Black-box reconstruction of depth three circuits with top fan-in two

Thesis by

Gaurav Sinha

In Partial Fulfillment of the Requirements for the
degree of
Ph.D.

CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2016
Defended May 25, 2016

Gaurav Sinha
ORCID: orcid.org/0000-0002-3590-9543

ACKNOWLEDGEMENTS

I have been very lucky to receive guidance, advice, friendship, love and support from a lot of people over the course of my Ph.D.. I will try my best to thank some of them here and would also like to apologize to the people I miss.

First of all, I wish to thank my advisor Prof. Eric Rains for being a wonderful mentor. He was always open and willing to discussions and has played a major role in the way I approach problems. It was very thrilling to see him connect different areas and suggest completely fresh approaches whenever I was stuck.

My sincere thanks to Prof. Leonard Schulman who is a member of my thesis committee. Even though my interaction with him happened towards the end of my Ph.D., we spent a lot of time going over my thesis and discussing all ideas in detail. His suggestions have helped me improve my presentation as well as provided me new insights into improving the result. Some of the questions he asked me have become a part of my future research goals.

I'm grateful to Prof. Chris Umans, who has been both a great research role model and an amazing teacher to me. Before coming to caltech I had some initial introduction to his approach towards matrix multiplication algorithms and was very fascinated by it. I met him several times during my Ph.D. to discuss various aspects of the problems I worked upon.

I would like to thank Prof. Nets Katz who has been a great inspiration to me. During my second year, I did a reading course with him on the Kakeya conjecture to which he himself has contributed a lot. His class on special topics in analysis is one of my favorite classes at CalTech.

I am extremely thankful to Neeraj Kayal for introducing me to this problem. Sukhada Fadnavis, Neeraj Kayal and myself started working on the problem together during my summer internship at Microsoft Research India Labs in 2011. We solved the first important case together. I'm grateful to them for all helpful discussions, constant guidance and encouragement.

I would like to thank my friends at CalTech. Without them CalTech would not have been such an enjoyable experience. My discussions about science, politics, movies and everything else with Vikas and Vinamra are some of my most fond memories. Vikas has always been very motivating and helpful during the ups and downs of my

stay. I miss our walks around campus. I would also like to thank Prachi for helping me organize this thesis. Special thanks to all my friends including Utkarsh, Sisir, Karan, Manpreet.

Last but not the least, I would like to thank my parents, brother and sister. It is their hard work, love and support that has helped me become a better researcher and a better person. Their unconditional love is what keeps me going and I hope to make all of them proud.

ABSTRACT

Reconstruction of arithmetic circuits has been heavily studied in the past few years and has connections to proving lower bounds and deterministic identity testing. In this thesis we present a polynomial time randomized algorithm for reconstructing $\Sigma\Pi\Sigma(2)$ circuits over characteristic zero fields $\mathbb{F}$ i.e. depth-3 circuits with fan-in 2 at the top addition gate and having coefficients from a field of characteristic zero.

The algorithm needs only a black-box query access to the polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ of degree d, computable by a $\Sigma\Pi\Sigma(2)$ circuit C. In addition, we assume that the *"simple rank"* of this polynomial (essential number of variables after removing the gcd of the two multiplication gates) is bigger than a fixed constant. Our algorithm runs in time polynomial in n and d and with high probability returns an equivalent $\Sigma\Pi\Sigma(2)$ circuit.

The problem of reconstructing $\Sigma\Pi\Sigma(2)$ circuits over finite fields was first proposed by Shpilka [27]. The generalization to $\Sigma\Pi\Sigma(k)$ circuits, $k = O(1)$ (over finite fields) was addressed by Karnin and Shpilka in [18]. The techniques in these previous involve iterating over all objects of certain kinds over the ambient field and thus the running time depends on the size of the field $\mathbb{F}$. Their reconstruction algorithm uses lower bounds on the lengths of linear locally decodable codes with 2 queries. In our setting, such ideas immediately pose a problem and we need new techniques.

Our main techniques are based on the use of quantitative Sylvester Gallai theorems from the work of Barak et.al. [3] to find a small collection of *"nice"* subspaces to project onto. The heart of this work lies in subtle applications of the quantitative Sylvester Gallai theorems to prove why projections w.r.t. the *"nice"* subspaces can be ”glued”. We also use Brill’s equations from [9] to construct a small set of candidate linear forms (containing linear forms from both gates). Another important technique which comes very handy is the polynomial time randomized algorithm for factoring multivariate polynomials given by Kaltofen [17].

PUBLISHED MATERIAL IN THIS THESIS

[1] Gaurav Sinha. "Reconstruction of Real Depth-3 Circuits with Top Fan-In 2". In: *31st Conference on Computational Complexity (CCC 2016)*. Ed. by Ran Raz. Vol. 50. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016, 31:1–31:53. ISBN: 978-3-95977-008-8. DOI: http://dx.doi.org/10.4230/LIPIcs.CCC.2016.31. URL: http://drops.dagstuhl.de/opus/volltexte/2016/5854.

TABLE OF CONTENTS

Acknowledgements . . . iii
Abstract . . . v
Table of Contents . . . vii
Chapter I: Introduction . . . 1
1.1 Previous Work and Connections . . . 4
1.2 Allowing Randomization . . . 4
1.3 Preliminaries . . . 5
1.4 A quick introduction to arithmetic circuits . . . 9
1.5 Homogenization of $\Sigma\Pi\Sigma(2)$ circuits . . . 12
Chapter II: Some Definitions, Main Tools and Techniques . . . 14
2.1 Introduction . . . 14
2.2 Uniqueness of $\Sigma\Pi\Sigma(2)$ Structure . . . 16
2.3 Factoring forms of a polynomial . . . 17
2.4 Good forms and reconstructed multi-set . . . 20
Chapter III: Main result and overview . . . 24
3.1 Overview of the algorithm . . . 24
Chapter IV: Step One : Reconstruct the Ist Layer of C . . . 28
4.1 Introduction . . . 28
4.2 Random Transformation . . . 30
4.3 Restricting the input polynomial . . . 32
4.4 Computing the sets $\mathcal{P}_i$. . . . 33
4.5 Gluing $\mathcal{P}_i$'s to compute $\mathcal{P}$ . . . 35
Chapter V: Step Two : Reconstruct Layer II of C . . . 39
5.1 Introduction . . . 39
5.2 Lines connecting forms in $\mathbb{P}(R)$ ($\mathbb{P}(B)$ resp.) to $\mathbb{P}(M_2)$($\mathbb{P}(M_1)$ resp.) 40
5.3 Termination Case : Reconstructing one of $\mathbb{P}(M_1), \mathbb{P}(M_2)$ does the job 42
5.4 One of the multi-sets $\mathbb{P}(R), \mathbb{P}(B)$ is low dimensional . . . 45
5.5 Both multi-sets $\mathbb{P}(R), \mathbb{P}(B)$ are high dimensional . . . 48
Bibliography . . . 56
Appendix A: Random Transformation and Restrictions . . . 60
Appendix B: Brill's Equations - Characterizing polynomials which are product of linear forms . . . 64
Appendix C: Black-box Factoring of Polynomials . . . 67
Published Content and Contributions . . . 70
Appendix D: Proofs from Chapter IV . . . 72
Appendix E: Proofs from Chapter V . . . 74
Appendix F: Tools from Incidence Geometry . . . 78
Appendix G: Consent Form . . . 80

Chapter 1

INTRODUCTION

Recall the interpolation problem which requires finding coefficients of a polynomial such that a number (possibly large) of point evaluations of the polynomial have been given.

Recall 1 (Interpolation Problem)**.** Let $\Lambda = \{\lambda = (\lambda_1, \ldots, \lambda_n) : \lambda_1 + \ldots + \lambda_n \leq d\}$ be an indexing set. For any polynomial $f(\bar{x}) \in \mathbb{F}[\bar{x}]$ of degree d in variables $\bar{x} = (x_1, \ldots, x_n)$ over the field $\mathbb{F}$, we wish to compute coefficients c_λ such that

$$f(\bar{x}) = \sum_{\lambda \in \Lambda} c_\lambda \mathbf{x}^\lambda$$

where $\mathbf{x}^\lambda$ denotes the monomial $x_1^{\lambda_1} \ldots x_n^{\lambda_n}$.

We would want to solve this problem with as few evaluations as possible. Thankfully there are tight lower bounds which are also easy to prove. For example to interpolate a univariate polynomial of degree d we need at least $d + 1$ evaluations (folklore). One would also want to develop general algorithms to actually perform the task of computing the coefficients. The method of Lagrange interpolation is one such popular algorithm (see chapter 3 in [15]).

We consider a more general setup. Suppose the given polynomial has a special representation. Can we develop algorithms to reconstruct the polynomial in the desired representation. For example, if our polynomial is a product of linear forms we might want to compute these linear forms by just using evaluations at some set of points.

Arithmetic circuits are the most natural choice when one wants to talk about representations of polynomials. In the language of arithmetic circuits, the interpolation problem translates to finding an appropriate (generally most succinct) circuit by just using evaluations of the polynomial. In the reverse direction, they also provide efficient ways to evaluate the polynomial.

Informally, an arithmetic circuit is a weighted directed acyclic graph whose leaves will denote variables and constants, internal nodes compute either the product or linear combinations of their children and the root node(s) compute the required

polynomial. Weight of every edge is an element of the field and gets multiplied to the output of the source vertex for the edge. We give the formal definition of arithmetic circuits in section 1.4.

The last few years have seen significant progress towards interesting problems dealing with arithmetic circuits. Some of these problems include deterministic polynomial identity testing, reconstruction of circuits and recently lower bounds for arithmetic circuits. There has also been work connecting these three different aspects.

In this thesis, we will primarily be concerned with the reconstruction problem. Even though it's connections to identity testing and lower bounds are very exciting, the problem in itself has drawn a lot of attention because of elegant techniques and connections to learning theory.

The strongest version of the problem requires that for any $f \in \mathbb{F}[x_1, \ldots, x_n]$ with black-box(query) access given one wants to construct (roughly) most succinct representation i.e. the smallest possible arithmetic circuit computing the polynomial. This general problem appears to be very hard. Most of the work done has dealt with some special type of polynomials i.e. the ones which exhibit constant depth circuits with alternating addition and multiplication gates.

Our result adds to this by looking at polynomials computed by circuits of this type (alternating addition/multiplication gates but of depth 3). Our circuits will have variables at the leaves, operations $(+, \times)$ at the internal gates and scalars at the edges. We also assume that the top gate(root) has only two children and the *"simple rank"* of this polynomial (essential number of variables after removing gcd of the two multiplication gates at the middle layer) is bigger than a constant. The bottom most layer has addition gates and so computes linear forms, the middle layer then multiplies these linear forms together and the top layer adds two such products.

In this work, we assume only homogeneous computation, that is all polynomials computed at all internal nodes will be homogeneous polynomials. However, we would also like to remark that we can simulate a depth 3 inhomogeneous polynomial by another depth 3 homogeneous polynomial. For reconstruction purposes, there is no difference between the two, and so we can assume we are reconstructing a homogeneous polynomial. We discuss this in more detail in section 1.5.

Given homogeneity, our circuit computes a polynomial of the following form :

$$C(\bar{x}) = M_1(\bar{x}) + M_2(\bar{x})$$

Here, M_1 and M_2 are products of equal number of linear forms. Note that we can further factorize and pull out the gcd of M_1 and M_2.

$$C(\bar{x}) = Gcd(C)(R + B)$$

where $gcd(R, B) = 1$.

Our condition about the essential number of variables (after removing gcd from the multiplication gates) is called *"simple rank"* of the polynomial and is defined as

$$srank(C) = dim(sp\{ \text{ linear forms } l \text{ dividing } R, B\})$$

When the underlying field $\mathbb{F}$ is of characteristic zero ($\mathbb{Q}, \mathbb{R}$ or $\mathbb{C}$ for simplicity), we give an efficient randomized algorithm for reconstructing the circuit representation of such polynomials i.e. finding the two polynomials M_1, M_2. Formally our main theorem reads :

Theorem 1. *Let C be a homogeneous $\Sigma\Pi\Sigma(2)$ circuit computing a degree d polynomial $C(\bar{x})$ in n variables $x_1, \ldots, x_n$. Assume that black-box access to C has been given (along with parameters n, d). We give a randomized algorithm that runs in time $poly(n, d)$ and with probability $1 - o(1)$ outputs the following:*

- *When $srank(C) = \Omega(1)$, the output is a $\Sigma\Pi\Sigma(2)$ circuit computing $C(\bar{x})$.*

As per our knowledge this is the first algorithm that efficiently reconstructs such circuits (over characteristic zero fields). Over finite fields, the same problem has been considered by Shpilka in [27] and our method takes inspiration from their work. They also generalized this finite field version to circuits with arbitrary (but constant) top fan-in in [18]. However we need many new tools and techniques as their methods don't generalize at a lot of crucial steps. For eg:

- They iterate through linear forms in a finite field which we unfortunately cannot do.
- They use lower bounds for locally decodable codes given in [8] which again does not work in our setup.

We resolve these issues by

- Constructing candidate linear forms by solving simultaneous polynomial equations obtained from brill's equations (chapter 4, [9]).
- Using quantitative versions of the Sylvester Gallai theorems given in [3] and [6]. This new method enables us to construct *nice* subspaces, take projections onto them and glue the projections back to recover the circuit representation.

1.1 Previous Work and Connections

Efficient reconstruction algorithms are known for some concrete class of circuits. We list some here:

- Depth 2 $\Sigma\Pi$ circuits (sparse polynomials) in [21]
- Read-once arithmetic formulas in [28]
- Non-commutative ABP's [2]
- $\Sigma\Pi\Sigma(2)$ circuits over finite fields in [27], extended to *"generalized"* $\Sigma\Pi\Sigma(k)$ circuits (over finite fields) with $k = O(1)$ in [18].
- Random multi-linear formulas in [12]
- Depth 4 ($\Sigma\Pi\Sigma\Pi$) multi-linear circuits with top fan-in 2 in [11]
- Random arithmetic formulas in [13]

All of the above work introduced new ideas and techniques and have been greatly appreciated.

1.2 Allowing Randomization

It's easy to observe that a polynomial time deterministic reconstruction algorithm for a circuit class C also implies a polynomial time deterministic identity testing algorithm for the same class. The idea is pretty simple. If the reconstruction algorithm outputs any non-trivial circuit then we can claim that the polynomial is non-zero otherwise we say it is zero. Here is a way to see this. If the polynomial is identically zero all our queries to the black-box give zero and so we cannot reconstruct anything non-trivial. Conversely for a non-zero polynomial, the reconstruction algorithm sees at least one non zero evaluation. If not then it cannot reconstruct a correct circuit since we get no non-trivial information from evaluations.

From the works [1] and [14] it has been established that black-box identity testing for certain circuit classes imply super-polynomial circuit lower bounds for an explicit polynomial. Hence the general problem of deterministic reconstruction cannot be easier than proving super-polynomial lower bounds. So one might first try and relax the requirements and demand a randomized algorithm.

Another motivation to consider the probabilistic version comes from learning theory. A fundamental question called the *exact learning problem using membership queries* asks the following : **G**iven oracle access to a boolean function, compute a small description for it. This problem has attracted a lot of attention in the last few decades. For e.g. in [20][10] and [19] a negative result stating that a class of boolean circuits containing the trapdoor functions or pseudo-random functions has no efficient learning algorithms. Among positive works [26], [4], [22] show that when f has a small circuit (inside some restricted class) exact learning from membership queries is possible.

Our problem is a close cousin as we are looking for exact learning algorithms for arithmetic functions. Because of these connection with learning theory it makes sense to also allow randomized algorithms for reconstruction.

1.3 Preliminaries

$[n]$ denotes the set $\{1, 2, \ldots, n\}$. Throughout the paper we will work over a field $\mathbb{F}$ of characteristic zero. The reader may assume it to be $\mathbb{Q}, \mathbb{R}$ or $\mathbb{C}$ for convenience.

Let V be a finite dimensional $\mathbb{F}$ vector space and $S \subset V$, $sp(S)$ will denote the linear span of elements of S. $dim(S)$ is the dimension of the subspace $sp(S)$.

$(\bar{x})$ will be used for the tuple $(x_1, \ldots, x_n)$.

For any set of polynomials $S \subset \mathbb{F}[\bar{x}]$, we denote by $\mathbb{V}(S)$, the set of all complex simultaneous solutions of polynomials in S (this set is called the variety of S), i.e.

$$\mathbb{V}(S) = \{a \in \mathbb{C} : \text{ for all } f \in S, f(a) = 0\}$$

LI will be the abbreviation for linearly independent and **L**D will be the abbreviation for linearly dependent.

Notations - Projective spaces of linear forms

Let V denote the vector space of linear forms in variables $\bar{x}$, and $\mathbb{P}(V)$ be the corresponding projective space. We will see a number of definitions, observations and

lemmas below which will be used throughout the thesis.

Definition 1 (Projectivization of a linear form)**.** For a non-zero linear form l we denote it's projectivization as $[l] \in \mathbb{P}(V)$ which can be viewed as the set

$$[l] = sp(l) \setminus \{0\}.$$

So it is the one dimensional vector space generated l without the vector 0 (i.e. the additive identity of V). This will also be our working definition for points in the projective space i.e. for every point p in the projective space there exists l such that $p = [l]$. Also if l, l' are non-zero scalar multiples of each other then $[l] = [l']$.

Definition 2 (Projective linear forms)**.** Points in the projective space $\mathbb{P}(V)$ (corresponding to the vector space of linear forms V) will be called *projective linear forms*.

Note 1 (Sub projective spaces)**.** Let $W \subset V$ be a subspace and $\mathbb{P}(W)$ be the projective space of W then

$$\mathbb{P}(W) \subset \mathbb{P}(V).$$

We will often call $\mathbb{P}(W)$ as a sub projective space of $\mathbb{P}(V)$.

Definition 3 (Dependent and independent sets in $\mathbb{P}(V)$)**.** Consider points $p_1, \ldots, p_k \in \mathbb{P}(V)$. The points are called *dependent* if for any set of linearly dependent linear forms $l_1, \ldots, l_k \in V$ we have $p_i = [l_i], i \in [k]$. It can be checked that this definition is *"well defined"* i.e. if $p_1, \ldots, p_k$ are dependent then for all $l'_i, i \in [k]$ such that $p_i = [l'_i]$ the linear forms $l'_1, \ldots, l'_k$ are linearly dependent.

If the points are not dependent, they will be called *independent*.

Definition 4 (Dimension of a multi-set in $\mathbb{P}(V)$)**.** Let P be a finite multi-set of forms in $\mathbb{P}(V)$. The dimension of P denoted as $dim(P)$ is the maximum number of independent forms in P. It can be checked that this notion is well defined.

Definition 5 (Flat defined by an independent set)**.** Let $p_1, \ldots, p_k \in \mathbb{P}(V)$ be an independent set of points in the projective space. Let $l_1, \ldots, l_k$ be linear forms such that $p_i = [l_i], i \in [k]$. We define the flat spanned by $p_1, \ldots, p_k$ as

$$fl(p_1, \ldots, p_k) = \{[l] \in \mathbb{P}(V) : l = \alpha_1 l_1 + \ldots \alpha_k l_k : \text{ not all } \alpha_i \text{ zero } \}.$$

It can be checked that this is well defined i.e. if we started with l'_i instead of l_i, we get the same set.

When we have just two independent points p_1, p_2 we will call the flat a line. For three independent points we call the flat they define a plane and so on.

Definition 6 (Basis)**.** Let P be a finite multi-set of points in $\mathbb{P}(V)$. An independent subset $\{p_1, \ldots, p_k\}$ is called a *basis* for P if every $p \in P$ belongs to $fl(p_1, \ldots, p_k)$.

Lemma 1. *Let P be a finite multi-set in $\mathbb{P}(V)$. The following holds:*

$$\{p_1, \ldots, p_k\} \textit{ is a basis } \Leftrightarrow \{p_1, \ldots, p_k\} \textit{ is a maximal independent set in } P.$$

Proof. Fix $l_i \in V$ such that $p_i = [l_i]$.

For $\Rightarrow$, assume $\{p_1, \ldots, p_k\}$ is a basis. So by definition they are independent. Pick any $p_{k+1} \in P$. By definition of a basis we know that $p_{k+1} \in fl(p_1, \ldots, p_k)$ i.e. there exists $l_{k+1} \in sp(l_1, \ldots, l_k) \setminus \{0\}$ such that $p_{k+1} = [l_{k+1}]$. Clearly $l_1, \ldots, l_k, l_{k+1}$ are linearly dependent and thus $p_1, \ldots, p_k, p_{k+1}$ are dependent implying that $p_1, \ldots, p_k$ is a maximal independent set.

For $\Leftarrow$ assume $p_1, \ldots, p_k$ are a maximal independent set in P. Assume there exists $p \in P$ such that $p \notin fl(p_1, \ldots, p_k)$. Let $l \in V$ such that $p = [l]$. We claim that $p_1, \ldots, p_k, p$ are independent. If not then $l_1, \ldots, l_k, l$ are linearly dependent. so there exists $\alpha_1, \ldots, \alpha_k, \alpha$ (not all zero) such that

$$\alpha_1 l_1 + \ldots + \alpha_k l_k + \alpha l = 0.$$

If $\alpha = 0$ then $l_1, \ldots, l_k$ become linearly dependent which is not possible. Thus $\alpha \neq 0$. At least one of the $\alpha_i \neq 0$ otherwise $l = 0$. This implies that $p = [l] \in fl(p_1, \ldots, p_k)$ a contradiction to our assumption. Therefore there is no such p and all $p \in P$ belong to $fl(p_1, \ldots, p_k)$.

□

Definition 7 (Flat defined by a finite multi-set in $\mathbb{P}(V)$)**.** Let $\mathcal{S} \subset \mathbb{P}(V)$ be a finite multi-set. The flat defined by $\mathcal{S}$ denoted by $fl(\mathcal{S})$ will be $fl(b_1, \ldots, b_k)$ where $b_1, \ldots, b_k$ is a basis in $\mathcal{S}$.

Definition 8 (Kernel of a projective linear forms)**.** Let $p \in \mathbb{P}(V)$ and $l = \alpha_1 x_1 + \ldots \alpha_n x_n \in V$ be such that $p = [l]$. Further assume that $i \in [n]$ is such that $\alpha_i \neq 0$ and $\alpha_j = 0$, for all $j < i$. Define $ker(p) = \{(x_1, \ldots, x_n) \in \mathbb{F}^n : x_i = -\sum_{j=i+1}^{n} \frac{\alpha_j}{\alpha_i} x_j\}$.

For any polynomial $f(\bar{x}) \in \mathbb{F}[\bar{x}]$, we define the restricted polynomial $f(\bar{x})_{|ker(p)}$ as the polynomial

$$f(\bar{x})_{|ker(p)} = f(x_1, \ldots, x_{i-1}, -\sum_{j=i+1}^{n} \frac{\alpha_j}{\alpha_i} x_j, x_{i+1}, \ldots, x_n) \in \mathbb{F}[x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n].$$

It can be checked that this is well defined, i.e. if we started with a different choice of l, we get the same polynomial.

Definition 9 (Central Projection)**.** Let $p_1, \ldots, p_k \in \mathbb{P}(V)$ be a set of independent points. Let $l_i \in V$ be such that $p_i = [l_i], i \in [k]$. Since the p_i's are independent, the set $\{l_1, \ldots, l_k\}$ is linearly independent. Let $W = sp(l_1, \ldots, l_k)$ and extend the basis to get $W^\perp$ such that $W \oplus W^\perp = V$. Let $p \in \mathbb{P}(V) \setminus \mathbb{P}(W)$ be any point. Consider $l \in V$ such that $[l] = p$ and write $l = w + w^\perp$ such that $w \in W$ and $w^\perp \in W^\perp$. Since $p \notin \mathbb{P}(W)$ we can conclude that $w^\perp \neq 0$. We define the central projection of p (denoted as $\pi(p)$) onto $\mathbb{P}(W^\perp)$ as :

$$\pi(p) = [w^\perp] \in \mathbb{P}(W^\perp).$$

It can be checked that this is well defined i.e. if we started with l'_i s.t. $p = [l'_i]$ we get the same space W.

Definition 10 (Projective Factors)**.** For any homogeneous polynomial f we define the multi-set of *"Projective Factors"* of f as:

$$\mathbb{P}(f) = \{[\alpha_1 x_1 + \ldots + \alpha_n x_n] \in \mathbb{P}(V) : \alpha_1 x_1 + \ldots + \alpha_n x_n \text{ divides } f\}.$$

If a $p \in \mathbb{P}(V)$ belongs to $\mathbb{P}(f)$ then we say that p is a projective linear factor of f.

Definition 11. For any point $p \in \mathbb{P}(V)$ and multi-set $\mathcal{S} \subset \mathbb{P}(V)$ we define the multi-set of lines:

$$\mathcal{L}(p, \mathcal{S}) = \{fl(p, s) : p, s \text{ are independent (same as saying } p \neq s), s \in \mathcal{S}\}$$

The lemma below says that the line joining independent points $p = [l_p], q = [l_q]$ is the same as the line joining p and restriction $[l_{q_{|ker(p)}}]$.

Lemma 2. *Let p, q be two independent points in $\mathbb{P}(V)$. Let $l_q \in V$ be such that $q = [l_q]$. Consider the restricted form $l_{q_{|ker(p)}}$. Since p, q are independent, $l_{q_{|ker(p)}}$ is non-zero and therefore $[l_{q_{|ker(p)}}]$ is a projective linear form in the variables $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n$. We have :*

$$fl(p, q) = fl(p, [l_{q_{|ker(p)}}])$$

Note that the above definition is well defined i.e. if we started with l'_q (instead of l_q) such that $q = [l'_q]$, we would get the same set $fl(p,q)$.

Proof. Let $l_p \in V$ be a linear form such that $p = [l_p]$. Let l be a linear form in V such that $[l] \in fl(p,q)$. Then $l \in sp(l_p, l_q) \setminus \{0\}$ i.e. $l = \alpha_p l_p + \alpha_q l_q$ with at least one of α_p, α_q non-zero. It is obvious by definition that $l_q = \beta_p l_p + \beta_q l_{q_{|ker(p)}}$. This implies that $l = (\alpha_p + \alpha_q\beta_p)l_p + \alpha_q\beta_q l_{q_{|ker(p)}}$. Both $\alpha_p + \alpha_q\beta_p = 0$ and $\alpha_q\beta_q = 0$ imply that either both α_p, α_q are zero or both β_p, β_q are zero. Thus $[l] \in sp(p, [l_{q_{|ker(p)}}])$. The other direction follows by a similar argument.

□

1.4 A quick introduction to arithmetic circuits

In this section we will summarize a number of basic definitions about Arithmetic Circuits. We will be using a lot of definitions from [29]. Some recent results relevant to this thesis will also be mentioned.

Definition 12 (Arithmetic circuits, adapted from [29])**.** An arithmetic circuit Φ over the field $\mathbb{F}$ and variables $\bar{x} = (x_1, \ldots, x_n)$ is a weighted directed acyclic graph as follows. The vertices of Φ are called gates. Every gate in Φ of in-degree 0 is labeled by either a variable $x_i, i \in [n]$ or a field element from $\mathbb{F}$. Every other gate in Φ is labeled by either $\times$ or $+$ and has in-degree 2. Every edge is labeled by a scalar from $\mathbb{F}$.

For every arithmetic circuit Φ, we define the following (from [29]) :

- Gates of in-degree 0 are called input gates. Gate(s) of out-degree 0 are called output gate(s).
- Gates labeled by $\times$ are called product gates and gates labeled by $+$ are called sum gates.
- $size(\Phi)$ (denoted by $|\Phi|$) is the number of edges in Φ
- For every vertex $v \in \Phi$, $depth(v)$ is the length of the longest (directed) path from an input gate to v.
- $depth(\Phi)$ is the maximal depth of a gate in Φ.
- For gates u and v in Φ, if (u,v) is an edge in Φ, then u is called a child of v, and v is called a parent of u.

Polynomial computed by an arithmetic circuit, [29]

There is a natural way to compute a polynomial using an arithmetic circuit. An input gate (labeled by a variable or field element α) computes the polynomial α. A product gate (i.e. a gate labeled by $\times$) computes the product of the polynomials computed by its children. A sum gate computes a linear combination of its children polynomials, where the weights in the linear combinations are the weights of the connecting edges. The polynomial(s) computed by the output gate(s) is(are) called the polynomial(s) computed by the circuit.

Remark 1. *In this thesis we will restrict the discussion to bounded depth circuits. These are arithmetic circuits such that* $depth(\Phi) \leq C$ *for a constant* C *independent of* n *(the number of variables).*

Depth three circuits, [29]

In this thesis, we will be concerned with we will be concerned with a special class of bounded-depth circuit called depth three circuits also known as $\Sigma\Pi\Sigma$ circuits. A $\Sigma\Pi\Sigma$ circuit is a depth three circuit with an addition gate at the top, multiplication gates at the middle layer and addition gates at the bottom most layer. A closer look at this tells us that a $\Sigma\Pi\Sigma$ circuit will compute a polynomial of the form

$$\sum_{i=1}^{k} \prod_{j=1}^{d_i} l_{i,j}(x_1, \ldots, x_n) \tag{1.1}$$

where $l_{i,j}(x_1, \ldots, x_n)$ are affine forms over the variables $x_1, \ldots, x_n$. The number of summands in the outer most sum (i.e. k) is called the fan-in of the top most gate or the top fan-in. When the top fan-in is less than or equal to k, we call the circuit a $\Sigma\Pi\Sigma(k)$ circuit. A polynomial computed by such a circuit is called a $\Sigma\Pi\Sigma(k)$ polynomial.

There are two important classes of $\Sigma\Pi\Sigma(k)$ circuits, to which every other $\Sigma\Pi\Sigma(k)$ can be reduced.

Definition 13 (Simple $\Sigma\Pi\Sigma(k)$ circuit)**.** Let C be a $\Sigma\Pi\Sigma(k)$ circuit computing the polynomial $C(\bar{x}) = M_1 + \ldots + M_k$ where each M_i is a product of affine forms. We say that C is simple if

$$gcd(M_1, \ldots, M_k) = 1.$$

Definition 14 (Minimal $\Sigma\Pi\Sigma(k)$ circuit)**.** Let C be a $\Sigma\Pi\Sigma(k)$ circuit computing the polynomial $C(\bar{x}) = M_1 + \ldots + M_k$ where each M_i is a product of affine forms. We

say that C is minimal if for no proper sub collection of polynomials $M_1, \ldots, M_k$ sums to zero.

If one does put any restrictions of the top fan-in, it can be easily checked that any polynomial can be computed by this class of arithmetic circuits.

A number of interesting results are known when the top fan-in is bounded by a constant, i.e. when k is a constant. Efficient algorithms for some popular open problems have been developed over the last decade and we will briefly mention some of them below, and then add to the pool by solving yet another important problem when $k = 2$.

Depth three circuits with top fan-in 2

As mentioned above, when the top fan-in is equal to 2, we call the circuit a $\Sigma\Pi\Sigma(2)$ circuit. This will be the class of circuits we will be concerned with. The challenge in this thesis is to compute the circuit given only a black-box access to the polynomial. We give a polynomial time randomized algorithm to achieve this goal with a *"very mild"* assumption of the number of *"free variables"* in the circuit (called simple rank of the circuit).

Definition 15 (Simple rank ($srank(C)$))**.** Let C be a $\Sigma\Pi\Sigma(k)$ circuit computing the polynomial $C(\bar{x}) = M_1 + \ldots + M_k$. Let $Gcd(C)$ denote the polynomial $gcd(M_1, \ldots, M_k)$. It's easy to see that $Sim(C) = \frac{C}{Gcd(C)} = \sum_{i=1}^{k} \prod_{j=1}^{d_i} l_{i,j}$ is also a $\Sigma\Pi\Sigma(k)$ polynomial with $gcd(\prod_{j=1}^{d_1} l_{1,j}, \ldots, \prod_{j=1}^{d_k} l_{k,j}) = 1$. The simple rank of C denoted by $srank(C)$ is defined as

$$srank(C) = dim(sp\{l_{i,j}\}).$$

Previous (relevant) results about depth three circuits

Theorem 2 (Theorem 2 in [27])**.** *Let f be an n-variate polynomial computed by a $\Sigma\Pi\Sigma(2)$ circuit of degree d, over a field $\mathbb{F}$. Then there is a randomized interpolation algorithm that given black box access to f and the parameters d and n runs in quasi-polynomial time (in n, d, $|\mathbb{F}|$) and has the following properties:*

- *If $srank(f) = \Omega(\log^2(d))$, then with probability $1 - o(1)$ the algorithm outputs the (unique) $\Sigma\Pi\Sigma(2)$ circuit for f.*

- *If* $srank(f) = O(\log^2(d))$*, then the algorithm outputs, with probability* $1 - o(1)$*, a polynomial* $Lin(f)$*, a polynomial* $Q(y_1, \ldots, y_k)$ *and* k *linear functions* $L_1, \ldots, L_k$ *, where* $k \leq rank(f)$*, such that* $Lin(f)$ *is a the product of all the linear factors of* f *and* $Lin(f).Q(L_1, \ldots, L_k) = f$*.*

Our work extends the above result to characteristic zero fields. While doing so we also achieve much better time complexity i.e. $poly(n, d)$ as compared to the quasi-polynomial time algorithm given by the above theorem. Here is a simple table explaining similarities/ differences between their and our algorithms.

[Shpilka, 2007]	This work
Iterate over linear (affine) forms in $\mathbb{F}[x_1, \ldots, x_n]$	Find "candidate" linear forms using Brill's equations
Use lower bounds on locally decodable codes to show existence of certain "good" configurations	Use lower bounds on (high dimensional) Sylvester Gallai Theorems from Barak et. al. to show existence of certain "good" configurations
Use good configurations to reconstruct	Use good configurations to reconstruct
Running time $- quasipoly(n, d, \mathbb{F})$	Running Time - $poly(n, d)$
Depends on field size	Independent of field size
Needs simple rank $\geq \Omega(\log^2 d)$	Needs simple rank $\geq \Omega(1)$

A further extension of this result was given in [18]. They derandomized the above algorithm and gave generalizations to *"generalized"* $\Sigma\Pi\Sigma(k)$ circuits. The output of their algorithm is not exactly a $\Sigma\Pi\Sigma(k)$ circuit but something close enough. We do not state the theorem here and instead redirect the reader to their paper. Please see Theorem I in [18].

Theorem 3 (Rank-bounds for identically zero $\Sigma\Pi\Sigma(k)$ circuits over $\mathbb{F}$ $(\mathbb{Q}, \mathbb{R}, \mathbb{C})$, combination of theorem 1.4 in [25] and theorem [6])**.** *Let* C *be a simple and minimal* $\Sigma\Pi\Sigma(k)$ *circuit in* n *variables over* $\mathbb{F}$ *such that it computes the identically zero polynomial, then* $srank(C) < R(k, \mathbb{F})$*, where* $R(k, \mathbb{F})$ *just depends on* k*. In particular, when* k *is a constant* $srank(C) = O(1)$*.*

1.5 Homogenization of $\Sigma\Pi\Sigma(2)$ circuits

Let $f(\bar{x}) = M_1 + M_2$ be a polynomial such that M_1, M_2 are products of affine forms in variables $x_1, \ldots, x_n$. We will describe a homogenous polynomial $f^{hom}(x_1, \ldots, x_n, z) \in$

$\mathbb{F}[x_1, \ldots, x_n, z]$ associated to $f(\bar{x})$ such that the reconstruction problem for f^{hom} solves the reconstruction problem for f. Assume the degree of f is d and denote by $f^d(\bar{x})$ the degree d homogeneous component of f. From lemma 2.1 in [7] we see that in $poly(n, d)$ time we can get black-box access to the homogeneous component f^d, given black-box access to f. Define

$$f^{hom}(x_1, \ldots, x_n, z) = \begin{cases} z^d f(\frac{x_1}{z}, \ldots, \frac{x_n}{z}) & z \neq 0 \\ f^d(x_1, \ldots, x_n) & z = 0 \end{cases}$$

Lemma 3. *Given a black-box $\mathbb{B}$ for f and the parameters n,d, in time $poly(n, d)$ we can simulate a black-box $\mathbb{B}^{hom}$ for f^{hom}.*

Proof. Proof is straight-forward. We describe how to query $\mathbb{B}^{hom}$ at point $(x_1, \ldots, x_n, z)$. If $z \neq 0$, we query $\mathbb{B}$ at $(\frac{x_1}{z}, \ldots, \frac{x_n}{z})$ and then multiply the result with z^d. If $z = 0$, lemma 2.1 in [7] computes $\mathbb{B}^{hom}$ at the point $(x_1, \ldots, x_n, z)$ by computing $f^d(x_1, \ldots, x_n)$. □

Now suppose we have reconstructed $f^{hom}(x_1, \ldots, x_n, z)$, we can reconstruct $f(x_1, \ldots, x_n)$ by substituting $z = 1$ in the circuit for f^{hom}.

Chapter 2

SOME DEFINITIONS, MAIN TOOLS AND TECHNIQUES

2.1 Introduction

Let $\mathbb{F}$ be a field and $\bar{x} = (x_1, \ldots, x_n)$ be a tuple of variables.

Definition 16. An arithmetic circuit C is called a $\Sigma\Pi\Sigma(2)$ circuit if it computes a polynomial $C(\bar{x}) \in \mathbb{F}[\bar{x}]$ of the form:

$$C(\bar{x}) = M_1 + M_2.$$

where M_1, M_2 are products of linear forms. In this case we call $C(\bar{x})$ a $\Sigma\Pi\Sigma(2)$ polynomial.

Definition 17 (Some associated definitions and observations)**.** We define some other associated polynomials, multi-sets and make certain observations (note that the polynomials are defined up to scalar multiplication).

- We define $Gcd(C) \in \mathbb{F}[\bar{x}]$, to be the $g.c.d.$ (greatest common divisor) of the two products M_1, M_2 i.e. $Gcd(C) = gcd(M_1, M_2)$. We write $M_1 = Gcd(C)R$ and $M_2 = Gcd(C)B$ where R, B are products of linear forms and $gcd(R, B) = 1$. We can factorize the polynomial $C(\bar{x})$ as

  $$C(\bar{x}) = Gcd(C)(R + B). \tag{2.1}$$

  If $Gcd(C) = 1$, then we say that the circuit C is *"simple"*.

- $Sim(C) = R + B$, is called the *"simple part"* of circuit C, since the two products R, B don't have any common factors.

- Let $\mathbb{P}(R), \mathbb{P}(B), \mathbb{P}(Gcd(C))$ denote the multi-sets of projective linear factors of $R, B, Gcd(C)$ respectively.

- It is important to note that $Sim(C)$ itself might have more linear factors. As an example consider the polynomial $(x + y_1) \ldots (x + y_n) - y_1 \ldots y_n$, where $x, y_1, \ldots, y_n$ are variables. This polynomial is divisible by x but x does not divide any of the two products $(x + y_1) \ldots (x + y_n)$ and $y_1 \ldots y_n$.

– We define $Int(C)$ as the product of all linear factors (up to scalar multiplication) of $Sim(C)$ and call it *"internal factors"*. Similarly $Res(C)$ (called *"residual factors"*) is defined to be the product (again up to scalar multiplication) of all non-linear irreducible factors of $Sim(C)$.

– *"Simple rank"* of the circuit C denoted by $srank(C)$ is defined as dimension of the multi-set $\mathbb{P}(R) \cup \mathbb{P}(B)$ (see 4 for definition of dimension).

For any multi-sets $\mathcal{S},\mathcal{T}$ of projective linear forms in the projective space $\mathbb{P}(V)$ we define a set of projective linear forms called the *"intersection set"*. This set contains forms in $\mathcal{S} \cup \mathcal{T}$ along with the forms which lie at the intersection of distinct lines connecting $\mathcal{S}$ and $\mathcal{T}$. Formally:

Definition 18 (Intersection set)**.** Let $\mathcal{S},\mathcal{T} \subset \mathbb{P}(V)$ be two multi-sets. The *"intersection set"*, $\mathcal{I}(\mathcal{S},\mathcal{T})$ comprises of the following:

1. All distinct forms in $\mathcal{S},\mathcal{T}$.
2. Intersection of distinct lines $\vec{L}_1 = fl(s,t)$ and $\vec{L}_2 = fl(s',t')$ where $s,s' \in \mathcal{S}$ and $t,t' \in \mathcal{T}$.

Note 2. It's easy to see that $|\mathcal{I}(\mathcal{S},\mathcal{T})| \leq |\mathcal{S}|^2|\mathcal{T}|^2$.

In our application we will use $\mathcal{S} = \mathbb{P}(R)$ and $\mathcal{T} = \mathbb{P}(B)$. But first, let's prove a straight-forward lemma about multi-sets $\mathbb{P}(R),\mathbb{P}(B),\mathbb{P}(Int(C))$:

Lemma 4. *The following are true:*

$$\mathbb{P}(R) \cap \mathbb{P}(Int(C)) = \phi, \qquad \mathbb{P}(B) \cap \mathbb{P}(Int(C)) = \phi$$

Proof. The proofs are similar so we just show one of them. Let $p \in \mathbb{P}(Int(C))$. Since $Sim(C) = Int(C)Res(C)$, by restricting to $ker(p)$ we see that

$$R_{|_{ker(p)}} + B_{|_{ker(p)}} = 0 \Rightarrow R_{|_{ker(p)}} = -B_{|_{ker(p)}}.$$

If $p \in \mathbb{P}(R)$ then clearly $R_{|_{ker(p)}} = 0$ and thus $B_{|_{ker(p)}} = 0$ implying that $p \in \mathbb{P}(B)$. Therefore p divides both R,B but $gcd(R,B) = 1$, a contradiction. □

2.2 Uniqueness of $\Sigma\Pi\Sigma(2)$ Structure

Note that we defined a quantity called $srank(C)$ in definition 15 (and last part of definition 17). It has been studied very extensively in the last decade and a number of efficient algorithms for polynomial identity testing have been devised by proving clever bounds on it. See theorem 1.5 and 1.7 in [25] for details.

In this section we will show that high simple rank of a $\Sigma\Pi\Sigma(2)$ circuit implies unique circuit representation. Based on the underlying field we will have different values for *"high"*. It's captured by the function $R(k,\mathbb{F})$ mentioned in theorem 3.

Theorem 4 (Follows from Corollary 7 in [27])**.** *Let C be a $\Sigma\Pi\Sigma(2)$ circuit computing the polynomial $C(\bar{x})$ such that $srank(C) > R(4,\mathbb{F})$, then the circuit C is (essentially) the unique $\Sigma\Pi\Sigma(2)$ circuit computing $C(\bar{x})$.*

Proof. Using equation 2.1 in definition 17, write $C(\bar{x}) = Gcd(C)(R + B)$. Let C' be another $\Sigma\Pi\Sigma(2)$ circuit also computing $C(\bar{x})$. Therefore $C(\bar{x})$ also has the form $Gcd(C')(R' + B') \Rightarrow$

$$Gcd(C)(R + B) - Gcd(C')(R' + B') = 0$$

Note that $Gcd(C), Gcd(C')$ are products of linear forms. We can remove their common factors in the above identity i.e. let $\bar{G} = gcd(Gcd(C), Gcd(C'))$ and let $Gcd(C) = \bar{G}H, Gcd(C') = \bar{G}H'$ with $gcd(H,H') = 1$ giving the identity:

$$HR + HB - H'R' - H'B' = 0$$

The polynomial $HR + HB - H'R' - H'B'$ clearly has a $\Sigma\Pi\Sigma(4)$ circuit. It's easy to check that $gcd(HR, HB, H'R', H'B') = 1$. If not then there exists a linear form l dividing each of the four polynomials $HR, HB, H'R', H'B'$. In particular l divides $HR \Rightarrow l$ divides H or l divides R.

- If l divides H then it cannot divide H' (H, H' are co-prime). Therefore it divides both R' and B' which is not possible as they are co-prime.
- If l divides R then l does not divide B as R, B are co-prime. l divides $HB \Rightarrow l$ divides H which is not possible as was just shown.

So we have a simple $\Sigma\Pi\Sigma(4)$ circuit which is identically 0. If this is also minimal then by theorem 3 simple rank of this circuit is $\leq R(4,\mathbb{F})$ by [25] . But that is not

true since $srank(C) > R(4,\mathbb{F})$. Therefore it's not minimal i.e. some two gates sum to 0. Going through the several cases we get either

$$(HR, HB) = (H'R', H'B') \Rightarrow (Gcd(C)R, Gcd(C)B) = (Gcd(C')R', Gcd(C')B')$$

or

$$(HR, HB) = (H'B', H'R') \Rightarrow (Gcd(C)R, Gcd(C)B) = (Gcd(C')B', Gcd(C')R').$$

Both of these mean that the circuit C was unique (up to relabeling the multiplication gates). □

2.3 Factoring forms of a polynomial

Definition 19. Let $f(\bar{x}) \in \mathbb{F}[\bar{x}]$ be a polynomial. We say that a projective linear form p is a *factoring form* for $f(\bar{x})$ if $f(\bar{x})_{|_{ker(p)}}$ is a non-zero product of linear forms in the ring $\mathbb{F}[x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n]$. See definition 8 for definition of restriction to $ker(p)$.

The set of factoring forms for a polynomial $f(\bar{x})$ will be denoted by $\mathcal{P}(f)$.

We will now investigate properties of factoring forms for polynomials associated with $\Sigma\Pi\Sigma(2)$ polynomials.

Factoring forms for $\Sigma\Pi\Sigma(2)$ polynomials

Recall that $Sim(C) = R+B$ with $gcd(R,B) = 1$. Let $p \in \mathbb{P}(R)$ be a projective linear factor of R. We easily see that $Sim(C)_{|_{ker(p)}} = B_{|_{ker(p)}} \neq 0$ (since $gcd(R,B) = 1$). $B_{|_{ker(p)}}$ is a non-zero product of linear forms since B was a product of linear forms, implying that p is a factoring form for $Sim(C)$. Therefore forms in $\mathbb{P}(R)$ (similarly $\mathbb{P}(B)$) are factoring forms for $Sim(C)$ and thus belong to $\mathcal{P}(Sim(C))$.

This gives us motivation to compute $\mathcal{P}(Sim(C))$ as an approach to finding the products R, B. However there is a problem. We don't have access to $Sim(C)$. This problem can be circumvented by using $Res(C)$ (defined as residual factors in definition 17) instead. We later discuss how to get black-box access to $Res(C)$.

In the next theorem we will discuss properties of the set of *factoring forms* of $Res(C)$. This set turns out to be a subset of the *intersection set* $\mathcal{I}(\mathbb{P}(R), \mathbb{P}(B))$ (for definition of intersection set see definition 18).

Theorem 5. *Let C be a $\Sigma\Pi\Sigma(2)$ circuit computing the polynomial $C(\bar{x})$. Every form $p \in \mathbb{P}(R) \cup \mathbb{P}(B)$ belongs to the set of factoring forms $\mathcal{P}(Res(C))$. Further if*

we assume that $srank(C) > R(3,\mathbb{F}) + 2$*, then :*

$$\mathcal{P}(Res(C)) \subset \mathcal{I}(\mathbb{P}(R),\mathbb{P}(B))$$

i.e. the factoring forms are either forms in $\mathbb{P}(R),\mathbb{P}(B)$ *or lie at intersections of distinct lines joining forms in* $\mathbb{P}(R)$ *with forms in* $\mathbb{P}(B)$*.*

Proof. Let p be a factoring form for $Res(C)$. Recall the definition of internal factors $Int(C)$ from definition 17. On restricting to $ker(p)$ we get (see definition 8)

$$R_{|_{ker(p)}} + B_{|_{ker(p)}} - Int(C)_{|_{ker(p)}} Res(C)_{|_{ker(p)}} = 0 \tag{2.2}$$

p being a factoring form for $Res(C)$ implies that $Res(C)_{|_{ker(p)}}$ is a non-zero product of linear forms. This gives us an identically zero $\Sigma\Pi\Sigma(3)$ polynomial $R_{|_{ker(p)}} + B_{|_{ker(p)}} - Int(C)_{|_{ker(p)}} Res(C)_{|_{ker(p)}} = 0$.

Let's first pull out the g.c.d. G of the three products (multiplication gates) and define

$$GR' = R_{|_{ker(p)}}, \qquad GB' = B_{|_{ker(p)}}, \qquad GF' = Int(C)_{|_{ker(p)}} Res(C)_{|_{ker(p)}}.$$

with $gcd(R', B', F') = 1$. We have two cases:

1. **Case 1 :** Set of linear factors of G has dimension more than one : Since G divides $R_{|_{ker(p)}}$ and $B_{|_{ker(p)}}$, let's assume $R_{1|_{ker(p)}} (\sim B_{1|_{ker(p)}})$ and $R_{2|_{ker(p)}} (\sim B_{2|_{ker(p)}})$ are two linearly independent linear factors of G where R_1, R_2 divides R and B_1, B_2 divides B. So we see that p lies on lines $\vec{L}_1 = fl([R_1],[B_1])$ and $\vec{L}_2 = fl([R_2],[B_2])$ (recall that $[R_i],[B_i]$ are projectivizations of R_i, B_i respectively, see definition 1). Linear independence of $R_{1|_{ker(p)}}, R_{2|_{ker(p)}}$ implies that the lines are distinct. Therefore $p \in \mathcal{I}(\mathcal{R},\mathcal{B})$.

2. **Case 2:** Set of linear factors of G is one dimensional : Note that we assumed that $srank(C) \geq R(3,\mathbb{F}) + 2 \Rightarrow$ dimension of the linear factors of R, B is greater than $R(3,\mathbb{F}) + 2$. Therefore on restricting to the hyperplane the dimension goes down at most by one. That is dimension of linear factors of $R_{|_{ker(p)}}, B_{|_{ker(p)}}$ is greater than $R(3,\mathbb{F}) + 2 - 1$. By the assumption in this case we will get that dimension of linear factors of R', B' is greater than $R(3,\mathbb{F}) + 2 - 1 - 1 = R(3,\mathbb{F})$. On simplification (i.e. dividing equation 2.2 by G) we obtain an identically zero $\Sigma\Pi\Sigma(3)$ polynomial :

$$R' + B' - F' = 0.$$

The identically zero $\Sigma\Pi\Sigma(3)$ circuit computing the above polynomial is simple (i.e. the product gates are co-prime) and has *"simple rank"* $\geq R(3,\mathbb{F})$. Therefore by theorem 3 it cannot be minimal. Thus two of the products (multiplication gates) R', B', F' must sum to zero $\Rightarrow$ one of R', B', F' is zero.

- If $R' = \frac{R_{|ker(p)}}{G} = 0$, or $B' = \frac{B_{|ker(p)}}{G} = 0$ then p divides R or B and thus $p \in \mathbb{P}(R) \cup \mathbb{P}(B) \subset \mathcal{I}(\mathbb{P}(R), \mathbb{P}(B))$.
- If $F' = 0$ then $R' + B' = 0 \Rightarrow R' = -B'$. We know that dimension of linear factors of R', B' is greater than $R(3,\mathbb{F}) > 2$ (if not then we can just re-define $R(3,\mathbb{F}) = max(R(3,\mathbb{F}), 3)$). So exactly like case 1 above we can find projective linear forms R_1, R_2 dividing R and B_1, B_2 dividing B such that p lies on distinct lines $fl(R_i, B_i), i \in [2]$ further implying $p \in \mathcal{I}(\mathbb{P}(R), \mathbb{P}(B))$.

□

The next lemma will be very crucial in the reconstruction process. It states that if we have enough projective linear forms from $\mathbb{P}(R)$, then every factoring form connects one of them to some projective linear form in $\mathbb{P}(B)$. We call this the matching lemma. Proof is exactly like the last theorem but we still write it for completion.

Lemma 5 (Matching lemma)**.** *Fix $k = R(3,\mathbb{F}) + 2$. Let $[R_1], \ldots, [R_k]$ be independent projective linear forms in $\mathbb{P}(R)$ and p be any factoring form in $\mathcal{P}(Res(C))$ but not in $\mathbb{P}(R) \cup \mathbb{P}(B)$. Then there exists $[R_i], i \in [k]$ and $[B_i] \in \mathbb{P}(B)$ such that p lies on the line $fl([R_i], [B_i])$.*

Proof. Assume the converse i.e. for some factoring form p there does not exists such R_i, B_i. Exactly like the previous theorem we obtain

$$R_{|ker(p)} + B_{|ker(p)} - Int(C)_{|ker(p)} Res(C)_{|ker(p)} = 0$$

and then define G, R', B', F' such that

$$GR' = R_{|ker(p)}, \qquad GB' = B_{|ker(p)}, \qquad GF' = Int(C)_{|ker(p)} Res(C)_{|ker(p)}.$$

with $gcd(R', B', F') = 1$. Following the previous proof we have the identity

$$R' + B' - F' = 0$$

If for any $R_i, i \in [k]$, the restriction $R_{i|_{ker(p)}}$ divides G, then exactly like the previous proof there is a projective linear form $[B_i]$ dividing B such that $p \in fl([R_i],[B_i])$ and we are done.

So assume that all $R_{i|_{ker(p)}}, i \in [k]$ divide R'. We know that the restrictions $\{R_{i|_{ker(p)}} : i \in [k]\}$ have dimension equal to $k-1 > R(3,\mathbb{F})$ implying that simple rank of the $\Sigma\Pi\Sigma(3)$ circuit $R'+B'-F'$ is greater than $R(3,\mathbb{F}) \Rightarrow$ (exactly like last proof) it cannot be minimal, otherwise it violates rank bound for identically zero polynomials in theorem 3. So like the previous proof some product gate is zero. $p \notin \mathbb{P}(R) \cup \mathbb{P}(B)$ implies R', B' are non-zero $\Rightarrow F' = 0 \Rightarrow R' = -B' \neq 0$. Hence there exists projective linear form $[B_1]$ dividing B such that $B_{1|_{ker(p)}} \sim R_{1|_{ker(p)}}$ and p lies on line $fl([R_1],[B_1])$.

□

2.4 Good forms and reconstructed multi-set

In this section we explain the main object that accomplishes the goal of reconstruction in chapter 5. Later in chapter 5, we will show that these objects exist and apply them to reconstruct our multi-sets $\mathbb{P}(M_1), \mathbb{P}(M_2)$.

Definition 20 (Good form)**.** Let p be a projective linear form in $\mathbb{P}(V)$, $\mathcal{S}, \mathcal{T}$ be multi-sets in $\mathbb{P}(V)$. We say that p is a *"good form"* for $(\mathcal{S}, \mathcal{T})$ if there exists $s \in \mathcal{S}, t \in \mathcal{T}$ such that :

1. p, s, t are pairwise distinct.
2. The forms p, s, t are not collinear.
3. The plane $\Psi_{p,s,t} = fl(p,s,t)$ intersects $\mathcal{T}$ only along the line $fl(p,t)$.

We collect all $t \in \mathcal{T}$ for which there exists some $s \in \mathcal{S}$ such that p, s, t satisfy the above requirements. This multi-set will be very special for us. We'll be able to reconstruct it. Let's give it a name.

Definition 21. Suppose p is a *"good form"* for $(\mathcal{S}, \mathcal{T})$ we define the *"reconstructed multi-set"* $\mathcal{T}_p \subset \mathcal{T}$ as

$$\mathcal{T}_p = \{\tilde{t} \in \mathcal{T} : \exists s \in \mathcal{S} : (p,s,t) \text{ satisfy bullets 1, 2 and 3 in definition 20}\}$$

Later on we will show how we can reconstruct multi-sets when a good form is given. But first we describe an example of a good form. This example demonstrates two very important applications of good forms in chapter 5. Here is a lemma.

Lemma 6. *Suppose $p, s \in \mathbb{P}(V)$ are projective linear forms and $\mathcal{T}$ be a multi-set of projective linear forms. Assume*

- *$p \neq s$ and $fl(p,s) \cap fl(\mathcal{T}) = \phi$.*

Then p is a good form for $(\{s\}, \mathcal{T})$ and the "reconstructed multi-set" *is $\mathcal{T}_p = \mathcal{T}$.*

Proof. The proof is particularly simple. Consider any $t \in \mathcal{T}$.

- Since $p \neq s$ and $fl(p,s) \cap fl(\mathcal{T}) = \phi$ we get that p, s, t are pairwise distinct and independent. Therefore condition 1 and condition 2 in definition 20 are satisfied.
- Consider $\Psi = fl(p,s,t)$. Consider any $t'(\neq t) \in \Psi \cap \mathcal{T}$. So we may write $t' = \alpha_p p + \alpha_s s + \alpha_t t$, with at least one of $\alpha_p, \alpha_s, \alpha_t$ non-zero. If $\alpha_s \neq 0$, then we may re-write the equation as $\alpha_s s + \alpha_p p = t' - \alpha_t t \Rightarrow fl(s,p) \cap fl(t,t') \neq \phi$ (as $\alpha_s \neq 0$). Therefore we arrive at a contradiction to $fl(p,s) \cap fl(\mathcal{T}) = \phi \Rightarrow \alpha_s = 0 \Rightarrow \Psi \cap \mathcal{T} \subset fl(p,t)$. Therefore p, s, t satisfy condition 3 in definition 20.

Therefore p is a good form for $(\{s\}, \mathcal{T})$. Also since $t \in \mathcal{T}$ was arbitrary, bullets 1,2,3 in definition 20 hold for all $t \in \mathcal{T}$ and thus the reconstructed multi-set in this case is $\mathcal{T}_p = \mathcal{T}$. □

Lemma 7. *Suppose $p \in \mathbb{P}(V)$ is a "good form" for $(\mathcal{S}, \mathcal{T})$ and let $\mathcal{T}_p \subset \mathcal{T}$ be the "reconstructed multi-set". Further assume the following*

1. *The form p and the multi-set $\mathcal{S}$ are known. Multi-set $\mathcal{T}$ is unknown.*
2. *The multi-set of lines $\mathcal{L}(p, \mathcal{T})$ (see definition 11) is known.*
3. *For every $s \in \mathcal{S}$, the multi-set of lines $\mathcal{L}(s, \mathcal{T})$ is known.*

Then there exists a deterministic algorithm that runs in $poly(|\mathcal{S}|, |\mathcal{T}|, n)$ time and reconstructs the multi-set $\mathcal{T}_p$.

Proof. Let's first give the algorithm and then discuss correctness and time complexity.

Initialize $\tilde{\mathcal{T}_p} = \phi$.

for *each* $s(\neq p) \in \mathcal{S}$ **do**
  Let $\vec{L}$ be the line $fl(p,s)$.
  for *each line* $\vec{L}_p$ *inside* $\mathcal{L}(p,\mathcal{T}) \setminus \{\vec{L}\}$ **do**
    Consider the plane $\Psi = fl(\vec{L}_p, s)$ spanned by line $\vec{L}_p$ and form s.
    Find lines in $\mathcal{L}(p,\mathcal{T})$ lying on Ψ.
    if $\vec{L}_p$ *is the only such line* **then**
      for *each line* $\vec{L}_s \in \mathcal{L}(s,\mathcal{T}) \setminus \{\vec{L}\}$, *lying on* Ψ **do**
        Let $\tilde{t}$ be the intersection of lines $\vec{L}_p$ and $\vec{L}_s$.
        Update $\tilde{\mathcal{T}_p} = \tilde{\mathcal{T}_p} \cup \{\tilde{t}\}$ (note that this is multi-set union).
      end
    end
  end
end

Return $\tilde{\mathcal{T}_p}$.

Algorithm 1: Reconstruction using a Good form

Correctness Proof - We will show below that the set $\tilde{\mathcal{T}_p}$ computed by the above algorithm is the same as the "reconstructed multi-set" i.e. $\tilde{\mathcal{T}_p} = \mathcal{T}_p$.

1. **Proof of $\mathcal{T}_p \subset \tilde{\mathcal{T}_p}$:** Let $t \in \mathcal{T}_p$. By the definition of *"reconstructed multi-set"* above, we know that there is an $s \in \mathcal{S}$ such that (p,s,t) satisfy conditions in definition 20. The first for loop will select s at some point of time. Definition 20 implies that $fl(p,t)$ does not contain s and so after choosing s, the second for loop selects the line $\vec{L}_p = fl(p,t)$ at some point of time. Since (p,s,t) satisfy conditions of definition 20, $\vec{L}_p$ is the only line from $\mathcal{L}(p,\mathcal{T})$ on the plane $\Psi = fl(p,s,t)$, and so the if condition inside the second for loop will be true. Therefore the algorithm will further choose $\vec{L}_s = fl(s,t)$ at some point

of time. Clearly t is the intersection of lines $\vec{L}_p$ and $\vec{L}_s$. When this happens t gets added to the set $\tilde{\mathcal{T}_p}$ implying that $\mathcal{T}_p \subset \tilde{\mathcal{T}_p}$.

2. **Proof of $\tilde{\mathcal{T}_p} \subset \mathcal{T}_p$:** Consider a form $\tilde{t} \in \tilde{\mathcal{T}_p}$. We first show that $\tilde{t} \in \mathcal{T}$. The algorithm constructs $\tilde{t}$ as intersection of two lines $\vec{L}_p \in \mathcal{L}(p,\mathcal{T})$ and $\vec{L}_s \in \mathcal{L}(s,\mathcal{T})$ for some $s \in \mathcal{S}$. Both these lines are different from the line $\vec{L} = fl(p,s)$. Clearly $\vec{L}_s = fl(s,t)$ for some $t \in \mathcal{T}$.

   We show that $\tilde{t} = t$. Suppose not, then since $\tilde{t} \in \vec{L}_s$ and $s \neq \tilde{t}$ (otherwise $s \in \vec{L}_p$), we have three distinct forms $\{s,\tilde{t},t\}$ on $\vec{L}_s$. The line $\vec{L}_s$ was chosen to be on the plane $\Psi = fl(\vec{L}_p, s)$. Therefore there are two distinct lines $fl(p,t)$ and $\vec{L}_p = fl(p,\tilde{t})$ on this plane Ψ. This is a contradiction to the choice of $\vec{L}_p$, thus $\tilde{t} = t$.

   - p, s are different by choice of s in the second for loop. $\tilde{t}$ is intersection of two lines $\vec{L}_p, \vec{L}_s$ (different from $\vec{L}$) and thus $\tilde{t}$ is different from both p, s. Thus condition 1 in definition 20 is satisfied for $(p, s, \tilde{t})$.
   - Since $\vec{L}_p$ was different from the line $\vec{L}$, we get that $p, s, \tilde{t}$ are not collinear so condition 2 in definition 20 is satisfied for $(p, s, \tilde{t})$
   - Let $t \in \mathcal{T}$ be any projective linear form on the plane $\Psi = fl(s, \vec{L}_p) = fl(p, s, \tilde{t}) = \Psi_{p,s,\tilde{t}}$. If t lies on line $\vec{L}_p = fl(p,\tilde{t})$ we are fine. If not then $fl(p,t)$ is a line from $\mathcal{L}(p,\mathcal{T})$ different from $\vec{L}_p$, passing through p and lying on Ψ, which is a contradiction to the choice of $\vec{L}_p$. Therefore $\psi_{p,s,\tilde{t}} \cap \mathcal{T} \subset fl(p,\tilde{t})$ and condition 3 in definition 20 is satisfied for $(p, s, \tilde{t})$.

Time Complexity : We examine the nested loop structure. First loop runs $\leq |\mathcal{S}|$ times. Second loop runs $\leq |\mathcal{L}(p,\mathcal{T})|$ times. Inside the second loop finding all lines in $\mathcal{L}(p,\mathcal{T})$ takes $|\mathcal{L}(p,\mathcal{T})| poly(n)$ steps since testing whether a line lies on a plane in n dimensions can be solved using linear algebra in $poly(n)$ steps. Inner most loop runs $\leq |\mathcal{L}(s,\mathcal{T})|$ times and finding intersection of lines again takes $poly(n)$ time using linear algebra techniques. It's easy to see that $|\mathcal{L}(p,\mathcal{T})|, |\mathcal{L}(s,\mathcal{T})|$ are both $\leq |\mathcal{T}|$ and so overall we take $poly(|\mathcal{T}|, |\mathcal{S}|, n)$ time.

So we see that $\mathcal{T}_p$, the *"reconstructed multi-set"* is actually reconstructed by the above algorithm in $poly(|\mathcal{S}|, |\mathcal{T}|, n)$ time. □

Chapter 3

MAIN RESULT AND OVERVIEW

In this chapter we will state our main theorem and also give an outline of the algorithm which the theorem claims. Proof of this theorem is a combination of results in the next two chapters. The underlying field for this entire work will be $\mathbb{F}$, a field of characteristic zero. For simplicity we assume it to be $\mathbb{Q}, \mathbb{R}$ or $\mathbb{C}$. Our tuple of variables will be denoted by $\bar{x} = (x_1, \ldots, x_n)$. Fix N_0 to be large enough constant (see beginning of chapter 5 for details). Here is the main theorem:

Theorem 6. *Let C be a homogeneous $\Sigma\Pi\Sigma(2)$ circuit computing a degree d polynomial $C(\bar{x})$ in n variables $x_1, \ldots, x_n$. Assume that black-box access to C has been given (along with parameters n, d). We give a randomized algorithm that runs in time $poly(n, d)$ and with probability $1 - o(1)$ outputs the following:*

- *When $srank(C)$[1] $\geq N_0$, the output is a $\Sigma\Pi\Sigma(2)$ circuit computing $C(\bar{x})$.*

3.1 Overview of the algorithm

Recall that we are dealing with a $\Sigma\Pi\Sigma(2)$ circuit computing the polynomial:

$$C(\bar{x}) = M_1(\bar{x}) + M_2(\bar{x})$$

where M_1, M_2 are products of linear forms. Given black-box access to C, we wish to compute M_1, M_2. Our algorithm has two very broad steps :

Step I - Reconstruct I^{st} layer of the circuit

In this step we try to find a set of linear forms which appear at layer I in the circuit C. These are precisely the linear factors of M_1, M_2. We end up reconstructing a few extra linear forms and get rid of them afterwards. At the I^{st} layer, we have the following two categories of linear forms.

1. **Linear forms at layer I which divide $C(\bar{x})$ -** These are precisely the common linear factors of M_1, M_2. Such linear forms definitely divide the polynomial $C(\bar{x})$. However these may not be all linear factors of $C(\bar{x})$. Consider the

[1] This is just the rank of $Sim(C)$.

polynomial $(x+y_1)\ldots(x+y_n)-y_1\ldots y_n$. The linear form x divides the polynomial but does not divide any of the two products $(x+y_1)\ldots(x+y_n)$ and $y_1\ldots y_n$. Instead of computing the set of common linear factors of M_1, M_2, we compute the set of all linear factors of $C(\bar{x})$. Later on during step 2 of the algorithm, the bad forms get rejected. In order to find all linear factors of $C(\bar{x})$, we use the standard black-box factoring algorithm of [17]. The algorithm gives us access to black-boxes for the factors. We convert them into explicit coefficient form in algorithm C.

2. **Linear forms at layer I which don't divide** $C(\bar{x})$ **-** Let's write $M_1 = Gcd(M_1, M_2)R$ and $M_2 = Gcd(M_1, M_2)B$ where R, B are products of linear forms such that $gcd(R, B) = 1$. Then the linear forms at layer I which don't divide $C(\bar{x})$ are precisely the linear factors of R and B. We compute a set of size $poly(n, d)$ such that it contains all distinct linear factors of R and B. This is achieved by first making a random invertible transformation in Section 4.2 to make sure that our variables $x_1, \ldots, x_n$ become *"random"*. Next in $C(\bar{x})$ we set all but constant many variables to zero. The restriction of our polynomial to constant many variables can actually be computed in coefficient form efficiently using the original black-box.

   Now for this restricted polynomial, we find a set of linear forms which contains the (restricted) linear factors of R, B. This is done using brill's equations (see appendix B) which completely characterize the coefficients of polynomials which split into linear factors. We repeat the whole process for different subsets of constant many variables and compute a set containing restricted linear factors of R, B in each case. Finally we describe a method to glue all these sets of restricted linear forms. This gives us a set of linear forms over $x_1, \ldots, x_n$ containing linear factors of R, B. The linear forms in this final set has certain bad elements (forms which don't divide R, B). But these bad forms have certain structure and get rejected during the course of our algorithm.

3. The two multi-sets computed above are then sent to the next part of the algorithm which involves reconstructing the *"wiring"* of the circuit and finding the gates at layer II. Along with these sets, as a by-product of algorithm C we also compute a black-box computing the polynomial which is the product of all non-linear irreducible factors of $C(\bar{x})$. This is also used to reconstruct the gates at layer II.

Step II - Reconstruct IInd layer of the circuit

In this step we use the linear forms and the black-box (computing product of non-linear irreducible factors of $C(\bar{x})$) computed above and reconstruct the wiring in the graph of our circuit.

Suppose r, b are linear forms dividing R, B respectively. Using the outputs from step I, we can calculate the multi-sets

$$\{l \pmod r : l \text{ is a linear factor of } M_2\}, \quad \{l \pmod b : l \text{ is a linear factor of } M_1\}.$$

Viewing the linear forms as points in space, the above multi-sets enable us to find multi-sets of lines going from r to linear factors of M_2 and multi-sets of lines going from b to linear factors of M_1.

Next we look for non-degenerate planes $\Psi = sp\{r_1, r_2, l\}$ ($sp\{b_1, b_2, l\}$) where r_1, r_2 are linear factors of R and l is a linear factor of M_2 (resp. b_1, b_2 are linear factors of B and l is a linear factor of M_1) satisfying the following condition.

- Linear factors of M_2 (resp. M_1) lying on Ψ only lie on the line $\overrightarrow{r_1, l}$ (resp. $\overrightarrow{b_1, l}$).

We show that if such a configuration exists, then we can reconstruct l along with the multiplicity with which it divides M_2 (resp. M_1) by considering intersections of lines in Ψ. So the whole effort then goes into showing existence of such planes Ψ (and that it can be found efficiently in every iteration of the algorithm). To do this we use quantitative versions of the Sylvester Gallai theorem given in [3] and it's improvements from [6].

During the algorithm one of the problems we encounter is that the set $\mathcal{P}$ provided by step I contains points other than linear factors of R, B. So we need to make sure that we do not use these *bad points*. We do this by finding structure (see matching lemma, 5) in these bad points (due to the way they were constructed) and use this structure to eliminate them. If we can do this wisely then the reconstruction process goes smoothly.

Finally, if we have reconstructed all the linear factors for one of the products M_1, M_2 we compute an appropriate constant that we need to multiply to the product of our linear forms (since all linear forms will be obtained up to scalar multiplication). This is done by using brill's equations again and the algorithm has been explained

in subsection 5.3. This will be the last step of all our reconstruction algorithms in step II.

Once we have done the above our reconstruction is complete. For technical reasons we work with projective linear forms instead of linear forms in the entire discussion above. This is done to give better exposition by avoiding certain trivial technicalities that appear when a linear form is known only up to scalar multiplication.

Chapter 4

STEP ONE : RECONSTRUCT THE IST LAYER OF C

4.1 Introduction

Recall that we have access to a black-box $\mathbb{B}$ for a $\Sigma\Pi\Sigma(2)$ circuit C computing the polynomial $C(\bar{x}) = M_1 + M_2 = Gcd(C)(R + B) = Gcd(C)Int(C)Res(C)$ (see definition 17). In this chapter we wish to use $\mathbb{B}$ and compute all the projective linear forms corresponding to linear forms computed at the first layer in circuit C.

On looking closely we can see that the outputs at layer I are just (scalar multiples of) the linear factors of $Gcd(C) = gcd(M_1, M_2)$ and polynomials R, B. Thus our main objective for this chapter is to find the multi-set of projective linear forms $\mathbb{P}(Gcd(C)) \cup \mathbb{P}(R) \cup \mathbb{P}(B)$, where the union is a multi-set union. We will not be constructing this multi-set exactly but something close enough.

In this section we give algorithms to compute the following:

1. The multi-set $\mathbb{P}(Gcd(C)Int(C))$ which is the multi-set of projective linear factors of $C(\bar{x})$. By abuse of notation we say that a projective linear form divides a polynomial whenever a corresponding linear form divides the polynomial.

2. A set $\mathcal{P} \subset \mathbb{P}(V)$ of projective linear forms containing (distinct) projective linear forms from $\mathbb{P}(R)$ and $\mathbb{P}(B)$. We wish to emphasize that this set does not give us information about multiplicities of forms in their respective sets $\mathbb{P}(R)/\mathbb{P}(B)$. For that we develop methods in chapter 5. The set $\mathcal{P}$ that we compute here has size $poly(d)$ where d is the degree of $C(\bar{x})$.

The first part of the theorem i.e. computing $\mathbb{P}(Gcd(C)Int(C))$ has already been done in algorithm C of appendix C using kaltofen's black-box factoring algorithm from [17]. It also gives us a black-box $\mathbb{B}_{Res}$ computing $Res(C)$. We just invoke the algorithm here and move on to solve the second part. Thus our goal becomes:

Goal of this Section. Given a $\Sigma\Pi\Sigma(2)$ circuit C as a black-box, efficiently compute a set of projective linear forms $\mathcal{P}$ such that:

– $p \in \mathbb{P}(R) \cup \mathbb{P}(B) \Rightarrow p \in \mathcal{P}$, and $\quad |\mathcal{P}| = poly(d)$.

We achieve this goal by computing the set of *"factoring forms"* (see definition 19) for the polynomial $Res(C)$ i.e.

$$\mathcal{P} := \mathcal{P}(Res(C)).$$

As we mentioned before solving the first part using algorithm C already gave us access to a black-box $\mathbb{B}_{Res}$ computing $Res(C)$. The reasons for choosing this set $\mathcal{P}$ are mentioned below.

Note 3 (From theorem 5)**.** For any $\Sigma\Pi\Sigma(2)$ circuit C computing polynomial

$$C(\bar{x}) = Gcd(C)(R + B) = Gcd(C)Int(C)Res(C)$$

we have:

- Any $p \in \mathbb{P}(R) \cup \mathbb{P}(B)$ belongs to $\mathcal{P}(Res(C))$, i.e. p is a *"factoring form"* for the polynomial $Res(C)$.
- if $srank(C)$ is high enough ($\geq R(3,\mathbb{F})$+2), then $\mathcal{P}(Res(C)) \subset \mathcal{I}(\mathbb{P}(R), \mathbb{P}(B))$ and therefore $|\mathcal{P}(Res(C))| \leq d^4$.

So this set satisfies both properties we wanted in $\mathcal{P}$. From now onwards we set $\mathcal{P} := \mathcal{P}(Res(C))$ and try to compute it using the black-box.

Let's summarize our result in the following theorem:

Theorem 7. *Let C be a homogeneous $\Sigma\Pi\Sigma(2)$ circuit computing a degree d polynomial $C(\bar{x})$ in n variables $x_1, \ldots, x_n$. Assume that black-box access to C has been given (along with parameters n, d). Further assume that $srank(C) > max(R(3,\mathbb{F})+2, R(4,\mathbb{F}))$. There exists a randomized algorithm that runs in time $poly(n,d)$ and outputs two multi-sets $\mathcal{F}$ and $\mathcal{P}$ of projective linear forms such that*

$$Pr[\mathcal{F} = \mathbb{P}(Gcd(C)Int(C)) \quad and \quad \mathcal{P} = \mathcal{P}(Res(C))] \geq 1 - o(1)$$

To compute this set $\mathcal{P}$ we follow a standard *"restrict and lift"* technique. The broad idea is:

1. **Restriction Step -** Restrict $C(\bar{x})$ to a number of *"random"* low dimensional subspaces of $\mathbb{F}^n$ i.e. set many of the variables (they are *"random"* by the application of a random transformation on the inputs, see section 4.2) to zero. Then using the restricted polynomial we compute sets $\mathcal{P}_i$ which are (or at least contain) restrictions of $\mathcal{P}$. These sets $\mathcal{P}_i$ can be computed by solving a system of polynomial equations.

2. **Lifting Step -** Once we have the $\mathcal{P}_i$'s we glue them together. This gives us a set containing $\mathcal{P}$, since restrictions of forms in $\mathcal{P}$ are definitely glued. Then we prune this set to throw away the bad forms using the definition of $\mathcal{P}$ i.e. all forms in $\mathcal{P}$ are factoring forms for $Res(C)$. The random subspace is very important since it makes sure that we glue whatever is needed and we don't take too much time gluing. In short it introduces a lot of non-degeneracy among the restrictions.

4.2 Random Transformation

Before doing any computation with the input black-box $\mathbb{B}_{in}$, we *"apply"* a random transformation $\tilde{\Omega}$ to it. This has been explained in great detail in appendix A.

$\tilde{\Omega}$ is constructed with the help of an $n \times n$ matrix $\Omega = (\Omega_{i,j})$, where each entry is chosen uniformly randomly and independently from the a set $S \subset \mathbb{F}$. On the set $\{x_1, \ldots, x_n\}$, $\tilde{\Omega}$ maps $x_i \mapsto (\Omega\, \bar{x})_i$ ($\bar{x}$ is treated as a column vector $(x_1, \ldots, x_n)^T$). This map is then extended to an algebra homomorphism on $\mathbb{F}[\bar{x}]$. When the matrix Ω is invertible the map $\tilde{\Omega}$ becomes an isomorphism. We proceed only if Ω is invertible which happens with a high probability by lemma 22 in appendix A. If it's not invertible we output *"fail"*.

In order to *"apply"* $\tilde{\Omega}$ to our input black-box $\mathbb{B}_{in}$, we define a new black-box $\mathbb{B}$ such that for every $a \in \mathbb{F}^n$, $\mathbb{B}(\bar{a}) = \mathbb{B}_{in}(\Omega^{-1}(\bar{a}))$. This just means that to query the new black-box $\mathbb{B}$ at point $\bar{a}$, we query the old black-box at point $\Omega^{-1}(\bar{a})$. If $\mathbb{B}_{in}$ computed the polynomial $f(\bar{x})$ then $\mathbb{B}$ computes $\tilde{\Omega}(f)$.

From now onwards we assume that Ω is invertible and $\tilde{\Omega}$ has already been applied to the input black-box $\mathbb{B}_{in}$. We will work with the new black-box $\mathbb{B}$ which also represents a $\Sigma\Pi\Sigma(2)$ circuit C computing polynomial $C(\bar{x})$. We use all definitions in chapter 2 for this circuit/polynomial.

Assumption - Independence preserving restrictions of the intersection set

Recall the definition of intersection set in definition 18. Using the random transformation defined above we have created a new *"random"* set of variables $x_1, \ldots, x_n$. Applying $\tilde{\Omega}$ preserves linear independence since it is an algebra isomorphism and hence a linear isomorphism. In subsection 4.3, we will restrict our polynomial $C(\bar{x})$ to certain subsets of these variables. While doing this restriction, we want to preserve independence for points in the intersection set. If independence is preserved for restriction of points in $\mathcal{I}(\mathbb{P}(R), \mathbb{P}(B))$ then it automatically gets preserved for

restrictions of points in multi-sets $\mathbb{P}(R),\mathbb{P}(B)$ and the set $\mathcal{P}$ since they are all subsets of $\mathcal{I}(\mathbb{P}(R),\mathbb{P}(B))$. This turns out to be really beneficial for us when we try to *"glue"* the restrictions together.

We first define the subspaces we'll be restricting to. They will be used frequently and so we make a separate definition for them.

Definition 22. Define subspaces $W_{r-1} = \{(w_1,\ldots,w_{r-1},0\ldots,0) : w_1,\ldots,w_{r-1} \in \mathbb{F}\} \subset \mathbb{F}^n$ and $W_i = \{(w_1,\ldots,w_{r-1},0,\ldots,0,w_i,0,\ldots,0) : w_1,\ldots,w_{r-1},w_i \in \mathbb{F}\}$[1] $\subset \mathbb{F}^n, i \in \{r+1,\ldots,n\}$.

Restricting a polynomial to W_{r-1} corresponds to plugging $x_r = \ldots = x_n = 0$ and restricting it to W_i corresponds to plugging $x_r = \ldots = x_{i-1} = x_{i+1} = \ldots = x_n = 0$. Restriction of any polynomial f to the subspace W_i will be denoted by $f_{|W_i}$ and that to W_{r-1} will be denoted by $f_{|W_{r-1}}$.

We also define V_i as vector space of linear forms in $x_1,\ldots,x_{r-1},x_i$ and V_{r-1} as the vector space of linear forms in $x_1,\ldots,x_{r-1}$.

Definition 23 (Restrictions of points in projective space)**.** Let $p \in \mathbb{P}(V)$ and consider any $l \in V$ such that $p = [l]$. When $l_{|W_i} \neq 0$ (resp. $l_{|W_{r-1}} \neq 0$) we say $p_{|W_i}$ (resp $p_{|W_{r-1}}$) is defined, and define $p_{|W_i}$ (resp $p_{|W_{r-1}}$) as the point $[l_{|W_i}] \in \mathbb{P}(V_i)$ (resp. $[l_{|W_{r-1}}] \in \mathbb{P}(V_{r-1})$). Otherwise we say $p_{|W_i}$ (resp $p_{|W_{r-1}}$) is undefined.

It can be seen that when $p_{|W_i}$ (resp $p_{|W_{r-1}}$) is defined, it is in fact well defined (i.e. if we choose l' instead of l we get the same points as restrictions).

Lemma 8. *Let $r \geq 4$ and consider subspaces defined in definition 22 above. The following hold with high probability:*

1. *Let $p \in \mathcal{I}(\mathbb{P}(R),\mathbb{P}(B))$ be a point, then $p_{|W_i}$ is defined and belongs to $\mathbb{P}(V_i)$ (see definition 23).*
2. *Let $p_1,\ldots,p_s \in \mathcal{I}(\mathbb{P}(R),\mathbb{P}(B))$, $s \leq r$ be s independent points (see definition 3), then $p_{1|W_i},\ldots,p_{s|W_i}$ are defined and are independent points in $\mathbb{P}(V_i)$.*
3. *Let $p_1,\ldots,p_s \in \mathcal{I}(\mathbb{P}(R),\mathbb{P}(B))$, $s \leq 3$ be s independent points, then the restrictions $p_{1|W_{r-1}},\ldots,p_{s|W_{r-1}}$ are defined and are independent points in $\mathbb{P}(V_{r-1})$.*

[1] The w_i is in the i^{th} location.

Proof. Apply corollary 2 in appendix A for multi-set $\mathcal{I}(\mathbb{P}(R),\mathbb{P}(B))$. Note that $|\mathcal{I}(\mathbb{P}(R),\mathbb{P}(B))|\leq d^4$, therefore success probability is $\geq 1-\frac{poly(n,r,d^{4r})}{|S|}$. We will assume $|S|>> \Omega(poly(n,r,d^{4r}))$ and make the probability $\geq 1-o(1)$. From now onwards we assume that the statements in this lemma are always true i.e. our Ω is always among the good cases. This is done for better exposition. □

4.3 Restricting the input polynomial

Let $r \geq 4$ be any integer. Consider the subspaces W_i, $i \in \{r+1,\ldots,n\}$ defined in definition 22. We restrict[2] $C(\bar{x})$ to W_i giving:

$$C(\bar{x})_{|W_i} = M_{1|W_i} + M_{2|W_i}$$

Clearly this restriction is also a $\Sigma\Pi\Sigma(2)$ polynomial. We denote the above $\Sigma\Pi\Sigma(2)$ circuit computing it as $C_{|W_i}$. We can further factorize using definition 17.

$$C(\bar{x})_{|W_i} = Gcd(C_{|W_i})Sim(C_{|W_i}) = Gcd(C_{|W_i})Int(C_{|W_i})Res(C_{|W_i})$$

Lemma 9. *The following are true (up to multiplication by a scalar):*

1. $gcd(R_{|W_i},B_{|W_i}) = 1 \Rightarrow Sim(C_{|W_i}) = R_{|W_i} + B_{|W_i}$.
2. $srank(C_{|W_i}) = min(r,srank(C))$.
3. $Res(C_{|W_i}) = Res(C)_{|W_i}$ *with high probability.*

Proof. The proof is routine and long. See lemma 26 in appendix D. To not break continuity we urge the reader to believe the lemma and verify it later. □

Remark 2. *We make a few remarks:*

1. *Note that we have already computed a black-box $\mathbb{B}_{Res}$ computing the polynomial $Res(C)$ using algorithm C in appendix C. Part 3 in lemma 9 tells us that $Res(C_{|W_i}) = Res(C)_{|W_i}$, and so by feeding the black-box $\mathbb{B}_{Res}$ inputs from W_i we can obtain a black-box for $Res(C_{|W_i})$.*
2. *Assuming $srank(C) > r = R(3,\mathbb{F})+2$, in part 2 of lemma 9 we can say that $rank(C_{|W_j}) = r = R(3,\mathbb{F})+2$. Then we can use theorem 5 for this circuit and obtain $\mathcal{P}(Res(C_{|W_i})) \subset \mathcal{I}(R_{|W_i},B_{|W_i})$. Also part 1 of lemma 9 and theorem 5 together imply that all projective linear forms in $\mathbb{P}(R_{|W_i}),\mathbb{P}(B_{|W_i})$ are in $\mathcal{P}(Res(C_{|W_i}))$.*

[2] This just means plugging $x_r = \ldots = x_{i-1} = x_{i+1} = \ldots = x_n = 0$

Due to these reasons we try to compute $\mathcal{P}(Res(C_{|W_i}))$ using the black-box for $Res(C_{|W_i})$ described above. For shorthand we define $\mathcal{P}_i = \mathcal{P}(Res(C_{|W_i}))$.

4.4 Computing the sets $\mathcal{P}_i$.

To efficiently compute $\mathcal{P}_i$, we use brill's equations which completely characterize coefficients of polynomials expressible as product of linear forms. We discuss them in great detail in appendix B. Let's first give the main result about these equations and then an algorithm to compute the sets $\mathcal{P}_i$.

The lemma below says that there exists a family of polynomials $\mathcal{F} = \{F_1, \ldots, F_m\}$ such that coefficients of all totally decomposable polynomials (i.e. product of linear forms) are given by the variety $\mathbb{V}(\mathcal{F})$. Also this family $\mathcal{F}$ can be computed in $poly(d^r)$ time as shown in appendix B.

We first compute coefficient representation for $Res(C_{|W_i})$ using part 1 in remark 2 i.e. we do Lagrange interpolation on $Res(C)_{|W_i}$. This can be done by first restricting the black-box $\mathbb{B}_{Res}$ to W_i (by feeding inputs only from W_i) and then by using $O(d^r)$ points from W_i to interpolate. Next we consider any projective linear form in the r variables $x_1, \ldots, x_{r-1}, x_i$. Say the coefficient of x_j is non-zero $\Rightarrow$ we may write a linear form corresponding to this projective form as

$$x_j - \sum_{k \in I \setminus \{j\}} z_k x_k$$

(here I is the set $\{1, \ldots, r-1, i\}$) and then substitute for x_j in the polynomial $Res(C_{|W_i})$. We collect the coefficients of this polynomial as polynomials in the z_k's and use them as input into the variety $\mathbb{V}(\mathcal{F})$ described above. The solutions for these z_k's can then be obtained using any algorithm to compute roots e.g. buchberger's algorithm ([5]). Since $r = \Omega(1)$, this algorithm works in $poly(d)$ time as explained below.

Lemma 10 (Brill's equations, See corollary 3 in appendixB)**.** *Let $\mathbb{F} = \mathbb{C}$, and s, d be positive integers. Define the indexing set*

$$\Lambda := \{\lambda = (\lambda_1, \ldots, \lambda_s) : \lambda_i \geq 0 \text{ for all } i, \sum_{i \in [s]} \lambda_i \leq d\}$$

Define $t := |\Lambda| = \binom{s+d}{d}$. We know that Λ can be used to index the coefficients of any multivariate polynomial of degree d in s variables. For any coefficient vector $\boldsymbol{a} = (a_\lambda)_{\lambda \in \Lambda}$ we have the polynomial

$$f_{\boldsymbol{a}}(x_1, \ldots, x_s) = \sum_{\lambda = (\lambda_1, \ldots, \lambda_s) \in \Lambda} a_\lambda x_1^{\lambda_1} \ldots x_s^{\lambda_s}$$

There exists an explicit set of polynomials $F_1, \ldots, F_m \in \mathbb{C}[y_1, \ldots, y_t]$ *with* $m = poly(d)$*, such that*

$$f_{\boldsymbol{a}}(x_1, \ldots, x_s) \text{ is totally decomposable } \Leftrightarrow F_1(\boldsymbol{a}) = \ldots = F_m(\boldsymbol{a}) = 0$$

Also this set $\{F_1, \ldots, F_m\}$ *can be computed in* $poly(t, m)$ *time.*

Algorithm. Recall that we've already computed a black-box $\mathbb{B}_{Res}$ for $Res(C)$ using algorithm C. Here is the algorithm to compute $\mathcal{P}_i, k \in \{r, \ldots, n\}$ using these black-boxes:

for *each* $i \in \{r, \ldots, n\}$ **do**

Initialize $\mathcal{P}_i = \phi$.

Using part 1 in remark 2 and Lagrange interpolation, compute $Res(C_{|_{W_i}})$ in coefficient form.

for *each* $j \in I = \{1, \ldots, r-1, i\}$ **do**

Let $\{z_k\}_{k \in I \setminus j}$ be variables.

Substitute $x_j = \sum_{k \in I \setminus \{j\}} z_k x_k$ in $Res(C_{|_{W_i}})$ and compute the coefficient polynomials (in $\{z_k\}_{k \in I \setminus \{j\}}$) corresponding to monomials in the variables $\bar{x}$.

Let **a** be the vector of coefficient polynomials calculated above.

Solve polynomial system $\{F_j(\mathbf{a}) = 0, j \in [m]\}$ in $\mathbb{C}$, using Buchberger's Algorithm ([5]).

If a solution $(z_k)_{k \in I \setminus \{j\}}$ belongs to $\mathbb{F}^{r-1}$, add the form corresponding form $x_j - \sum_{k \in I \setminus \{j\} z_k x_k}$ as a projective form to the set $\mathcal{P}_i$.

end

end

Return the sets $\mathcal{P}_r, \mathcal{P}_{r+1}, \ldots, \mathcal{P}_n$.

Algorithm 2: Computing sets $\mathcal{P}_i = \mathcal{P}(Res(C_{|_{W_i}}))$

Note that since the number of solutions were $O(d^4)$, the polynomial system has at most these many solutions and using Buchberger's algorithm we can find all of them. Now let's look at the time complexity of this algorithm.

- When we substitute and expand any monomial in $Res(C_{|W_i})$, we spend $O(d^r)$ time in computing coefficients of all monomials in the $\bar{x}$ variables.
- Buchberger's algorithm takes $O(d^{2^r})$ time (See [5]).

Rest of the steps are $poly(r,d)$ time. So overall we take $O(d^{2^r})$ time. Since r was set to be a constant ($= R(3,\mathbb{F}) + 2$) in remark 2, the time taken is $poly(d)$. We will perform this for all $i \in \{r,\ldots,n\}$ and thus in $poly(d,n)$ time we would have computed our all the sets $\mathcal{P}_i = \mathcal{P}(Res(C_{|W_i}))$. Now it's time to glue them together and compute the candidate set $\mathcal{P}$.

4.5 Gluing $\mathcal{P}_i$'s to compute $\mathcal{P}$

We start with the following observation:

Observation 1. Consider projective linear forms $p_r \in \mathcal{P}_r$ and $p_i \in \mathcal{P}_i, i \in \{r+1,\ldots,n\}$ such that for linear forms $l_r = \sum_{j=1}^{r-1} \alpha_j x_j + \alpha_r x_r$, $l_i = \sum_{j=1}^{r-1} \beta_j x_j + \beta_i x_i$ with $p_r = [l_r]$ and $p_i = [l_i]$ and

$$l_{r|_{W_{r-1}}} \sim l_{i|_{W_{r-1}}} (\neq 0).$$

Then there exists a linear form l in variables $x_1,\ldots,x_r,x_i$ which is a lift of both forms l_r and l_i. This can be seen as follows. Clearly there is some $j \in \{1,\ldots,r-1\}$ such that $\alpha_j \neq 0$. Without loss of generality let's assume $j = 1$ (note that this also implies that $\beta_1 \neq 0$), then we define l as

$$l = x_1 + \frac{\alpha_2}{\alpha_1} x_2 + \ldots \frac{\alpha_r}{\alpha_1} x_r + \frac{\beta_i}{\beta_1} x_i.$$

and $p = [l]$ can be seen (or defined) to be a lift on (p_r, p_i). Note that by definition 23, instead of saying $l_{r|_{W_{r-1}}} \sim l_{i|_{W_{r-1}}}$ we could have said that $p_{r|_{W_{r-1}}}, p_{i|_{W_{r-1}}}$ are defined and are equal points in $\mathbb{P}(V_{r-1})$.

Definition 24. A pair of projective linear forms $(p_r, p_i) \in \mathcal{P}_r \times \mathcal{P}_i$ are called *"gluable"* if $p_{r|_{W_{r-1}}}, p_{i|_{W_{r-1}}}$ are defined and are equal points in $\mathbb{P}(V_{r-1})$.

For $p_r \in \mathcal{P}_r$ our algorithm tries to find $p_i \in \mathcal{P}_i, i \in \{r+1,\ldots,n\}$ such that (p_r, p_i) are gluable. If we can successfully find such p_i's for every $i \in \{r+1,\ldots,n\}$ then gluing p_r with all of them gives us a projective linear form p in variables $x_1,\ldots,x_n$.

For our algorithm to work in polynomial time and recover $\mathcal{P}$ using gluing we need to make sure of two requirement:

Lemma 11. *The following hold:*

1. *For every $p \in \mathcal{P}$ and $i \in \{r, \ldots, n\}$, $p_{|W_i} \in \mathcal{P}_i$.*
2. *For any $p_r \in \mathcal{P}_r$, there is at-most one $p_i \in \mathcal{P}_i, i \in \{r+1, \ldots, n\}$ it can be glued to.*

Proof. For cleaner exposition we move these proofs to appendix D. □

To not break continuity, we urge the reader to continue reading at this moment and verify the proof later. Now we are ready to give the gluing algorithm and recover the set $\mathcal{P}$. Let's summarize it in the following lemma

Lemma 12. *Given sets $\mathcal{P}_i$ and black-box $\mathbb{B}_{Res}$, we give a randomized algorithm that runs in time $poly(n, d)$ and outputs a set $\tilde{\mathcal{P}} \subset \mathbb{P}(V)$ such that*

$$Pr[\tilde{\mathcal{P}} = \mathcal{P}] \geq 1 - o(1).$$

Proof. Let's first give the algorithm and then discuss it's correctness and time complexity.

Initialize $\tilde{\mathcal{P}} = \phi$.

for $p_r \in \mathcal{P}_r$ **do**

Pick a linear form $l_r = \alpha_1 x_1 + \ldots + \alpha_{r-1} x_{r-1} + \alpha_r x_r$ such that $p_r = [l_r]$.

Initialize linear form $\hat{l} = l_r$.

for $i = r + 1$ *to* n **do**

Find $p_i \in \mathcal{P}_i$ such that (p_r, p_i) are *"gluable"*.

If more than one p_i exist or no such p_i exists then discard p_r and break out of this loop.

Else pick a linear form l_i such that $p_i = [l_i]$.

Find $\beta \in \mathbb{F}$ such that $\beta l_i = \alpha_1 x_1 + \ldots + \alpha_{r-1} + \alpha_i x_i$.

Update $\hat{l} = \hat{l} + \alpha_i x_i$.

end

If $\hat{l} = \alpha_1 x_1 + \ldots + \alpha_n x_n$, define $p = [\alpha_1 x_1 + \ldots + \alpha_n x_n] \in \mathbb{P}(V)$.

Compute $Res(C)_{|_{ker(p)}}$ by restricting the black-box $\mathbb{B}_{Res}$.

Check if this restriction factors into a product of linear factors using algorithm C and randomized black-box PIT algorithm (Schwartz Zippel lemma).

If yes then add p to $\tilde{\mathcal{P}}$.

end

Return $\tilde{\mathcal{P}}$.

Algorithm 3: Gluing the $\mathcal{P}_i$'s

Correctness Proof - ($\tilde{\mathcal{P}} \subset \mathcal{P}$) Consider any $p \in \tilde{\mathcal{P}}$. Just before the end of the outer for loop using algorithm C we check whether $Res(C)_{|_{ker(p)}}$ factors as a product of linear factors or not. Algorithm C returns the multi-set of projective linear factors of $Res(C)$ (with substitution) and then using randomized black-box polynomial identity testing(Schwarz-Zippel lemma) we can check whether the product computes the same polynomial as $\mathbb{B}_{Res}$ or not. So with probability $1 - o(1)$ we will

be right in checking whether p is a *"factoring form"* of $Res(C)$ or not and thus with probability $1 - o(1)$, $p \in \mathcal{P}$. Since the size of $\tilde{\mathcal{P}}$ is less than d^4, with probability $1 - o(1)$, $\tilde{\mathcal{P}} \subset \mathcal{P}$.

($\mathcal{P} \subset \tilde{\mathcal{P}}$) Let $p \in \mathcal{P}$. By part 1 of lemma 11 we know that $p_{|W_i} \in \mathcal{P}_i$ for every $i \in \{r, \ldots, n\}$. Thus the outer for loop is called for $p_{|W_r}$ at some point of time. By part 2 of lemma 11, there is only one form $p_{|W_i}$ in each $\mathcal{P}_i, i \in \{r+1, \ldots, n\}$ gluable to $p_{|W_r}$ and thus we glue $p_{|W_r}$ with each $p_{|W_i}, i \in \{r+1, \ldots, n\}$ and form p. Since p is a factoring form with probability $\geq 1 - o(1)$ it passes all the checks after the for loop and thus gets included in $\tilde{\mathcal{P}}$. So with probability $1 - o(1)$ $p \in \tilde{\mathcal{P}}$. Since $\mathcal{P}$ has size $\leq d^4$, with probability $1 - o(1)$, $\tilde{\mathcal{P}} \subset \mathcal{P}$

Time Complexity - Since $|\mathcal{P}_r| = poly(d)$, the loops run $poly(n, d)$ times. We use algorithm C which takes $poly(n, d)$ time. All other steps can easily be seen to be $poly(n, d)$ time.

□

Chapter 5

STEP TWO : RECONSTRUCT LAYER II OF C

For this chapter we fix $\delta \in (0, \frac{1}{8})$, $k = max(R(3,\mathbb{F}) + 2, R(4,\mathbb{F}))$ and N_0 to be a constant $> \alpha\frac{k}{\delta} + 2k$ where α is the constant in theorem 13.

5.1 Introduction

Recall that we are trying to reconstruct the $\Sigma\Pi\Sigma(2)$ structure of a polynomial (given as a black-box) $C(\bar{x}) = M_1 + M_2$ with M_1, M_2 products of linear forms. In chapter 4, we gave efficient randomized algorithms to compute the following:

1. The set $\mathcal{P}$ of *"factoring forms"* of polynomial $Res(C)$. (See definition 19 for definition of factoring form)
2. The multi-set $\mathbb{P}(Gcd(C)Int(C))$ containing the projective linear factors of $Gcd(C)Int(C)$. These are same as projective linear factors of polynomial $C(\bar{x})$.
3. A black-box $\mathbb{B}_{Res}$ computing the polynomial $Res(C)$.

(See definition 17 for definitions of $Int(C), Res(C)$).

In this chapter we will use all of the above and compute the polynomials M_1, M_2 as lists of linear forms finishing our reconstruction job.

Let's summarize the results of this chapter in the following theorem. It uses definition 17 and 19 from chapter 2)

Theorem 8. *Let C be a homogeneous $\Sigma\Pi\Sigma(2)$ circuit computing a degree d polynomial $C(\bar{x})$ in n variables $x_1, \ldots, x_n$. Assume that $srank(C) \geq N_0$ (defined at beginning of this chapter). Further assume we have access to the following:*

- *parameters n, d.*
- *black-boxes computing polynomials $C(\bar{x})$ and $Res(C)$.*

– *multi-set* $\mathbb{P}(Gcd(C)Int(C))$ *as an explicit list of projective linear forms.*

– *set* $\mathcal{P}(Res(C))$ *as an explicit list of projective linear forms.*

We give a randomized algorithm that runs in time $poly(n,d)$ *and with probability* $1 - o(1)$ *outputs (as explicit lists of linear forms) two polynomials* $M_1(\bar{x}), M_2(\bar{x})$ *which are both products of linear forms such that*

$$C(\bar{x}) = M_1(\bar{x}) + M_2(\bar{x})$$

5.2 Lines connecting forms in $\mathbb{P}(R)$ ($\mathbb{P}(B)$ resp.) to $\mathbb{P}(M_2)$($\mathbb{P}(M_1)$ resp.)

Recall definition 11 which defines $\mathcal{L}(p,\mathcal{S})$ as the multi-set of lines connecting p and $\mathcal{S}$, where $p \in \mathbb{P}(V)$ is a projective linear form and $\mathcal{S} \subset \mathbb{P}(V)$ is a multi-set of projective linear forms.

Lemma 13. *Given multi-sets* $\mathbb{P}(Gcd(C)Int(C))$*, black-box* $\mathbb{B}_{Res}$ *(computing polynomial* $Res(C)$*) and a form* $r \in \mathbb{P}(R)$*, there is a* $poly(n,d)$ *time algorithm to compute the multi-set of lines* $\mathcal{L}(r,\mathbb{P}(M_2))$ *(Recall that* $\mathbb{P}(M_2) = \mathbb{P}(B) \cup \mathbb{P}(Gcd(C))$*, where* $\cup$ *denotes the multi-set union).*

Proof. Let r^e be the largest power of r dividing $Gcd(C)Int(C)$. This implies that r^e divides $Gcd(C)$ (By lemma 4 we know that $\mathbb{P}(R) \cap \mathbb{P}(Int(C)) = \phi$). Thus $Gcd(C) = r^e G'$ and

$$\mathbb{P}(Gcd(C)Int(C)) = \underbrace{\{r,\ldots,r\}}_{e \text{ times}} \cup \mathbb{P}(G') \cup \mathbb{P}(Int(C)).$$

So we iterate through $\mathbb{P}(Gcd(C)Int(C))$ and remove all occurrences of r (this takes $poly(n,d)$ time). Then we will be left with $\mathbb{P}(G') \cup \mathbb{P}(Int(C))$. Note that

$$G'Int(C)Res(C) = G'(R+B).$$

On restricting both sides to the hyperplane $ker(r)$ (see definition 8) we get

$$G'_{|_{ker(r)}} Int(C)_{|_{ker(r)}} Res(C)_{|_{ker(r)}} = G'_{|_{ker(r)}} B_{|_{ker(r)}}. \tag{5.1}$$

Both sides are non-zero since r does not divide G', B. Now by lemma 2, for any $p \neq r$ and $l \in V$ such that $p = [l]$, line joining r,p is the same as line joining $r,[l_{|_{ker(r)}}]$.The multi-set of lines joining r with $\mathbb{P}(M_2)$ then becomes

$$\mathcal{L}(r,\mathbb{P}(M_2)) = \{fl(r,p) : p \text{ divides } G'_{|_{ker(r)}} B_{|_{ker(r)}}\}, \quad \text{since } M_2 = r^e G'B. \tag{5.2}$$

and so using the above two equations we get,

$$\mathcal{L}(r,\mathbb{P}(M_2)) = \{fl(r,p) : p \text{ divides } G'_{|_{ker(r)}} Int(C)_{|_{ker(r)}} Res(C)_{|_{ker(r)}}\}. \quad (5.3)$$

$$= \{fl(r,p) : p \in \mathbb{P}(G'Int(C))\} \cup \{fl(r,p) : p \text{ divides } Res(C)_{|_{ker(r)}}\}$$

We know the multi-set of $\mathbb{P}(G'Int(C))$ (by removing instances of r from $\mathbb{P}(Gcd(C)Int(C))$) and so we can compute the lines $\{fl(r,p) : p \in \mathbb{P}(G'Int(C))\}$. Using black-box $\mathbb{B}_{Res}$ for $Res(C)$ we can compute black-box for $Res(C)_{|_{ker(r)}}$ (by feeding the black-box inputs from $ker(r)$) and then factorize it using algorithm C. So we also have all the projective linear factors of $Res(C)_{|_{ker(r)}}$ and we can find all the lines joining r to it's factors. This process clearly takes $poly(n,d)$ time since algorithm C runs in $poly(n,d)$ time and there are less than or equal to d factors $\Rightarrow$ forming the lines takes $poly(n,d)$ time. Equation 5.3 implies that this process computes the set $\mathcal{L}(r,\mathbb{P}(M_2))$. □

We summarize the whole algorithm below.

Initialize $\mathcal{L}(r,\mathbb{P}(M_2)) = \phi$.

for *each* $p(\neq r) \in \mathbb{P}(Gcd(C)Int(C))$ **do**

  Add the line $fl(r,p)$ to $\mathcal{L}(r,\mathbb{P}(M_2))$.

end

Using algorithm C in appendix C and the black-box $\mathbb{B}_{Res}$ for $Res(C)$, compute the multi-set of factors $\mathbb{P}(f)$ (see definition 10) where $f = Res(C)_{|_{ker(r)}}$. f is a product of linear forms since r is a factoring form.

for *each projective linear form p in* $\mathbb{P}(f)$ **do**

  Add the line $fl(r,p)$ to $\mathcal{L}(r,\mathbb{P}(M_2))$.

end

Return $\mathcal{L}(r,\mathbb{P}(M_2))$.

Algorithm 4: Find connecting Lines

Correctness Proof - Same as proof of the lemma above.

Time Complexity - Both the for loops run $poly(n,d)$ times and algorithm C takes $poly(n,d)$ time. So the overall complexity is $poly(n,d)$.

Even though we gave the algorithm only for $r \in \mathbb{P}(R)$, a similar algorithm works for $b \in \mathbb{P}(B)$ computing the set of lines $\mathcal{L}(b,\mathbb{P}(M_1))$.

5.3 Termination Case : Reconstructing one of $\mathbb{P}(M_1),\mathbb{P}(M_2)$ does the job

We discuss a situation which arises at the end of each reconstruction algorithm in this chapter. Assume we have been able to reconstruct one of the two multi-sets $\mathbb{P}(M_1),\mathbb{P}(M_2)$. Note that these multi-sets contain projective linear factors of M_1, M_2 respectively. We will give an algorithm to reconstruct both M_1, M_2 as explicit lists of their respective linear factors. Without loss of generality let's assume we know the multi-set $\mathbb{P}(M_1)$. We summarize this in the following lemma.

Lemma 14. *Let C be a homogeneous $\Sigma\Pi\Sigma(2)$ circuit computing a degree d polynomial $C(\bar{x}) = M_1 + M_2$, where both M_1, M_2 are products of d linear forms in n variables $x_1,\ldots,x_n$. Assume that $srank(C) \geq N_0$ (defined at beginning of this chapter). Further assume that we are given a set $\mathcal{S}$ of projective linear forms. We give an algorithm that works in time $poly(n,d)$ and does the following:*

- *If $\mathcal{S} \neq \mathbb{P}(M_1)$ and $\mathcal{S} \neq \mathbb{P}(M_2)$, it outputs* "fail".
- *If $\mathcal{S} = \mathbb{P}(M_1)$ or $\mathcal{S} = \mathbb{P}(M_2)$ with probability $1 - o(1)$ it outputs two products (of linear forms) M_1, M_2 (each given as a list of its linear factors) such that $C(\bar{x}) = M_1 + M_2$. When it does not output the two products it outputs* "fail".

Proof. We first give the algorithm and then talk about it's correctness and time

complexity. Here is the algorithm:

1. Let $\mathcal{S} = \{p_1, \ldots, p_d\}$. Fix linear forms l_i such that $p_i = [l_i]$.

2. Restrict black-box $\mathbb{B}$ to $W_r = \{(w_1, \ldots, w_r, 0, \ldots, 0)\}$ of dimension $r = R(4, \mathbb{F})$ and interpolate using $O(d^r)$ points from W_r to get coefficient representation of $C_{|W_r}$.

3. For $i \in [d]$, compute $l_{i|_{W_r}}$ by plugging $x_{r+1} = \ldots = x_n = 0$. Compute the polynomial $M(\bar{x}) = l_{1|_{W_r}} \ldots l_{d|_{W_r}}$ as a list of coefficients by multiplying out all linear forms.

4. Let α be a variable and compute the coefficient vector of the polynomial $C(\bar{x}) - \alpha M(\bar{x})$. Note that every entry of the vector is a degree 1 polynomial in α.

5. Plug the coefficient vector of $C(\bar{x}) - \alpha M(\bar{x})$ into the system of polynomials given by brill's equations (see lemma 10) giving a polynomial system in one variable α.

6. Using your favorite exact root finding algorithm for univariates, solve for α. If no (or multiple) α is found output *"fail"*.

7. Else using algorithm C compute all projective linear factors (explicitly) of $C(\bar{x}) - \alpha l_1 \ldots l_d$ (use black-boxes for $C(\bar{x})$ and $l_1 \ldots l_d$). If there are d such forms(with multiplicity), store them in a multi-set $\mathcal{S}'$. Let $\mathcal{S}' = \{l_1', \ldots, l_d'\}$ and repeat all previous steps $\mathcal{S}'$ instead of $\mathcal{S}$ and compute all possible α' such that $C_{|W_r} - \alpha' l'_{1|_{W_r}} \ldots l'_{d|_{W_r}}$ is a product of linear forms. Using deterministic black-box polynomial identity testing algorithm for $\Sigma\Pi\Sigma(4)$ circuits, if $C(\bar{x}) - \alpha l_1 \ldots l_d - \alpha' l_1' \ldots l_d'$ is an identically zero polynomial.

8. If yes then we return the gates of $C(\bar{x})$ as

$$M_1 = \alpha l_1 \ldots l_d \quad \text{and} \quad M_2 = \alpha' l_1' \ldots l_d'.$$

   Output is given as two lists of linear forms $\{\alpha l_1, l_2, \ldots, l_d\}$ and $\alpha' l_1', l_2', \ldots, l_d'$.

9. If for none of the α's, we were able to reconstruct, output *"fail"*.

Algorithm 5: Reconstruction using one of $\mathbb{P}(M_1), \mathbb{P}(M_2)$

Correctness Proof - First let's consider the case when $\mathcal{S} \neq \mathbb{P}(M_1)$ and $\mathcal{S} \neq \mathbb{P}(M_2)$. Assume that $\mathcal{S} = \{p_1, \ldots, p_d\}$ and l_i are such that $p_i = [l_i]$. We show that the algorithm outputs *"fail"*. If not then the algorithm would have returned lists of linear forms and before that it would have checked using deterministic polynomial identity testing algorithm (in [25]) for $\Sigma\Pi\Sigma(4)$ circuits whether $C(\bar{x}) - \alpha l_1 \ldots l_d - \alpha' l'_1 \ldots l'_d = 0$. But this implies that $\mathcal{S}$ is actually one of the multi-sets $\mathbb{P}(M_1), \mathbb{P}(M_2)$. So we have a contradiction and the algorithm outputs *"fail"*.

For the other part without loss of generality assume that $\mathcal{S} = \mathbb{P}(M_1)$. By step 1 we would have assumed $\mathcal{S} = \{p_1, \ldots, p_t\}$ and fixed linear forms l_i such that $p_i = [l_i]$. Since $\mathcal{S} = \mathbb{P}(M_1)$ we know there exists $\alpha \in \mathbb{F}$ such that $C(\bar{x}) = \alpha l_1 \ldots l_t + M_2$, where M_2 is a product of linear forms. Also $srank(C) \geq R(4, \mathbb{F}) \Rightarrow$ (by theorem 4) that the circuit is unique and thus α is unique. Part 2 in lemma 9 implies that with probability $1 - o(1)$, $srank(C_{|_{W_r}}) = r = R(4, \mathbb{F})$, further implying that $\Sigma\Pi\Sigma(2)$ structure of $C(\bar{x})_{|_{W_r}}$ will be unique. This shows that there is a unique α such that $C(\bar{x})_{|_{W_r}} - \alpha l_{1|_{W_r}} \ldots l_{t|_{W_r}}$ is a product of linear forms. So the coefficients (in terms of α) of $C(\bar{x})_{|_{W_r}} - \alpha l_{1|_{W_r}} \ldots l_{t|_{W_r}}$ will satisfy brill's equations (see appendix B) and on solving them we would be able to determine the unique α exactly. If an α was found then $M_2 = C(\bar{x}) - \alpha l_1 \ldots l_t$ factors into a product of linear forms. To find these linear forms we used the factoring algorithm C which outputs a multi-set of projective linear forms. With high probability this multi-set will be $\mathbb{P}(M_2)$ since with probability $1 - o(1)$ the algorithm C outputs the correct factors. Again on repeating steps $1 - 6$ above with this multi-set $\mathbb{P}(M_2)$ we would be able to compute (with probability $1 - o(1)$) the unique α' such that $C(\bar{x}) - \alpha' l'_1 \ldots l'_t$ is a product of linear forms. So till now we would have computed two products M_1, M_2 such that with probability $1 - o(1)$, $C(\bar{x}) = M_1 + M_2$. We change this into only one sided error i.e. make sure that if we output M_1, M_2 they are always right by using the deterministic $\Sigma\Pi\Sigma(4)$ polynomial identity testing algorithm in [25]. Therefore in this case with probability $1 - o(1)$ we will output M_1, M_2 and they will be correct.

Time Complexity - Steps $1-4$ are clearly polynomial time since r is constant. Step 5 and 6 involve solving $poly(d)$ polynomials in one variable. This can be easily done by solving one of them and satisfying the rest. This clearly takes $poly(d)$ time. To solve one equation we could factorize into irreducibles over $\mathbb{F}$ and look for linear factors. This is also polynomial time. Step 7 invokes algorithm C, repeats steps $1-6$ and performs the polynomial time deterministic PIT for $\Sigma\Pi\Sigma(4)$ circuits, thus taking $poly(n, d)$ time. Therefore the algorithm takes $poly(n, d)$ time overall.

□

5.4 One of the multi-sets $\mathbb{P}(R), \mathbb{P}(B)$ is low dimensional

Recall that we've assumed $srank(C) = dim(\mathbb{P}(R) \cup \mathbb{P}(B)) \geq N_0$ (defined at the beginning of this chapter). In this section we show that our task is much easier if any of the two sets $\mathbb{P}(R), \mathbb{P}(B)$ has dimension $< N_0 - 1$. Without loss of generality we assume $dim(\mathbb{P}(B)) < N_0 - 1$. Since $dim(\mathbb{P}(R) \cup \mathbb{P}(B)) \geq N_0$, there exist independent $r_1, r_2 \in \mathbb{P}(R)$ such that $fl(r_1, r_2) \cap fl(\mathbb{P}(B)) = \phi$.

Our very first result in this case will make life a lot simpler. We claim that the polynomial $Int(C)$ is a constant i.e. the only linear factors of C are the ones dividing $Gcd(C)$, in other words the multi-set $\mathbb{P}(Int(C))$ is empty and $\mathbb{P}(Gcd(C))$ is equal to $\mathbb{P}(Gcd(C)Int(C))$, which has already been computed. This also means $\mathbb{B}_{Res}$ (to which we have access) is a black-box for $Sim(C)$.

Claim 1. $\mathbb{P}(Int(C)) = \phi$.

Proof. Assume $p \in \mathbb{P}(Int(C))$. Then $R+B = Int(C)Res(C) \Rightarrow R_{|ker(p)} = -B_{|ker(p)}$. Using lemma 4 we know that $\mathbb{P}(R) \cap \mathbb{P}(Int(C)) = \phi$ and thus both $R_{|ker(p)}, B_{|ker(p)}$ are non-zero.

By unique factorization there exist B_1, B_2 dividing B such that $r_{1|ker(p)}, B_{1|ker(p)}$ are scalar multiples and $r_{2|ker(p)}, B_{2|ker(p)}$ are scalar multiples. A little bit of manipulation implies that $r_1 \in fl(p, [B_1])$. Similarly $r_2 \in fl(p, [b_2])$. We can use these two to eliminate p and conclude that $fl(r_1, r_2) \cap fl(b_1, b_2) \neq \phi$ which contradicts the choice of r_1, r_2. □

Now we need to reconstruct $\mathbb{P}(R), \mathbb{P}(B)$. The form r_1 (that we chose above) outside $\mathbb{P}(B)$ turns out to be a good form (see definition 20) and helps us recover the entire $\mathbb{P}(B)$. Please see section 2.4 for better understanding of the following lemma and algorithm.

Lemma 15. *There exists $r_1 \in \mathbb{P}(R)$ and $r_2 \in \mathbb{P}(R)$ (different from r_1) such that r_1 is a good form for $(\{r_2\}, \mathbb{P}(B))$ and the* "reconstructed multi-set" *(see definition 21) in this case $\mathbb{P}(B)_{r_1} = \mathbb{P}(B)$.*

Proof. Let r_1, r_2 be as chosen at the beginning of this section (also used in previous claim). Clearly $r_1 \neq r_2$. Also $fl(r_1, r_2) \cap fl(\mathbb{P}(B)) = \phi$ by choice of r_1, r_2. Therefore by lemma 6 we see that r_1 is a good form for $(\{r_2\}, \mathbb{P}(B))$ and the reconstructed multi-set is $\mathbb{P}(B)_{r_1} = \mathbb{P}(B)$. □

Now we are ready to give the reconstruction algorithm in this case. We summarize it in the following lemma.

Lemma 16. *Let C be a homogeneous $\Sigma\Pi\Sigma(2)$ circuit computing a degree d polynomial $C(\bar{x})$ in n variables $x_1, \ldots, x_n$. Assume that $srank(C) \geq N_0$. Also assume we have access to the following:*

- *parameters n, d.*
- *black-boxes computing polynomials $C(\bar{x})$ and $Res(C)$.*
- *multi-set $\mathbb{P}(Gcd(C)Int(C))$ as an explicit list of projective linear forms.*
- *set $\mathcal{P}(Res(C))$ as an explicit list of projective linear forms.*

Further assume that $dim(\mathbb{P}(B)) < N_0 - 1$. We give a randomized algorithm that runs in time $poly(n, d)$ and with probability $1 - o(1)$ outputs (as explicit lists of linear forms) two polynomials $M_1(\bar{x}), M_2(\bar{x})$ which are both products of linear forms such that

$$C(\bar{x}) = M_1(\bar{x}) + M_2(\bar{x})$$

Proof. Let's first give the algorithm and then discuss correctness and time complexity.

for *all pairs* $(r_1, r_2) \in \mathcal{P} \times \mathcal{P}$ **do**

Using algorithm 4, compute the multi-sets $\mathcal{L}(r_i, \mathbb{P}(M_2))$. From this remove all lines in $\mathcal{L}(r_i, \mathbb{P}(Gcd(C)))$ (Note that $\mathbb{P}(Gcd(C), Int(C)) = \mathbb{P}(Gcd(C))$ in this case). This gives multi-sets of lines $\mathcal{L}(r_i, \mathbb{P}(B)), i \in [2]$.

Using these multi-sets $\mathcal{L}(r_i, \mathbb{P}(B))$ and forms r_1, r_2 as input to algorithm 1 compute multi-set $\mathbb{P}(B)_{r_1}$ i.e. the *"reconstructed multi-set"*.

Combine multi-sets $\mathbb{P}(B)_{r_1}$ and $\mathbb{P}(Gcd(C))$. Send this new multi-set (guessing that it is $\mathbb{P}(M_2)$) as input to algorithm 5 and try to compute M_1, M_2.

If successful return M_1, M_2. Else continue.

end

Return *"fail"*.

Algorithm 6: $\mathbb{P}(B)$ is low dimensional

Correctness Proof - Suppose that the algorithm returns *"fail"*. We know that the outer most for loop will at some point of time use r_1, r_2 mentioned in lemma 15. For this choice we know that the reconstructed multi-set is $\mathbb{P}(B)$ which is actually reconstructed by algorithm 1. Therefore the input to algorithm 5 was actually $\mathbb{P}(M_2)$ but it did not reconstruct M_1, M_2. The probability that this could happen is small. Therefore the probability that the algorithm outputs *"fail"* is small.

Suppose the algorithm returns two polynomials M_1, M_2 (as explicit lists of linear forms), then by the correctness of algorithm 5, it ought to be correct since that is exactly the step which returns the two products M_1, M_2 and always gives the right answer (uses deterministic polynomial identity testing at end).

Therefore with probability $1 - o(1)$ we output two polynomials M_1, M_2 (as lists of linear forms) such that $C(\bar{x}) = M_1 + M_2$.

Time Complexity - The for loop runs $poly(n, d)$ times. Each algorithm called runs in $poly(n, d)$ time as discussed in their proofs and therefore the algorithm takes $poly(n, d)$ time.

□

5.5 Both multi-sets $\mathbb{P}(R), \mathbb{P}(B)$ are high dimensional

Now we assume that $dim(\mathbb{P}(R)) \geq N_0 - 1$ and $dim(\mathbb{P}(B)) \geq N_0 - 1$. Define the sets $\mathcal{R} = supp(\mathbb{P}(R))$ and $\mathcal{B} = supp(\mathbb{P}(B))$[1]. Without loss of generality we also assume that $|\mathcal{R}| \geq |\mathcal{B}|$.

Define a function $v(\delta) = 3\delta - 4\delta^2$. Here is the main lemma for this section.

Lemma 17. *Let C be a homogeneous $\Sigma\Pi\Sigma(2)$ circuit computing a degree d polynomial $C(\bar{x})$ in n variables $x_1, \ldots, x_n$. Assume that $srank(C) \geq N_0$. Also assume we have access to the following:*

- *parameters n, d.*
- *black-boxes computing polynomials $C(\bar{x})$ and $Res(C)$.*
- *multi-set $\mathbb{P}(Gcd(C)Int(C))$ as an explicit list of projective linear forms.*
- *set $\mathcal{P}(Res(C))$ as an explicit list of projective linear forms.*

Assume that $dim(\mathbb{P}(R)) \geq N_0 - 1$ and $dim(\mathbb{P}(B)) \geq N_0 - 1$. We give a randomized algorithm that runs in time $poly(n, d)$ and with probability $1 - o(1)$ outputs (as explicit lists of linear forms) two polynomials $M_1(\bar{x}), M_2(\bar{x})$ which are both products of linear forms such that

$$C(\bar{x}) = M_1(\bar{x}) + M_2(\bar{x})$$

To prove this we follow a number of steps. But first we define a bunch of multi-sets that will be useful throughout this section. We will try to give an intuition behind most definitions. Then we will prove lower and upper bounds on the size of multi-sets we define. Along with this whenever required we will also talk about efficient computation of these multi-sets and related objects. Finally we give the proof of the above lemma accomplishing the reconstruction goal.

Definition 25. For every choice of elements $r_1, \ldots, r_k \in \mathcal{R}$ and $b_1, \ldots, b_k \in \mathcal{B}$, we define the sets

$$\mathcal{R}(r_1, \ldots, r_k) = \{r \in \mathcal{R} : r \notin fl(r_1, \ldots, r_k), \text{ and } fl(r_1, \ldots, r_k, r) \cap (\mathcal{R} \cup \mathcal{B}) \subset fl(r_1, \ldots, r_k) \cup \{r\}\}, \text{ and}$$

[1] For a multi-set $\mathcal{X}$, $supp(\mathcal{X})$ is the set of distinct elements from $\mathcal{X}$.

$$\mathcal{B}(b_1,\ldots,b_k) = \{b \in \mathcal{B} : b \notin fl(b_1,\ldots,b_k), \text{ and } fl(b_1,\ldots,b_k,b) \cap (\mathcal{R} \cup \mathcal{B}) \subset fl(b_1,\ldots,b_k) \cup \{b\}\}.$$

Basically these are points $r \in \mathcal{R}$ ($b \in \mathcal{B}$ resp.) lying outside $fl(r_1,\ldots,r_k)$ ($fl(b_1,\ldots,b_k)$ resp.) such that the flat r (b resp.) forms with $r_1,\ldots,r_k$ ($b_1,\ldots,b_k$ resp.) is ordinary inside the set $\mathcal{R} \cup \mathcal{B}$ (for definition of ordinary flat see appendix F).

Definition 26. We also give names to the complements of $\mathcal{R}(r_1,\ldots,r_k)$ ($\mathcal{B}(b_1,\ldots,b_k)$ resp.) inside $\mathcal{R}$ ($\mathcal{B}$ resp.). Define

$$\mathcal{R}'(r_1,\ldots,r_k) = \mathcal{R} \setminus \mathcal{R}(r_1,\ldots,r_k) \quad \text{and} \quad \mathcal{B}'(b_1,\ldots,b_k) = \mathcal{B} \setminus \mathcal{B}(b_1,\ldots,b_k).$$

Next we define certain sub multi-sets of points in $\mathbb{P}(Gcd(C)Int(C))$. For any choice of independent points $p_1,\ldots,p_k$ we look at points $p \in \mathbb{P}(Gcd(C)Int(C))$ such that $p \notin fl(p_1,\ldots,p_k)$ and the flat $fl(p_1,\ldots,p_k,p) \setminus fl(p_1,\ldots,p_k)$ contains two distinct points from the set $\mathcal{P}$ we computed earlier in section 4.

Definition 27. Let $p_1,\ldots,p_k \in \mathbb{P}(V)$ be independent points, we define

$$\mathcal{G}(p_1,\ldots,p_k) = \{p \in \mathbb{P}(Gcd(C)Int(C)) : p \notin sp\{p_1,\ldots,p_k\}, \text{ and } |(fl(p_1,\ldots,p_k,p) \setminus fl(p_1,\ldots,p_k)) \cap \mathcal{P}| \geq 2\}. \quad (5.3)$$

We will be concerned with the above sets only when $p_1,\ldots,p_k$ are all inside $\mathbb{P}(R)$ or all inside $\mathbb{P}(B)$. The aim will be to show that for appropriate choices of $p_1,\ldots,p_k$, central projections of these multi-sets with respect to $fl(p_1,\ldots,p_k)$ has small size. We will need to compute lines going in to these sets and to make that work we show that whenever $\mathcal{R}(r_1,\ldots,r_k)$ ($\mathcal{B}(b_1,\ldots,b_k)$ resp.) is non-trivial then $\mathbb{P}(Int(C)) \subset \mathcal{G}(r_1,\ldots,r_k)$ ($\mathcal{G}(b_1,\ldots,b_k)$ resp.).

We define a subset of $\mathbb{P}(R)$ ($\mathbb{P}(B)$ resp.) and later give an efficient algorithm to compute this set. This set will give us points which along with an appropriate choice of $r_1,\ldots r_k$ ($b_1,\ldots,b_k$ resp.) will reconstruct points in $\mathbb{P}(M_2)$ ($\mathbb{P}(M_1)$ resp.).

Definition 28. Given independent points $r_1,\ldots,r_k \in \mathcal{R}$ and $b_1,\ldots,b_k \in \mathcal{B}$, we define the sets

$$\mathcal{S}(r_1,\ldots,r_k) = \{p \in \mathcal{P} : \forall\, i \in [k], fl(p,r_i) \cap (\mathcal{G}(r_1,\ldots,r_k) \cup \mathcal{B}) = \phi\}, \text{ and}$$

$$\mathcal{S}(b_1,\ldots,b_k) = \{p \in \mathcal{P} : \forall\, i \in [k], fl(p,b_i) \cap (\mathcal{G}(b_1,\ldots,b_k) \cup \mathcal{R}) = \phi\}.$$

Now we are ready to prove results about the multi-sets we defined. Almost all proofs are long and are given in the appendix. We suggest the reader to believe the lemmas and proceed to the main theorem and later come back to see the proofs.

Lemma 18. *The following holds:*

1. *If* $r_1,\ldots,r_k \in \mathcal{R}$, *then for any* $p \in \mathcal{G}(r_1,\ldots,r_k)$

   $(fl(r_1,\ldots,r_k,p)\backslash fl(r_1,\ldots,r_k))$ *and* $\mathcal{R}'(r_1,\ldots,r_k)\cup\mathcal{B}$ *intersect non-trivially.*

2. *If* $b_1,\ldots,b_k \in \mathcal{B}$, *then for any* $p \in \mathcal{G}(b_1,\ldots,b_k)$

   $(fl(b_1,\ldots,b_k,p)\backslash fl(b_1,\ldots,b_k))$ *and* $\mathcal{B}'(b_1,\ldots,b_k)\cup\mathcal{R}$ *intersect non-trivially.*

3. $\mathcal{R}(r_1,\ldots,r_k) \neq \phi \Rightarrow \mathbb{P}(Int(C)) \subset \mathcal{G}(r_1,\ldots,r_k)$. *Similarly* $\mathcal{B}(b_1,\ldots,b_k) \neq \phi \Rightarrow \mathbb{P}(Int(C)) \subset \mathcal{G}(b_1,\ldots,b_k)$.

4. *Given independent points* $r_1,\ldots,r_k \in \mathcal{R}$ ($b_1,\ldots,b_k \in \mathcal{B}$ *resp.),* $r \in \mathcal{R}$ ($b \in \mathcal{B}$ *resp.) the multi-set* $\mathbb{P}(Gcd(C)Int(C))$ *and the set* $\mathcal{P}$, *there exists efficient algorithms to compute the multi-set* $\mathbb{P}(M_2) \setminus \mathcal{G}(r_1,\ldots,r_k)$ ($\mathbb{P}(M_1) \setminus \mathcal{G}(b_1,\ldots,b_k)$ *resp.) and multi-sets of lines* $\mathcal{L}(r,\mathcal{G}(r_1,\ldots,r_k))$ ($\mathcal{L}(b,\mathcal{G}(b_1,\ldots,b_k))$ *resp.).*

Proof. Since the proof is long, for better exposition we move it to the appendix. See 28. □

Lemma 19. *The following hold for all independent* $r_1,\ldots,r_k \in \mathcal{R}$.

1. $\mathcal{R}(r_1,\ldots,r_k) \subset \mathcal{S}(r_1,\ldots,r_k) \subset \mathbb{P}(R)$.

2. $\mathcal{B}(r_1,\ldots,r_k) \subset \mathcal{S}(r_1,\ldots,r_k) \subset \mathbb{P}(B)$.

3. *Given* $r_1,\ldots,r_k$ ($b_1,\ldots,b_k$ *resp.), the set* $\mathcal{P}$, *multi-set* $\mathbb{P}(Gcd(C)Int(C))$ *and black-box* $\mathbb{B}_{Res}$, *there exist efficient algorithms to compute* $\mathcal{S}(r_1,\ldots,r_k)$ ($\mathcal{S}(b_1,\ldots,b_k)$ *resp.).*

Proof. Since the proof is long, for better exposition we move it to the appendix. See 29. □

Now we come to the application of robust high dimensional Sylvester gallai theorems from [3] and [6]. This will tell us about a really good choice for $r_1,\ldots,r_k$ ($b_1,\ldots,b_k$ resp.) in all the above lemmas.

Lemma 20. *One of the following always holds:*

1. $\exists\, r_1,\ldots,r_k \in \mathcal{R} : |\mathcal{R}(r_1,\ldots,r_k)| \geq v(\delta)|\mathcal{R}|$, *or*
2. $\exists\, b_1,\ldots,b_k \in \mathcal{R} : |\mathcal{B}(b_1,\ldots,b_k)| \geq v(\delta)|\mathcal{R}|$.

Proof. Since the proof is long, for better exposition we move it to the appendix. See 30. □

Lemma 20 then has the following obvious corollary:

Corollary 1. *One of the following always holds:*

1. *If Part I in Lemma 20 holds then* $|\mathcal{R}'(r_1,\ldots,r_k)| \leq (1 - v(\delta))|\mathcal{R}|$, *or*
2. *If Part II in Lemma 20 holds then* $|\mathcal{B}'(b_1,\ldots,b_k)| \leq (1 - v(\delta))|\mathcal{R}|$.

Proof.
1. Part I of Lemma 20 implies that $\mathcal{R}(r_1,\ldots,r_k) \geq v(\delta)|\mathcal{R}|$, therefore $\mathcal{R}'(r_1,\ldots,r_k) \leq |\mathcal{R}| - v(\delta)|\mathcal{R}| = (1 - v(\delta))|\mathcal{R}|$.
2. Similarly Part II of Lemma 20 implies that $\mathcal{B}(b_1,\ldots,b_k) \geq v(\delta)|\mathcal{R}|$, then $\mathcal{B}'(b_1,\ldots,b_k) \leq |\mathcal{B}| - v(\delta)|\mathcal{R}| \leq |\mathcal{R}| - v(\delta)|\mathcal{R}| = (1 - v(\delta))|\mathcal{R}|$.

□

All of these results above were proved to show the existence of certain good forms. This is the main ingredient in proof of lemma 17. We discuss the existence of good forms in the lemma below.

Lemma 21. *The following holds:*

1. *If Part I of Lemma 20 holds, then for any sub multi-set* $\mathcal{X} \subset \mathcal{G}(r_1,\ldots,r_k)$ *with* $dim(\mathcal{X}) \geq N_0 - 1$, r_1 *is a good form for* $(\mathcal{S}(r_1,\ldots,r_k),\mathcal{X})$.
2. *If Part II of Lemma 20 holds, then for any sub multi-set* $\mathcal{X} \subset \mathcal{G}(b_1,\ldots,b_k)$ *with* $dim(\mathcal{X}) \geq N_0 - 1$, b_1 *is a good form for* $(\mathcal{S}(b_1,\ldots,b_k),\mathcal{X})$.

Proof. We will replace $\mathcal{S}(r_1,\ldots,r_k)$ ($\mathcal{S}(b_1,\ldots,b_k)$ resp.) in the this lemma with $\mathcal{R}(r_1,\ldots,r_k)$ ($\mathcal{B}(b_1,\ldots,b_k)$ resp.). It can be easily seen that this implies the required statements since by lemma 19 above $\mathcal{R}(r_1,\ldots,r_k) \subset \mathcal{S}(r_1,\ldots,r_k)$ ($\mathcal{B}(b_1,\ldots,b_k) \subset \mathcal{S}(b_1,\ldots,b_k)$ resp.).

We will just show one of the two parts given in this lemma, the other follows identically. Consider the choice of $r_1,\ldots,r_k$ as in Lemma 20. We know that $\mathcal{R}(r_1,\ldots,r_k) \geq v(\delta)|\mathcal{R}|$. Note that lemma 18 implies that $\mathcal{G}(r_1,\ldots,r_k) \subset \mathcal{R}'(r_1,\ldots,r_k) \cup \mathcal{B}$. Let $\mathcal{X}$ be any sub multi-set of $\mathcal{G}(r_1,\ldots,r_k)$ such that $dim(\mathcal{X}) \geq N_0 - 1$. We want to show that r_1 is a good point for $(\mathcal{R}(r_1,\ldots,r_k),\mathcal{X})$.

Let's first fix $l_1,\ldots,l_k \in V$ such that $r_i = [l_i]$. Let $W = sp(l_1,\ldots,l_k)$ and $W^\perp$ be any subspace of V such that $W \oplus W^\perp = V$. Using definition 9, we consider central projections of points in $\mathcal{X}$ and $\mathcal{R}(r_1,\ldots,r_k)$ (note that no point in $\mathcal{X}$ or $\mathcal{R}(r_1,\ldots,r_k)$ belongs to $fl(r_1,\ldots,r_k)$) with respect to the flat $fl(r_1,\ldots,r_k)$ on to the projective space $\mathbb{P}(W^\perp)$. We will refer to this projection as a map π from here onwards. The following are easy to see (we leave the verification for the reader).

1. For distinct $r,r' \in \mathcal{R}(r_1,\ldots,r_k)$, $\pi(r) \neq \pi(r')$ (by definition of the set $\mathcal{R}(r_1,\ldots,r_k)$) and thus $|\pi(\mathcal{R}(r_1,\ldots,r_k))| \geq v(\delta)|\mathcal{R}|$.

2. Let $p \in \mathcal{X} \subset \mathcal{G}(r_1,\ldots,r_k)$. By lemma 18, for every $p \in \mathcal{G}(r_1,\ldots,r_k)$, there exists $p' \in \mathcal{R}'(r_1,\ldots,r_k) \cup \mathcal{B}$ such that

$$p' \in fl(r_1,\ldots,r_k,p) \setminus fl(r_1,\ldots,r_k).$$

   Thus $\pi(p') = \pi(p)$ giving us the bound $|\pi(\mathcal{G}(r_1,\ldots,r_k))| \leq |\mathcal{R}'(r_1,\ldots,r_k)| + |\mathcal{B}| \leq (1 - v(\delta))|\mathcal{R}| + |\mathcal{R}| = (2 - v(\delta))|\mathcal{R}|$.

Using lemma 31 we know that $\frac{2-v(\delta)}{v(\delta)} \leq \frac{1-\delta}{\delta}$. The two sets $\pi(\mathcal{X}), \pi(\mathcal{R}(r_1,\ldots,r_k))$ are disjoint sets in $\mathbb{P}(W^\perp)$. If not, consider $r_{k+1} \in \mathcal{R}(r_1,\ldots,r_k)$ such that $\pi(r_{k+1}) \in \pi(\mathcal{X})$. This implies that $r_{k+1} \in fl(r_1,\ldots,r_k,p) \setminus fl(r_1,\ldots,r_k)$ for some $p \in \mathcal{X}$. Also, since $\mathcal{X} \subset \mathcal{G}(r_1,\ldots,r_k)$, by lemma 18 we know that there is a $p' \in \mathcal{R}'(r_1,\ldots,r_k) \cup \mathbb{P}(B)$ such that $p' \in fl(r_1,\ldots,r_k,p) \setminus fl(r_1,\ldots,r_k)$.

Together these imply that $p' \in fl(r_1,\ldots,r_k,r_{k+1}) \setminus fl(r_1,\ldots,r_k)$. Clearly by definition of $\mathcal{R}(r_1,\ldots,r_k)$ the form $p' \notin \mathcal{R}'(r_1,\ldots,r_k)$ (since the flat $fl(r_1,\ldots,r_k,r_{k+1})$ is ordinary in $\mathcal{R} \cup \mathcal{B}$). It also cannot belong to $\mathbb{P}(B)$ due to the exact same reason. Therefore we have a contradiction and $\pi(\mathcal{X}), \pi(\mathcal{R}(r_1,\ldots,r_k))$ are disjoint.

Since we took a central projection with respect to $fl(r_1,\ldots,r_k)$, $dim(\mathcal{X}) \geq N_0 - 1 - k > \frac{\alpha}{\delta}$ inside $\mathbb{P}(W^\perp)$. By corollary 4 we get that there exists a line inside $\mathbb{P}(W^\perp)$ which has exactly one point from $\pi(\mathcal{X})$ (say $\pi(p)$ with $p \in \mathcal{P}$) and atleast one point from $\mathcal{R}(r_1,\ldots,r_k)$ (say $\pi(r_{k+1})$ with $r_{k+1} \in \mathcal{R}(r_1,\ldots,r_k)$). We will show the three conditions in definition 20 hold for the triple (r_1, r_{k+1}, p).

1. Since $r_{k+1}, p \notin fl(r_1,\ldots,r_k)$, clearly r_1, r_{k+1} are distinct and r_1, p are distinct. p, r_{k+1} are distinct as $\pi(p)$ and $\pi(r_{k+1})$ are distinct points in $\mathbb{P}(W^\perp)$.

2. r_1, r_{k+1} and p are independent. If not then the central projections $\pi(p)$ and $\pi(r_{k+1})$ would be equal which is not true.

3. Let $\Psi = fl(r_1, r_{k+1}, p)$. Assume there is another form $p' \in \mathcal{X}$ such that $p' \in \Psi$. Let $l_{k+1}, l_p, l_{p'} \in V$ such that $r_{k+1} = [l_{k+1}], p = [l_p]$ and $p' = [l_{p'}]$. $p' \in fl(r_1, r_{k+1}, p) \Rightarrow$ there are scalars a, b, c (not all zero) such that $l_{p'} = al_1 + bl_{k+1} + cl_p$. If $b = c = 0$ then $l_{p'} = al_1 \Rightarrow p' = r_1$, which cannot happen as $p' \notin fl(r_1,\ldots,r_k)$ (remember it is a point of $\mathcal{G}(r_1,\ldots,r_k)$). So at least one of b, c is non-zero. Write $l_{k+1} = w_{k+1} + w_{k+1}^\perp$ with $w_{k+1} \in W, w_{k+1}^\perp \in W^\perp \Rightarrow r_{k+1} = [w_{k+1}^\perp]$ and $l_p = w_p + w_p^\perp$ with $w_p \in W$ and $w_p^\perp \in W^\perp \Rightarrow p = [w_p^\perp]$.

   Therefore $l_{p'} = al_1 + bw_{k+1} + bw_{k+1}^\perp + cw_p + cw_p^\perp \Rightarrow l_{p'|_{W^\perp}} = bw_{k+1}^\perp + cw_p^\perp$. Atleast one of b, c is non-zero and thus $\pi(p') = [l_{p'|_{W^\perp}}] \in fl([w_{k+1}^\perp], [w_p^\perp]) = fl(\pi(r_{k+1}), \pi(p))$. Both $\pi(p'), \pi(p)$ are in $\mathcal{X}$ and the line $fl(\pi(r_{k+1}), \pi(p))$ contains only one point from $\pi(\mathcal{X}) \Rightarrow \pi(p) = \pi(p') \Rightarrow b = 0$ and thus $p' \in fl(r_1, p)$.

Therefore r_1, r_{k+1}, p satisfy all requirements in definition 20 and thus r_1 is a good form for $(\mathcal{R}(r_1,\ldots,r_k), \mathcal{X})$. □

Now we are ready for the proof of lemma 17 which was the main reconstruction result of this section.

Proof. We first give the algorithm promised in the lemma and then discuss correct-

ness and time complexity.

Fix $k = R(3,\mathbb{F}) + 2$.

for *each* $r_1,\ldots,r_k \in \mathcal{P}$ **do**

Compute multi-set $\mathcal{K} = \mathbb{P}(M_2) \setminus \mathcal{G}(r_1,\ldots,r_k)$ using algorithm in last part of lemma 18 and define multi-set $\mathcal{X} = \mathcal{G}(r_1,\ldots,r_k)$ (note that we do not know $\mathcal{X}$).

Compute the set $\mathcal{S}(r_1,\ldots,r_k)$ using algorithm in last part of lemma 19.

Compute multi-set of lines $\tilde{\mathcal{L}}(r_1) = \mathcal{L}(r_1, \mathcal{G}(r_1,\ldots,r_k) \cup \mathcal{B})$ and $\tilde{\mathcal{L}}(s) = \mathcal{L}(s, \mathcal{G}(r_1,\ldots,r_k) \cup \mathcal{B})$ for all $s \in \mathcal{S}(r_1,\ldots,r_k)$ using algorithm in last part of lemma 18.

while *true* **do**

Assume $dim(\mathcal{X}) < N_0 - 1$. Since $dim(\mathcal{R} \cup \mathcal{B}) \geq N_0$, there exists $r_1 \neq r_2 \in \mathcal{R}$ such that $sp(r_1,r_2) \cap sp(\mathcal{X}) = \phi$. Therefore r_1 is a good form for $(\{r_2\},\mathcal{X})$ by lemma 15. Using algorithm 1 compute $\mathcal{X}$ and update $\mathcal{K} = \mathcal{K} \cup \mathcal{X}$. Using algorithm 5 compute M_1, M_2. If output is not *"fail"* then break out of this while loop and output M_1, M_2. Else continue

Using r_1, set $\mathcal{S}(r_1,\ldots,r_k)$ and lines $\tilde{\mathcal{L}}(r_1), \tilde{\mathcal{L}}(s)(\forall s \in \mathcal{S}(r_1,\ldots,r_k))$ as input to algorithm 1, recover a multi-set of points $\mathcal{X}_{r_1}$.

If $\mathcal{X}_{r_1} = \phi$. Then break out of this while loop.

Else update $\mathcal{K} = \mathcal{K} \cup \mathcal{U}_{r_1}$, $\tilde{\mathcal{L}}(r_1) = \tilde{\mathcal{L}}(r_1) \setminus \mathcal{L}(r_1, \mathcal{X}_{r_1})$ and $\tilde{\mathcal{L}}(s) = \tilde{\mathcal{L}}(s) \setminus \mathcal{L}(s, \mathcal{X}_{r_1})$ (for all $s \in \mathcal{S}(r_1,\ldots,r_k)$).

Update $\mathcal{X} = \mathcal{X} \setminus \mathcal{X}_{r_1}$.

end

end

Output *"fail"*.

Algorithm 7: Both $\mathbb{P}(R), \mathbb{P}(B)$ have large dimension

Correctness - The algorithm returns polynomials M_1, M_2 using algorithm 5 and therefore if an answer is returned it is always correct.

Suppose we output *"fail"*. We will show that the chances of this happening are very small ($o(1)$ to be precise). We know by lemma 21 that there exists a choice of

$r_1, \ldots, r_k$ such that r_1 is a good form for $(\mathcal{S}(r_1, \ldots, r_k), \mathcal{X})$ for all $\mathcal{X} \subset \mathcal{G}(r_1, \ldots, r_k)$ and $dim(\mathcal{X}) \geq N_0 - 1$. Since we output *"fail"* at some point of time we must have used these $r_1, \ldots, r_k$ in the for loop. Clearly from the algorithm at any point of time $\mathcal{X} \subset \mathcal{G}(r_1, \ldots, r_k)$.

If $dim(\mathcal{X}) < N_0 - 1$, then clearly algorithm 1 would not recover $\mathcal{X}$ with probability $\leq o(1)$. So the probability we output fail is small.

If $dim(\mathcal{X}) \geq N_0 - 1$ then by lemma 21 we would have recovered $\mathcal{X}_{r_1} \subset \mathcal{X}$. This would have gone on until $dim(\mathcal{X}) < N_0 - 1$ and then with probability $\leq o(1)$ we would not recover $\mathcal{X}$. So we output fail with probability $o(1)$.

Therefore with probability $1 - o(1)$ we output two products M_1, M_2 and our answer is always right.

Time Complexity - All algorithms used run in $poly(n, d)$ time. The outer for loop runs $poly(n, d)$ times since k is a constant. So for the entire algorithm to run in $poly(n, d)$ time we need to show that the while loop runs $poly(n, d)$ times. If at any stage $dim(\mathcal{X}) < N_0 - 1$ then we halt in one step. Else if $\mathcal{X}_{r_1} = \phi$ we break out of the loop. Else $\mathcal{K} = \mathcal{K} \cup \mathcal{X}_{r_1}$ and so the size of $\mathcal{K}$ grows at least by 1. $\mathcal{K} \subset \mathbb{P}(M_2)$ and so it can grow at most $poly(n, d)$ times, thus the while loop runs $poly(n, d)$ times.

□

BIBLIOGRAPHY

[1] Manindra Agrawal. “Proving lower bounds via pseudo-random generators”. In: *FSTTCS 2005: Foundations of Software Technology and Theoretical Computer Science, 25th International Conference, Hyderabad, India, December 15-18, 2005, Proceedings, volume 3821 of Lecture*. Springer, 2005, pp. 92–105.

[2] V. Arvind, Partha Mukhopadhyay, and Srikanth Srinivasan. “New Results on Noncommutative and Commutative Polynomial Identity Testing”. In: *2012 IEEE 27th Conference on Computational Complexity* (2008), pp. 268–279. ISSN: 1093-0159. DOI: http://doi.ieeecomputersociety.org/10.1109/CCC.2008.22.

[3] B. Barak et al. “Rank bounds for design matrices with applications to combinatorial geometry and locally correctable codes”. In: *Proceedings of the 43rd annual ACM symposium on Theory of computing*. STOC ’11. San Jose, California, USA: ACM, 2011, pp. 519–528. ISBN: 978-1-4503-0691-1. URL: ./BDWY11.pdf.

[4] Amos Beimel et al. “Learning Functions Represented As Multiplicity Automata”. In: *J. ACM* 47.3 (May 2000), pp. 506–530. ISSN: 0004-5411. DOI: 10.1145/337244.337257. URL: http://doi.acm.org/10.1145/337244.337257.

[5] B. Buchberger. “A Theoretical Basis for the Reduction of Polynomials to Canonical Forms”. In: *SIGSAM Bull.* 10.3 (Aug. 1976), pp. 19–29. ISSN: 0163-5824. DOI: 10.1145/1088216.1088219. URL: http://doi.acm.org/10.1145/1088216.1088219.

[6] Z. Dvir, S. Saraf, and A. Wigderson. “Improved rank bounds for design matrices and a new proof of Kelly’s theorem”. Forum of mathematics - Sigma (to appear). 2012. URL: DSW12.pdf.

[7] Zeev Dvir, Rafael Mendes de Oliveira, and Amir Shpilka. “Automata, Languages, and Programming: 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I”. In: ed. by Javier Esparza et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014. Chap. Testing Equivalence of Polynomials under Shifts, pp. 417–428. ISBN: 978-3-662-43948-7. DOI: 10.1007/978-3-662-43948-7_35. URL: http://dx.doi.org/10.1007/978-3-662-43948-7_35.

[8] Zeev Dvir and Amir Shpilka. “Locally decodable codes with 2 queries and polynomial identity testing for depth 3 circuits”. In: *SIAM J. COMPUT* 36.5 (2007), pp. 1404–1434.

[9] Izrail Moiseevitch Gelfand, Mikhail M. Kapranov, and Andrei V. Zelevinsky. *Discriminants, resultants, and multidimensional determinants*. Mathematics : theory & applications. Autre tirage de l'édition Birkhäuser chez Springer Science+ Business Media. Boston, Basel, Berlin: Birkhäuser, 1994. ISBN: 0-8176-3660-9. URL: http://opac.inria.fr/record=b1103027.

[10] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. "How to Construct Random Functions". In: *J. ACM* 33.4 (Aug. 1986), pp. 792–807. ISSN: 0004-5411. DOI: 10.1145/6490.6503. URL: http://doi.acm.org/10.1145/6490.6503.

[11] Ankit Gupta, Neeraj Kayal, and Satya Lokam. "Reconstruction of Depth-4 Multilinear Circuits with Top Fan-in 2". In: *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing*. STOC '12. New York, New York, USA: ACM, 2012, pp. 625–642. ISBN: 978-1-4503-1245-5. DOI: 10.1145/2213977.2214035. URL: http://doi.acm.org/10.1145/2213977.2214035.

[12] Ankit Gupta, Neeraj Kayal, and Satyanarayana V. Lokam. "Efficient Reconstruction of Random Multilinear Formulas". In: *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*. 2011, pp. 778–787. DOI: 10.1109/FOCS.2011.70. URL: http://dx.doi.org/10.1109/FOCS.2011.70.

[13] Ankit Gupta, Neeraj Kayal, and Youming Qiao. "Random arithmetic formulas can be reconstructed efficiently". English. In: *computational complexity* 23.2 (2014), pp. 207–303. ISSN: 1016-3328. DOI: 10.1007/s00037-014-0085-0. URL: http://dx.doi.org/10.1007/s00037-014-0085-0.

[14] J. Heintz and C. P. Schnorr. "Testing Polynomials Which Are Easy to Compute (Extended Abstract)". In: *Proceedings of the Twelfth Annual ACM Symposium on Theory of Computing*. STOC '80. Los Angeles, California, USA: ACM, 1980, pp. 262–272. ISBN: 0-89791-017-6. DOI: 10.1145/800141.804674. URL: http://doi.acm.org/10.1145/800141.804674.

[15] Begnaud Francis Hildebrand. *Introduction to Numerical Analysis: 2Nd Edition*. New York, NY, USA: Dover Publications, Inc., 1987. ISBN: 0-486-65363-3.

[16] Erich Kaltofen. "Effective Noether Irreducibility Forms and Applications". In: *Proceedings of the Twenty-third Annual ACM Symposium on Theory of Computing*. STOC '91. New Orleans, Louisiana, USA: ACM, 1991, pp. 54–63. ISBN: 0-89791-397-3. DOI: 10.1145/103418.103431. URL: http://doi.acm.org/10.1145/103418.103431.

[17] Erich Kaltofen and Barry M. Trager. "Computing with Polynomials Given Byblack Boxes for Their Evaluations: Greatest Common Divisors, Factorization, Separation of Numerators and Denominators". In: *J. Symb. Comput.* 9.3 (Mar. 1990), pp. 301–320. ISSN: 0747-7171. DOI: 10.1016/S0747-

7171(08)80015-6. URL: http://dx.doi.org/10.1016/S0747-7171(08)80015-6.

[18] Zohar S. Karnin and Amir Shpilka. "Reconstruction of Generalized Depth-3 Arithmetic Circuits with Bounded Top Fan-in". In: *Proceedings of the 24rd Annual CCC*. 2009, pp. 274–285.

[19] Michael Kearns and Leslie Valiant. "Cryptographic Limitations on Learning Boolean Formulae and Finite Automata". In: *J. ACM* 41.1 (Jan. 1994), pp. 67–95. ISSN: 0004-5411. DOI: 10.1145/174644.174647. URL: http://doi.acm.org/10.1145/174644.174647.

[20] Michael Kharitonov. "Cryptographic Lower Bounds for Learnability of Boolean Functions on the Uniform Distribution". In: *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*. COLT '92. Pittsburgh, Pennsylvania, USA: ACM, 1992, pp. 29–36. ISBN: 0-89791-497-X. DOI: 10.1145/130385.130388. URL: http://doi.acm.org/10.1145/130385.130388.

[21] Adam R. Klivans and Daniel Spielman. "Randomness Efficient Identity Testing of Multivariate Polynomials". In: *Proceedings of the Thirty-third Annual ACM Symposium on Theory of Computing*. STOC '01. Hersonissos, Greece: ACM, 2001, pp. 216–223. ISBN: 1-58113-349-9. DOI: 10.1145/380752.380801. URL: http://doi.acm.org/10.1145/380752.380801.

[22] Adam Klivans and Amir Shpilka. "Learning restricted models of arithmetic circuits." In: *Theory of computing* 2.10 (2006), pp. 185–206.

[23] S. Kopparty, S. Saraf, and A. Shpilka. "Equivalence of Polynomial Identity Testing and Deterministic Multivariate Polynomial Factorization". In: *Computational Complexity (CCC), 2014 IEEE 29th Conference on*. June 2014, pp. 169–180. DOI: 10.1109/CCC.2014.25.

[24] Gary L. Mullen and Daniel Panario. *Handbook of Finite Fields*. 1st. Chapman & Hall/CRC, 2013. ISBN: 143987378X, 9781439873786.

[25] Nitin Saxena and C. Seshadhri. "From Sylvester-gallai Configurations to Rank Bounds: Improved Blackbox Identity Test for Depth-3 Circuits". In: *J. ACM* 60.5 (Oct. 2013), 33:1–33:33. ISSN: 0004-5411. DOI: 10.1145/2528403. URL: http://doi.acm.org/10.1145/2528403.

[26] Robert E. Schapire and Linda M. Sellie. "Learning Sparse Multivariate Polynomials over a Field with Queries and Counterexamples". In: *In Proceedings of the Sixth Annual ACM Workshop on Computational Learning Theory*. 1996, pp. 17–26.

[27] Amir Shpilka. "Interpolation of depth-3 arithmetic circuits with two multiplication gates". In: *In STOC '07: Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*. ACM Press, 2007, pp. 284–293.

[28] Amir Shpilka and Ilya Volkovich. "Improved polynomial identity testing for read-once formulas". In: (2009), pp. 700–713.

[29] Amir Shpilka and Amir Yehudayoff. “Arithmetic Circuits: A Survey of Recent Results and Open Questions”. In: *Foundations and Trends in Theoretical Computer Science* 5.3–4 (2010), pp. 207–388. ISSN: 1551-305X. DOI: 10.1561/0400000039. URL: http://dx.doi.org/10.1561/0400000039.

[30] Madhu Sudan. “Algebra and Computation”. In: 1998. URL: http://people.csail.mit.edu/madhu/FT98/.

Appendix A

RANDOM TRANSFORMATION AND RESTRICTIONS

For technical purposes we wish to randomly transform the variables in our polynomial. In order to do this we use an $n \times n$ matrix $\Omega = (\Omega_{i,j})$. First we define a map $\tilde{\Omega} : x_i \mapsto (\Omega\bar{x})_i$, where $\bar{x}$ is viewed as the $n \times 1$ column vector $(x_1, \ldots, x_n)^T$. This can be extended to a linear transformation on V, the vector space of linear forms in variables $\bar{x} = (x_1, \ldots, x_n)$. Once we have such a linear transformation $\tilde{\Omega}$, we extend it further to polynomials in the most natural way i.e. we define $\tilde{\Omega}(f)(x_1, \ldots, x_n) = f(\tilde{\Omega}(x_1), \ldots, \tilde{\Omega}(x_n))$. We leave it to the readers to check that this map is well defined and is an extension of the linear transformation $\tilde{\Omega}$. Note that by abuse of notation we used $\tilde{\Omega}$ for both maps. It can be further checked that $\tilde{\Omega}$ an algebra homomorphism on $\mathbb{F}[\bar{x}]$.

In our application we choose the matrix Ω such that every entry $\Omega_{i,j}$ is picked uniformly randomly and independently from a fixed finite set $S \subset \mathbb{F}$. As a first step of our algorithm we will apply $\tilde{\Omega}$ to our input polynomial C. This will ensure that the new variables we have are *"random"* in some sense. A problem we face is that our input is given as a black-box and so what does *"applying"* $\tilde{\Omega}$ mean? This is easily resolved by augmenting our input black-box i.e. we apply $\tilde{\Omega}$ to the black-box.

To evaluate the polynomial $\tilde{\Omega}(C)$ (for some polynomial C) at the point $\bar{a} = (a_1, \ldots, a_n)$, we query the original black-box for C at the form $\Omega^{-1}(\bar{a})$. Symbolically we define a new black-box $\mathbb{B}$ such that $\mathbb{B}(\bar{a}) = \mathbb{B}_{in}(\Omega^{-1}(\bar{a}))$, where $\mathbb{B}_{in}$ is the input black-box for polynomial C. Note that this needs Ω to be invertible. We check this just after choosing Ω. It will be invertible with a high probability as shown in lemma 22 below. If it is not then we output *"fail"*. We also point out that when Ω is invertible, $\tilde{\Omega}$ becomes an algebra isomorphism.

The randomness in our algorithms is precisely due to this choice we make at the beginning. A lot of results we prove in this work will be true with high probability over the choice of Ω.

Lemma 22. *Let Ω be the matrix chosen above. We can show*

$$Pr[\Omega \text{ is not-invertible }] \leq \frac{n}{|S|}$$

Proof. This is equivalent to saying that the determinant $det(\Omega) = 0$. Determinant of Ω is a polynomial of degree n in n^2 variables $\{\Omega_{i,j} : (i.j) \in [n] \times [n]\}$ and is not identically zero since coefficient of the monomial $\Omega_{1,1} \ldots \Omega_{n,n}$ is equal to 1. Therefore by Schwartz-Zippel lemma

$$Pr[\Omega : det(\Omega) = 0] \leq \frac{n}{|S|}$$

□

and hence with high probability Ω is invertible.

Let $\mathbb{P}(V)$ be the projective space corresponding to the vector space V of linear forms in $\bar{x}$. For $j \in \{r+1, \ldots, n\}$, consider the subspaces:

$$W_i = \{(w_1, \ldots, w_{r-1}, 0, \ldots, 0, w_j, 0, \ldots, 0) : w_j \in \mathbb{F} \text{ for } j \in \{1, \ldots, r-1, j\}\}$$

Below we show that with high probability a projective form cannot lie in the projective space of the random subspace above. Therefore when we restrict to this subspace the form remains non-zero.

Lemma 23. *Let $\mathcal{T} \subset \mathbb{P}(V)$ be a multi-set of forms. The following is true:*

$$Pr[\Omega : \exists \, form \, t \in \mathcal{T} : \Omega(t)_{|_{W_i}} = 0] \leq \frac{|\mathcal{T}|}{|S|}$$

Proof. Fix $t = [t_1 x_1 + \ldots + t_n x_n]$. We know that coefficient of x_1 in $\Omega(t)$ is equal to $\Omega(t)_1 = \sum_k t_k \Omega_{k,1}$. This is a polynomial in the variables $\Omega_{1,1}, \ldots, \Omega_{n,1}$. If this polynomial were identically zero then all $t_i = 0, i \in [n] \Rightarrow t \notin \mathbb{P}(\mathbb{V})$. Thus this polynomial is not identically zero. By the Schwartz-Zippel lemma

$$Pr[\Omega : \Omega(t)_1 = 0] \leq \frac{1}{|S|}.$$

Clearly the variable x_1 and coefficient $\Omega(t)_1$ appear on restriction of t to W_i. Thus $\Omega(t)_1 \neq 0 \Rightarrow \Omega(t)_{|_{W_i}}$ is non-zero . A union bound over forms in $\mathcal{T}$ give us:

$$Pr[\Omega : \exists \text{ form } t \in \mathcal{T} : \Omega(t)_{|_{W_i}} = 0 \leq \frac{|\mathcal{T}|}{|S|}.$$

□

Next we show that an independent subset of size r, inside a multi-set $\mathcal{T} \subset \mathbb{P}(V)$, continues to be independent when restricted to certain r variables in the new *"random"* set of variables defined by $\tilde{\Omega}$.

Lemma 24. *Let $\mathcal{T} \subset \mathbb{P}(V)$ be a multi-set of forms. The following is true:*

$$Pr[\exists j \in \{r,\ldots,n\} \text{ and independent forms } t_1,\ldots,t_r \in \mathcal{T} : \Omega(t_1)_{|W_j},\ldots,\Omega(t_r)_{|W_j} \text{ are dependent}] \leq \frac{(n-r+1)r|T|^r}{|S|}$$

Proof. Fix independent forms $t_1,\ldots,t_r \in \mathcal{T}$. Let $t_i = [t_{i,1}x_1 + \ldots t_{i,n}x_n]$ with $t_{i,k} \in \mathbb{F}$. Clearly for $i \in [r]$ the restrictions of $\Omega(t_i)$ are:

$$\Omega(t_i)_{|W_j} = [\sum_k t_{i,k}\Omega_{k,1}x_1 + \ldots \sum_k t_{i,k}\Omega_{k,r-1}x_{r-1} + \sum_k t_{i,j}\Omega_{k,j}x_j]$$

The set $\{\Omega(t_1)_{|W_i},\ldots,\Omega(t_r)_{|W_i}\}$ are dependent if and only if the determinant

$$g(\Omega) = \begin{vmatrix} \sum_k t_{1,k}\Omega_{k,1} & \ldots & \sum_k t_{1,k}\Omega_{k,r-1} & \sum_k t_{1,k}\Omega_{k,j} \\ \sum_k t_{2,k}\Omega_{k,1} & \ldots & \sum_k t_{2,k}\Omega_{k,r-1} & \sum_k t_{2,k}\Omega_{k,j} \\ \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots \\ \sum_k t_{r,k}\Omega_{k,1} & \ldots & \sum_k t_{r,k}\Omega_{k,r-1} & \sum_k t_{r,k}\Omega_{k,j} \end{vmatrix}$$

is zero. If the above determinant is identically zero (i.e. as a polynomial) then by plugging in suitable values of Ω's we get that the determinant

$$\begin{vmatrix} t_{1,1} & \ldots & \ldots & t_{1,r-1} & t_{1,j} \\ t_{2,1} & \ldots & \ldots & t_{2,r-1} & t_{2,j} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ t_{r,1} & \ldots & \ldots & t_{r,r-1} & t_{r,j} \end{vmatrix} = 0$$

is zero. This is a minor of the matrix formed by coefficients of $t_1,\ldots,t_r$ and therefore cannot be zero. So the determinant (polynomial) $g(\Omega)$ is not identically zero. Hence by the Schwartz Zippel Lemma :

$$Pr[\Omega : g(\Omega) = 0] \leq \frac{r}{|S|}$$

Now doing a union bound over all $\{t_1,\ldots,t_r\} \in \mathcal{T}$ and $j \in \{r,\ldots,n\}$ we get that

$$Pr[\exists j \in \{r,\ldots,n\} \text{ and forms } t_1,\ldots,t_r \in \mathcal{T} : \Omega(t_1)_{|W_i},\ldots,\Omega(t_r)_{|W_i} \text{ are dependent}] \leq \frac{(n-r+1)r|\mathcal{T}|^r}{|S|}$$

□

Another lemma of the same sort which will be useful is stated below.

Lemma 25. *Let* $W_{r-1} = \{(a_1, \ldots, a_{r-1}, 0, \ldots, 0)\} \subset \mathbb{F}^n, r \geq 4$. *Consider multi-set* $\mathcal{T} \subset \mathbb{P}(V)$. *Then:*

$$Pr[\exists \text{ independent forms } t_1, \ldots, t_s \in \mathcal{T}, s \leq 3 : \Omega(t_1)_{|W_{r-1}}, \ldots, \Omega(t_s)_{|W_{r-1}} \text{ are dependent}] \leq \frac{poly(n, r, |\mathcal{T}|^r)}{|S|}$$

Proof. Exactly like the previous proof with some minor changes. We do not repeat it for cleaner exposition. □

Now we summarize the above lemmas in the way we wish to apply them.

Corollary 2. *Let* $\mathcal{T} \subset \mathbb{P}(V)$ *and* $r \geq 4$. *The following statements are true with probability* $\geq 1 - \frac{poly(n,r,|\mathcal{T}|^r)}{|S|}$:

1. *Let* $t \in \mathcal{T}$ *be , then with high probability* $\Omega(t)_{|W_i}$ *is a well defined projective form i.e. is non-zero.*
2. *For every independent* $t_1, \ldots, t_s \in \mathcal{T}$ *with* $s \leq r$, $\Omega(t_1)_{|W_j}, \ldots, \Omega(t_s)_{|W_j}$ *are independent.*
3. *For every independent* $t_1, \ldots, t_s \in \mathcal{T}$ *with* $s \leq 3$, $\Omega(t_1)_{|W_{r-1}}, \ldots, \Omega(t_s)_{|W_{r-1}}$ *are independent.*

Proof. For Part 1 we use Lemma 23 given in this appendix. For part 2 extend the set $\{t_1, \ldots, t_s\}$ of forms to an independent set of size r inside $\mathcal{T}$ and then use Lemma 24. Part 3 follows by Lemma 25 above. We want all three to be true and so we add their failure probabilities i.e. take a union bound over the bad cases. □

In our application $|S| >> \Omega(poly(n, r, |\mathcal{T}|^r))$ and so we will assume that the above corollary holds deterministically for the set $\mathcal{T}$.

Appendix B

BRILL'S EQUATIONS - CHARACTERIZING POLYNOMIALS WHICH ARE PRODUCT OF LINEAR FORMS

Let $f(\bar{x}) \in \mathbb{C}[\bar{x}]$ be a polynomial of degree d in variables $\bar{x} = (x_1, \ldots, x_n)$ such that $f(\bar{x})$ factors into a product of linear forms. These are called totally decomposable polynomials. In this chapter we will give a characterization for the coefficients of f.

This has been well studied in mathematics literature. The basic idea is to construct a family of polynomials in many variables which vanish exactly at the coefficients of totally decomposable polynomials. A clean mathematical construction is given by Brill's Equations given in Chapter 4, [9].

However we still need to calculate the time complexity. But before that we define some operations on polynomials and calculate the time taken by the operation along with the size of the output. Note that all polynomials are over the field of complex numbers $\mathbb{C}$ and all computations are also done for the complex polynomial rings.

Let $\bar{x} = (x_1, \ldots, x_r)$ and $\bar{y} = (y_1, \ldots, y_r)$ be variables. For any homogeneous polynomial $f(\bar{x})$ of degree d, define

$$f_{\bar{x}^k}(\bar{x}, \bar{y}) = \frac{(d-k)!}{d!}(\sum_i x_i \frac{\partial}{\partial y_i})^k f(\bar{y})$$

Expanding $(\sum_i x_i \frac{\partial}{\partial y_i})^k$ as a polynomial of differentials takes $O((r+k)^r)$ time and has the same order of terms in it. $f(\bar{y})$ has $O((r+k)^r)$ terms. Taking partial derivatives of each term takes constant time and therefore overall computing $(\sum_i x_i \frac{\partial}{\partial y_i})^k f(\bar{y})$ takes $O((r+k)^{2r})$ time. Also the expression obtained will have at most $O((r+k)^{2r})$ terms. Computing the external factor takes $poly(d)$ time and so for an arbitrary $f(\bar{x})$ computing all $f_{\bar{x}^k}(\bar{x}, \bar{y})$ for $0 \leq k \leq d$ takes $poly((r+d)^r)$ time and has $poly((r+d)^r)$ terms in it. From Section E., Chapter 4 in [9] we also know that $f_{\bar{x}^k}(\bar{x}, \bar{y})$ is a bi-homogeneous form of degree k in $\bar{x}$ and degree $d-k$ in $\bar{y}$. It is called the k^{th} polar of f.

Next we define an $\odot$ operation between homogeneous forms. Let $f(\bar{x})$ and $g(\bar{x})$ be homogeneous polynomials of degrees d, define

$$(f \odot g)(\bar{x}, \bar{y}) = \frac{1}{d+1} \sum_{k=0}^{d} (-1)^k \binom{d}{k} f_{\bar{y}^k}(\bar{y}, \bar{x}) g_{\bar{x}^k}(\bar{x}, \bar{y})$$

From the discussion above we know that computing $f_{\bar{y}^k}(\bar{y}, \bar{x}) g_{\bar{x}^k}(\bar{x}, \bar{y})$ takes $poly((r+d)^r)$ time and it is obvious that this product has $poly((r+d)^r)$ terms. Rest of the operations take $poly(d)$ time and therefore computing $(f \odot g)(\bar{x}, \bar{y})$ takes $poly((r+d)^r)$ time and has $poly((r+d)^r)$ terms. From the discussion before we may also easily conclude that the degrees of $\bar{x}, \bar{y}$ in $(f \odot g)(\bar{x}, \bar{y})$ are $poly(d)$. The form $(f \odot g)$ is called the vertical(Young) product of f and g. See Section G., Chapter 4 in [9].

Next for $k \in \{0, \ldots, d\}$ and $\bar{z} = (z_1, \ldots, z_r)$ consider homogeneous forms:

$$e_k = \binom{d}{k} f_{\bar{x}^k}(\bar{x}, \bar{z}) f(\bar{z})^{k-1}$$

Following arguments from above, it's straightforward to see that computing e_k takes $poly((r+d)^r)$ time and has $poly((r+d)^r)$ terms. Each e_k is a homogeneous form in $\bar{x}, \bar{z}$ and f. It has degree k in $\bar{x}$, degree $k(d-1)$ in z, and k in coefficients of f. See Section H. of Chapter 4 in [9]. Let's define the following function of $\bar{x}$ with parameters f, z

$$P_{f,z}(\bar{x}) = (-1)^d d \sum_{i_1 + 2i_2 + \ldots + r i_r = d} (-1)^{(i_1 + \ldots + i_r)} \frac{(i_1 + \ldots + i_r - 1)!}{i_1! \ldots i_r!} e_1^{i_1} \ldots e_r^{i_r}$$

Note that $\{(i_1, \ldots, i_r) : i_1 + 2i_2 + \ldots + r i_r = d\} \subseteq \{(i_1, \ldots, i_r) : i_1 + i_2 + \ldots + i_r \leq d\}$ and therefore the number of additions in the above summand is $O(poly(r+d)^r)$. For every fixed $(i_1, \ldots, i_r)$ computing the coefficient $\frac{(i_1 + \ldots + i_r - 1)!}{i_1! \ldots i_r!}$ takes $O(poly((r+d)^r))$ time using multinomial coefficients. Each e_k takes $poly((r+d)^r)$ time to compute. There are r of them in each summand and so overall we take $O(poly((r+d)^r))$ time. A similar argument shows that number of terms in this polynomial is $O(poly((r+d)^r))$. Some more analysis shows that $P_{f,z}(\bar{x})$ is a form of degree d in $\bar{x}$ whose coefficients are homogeneous polynomials of degree d in f and degree $d(d-1)$ in $\bar{z}$. Let

$$B_f(\bar{x}, \bar{y}, \bar{z}) = (f \odot P_{f,z})(\bar{x}, \bar{y})$$

By the arguments given above calculating this form also takes time $poly((r+d)^r)$ and it has $poly((r+d)^r)$ terms. This is a homogeneous form in $(\bar{x}, \bar{y}, \bar{z})$ of multi-degree $(d, d, d(d-1))$ and it's coefficients are forms of degree $(d+1)$ in the coefficients of f. See Section H., Chapter 4 in [9]. So in time $poly((r+d)^r)$ we can compute $B_f(\bar{x}, \bar{y}, \bar{z})$ explicitly.

Now we arrive at the main theorem

Theorem 9 (**B**rill's Equation, See 4.H, [9])**.** *A form $f(\bar{x})$ is a product of linear forms if and only if the polynomial $B_f(\bar{x}, \bar{y}, \bar{z})$ is identically* 0*.*

We argued above that computing $B_f(\bar{x}, \bar{y}, \bar{z})$ takes $O(poly((r+d)^r))$ time. It's degrees in $\bar{x}, \bar{y}, \bar{z}$ are all $poly(d)$ and so the number of coefficients when written as a polynomial over the $3r$ variables

$(x_1, \ldots, x_r, y_1, \ldots, y_r, z_1, \ldots, z_r)$ is $poly((r+d)^r)$. We mentioned that each coefficient is a polynomial of degree $(d+1)$ in the coefficients of f. Therefore we have the following corollary.

Corollary 3. *Let*

$$I := \{(\alpha_1, \ldots, \alpha_n) : \forall i : \alpha_i \geq 0, \sum_{i \in [r]} \alpha_i = d\}$$

be the set capturing the indices of all possible monomials of degree exactly d in r variables $(x_1, \ldots, x_r)$. Let $f_{\boldsymbol{a}}(y_1, \ldots, y_r) = \sum_{\alpha \in I} a_\alpha \boldsymbol{y}^\alpha$ denote an arbitrary homogeneous polynomial. The coefficient vector then becomes $\boldsymbol{a} = (a_\alpha)_{\alpha \in I}$. Then there exists an explicit set of polynomials $F_1(\boldsymbol{a}), \ldots, F_m(\boldsymbol{a})$ on $poly((r+d)^r)$ variables $(\boldsymbol{a} = (a_\alpha)_{\alpha \in I})$, with $m = poly((r+d)^r)$, $deg(F_i) \leq poly(d)$ such that for any particular value of $\boldsymbol{a}$, the corresponding polynomial $f_{\boldsymbol{a}}(\boldsymbol{y}) \in \Pi\Sigma^d_{\mathbb{F}}[\bar{y}]$ if and only if $F_1(\boldsymbol{a}) = \ldots = F_m(\boldsymbol{a}) = 0$. Also this set $\{F_i, i \in [m]\}$ can be computed in time $poly((r+d)^r)$ time.

Proof. Clear from the theorem and discussion above.

Note that in our application $r = O(1)$ and so $poly((d+r)^r) = poly(d)$.

Appendix C

BLACK-BOX FACTORING OF POLYNOMIALS

In this section we will develop algorithms to factorize black-boxes of multivariate polynomials and extract all the linear factors explicitly. We also compute the product of all non-linear irreducible factors.

Consider variables $\bar{x} = (x_1, \ldots, x_n)$ and field $\mathbb{F}$. Let V be the vector space of linear forms in $\mathbb{F}[\bar{x}]$ and $\mathbb{P}(V)$ be the corresponding projective space. Let $f(\bar{x}) \in \mathbb{F}[\bar{x}]$ be a degree d polynomial. Factorize $f(\bar{x})$ to get the following form:

$$f(\bar{x}) = l_1(\bar{x}) \ldots l_t(\bar{x}) Res(f)(\bar{x}). \tag{C.1}$$

- Every $l_i(\bar{x})$ is an linear form.
- The polynomial $Res(f)(\bar{x})$ called *"residual factors"* has no linear factors.
- Define the multi-set $\mathcal{F} = \{[l_1], \ldots, [l_t]\} \subset \mathbb{P}(V)$.

Goals for this chapter. Given black-box access to $f(\bar{x})$, we will discuss efficient (randomized) algorithms to recover:

- The set $\mathcal{F}$ explicitly, i.e. all $[l_i]$ explicitly.
- Black-box access to polynomial $Res(f)(\bar{x})$.

Black-box access to factors. We first compute black-boxes for all irreducible factors of $f(\bar{x})$. An algorithm to do the same was given by Kaltofen and Trager in [17]. Even though their algorithm was meant for characteristic zero fields, with minor changes it works over all finite fields. Details can be found in lecture 9 from MIT's algebra and computation course lecture notes [30]. A short and sweet description fulfilling all important details is given in remark 11.5.66 in [24].

The essence of their algorithm is an *"effective"* Hilbert's irreducibility theorem (see theorem 11), which says that an irreducible polynomial continues to be irreducible when restricted to certain random 3− dimensional subspaces. This has been discussed in great detail in section 4 in [16]. In particular the result can be found in

corollary 2 of section 4 in [16]. The theorem is sometimes also called *"quantitative Bertini theorem"*. See Theorem 11.5.33 in [24] for details and further references.

Here is the black-box factorization theorem:

Theorem 10 (Black-box factorization, Section 2 in [17], lecture 9 in [30], remark 11.5.66 in [24])**.** *Let $f(\bar{x}) \in \mathbb{F}[\bar{x}]$ be an $n-$variate, degree d polynomial. Assume $f(\bar{x}) = h_1(\bar{x})^{e_1} \dots h_k(\bar{x})^{e_k}$, where $\{h_i(\bar{x})\}_{i=1}^{k}$ are distinct irreducible polynomials in $\mathbb{F}[\bar{x}]$ and let $\boldsymbol{e} = (e_1, \dots, e_n)$ denote the tuple of exponents. Then there is a randomized algorithm, that given black-box access to $f(\bar{x})$ (and parameter n), runs in time $poly(d,n)$ and outputs a tuple of integers $\boldsymbol{e} = (e'_1, \dots, e'_k)$ along with a collection of black-boxes computing $h'_1, \dots h'_k$ such that*

$$Pr[\boldsymbol{e'} = \boldsymbol{e}, \exists \gamma_i \in \mathbb{F} \setminus \{0\}, i \in [k] : \forall \bar{a} \in \mathbb{F}^n, h'_i(\bar{a}) = \gamma_i h_i(\bar{a})] \geq 1 - \frac{poly(d)}{|\mathbb{F}|}$$

Now we are ready to give [1]

BIBLIOGRAPHY

[1] Gaurav Sinha. “Reconstruction of Real Depth-3 Circuits with Top Fan-In 2”. In: *31st Conference on Computational Complexity (CCC 2016)*. Ed. by Ran Raz. Vol. 50. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016, 31:1–31:53. ISBN: 978-3-95977-008-8. DOI: http://dx.doi.org/10.4230/LIPIcs.CCC.2016.31. URL: http://drops.dagstuhl.de/opus/volltexte/2016/5854.

PUBLISHED CONTENT AND CONTRIBUTIONS

[Include a bibliography of published articles or other material that are included as part of the thesis. Describe your role with the each article and its contents. Citations must include DOIs or publisher URLs if available electronically.

If you are incorporating any third-party material in the thesis, including works that you have authored/co-authored but for which you have transferred copyright, you must indicate that permission has been secured to use the material. For example: "Fig. 2 reprinted with permission from the copyright holder, holder name"

Add the option `iknowwhattodo` to this environment to dismiss this message.]

[1] Gaurav Sinha. "Reconstruction of Real Depth-3 Circuits with Top Fan-In 2". In: *31st Conference on Computational Complexity (CCC 2016)*. Ed. by Ran Raz. Vol. 50. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016, 31:1–31:53. ISBN: 978-3-95977-008-8. DOI: http://dx.doi.org/10.4230/LIPIcs.CCC.2016.31. URL: http://drops.dagstuhl.de/opus/volltexte/2016/5854.

the algorithm. We state it in an informal way and leave the details for the reader to verify. The algorithm is a very simple application of Theorem 10.

Algorithm

1. Using Theorem 10 obtain the tuple of numbers $(e'_1, \ldots, e'_k)$ and the tuple of black-boxes $(h'_1, \ldots, h'_k)$.
2. Iterate over all irreducible factors (black-boxes) h'_i and using $O(n)$ queries interpolate a linear form $l_i = \alpha_0 + \alpha_1 x_1 + \ldots + \alpha_n x_n$.
3. Then using the randomized black-box polynomial identity testing algorithm (a.k.a schwartz zippel lemma) we check if $h_i - l_i$ is an identically zero polynomial. This just involves checking $h'_i(\bar{a}) - l_i(\bar{a})$ for $\bar{a}$ chosen uniformly randomly from a large subset $S \subset \mathbb{F}$.
4. If yes, we add e_i copies of the form $[l_i]$ to our set $\mathcal{F}$.
5. If no, we add e_i copies of the black-box h_i to another multi-set $\mathcal{F}'$.

6. At the end of the iteration, we define a black-box $Res(f) = \prod_{h \in \mathcal{F}'} h$. This black-box can be simulated by querying each of the black-boxes in $\mathcal{F}'$ and multiplying the outputs.

7. Finally we return the set $\mathcal{F}$ and the black-box $Res(f)$.

It's trivial to see that the time complexity of this algorithm is $poly(n, d)$.

As mentioned before the algorithm in [17] uses the effective Hilbert irreducibility theorem. This theorem is also useful for us in this thesis and so we state it here.

Theorem 11 (Effective Hilbert irreducibility / Quantitative Bertini theorem)**.** *Let $\mathbb{F}$ be a perfect field and $g(\bar{x}) \in \mathbb{F}[x]$ be a degree d irreducible polynomial. Pick tuples, $\boldsymbol{a} = (a_2, \ldots, a_n)$, $\boldsymbol{b} = (b_1, \ldots, b_n)$, $\boldsymbol{c} = (c_1, \ldots, c_n)$ such that every a_i, b_j, c_k is chosen uniformly randomly and independently from a set $S \subset \mathbb{F}$. Consider the bi-variate restriction*

$$\hat{g}(X, Y) = g(X + b_1 Y + c_1, a_2 X + b_2 Y + c_2, \ldots a_n X + b_n Y + c_n)$$

Then

$$P[(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}) \in S^{n-1} \times S^n \times S^n : \hat{f}(X, Y) \text{ is irreducible }] \leq \frac{2d^4}{|S|}$$

Proof. See corollary 2 in [16], remark 11.5.33 and remark 11.5.66 in [24], theorem 1.1 in [23].

Appendix D

PROOFS FROM CHAPTER IV

Lemma 26. *The following are true :*

1. $gcd(R_{|W_i}, B_{|W_i}) = 1 \Rightarrow Sim(C_{|W_i}) = R_{|W_i} + B_{|W_i}$.
2. $srank(C_{|W_i}) = min(r, srank(C))$.
3. $Res(C_{|W_i}) = Res(C)_{|W_i}$ *with high probability.*

Proof. We prove them one by one below:

1. Let R_1, B_1 be any linear factors of R, B respectively. By Part 1 and 3 of Lemma 8 we know that $[R_{1|W_i}], [B_{1|W_i}]$ are distinct forms. This implies that $R_{1|W_i}, B_{1|W_i}$ are LI further implying $gcd(R_{|W_i}, B_{|W_i}) = 1$. $Sim(C_{|W_i}) = Sim(C)_{|W_i}$ follows from this directly.

2. $srank(C_{|W_i}) \leq srank(C)$ since the dimension clearly cannot increase on restriction. $srank(C) \leq r$ since it is the dimension of a set of forms which are projectivisations of linear forms in $x_1, \ldots, x_{r-1}, x_i$. This set has dimension r. Thus $srank(C_{|W_i}) \leq min(r, srank(C))$.

   If $s = min(srank(C), r)$ then there exists $p_1, \ldots, p_s$ forms in $\mathbb{P}(R) \cup \mathbb{P}(B)$. They also belong to $\mathcal{I}(\mathbb{P}(R), \mathbb{P}(B))$, therefore by Parts 1 and 2 of Lemma 8, $p_{1|W_i}, \ldots, p_{s|W_i}$ are independent forms in $\mathbb{P}(R_{|W_i}) \cup \mathbb{P}(B_{|W_i})$ giving $s \leq srank(C_{|W_i})$ and we are done.

3. Consider any irreducible factor of $Res(C)$. It is non-linear and so by effective Hilbert irreducibility (see theorem 11 in appendix C) it remains non-linear (actually irreducible) with high probability as long as $r \geq 3$. There are at the most d such factors and so by a union bound all of them remain non-linear with high probability. This implies that $Res(C)_{|W_i}$ is a product of non-linear irreducibles dividing $C(\bar{x})_{|W_i}$ and thus divides $Res(C_{|W_i})$. The other direction is simpler. Note that $Res(C_{|W_i})$ divides $C(\bar{x})_{|W_i} = Gcd(C)_{|W_i} Int(C)_{|W_i} Res(C)_{|W_i}$. The product $Gcd(C)_{|W_i} Int(C)_{|W_i}$ only has linear factors and thus $Res(C_{|W_i})$ divides $Res(C)_{|W_i}$.

□

Proof of Part 1 in Lemma 9 above also tells us that the multi-set $\mathbb{P}(R_{|W_i}), \mathbb{P}(R_{|W_i})$ are the same as the multi-sets $\mathbb{P}(R)_{|W_i}, \mathbb{P}(B)_{|W_i}$. We will try to reconstruct multi-sets containing them and then glue these reconstructions. But first we determine what set to reconstruct. In Part 2 of Lemma 9 above if we assume r to be any constant $\geq R(3,\mathbb{F}) + 2$, we get that $srank(C_{|W_i}) \geq R(3,\mathbb{F}) + 2$. This enables us to use the Structure Theorem for Factoring Forms of $Res(C_{|W_i})$. We use the shorthand $\mathcal{P}_i = \mathcal{P}(Res(C_{|W_i}))$ for the factoring forms of $Res(C_{|W_i})$.

Lemma 27. *The following hold:*

1. *For every $p \in \mathcal{P}$ and $i \in \{r,\ldots,n\}$, $p_{|W_i} \in \mathcal{P}_i$.*
2. *For any $p_r \in \mathcal{P}_r$, there is at-most one $p_i \in \mathcal{P}_i, i \in \{r+1,\ldots,n\}$ it can be glued to.*

Proof. 1. Let $p = [l] \in \mathcal{P}$ and assume $l = \alpha_1 x_1 + \ldots \alpha_n x_n$ where $\alpha_i \neq 0$ and $\alpha_j = 0$ for all $j < i$. Recall $ker(p) = \{(x_1,\ldots,x_n) \in \mathbb{F}^n : x_i = -\sum_{j=i+1}^{n} \frac{\alpha_j}{\alpha_i} x_j\}$. By lemma 8 $p_{|W_i}$ is defined for all $i \in \{r,\ldots,n\}$. We show that $p_{|W_i}$ is a factoring form for $Res(C_{|W_i})$ and therefore belongs to $\mathcal{P}_i$. By lemma 9, $Res(C_{|W_i}) = Res(C)_{|W_i}$. Also note that $ker(p_{|W_i}) = ker(p) \cap W_i$. This gives us

$$Res(C_{|W_i})_{|ker(p)\cap W_i} = Res(C)_{|W_i|_{ker(p)\cap W_i}} = Res(C)_{|ker(p)\cap W_i} = Res(C)_{|ker(p)|_{ker(p)\cap W_i}}.$$

Since $Res(C_{|W_i})$ has no linear factors, none of the expressions above is zero. Also since $p \in \mathcal{P}$, $Res(C)_{|ker(p)}$ is a non-zero product of linear forms $\Rightarrow$ $Res(C)_{|ker(p)|_{ker(p)\cap W_i}}$ is a non-zero product of linear forms $\Rightarrow$ $p_{|W_i} \in \mathcal{P}_i$ (by the equation above).

2. Let $p_r \in \mathcal{P}_r$ and assume there exists distinct $p_i, p_i' \in \mathcal{P}_i$ such that (p_r, p_i) and (p_r, p_i') are gluable. Clearly $p_i, p_i' \in \mathcal{I}(\mathbb{P}(R_{|W_i}), \mathbb{P}(B_{|W_i}))$ (using theorem 5 since r is high enough and also using part 1 of lemma 9). It's easy to see using part 2 of lemma 8 that $\mathcal{I}(\mathbb{P}(R_{|W_i}), \mathbb{P}(B_{|W_i})) \subset \mathcal{I}(\mathbb{P}(R), \mathbb{P}(B))_{|W_i}$. Now part 3 of lemma 8 implies that $p_{i|_{W_{r-1}}}, p'_{i|_{W_{r-1}}}$ are distinct which is a contradiction to both (p_r, p_i) and (p_r, p_i') being gluable.

□

Appendix E

PROOFS FROM CHAPTER V

Lemma 28. *The following holds:*

1. *If* $r_1,\ldots,r_k \in \mathcal{R}$*, then for any* $p \in \mathcal{G}(r_1,\ldots,r_k)$

   $(fl(r_1,\ldots,r_k,p)\backslash fl(r_1,\ldots,r_k))$ *and* $\mathcal{R}'(r_1,\ldots,r_k)\cup\mathcal{B}$ *intersect non-trivially.*

2. *If* $b_1,\ldots,b_k \in \mathcal{B}$*, then for any* $p \in \mathcal{G}(b_1,\ldots,b_k)$

   $(fl(b_1,\ldots,b_k,p)\backslash fl(b_1,\ldots,b_k))$ *and* $\mathcal{B}'(b_1,\ldots,b_k)\cup\mathcal{R}$ *intersect non-trivially.*

3. $\mathcal{R}(r_1,\ldots,r_k) \neq \phi \Rightarrow \mathbb{P}(Int(C)) \subset \mathcal{G}(r_1,\ldots,r_k)$*. Similarly* $\mathcal{B}(b_1,\ldots,b_k) \neq \phi \Rightarrow \mathbb{P}(Int(C)) \subset \mathcal{G}(b_1,\ldots,b_k)$*.*

4. *Given independent points* $r_1,\ldots,r_k \in \mathcal{R}$ *(*$b_1,\ldots,b_k \in \mathcal{B}$ *resp.),* $r \in \mathcal{R}$ *(*$b \in \mathcal{B}$ *resp.) the multi-set* $\mathbb{P}(Gcd(C)Int(C))$ *and the set* $\mathcal{P}$*, there exists efficient algorithms to compute the multi-set* $\mathbb{P}(M_2) \setminus \mathcal{G}(r_1,\ldots,r_k)$ *(*$\mathbb{P}(M_1) \setminus \mathcal{G}(b_1,\ldots,b_k)$ *resp.) and multi-sets of lines* $\mathcal{L}(r,\mathcal{G}(r_1,\ldots,r_k))$ *(*$\mathcal{L}(b,\mathcal{G}(b_1,\ldots,b_k))$ *resp.).*

Proof. 1. Let $p \in \mathcal{G}(r_1,\ldots,r_k)$. By definition of $\mathcal{G}(r_1,\ldots,r_k)$ there exists at least two distinct points p_1,p_2 from $\mathcal{P}$ on $fl(r_1,\ldots,r_k,p) \setminus fl(r_1,\ldots,r_k)$. If any of p_1,p_2 belongs to $\mathcal{B}$ we are done. If any of them belongs to $\mathcal{P} \setminus \mathcal{R} \cup \mathcal{B}$, then since $k \geq R(3,\mathbb{F}) + 2$ by lemma 5 there exists $i \in [k]$ such that $fl(r_i,p)$ contains a point b from $\mathcal{B}$. Clearly b then belongs $fl(r_1,\ldots,r_k,p) \setminus fl(r_1,\ldots,r_k)$ and we are done. So the only case left is when both $p_1,p_2 \in \mathcal{R}$. If any of p_1,p_2 (say p_1) belongs to $\mathcal{R}(r_1,\ldots,r_k)$ then $p_2 \notin fl(r_1,\ldots,r_k,p_1) \setminus fl(r_1,\ldots,r_k) = fl(r_1,\ldots,r_k,p) \setminus fl(r_1,\ldots,r_k)$ (by definition of points in $\mathcal{R}(r_1,\ldots,r_k)$). So we arrive at a contradiction and so p_1 either belongs to $\mathcal{B}$ or $\mathcal{R}'(r_1,\ldots,r_k)$. Hence proved.

2. This is identical to the above proof.

3. Consider $r_{k+1} \in \mathcal{R}(r_1,\ldots,r_k)$. By definition $r_{k+1} \notin fl(r_1,\ldots,r_k)$ and $fl(r_1,\ldots,r_k,r_{k+1}) \cap (\mathcal{R} \cup \mathcal{B}) \subset fl(r_1,\ldots,r_k) \cup \{r_{k+1}\}$. Consider any point

$p \in \mathbb{P}(Int(C))$. By restricting to $ker(p)$ and following an argument we've seen multiple times before, there exists $b_{k+1} \in \mathcal{B}$ such that $b_{k+1} \in fl(r_{k+1}, p)$. If $p \in fl(r_1, \ldots, r_k)$, then this implies that $b_{k+1} \in fl(r_1, \ldots, r_k, r_{k+1}) \setminus fl(r_1, \ldots, r_k)$ which is a contradiction since no point from $\mathcal{B}$ lies on the set $fl(r_1, \ldots, r_k, r_{k+1}) \setminus fl(r_1, \ldots, r_k)$. Next since $k \geq 3$, by the above argument there exist three points $b_1, b_2, b_3 \in \mathcal{B}$ such that $b_i \in fl(p, r_i), i \in [3]$. Also note that all $p \neq r_i$ and $p \neq b_i$ for $i \in [3]$ (by lemma 4). We claim that at least two of the b_i's are different, otherwise the set $\{b_1, b_2, b_3, p\}$ has dimension ≤ 2. Let p, q (or just p) be a basis for this set. This is not possible since the three dimensional flat, $fl(r_1, r_2, r_3) \subset fl(p, q)$ (as $r_i \in fl(p, b_i), i \in [3]$). So at least two b_i's are distinct. Also the b_i's don't lie on $fl(r_1, \ldots, r_k)$ since p does not. Therefore we found two distinct points from $\mathcal{P}$ on $fl(r_1, \ldots, r_k, p) \setminus fl(r_1, \ldots, r_k)$ for every $p \in \mathbb{P}(Int(C)) \Rightarrow \mathbb{P}(Int(C)) \subset \mathcal{G}(r_1, \ldots, r_k)$. The other part is identical.

4. Iterate through every point $p \in \mathbb{P}(Gcd(C)Int(C))$. If $p \notin fl(r_1, \ldots, r_k)$, by iterating through $\mathcal{P}$ check whether the set $fl(r_1, \ldots, r_k, p) \setminus fl(r_1, \ldots, r_k)$ contains at most one distinct points from $\mathcal{P}$. This can be checked by simple linear algebra. This computes the multi-set $\mathbb{P}(M_2) \setminus \mathcal{G}(r_1, \ldots, r_k)$.

   Recall that using algorithm 4 we already know how to compute the lines $\mathcal{L}(r, \mathbb{P}(M_2))$. Start with $\mathcal{L}(r, \mathcal{G}(r_1, \ldots, r_k) \cup \mathcal{B}) = \mathcal{L}(r, \mathbb{P}(M_2))$. Compute the multi-set $\mathcal{L}(r, \mathbb{P}(M_2)) \setminus (\mathcal{L}(r, \mathbb{P}(M_2) \setminus \mathcal{G}(r_1, \ldots, r_k)))$. This computes the required multi-set of lines $\mathcal{L}(r, \mathcal{G}(r_1, \ldots, r_k) \cup \mathcal{B})$.

□

Lemma 29. *The following hold for all independent $r_1, \ldots, r_k \in \mathcal{R}$.*

1. $\mathcal{R}(r_1, \ldots, r_k) \subset \mathcal{S}(r_1, \ldots, r_k) \subset \mathbb{P}(R)$.
2. $\mathcal{B}(r_1, \ldots, r_k) \subset \mathcal{S}(r_1, \ldots, r_k) \subset \mathbb{P}(B)$.
3. *Given $r_1, \ldots, r_k$ ($b_1, \ldots, b_k$ resp.), the set $\mathcal{P}$, multi-set $\mathbb{P}(Gcd(C)Int(C))$ and black-box $\mathbb{B}_{Res}$, there exist efficient algorithms to compute $\mathcal{S}(r_1, \ldots, r_k)$ ($\mathcal{S}(b_1, \ldots, b_k)$ resp.).*

Proof. 1. Let $r_{k+1} \in \mathcal{R}(r_1, \ldots, r_k)$. This implies that $r_{k+1} \notin fl(r_1, \ldots, r_k)$ and $fl(r_1, \ldots, r_k, r_{k+1}) \cap (\mathcal{R} \cup \mathcal{B}) \subset fl(r_1, \ldots, r_k) \cup \{r_{k+1}\}$. Assume $r_{k+1} \notin \mathcal{S}(r_1, \ldots, r_k)$ i.e. on one of the lines say $fl(r_1, r_{k+1})$ there is a point $p \in$

$\mathcal{G}(r_1,\ldots,r_k)\cup\mathcal{B}$. p cannot be in $\mathcal{B}$ by the choice of r_{k+1}. So $p \in \mathcal{G}(r_1,\ldots,r_k)$. By lemma 18 there exists $p' \in \mathcal{R}'(r_1,\ldots,r_k) \cup \mathcal{B}$ lying on $fl(r_1,\ldots,p) \setminus fl(r_1,\ldots,r_k)$. This implies that $p' \in fl(r_1,\ldots,r_k,r_{k+1}) \setminus fl(r_1,\ldots,r_k)$. Since $p' \in \mathcal{R}^(r_1,\ldots,r_k) \cup \mathcal{B} \subset \mathcal{R} \cup \mathcal{B}$ we conclude that $p' = r_{k+1}$ but that is a contradiction since $\mathcal{R}'(r_1,\ldots,r_k), \mathcal{R}(r_1,\ldots,r_k)$ were complements inside $\mathcal{R}$.

For the other inclusion, let $p \in \mathcal{S}(r_1,\ldots,r_k)$. Then the lines $fl(p,r_i)$ do not intersect $\mathcal{G}(r_1,\ldots,r_k) \cup \mathcal{B}$ for any i. By definition $p \in \mathcal{P}$. $p \notin \mathcal{B}$ by definition. If $p \in \mathcal{P} \setminus (\mathcal{R} \cup \mathcal{B})$ then by matching lemma (lemma 5) there exists $r_i, i \in [k]$ such that $b \in fl(r_i,p)$ a contradiction to the choice of p. Thus $p \in \mathcal{R}$. Hence proved.

2. Exactly identical to the proof above.

3. First compute the multi-set of lines $\mathcal{L}(r,\mathcal{G}(r_1,\ldots,r_k) \cup \mathcal{B})$ using the algorithm given in the last part of lemma 18. Iterate through $p \in \mathcal{P}$ and add it to the set $\mathcal{S}(r_1,\ldots,r_k)$ if none of the lines $fl(r,p)$ belongs to the set $\mathcal{L}(r,\mathcal{G}(r_1,\ldots,r_k) \cup \mathcal{B})$.

□

Lemma 30. *One of the following always holds:*

1. $\exists\, r_1,\ldots,r_k \in \mathcal{R} : |\mathcal{R}(r_1,\ldots,r_k)| \geq v(\delta)|\mathcal{R}|$, *or*
2. $\exists\, b_1,\ldots,b_k \in \mathcal{R} : |\mathcal{B}(b_1,\ldots,b_k)| \geq v(\delta)|\mathcal{R}|$.

Proof. The proof is divided into two cases:

Case I - $|\mathcal{B}| \leq 4\delta|\mathcal{R}|$:

Since $dim(\mathcal{R}) \geq \frac{2Ck}{\delta} + k > \frac{Ck}{\delta}$, we know by Corollary 4, there exist k linearly independent points $r_1,\ldots,r_k \in \mathcal{R}$ such that the set $\mathcal{Y} = \{r \in \mathcal{R} : fl(r_1,\ldots,r_k,r) \cap \mathcal{R} \subset fl(r_1,\ldots,r_k) \cup \{r\}\}$ has size $\geq (1-\delta)|\mathcal{R}|$. That is there are a large number of points in $\mathcal{R}$ forming an ordinary flat with $r_1,\ldots,r_k$ inside $\mathcal{R}$. From this set we throw away all those r's whose flat with $r_1,\ldots,r_k$ contains a point from $\mathcal{B}$ outside $fl(r_1,\ldots,r_k)$. The remaining r's are such that their flats with $\{r_1,\ldots,r_k\}$ are ordinary in $\mathcal{R} \cup \mathcal{B}$. The left over set is exactly the set $\mathcal{R}(r_1,\ldots,r_k)$. We would have thrown away $\leq |\mathcal{B}|$ points overall and therefore the set the left over set i.e. $R(r_1,\ldots,r_k)$ has size $\geq |\mathcal{Y}| - |B| \geq (1-\delta)|\mathcal{R}| - 4\delta|\mathcal{R}| = (1-5\delta)|\mathcal{R}| \geq (3\delta - 4\delta^2)|\mathcal{R}| = v(\delta)|\mathcal{R}|$ (note that when $\delta < \frac{1}{8}$, $1 - 5\delta \geq 3\delta - 4\delta^2$).

Case II - $4\delta|\mathcal{R}| \leq |\mathcal{B}| \leq |\mathcal{R}|$: In this case we will use Corollary 4 again but with $2k$ points. We know that $dim(\mathcal{R} \cup \mathcal{B}) \geq \frac{2Ck}{\delta} + k > \frac{2Ck}{\delta}$ and therefore there are $2k$ linearly independent points $p_1, \ldots, p_{2k} \in \mathcal{R} \cup \mathcal{B}$ such that the set $\mathcal{Y} = \{p \in \mathcal{R} \cup \mathcal{B} : fl(p_1, \ldots, p_{2k}, p) \cap \mathcal{R} \cup \mathcal{B} \subset fl(p_1, \ldots, p_{2k}) \cup \{p\}\}$ has size $\geq (1-\delta)|\mathcal{R} \cup \mathcal{B}| = (1-\delta)(|\mathcal{R}|+|\mathcal{B}|)$ since $\mathcal{R}, \mathcal{B}$ are disjoint. Out of the $2k$ points, by pigeon hole principle at least k, say $p_1, \ldots, p_k$ belong to $\mathcal{R}$ or $\mathcal{B}$. If $p_1, \ldots, p_k \in \mathcal{R}$, we consider the set $\mathcal{R}(p_1, \ldots, p_k) = \{r \in \mathcal{R} : fl(p_1, \ldots, p_k, r) \cap \mathcal{R} \cup \mathcal{B} \subset fl(p_1, \ldots, p_k)\}$. This set clearly contains the set $\mathcal{Y} \setminus \mathcal{B}$ and thus it has size $\geq |\mathcal{Y}| - |\mathcal{B}| = (1-\delta)(|\mathcal{R}|+|\mathcal{B}|) - |\mathcal{B}| \geq (1-\delta)(1+4\delta)|\mathcal{R}| - |\mathcal{R}|$ (since $4\delta|\mathcal{R}| \leq |\mathcal{B}| \leq |\mathcal{R}|$). So we get that the size of the desired set is $\geq ((1-\delta)(1+4\delta) - 1)|\mathcal{R}| = (3\delta - 4\delta^2)|\mathcal{R}| = v(\delta)|\mathcal{R}|$. When $p_1, \ldots, p_k \in \mathcal{B}$, we use the same approach to conclude that the set $\mathcal{B}(p_1, \ldots, p_k) = \{b \in \mathcal{B} : sp\{p_1, \ldots, p_k, b\} \cap \mathcal{R} \cup \mathcal{B} \subset sp\{p_1, \ldots, p_k\} \cup \{b\}\}$ contains the set $\mathcal{Y} \setminus \mathcal{R}$ and therefore has size $\geq |\mathcal{Y}| - |\mathcal{R}| \geq (1-\delta)(|\mathcal{R}|+|\mathcal{B}|) - |\mathcal{R}| \geq ((1-\delta)(1+4\delta) - 1)|\mathcal{R}| = v(\delta)|\mathcal{R}|$. So we know that $|\mathcal{B}(b_1, \ldots, b_k)| \geq v(\delta)|\mathcal{R}|$.

Clearly one of the two cases will hold and we get the desired result.

□

Lemma 31. *Let $v(\delta) = 3\delta - 4\delta^2$. Then for $\delta \in (0, \frac{1}{8})$,*

$$\frac{(2 - v(\delta))}{v(\delta)} \leq \frac{1-\delta}{\delta}.$$

Proof. Note that

$$\frac{(2 - v(\delta))}{v(\delta)} = \frac{2 - 3\delta + 4\delta^2}{3\delta - 4\delta^2}.$$

We know that $\delta < \frac{1}{8}$ and so $4\delta^2 < \delta \Rightarrow 2\delta < 3\delta - 4\delta^2 \Rightarrow$

$$2\delta - 3\delta^2 + 4\delta^3 < 3\delta - 4\delta^2 - 3\delta^2 + 4\delta^3 \Rightarrow$$

$$\delta(2 - 3\delta + 4\delta^2) < (1-\delta)(3\delta - 4\delta^2).$$

We know that $3\delta - 4\delta^2 > 0$ and $\delta > 0$ as $\delta \in (0, \frac{1}{8})$. Therefore we see that

$$\frac{(2 - v(\delta))}{v(\delta)} = \frac{2 - 3\delta + 4\delta^2}{3\delta - 4\delta^2} < \frac{1-\delta}{\delta}.$$

□

Appendix F

TOOLS FROM INCIDENCE GEOMETRY

Later in the paper we will use the quantitative version of Sylvester-Gallai Theorem from [3] and [6]. In this subsection we do preparation for the same. The results in [3] and [6] are given for linear and affine spaces.

It is well known that all incidence results continue to hold if we consider projective spaces instead of affine or vector spaces. In this chapter we just state the results as they were given in [3], but when we use them we use the projective space analogue. We encourage the reader to show equivalence of the two results.

Our main application will also involve a corollary we prove towards the end of this subsection.

Definition 29 ([3])**.** Let S be a set of n distinct points in complex space $\mathbb{C}^r$ (or projective space $\mathbb{P}(\mathbb{C}^r)$). A k- flat is ordinary if its intersection with S is contained in the union of a $(k-1)$ flat and a single point.

Definition 30 ([3])**.** Let S be a set of n distinct points in $\mathbb{C}^r$. S is called a $\delta - SG_k^\star$ configuration if for every independent $s_1,\ldots,s_k \in S$ there are at least δn points $t \in S$ such that either $t \in fl(\{s_1,\ldots,s_k\})$ or the $k-$flat $fl(\{s_1,\ldots,s_k,t\})$ contains a point outside $fl(\{s_1,\ldots,s_k\}) \cup \{t\}$.

Theorem 12 ([3])**.** *Let S be a $\delta - SG_k$ configuration then $dim(S) \leq O((\frac{k}{\delta})^2)$.*

This bound on the dimension of S was further improved by Dvir et. al. in [6]. The latest version now states

Theorem 13 ([6])**.** *Let S be a $\delta - SG_k^\star$ configuration then $dim(S) \leq \alpha\frac{k}{\delta}$ for some $\alpha > 0$.*

Corollary 4. *Let $dim(S) > \alpha\frac{k}{\delta}$ (for C in above theorem) then S is not a $\delta - SG_k^\star$ configuration i.e. there exists a set of independent points $\{s_1,\ldots,s_k\}$ and $\geq (1-\delta)n$ points t such that $fl(\{s_1,\ldots,s_k,t\})$ is an ordinary $k - flat$. That is:*

- $t \notin fl(\{s_1,\ldots,s_k\})$
- $fl(\{s_1,\ldots,s_k,t\}) \cap S \subset fl(\{s_1,\ldots,s_k\}) \cup \{t\}$.

Lemma 32 (Bi-chromatic semi-ordinary line)**.** *Let X and Y be disjoint finite sets in $\mathbb{C}^r$ satisfying the following conditions.*

1. $dim(Y) > \frac{\alpha}{\delta}$.
2. $|Y| \leq c|X|$ *with* $c < \frac{1-\delta}{\delta}$.

Then there exists a line l such that $|l \cap Y| = 1$ and $|l \cap X| \geq 1$

Proof. We consider two cases:

Case 1 : $c|X| \geq |Y| \geq |X|$

Since $dim(Y) > C_1$, using the corollary above for $S = X \cup Y, k = 1$ we can get a point $s_1 \in X \cup Y$ for which there exist $(1-\delta)(|X|+|Y|)$ points t in $X \cup Y$ such that $t \notin fl\{s_1\}$ and $fl\{s_1,t\}$ is elementary. If $s_1 \in X$ then $(1-\delta)(|X|+|Y|) - |X| \geq (1-2\delta)|X| > 0$ of these flats intersect Y and thus we get such a line l. If $s_1 \in Y$ then $(1-\delta)(|X|+|Y|) - |Y| \geq ((1-\delta)(\frac{1}{c}+1) - 1)|Y| > 0$ of these flats intersect X giving us the required line l with $|l \cap X| = 1$ and $|l \cap Y| = 1$.

Case 2: $|Y| \leq |X|$

Now choose a subset $X_1 \subseteq X$ such that $|X_1| = |Y|$. Now using the same argument as above for $S = X_1 \cup Y$ there is a point $s_1 \in X_1 \cup Y$ such that $(1-\delta)(|X_1|+|Y|) = 2(1-\delta)|Y| = 2(1-\delta)|X_1|$ flats through it are elementary in $X_1 \cup Y$. If $s_1 \in Y$ $(1-2\delta)|Y| > 0$ of these flats intersect X_1. If $s_1 \in X_1$, $(1-2\delta)|X_1| > 0$ of these flats intersect Y. In both these above possibilities the flat intersects Y and X_1 in exactly one point each. But it may contain more points from $X \setminus X_1$ so we can find a line l such that $|l \cap Y| = 1$ and $|l \cap X| \geq 1$.