Electronic Colloquium on Computational Complexity, Report No. 73 (2010)

Algorithms for Arithmetic Circuits

Neeraj Kayal *
neeraka@microsoft.com

April 21, 2010

Abstract

Given a multivariate polynomial $f(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$ as an arithmetic circuit we would like to efficiently determine:

1. **Identity Testing.** Is $f(\mathbf{X})$ identically zero?
2. **Degree Computation.** Is the degree of the polynomial $f(\mathbf{X})$ at most a given integer d .
3. **Polynomial Equivalence.** Upto an invertible linear transformation of its variables, is $f(\mathbf{X})$ equal to a given polynomial $g(\mathbf{X})$.

The algorithmic complexity of these problems is studied. Some new algorithms are provided here while some known ones are simplified. For the first problem, a deterministic algorithm is presented for the special case where the input circuit is a "sum of powers of sums of univariate polynomials" . For the second problem, a coRP$^{\text{PP}}$-algorithm is presented. Finally, randomized polynomial-time algorithms are presented for certain special cases of the third problem.

1 Introduction

Polynomials are used extensively in computer algebra. The naive way of encoding polynomials is to write down the list of the coefficients of all monomials but this is not always suitable, especially when we are dealing with multivariate polynomials where even low degree polynomials may have an exponentially large number of monomials. [1] Arithmetic circuits can sometimes remedy this situation since their size is in general much smaller than the list of all coefficients. Arithmetic circuits also provide a natural and elegant model for computing polynomials. Of particular interest in computer science is the determination of the smallest arithmetic circuit computing a given polynomial.

Having found a versatile and compact way to represent polynomials, the focus now naturally shifts to algorithms for basic operations involving polynomials represented as arithmetic circuits. Substantial progress has been made in this direction. Efficient (randomized) algorithms have been devised for testing the equality of two polynomials (see e.g. [Sch80]), for computing the gcd of two polynomials [Kal88], for factoring a low-degree multivariate polynomial [Kal89] and for computing the set of partial derivatives of a polynomial [BS83]. At times algorithmic results such as the one of Baur and Strassen [BS83] have yielded lower bounds on the arithmetic circuit complexity of a polynomial.

Though compact, this manner of representing polynomials is not always amenable to efficient computation - some very natural questions become hard when dealing with arithmetic circuits. It is known for example that computing the coefficient of a given monomial is #P- hard [Mal07, AKPBM06]. Thus it seems natural to wonder which properties of the polynomial described by a given arithmetic circuit can be computed efficiently. Zeroness/nonzeroness and degree are clearly basic algebraic properties of a polynomial and with this, we dispense with the need to motivate the associated computational problems. We now motivate polynomial equivalence testing.

*Microsft Research India. Part of this work done while the author was at DIMACS, Rutgers University.

[1] A multivariate polynomial is said to be a low degree polynomial if its degree is bounded above by a polynomial in the number of variables.

ISSN 1433-8092

Motivation. We consider the task of understanding polynomials upto invertible linear transformations of the variables. We will say that two n-variate polynomials $f(\mathbf{X})$ and $g(\mathbf{X})$ are equivalent, denoted $f \sim g$ if there exists an invertible linear transformation $A \in \mathbb{F}^{n\times n}$ such that $f(\mathbf{X}) = g(A \cdot \mathbf{X})$. The following well-known lemma constructively classifies quadratic polynomials upto equivalence.

Lemma 1. (Structure of quadratic polynomials). *Let $\mathbb{F}$ be an algebraically closed field of characteristic different from 2. For any homogeneous quadratic polynomial $f(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$ there exists an invertible linear transformation $A \in \mathbb{F}^{n\times n}$ and a natural number $1 \leq r \leq n$ such that*

$$f(A \cdot \mathbf{X}) = x_1^2 + x_2^2 + \ldots + x_r^2.$$

Moreover, the linear transformation A involved in this equivalence can be computed efficiently. Furthermore, two quadratic forms are equivalent if and only if they have the same number r of variables in the above canonical representation.

This lemma allows us to understand many properties of a given quadratic polynomial. We give one example.

Example 2. Formula size of a quadratic polynomial. *Let Φ be an arithmetic formula. The size of the formula Φ, denoted $L(\Phi)$ is defined to be the number of multiplication gates in it.* [2] *For a polynomial f, $L(f)$ is the size of the smallest formula computing f. Then for a homogeneous quadratic polynomial f, we have that*

$$L(f) = \left\lceil \frac{r}{2} \right\rceil,$$

where r is as given by lemma 1.

Sketch of Proof: Let $f(\mathbf{X})$ be equivalent to $x_1^2+\ldots+x_r^2$. Using the identity $y^2+z^2 = (y+\sqrt{-1}z)\cdot(y-\sqrt{-1}z)$, we can replace the *sum of squares* representation above with a *sum of products of pairs*. That is,

$$f(\mathbf{X}) \sim \begin{cases} x_1x_2 + x_3x_4 + \ldots + x_{r-1}x_r & \text{if } r \text{ is even,} \\ x_1x_2 + x_3x_4 + \ldots + x_{r-2}x_{r-1} + x_r^2 & \text{if } r \text{ is odd,} \end{cases}$$

Let $g(\mathbf{X}) \stackrel{\text{def}}{=} x_1x_2+x_3x_4+\ldots+x_{r-1}x_r$. For any homogeneous quadratic polynomial, there is a homogeneous $\Sigma\Pi\Sigma$ (sum of product of sums) formula of minimal formula size for computing that polynomial. Using this the formula size for $g(\mathbf{X})$ can be deduced to be $\frac{r}{2}$ and furthermore that $L(f)$ is exactly $\left\lceil \frac{r}{2} \right\rceil$.

□

No generalization of the above example to higher degree polynomials is known. Indeed, no *explicit* family of *cubic* (i.e. degree three) polynomials is known which has superlinear formula-size complexity. [3] One might naïvely hope that an appropriate generalization of lemma 1 to cubic polynomials might shed some light on the formula size complexity of a cubic polynomial. That is, one wants a characterization of cubic polynomials upto equivalence. Despite intensive effort (cf. [MH74, Har75]), no 'explicit' characterization of cubic forms was obtained. In a recent work, Agrawal and Saxena [AS06] 'explained' this lack of progress: they showed that the well-studied but unresolved problem of graph isomorphism reduces to the problem of testing equivalence of cubic forms. A simpler proof of a slightly weaker version of their result is presented in example 3. This means that the polynomial equivalence problem is likely to be very challenging, even when the polynomials are given verbosely via a list of coefficients. In this work, we do not tackle the general polynomial equivalence problem, but rather some special cases of it which are motivated by the desire to present a given polynomial in an "easier way". The "easier" ways of presenting that we look at are motivated by the characterization of quadratic polynomials as given in lemma 1 and example 2. Let us describe these special cases of polynomial equivalence.

[2] The size of an arithmetic formula is usually defined as the total number of gates, addition as well as multiplication, in the formula. Our definition is equivalent to this definition upto quadratic factors. In many situations it is more convenient to work with the number of multiplication gates as a measure of the size of the formula. This definition has the further desirable property that if two polynomials are equivalent then they have the same formula size.

[3] A dimension counting arguments assures us that over every field $\mathbb{F}$, there exists a family of $\{f_n\}$ of cubic n-variate polynomials which has formula size $\Omega(n^{\frac{3}{2}})$. No explicit family of cubic polynomials with superlinear formula size is known.

The integer r of lemma 1 is referred to in the literature as the rank of the quadratic form. Notice that upto equivalence, it is the smallest number of variables which the given polynomial f depends on. One then asks whether a given polynomial is equivalent to another polynomial which depends on a fewer number of variables. The canonical form for a quadratic polynomial is as a sum of squares of linear forms. The natural question for higher degree polynomials then is whether the given polynomial is a sum of appropriate powers of linear forms. i.e whether a given polynomial of degree d is equivalent to

$$x_1^d + x_2^d + \ldots + x_n^d.$$

It should be noted that unlike quadratic forms, not every polynomial of degree $d \geq 3$ can be presented in this fashion. We devise an efficient randomized algorithm for this special case of equivalence testing. We then consider some other classes of polynomials and do equivalence testing for those. In particular, we devise algorithms to test whether the given polynomial is equivalent to an elementary symmetric polynomial. The algorithms that we devise can be generalized quite a bit and these generalizations (which we call polynomial decomposition and polynomial multilinearization) are explained in section 9 of this article. Before we go on let us motivate our consideration of such special cases by obtaining a hardness result for polynomial equivalence.

Example 3. *Graph Isomorphism many-one reduces to testing equivalence of cubic polynomials.* [4]

Sketch of Proof : Let the two input graphs be $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$. Let $|V_1| = |V_2| = n$. Define the cubic polynomial f_{G_1} as follows:

$$f_{G_1}(\mathbf{X}) \stackrel{\text{def}}{=} \sum_{i=1}^{n} x_i^3 + \sum_{\{i,j\} \in E_1} x_i \cdot x_j.$$

Polynomial f_{G_2} is defined analogously. It suffices to prove that G_1 is isomorphic to G_2 if and only if f_{G_1} is equivalent to f_{G_2}. The forward direction is easy. For the other direction, assume that f_{G_1} is equivalent to f_{G_2} via the matrix A. i.e.

$$f_{G_1}(A \cdot \mathbf{X}) = f_{G_2}(\mathbf{X}).$$

Then the homogeneous cubic part of f_{G_1} must be equivalent, via A, to the homogeneous cubic part of f_{G_2} and the same thing holds for the homogeneous quadratic part. Corollary 23 describes the automorphisms of the polynomial $x_1^3 + \ldots + x_n^3$ and it says that there exists a permutation $\pi \in S_n$ and integers $i_1, i_2, \ldots, i_n \in \{0, 1, 2\}$ such that

$$A \cdot \mathbf{X} = (\omega_1^i \cdot x_{\pi(1)}, \omega_2^i \cdot x_{\pi(2)}, \ldots, \omega_n^i \cdot x_{\pi(n)}).$$

Using the equivalence via A of the homogeneous quadratic parts of f_{G_1} and f_{G_2}, we obtain that π in fact describes an isomorphism from G_1 to G_2.

□

1.1 Previous work and our results

Identity Testing

It is well known that polynomial identity testing, the problem of determining whether a given arithmetic circuit computes the identically zero polynomial or not, admits a randomized algorithm [Sch80, Zip79]. No deterministic algorithm is known. In recent years, the problem has been under intense attack and a number of special cases have been tackled and resolved [LV98, KS01, AB03, RS04, DS05, KS06, Sax08, SV08, KS09]. The reason is twofold. Firstly, besides being a natural problem, many other interesting algorithmic problems such as primality testing [AB03] and bipartite matching [MVV87] are special cases of this problem. Perhaps more importantly, it is known that derandomizing identity testing will lead to arithmetic circuit lower bounds

[4] Agrawal and Saxena [AS06] showed the stronger result that graph isomorphism reduces to testing equivalence of *homogeneous* cubic polynomials, also known as cubic forms .

[Agr05, IK03]. Here we consider the special case of identity testing where the circuit is *a sum of powers of sums of univariate forms.* We will test whether an expression of the form

$$\sum_{i=1}^{k} a_i(g_{i1}(x_1) + g_{i2}(x_2) + \ldots + g_{in}(x_n))^d$$

is identically zero or not. This situation has been examined before - a deterministic polynomial time algorithm was devised by Saxena [Sax08]. The algorithm that we devise is self contained and arguably much simpler. [5]

Degree of a polynomial

As an algorithmic problem the complexity of computing the degree of the polynomial computed by a given arithmetic circuit (shortened to DegSLP) has been studied in by Allender et al [ABKPM09] and by Koiran and Perifel [KP07]. The first work gives an upper bound in the counting hierarchy while the second work improves it to coNP^{PP}. We improve this further to coRP^{PP}. Furthermore, our algorithm works even when the finite field itself is part of the input rather than being fixed.

Polynomial Equivalence

Polynomial equivalence is a relatively less well-studied problem. The problem of minimizing the number of variables in a polynomial upto equivalence was considered earlier from a more practical perspective and an efficient algorithm for verbosely represented polynomial (i.e. polynomials given via a list of coefficients) was devised by Carlini [Car06] and implemented in the computer algebra system **CoCoA**. We observe here that this problem admits an efficient randomized algorithm even when the polynomial is given as an arithmetic circuit. In his thesis [Sax06], Saxena notes that the work of Harrison [Har75] can be used to solve certain special cases of polynomial equivalence but the time complexity deteriorates exponentially with the degree. In particular, the techniques of Harrison imply that one can *deterministically* test whether a given polynomial is equivalent to $x_1^d + \ldots + x_n^d$ but the time taken is exponential in the degree d. Here we give a *randomized* algorithm with running time polynomial in d and the size of the input circuit. We also present a new randomized polynomial time algorithm to test whether a given polynomial is equivalent to an elementary symmetric polynomial. Our algorithms generalize somewhat and we obtain efficient randomized algorithms for polynomial decomposition and multilinearization. See theorems 31 and 30 for the precise statements of these generalizations and the degenerate cases which need to be excluded.

Linear Dependence among polynomials

One natural subproblem is common to many of these algorithms. We call it POLYDEP and it is studied in section 3. There we show the relationship of this problem to identity testing and give an efficient randomized algorithm for it.

Organization

The rest of this article is organized as follows. We fix some notation and terminology in section 2. We then define and investigate the problem of computing linear dependencies among polynomials in section 3. Thereafter, we look at identity testing in section 4. We move on to the complexity of computing the degree in section 5. Then in section 6, we devise an efficient randomized algorithm to minimize the number of variables occuring in a given polynomial. Thereafter we consider polynomial equivalence in sections 7, 8 and 9. We conclude by posing some new problems.

[5] The algorithm of Saxena [Sax08] uses a noncommutative formula identity testing algorithm by Raz and Shpilka [RS04]

2 Notation

We will abbreviate the vector of indeterminates $(x_1, x_2, \ldots, x_n)$ by $\mathbf{X}$. The set $\{1, 2, \ldots, n\}$ will be abbreviated as $[n]$. We will be consider polynomials in n variables over some field $\mathbb{F}$. We will need the notion of the formal degree of an arithmetic circuit.

Definition 4. *The* formal degree *of a vertex in an arithmetic circuit is defined inductively as follows:*

- *the formal degree of an input node is* 1.
- *the formal degree of a + gate is the maximum of the formal degrees of its entries.*
- *the formal degree of a × gate is the sum of the formal degrees of its entries.*

The formal degree of an arithmetic circuit is the formal degree of its output node.

A polynomial of degree one is called an affine form. Affine forms whose constant term is zero are called linear forms. We will say that a given polynomial $f(\mathbf{X})$ is a *low-degree* polynomial if its degree is bounded above by a polynomial in the size of the arithmetic circuit computing $f(\mathbf{X})$.
For a linear transformation

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ a_{21} & \cdots & a_{2n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \in \mathbb{F}^{n \times n},$$

we shall denote by $A \cdot \mathbf{X}$ the tuple of polynomials

$$(a_{11}x_1 + \ldots + a_{1n}x_n, \quad \ldots \quad , a_{n1}x_1 + \ldots + a_{nn}x_n).$$

Thus, for a polynomial $f(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$, $f(A \cdot \mathbf{X})$ denotes the polynomial obtained by making the linear transformation A on the variables in f.
We also set up a compact notation for partial derivatives and substitution maps. Let $f(x_1, \ldots, x_n) \in \mathbb{F}[x_1, \ldots, x_n]$ be a polynomial. Then:

- **Sets of derivatives.** $\partial^k f$ shall denote the set of k-th order partial derivatives of f. Thus $\partial^1 f$, abbreviated as ∂f, shall equal

  $$\left\{ \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n} \right\}.$$

  $\partial^2 f$ is the set

  $$\left\{ \frac{\partial^2 f}{\partial x_i \cdot \partial x_j} : 1 \leq i < j \leq n \right\},$$

  and so on.

- **Derivatives and substitution maps.** We also have

  $$\partial_i f \stackrel{\text{def}}{=} \frac{\partial f}{\partial x_i} \quad \text{and} \quad \sigma_i f \stackrel{\text{def}}{=} f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n).$$

3 Linear dependencies among polynomials

In this section we isolate and study a subproblem which is common to many of the problems studied here. We call it the problem of computing linear dependencies among polynomials and denote it by POLYDEP.

Definition 5. *Let* $\mathbf{f}(\mathbf{X}) \stackrel{\text{def}}{=} (f_1(\mathbf{X}), f_2(\mathbf{X}), \ldots, f_m(\mathbf{X})) \in (\mathbb{F}[\mathbf{X}])^m$ *be a vector of polynomials over a field* $\mathbb{F}$*. The set of* $\mathbb{F}$-linear dependencies *in* $\mathbf{f}$*, denoted* $\mathbf{f}^\perp$*, is the set of all vectors* $\mathbf{v} \in \mathbb{F}^m$ *whose inner product with* $\mathbf{f}$ *is the zero polynomial, i.e.,*

$$\mathbf{f}^\perp \stackrel{\text{def}}{=} \Big\{(a_1, \ldots, a_m) \in \mathbb{F}^m : a_1 f_1(\mathbf{X}) + \ldots + a_m f_m(\mathbf{X}) = 0\Big\}$$

If $\mathbf{f}^\perp$ *contains a nonzero vector, then the* f_i*'s are said to be* $\mathbb{F}$*-linearly dependent.*

The set $\mathbf{f}^\perp$ is clearly a linear subspace of $\mathbb{F}^m$. In many of our applications, we will want to efficiently compute a basis of $\mathbf{f}^\perp$ for a given tuple $\mathbf{f} = (f_1(\mathbf{X}), \ldots, f_m(\mathbf{X}))$ of polynomials. Let us capture this as a computational problem.

Definition 6. *The problem of computing linear dependencies between polynomials, denoted POLYDEP, is defined to be the following computational problem: given as input* m *arithmetic circuits computing polynomials* $f_1(\mathbf{X}), \ldots, f_m(\mathbf{X})$ *respectively, output a basis for the subspace* $\mathbf{f}^\perp = (f_1(\mathbf{X}), \ldots, f_m(\mathbf{X}))^\perp \subseteq \mathbb{F}^m$*.*

Clearly, identity testing is a special case of POLYDEP (put $m = 1$). Like identity testing, POLYDEP admits an efficient randomized algorithm. This randomized algorithm will form a basic building block of our algorithms Section 3.1. We do not know whether POLYDEP is equivalent (via deterministic polynomial-time Turing reductions) to identity testing. In practice what is seen is that the known polynomial-time algorithms to do identity testing for certain special families of arithmetic circuits can all be tweaked into polynomial-time algorithms to solve POLYDEP for such families of arithmetic circuits. In the other direction we have:

Exercise 7. *Let the* search version *of identity testing be the following computational problem: given an arithmetic circuit* $\mathcal{C}$ *output an* $\mathbf{a} \in \mathbb{F}^n$ *such that* $\mathcal{C}(\mathbf{a}) \neq 0$*, if such an* $\mathbf{a}$ *exists; else output* 'No such $\mathbf{a}$'. [6] *Show that POLYDEP reduces to the search version of identity testing.* [7]

Notice that for low-degree arithmetic circuits [8], the search version of identity testing is equivalent (via polynomial-time turing reductions) to the decision version. For this reason, the reader should think of POLYDEP as being a problem that is "morally equivalent" to identity testing.

In section 4, we will devise a deterministic polynomial-time algorithm for POLYDEP over a special family of polynomials. As a corollary, it gives us a deterministic polynomial-time algorithm for identity testing. In particular, we will show how to efficiently find a basis for the space of all $\mathbb{F}$-linear dependencies between polynomials of the form

$$\Big(g_1(x_1) + g_2(x_2) + \ldots + g_n(x_n)\Big)^D,$$

where each g_i is a univariate polynomial and is given explicitly.

3.1 A randomized algorithm for POLYDEP.

Lemma 8. *Given a vector of* m *polynomials* $\mathbf{f} = (f_1(\mathbf{X}), f_2(\mathbf{X}), \ldots, f_m(\mathbf{X}))$ *in which every* f_i *is as usual specified by a circuit, we can compute a basis for the space* $\mathbf{f}^\perp$ *in randomized polynomial time.*

Proof. We will prove this by showing that $\mathbf{f}^\perp$ is actually the nullspace of a small, efficiently computable matrix. Suppose that $\mathbf{f}^\perp$ is spanned by $\mathbf{b}_1, \ldots, \mathbf{b}_t$. Pick m points $\mathbf{a}_1, \ldots, \mathbf{a}_m$ in $\mathbb{F}^n$ and consider the $m \times m$ matrix M defined as follows:

$$M \stackrel{\text{def}}{=} \begin{pmatrix} f_1(\mathbf{a}_1) & f_2(\mathbf{a}_1) & \ldots & f_m(\mathbf{a}_1) \\ f_1(\mathbf{a}_2) & f_2(\mathbf{a}_2) & \ldots & f_m(\mathbf{a}_2) \\ \vdots & \vdots & \ddots & \vdots \\ f_1(\mathbf{a}_m) & f_2(\mathbf{a}_m) & \ldots & f_m(\mathbf{a}_m) \end{pmatrix} \tag{1}$$

[6] The underlying field $\mathbb{F}$ is assumed to be large enough — say at least twice the formal degree of $\mathcal{C}$.

[7] For low-degree arithmetic circuits, i.e. arithmetic circuits whose formal degree is bounded by a polynomial in the size of the circuit, the search version of identity testing is equivalent to the decision version.

[8] An arithmetic is said to be of low degree if the degree of the polynomial computed by it is bounded by a polynomial in the circuit size.

Notice that for each $\mathbf{b}_i$ in the basis of $\mathbf{f}^\perp$, we always have $M \cdot \mathbf{b}_i = 0$ by definition. We now claim that with high probability over a random choice of the points $\mathbf{a}_1, \ldots, \mathbf{a}_m \in \mathbb{F}^n$, the matrix M has rank $(m-t)$. If this happens, then it means that the nullspace of M is exactly the space spanned by $\mathbf{b}_1, \ldots, \mathbf{b}_t$, thereby enabling us to compute a basis of $\mathbf{f}^\perp$ efficiently. Towards this end, it is sufficient to prove the following claim:

Claim 8.1. *Let $P(\mathbf{X}_1, \ldots, \mathbf{X}_m)$ be the $m \times m$ matrix with entries in $\mathbb{F}(\mathbf{X}_1, \ldots, \mathbf{X}_m)$ defined as follows:*

$$P(\mathbf{X}_1, \ldots, \mathbf{X}_m) \stackrel{\text{def}}{=} \begin{pmatrix} f_1(\mathbf{X}_1) & f_2(\mathbf{X}_1) & \ldots & f_m(\mathbf{X}_1) \\ f_1(\mathbf{X}_2) & f_2(\mathbf{X}_2) & \ldots & f_m(\mathbf{X}_2) \\ \vdots & \vdots & \ddots & \vdots \\ f_1(\mathbf{X}_m) & f_2(\mathbf{X}_m) & \ldots & f_m(\mathbf{X}_m) \end{pmatrix}. \tag{2}$$

Then P has rank $(m-t)$.

Proof of Claim 8.1: Without loss of generality we can assume that the polynomials $f_1(\mathbf{X}), f_2(\mathbf{X}), \ldots, f_{m-t}(\mathbf{X})$ are $\mathbb{F}$-linearly independent and the rest of the polynomials are $\mathbb{F}$-linear combinations of these first $(m-t)$ polynomials. It is then sufficient to prove that the submatrix

$$Q \stackrel{\text{def}}{=} \begin{pmatrix} f_1(\mathbf{X}_1) & f_2(\mathbf{X}_1) & \ldots & f_{m-t}(\mathbf{X}_1) \\ f_1(\mathbf{X}_2) & f_2(\mathbf{X}_2) & \ldots & f_{m-t}(\mathbf{X}_2) \\ \vdots & \vdots & \ddots & \vdots \\ f_1(\mathbf{X}_{m-t}) & f_2(\mathbf{X}_{m-t}) & \ldots & f_{m-t}(\mathbf{X}_{m-t}) \end{pmatrix}$$

has full rank, or equivalently, the determinant of Q is a nonzero polynomial. Now, expanding $\textsc{Det}(Q)$ along the first row we have

$$\textsc{Det}(Q) = f_1(\mathbf{X}_1) \cdot Q_{11} - f_2(\mathbf{X}_1) \cdot Q_{12} + \ldots + (-1)^{m-t+1} f_{m-t}(\mathbf{X}_1) \cdot Q_{1m-t},$$

where Q_{ij} is the determinant of the ij-th minor.
Notice that every Q_{1k}, $k \in [m-t]$, is a polynomial in the set of variables $\mathbf{X}_2, \ldots, \mathbf{X}_{m-t}$. By induction, every Q_{1k} is a nonzero polynomial (since every subset of a set of $\mathbb{F}$-linearly independent polynomials is also $\mathbb{F}$-linearly independent). If $\textsc{Det}(Q)$ was the zero polynomial then plugging in random values for $\mathbf{X}_2, \ldots, \mathbf{X}_{m-t}$ would give us a nonzero $\mathbb{F}$-linear dependence among $f_1(\mathbf{X}_1), f_2(\mathbf{X}_1), \ldots, f_{m-t}(\mathbf{X}_1)$, which is a contradiction. Hence $\textsc{Det}(Q)$ must be nonzero, proving the claim. □

This also completes the proof of Lemma 8. □

4 Identity testing

Despite much effort, no deterministic polynomial-time algorithm is known for identity testing. Because of the difficulty of the general problem, research has focussed on special cases. We will present an efficient deterministic algorithm for a special family of polynomials. It is a conceptually simpler and self-contained version of an algorithm of [Sax08]. As observed in the previous section, identity testing is "morally equivalent" to POLYDEP. We will focus our efforts on POLYDEP and devise an efficient deterministic algorithm for POLYDEP$(f_1(\mathbf{X}), \ldots, f_m(\mathbf{X}))$ when each polynomial $f_i(\mathbf{X})$ is a *power of a sum of univariate polynomials*, i.e.,

$$\forall i \in [m] : f_i(\mathbf{X}) = \Big(g_{i1}(x_1) + g_{i2}(x_2) + \ldots + g_{in}(x_n)\Big)^D,$$

and each $g_{ij}(x_j)$ is a univariate polynomial of degree at most d. For a set of polynomials of this form the brute force algorithm that expands every polynomial takes time about n^{d+D}. In this section we will devise a poly$(nmdD)$-time algorithm. We need a few technical claims in order to devise our algorithm. The fairly straightforward proofs of these claims are pushed back to the following subsection.
Our first claim is that knowing a basis for the space of linear dependencies between $(h_1(\mathbf{X}), \ldots, h_m(\mathbf{X}))$, we can efficiently compute a basis for the space of linear dependencies between known linear combinations of the h_i's.

Claim 8.2. *Let $\mathbf{h} = (h_1(\mathbf{X}), \ldots, h_t(\mathbf{X})) \in \mathbb{F}[\mathbf{X}]^t$ be a vector of polynomials. Given a basis for $\mathbf{h}^\perp$ and given vectors $\mathbf{a}_1, \ldots, \mathbf{a}_m \in \mathbb{F}^t$, we can efficiently compute a basis for $(f_1(\mathbf{X}), \ldots, f_m(\mathbf{X}))^\perp$, where*

$$f_i(\mathbf{X}) = \mathbf{a}_i \cdot \mathbf{h} \quad \text{for each } 1 \leq i \leq m.$$

The next claim is a restatement of the fact that a univariate polynomial of degree at most d is zero if and only if the constant term of each one of its derivatives of order upto d is zero.

Claim 8.3. *Let $f(\mathbf{X})$ be a polynomial of degree at most d. Then f is identically zero if and only if $\sigma_1 \partial_1^j f = 0$ for all $0 \leq j \leq d$.*

The next claim uses the above one and shows how to reduce a given instance of the POLYDEP problem into a large number of "smaller" instances of the POLYDEP problem using partial derivatives.

Claim 8.4. [Reducing an instance of POLYDEP in n variables to a number of instances of POLYDEP in $n-1$ variables]
Let $\mathbf{f} = (f_1(\mathbf{X}), \ldots, f_m(\mathbf{X}))$. Suppose that the degree of each $f_i(\mathbf{X})$ is at most d. Suppose that the underlying field $\mathbb{F}$ has characteristic larger than d. Then we have

$$\mathbf{f}^\perp = \bigcap_{j=0}^{d} V_j,$$

where for each $j \in \{0, 1, \ldots, d\}$, V_j is defined to be

$$V_j \stackrel{\text{def}}{=} \left(\sigma_1 \partial_1^j f_1, \ldots, \sigma_1 \partial_1^j f_m\right)^\perp.$$

The observation above suggests a natural strategy for POLYDEP — recursively compute the dependencies among

$$\left(\sigma_1 \partial_1^j f_1, \ldots, \sigma_1 \partial_1^j f_m\right)$$

for $j \in \{0, \ldots, d\}$ and then take the intersection of all the subspaces so obtained. This naïve strategy in general only gives an exponential algorithm that is little better than brute force expansion. However, for the case when f_i's are the powers of sums of univariate polynomials, one can show that all the polynomials in

$$\left\{\sigma_1 \partial_1^j f_k : 0 \leq j \leq d, 1 \leq k \leq m\right\}$$

are efficiently expressible as linear combinations of a small number of polynomials $\{h_1, h_2, \ldots, h_t\} \subseteq \mathbb{F}[x_2, \ldots, x_n]$ where h_i's are also powers of sums of univariate polynomials. Then *using just one recursive call* POLYDEP$(h_1, \ldots, h_t)$ and using the algorithm of claim 8.2, we can compute each V_j for $0 \leq j \leq d$. Thereafter, taking the intersection of the V_j's allows us to compute the linear dependencies between the original polynomials. The algorithm is efficient because in the recursive step we make just one recursive call to POLYDEP rather than $(d+1)$ calls.

Lemma 9. *Let $f(x_1, \ldots, x_n) = (g_1(x_1) + \ldots + g_n(x_n))^D$, where the g_i's are univariate polynomials of degree at most d. Let $G = g_1(x_1) + g_2(x_2) + \ldots + g_n(x_n)$. Then*

$$\sigma_1 \partial_1^j f = \sum_{k=0}^{j} a_{jk} \cdot \sigma_1 G^{D-k}, \quad \text{where each } a_{jk} \in \mathbb{F}.$$

Furthermore, the a_{jk}'s occurring in the above expression can be computed in time poly(dD).

Proof. It suffices to prove that

$$\partial_1^j f = \sum_{k=0}^{j} \sum_{\ell=0}^{(d-1)j} b_{k\ell} \cdot G^{D-k} \cdot x_1^\ell, \tag{3}$$

where the b_{kl}'s are computed efficiently. We prove it by induction on j with the base case of $j = 0$ being trivial. Now assume equation 3 holds then differentiating both sides of equation 3 with respect to x_1, we get an expression for $\partial_1^{j+1} f$. By linearity of derivatives it suffices to examine just one summand which is of the form $\partial_1 G^{D-k} x_1^\ell$. We have

$$\begin{aligned}
\partial_1 G^{D-k} x_1^\ell &= \ell \cdot G^{D-k} x_1^{\ell-1} + (D-k) \cdot G^{D-k-1} x_1^\ell \cdot \partial_1 G \\
&= \ell \cdot G^{D-k} x_1^{\ell-1} + (D-k) \cdot G^{D-k-1} x_1^\ell \cdot \partial_1 (g_1(x_1)) \\
&= \ell \cdot G^{D-k} x_1^{\ell-1} + \sum_{i=0}^{d-1} a_i (D-k) \cdot G^{D-k-1} x_1^{\ell+i},
\end{aligned}$$

where $\partial_1 g_1 = \sum_{i=0}^{d-1} a_i x_1^i$. This completes the proof of the lemma. □

We are now ready to prove our first result on POLYDEP and consequently on identity testing.

Theorem 10. *For $i \in [m]$ and $j \in \{0, \ldots, D\}$, let $f_{ij}(\mathbf{X}) = G_i(\mathbf{X})^j$ where each G_i is a sum of univariate polynomials of degree at most d, i.e., each G_i is of the form*

$$(g_{i1}(x_1) + \ldots + g_{in}(x_n)),$$

the g_{ik}'s being univariate polynomials of degree at most d. There is a deterministic algorithm for

$$POLYDEP\Big(f_{ij} : i \in [m], j \in \{0, \ldots, D\}\Big),$$

whose running time is bounded by poly$(nmdD)$.

Proof. The algorithm is as follows.

Step 1: If $n = 0$, then the f_{ij}'s are all field elements and thus, the computation of their linear dependencies is trivial.

Step 2: If $n \geq 1$, then by making a recursive call to

$$\text{POLYDEP}\Big(\sigma_1(G_i)^j : i \in [m], j \in \{0, \ldots, D\}\Big),$$

we get a basis for

$$\Big(\sigma_1(G_i)^j : i \in [m], j \in \{0, \ldots, D\}\Big)^\perp.$$

Step 3: Use the algorithm of Lemma 9 to compute a_{ijks}'s such that

$$\sigma_1 \partial_1^k G_i^j = \sum_{s=0}^{D} a_{ijks} \cdot \sigma_1(G_i)^s.$$

Step 4: From the data above and using the algorithm of claim 8.2, compute a basis for

$$V_k \stackrel{\text{def}}{=} \Big\{\sigma_1 \partial_1^k G_i^j : i \in [m], j \in \{0, \ldots, D\}\Big\}^\perp.$$

Step 5: Output

$$\bigcap_{k=0}^{dD} V_k.$$

The correctness follows from claim 8.4. If the time taken is denoted by $T(n, m, d, D)$ then the recurrence relation is

$$T(n, m, d, D) = T(n-1, m, d, D) + \text{poly}(mdD)$$

which solves out to give $T(n, m, d, D) = \text{poly}(nmdD)$. □

4.1 Proofs of the technical claims

In this subsection we provide the proofs of some of the technical claims.

Claim 8.2. Let $\mathbf{h} = (h_1(\mathbf{X}), \ldots, h_t(\mathbf{X})) \in \mathbb{F}[\mathbf{X}]^t$ be a vector of polynomials. Given a basis for $\mathbf{h}^\perp$ and given vectors $\mathbf{a}_1, \ldots, \mathbf{a}_m \in \mathbb{F}^t$, we can efficiently compute a basis for $(f_1(\mathbf{X}), \ldots, f_m(\mathbf{X}))^\perp$, where

$$f_i(\mathbf{X}) = \mathbf{a}_i \cdot \mathbf{h} \quad \text{for each } 1 \leq i \leq m.$$

Proof. Let $r = t - \dim((h_1(\mathbf{X}), \ldots, h_t(\mathbf{X}))^\perp)$. r is the dimension of the $\mathbb{F}$-space generated by $\{h_1, \ldots, h_t\}$. Using the basis for $\mathbf{h}^\perp$ and the Gaussian elimination algorithm, we can find a linearly independent set of r h_i's. Without loss of generality, let us assume that $h_1, \ldots, h_r$ are $\mathbb{F}$-linearly independent. Now express each $f_j(\mathbf{X})$ as a linear combination of $h_1(\mathbf{X}), \ldots, h_r(\mathbf{X})$. In this way, for the purpose of the rest of the algorithm, we can treat the f_j's simply as r-dimensional vectors over $\mathbb{F}$. Finally, we iterate over i and compute two sets $F_i \subseteq \{f_1(\mathbf{X}), \ldots, f_i(\mathbf{X})\}$ and $V_i \subset \mathbb{F}^t$. F_i consitutes a maximal set of $\mathbb{F}$- linearly independent vectors among $\{f_1(\mathbf{X}), \ldots, f_i(\mathbf{X})\}$ whereas V_i will be a basis of the linear dependencies involving $f_1(\mathbf{X}), f_2(\mathbf{X}), \ldots, f_i(\mathbf{X})$. Assume that we have F_i and V_i and we wish to compute F_{i+1}, V_{i+1}.

- **Case 1.** $f_{i+1}(\mathbf{X}) \notin \text{Span}(f_1(\mathbf{X}), \ldots, f_i(\mathbf{X}))$. Then

$$F_{i+1} \stackrel{\text{def}}{=} F_i \uplus \{f_{i+1}\} \quad \text{and} \quad V_{i+1} \stackrel{\text{def}}{=} V_i.$$

- **Case 2.** $f_{i+1}(\mathbf{X}) \in \text{Span}(f_1(\mathbf{X}), \ldots, f_i(\mathbf{X}))$. Let

$$f_{i+1}(\mathbf{X}) = \alpha_1 f_1(\mathbf{X}) + \alpha_2 f_2(\mathbf{X}) + \ldots + \alpha_i f_i(\mathbf{X})$$

  Then

$$F_{i+1} \stackrel{\text{def}}{=} F_i \quad \text{and} \quad V_{i+1} \stackrel{\text{def}}{=} V_i \uplus (\alpha_1, \alpha_2, \ldots, \alpha_i, -1, 0, \ldots 0).$$

After m steps of the the iterative process, we output V_m. Clearly, this is an efficient algorithm. □

Claim 8.4. [Reducing an instance of POLYDEP in n variables to a number of instances of POLYDEP in $n-1$ variables]
Let $\mathbf{f} = (f_1(\mathbf{X}), \ldots, f_m(\mathbf{X}))$. Suppose that the degree of each $f_i(\mathbf{X})$ is at most d. Suppose that the underlying field $\mathbb{F}$ has characteristic larger than d. Then we have

$$\mathbf{f}^\perp = \bigcap_{j=0}^{d} V_j,$$

where for each $j \in \{0, 1, \ldots, d\}$, V_j is defined to be

$$V_j \stackrel{\text{def}}{=} \left(\sigma_1 \partial_1^j f_1, \ldots, \sigma_1 \partial_1^j f_m\right)^\perp.$$

Proof. Let $g(\mathbf{X}) = \sum_{i \in [m]} \alpha_i f_i(\mathbf{X})$. Viewing $g(\mathbf{X})$ as a univariate polynomial in x_1 and using claim 8.3, we get that $g(\mathbf{X})$ is zero if and only if $\sigma_1 \partial_1^j g = 0$ for all $j \in \{0, 1, \ldots, d\}$. By the linearity of ∂_1 and of σ_1, we get that

$$\sum_{i \in [m]} \alpha_i \cdot \sigma_1 \partial_1^j f_i = 0$$

for all $j \in \{0, \ldots, d\}$. Thus $(\alpha_1, \ldots, \alpha_m) \in V_j$ for all $j \in \{0, \ldots, d\}$. □

5 The complexity of DegSLP

In this section we consider the algorithmic complexity of the following problem: given a polynomial $f(\mathbf{X})$ as an arithmetic circuit, and an integer d in binary determine if $\deg(f) \leq d$. Towards this end, we need to define another computational problem which is interesting in its own right.

> **CoeffSLP** : given a polynomial $f(\mathbf{X})$ over a finite field $\mathbb{F}$ and a monomial $\mathbf{X}^{\alpha}$, determine the coefficient of $\mathbf{X}^{\alpha}$ in $f(\mathbf{X})$.

We will need the following theorem from [KP07]. A simpler, self-contained proof is given in the appendix.

Theorem 11. *[KP07] CoeffSLP is #P-complete.*

We will also need a lemma originally due to Edouard Lucas.

Lemma 12. *[vL99, p.55] Let n, m be positive inte gers whose p-ary representation is the following:*

$$m = m_0 + m_1 p + \ldots + m_d p^d, \quad \forall i : 0 \leq m_i \leq p-1$$

$$n = n_0 + n_1 p + \ldots + n_d p^d \quad \forall i : 0 \leq n_i \leq p-1.$$

Then

$$\binom{n}{m} = \binom{n_0}{m_0} \cdot \binom{n_1}{m_1} \cdot \ldots \cdot \binom{n_d}{m_d} \pmod{p}$$

In particular, for any intger $n \geq 1$, the binomial coefficient $\binom{n}{p^i}$ is divisible by p if and only if the n_i, the i-th digit in the p-ary representation of n is zero.

Lemma 13. *Over a field of characteristic larger than d, the coefficient of x^d in $f(x+\beta)$ is precisely $\frac{1}{d!} f_d(\beta)$.*

Proof. By the linearity of derivatives, it is sufficient to show this for monomials. So let $f(x) = a \cdot x^e$. If $e < d$ then $f_d(x)$ is the zero polynomial and we are done. So let $e \leq d$. Expanding $(x+\beta)^e$ using binomial theorem we get that coefficient of x^d is $a \cdot \binom{e}{d} \cdot \beta^{e-d}$. On the other hand $f_d(x) = a \cdot e \cdot (e-1) \cdot \ldots \cdot (e-d+1) x^{e-d}$. It is now easily verified that the coefficient of x^d in $f(x+\beta)$ is $\frac{1}{d!} f_d(\beta)$. □

Theorem 14.

$$\textit{DegSLP} \leq_T^{coRP} \textit{CoeffSLP} .$$

Proof. We first reduce the multivariate to the univariate problem by making a random substitution of the form $g(z) = f(a_1 \cdot z, a_2 \cdot z, \ldots, a_n \cdot z)$.

Claim 14.1. *With high probability over a random choice of the vector $\mathbf{a} = (a_1, \ldots, a_n)$ we have:* $\deg(g(z)) = \deg(f(\mathbf{X}))$.

Proof of Claim 14.1: Let f have degree d. We can write the polynomial $f(\mathbf{X})$ as $\sum_{i=0}^{d} f_i(\mathbf{X})$, where each $f_i(\mathbf{X})$ is a homogeneous polynomial of degree i. Applying the substitution $x_i := a_i \cdot z$, we get that

$$g(z) = f_0 + z \cdot f_1(\mathbf{a}) + z^2 \cdot f_2(\mathbf{a}) + \ldots + z^d \cdot f_d(\mathbf{a}).$$

By the Schwartz-Zippel lemma, $f_d(\mathbf{a})$ is nonzero with high probability so that $\deg(g(z)) = \deg(f(\mathbf{X}))$ also with high probability. □

We will first show the reduction to CoeffSLP for fields which have characteristic zero or at least characteristic larger than the formal degree of the polynomial that is computed. The reduction in smaller characteristic is a little more involved. Our problem is the following: given an arithmetic circuit computing a univariate polynomial $g(z)$ and an integer d in binary, we want to determine if $\deg(g(z)) \geq d$. Let $g_d(z)$ denote the d-th derivative of $g(z)$. Over fields of appropriately large characteristic, $\deg(g(z)) \geq d$ if and only if $g_d(z)$ is not the identically zero polynomial. By lemma 13, the coefficient of z^d in $g(z+\beta)$ is $\frac{1}{d!} g_d(\beta)$. Thus, with high probability over a random choice of β, the coefficient of z^d in $g(z+\beta)$ is nonzero if and only if $\deg(g(z)) \geq d$. This we can determine by making an oracle call to CoeffSLP .

To get the reduction over fields of small characteristic, we need to examine the polynomial $g(z+y)$ and determine as to when does it happen that the coefficient of z^d as a polynomial in y is the identically zero polynomial. We sketch the proof below. Let the size of the circuit computing f be s. Then the formal degree of f is bounded by 2^s. First observe that multiplying $g(z)$ with a suitable power of z, we may assume without loss of generality that d is a power of p, say $d = p^t$. Notice that $\deg(g(z)) \geq p^t$ if and only if $g(z)$ contains a monomial z^m where the p-ary (base-p) representation of the positive integer m contains a non-zero digit at the i-th position, for some $t \leq i \leq s$. Thus, to achieve our objective, it is sufficient to devise a randomized procedure that given an integer $i \in [s]$, tests whether $g(z)$ contains any non-zero monomial z^m such that in the p-ary representation of the integer m, the i-th digit is non-zero. This procedure works as before: choose a random β (in a suitably large field extension of $\mathbb{F}_p$) and accept if and only if the the coefficient of z^{p^i} (computed via an oracle call to CoeffSLP) is nonzero. We next describe why the test gives the correct answer with high probability. Suppose that

$$g(z) = \sum_{0 \leq m \leq 2^s} a_m \cdot z^m.$$

Then the coefficient of z^{p^i} in $g(z+\beta)$ is given by

$$h(\beta) = \sum_{p^i \leq m \leq 2^s} a_m \cdot \binom{m}{p^i} \cdot \beta^{m-p^i}.$$

Our test accepts with high probability if and only if $h(\beta)$ is not the identically zero polynomial with respect to β. Use Lucas's lemma 12 to observe that $\binom{m}{p^i}$ is zero modulo p if and only if the i-th digit in the p-ary representation of m is zero. Thus $h(\beta)$ is not the zero polynomial if and only if there exists an $p^i \leq m \leq 2^s$, such that a_m is nonzero and in the p-ary representation of the integer m, the i-th digit is nonzero. Thus $h(\beta)$ is nonzero if and only if $g(z)$ contains a non-zero monomial z^m such that in the p-ary representation of the integer m, the i-th digit is non-zero. This completes the proof of the theorem. □

Combining theorems 11 and 14, we immediately get:

Theorem 15. *DegSLP is in coRP^{PP}.*

6 Minimizing Variables

From now on we will only consider fields of characteristic zero. All the results stated here will also hold true for fields of characteristic larger than the degrees of the relevant polynomials. In this section we study how to elimiate redundant variables from a polynomial.

Definition 16. *Let $f(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$ be a polynomial. We will say that $f(\mathbf{X})$ is independent of a variable x_i if no monomial of $f(\mathbf{X})$ contains x_i. We will say that* the number of essential variables *in $f(\mathbf{X})$ is t if we can make an invertible linear $A \in \mathbb{F}^{(n\times n)*}$ transformation on the variables such that $f(A \cdot \mathbf{X})$ depends on only t variables $x_1, \ldots, x_t$. The remaining $(n-t)$ variables $x_{t+1}, \ldots, x_n$ are said to be* redundant *variables. We will say that $f(\mathbf{X})$ is a* regular *if it has no redundant variables.*

Example 17. *The number of essential variables in the quadratic polynomial $f(x_1, x_2, x_3) = x_1^2 + 2x_1x_2 + x_2^2 + x_3^2$ is just two because notice that $f = (x_1 + x_2)^2 + x_3^2$ and thus after making the invertible linear transformation*

$$A \quad : \quad \begin{array}{l} x_1 + x_2 \mapsto x_1 \\ x_3 \mapsto x_2 \\ x_2 \mapsto x_3 \end{array}$$

we get that $f(A \cdot \mathbf{X}) = x_1^2 + x_2^2$ is just a function of two variables x_1 and x_2.

The vanishing of partials.

We now reprove a lemma due to Carlini [Car06]. Let us examine the situation when a variable is redundant. Let $g(\mathbf{X}) = f(A \cdot \mathbf{X})$ where A is an $n \times n$ invertible matrix. If g does not depend upon x_i then

$$\frac{\partial g}{\partial x_i} = 0 \quad \Leftrightarrow \quad \sum_{k=1}^{n} a_{ki} \cdot \frac{\partial f}{\partial x_k}(A \cdot \mathbf{X}) = 0$$

Thus there exists a vector $\mathbf{a} \in \mathbb{F}^n$ such that $\mathbf{a} \cdot \partial(f) = 0$, where $\partial(f) \stackrel{\text{def}}{=} (\partial_1 f, \partial_2 f, \ldots, \partial_n f)$. Using the notation of section 3, this can be written succintly as

$$\mathbf{a} \in \partial(f)^{\perp}$$

Suppose that $\mathbf{b}_1, \ldots, \mathbf{b}_t \in \mathbb{F}^n$ is a basis of the space $\partial(f)^{\perp}$. Now there exists $n - t$ independent vectors $\mathbf{a}_1, \ldots, \mathbf{a}_{n-t}$ such that the vector space $\mathbb{F}^n$ is spanned by $\mathbf{a}_1, \ldots, \mathbf{a}_{n-t}, \mathbf{b}_1, \ldots, \mathbf{b}_t$. Consider the invertible matrix whose columns are $\mathbf{a}_1, \ldots, \mathbf{a}_{n-t}, \mathbf{b}_1, \ldots, \mathbf{b}_t$ respectively. Let $g(\mathbf{X}) \stackrel{\text{def}}{=} f(A \cdot \mathbf{X})$. Then for $n - t + 1 \leq i \leq n$,

$$\begin{aligned} \frac{\partial g}{\partial x_i} &= \mathbf{b}_{i-n+t} \cdot \partial(f)(A \cdot \mathbf{X}) \\ &= 0 \quad (\text{since } \mathbf{b}_{i-n+t} \cdot \partial(f)(\mathbf{X}) = 0) \end{aligned}$$

We thus have:

Lemma 18. *(Carlini [Car06]) The number of redundant variables in a polynomial $f(\mathbf{X})$ equals the dimension of $\partial(f)^{\perp}$. Furthermore, given a basis of $\partial(f)^{\perp}$, we can easily come up with a linear transformation A on the variables such that the polynomial $f(A \cdot \mathbf{X})$ depends on only the first $(n - \dim(\partial(f)^{\perp}))$ variables.*

Notice that arithmetic circuits for each polynomial in $\partial(f)$ can be easily computed in poly($|f|$) time, where $|f|$ is the size of the circuit for f. This computation can be made even more efficient using the algorithm of Baur and Strassen [BS83]. Thereafter, a basis for the space $\partial(f)^{\perp}$ can be efficiently computed using lemma 8. In this way we have:

Theorem 19. *Given a polynomial $f(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$ with m essential variables, we can compute in randomized polynomial time an invertible linear transformation $A \in \mathbb{F}^{(n \times n)*}$ such that $f(A \cdot \mathbf{X})$ depends on the first m variables only.*

7 Equivalence to sums of d-th powers

Consider the following problem: given a homogeneous polynomial $f(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$ of degree d, does there exist a linear transformation $A \in \mathbb{F}^{n \times n}$ and constants $a_1, \ldots, a_n \in \mathbb{F}$ such that

$$f(A \cdot \mathbf{X}) = a_1 \cdot x_1^d + a_2 \cdot x_2^d + \ldots + a_n \cdot x_n^d.$$

Equivalently, the problem can be restated as follows: given a homogeneous polynomial $f(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$ of degree d, determine n independent linear forms $\ell_1, \ldots, \ell_n \in \mathbb{F}[\mathbf{X}]$ and constants $a_1, \ldots, a_n \in \mathbb{F}$ such that

$$f(\mathbf{X}) = a_1 \cdot \ell_1(\mathbf{X})^d + \ldots + a_n \cdot \ell_n(\mathbf{X})^d.$$

We will devise a randomized polynomial-time algorithm that given $f(\mathbf{X})$, computes the constants and the set of linear forms $\ell_1(\mathbf{X}), \ldots, \ell_n(\mathbf{X})$. The key idea involved in this is the Hessian matrix.

Definition 20. *For a polynomial $f(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$, the Hessian Matrix $H_f(\mathbf{X}) \in (F[\mathbf{X}])^{n \times n}$ is defined as follows.*

$$H_f(\mathbf{X}) \stackrel{\text{def}}{=} \begin{bmatrix} \frac{\partial^2 f}{\partial x_1 \cdot \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_1 \cdot \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \cdot \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n \cdot \partial x_n} \end{bmatrix}$$

The most interesting property of the hessian matrix of a polynomial is the effect that a linear transformation of the variables has on it.

Lemma 21. *Let $f(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$ be an n-variate polynomial and $A \in \mathbb{F}^{n\times n}$ be a linear transformation. Let $F(\mathbf{X}) \stackrel{\text{def}}{=} f(A \cdot \mathbf{X})$. Then,*

$$H_F(\mathbf{X}) = A^T \cdot H_f(A \cdot \mathbf{X}) \cdot A.$$

In particular,

$$\textsc{Det}(H_F(\mathbf{X})) = \textsc{Det}(A)^2 \cdot \textsc{Det}(H_f(A \cdot \mathbf{X}))$$

Proof. By the chain for differentiation we have for all $1 \leq i \leq n$:

$$\frac{\partial F}{\partial x_i} = \sum_{k=1}^{n} a_{ki} \cdot \frac{\partial f}{\partial x_k}(A \cdot \mathbf{X})$$

Therefore for all $1 \leq i, j \leq n$:

$$\begin{aligned} \frac{\partial^2 F}{\partial x_i \cdot \partial x_j} &= \sum_{k=1}^{n} a_{ki} \cdot (\sum_{\ell=1}^{n} a_{\ell j} \frac{\partial^2 f}{\partial x_k \cdot \partial x_l}(A \cdot \mathbf{X})) \\ &= \sum_{k\in[n],\ell\in[n]} a_{ki} \cdot \frac{\partial^2 f}{\partial x_k \cdot \partial x_l}(A \cdot \mathbf{X}) \cdot a_{\ell j} \end{aligned}$$

Putting these equations into matrix form immediate gives us the lemma. □

Now consider a homogeneous polynomial $f(\mathbf{X})$ of degree $d \geq 3$ which has the property that there exists a linear transformation A of the variables such that

$$f(A \cdot \mathbf{X}) = x_1^d + x_2^d + \ldots + x_n^d.$$

Set $F(\mathbf{X}) \stackrel{\text{def}}{=} x_1^d + x_2^d + \ldots + x_n^d$. Observe that

$$\frac{\partial^2 F}{\partial x_i \cdot \partial x_j} = \begin{cases} 0 & \text{if } i \neq j, \\ d(d-1)x_i^{d-2} & \text{if } i = j. \end{cases}$$

Thus the matrix $H_F(\mathbf{X})$ is a diagonal matrix so that we have

$$\textsc{Det}(H_F(\mathbf{X})) = d(d-1) \cdot \prod_{i=1}^{n} x_i^{d-2}.$$

By the lemma 21 above we get that

$$\textsc{Det}(H_f(\mathbf{X})) = d(d-1) \cdot \textsc{Det}(A)^{-2} \cdot \prod_{i=1}^{n} \ell_i(\mathbf{X})^{d-2},$$

where the $\ell_i(\mathbf{X})$'s are linear forms corresponding to the different rows of the matrix A^{-1}. Let us record this as a lemma.

Lemma 22. *For a polynomial $f(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$ of degree d, if*

$$f(\mathbf{X}) = \sum_{i=1}^{n} a_i \cdot \ell_i(\mathbf{X})^d,$$

where $\ell_1(\mathbf{X}), \ldots \ell_n(\mathbf{X})$ are independent linear forms then

$$\textsc{Det}(H_f(\mathbf{X})) = c \cdot \prod_{i=1}^{n} \ell_i(\mathbf{X})^{d-2},$$

where $c \in \mathbb{F}$ is a nonzero constant.

Using lemma 22 and applying unique factorization of polynomials to $\textsc{Det}(H_f(\mathbf{X}))$, we immediately get the following corollary.

Corollary 23. *If $\sum_{i\in[n]} x_i^3 = \sum_{i\in[n]} \ell_i(\mathbf{X})^3$, where the ℓ_i's are independent linear forms then there exists a permutation $\pi \in S_n$ such that $\ell_i = \omega^j \cdot x_{\pi(i)}$, where ω is a primitive third root of unity.*

Lemma 22 can be used to devise a randomized polynomial-time algorithm for our problem as follows.

Input. An n-variate polynomial $f(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$ of degree d.

Output. A set of independent linear forms $\ell_1(\mathbf{X}), \ldots, \ell_n(\mathbf{X})$ and constants $a_1, \ldots, a_n$ such that

$$f(\mathbf{X}) = a_1 \cdot \ell_1(\mathbf{X})^d + \ldots + a_n \cdot \ell_n(\mathbf{X})^d,$$

if such a set of ℓ_i's exist.

The Algorithm.

1. Compute an arithmetic circuit $C(\mathbf{X})$ which computes $\textsc{Det}(H_f(\mathbf{X}))$.
2. Use Kaltofen's factorization algorithm [Kal89] to factor $C(\mathbf{X})$ in random polynomial time. If it is not the case that

$$C(\mathbf{X}) = \prod_{i=1}^{n} \ell_i(\mathbf{X})^{d-2},$$

where each $\ell_i(\mathbf{X})$ is a linear form then output No such forms. else (by solving a system of linear equations) compute constants $a_1, \ldots, a_n$ such that

$$f(\mathbf{X}) = \sum_{i=1}^{n} a_i \cdot \ell_i(\mathbf{X})^d.$$

Output $(\ell_1(\mathbf{X}), \ldots, \ell_n(\mathbf{X})), (a_1, \ldots, a_n)$.

This completes the description of the decomposition algorithm for the special case of sum of powers.

8 Equivalence to an elementary symmetric polynomial

The problem that we now tackle is the following — given an arithmetic circuit which computes an n-variate homogeneous polynomial $f(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$, is there an invertible linear transformation A such that $f(A \cdot \mathbf{X})$ is the elementary symmetric polynomial of degree d? Recall that the elementary symmetric polynomial of degree d [9] is

$$\mathrm{SYM}_n^d \stackrel{\text{def}}{=} \sum_{S\subseteq[n],|S|=d} \prod_{i\in S} x_i.$$

Observe that SYM_n^d is a multilinear polynomial and therefore we have

$$\partial_i^2 \mathrm{SYM}_n^d = 0, \quad \text{for all } i \in [n]. \tag{4}$$

More interestingly, these are essentially the only second-order partial derivatives of SYM_n^d which vanish. The following lemma shows that most of the these partial derivatives are linearly independent.

Lemma 24. *For $d \geq 4$, we have*

$$\dim\left(\partial^2(SYM_n^d)\right) = \binom{n}{2}.$$

[9] SYM_n^d is the unique (upto scalar multiples) homogeneous *multilinear* polynomial of degree d in n variables, which is invariant under every permutation of the variables.

Proof. See [KN97, pp.22–23]. □

This means that if f is equivalent to SYM_n^d then $\partial^2(f)$ has dimension $\binom{n}{2}$. Indeed our method shows that for any polynomial $f \in \mathbb{F}(\mathbf{X})$ which has the property that $\partial^2(f)$ has dimension $\binom{n}{2}$, we can efficiently determine whether f is equivalent to a multilinear polynomial and if so, find an invertible matrix A such that $f(A \cdot \mathbf{X})$ is multilinear. Now let

$$g(\mathbf{X}) \stackrel{\text{def}}{=} f(A \cdot \mathbf{X})$$

be multilinear. It will also follow from our proof that this multilinear polynomial $g(\mathbf{X})$ is equivalent to an elementary symmetric polynomial if and only if there is a diagonal matrix B such that

$$g(B \cdot \mathbf{X}) = \mathrm{SYM}_n^d.$$

It is then a relatively easy exercise to determine whether such a diagonal matrix B exists or not.

Exercise 25. *Show that given a multilinear polynomial $g(\mathbf{X})$, one can efficiently determine whether there exist $\lambda_1, \ldots, \lambda_n \in \mathbb{F}$ such that*

$$g(\lambda_1 x_1, \ldots, \lambda_n x_n) = SYM_n^d(\mathbf{X})$$

In the rest of this section, we will assume that $f(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$ is a polynomial that satisfies $\dim(\partial^2(f)) = \binom{n}{2}$. We will tackle the problem of finding an invertible matrix A such that $f(A \cdot \mathbf{X})$ is multilinear, if such an A exists. We will first observe that our problem boils down to finding an "nicer" basis for a given space of matrices. By a "nicer" basis, we will mean a basis consisting of rank one matrices. We then devise an efficient randomized algorithm for the latter problem.

8.1 Reduction to finding a good basis for a space of matrices.

We first consider linear transformations of the variables of a polynomial which make the polynomial multilinear. Let

$$g(\mathbf{X}) = f(A \cdot \mathbf{X}) = f\left(\sum_j a_{1j}x_j, \sum_j a_{2j}x_j, \ldots, \sum_j a_{nj}x_j\right).$$

be the polynomial obtained by applying the transformation A to the variables in f. Then $\partial_i^2 g = 0$ if and only if

$$(a_{1i}\partial_1 + a_{2i}\partial_2 + \ldots + a_{ni}\partial_n)^2 f = 0.$$

Therefore, if $g(\mathbf{X})$ is multilinear then every column vector of A satisfies

$$(a_1\partial_1 + a_2\partial_2 + \ldots + a_n\partial_n)^2 f = 0,$$

and these n vectors are linearly independent since A is invertible.

We will apply the above observation algorithmically as follows. Given f, we first compute the set $\partial^2 f \stackrel{\text{def}}{=} \{\frac{\partial^2 f}{\partial i \cdot \partial j} : i \neq j\}$ and then using the randomized algorithm for POLYDEP, we obtain a basis for the set of all quadratic differential operators $D(\partial_1, \ldots, \partial_n)$ such that $Df = 0$. Since $\dim(\partial^2(f)) = \binom{n}{2}$ we have $\dim(D(\partial_1, \ldots, \partial_n)) = n$. By the observation above our problem boils down to finding a basis for $D(\partial_1, \ldots, \partial_n)$ such that every quadratic operator in the basis has the following form:

$$(a_1\partial_1 + a_2\partial_2 + \ldots + a_n\partial_n)^2 f = 0.$$

Towards this end, we associate every n-variate quadratic operator D with an $n \times n$ symmetric matrix $\hat{D}$ in the following natural way. Let $D \in \mathbb{F}[\partial_1, \ldots, \partial_n]$ be a quadratic polynomial, where

$$D = \sum_{i \in [n]} \alpha_i \partial_i^2 + \sum_{1 \leq i < j \leq n} \beta_{ij} \partial_i \partial_j.$$

The matrix $\hat{D}$ associated with this operator D is the following:

$$\hat{D} \stackrel{\text{def}}{=} \begin{pmatrix} \alpha_1 & \frac{1}{2}\beta_{12} & \dots & \frac{1}{2}\beta_{1n} \\ \frac{1}{2}\beta_{12} & \alpha_2 & \dots & \frac{1}{2}\beta_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{2}\beta_{1n} & \frac{1}{2}\beta_{2n} & \dots & \alpha_n \end{pmatrix}. \tag{5}$$

This way of associating a quadratic differential operator with a symmetric matrix has the following property.

Property 26. *Over an algebraically closed field $\mathbb{F}$ of characteristic different from 2, the quadratic polynomial D is equivalent to a sum of r squares if and only if the corresponding symmetric matrix $\hat{D}$ is of rank r. In particular, the polynomial D is a perfect square if and only if $\hat{D}$ is of rank one.*

Using this property, our problem is equivalent to finding a basis of a given space of symmetric matrices consisting of rank one symmetric matrices in the following way.

1. Given an arithmetic circuit of size s for the polynomial $f(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$, we use the naive method of computing derivatives to obtain a new circuit of size $O(sn^2)$, whose outputs are the second-order partial derivatives $\partial^2(f)$ of f.

2. Using the randomized algorithm for POLYDEP, we obtain a basis for $(\partial^2(f))^{\perp}$. Each element in the basis of $(\partial^2(f))^{\perp}$ is a homogeneous quadratic polynomial in $\mathbb{F}[\partial_1, \dots, \partial_n]$ in the natural way. Let this basis be

$$\{D_1, \dots, D_n\} \subset \mathbb{F}[\partial_1, \dots, \partial_n].$$

3. From $D_1, \dots, D_n$, we get the corresponding symmetric matrices $\hat{D}_1, \dots, \hat{D}_n$. Using the randomized algorithm given below, we obtain another basis $\{\hat{E}_1, \dots, \hat{E}_n\}$ of the vector space generated by $\{\hat{D}_1, \dots, \hat{D}_n\}$ such that each $\hat{E}_i$ is a rank one symmetric matrix [10], if such a basis exists.

   Their corresponding quadratic polynomials $E_1, \dots, E_n \subset \mathbb{F}[\partial_1, \dots, \partial_n]$ are then perfect squares. Let

$$E_i = \left(\sum_{j \in [n]} a_{ij} \partial_j \right)^2.$$

   The matrix $A = (a_{ij})_{i,j \in [n]}$ is then the required linear transformation which makes f multilinear.

We now present an efficient randomized algorithm that given n linearly independent matrices of dimension $n \times n$, finds a basis consisting of rank-one matrices, if such a basis exists. Our proof will also show that such a basis, if it exists, is unique up to scalar multiples and permutations of the basis elements.

8.2 Randomized algorithm for finding a basis consisting of rank-one matrices.

We are given n symmetric matrices $\hat{D}_1, \dots, \hat{D}_n$, and we want to find another basis $\hat{E}_1, \dots, \hat{E}_n$ of the space generated by the given matrices such that each $\hat{E}_i$ is of rank one. A rank one symmetric matrix is the outer product of a vector with itself. So for each $i \in [n]$, let $\hat{E}_i = \mathbf{v}_i^T \mathbf{v}_i$ where $\mathbf{v}_i \in \mathbb{F}^n$.

Lemma 27. *Suppose that $\mathbf{v}_1, \dots, \mathbf{v}_n \in \mathbb{F}^n$ are vectors. Then*

$$\textsc{Det}(z_1 \mathbf{v}_1^T \cdot \mathbf{v}_1 + \dots + z_n \mathbf{v}_n^T \cdot \mathbf{v}_n) = z_1 z_2 \dots z_n \cdot (\textsc{Det}(V))^2, \tag{6}$$

where $V = [\mathbf{v}_1^T \dots \mathbf{v}_n^T]$ is the matrix whose columns are the $\mathbf{v}_i$'s.

[10] Here we are thinking of matrices as n^2-dimensional vectors

Proof. Let $M(\mathbf{z}) \stackrel{\text{def}}{=} z_1\mathbf{v}_1^T \cdot \mathbf{v}_1 + \ldots + z_n\mathbf{v}_n^T \cdot \mathbf{v}_n$. Then $\textsc{Det}(M(\mathbf{z}))$ is a polynomial of degree n in the formal variables $z_1, \ldots, z_n$. If $z_i = 0$ then for every setting of the remaining variables, the matrix M is singular because its image is spanned by the vectors $\mathbf{v}_1, \ldots, \mathbf{v}_{i-1}, \mathbf{v}_{i+1}, \ldots, \mathbf{v}_n$, and is of rank at most $n-1$. Thus z_i divides $\textsc{Det}(M(\mathbf{z}))$ for all $i \in [n]$. Using Chinese remaindering, we have that $\prod z_i$ divides $\textsc{Det}(M(\mathbf{z}))$. Because the degree of $\textsc{Det}(M(\mathbf{z}))$ is n, we have

$$\textsc{Det}(M(\mathbf{z})) = \lambda \prod_{i \in [n]} z_i,$$

for some scalar $\lambda \in \mathbb{F}$. Setting all the z_i's to 1, we get

$$\lambda = \textsc{Det}\left(\sum_{i \in [n]} \mathbf{v}_i^T \cdot \mathbf{v}_i\right) = \textsc{Det}(V \cdot V^T) = \textsc{Det}(V)^2.$$

We thus have $\textsc{Det}(M(\mathbf{z})) = z_1 z_2 \ldots z_n \cdot (\textsc{Det}(V))^2$. □

Corollary 28. *Let $\hat{D}_1, \ldots, \hat{D}_n \in \mathbb{F}^{n \times n}$ be symmetric matrices. Suppose that there exist vectors $\mathbf{v}_1, \ldots \mathbf{v}_n$ such that*

$$\hat{D}_i = \sum_{j=1}^{n} \alpha_{ij} \mathbf{v}_j^T \cdot \mathbf{v}_j. \tag{7}$$

Then

$$\textsc{Det}(z_1\hat{D}_1 + \ldots z_n\hat{D}_n) = \textit{constant} \cdot \ell_1\ell_2 \ldots \ell_n,$$

where for all $j \in [n]$, $\ell_j = \sum_{i=1}^{n} \alpha_{ij} z_i$ is a linear form over $z_1, \ldots, z_n$.

Corollary 28 suggests an algorithm.

Theorem 29. *There exists a randomized polynomial-time algorithm that given n symmetric matrices $\hat{D}_1, \ldots, \hat{D}_n \in \mathbb{F}^{n \times n}$, finds a basis for the space generated by them consisting of matrices of rank one, if such a basis exists.*

Proof. We write down an arithmetic circuit for the polynomial

$$F(z_1, \ldots, z_n) \stackrel{\text{def}}{=} \textsc{Det}(z_1\hat{D}_1 + \ldots + z_n\hat{D}_n).$$

Then we use Kaltofen's algorithm [Kal89] to factor $F(z_1, z_2, \ldots, z_n)$ in randomized polynomial time. By Corollary 28, we can use the linear factors $\ell_1, \ell_2, \ldots, \ell_n$ of this polynomial, which are unique up to scalar multiples and permutations, to solve the equations (7), and get the rank one matrices as required. □

This completes the description of our algorithm.

9 Generalizations of equivalence testing.

The algorithm presented in the previous section generalizes and we obtain:

Theorem 30. Multilinearization. *Given a polynomial $f \in \mathbb{F}(\mathbf{X})$ which has the property that $\partial^2(f)$ has dimension $\binom{n}{2}$, we can efficiently determine whether f is equivalent to a multilinear polynomial and if so, find an invertible matrix A such that $f(A \cdot \mathbf{X})$ is multilinear.* [11]

The algorithm for testing equivalence to sum of powers of linear forms also generalizes although certain degenerate cases need to be ruled out.

Theorem 31. Polynomial Decomposition *There is a randomized polynomial-time algorithm that given an n-variate polynomial $f(\mathbf{X})$ as an arithmetic circuit, finds a decomposition of $f(\mathbf{X})$, if it exists, provided $\textsc{Det}(H_f(\mathbf{X}))$ is a regular polynomial, i.e. it has n variables upto equivalence.*

We do not know whether there exists such an efficient polynomial decomposition algorithm for *all* polynomials. We postpone this generalization to the appendix.

[11] Notice that if f is equivalent to a multilinear polynomial then $\partial^2(f)$ can have dimension at most $\binom{n}{2}$.

10 Discussion and open problems

We feel that the complexity of the problem DegSLP is one of the most intriquing questions in the area of computational algebra. Our work on this problem was motivated by the work of Allender et al [AKPBM06] combined with the observation that this problem is to polynomials what the problem PosSLP is to integers. As shown in [AKPBM06], the latter problem captures a huge chunk of the field of numerical analysis and is therefore unlikely to admit efficient algorithms. We make a corresponding conjecture for DegSLP :

Conjecture. There is no efficient randomized algorithm for DegSLP unless the polynomial hierarchy collapses.

The problem DegSLP has a natural generalization to low-degree multivariate polynomials: given a polynomial $f(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$, determine its leading coefficient under say the natural lexicographic ordering of monomials. We do not understand the complexity of this latter problem even for arithmetic circuits of depth three.

Challenge Problem. *Complexity of computing the leading monomial of depth-three arithmetic circuits.* Either:

> Devise a randomized polynomial-time algorithm to compute the leading monomial of an arithmetic circuit of depth three ($\Sigma\Pi\Sigma$-circuits).

Or:

> Show that the existence of any efficient algorithm for this problem will lead to a collapse of the polynomial hiearchy.

We would also like to pose the following two problems which are special cases of polynomial equivalence testing: devise an efficient algorithm (if such an algorithm exists) to test if a given polynomial is equivalent to

1. the determinant
2. the permanent.

References

[AB03] Manindra Agrawal and Somenath Biswas. Primality and identity testing via chinese remaindering. *J. ACM*, 50(4):429–443, 2003.

[ABKPM09] Eric Allender, Peter Bürgisser, Johan Kjeldgaard-Pedersen, and Peter Bro Miltersen. On the complexity of numerical analysis. *SIAM J. Comput.*, 38(5):1987–2006, 2009.

[Agr05] Manindra Agrawal. Proving lower bounds via pseudo-random generators. In R. Ramanujam and Sandeep Sen, editors, *FSTTCS 2005: Foundations of Software Technology and Theoretical Computer Science, 25th International Conference, Hyderabad, India, December 15-18, 2005, Proceedings*, volume 3821 of *Lecture Notes in Computer Science*, pages 92–105. Springer, 2005.

[AKPBM06] Allender, Kjeldgaard-Pedersen, Burgisser, and Miltersen. On the complexity of numerical analysis. In *Annual IEEE Conference on Computational Complexity (formerly Annual Conference on Structure in Complexity Theory)*, volume 21, 2006.

[AS06] Manindra Agrawal and Nitin Saxena. Equivalence of f-algebras and cubic forms. In *Proceedings of the Symposium on Theoretical Aspects of Computer Science*, volume 3884 of *Lecture Notes in Computer Science*. Springer, 2006.

[BS83] Walter Baur and Volker Strassen. The complexity of partial derivatives. *Theoretical Computer Science*, 22:317–330, 1983.

[Car06] E. Carlini. *Reducing the number of variables of a polynomial,* Algebraic geometry and geometric modelling, pages 237–247. Mathematics and Visualization. Springer, 2006.

[DS05] Zeev Dvir and Amir Shpilka. Locally decodable codes with 2 queries and polynomial identity testing for depth 3 circuits. In *STOC*, pages 592–601, 2005.

[Har75] D. K. Harrison. A grothendieck of higher degree forms. *Journal of Algebra*, 35:123–128, 1975.

[IK03] Impagliazzo and Kabanets. Derandomizing polynomial identity tests means proving circuit lower bounds. In *STOC: ACM Symposium on Theory of Computing (STOC)*, 2003.

[Kal88] Erich Kaltofen. Greatest common divisors of polynomials given by straight-line programs. *Journal of the ACM*, 1(35):231–264, 1988.

[Kal89] Erich Kaltofen. Factorization of polynomials given by straight-line programs. *Randomness and Computation*, 5, 1989.

[KN97] E. Kushilevitz and N. Nisan. *Communication Complexity.* Cambridge University Press Cambridge, 1997.

[KP07] Pascal Koiran and Sylvain Perifel. The complexity of two problems on arithmetic circuits. *Theoretical Computer Science*, 1-2(389):172–181, 2007.

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

[KS06] Neeraj Kayal and Nitin Saxena. Polynomial identity testing for depth 3 circuits. In *Proceedings of the twenty-first Annual IEEE Conference on Computational Complexity (CCC)*, 2006.

[KS09] Neeraj Kayal and Shubhangi Saraf. Blackbox polynomial identity testing for depth 3 circuits. *Electronic Colloquium on Computational Complexity (ECCC)*, 16(032), 2009.

[LV98] Daniel Lewin and Salil P. Vadhan. Checking polynomial identities over any field: Towards a derandomization? In *STOC*, pages 438–447, 1998.

[Mal07] Guillaume Malod. The complexity of polynomials and their coefficient functions. In *Proceedings of the Conference on Computational Complexity*, pages 193–204, 2007.

[MH74] Y. I. Manin and M. Hazewinkel. *Cubic forms: algebra, geometry, arithmetic.* North-Holland Publishing Co., Amsterdam, 1974.

[MVV87] Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987.

[RS04] Ran Raz and Amir Shpilka. Deterministic polynomial identity testing in non-commutative models. In *IEEE Conference on Computational Complexity*, pages 215–222, 2004.

[Sax06] Nitin Saxena. *Automorphisms of rings and applications to complexity.* PhD thesis, Indian Institute of Technology Kanpur, 2006.

[Sax08] Nitin Saxena. Diagonal circuit identity testing and lower bounds. In *ICALP (1)*, pages 60–71, 2008.

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

[SV08] Amir Shpilka and Ilya Volkovich. Read-once polynomial identity testing. In *STOC*, pages 507–516, 2008.

[vL99] Jacobus Hendricus van Lint. *Introduction to coding theory.* Springer, 1999.

[Zip79] R. Zippel. Probabilistic algorithms for sparse polynomials. In *ISSAC '79: Proc. Int'l. Symp. on Symbolic and Algebraic Computation,* Lecture Notes in Computer Science, Vol. 72. Springer-Verlag, 1979. Zippel discusses probabilistic methods for testing polynomial identities and properties of systems of polynomials.

Appendix

A The complexity of CoeffSLP

The aim of this section of the appendix is to give a simpler, self-contained proof of the following theorem.

Theorem 11. CoeffSLP is #P-complete.

We first give some warm-up lemmas.

Lemma 32. *For any $m \geq 7$, the lcm of the first m numbers is at least 2^m.*

Lemma 33. *For any integer $t \in \mathbb{Z}_{\geq 1}$ and prime p, there is a prime $r = O(t^2 \cdot \log p)$ such that the ring*

$$R \stackrel{\text{def}}{=} \mathbb{F}_p[z]/\langle \frac{z^r - 1}{z - 1} \rangle$$

is the direct sum of finite fields of size $q > p^t$.

Proof. Consider the integer

$$M := (p-1) \cdot (p^2 - 1) \cdot \ldots \cdot (p^t - 1).$$

Then $M < p^{t^2}$. By lemma 32, there exists a prime $r < \log M$ such that r does not divide M. This is the prime r that we seek. Let m denote the order of p modulo r , i.e. m is the smallest positive integer such that $p^m = 1 \pmod r$. Since r does not divide $M = \prod_{i \in [t]}(p^i - 1)$, therefore r does not divide any $(p^i - 1)$ for $1 \leq i \leq t$ and therefore $m > t$. Let $\phi_r(z)$ denote the r-th cyclotomic polynomial, that is

$$\phi_r(z) \stackrel{\text{def}}{=} \frac{z^r - 1}{z - 1}.$$

It is known that over $\mathbb{F}_p$, $\phi_r(z)$ factors into $\frac{r-1}{m}$ irreducible polynomials each of degree m. Thus, $R \stackrel{\text{def}}{=} \mathbb{F}_p[z]/\langle \phi_r(z) \rangle$ is the direct sum of finite fields of size $p^m > p^t$. □

Proof of Theorem 11: The #P-hardness of this problem is well-known and a proof can be found for example in [AKPBM06]. It is sufficient to show this for univariate polynomials (by replacing each indeterminate x_i by an exponentially increasing sequence of monomials, if necessary). That is, our problem now becomes the following: given a circuit of size s computing a univariate polynomial $f(x)$ and an $\alpha \in \mathbb{Z}_{\geq 0}$ given in binary, compute the coefficient of x^α in $f(x)$. Notice that $D \stackrel{\text{def}}{=} 2^s$ is an upper bound on $\deg(f(x))$. Using lemma 33, we obtain an extension ring R of the form $R = \mathbb{F}_p[z]/\langle \frac{z^r-1}{z-1} \rangle$ such that $r \leq (\log D)^2 \cdot (\log p)$ and

$$R \cong \mathbb{F}_q \oplus \ldots \oplus \mathbb{F}_q,$$

with $q - 1 > D$. We now observe that the coefficient of x^α in $f(x)$ is given by

$$\text{Coeff}(x^\alpha, f(x)) = -\sum_{\beta \in R^*} \beta^\alpha \cdot f(\beta^{-1}).$$

The number of terms in the above summation is exponentially large but notice that each summand in the above expression, $(\beta \cdot f(\beta^{-1}))$, is polynomial-time computable so that overall this sum is computable in $P^{\#\text{P}}$. □

B Polynomial Decomposition

We are now set to generalize the sum of powers problem considered in section 7. Given a polynomial $f(\mathbf{X})$, we want to write it as the sum of two polynomials on disjoint sets of variables. That is, our aim is to find an invertible linear transformation A on the variables such that

$$f(A \cdot \mathbf{X}) = g(x_1, \ldots, x_t) + h(x_{t+1}, \ldots, x_n)$$

We first consider the special case of the above we just want to partition the set of variables $X = Y \uplus Z$ so that $f(\mathbf{X}) = g(\mathbf{y}) + h(\mathbf{z})$.

Lemma 34. *Given a low-degree polynomial $f(\mathbf{X})$, we can efficiently compute a partition $X = Y \uplus Z$ of the variables such that $f(\mathbf{X}) = g(\mathbf{y}) + h(\mathbf{z})$, if such a partition exists.*

Proof. Observe that given $f(\mathbf{X})$ and two variables x_i and x_j we can efficiently determine whether there is any monomial in f which contains both these variables by plugging in randomly chosen value for the remaining variables and determining whether the resulting bivariate polynomial has any such monomial or not. Now create an undirected graph G_f whose nodes are the variables and there is and edge between the nodes x_i and x_j if and only if there is a monomial in $f(\mathbf{X})$ which contains both x_i and x_j. We find the connected components of G_f. The partitioning of the set of variables induced by the connected components of G_f gives the required partition of variables needed for decomposition. □

Our main interest though is in devising an algorithm for polynomial decomposition that allows arbitary invertible linear transformations of the variables. Now let $f(\mathbf{X})$ be a regular polynomial. Suppose that for some invertible linear transformation $A \in \mathbb{F}^{(n \times n)*}$:

$$f(A \cdot \mathbf{X}) = g(x_1, \ldots, x_t) + h(x_{t+1}, \ldots, x_n)$$

Without loss of generality, we can assume that $\textsc{Det}(A) = 1$. Let $F(\mathbf{X}) = f(A \cdot \mathbf{X})$. Then observe that

$$\textsc{Det}(H_F)(\mathbf{X}) = \textsc{Det}(H_g)(\mathbf{X}) \cdot \textsc{Det}(H_h)(\mathbf{X})$$

Now by lemma 21 we have

$$\textsc{Det}(H_f)(A \cdot \mathbf{X}) = \textsc{Det}(H_g)(A \cdot \mathbf{X}) \cdot \textsc{Det}(A \cdot H_h)(\mathbf{X}).$$

Also observe that $\textsc{Det}(H_g)(\mathbf{X})$ is in fact a polynomial in the variables $x_1, \ldots, x_t$ whereas $\textsc{Det}(H_h)(\mathbf{X})$ is a polynomial in the remaining $(n-t)$ variables $x_{t+1}, \ldots, x_n$. This motivates us to look at a multiplicative version of the polynomial decomposition problem. Let $D(\mathbf{X})$ be the polynomial $\textsc{Det}(H_f)(\mathbf{X})$. Then we want to make an invertible linear transformation on the variables and write D as the product of polynomials on disjoint sets of variables.

A multiplicative version of polynomial decomposition

We are given a polynomial $D(\mathbf{X})$ and we want to make a linear transformation B on the variables to get a factorization of the form

$$D(B \cdot \mathbf{X}) = C_1(x_1, \ldots, x_{t_1}) \cdot C_2(x_{t_1+1}, \ldots, x_{t_1+t_2}) \cdot \ldots \cdot C_k(x_{n-t_n+1}, \ldots, x_n),$$

where the individual C_i's are 'multiplicatively indecomposable'.
Towards this end, let us make a definition. For a polynomial $f(\mathbf{X})$, we denote by $f^{\perp\perp}$ the vector space orthogonal to $\partial(f)^{\perp}$. That is,

$$f^{\perp\perp} \stackrel{\text{def}}{=} \{\mathbf{a} \in \mathbb{F}^n | \mathbf{a} \cdot \mathbf{v} = 0 \ \ \forall \mathbf{v} \in \partial(f)^{\perp}\}$$

Intuitively, a basis for $f^{\perp\perp}$ corresponds to the essential variables of $f(\mathbf{X})$.

Notice that any factor $C(\mathbf{X})$ of the multivariate polynomial $D(\mathbf{X})$ depends on a subset of the variables which $D(\mathbf{X})$ itself depends upon. Furthermore $D(\mathbf{X})$ does depend on all the variables in any divisor $C(\mathbf{X})$.

Lemma 35. *If a polynomial $D(\mathbf{X})$ has the factorization*

$$D(\mathbf{X}) = C_1(\mathbf{X})^{e_1} \cdot C_2(\mathbf{X})^{e_2} \cdot \ldots \cdot C_k(\mathbf{X})^{e_k},$$

then the space $D^{\perp\perp}$ is the linear span of the spaces ${C_1}^{\perp\perp}, {C_2}^{\perp\perp}, \ldots, {C_k}^{\perp\perp}$.

Lemma 35 together with Kaltofen's algorithm for factoring low-degree polynomials allows us to devise an efficient algorithm for a multiplicative version of polynomial decomposition.

Theorem 36. *There exists an efficient randomized algorithm that given a regular low-degree polynomial $D(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$, computes an invertible linear transformation $A \in \mathbb{F}^{(n\times n)*}$ such that*

$$D(A \cdot \mathbf{X}) = C_1(x_1, \ldots, x_{t_1}) \cdot C_2(x_{t_1+1}, \ldots, x_{t_1+t_2}) \cdot \ldots \cdot C_k(x_{n-t_n+1}, \ldots, x_n),$$

where the individual C_i's are multiplicatively indecomposable, if such a transformation A exists.

Polynomial Decomposition Algorithm

We now give the algorithm for the usual notion of decomposition of polynomials.

Input. A regular low-degree n-variate polynomial $f(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$.

Output. An invertible linear transformation A such that $f(A \cdot \mathbf{X})$ is the sum of two polynomials on disjoint sets of variables.

The Algorithm.

1. Compute an arithmetic circuit $D(\mathbf{X})$ which computes $\textsc{Det}(H_f(\mathbf{X}))$.
2. Use the multiplicative polynomial decomposition algorithm of theorem 36 to determine a linear transformation $A \in \mathbb{F}^{(n\times n)*}$ such that

   $$D(A \cdot \mathbf{X}) = C_1(x_1, \ldots, x_{t_1}) \cdot C_2(x_{t_1+1}, \ldots, x_{t_1+t_2}) \cdot \ldots \cdot C_k(x_{n-t_n+1}, \ldots, x_n),$$

   where the individual C_i's are multiplicatively indecomposable. If no such A exists then output no decomposition exists.
3. Use the algorithm of lemma 34 check if $f(A \cdot \mathbf{X})$ can be written as the sum of two polynomials on disjoint sets of variables. If so output A else output no such decomposition exists.

The following theorem summarizes the conditions under which the above algorithm is guaranteed to give the right answer.

Theorem 37. *Given a n-variate polynomial $f(\mathbf{X}) \in \mathbb{F}[\mathbf{X}]$, the algorithm above finds a decomposition of $f(\mathbf{X})$, if it exists, in randomized polynomial time provided $\textsc{Det}(H_f(\mathbf{X}))$ is a regular polynomial, i.e. it has n variables upto equivalence.*

ECCC ISSN 1433-8092
http://eccc.hpi-web.de