i=2}^{d_1} L_i(S_i) \;+\; M_1|_{x=-M_1'/\alpha} \cdot \prod_{i=2}^{d_2} M_i(S_i') \;=\; L_1|_{x=-M_1'/\alpha} \cdot \prod_{i=2}^{d_1-1} L_i(S_i) \cdot L_{d_1}.$$

Since we know L_1, we can compute $L_1|_{x=-M_1'/\alpha}$ and we also know $L_2, \ldots, L_{d_1-1}$. Using these we can determine L_{d_1}. Hence T_1 will be completely determined. Then using $f - T_1$, all the M_i's will be determined.

On the other hand, suppose none of the L_i and M_j share a common variable. . Since $d_2 \geq 3$ and we know $L_1, \ldots, L_{d_1-1}$, in Step 4(c.ii), we determine $L_{d_1} \pmod{M_1}$ and $L_{d_1} \pmod{M_2}$ after factoring $f \pmod{M_1}$ and $f \pmod{M_2}$. We then compute an invertible linear transformation ϕ such that $\phi(M_1) = y_1$ and $\phi(M_2) = y_2$ (as M_1, M_2 have disjoint variable sets). Hence, we exactly know $\phi(L_{d_1}) (\mathrm{mod}\ y_1)$ and $\phi(L_{d_1}) (\mathrm{mod}\ y_2)$ using which it is trivial to determine $\phi(L_{d_1})$ (which will be unique) and then L_{d_1}, using ϕ^{-1}, which will again be unique.

Low degree cases: As already analyzed, the only possibility for the steps 1-3 to not succeed are that either $d_1 \leq 2$ or $d_2 \leq 2$ which are handled in the remaining steps. We only analyze one of these cases as the rest either follow from above or are self-explanatory. Suppose $d_1 \geq 3, d_2 \leq 1$. The algorithm picks an $x \in \mathsf{var}(L_1)$ and guesses a $y \in \mathsf{var}(L_2)$. We have,

$$f(X_n) = T_1 + T_2 = (\alpha x + L_1')(y + L_2')(\beta z + L_3') \prod_{i=4}^{d_1} L_i(S_i) + M.$$

The coefficient polynomial of xy in f, scaled by α^{-1}, is $L_3 \cdot \ldots L_{d_1}$ which can be factorized to obtain $L_3, \ldots, L_{d_1}$. The coefficient polynomial of xz in f, scaled by β^{-1}, is $\alpha(y + L_2') \cdot L_4 \cdot \ldots L_{d_1}$. As L_i's are on disjoint sets of variables we can factor the obtained polynomial and determine $(y + L_2')$ after scaling it to make y monic. Having determined all the L_i's determining M_1 from $f - \prod_i L_i$ is trivial. ■

3.3 Reconstructing Set-Multilinear $\Sigma\Pi\Sigma(2)$ Circuits

Having reduced the problem of reconstructing a multilinear $\Sigma\Pi\Sigma(2)$ circuit C to that of computing a non-trivial factor of any of the multiplication gates of $\mathrm{sim}(C)$, we now present an algorithm below (Lemma 9) to exactly reconstruct *set-multilinear* $\Sigma\Pi\Sigma(2)$ circuits. We will be using this algorithm in the final step of our reconstruction algorithm for multilinear $\Sigma\Pi\Sigma(2)$ circuits to prove Theorem 2. We note that the ideas used in the following algorithm are similar to those used in a recent result of Kayal [Kay11] on reconstructing general $\Sigma\Pi\Sigma(k)$ circuits (over an arbitrary $\mathbb{F}$) satisfying certain mild technical conditions.

Lemma 9. *Let $f \in \mathbb{F}[X_n]$ be computed by the following simple set-multilinear $\Sigma\Pi\Sigma(2)$ circuit where L_i's, M_i's are linear forms*

$$f(X_n) = T_1 + T_2 = \prod_{i=1}^{d} L_i(\bar{x}_i) + \prod_{i=1}^{d} M_i(\bar{x}_i).$$

Then, given as input the partition $\{\bar{x}_i\}_{i\in[d]}$ and the blackbox to f, we can determine $\{L_i\}_{i\in[d]}, \{M_i\}_{i\in[d]}$ in randomized time $\mathrm{poly}(n, c_{\max})$, where $c_{\max}$ is the maximum bit length of any coefficient appearing in f.

Proof. If $d = 1$ then we can simply interpolate. Let $d \geq 2$. Substitute all the variables in $X_n \backslash \bar{x}_1$ to independent random values over $\mathbb{F}$ (name this substitution $\bar{R}_1$) and interpolate to get the linear function $L_1'(\bar{x}_1)$. Let $\bar{R}_2$ be another such independent substitution and $M_1'(\bar{x}_1)$ be the one obtained after interpolation. Then,

$$L_1'(\bar{x}_1) = f(\bar{x}_1, \bar{R}_1) = \alpha L_1(\bar{x}_1) + \beta M_1(\bar{x}_1) \quad \text{and} \quad M_1'(\bar{x}_1) = f(\bar{x}_1, \bar{R}_2) = \gamma L_1(\bar{x}_1) + \delta M_1(\bar{x}_1).$$

Clearly with probability $1 - O(n)/|\mathbb{F}|$ (recall that we assumed $|\mathbb{F}| > n^5$), $\alpha, \beta, \gamma, \delta \neq 0$. Also, from simplicity, the polynomials $\prod_{i=2}^{d} L_i$, $\prod_{i=2}^{d} M_i$ are LI and hence the polynomial $\alpha \prod_{i=2}^{d} L_i - \beta \prod_{i=2}^{d} M_i \not\equiv 0$. Now as $\bar{R}_1$, $\bar{R}_2$ are independent substitutions, w.h.p. $\alpha\delta - \beta\gamma = \alpha \prod_{i=2}^{d} L_i(\bar{R}_2) - \beta \prod_{i=2}^{d} M_i(\bar{R}_2) \neq 0$. Hence as L_1, M_1 are LI linear forms, so are L_1', M_1'. Now as $\left(\begin{smallmatrix} \alpha & \beta \\ \gamma & \delta \end{smallmatrix}\right)$ is invertible, $\exists p, q, r, s \neq 0$ such that

$$L_1(\bar{x}_1) = pL_1' + qM_1' \quad \text{and} \quad M_1(\bar{x}_1) = rL_1' + sM_1'.$$

Repeating this for every $\bar{x}_i$ we have LI linear forms $L_1', \ldots, L_d', M_1', \ldots, M_d'$ such that $\forall i \in [d]$:

$$L_i(\bar{x}_i) = p_i L_i' + q_i M_i' \quad \text{and} \quad M_i(\bar{x}_i) = r_i L_i' + s_i M_i'.$$

and hence, $f(X_n) = \prod_{i=1}^{d}(p_i L_i' + q_i M_i') + \prod_{i=1}^{d}(r_i L_i' + s_i M_i') = \alpha \prod_{i=1}^{d}(L_i' + a_i M_i') + \beta \prod_{i=1}^{d}(L_i' + b_i M_i')$, for some $\alpha, \beta \in \mathbb{F}$. As we already have the L_i''s and M_i''s we just need to determine α, β, a_i's, b_i's.

Determine an invertible linear transformation $\phi : \mathbb{F}^n \mapsto \mathbb{F}^n$ such that $\forall i \in [d] : \phi(L'_i) = y_i,\ \phi(M'_i) = z_i$. We have,

$$\phi(f)(y_1, \ldots, y_d, z_1, \ldots, z_d) = \alpha \prod_{i=1}^{d} (y_i + a_i z_i) + \beta \prod_{i=1}^{d} (y_i + b_i z_i) \tag{6}$$

Clearly, having ϕ^{-1} and blackbox access to f, we have blackbox access to $\phi(f)(y_1, \ldots, y_d, z_1, \ldots, z_d)$. Let $d \geq 3$. Substitute $y_4, \ldots, z_4, \ldots$ randomly over $\mathbb{F}$ and interpolate the resulting sparse polynomial $\phi(f)'$ where $a_i \neq b_i$ and w.h.p $\alpha', \beta' \neq 0$

$$\begin{aligned} \phi(f)' &= \alpha'(y_1 + a_1 z_1)(y_2 + a_2 z_2)(y_3 + a_3 z_3) + \beta'(y_1 + b_1 z_1)(y_2 + b_2 z_2)(y_3 + b_3 z_3) && (7) \\ &= y_3(c_1.y_1 y_2 + c_2.z_1 y_2 + c_3.y_1 z_2 + c_4.z_1 z_2) + z_3(c_5.y_1 y_2 + c_6.z_1 y_2 + c_7.y_1 z_2 + c_8.z_1 z_2) && (8) \\ &:= y_3.g(y_1, y_2, z_1, z_2) + z_3.h(y_1, y_2, z_1, z_2) && (9) \end{aligned}$$

Note that we know c_i's. Substitute $y_1 = r z_1$ and $y_2 = s z_2$ to get,

$$z_1 z_2[y_3(c_1.rs + c_2.s + c_3.r + c_4) + z_3(c_5.rs + c_6.s + c_7.r + c_8)] = y_3.g(r z_1, s z_2, z_1, z_2) + z_3.h(r z_1, s z_2, z_1, z_2)$$

and determine all the pairs (r, s) such that $g(r z_1, s z_2, z_1, z_2) = h(r z_1, s z_2, z_1, z_2) = 0$ i.e.

$$c_1.rs + c_2.s + c_3.r + c_4 \ = \ c_5.rs + c_6.s + c_7.r + c_8 \ = \ 0.$$

Eliminating rs from these equations results in a linear function using which one can solve r in terms of s and substitute in one of the above equations to get a quadratic equation in s which can be easily solved. We now show that there will exactly be two (r, s) pairs $(-a_1, -b_2)$ and $(-a_2, -b_1)$ that satisfy the above equations. For such a pair (r, s), from equation (7) we have,

$$z_1 z_2[\alpha'(r+a_1)(s+a_2)(y_3+a_3 z_3)+\beta'(r+b_1)(s+b_2)(y_3+b_3 z_3)] = y_3.g(r z_1, s z_2, z_1, z_2)+z_3.h(r z_1, s z_2, z_1, z_2) = 0$$

But as $(y_3 + a_3 z_3)$ and $(y_3 + b_3 z_3)$ are LI, the only way this this equation holds is that $(r + a_1)(s + a_2) = (r + b_1)(s + b_2) = 0$. As $a_i \neq b_i$, this exactly results in the two pairs $(-a_1, -b_2)$ and $(-a_2, -b_1)$. Also, as $a_i \neq b_i$, repeating this process with coefficient polynomials of y_2, z_2 gives us a_3, b_3. Similarly, repeating this procedure for $y_1, y_2, y_i, z_1, z_2, z_i$ correctly and uniquely determines a_i's and b_i's. Having determined a_i's and b_i's, we can determine α, β by evaluating $\phi(f)(\bar{y}, \bar{z}) := \alpha P(\bar{y}, \bar{z}) + \beta Q(\bar{y}, \bar{z})$ at two independent, randomly chosen substitutions $\bar{S}_1$ and $\bar{S}_2$ to get $\alpha P(\bar{S}_1) + \beta Q(\bar{S}_1) = \mu_1$ and $\alpha P(\bar{S}_2) + \beta Q(\bar{S}_2) = \mu_2$. As we know a_i's and b_i's, from equation (6), we know P, Q. Again from the earlier argument w.h.p $\left(\begin{smallmatrix} P(\bar{S}_1) & Q(\bar{S}_1) \\ P(\bar{S}_2) & Q(\bar{S}_2) \end{smallmatrix}\right)$ is invertible and hence we can determine α, β.

Finally we handle the case when $d = 2$. Interpolate the sparse polynomial $\phi(f)(y_1, y_2, z_1, z_2) = (p_1 y_1 + q_1 z_1)(p_2 y_2 + q_2 z_2) + (r_1 y_1 + s_1 z_1)(r_2 y_2 + s_2 z_2) = \mu_1 y_1 y_2 + \mu_2 y_1 z_2 + \mu_3 z_1 y_2 + \mu_4 z_1 z_2$ to get the μ_i's. Then this equality can also be represented as,

$$\begin{pmatrix} p_1 & r_1 \\ q_1 & s_1 \end{pmatrix} \begin{pmatrix} p_2 & q_2 \\ r_2 & s_2 \end{pmatrix} \ = \ \begin{pmatrix} \mu_1 & \mu_2 \\ \mu_3 & \mu_4 \end{pmatrix}.$$

Clearly this equation has multiple solutions and hence $\phi(f)$ has multiple allowed representations. For our purpose it is enough to choose any one and we choose $\phi(f) = (\mu_1 y_1 + \mu_3 z_1) y_2 + (\mu_2 y_1 + \mu_4 z_1) z_2$. ■

3.4 Proof of Theorem 2

We now combine all the ingredients from previous subsections to prove Theorem 2. We begin by presenting the algorithm `SPSRECON` claimed in the theorem.

Algorithm : $\texttt{SPSRECON}(n, \mathcal{O}_f)$
Input: oracle $\mathcal{O}_f$ for the polynomial $f \in \mathbb{F}[X_n]$ computable by a multilinear $\Sigma\Pi\Sigma(2)$ circuit C as given in equation (4).
Output: sets of linear functions $\{G_1, \ldots, G_r\}, \{L_1, \ldots, L_{d_1}\}, \{M_1, \ldots, M_{d_2}\}$ s.t. equation (4) holds, else $\texttt{FAIL}$.

---

1. *Obtaining G_i's and oracle to* $\text{sim}(C)$: Using Algorithm 2, obtain blackboxes for the factors of f and test their linearity. Output the linear factors as G_i's and define f_s to be the product of non-linear factors. To avoid introducing more notations we assume $\mathsf{var}(f_s) = X_n$.

2. $\mathsf{var}(T_1) \neq \mathsf{var}(T_2)$ *case* : By iterating over $x \in X_n$ guess a variable x (if it exists) such that either $x \in \mathsf{var}(T_1)$ but $x \notin \mathsf{var}(T_2)$ or vice versa. Obtain the coefficient polynomial of x, factorize it and determine a linear factor ℓ, if it exists. If the output of $\texttt{RECONFACTOR}(n, \mathcal{O}_{f_s}, \ell)$ is $\texttt{FAIL}$, proceed, else test whether the output $\{L'_1, \ldots, L'_{d_1}\}, \{M'_1, \ldots, M'_{d_2}\}$ form a multilinear $\Sigma\Pi\Sigma(2)$ circuit for f_s and if they do, output these sets, else proceed.

3. *Case $d_1 = 1$ or $d_2 = 1$.* W.l.o.g. let $d_2 = 1$. For $d_1 = 2$, compute the degree 2 homogenous part and factor it to get $L_1 L_2$. Take M_1 to be the linear part of f_s. For $d_1 \geq 3$, iterate over $x, y \in X_n$ to guess an $x \in \mathsf{var}(L_1)$ and a $y \in \mathsf{var}(L_2)$. Factorize the coefficient polynomial of xy in f_s and determine a linear factor ℓ, if it exists. If the output of $\texttt{RECONFACTOR}(n, \mathcal{O}_{f_s}, \ell)$ is $\texttt{FAIL}$, proceed, else test whether the output $\{L'_1, \ldots, L'_{d_1}\}, \{M'_1, \ldots, M'_{d_2}\}$ form a multilinear $\Sigma\Pi\Sigma(2)$ circuit for f_s and if they do, output these sets, else proceed to the next pair.

4. $\mathsf{par}(T_1) \neq \mathsf{par}(T_2)$ *case* : By iterating over pairs of distinct variables $x, y \in X_n$ guess a pair x, y (if it exists) such that x, y are in distinct sets of $\mathsf{par}(T_1)$ but in one set of $\mathsf{par}(T_2)$, or vice versa. Obtain the coefficient polynomial of xy in f_s, factorize it and determine a linear factor ℓ, if it exists. If the output of $\texttt{RECONFACTOR}(n, \mathcal{O}_{f_s}, \ell)$ is $\texttt{FAIL}$, proceed, else test whether the output $\{L'_1, \ldots, L'_{d_1}\}, \{M'_1, \ldots, M'_{d_2}\}$ form a multilinear $\Sigma\Pi\Sigma(2)$ circuit for f_s and if they do, output these sets, else proceed.

5. *Case $d_1 = 2$ or $d_2 = 2$.* W.l.o.g. let $d_2 = 2$. If $d_1 \geq 4$, again as in step 3, we can guess a triplet (x, y, z) such that $x \in \mathsf{var}(L_1), y \in \mathsf{var}(L_2), z \in \mathsf{var}(L_3)$, factorize the coefficient polynomial of xyz in f_s to get a linear factor of T_1 and use $\texttt{RECONFACTOR}$. If $d_1 = 3$, again guess a pair x, y such that $x \in \mathsf{var}(L_1), y \in \mathsf{var}(L_2)$, factorize the coefficient polynomial of xy in f_s to get a linear factor of T_1 and use $\texttt{RECONFACTOR}$. Let $d_1 = 2$. By iterating over pairs of distinct variables $x, y \in X_n$ guess a pair x, y (if it exists) such that x, y are in distinct sets of $\mathsf{par}(T_1)$ but in one set of $\mathsf{par}(T_2)$, or vice versa. Obtain $\Delta_{xy}(f_s)$ and factorize it to obtain at most 2 linear factors out of which, at least one would be a factor of T_2. Use $\texttt{RECONFACTOR}$ as earlier. If no pair is successful then construct a partition of the variable set assuming that $x \neq y$ are in different sets iff the coefficient of xy in f_s is non-zero, and use the following case.

6. *Set-multilinear case* : Let $\mathsf{par}(T_1) = \mathsf{par}(T_2) = \{\bar{x}_i\}_{i \in [d]}$. Determine this partition by iterating over all pairs of distinct variables $x, y \in X_n$ and concluding that x, y are in the same set of the partition iff the coefficient polynomial of xy in f_s is 0. For each $i \in [d]$, homogenize the linear forms on $\bar{x}_i$ by a distinct variable y_i i.e. obtain the oracle for the polynomial $f_s^H = f_s(\frac{\bar{x}_1}{y_1}, \ldots, \frac{\bar{x}_d}{y_d}) \cdot y_1 \cdot \ldots y_d$ where $\frac{\bar{x}_1}{y_1} = (\frac{x}{y_1},)_{x \in \bar{x}_1}$. Using the oracle to f_s^H and Algorithm 9 reconstruct the multilinear $\Sigma\Pi\Sigma(2)$ circuit for f_s^H in which substitute the y_i's by 1 to obtain a multilinear $\Sigma\Pi\Sigma(2)$ circuit for f_s. Output the appropriate sets of linear functions corresponding to L_i's and M_i's.

Now, we prove the running time and correctness of the above algorithm.

Proof. Lets first bound the running time. In step 1, the time to factorize a multilinear polynomial is $K(n, n, c_{\max})$. Checking linearity of at most n factors can be done in $O(n^4)$ time and linear interpolation in $n.I(n, n, n, c_{\max})$ time. In step 2, for each of at most n variables, the time spent to factorize, test linearity, interpolating linear factors and `RECONFACTOR` is again $\text{poly}(n, c_{\max})$. In steps 3, 4 and 5, analysis is similar, except that we iterate over $O(n^2)$ pairs or $O(n^3)$ triplets of variables. In step 6, again the time is $\text{poly}(n, c_{\max})$. The failure probability of the algorithm is $2^{-\Omega(n)}$ from an analysis similar to the one in the proof of Lemma 8.

Correctness : We will show that the above algorithm succeeds in at least one of the steps based on the cases handled by those steps. Let f be the multilinear polynomial computed by a multilinear $\Sigma\Pi\Sigma(2)$ circuit C, as given in equation (4) restated below

$$f(X_n) = G \cdot (T_1 + T_2) = G_1 \cdot \ldots G_r \cdot \left(\prod_{i=1}^{d_1} L_i(S_i) + \prod_{i=1}^{d_2} M_i(S_i') \right)$$

From Lemma 7 it follows that G_i's exactly comprise of the linear factors of f and hence f_s is the polynomial computed by $\text{sim}(C)$. Now if $\mathsf{var}(T_1) \neq \mathsf{var}(T_2)$ then w.l.o.g. say $\exists x \in \mathsf{var}(T_1) \setminus \mathsf{var}(T_2)$. In step 2, due to an iteration over all the variables, this variable would be guessed in one of the iterations. W.l.o.g. let $L_1 = \alpha x + L_1'$. At this point c the coefficient polynomial of x would be $\alpha L_2 \cdot \ldots L_{d_1}$ and on any linear factor ℓ from this list, `RECONFACTOR` would correctly output the L_i's and M_i's. The only possibility for this to not succeed is the case when $d_1 = 1$ which is handled in step 3 as if, say $f_s = L_1 + (\alpha x + M_1')(\beta y + M_2')M_3 \ldots$, then in one of the iterations the pair x, y would be guessed and the coefficient polynomial of xy would be $\alpha\beta M_3 \ldots$ which has $M_3, \ldots$ as its linear factors and hence `RECONFACTOR` would succeed.

Hence, the only possibility for steps 2-3 to not succeed is the case $\mathsf{var}(T_1) = \mathsf{var}(T_2)$ and hence we are now reduced to this case. Now if it is the case that $\mathsf{par}(T_1) \neq \mathsf{par}(T_2)$ then, as $\mathsf{var}(T_1) = \mathsf{var}(T_2)$, there would exist a pair x, y such that w.l.o.g. x, y occur in the same linear factor of T_2 but different factors of T_1, say x occurs in L_1, y occurs in L_2 but both x, y occur in M_1. In one of the iterations, this pair would be guessed and we would have the coefficient polynomial of xy to have $L_3, \ldots$ as its linear factors. On any such factor `RECONFACTOR` would correctly output the L_i's and M_i's. The only possibility for this to not succeed is the case when $d_1 = 2$ which is handled in step 5. If $d_2 \geq 4$ then the argument is similar to the one in the $d_1 = 1$ case above. If $d_2 = 3$, and as $\mathsf{var}(T_1) = \mathsf{var}(T_2)$, there would exist a pair x, y such that x, y occur in the same L_i but different M_i's, as if not then either $\mathsf{par}(T_2) = \mathsf{par}(T_1)$ or $|\mathsf{par}(T_2)| = 1$. Hence from a similar argument step 5 would succeed. If $d_2 = 2$ and $\mathsf{par}(T_1) \neq \mathsf{par}(T_2)$ then w.l.o.g. , as before, a pair x, y would be guessed such that both occur in the same linear factor of T_2 but different factors of T_1, say x occurs in L_1, y occurs in L_2 but both x, y occur in M_1. In this case, we leave it to the reader to verify that $M_2 \mid \Delta_{xy}(f_s) \not\equiv 0$, as f_s is irreducible of degree 2 (see Corollary 1).

The only possibility for steps 2-5 to not succeed is the case when $\mathsf{var}(T_1) = \mathsf{var}(T_2)$, $\mathsf{par}(T_1) = \mathsf{par}(T_2)$ and $d = d_1 = d_2 \geq 3$. Hence we are now reduced to the case when C is set-multilinear. First we show that step 6 correctly determines the partition i.e. for every pair of distinct variables x, y, the coefficient polynomial of xy is zero iff they are in the same set of the partition. If they are in the same set then clearly it is 0. If they are not, say $f_s = (\alpha x + L_1')(\beta y + L_2')L_3 \ldots + (\gamma x + M_1')(\delta y + M_2')M_3 \ldots$ then the coefficient polynomial of xy would be $\alpha\beta L_3 \ldots + \gamma\delta M_3 \ldots$. If this is zero then, by set-multilinearity, we would have $L_3 \sim M_3$ which violates simplicity. Having determined the correct partition, and homogenizing with distinct variables, we would be assured that Algorithm 9 would correctly output the required linear forms. ■

4 Reconstruction of Multilinear $\Sigma\Pi\Sigma\Pi(2)$ Circuits

In this section, we prove our main result (Theorem 1) by presenting an algorithm to reconstruct multilinear $\Sigma\Pi\Sigma\Pi(2)$ circuits over an arbitrary field $\mathbb{F}$. Recall that our primary objects of interest are polynomials of the form given by

$$f(x_1, \dots, x_n) = \mathrm{gcd}(C) \cdot \mathrm{sim}(C) = G \cdot (T_1 + T_2) = G_1 \cdot \dots G_r \cdot \left(\prod_{i=1}^{d_1} P_i + \prod_{i=1}^{d_2} Q_i \right), \tag{10}$$

where $\{G_i\}_{i\in[r]}$, $\{P_i\}_{i\in[d_1]}$ and $\{Q_i\}_{i\in[d_2]}$ are sets of variable-disjoint s-sparse irreducible multilinear polynomials. The main ingredients of the proof, as in the case of depth-3, are: (i) obtaining blackbox access $\mathrm{sim}(C)$ by separating out $\mathrm{gcd}(C)$, (ii) reconstructing when a factor of one of the T_i's is known, and (iii) reconstructing set-multilinear $\Sigma\Pi\Sigma\Pi(2)$ circuits. The crucial difference is, whereas we had linear factors in the depth-3 case, here we have s-sparse multilinear polynomials as factors. This difference makes the case of $\Sigma\Pi\Sigma\Pi(2)$ reconstruction significantly more intricate than $\Sigma\Pi\Sigma(2)$ circuits.

Before we move on to the proof, as promised in the introduction, we will **handle the degeneracy conditions** avoided in the formal statement of Theorem 1, namely, the conditions $\|C\| \le 4s^4$ or $\deg_{\mathrm{m}}(C) < 3$. If $\|C\| \le 4s^4$, we simply interpolate the polynomial as a $\Sigma\Pi$ circuit using Lemma 3 with sparsity parameter $8s^4$. Clearly this is also a $\Sigma\Pi\Sigma\Pi(2)$ circuit of size only polynomially larger than C. For the other condition, note that if *both* d_1 and d_2 are at most 2, then $\|T_1\|, \|T_2\| \le s^2$ and this case has already been handled by the above interpolation solution. So, we can assume that, say, $d_1 \ge 3$ and $d_2 \le 2$. The arguments needed to handle these cases are already contained in the proof to be given in Section 4.4. For example, if $d_1 \ge 3$ and $d_2 = 2$, the arguments when $\mathsf{var}(T_1) = \mathsf{var}(T_2)$ but $\mathsf{par}(T_1) \ne \mathsf{par}(T_2)$ that are used to justify Step 3 of algorithm `SPSPRECON` would help find a factor Q_2 of T_2 and then we can use the algorithm `SPSPFACTOR` to completely reconstruct C (note that we can, by now, assume that $\|C\| \ge 4s^4$. The other degenerate cases are handled similarly.

4.1 Obtaining blackbox access to $\mathrm{sim}(C)$

Lemma 10 (Density Lemma). *Let $0 \not\equiv P, Q \in \mathbb{F}[X_n]$ be polynomials such that P is multilinear. Then $P \mid Q \Rightarrow ||Q|| \ge ||P||$.*[16]

Proof. Let $Q = P.R$ for some $0 \not\equiv R \in \mathbb{F}[X_n]$ of degree at most d. The proof is via induction on $|\mathsf{var}(R)|$. We first reduce the claim to the case when $\mathsf{var}(R) \subseteq \mathsf{var}(P)$. If $\mathsf{var}(R) = \emptyset$, then $R \in \mathbb{F}^*$ and hence the assertion holds. For an $x \in \mathsf{var}(R) \setminus \mathsf{var}(P)$, let $R = \sum_{i=0}^{d} R_i.x^i$ for some $R_i's \in \mathbb{F}[\mathsf{var}(R)\setminus\{x\}]$ and $R_k \not\equiv 0$ for some $k \in [0:d]$. Then, $Q = \sum_{i=0}^{d} P.R_i.x^i$ and hence $||Q|| = \sum_{i=0}^{d} ||P.R_i||$. If $||P.R_k|| \ge ||P||$ then $||Q|| \ge ||P||$. Hence w.l.o.g. we assume $\mathsf{var}(R) \subseteq \mathsf{var}(P)$.

Let $x \in \mathsf{var}(R) \cap \mathsf{var}(P)$. Let $P = P_1 x + P_0$ and $R = \sum_{i=l}^{h} R_i.x^i$ where $P_0, P_1 \in \mathbb{F}[\mathsf{var}(P)\setminus\{x\}]$ are multilinear (also $P_1 \not\equiv 0$) and $R_i's \in \mathbb{F}[\mathsf{var}(R)\setminus\{x\}]$ are such that x^h, x^l are the highest and lowest powers of x appearing in R respectively, i.e. $R_h, R_l \not\equiv 0$. Then, $Q = (P_1 x + P_0)(\sum_{i=l}^{h} R_i.x^i) = P_1.R_h.x^{h+1} + \dots + P_0.R_l.x^l$ and hence $||Q|| \ge ||P_1.R_h|| + ||P_0.R_l||$. By induction hypothesis, $||P_1.R_h|| \ge ||P_1||$ and $||P_0.R_l|| \ge ||P_0||$ (when $P_0 \not\equiv 0$). Hence, $||Q|| \ge ||P_1|| + ||P_0|| = ||P||$. If $P_0 \equiv 0$ then still $||Q|| \ge ||P_1|| = ||P||$. ■

Lemma 11. *Let $f \in \mathbb{F}[X_n]$ be the polynomial computed by a multilinear $\Sigma\Pi\Sigma\Pi(2)$ circuit C of size s with $||C|| \ge 2s^3$ as given by equation (10). Then, for any s-sparse multilinear polynomial R, $R \mid f \iff R \mid G$.*

Proof. Clearly if $R \mid G$ then $R \mid f$. For the converse, suppose $R \mid f$ but $R \nmid G$ and hence $R \mid \mathrm{sim}(C)$. Since any s-sparse multilinear polynomial is a product of irreducible s-sparse multilinear polynomials,

[16] In general, for two polynomials P, Q, $P \mid Q \not\Rightarrow ||Q|| \ge ||P||$. For instance $\sum_{i=0}^{n-1} x^i \mid x^n - 1$.

w.l.o.g. we show a contradiction assuming R is irreducible. Since $||C|| = \max\{||T_1||, ||T_2||\} > 2s^3$, w.l.o.g. let $||T_1|| > 2s^3$. Let $x \in \mathsf{var}(R)$. Then from Lemma 6 we have $D_x(T_1+T_2, R) = D_x(T_1, R)+D_x(T_2, R) \equiv 0$. Hence if $D_x(T_1, R) \equiv 0$ then $D_x(T_2, R) \equiv 0$, and vice versa. But if $D_x(T_1, R) = D_x(T_2, R) \equiv 0$ then $R \mid \gcd(T_1, T_2) = 1$ which is contradiction. Hence $D_x(T_1, R), D_x(T_2, R) \not\equiv 0$. W.l.o.g. let $x \in \mathsf{var}(P_1) \cap \mathsf{var}(Q_1)$. Then we have,

$$D_x(P_1, R) \prod_{i=2}^{d_1} P_i = -D_x(Q_1, R) \prod_{i=2}^{d_2} Q_i.$$

Since $\gcd\left(\prod_{i=2}^{d_1} P_i, \prod_{i=2}^{d_1} Q_i\right) = 1$, we have $\prod_{i=2}^{d_1} P_i \mid D_x(Q_1, R)$. As R, Q_1 are s-sparse multilinear polynomials we have $||D_x(Q_1, R)|| \leq 2s^2$. Also, since $||T_1|| > 2s^3$ and P_1 is s-sparse, $|| \prod_{i=2}^{d_1} P_i|| \geq 2s^2$. But as $\prod_{i=2}^{d_1} P_i$ is multilinear, from Lemma 10, $|| \prod_{i=2}^{d_1} P_i|| \leq ||D_x(Q_1, R)||$, a contradiction. ■

Given this lemma, we can obtain each of the G_i's by first factoring f (Lemma 2) into irreducible factors and then testing each of these factors if it is s-sparse (Lemma 3). By dividing f by the product $G = \prod_{i=1}^{r} G_i$, we obtain blackbox access for $\mathrm{sim}(C)$.

4.2 Determining a factor of any one of the product gates

The following lemma shows that for the purpose of reconstructing a simple multilinear $\Sigma\Pi\Sigma\Pi(2)$ circuit (except for a few boundary cases), it is enough to determine a factor of one of the two multiplication gates.

Lemma 12. *Let $f \in \mathbb{F}[X_n]$ be the polynomial computed by the following* simple *multilinear $\Sigma\Pi\Sigma\Pi(2)$ circuit C of size s, where the P_i's and the Q_i's are irreducible, $\{S_i\}_{i\in[d_1]}$ and $\{S'_i\}_{i\in[d_2]}$ are partitions of X_n, $||C|| > 4s^3$, and $d_1, d_2 \geq 3$:*

$$f(X_n) = T_1 + T_2 = \prod_{i=1}^{d_1} P_i(S_i) + \prod_{i=1}^{d_2} Q_i(S'_i). \tag{11}$$

Given n, s, blackbox access to f, and the s-sparse polynomial P_1 explicitly, there is a randomized algorithm `SPSPFACTOR` *that outputs the P_i's and the Q_i's. The algorithm runs in time $\mathrm{poly}(n, s, c_{\max})$, where $c_{\max}$ is the maximum bit length of any coefficient in f.*

Before moving on to the proof of Lemma 12, we first show that it is enough to determine a factor of each of the two multiplication gates, i.e., T_1 and T_2, such that these two factors depend on a common variable.

Lemma 13. *Let $f \in \mathbb{F}[X_n]$ be the polynomial computed by a simple multilinear $\Sigma\Pi\Sigma\Pi(2)$ circuit C of size s as given in equation (11). Given n, s, blackbox access to f, and the irreducible multilinear polynomials P_1 and Q_1 explicitly such that $\mathsf{var}(P_1) \cap \mathsf{var}(Q_1) \neq \emptyset$, there is a randomized algorithm* `RECONPAIR` *described below that outputs the P_i's and Q_i's in time $\mathrm{poly}(n, s, c_{\max})$, where $c_{\max}$ is the maximum bit length of any coefficient in f.*

Algorithm : `RECONPAIR`$(n, s, \mathcal{O}_f, P_1, Q_1)$
Input: oracle $\mathcal{O}_f$ for the polynomial $f \in \mathbb{F}[X_n]$ computable by a simple size-s multilinear $\Sigma\Pi\Sigma\Pi(2)$ circuit C as given in equation (11) and irreducible factors P_1 of T_1 and Q_1 of T_2.
Output: sets of s-sparse irreducible polynomials $\{P_1, \ldots, P_{d_1}\}, \{Q_1, \ldots, Q_{d_2}\}$ s.t. equation (11) holds, else `FAIL`.

---

1. Using blackbox-PIT determine an $x \in \mathsf{var}(P_1) \cap \mathsf{var}(Q_1)$, if it exists and if not, output `FAIL`. Obtain the blackbox to $D_x(Q_1, P_1)$ and output `FAIL` if it is zero. Else using Algorithm 2 compute the blackboxes to the irreducible factors of $D_x(Q_1, P_1)$ with their respective multiplicities. Determine the multilinearity of each of these factors by testing the identity of the coefficient polynomial of x^2 for every variable x. Interpolate the multilinear s-sparse ones using Algorithm 3, aimed to interpolate s-sparse polynomials and identity testing.

2. Obtain blackboxes to the irreducible multilinear factors of $D_x(f, P_1)$ with their respective multiplicities and interpolate them. Using these factors and their respective multiplicities determine the factors of $D_x(f, P_1)/D_x(Q_1, P_1)$ and let them be $Q_2, \dots, Q_{d_2}$, after easily taking care of a scalar multiple by evaluating at a random input. Similarly, from $D_x(f, Q_1)$ determine $P_2, \dots, P_{d_1}$ and output the sets $\{P_1, \dots\}$, $\{Q_1, \dots\}$.

Proof. From Lemma 6, $D_x(f, P_1) = D_x(T_1 + T_2, P_1) = D_x(T_2, P_1) = D_x(Q_1, P_1)Q_2 \cdot \dots Q_{d_2}$, where $D_x(Q_1, P_1) \not\equiv 0$ by simplicity. From this $Q_2, \dots, Q_{d_2}$ can be obtained by first obtaining blackboxes for the factors of $(D_x(f, P_1)) / (D_x(Q_1, P_1))$ (Lemma 2) and interpolating the factors using the sparse polynomial interpolation algorithm (Lemma 3). ■

We now present the algorithm `SPSPFACTOR` needed in the proof of Lemma 12.

Algorithm : `SPSPFACTOR`$(n, s, \mathcal{O}_f, P_1)$.
Input: oracle $\mathcal{O}_f$ for the polynomial $f \in \mathbb{F}[X_n]$ computable by a simple size-s multilinear $\Sigma\Pi\Sigma\Pi(2)$ circuit C as given in equation (11) and irreducible factor P_1 of T_1.
Output: sets of s-sparse irreducible polynomials $\{P_1, \dots, P_{d_1}\}, \{Q_1, \dots, Q_{d_2}\}$ s.t. equation (11) holds, else `FAIL`.

---

1. $\mathsf{var}(P_1) \not\subseteq \mathsf{var}(T_2)$ *case* : For every $x \in \mathsf{var}(P_1)$ obtain the blackbox for $D_x(f, P_1)$ and using Lemma 2 compute the blackboxes to its irreducible factors with respective multiplicities. Among these, remove the factors of $D_x(1, P_1)$ and scale the rest appropriately such that their product equals $D_x(f, P_1)/D_x(1, P_1)$. Test whether the remaining factors $Q_1, \dots, Q_{d_2}$ are s-sparse multilinear polynomials on disjoint set of variables. If not, proceed to the next x, else test whether $f - \prod_i Q_i$ is a product of s-sparse multilinear polynomials $P_1, \dots, P_{d_1}$ on disjoint set of variables. If yes, output the sets $\{P_1, \dots\}$, $\{Q_1, \dots\}$, else proceed to the next x.

2. Pick any $x \in \mathsf{var}(P_1)$, obtain blackbox for $D_x(f, P_1)$, using Lemma 2 compute the blackboxes to its s-sparse multilinear factors and interpolate them to get a list $\mathcal{L}$. For each $Q \in \mathcal{L}$, using Step 1, ensure that if Q is some Q_i then $\mathsf{var}(Q) \subseteq \mathsf{var}(T_1)$, or else output the reconstructed circuit. For every $Q \in \mathcal{L}$ and $z \in \mathsf{var}(Q)$ obtain the s-sparse multilinear factors of $D_z(f, Q)$ and interpolate them to get a list $\mathcal{L}_Q^z$. For every $Q' \in \mathcal{L}$ and $P \in \mathcal{L}_Q^z \cup \{P_1\}$ test if `RECONPAIR`$(n, s, \mathcal{O}_f, P, Q')$ outputs a multilinear $\Sigma\Pi\Sigma\Pi(2)$ circuit for f.

3. For every $Q \in \mathcal{L}$, repeat Step 2 for Q (instead of P_1).

4. We are reduced to the case when w.l.o.g. $\mathsf{var}(P_1 \cdot \dots P_{d_1 - 1}) \subseteq \mathsf{var}(Q_{d_2})$ and $\mathsf{var}(Q_1 \cdot \dots Q_{d_2 - 1}) \subseteq \mathsf{var}(P_{d_1})$. Let $\mathcal{L}' \subseteq \mathcal{L}$ s.t. it contains all the 1-dense polynomials in $\mathcal{L}$. If $|\mathcal{L}'| < 3 \log s + 2$ then iterate over all subsets $\mathcal{B}$ of $\mathcal{L}'$ of size at most $2 \log s + 1$ and do:

   (a) Determine $\mathcal{G} \subseteq \mathcal{L}' \backslash \mathcal{B}$ such that $s < \prod_{Q \in \mathcal{G}} ||Q|| \leq s^2 - 1$. Let $g = \prod_{Q \in \mathcal{G}} Q$ and M_g be the set of non-zero monomials in g (as subsets of $\mathsf{var}(g)$).

   (b) Obtain blackboxes to all f_S's such that $S \in M_g$ and where $f = \sum_{S \subseteq \mathsf{var}(g)} f_S(X_n \backslash \mathsf{var}(g)) \prod_{x \in S} x$.

(c) For every $S \in M_g$, obtain blackboxes to the s-sparse multilinear factors of f_S and interpolate them to get a list $\mathcal{L}_S$. For every $Q \in \mathcal{L}$, $z \in \mathsf{var}(Q)$, $P \in \mathcal{L}_Q^z$ and $Q' \in \mathcal{L}_S$ test if $\mathtt{RECONPAIR}(n, s, \mathcal{O}_f, P, Q')$ outputs a multilinear $\Sigma\Pi\Sigma\Pi(2)$ circuit for f.

Else if $|\mathcal{L}'| \geq 3\log s + 2$, pick any $3\log s + 2$-sized subset $\mathcal{D}$ of $\mathcal{L}'$ and carry out the above sub-steps for $\mathcal{D}$.

5. For every $Q \in \mathcal{L}$ and $z \in \mathsf{var}(Q)$ repeat Step 4 for $\mathcal{L}_Q^z$ (instead of $\mathcal{L}$). Output FAIL.

Proof of Lemma 12. First let's analyze the running time. Step 1 requires $\mathrm{poly}(n, s, c_{\max})$ time by earlier analysis. In Step 2, as f and P_1 are multilinear, $D_x(f, P_1)$ has degree of every variable at most 2 and hence has at most $2n$ factors and so $|\mathcal{L}| \leq 2n$. Determining s-sparsity and interpolation can be done in $\mathrm{poly}(n, s, c_{\max})$ time. As Step 1, RECONPAIR can be done in $\mathrm{poly}(n, s, c_{\max})$ time and as the sizes of the lists produced is at most $2n$, Step 2 uses them at most $\mathrm{poly}(n)$ times and hence the running time of Step 2 is $\mathrm{poly}(n, s, c_{\max})$. Step 3 uses Step 2 at most $2n$ times. In Step 4, the number of subsets of a $(3\log s + 2)$-sized set is at most $4s^3$ and hence the sub-steps are carried out at most $4s^3$ times. Step 4(a) can be done in $\mathrm{poly}(n, s)$ time. In Step 4(b), as $|M_g| \leq s^2$, it can be done in $\mathrm{poly}(n, s)$ time. Step 4(c) again uses $\mathrm{poly}(n, s, c_{\max})$ time. As Step 5 uses Step 4 $\mathrm{poly}(n)$ times the time bound follows.
Correctness: Suppose, to begin with, that $\mathsf{var}(P_1) \not\subseteq \mathsf{var}(T_2)$. Let $x \in \mathsf{var}(P_1) \setminus \mathsf{var}(T_2)$ and note that

$$D_x(f, P_1) = D_x(T_1, P_1) + D_x(T_2, P_1) = 0 + D_x(1, P_1) \cdot T_2.$$

Thus, $Q_1, \ldots, Q_{d_2}$ are among the s=sparse multilinear factors of $D_x(f, P_1)$. We remove the factors of $D_x(1, P_1)$ from those of $D_x(f, P_1)$ scaling the rest appropriately, we obtain $D_x(f, P_1)/D_x(1, P_1)$. If the remaining factors are s-sparse multilinear polynomials on disjoint sets of variables, then we output them as $Q_1, \ldots, Q_{d_2}$. By factoring $f - \prod Q_j$ and testing if those factors are also s-sparse and multilinear on disjoint sets of variables, we obtain $P_1, \ldots, P_{d_1}$. Thus, we are done in the case when $\mathsf{var}(P_1) \not\subseteq \mathsf{var}(T_2)$.

So, we now assume that $\mathsf{var}(P_1) \subseteq \mathsf{var}(T_2)$. For concreteness, suppose that P_1 and Q_{d_2} share a variable x. Now,

$$D_x(f, P_1) = 0 + D_x(Q_{d_2}, P_1) \cdot Q_1 \cdot \ldots \cdot Q_{d_2 - 1}.$$

Let $\mathcal{L}$ be the list of s-sparse multilinear factors of $D_x(f, P_1)$. Since $d_2 \geq 2$, the list $\mathcal{L}$ contains at least one "genuine" factor of T_2,namely, $Q_1, \ldots, Q_{d-2-1}$ and some "spurious" factors, namely, those of $D_x(Q_{d_2}, P_1)$. We *nondeterministically* guess a genuine factor of T_2, say Q_1 for concreteness[17]. We thus have a (guessed) factor Q_1 of T_2 and will be done by an argument similar to the above paragraph if $\mathsf{var}(Q_1) \not\subseteq \mathsf{var}(T_1)$. It follows that we are done if $\exists j,\ 1 \leq j \leq d_2 - 1,\ \mathsf{var}(Q_j) \not\subseteq \mathsf{var}(T_1)$.

Hence we can assume now that $\mathsf{var}(Q_1 \cdot \ldots \cdot Q_{d_2-1}) \subseteq \mathsf{var}(T_1)$. We will argue that we can in fact replace T_1 here by a single factor of T_1. Suppose $\mathsf{var}(Q_1 \cdot \ldots \cdot Q_{d_2-1})$ is split between two factors P_i and P_j of T_1, i.e., we have two variables $y, z \in \mathsf{var}(Q_1 \cdot \ldots \cdot Q_{d_2-1})$ such that $y \in \mathsf{var}(P_i) \setminus \mathsf{var}(P_j)$ and $z \in \mathsf{var}(P_j) \setminus \mathsf{var}(P_i)$. Let $y \in \mathsf{var}(Q)$, for some Q among $Q_1, \ldots, Q_{d_2-1}$. Now, $D_y(f, Q)$ is $D_y(P_i, Q)P_j \ldots$ and hence has P_j intact among its s-sparse multilinear factors. Then, the variable z is common between P_j and some Q' among $Q_1, \ldots, Q_{d_2-1}$. We can now run RECONPAIR from Lemma 13 with the factors P_i and Q' and be done. Hence if $\mathsf{var}(Q_1 \cdot \ldots \cdot Q_{d_2-1})$ is split between two or more factors of T_1, the algorithm nondeterministically guesses Q, P_i, and y as above, computes $D_y(f, Q)$, and guesses P_j among the factors of $D_y(f, Q)$, guesses Q', and runs $\mathtt{RECONPAIR}(n, s, \mathcal{O}_f, P_i, Q')$.

Thus, we only need to consider the case when $\mathsf{var}(Q_1 \cdot \ldots \cdot Q_{d_2-1})$ is not split among factors of T_1. W.l.o.g., we assume $\mathsf{var}(Q_1 \cdot \ldots \cdot Q_{d_2-1}) \subseteq \mathsf{var}(P_{d_1})$. By now, we guessed a factor of T_2, say Q_1. Applying

[17] The algorithm SPSPFACTOR implements this guess by iterating through all choices. It is easy to see there are only a polynomial number (in s) of such choices and the correctness of a guess can be verified in randomized polynomial time.

the foregoing argument to Q_1 and T_2 (just as we did for P_1 as a factor of T_1), we again conclude we only need to consider the case, w.l.o.g., that $\mathsf{var}(P_1 \cdot \ldots \cdot P_{d_1}) \subseteq \mathsf{var}(Q_{d_2})$.

To summarize the argument so far, we are given the factor P_1 of T_1, a lists containing factors $Q_1, \ldots, Q_{d_2-1}$ of T_2, and we can assume that $\mathsf{var}(Q_1 \cdot \ldots \cdot Q_{d_2-1}) \subseteq \mathsf{var}(P_{d_1})$ and $\mathsf{var}(P_1 \cdot \ldots \cdot P_{d_1}) \subseteq \mathsf{var}(Q_{d_2})$. Our goal now is to fish for a factor Q of T_2 and then apply `RECONPAIR` on P_1 and that Q. How do we do that? Recall the list $\mathcal{L}$ consisting of s-sparse multilinear factors of $D_x(f, P_1)$ and containing genuine factors $Q_1, \ldots, Q_{d_2-1}$ and spurious factors from $D_x(Q_{d_2}, P_1)$. Because both P_1 and Q_{d_2} are s-sparse, $\|D_x(Q_{d_2}, P_1)\| \leq 2s^2$. Thus the product of the sparsity of the spurious factors is at most $2s^2$.

By assumption $\|C\| > 4s^3$ and hence $\|T_1\| > 4s^3$ or $\|T_2\| > 4s^3$. Let's assume the latter. Then, since Q_{d_2} is s-sparse, we must have $\|Q_1 \cdot \ldots \cdot Q_{d_2-1}\| > 4s^2$. But we are in the case when $\mathsf{var}(Q_1 \cdot \ldots \cdot Q_{d_2-1}) \subseteq \mathsf{var}(P_{d_1})$ and $\|P_{d_1}\| \leq s$. Thus, there must be a monomial $X_S = \prod_{i \in S} x_i$ for $S \subseteq \mathsf{var}(P_1)$ that is produced by $Q_1 \cdot \ldots \cdot Q_{d_2-1}$ but not by P_1. The coefficient polynomial f_S of X_S in f must contain Q_{d_2} as a factor. Our task is to compute f_S and extract Q_{d_2} from it by factoring.

We first clean up $\mathcal{L}$ to obtain $\mathcal{L}' \subseteq \mathcal{L}$ by removing factors that are single variables, i.e., every polynomial in $\mathcal{L}'$ is 1-dense. Since $\|D_x(Q_{d_2}, P_1)\| \leq 2s^2$, the number of spurious factors in $\mathcal{L}'$ is at most $2 \log s + 1$. Hence if we take any subset $3 \log s + 2$, we will have at least $\log s + 1$ 1-dense genuine factors Q_i in it. Clearly, the product of these Q_i's is at least $2s$-dense (recall that $\mathsf{var}(Q_i)$ are disjoint). Furthermore, since $\|Q_1 \cdot \ldots \cdot Q_{d_2-1}\| > 4s^2$ and $\|Q_i\| \leq s$, there must be a subset of these Q's such that their product has sparsity at most s^2. Therefore, we *guess* a subset $\mathcal{B}$ of size at most $2 \log s + 1$ of spurious factors, and a subset $\mathcal{G} \subseteq \mathcal{L}' \setminus \mathcal{B}$ of $\log s + 1$ good factors Q_i such that $g := \prod_{i \in \mathcal{G}} Q_i$ and $s < \|g\| < s^2$. ■

4.3 Reconstructing set-multilinear $\Sigma\Pi\Sigma\Pi(2)$ circuits

Our final ingredient handles the reconstruction of the special case of set-multilinear $\Sigma\Pi\Sigma\Pi(2)$ circuits:

Lemma 14. *Let $f \in \mathbb{F}[X_n]$ be the polynomial computed by the following simple* set-multilinear $\Sigma\Pi\Sigma\Pi(2)$ *circuit of size s where P_i's, Q_i's are irreducible and $d > 2$:*

$$f(X_n) = T_1 + T_2 = \prod_{i=1}^{d} P_i(\bar{x}_i) + \prod_{i=1}^{d} Q_i(\bar{x}_i).$$

Then, given the partition $\{\bar{x}_i\}_{i \in [d]}$ and blackbox access to f, we can determine $\{P_i\}_{i \in [d]}, \{Q_i\}_{i \in [d]}$ in randomized time $\mathrm{poly}(n, s, c_{\max})$*, where $c_{\max}$ is the maximum bit length of any coefficient appearing in f.*

Proof. The algorithm is very similar to the one for reconstructing set-multilinear $\Sigma\Pi\Sigma(2)$ circuits in Lemma 9. Substitute all the variables in $X_n \backslash \bar{x}_1$ to independent random values over $\mathbb{F}$ (name this substitution $\bar{R}_1$) and interpolate to get the $2s$-sparse polynomial $P_1'(\bar{x}_1)$. Let $\bar{R}_2$ be another such independent substitution and $Q_1'(\bar{x}_1)$ be the one obtained after interpolation. Then,

$$P_1'(\bar{x}_1) = f(\bar{x}_1, \bar{R}_1) = \alpha P_1(\bar{x}_1) + \beta Q_1(\bar{x}_1) \quad , \quad Q_1'(\bar{x}_1) = f(\bar{x}_1, \bar{R}_2) = \gamma P_1(\bar{x}_1) + \delta Q_1(\bar{x}_1).$$

Clearly with probability $1 - O(n)/|\mathbb{F}|$ (recall that we assumed $|\mathbb{F}| > n^5$), $\alpha, \beta, \gamma, \delta \neq 0$. Also, from simplicity, the polynomials $\prod_{i=2}^{d} P_i$, $\prod_{i=2}^{d} Q_i$ are LI and hence the polynomial $\alpha \prod_{i=2}^{d} P_i - \beta \prod_{i=2}^{d} Q_i$ is nonzero. Now as $\bar{R}_1$, $\bar{R}_2$ are independent substitutions, w.h.p. $\alpha\delta - \beta\gamma = \alpha \prod_{i=2}^{d} P_i(\bar{R}_2) - \beta \prod_{i=2}^{d} Q_i(\bar{R}_2) \neq 0$. Now as $\left(\begin{smallmatrix} \alpha & \beta \\ \gamma & \delta \end{smallmatrix}\right)$ is invertible, $\exists p, q, r, s \neq 0$ such that

$$P_1(\bar{x}_1) = pP_1' + qQ_1' \quad , \quad Q_1(\bar{x}_1) = rP_1' + sQ_1'.$$

Also, using the irreducibility of P_1, Q_1 and an argument on $D_y(\frac{\partial P_1'}{\partial x}, P_1'|_{x=0})$ for all $x \neq y \in \bar{x}_1$, it can be easily shown that w.h.p. P_1' is irreducible (see Corollary 1). Repeating this for every $\bar{x}_i$ we have polynomials $P_1', \ldots, P_d', Q_1', \ldots, Q_d'$ such that $\forall i \in [d]$:

$$P_i(\bar{x}_i) = p_i P_i' + q_i Q_i' \quad , \quad Q_i(\bar{x}_i) = r_i P_i' + s_i Q_i'.$$

and hence, $f(X_n) = \prod_{i=1}^d (p_i P_i' + q_i Q_i') + \prod_{i=1}^d (r_i P_i' + s_i Q_i') = \alpha \prod_{i=1}^d (P_i' + a_i Q_i') + \beta \prod_{i=1}^d (P_i' + b_i Q_i')$, for some $\alpha, \beta \in \mathbb{F}$. As we already have the P_i''s and Q_i''s we just need to determine α, β, a_i's, b_i's. We first determine $a_1, a_2, a_3, b_1, b_2, b_3$. In f, substitute all the variables except $\bar{x}_1, \bar{x}_2, \bar{x}_3$ to independent random values over $\mathbb{F}$, to get $\hat{f}(\bar{x}_1, \bar{x}_2, \bar{x}_3)$. We have,

$$\hat{f} = \alpha'(P_1' + a_1 Q_1')(P_2' + a_2 Q_2')(P_3' + a_3 Q_3') + \beta'(P_1' + b_1 Q_1')(P_2' + b_2 Q_2')(P_3' + b_3 Q_3'). \tag{12}$$

where $a_i \neq b_i$ and w.h.p $\alpha', \beta' \neq 0$. Hence,

$$\hat{f} = P_3'(c_1.P_1'P_2' + c_2.Q_1'P_2' + c_3.P_1'Q_2' + c_4.Q_1'Q_2') + Q_3'(c_5.P_1'P_2' + c_6.Q_1'P_2' + c_7.P_1'Q_2' + c_8.Q_1'Q_2').$$

As described in the proof of Lemma 9, to determine the a_i's,b_i's, it is sufficient to determine the c_i's. W.l.o.g. we only show how to determine c_1 i.e. the coefficient of $P_1'(\bar{x}_1)P_2'(\bar{x}_2)P_3'(\bar{x}_3)$ in $\hat{f}$. First we construct a substitution $\bar{S}_1$ to the variables in $\bar{x}_1$ such that $Q_1'(\bar{S}_1) = 0$ but $P_1'(\bar{S}_1) \neq 0$. For some $x \in \mathsf{var}(Q_1')$, let $P_1'(\bar{x}_1) = A(\bar{x}_1 \backslash \{x\})x + B(\bar{x}_1 \backslash \{x\})$ and $Q_1'(\bar{x}_1) = C(\bar{x}_1 \backslash \{x\})x + D(\bar{x}_1 \backslash \{x\})$ where $C \not\equiv 0$. Substitute the variables in $\bar{x}_1 \backslash \{x\}$ to independent random field elements $\bar{S}_1'$ to get $P_1'(x, \bar{S}_1') = ax + b$ and $Q_1'(x, \bar{S}_1') = cx + d$ where w.h.p. $c \neq 0$. As P_1', Q_1' are LI and irreducible, the polynomial $AD - BC$ is non-zero and hence w.h.p. $ad - bc \neq 0$. Define the substitution $\bar{S}_1$ to be $\bar{S}_1'$ and $x = -d/c$. Clearly, $P_1'(\bar{S}_1) = bc - ad \neq 0$ and $Q_1'(\bar{S}_1) = 0$. Similarly, construct substitutions $\bar{S}_2, \bar{S}_3$ for the variables in $\bar{x}_2, \bar{x}_3$ such that $Q_2'(\bar{S}_2) = Q_3'(\bar{S}_3) = 0$ but $P_2'(\bar{S}_2), P_3'(\bar{S}_3)$ are non-zero. Hence, $\hat{f}(\bar{S}_1, \bar{S}_2, \bar{S}_3) = c_1.P_1'(\bar{S}_1)P_2'(\bar{S}_2)P_3'(\bar{S}_3)$ where we know P_i's explicitly and $P_1'(\bar{S}_1)P_2'(\bar{S}_2)P_3'(\bar{S}_3) \neq 0$. Hence we can determine c_1. Similarly, other c_i's can be determined.

Similarly, repeating this procedure for $P_1', P_2', P_i', Q_1', Q_2', Q_i'$ correctly and uniquely determines a_i's and b_i's. Having determined a_i's and b_i's, we can determine α, β by evaluating f at two independent, randomly chosen substitutions, as described earlier in the proof of Lemma 9. ■

4.4 Proof of Theorem 1

We now put together the ingredients from previous subsections to prove our main result (Theorem 1). We begin by presenting the algorithm `SPSPRECON` claimed in the theorem.

Algorithm : `SPSPRECON`$(n, s, \mathcal{O}_f)$
Input: oracle $\mathcal{O}_f$ for the polynomial $f \in \mathbb{F}[X_n]$ computable by a size-s multilinear $\Sigma\Pi\Sigma\Pi(2)$ circuit C as given by (3) with $||C|| > 4s^4$ and $d_1, d_2 \geq 3$.
Output: sets of s-sparse multilinear polynomials $\{G_1, \ldots, G_r\}, \{P_1, \ldots, P_{d_1}\}, \{Q_1, \ldots, Q_{d_2}\}$ s.t. equation (3) holds or `FAIL`.

---

1. *Obtaining G_i's and oracle to* $\mathrm{sim}(C)$: Using the algorithm of Lemma 2, obtain blackboxes for the irreducible factors of f. For each factor, test if it is s-sparse by first running the algorithm of Lemma 3 set for interpolating s-sparse polynomials and then using blackbox-PIT to test whether it has been interpolated correctly. Output the s-sparse factors as G_i's and define f_s to be the product of (the remaining) s-dense factors. To avoid introducing more notation, we assume below that $\mathsf{var}(f_s) = X_n$.

2. $\mathsf{var}(T_1) \neq \mathsf{var}(T_2)$ *case*: By iterating over $x \in X_n$ guess a variable x (if it exists) s.t. either $x \in \mathsf{var}(T_1)$ but $x \notin \mathsf{var}(T_2)$ or vice versa. Obtain the coefficient polynomial of x, factor it, and

test whether it is a product of s-sparse multilinear polynomials on disjoint sets of variables. If not, proceed to the next x, else, for every such factor H, test if $\texttt{SPSPFACTOR}(n, s, \mathcal{O}_{f_s}, H)$ outputs a multilinear $\Sigma\Pi\Sigma\Pi(2)$ circuit for f_s. If this succeeds for some $x \in X_n$, output the corresponding circuit f_s and the G_i's from Step 1 and STOP; else move to the next step.

3. $\mathsf{par}(T_1) \neq \mathsf{par}(T_2)$ *or* $\mathsf{par}(T_1) = \mathsf{par}(T_2) \neq \mathsf{par}(f_s)$ *case*: Iterate over pairs of distinct variables $x, y \in X_n$ and do: Obtain the blackbox for $\Delta_{xy}(f_s)$, factor it and interpolate its s-sparse multilinear factors. For every such factor H, test if $\texttt{SPSPFACTOR}(n, s, \mathcal{O}_{f_s}, H)$ outputs a multilinear $\Sigma\Pi\Sigma\Pi(2)$ circuit for f_s. If such a pair $x, y \in X_n$ could be found, output the corresponding circuit f_s and the G_i's from Step 1 and STOP; else move to the next step.

4. *Set-multilinear case*: Using blackboxes for the factors of f_s (obtained in the first step), determine $\mathsf{par}(f_s)$. Using the algorithm of Lemma 14 on f_s and $\mathsf{par}(f_s)$, determine a multilinear $\Sigma\Pi\Sigma\Pi(2)$ circuit for f_s.

Proof. Running time follows easily from the earlier analysis and hence we only prove correctness. We will show that the above algorithm succeeds in one of the steps based on the cases handled by those steps.

Let f be the multilinear polynomial computed by a multilinear $\Sigma\Pi\Sigma\Pi(2)$ circuit C as given below:

$$f(x_1, \ldots, x_n) = \gcd(C) \cdot \mathrm{sim}(C) = G \cdot (T_1 + T_2) = G_1 \cdot \ldots G_r \cdot \left(\prod_{i=1}^{d_1} P_i + \prod_{i=1}^{d_2} Q_i\right) \tag{13}$$

From Lemma 11, it follows that G_i's exactly comprise of the s-sparse factors of f and hence f_s in Step 1 is the polynomial computed by $\mathrm{sim}(C)$.

Suppose now that $\mathsf{var}(T_1) \neq \mathsf{var}(T_2)$. Assume w.l.o.g. that $\exists x \in \mathsf{var}(T_1) \backslash \mathsf{var}(T_2)$ and also w.l.o.g. let $P_1 = Rx + S$. It follows that the coefficient polynomial of x is $R \cdot P_2 \cdot \ldots P_{d_1}$ and hence all its factors are s-sparse polynomials on disjoint sets of variables. As $d_1, d_2 > 2$, one of those factors would be P_2 and hence $\texttt{SPSPFACTOR}$ from Lemma 12 would correctly output the P_i's and Q_i's. Since we iterate over all $x \in X_n$ in Step 2, such an x will be found if $\mathsf{var}(T_1) \neq \mathsf{var}(T_2)$ and we will output the correct $\Sigma\Pi\Sigma\Pi(2)$ for f_s.

Thus, the only possibility for Step 2 to not succeed is if $\mathsf{var}(T_1) = \mathsf{var}(T_2)$ and hence we consider this case now. Let us first suppose that $\mathsf{par}(T_1) \neq \mathsf{par}(T_2)$. Then we must have x, y such that w.l.o.g. x, y occur in distinct P_i, P_j respectively but both occur in the same Q_k. Say $P_1 = Rx + S$, $P_2 = Uy + W$ but $Q_1 = Axy + Bx + Cy + D$. Hence, $f_s = (Rx+S)(Uy+W) \prod_{i=3}^{d_1} P_i + (Axy + Bx + Cy + D) \prod_{i=2}^{d_2} Q_i$. Note now that

$$\frac{\partial f_s}{\partial x} = R(Uy + W) \prod_{i=3}^{d_1} P_i + (Ay + B) \prod_{i=2}^{d_2} Q_i \quad \text{and} \quad f_s|_{x=0} = S(Uy + W) \prod_{i=3}^{d_1} P_i + (Cy + D) \prod_{i=2}^{d_2} Q_i.$$

Thus, we have

$$\Delta_{xy}(f_s) = \prod_{i=2}^{d_2} Q_i \cdot \left[[R(UD - WC) + S(AW - BU)] \prod_{i=3}^{d_1} P_i + (AD - BC) \prod_{i=2}^{d_2} Q_i \right].$$

Claim 1. *$\Delta_{xy}(f_s)$ is nonzero.*

Proof of Claim 1: Since $\gcd\left(\prod_{i=3}^{d_1} P_i, \prod_{i=2}^{d_2} Q_i\right) = 1$ and $AD - BC \not\equiv 0$ by irreducibility of Q_1, we have

$$\Delta_{xy}(f_s) \equiv 0 \Longrightarrow \prod_{i=3}^{d_1} P_i \Big| AD - BC \quad \text{and} \quad \prod_{i=2}^{d_2} Q_i \Big| R(UD - WC) + S(AW - BU) \not\equiv 0.$$

But since $||AD-BC|| \leq 2s^2$ and degree of every variable in it is at most 2, from Lemma 10, $||\prod_{i=3}^{d_1} P_i|| \leq 2s^2$. Similarly, $||\prod_{i=2}^{d_2} Q_i|| \leq 4s^2$. As $||C|| > 4s^4$, either $||T_1|| > 4s^4$ or $||T_2|| > 4s^4$. In the former, as $||P_1P_2|| \leq s^2$ we have $||\prod_{i=3}^{d_1} P_i|| > 4s^2$ and in the later $||\prod_{i=2}^{d_2} Q_i|| > 4s^3$. We get a contradiction in both cases, proving the claim. □

Since $\prod_{i=2}^{d_2} Q_i \mid \Delta_{xy}(f_s)$, and $d_1, d_2 > 2$, Step 3 would be able to find a Q_i as one of its s-sparse multilinear factors and hence SPSPFACTOR would succeed. We conclude that Step 3 succeeds in the case that $\mathsf{var}(T_1) = \mathsf{var}(T_2)$ but $\mathsf{par}(T_1) \neq \mathsf{par}(T_2)$.

We next consider the case that both $\mathsf{var}(T_1) = \mathsf{var}(T_2)$ and $\mathsf{par}(T_1) = \mathsf{par}(T_2)$, but neither of these partitions is the partition $\mathsf{par}(f_s)$ for f_s. Let $f_s = f_1(\bar{x}_1) \cdot \ldots f_d(\bar{x}_d)$ be the factorization of f_s, where f_i's are irreducible. Returning to the case of Step 3 when $\mathsf{par}(T_1) = \mathsf{par}(T_2) \neq \mathsf{par}(f_s)$, we observe that $\mathsf{par}(f_s)$ can only be coarser than $\mathsf{par}(T_1) = \mathsf{par}(T_2)$. Specifically,

Claim 2. *If a pair of variables x, y occurs in different sets of $\mathsf{par}(f_s)$ then this pair must also occur in different sets in $\mathsf{par}(T_1) = \mathsf{par}(T_2)$*

Proof of Claim 2: Suppose not. Then we have x, y occurring in different sets of $\mathsf{par}(f_s)$ but in the same set of $\mathsf{par}(T_1)$. Then, by Corollary 1, $\Delta_{xy}(f_s) \equiv 0$. Now, let $P_1 = Rxy + Sx + Uy + W$ and $Q_1 = Axy + Bx + Cy + D$ so that $f_s = (Rxy + Sx + Uy + W)\prod_{i=2}^{d_1} P_i + (Axy + Bx + Cy + D)\prod_{i=2}^{d_2} Q_i$. We then have,

$$0 \equiv \Delta_{xy}(f_s) = \prod_{i=2}^{d_1} P_i \left[(RW - SU)\prod_{i=2}^{d_1} P_i + (RD + AW - SC - BU)\prod_{i=2}^{d_2} Q_i\right] + (AD - BC)\left(\prod_{i=2}^{d_2} Q_i\right)^2.$$

By the pairwise coprimality of the P_i's and the Q_i's we get

$$\prod_{i=2}^{d_1} P_i \Big| (AD - BC) \quad \text{and} \quad \prod_{i=2}^{d_2} Q_i \Big| (RW - SU) \quad \text{and} \quad \prod_{i=2}^{d_2} Q_i \Big| (RD + AW - SC - BU).$$

But since $||AD - BC||, ||RW - SU||, ||RD + AW - SC - BU|| \leq 4s^2$ and the degree of any variable in these is at most 2, by Lemma 10 and the assumption that $||C|| \geq 4s^2$, we arrive at a contradiction, proving the claim. □

Thus $\mathsf{par}(T_1) = \mathsf{par}(T_2) \neq \mathsf{par}(f_s)$ means that there exists a pair x, y of variables s.t. x, y occur in different sets of $\mathsf{par}(T_1)$ but in the same set of $\mathsf{par}(f_s)$. By corollary 1, we then have $\Delta_{xy}(f_s) \not\equiv 0$. Let us now say $P_1 = Rx + S$, $P_2 = Uy + W$, $Q_1 = Ax + B$, and $Q_2 = Cy + D$ so that $f_s = (Rx + S)(Uy + W)\prod_{i=3}^{d_1} P_i + (Ax + B)(Cy + D)\prod_{i=3}^{d_2} Q_i$. Therefore,

$$\frac{\partial f_s}{\partial x} = R(Uy + W)\prod_{i=3}^{d_1} P_i + A(Cy + D)\prod_{i=3}^{d_2} Q_i \text{ and } f_s|_{x=0} = S(Uy + W)\prod_{i=3}^{d_1} P_i + B(Cy + D)\prod_{i=3}^{d_2} Q_i.$$

Altogether we then have,

$$0 \not\equiv \Delta_{xy}(f_s) = (RUBD + ACSW - RWBC - ADSU)\prod_{i=3}^{d_1} P_i \cdot \prod_{i=3}^{d_2} Q_i.$$

It follows we will be able to catch one of the P_i or Q_i for $i \geq 3$, among the factors of $\Delta_{xy}(f_s)$ in Step 3 and succeed in reconstructing the circuit for f_s. Hence, we are finally left with the case that $\mathsf{par}(T_1) = \mathsf{par}(T_2) = \mathsf{par}(f_s)$. Clearly, this partition can be determined by factoring f_s and hence the algorithm for the set-multilinear case given by Lemma 14 would succeed. ■

5 Discussion and Open Problems

The problem of reconstructing polynomials from arithmetic complexity classes is, in a broad sense, analogous to learning concept classes of Boolean functions using membership and equivalence queries (cf. Chapter 5 of [SY10]). Over the last several decades, research on the theory of learnability in the Boolean world has evolved into a mature discipline. However, circuit reconstruction in the arithmetic world has been gaining momentum only in the recent years. Because of connections to major challenges such as Polynomial Identity Testing (PIT) and explicit lower bounds, progress on reconstruction has largely been limited to restricted models of computation. In this paper, we presented a randomized reconstruction algorithm for $\Sigma\Pi\Sigma\Pi(2)$ multilinear circuits. Several open problems remain in this area, some of which are listed below.

- **Handle larger (constant) top fan-in:** It would be interesting to generalize our results to multilinear $\Sigma\Pi\Sigma\Pi(k)$ circuits, with $k = O(1)$. Handling non-constant top fan-in for depth-4 multilinear circuits appears to be a more serious challenge. We do not "even" know PIT algorithms for such circuits (cf. Saraf and Volkovich [SV11] for constant k).
- **Remove dependence on field size for $\Sigma\Pi\Sigma(k)$ (no multilinear restriction) circuits:** The running time of the algorithm by Karnin & Shpilka, for reconstructing $\Sigma\Pi\Sigma(k)$ circuits with $k = O(1)$, has a dependence on the field size. An interesting problem is to remove this dependence, even for the $\Sigma\Pi\Sigma(2)$ case.

In addition to considering weak models of computation, another research direction to make progress in on the reconstruction problem is to consider its *distributional complexity* on random (according to that distribution) instances, but computable by *more powerful models*. Very recently, we [GKL11] were able to make progress in this direction for "random" multilinear formulas i.e. formulas sampled from a certain natural distribution over the set of multilinear formulas.

- **Average case reconstruction of random arithmetic formulas:** Is it possible to efficiently reconstruct random (general) arithmetic formulas? This would be very surprising as no non-trivial size lower bound is known for general formulas and the techniques might even lead to non-trivial lower bounds for some interesting sub-class of formulas.
- **Reconstruction for random depth-3 circuits:** In [GK98, GR98], exponential lower bounds for depth-3 circuits over finite fields were proved. Consider a "random" $\Sigma\Pi\Sigma(k)$ circuit over a finite field $\mathbb{F}$ to be one in which the affine forms computed at the bottom Σ layer have their coefficients chosen randomly from $\mathbb{F}$. Is there an algorithm to reconstruct such a circuit w.h.p. over the choice of the coefficients with running time efficient in n, k?

References

[Agr05] Manindra Agrawal. Proving lower bounds via pseudo-random generators. In *FSTTCS*, pages 92–105, 2005.

[AL86] Leonard M. Adleman and Hendrik W. Lenstra, Jr. Finding irreducible polynomials over finite fields. In *STOC*, pages 350–355, 1986.

[AMS10] Vikraman Arvind, Partha Mukhopadhyay, and Srikanth Srinivasan. New results on non-commutative and commutative polynomial identity testing. *Computational Complexity*, 19(4):521–558, 2010.

[AV08] Manindra Agrawal and V. Vinay. Arithmetic circuits: A chasm at depth four. In *FOCS*, pages 67–75, 2008.

[BBB$^+$00] Amos Beimel, Francesco Bergadano, Nader H. Bshouty, Eyal Kushilevitz, and Stefano Varricchio. Learning functions represented as multiplicity automata. *J. ACM*, 47(3):506–530, 2000.

[CLO97] David A. Cox, John Little, and Donal O'Shea. *Ideals, varieties, and algorithms - an introduction to computational algebraic geometry and commutative algebra*. Springer, second edition, 1997.

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

[GK98] Dima Grigoriev and Marek Karpinski. An exponential lower bound for depth 3 arithmetic circuits. In *STOC*, pages 577–582, 1998.

[GKL11] Ankit Gupta, Neeraj Kayal, and Satya Lokam. Efficient reconstruction of random multilinear formulas. In *FOCS*, pages 778–787, 2011.

[GR98] Dima Grigoriev and Alexander A. Razborov. Exponential complexity lower bounds for depth 3 arithmetic circuits in algebras of functions over finite fields. In *FOCS*, pages 269–278, 1998.

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

[HS80] Joos Heintz and Claus-Peter Schnorr. Testing polynomials which are easy to compute (extended abstract). In *STOC*, pages 262–272, 1980.

[Kal89] Erich Kaltofen. Factorization of polynomials given by straight-line programs. In *Randomness and Computation*, pages 375–412. JAI Press, 1989.

[Kay11] Neeraj Kayal. Affine projections of polynomials. *Electronic Colloquium on Computational Complexity (ECCC)*, 18:61, 2011.

[KMSV10] Zohar Shay Karnin, Partha Mukhopadhyay, Amir Shpilka, and Ilya Volkovich. Deterministic identity testing of depth-4 multilinear circuits with bounded top fan-in. In *STOC*, pages 649–658, 2010.

[KS01] Adam R. Klivans and Daniel A. Spielman. Randomness efficient identity testing of multivariate polynomials. In *STOC*, pages 216–223, 2001.

[KS06] Adam R. Klivans and Amir Shpilka. Learning restricted models of arithmetic circuits. *Theory of Computing*, 2(1):185–206, 2006.

[KS09] Zohar Shay Karnin and Amir Shpilka. Reconstruction of generalized depth-3 arithmetic circuits with bounded top fan-in. In *IEEE Conference on Computational Complexity*, pages 274–285, 2009.

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

[Raz09] Ran Raz. Multi-linear formulas for permanent and determinant are of super-polynomial size. *J. ACM*, 56(2), 2009.

[Raz10] Ran Raz. Elusive functions and lower bounds for arithmetic circuits. *Theory of Computing*, 6(1):135–177, 2010. Preliminary version in STOC 2008, pp. 711 - 720.

[RS05] Ran Raz and Amir Shpilka. Deterministic polynomial identity testing in non-commutative models. *Computational Complexity*, 14(1):1–19, 2005.

[RSY08] Ran Raz, Amir Shpilka, and Amir Yehudayoff. A lower bound for the size of syntactically multilinear arithmetic circuits. *SIAM J. Comput.*, 38(4):1624–1647, 2008.

[RY09] Ran Raz and Amir Yehudayoff. Lower bounds and separations for constant depth multilinear circuits. *Computational Complexity*, 18(2):171–207, 2009.

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

[Shp09] Amir Shpilka. Interpolation of depth-3 arithmetic circuits with two multiplication gates. *SIAM J. Comput.*, 38(6):2130–2161, 2009.

[SV09] Amir Shpilka and Ilya Volkovich. Improved polynomial identity testing for read-once formulas. In *APPROX-RANDOM*, pages 700–713, 2009.

[SV11] Shubhangi Saraf and Ilya Volkovich. Black-box identity testing of depth-4 multilinear circuits. In *STOC*, pages 421–430, 2011.

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

[Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. In Edward W. Ng, editor, *EUROSAM*, volume 72 of *Lecture Notes in Computer Science*, pages 216–226. Springer, 1979.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de