Electronic Colloquium on Computational Complexity, Report No. 21 (2011)

A Case of Depth-3 Identity Testing, Sparse Factorization and Duality

Chandan Saha
Max Plank Institute for Informatics
Saarbrücken, Germany

Ramprasad Saptharishi*
Chennai Mathematical Institute
Chennai, India

Nitin Saxena
Hausdorff Center for Mathematics
Bonn, Germany

February 14, 2011

Abstract

Finding an efficient solution to the general problem of polynomial identity testing (PIT) is a challenging task. In this work, we study the complexity of two special but natural cases of identity testing - first is a case of depth-3 PIT, the other of depth-4 PIT.

Our first problem is a vast generalization of: verify whether a bounded top fanin depth-3 circuit equals a *sparse* polynomial (given as a sum of monomial terms). Formally, given a depth-3 circuit C, having constant many general product gates and arbitrarily many *semidiagonal* product gates, test if the output of C is identically zero. A semidiagonal product gate in C computes a product of the form $m \cdot \prod_{i=1}^{b} \ell_i^{e_i}$, where m is a monomial, ℓ_i is a linear polynomial and b is a constant. We give a deterministic polynomial time test, along with the computation of *leading* monomials of semidiagonal circuits over local rings.

The second problem is on verifying a given sparse polynomial factorization, which is a classical question (von zur Gathen, FOCS 1983): Given multivariate sparse polynomials $f, g_1, \ldots, g_t$ explicitly, check if $f = \prod_{i=1}^{t} g_i$. For the special case when every g_i is a sum of univariate polynomials, we give a deterministic polynomial time test. We characterize the factors of such g_i's and even show how to test the divisibility of f by the powers of such polynomials.

The common tools used are Chinese remaindering and *dual* representation. The dual representation of polynomials (Saxena, ICALP 2008) is a technique to express a product-of-sums of univariates as a sum-of-products of univariates. We generalize this technique by combining it with a generalized Chinese remaindering to solve these two problems (over any field).

*Supported by MSR India Ph.D Fellowship

ISSN 1433-8092

1 Introduction

Polynomial identity testing (PIT) is one of the most fundamental problems of algebraic complexity theory. A central object of study in the subject of algebraic complexity is the arithmetic circuit model of computation. Arithmetic circuits, an algebraic analogue of boolean circuits with addition and multiplication gates replacing 'or' and 'and' gates, form a natural and concise way to represent polynomials in many variables. It has always been an interesting and challenging task for the research community to understand the powers and limits of this model.

Identity testing is an algorithmic problem on arithmetic circuits. It is the task of checking if the output of a given arithmetic circuit is zero as a formal polynomial. Interestingly, this natural algebraic problem is deeply connected to fundamental lower bound questions in complexity theory [KI03, Agr05], and is also critically used in the designing of efficient algorithms for several other important problems [AKS04, Lov79, CDGK91]. Some of the other milestone results in complexity theory, like $\mathsf{IP} = \mathsf{PSPACE}$ [LFKN90, Sha90] and the PCP theorem [ALM$^+$98] also involve PIT.

Identity testing does admit a randomized polynomial time algorithm - choose a random point from a sufficiently large field and evaluate the circuit [Sch80, Zip79]. With high probability, the output is non-zero if the circuit computes a non-zero polynomial. There are several other more efficient randomized algorithms [CK97, LV98, AB99, KS01]. Refer to the survey [AS09] for a detailed account of them.

Derandomizing identity testing is important from both the algorithmic and the lower bound perspectives. For instance, the deterministic primality test in [AKS04] involves derandomization of a certain polynomial identity test. On the other hand, it is also known [KI03, Agr05] that a certain strong derandomization of PIT implies that the permanent polynomial requires super-polynomial sized arithmetic circuits (in other words, $\mathsf{VP} \neq \mathsf{VNP}$).

Derandomizing the general problem of PIT has proven to be a difficult endeavor. Restricting the problem to constant depth circuits, the first non-trivial case arise for depth-3 circuits. Quite strikingly, it is now known [AV08] that depth-4 circuits capture the difficulty of the general problem of PIT: A black-box identity test for depth-4 circuits gives a quasi-polynomial time PIT algorithm for circuits computing low degree polynomials. (Another notable result [Raz10], but more in the flavor of lower bounds, shows that strong enough depth-3 circuit lower bounds imply super-polynomial lower bounds for general arithmetic formulas.) Although, the general case of depth-3 PIT is still open, some special cases, like depth-3 circuits with bounded top fanin [KS07, SS11] and *diagonal* depth-3 circuits [Sax08] are known to have deterministic polynomial time solutions. In the depth-4 setting, it is known how to do (black-box) PIT for multilinear circuits with bounded top fan-in [SV11] also in polynomial time. Refer to the surveys [Sax09] and [SY10] for details on results concerning depth-3 and depth-4 PIT (and much more).

1.1 The motivation

The motivation behind our work is a question on 'composition of identity tests'. Suppose we know how to perform identity tests efficiently on two classes of circuits $\mathcal{A}$ and $\mathcal{B}$. How easy is it to solve PIT on the class of circuits $\mathcal{A}+\mathcal{B}$? The class $\mathcal{A}+\mathcal{B}$ is made up of circuits C of the form $C_1 + C_2$, where $C_1 \in \mathcal{A}$ and $C_2 \in \mathcal{B}$. In other words, the root node of C is an addition gate with the roots of C_1 and C_2 as its children. (Notice that, PIT on the class $C_1 \times C_2$ is trivial.) Depending on the classes $\mathcal{A}$ and $\mathcal{B}$, this question can be quite non-trivial to answer. For instance, suppose we are given $t+1$ sparse polynomials $f, g_1, \ldots, g_t$, explicitly as sums of monomials, and asked to check if $f = \prod_{i=1}^{t} g_i$. Surely, it is easy to check if f or $\prod_{i=1}^{t} g_i$ is zero. But, it is not clear how to perform the test $f - \prod_{i=1}^{t} g_i \overset{?}{=} 0$. (This problem has also been declared open in a work by von zur Gathen [vzG83] on sparse multivariate polynomial factoring.) The test $f - \prod_{i=1}^{t} g_i \overset{?}{=} 0$ is one of the most basic cases of depth-4 PIT that is still open. Annoyingly enough, it shows that while PIT on depth-3 circuits with top fanin 2 is trivial, PIT on depth-4 circuits with top fanin 2 is far from trivial.

We wonder what can be said about composition of subclasses of depth-3 circuits. Two of the non-trivial classes of depth-3 circuits for which efficient PIT algorithms are known are the classes of bounded top fanin [KS07] and diagonal (or semidiagonal) circuits [Sax08]. The question is - Is it possible to glue together the seemingly disparate methods of [KS07] and [Sax08] and give a PIT algorithm for the composition of bounded top fanin and semidiagonal circuits? In this work, we answer this question in the affirmative. Our technique also applies to a special case of the depth-4 problem: $f - \prod_{i=1}^{t} g_i \overset{?}{=} 0$.

1.2 Our contribution

We give deterministic polynomial time algorithms for two problems on identity testing - one is on a class of depth-3 circuits, while the other is on a class of depth-4 circuits. As mentioned in Section 1.1, both these classes can be viewed as composition of subclasses of circuits over which we already know how to perform PIT.

Our first problem is a common generalization of the problems studied in [KS07] and [Sax08]. We need the following definition of a *semidiagonal* circuit.

Definition 1. (Semidiagonal circuit) *Let C be a depth-3 circuit, i.e. a sum of products of linear polynomials. The* top fanin *of C is the number of such product gates. If each product gate in C computes a polynomial of the form $m \cdot \prod_{i=1}^{b} \ell_i^{e_i}$, where m is a monomial, ℓ_i is a linear polynomial in the n input variables and b is a constant, then C is a* semidiagonal circuit.

Remark - Our results hold even with the relaxed definition of semidiagonal circuits where b is not necessarily a constant but $\prod_{i=1}^{b}(1+e_i) \leq \mathsf{poly}(\text{size}(C))$. For this, we need a slightly tighter analysis for dual representation in Section 2 (refer to [Sax08]). This gets interesting when all e_i's are $O(1)$ as our test could then afford upto $b = O(\log \text{size}(C))$.

Problem 1. *Given a depth-3 circuit C_1 with bounded top fanin and given a semidiagonal circuit C_2, test if the output of the circuit $C_1 + C_2$ is identically zero.*

Our second problem is a special case of checking a given sparse multivariate polynomial factorization (thus, a case of depth-4 top fanin 2 PIT).

Problem 2. *Given $t+1$ polynomials $f, g_1, \ldots, g_t$ explicitly as sum of monomials, where every g_i is a sum of univariate polynomials, check if $f = \prod_{i=1}^{t} g_i$.*

It is possible that though f is sparse, some of its factors are *not* sparse (an example is provided in Section 5). So, multiplying the g_i's in a brute force fashion is not a feasible option. In this paper, we show the following:

Theorem 1.1. *Problems 1 and 2 can be solved in deterministic polynomial time.*

The asymptotics of Theorem 1.1 appear at the end of Sections 4 and 5.

1.3 An overview of our approach

Our first common tool in solving both Problem 1 and Problem 2 is called the *dual representation* of polynomials [Sax08]. It is a technique to express a power of a sum of univariates as a 'small' sum of product of univariates. In the latter representation we show how to efficiently compute the leading monomial (and its coefficient), under the natural lexicographic ordering (Section 3). This observation turns out to be crucial in solving Problem 1. Finding the leading monomial of a general depth-3 circuit is supposedly a harder problem than identity testing [KP07]. But, fortunately, it turns out to be efficiently doable in the particular case we are interested in.

The second common tool is called *Chinese remaindering* (CR). In the most basic form, CR says that if two coprime polynomials f and g divide h then fg divides h. In Problem 1 we use a more involved version of CR over certain local rings [KS07]. Over these local rings, we show how to compute and utilize the dual representation to do PIT (Section 4.2). In Problem 2 the CR is over the base field but to exploit it we develop a divisibility test using duality (Section 5.1) and characterize the factors of a sum of univariates (Section 5.2). The heart of the paper is in making those combinations of duality and CR actually work.

Remark - The results of this paper apply to any field $\mathbb{F}$. However, for the sake of better readability, we have presented our results assuming the characteristic of $\mathbb{F}$ to be zero or sufficiently large. In Appendix A, we point out the necessary adjustments to our arguments which make them work for small characteristic fields as well. In that case, the overall ideas are similar but significant algebraic generalizations are used. A common theme is to do computations over a local ring with *prime-power* characteristic.

1.3.1 Approach to solving Problem 1

Let p and f be the polynomials computed by the circuits C_1 and C_2, respectively. The general idea is to begin by applying the Kayal-Saxena test [KS07] to the polynomial p. This test uses some invertible linear maps to transform the polynomial p, thereby altering f in the process. However, since C_1 has bounded top fanin, the transformed f continues to retain its semidiagonal property (Definition 1) over a local ring. Identity testing of a semidiagonal circuit is made easy via its dual representation. So, if we can ensure that the Kayal-Saxena (partial) test on $p + f$ reduces to identity testing of semidiagonal circuits over local rings then we are done. Let $p = \sum_{i=1}^{k} P_i$, where P_i is a product of linear polynomials and k is a constant. The original Kayal-Saxena test (where we test if $p = 0$) chooses a P_j such that $LM(P_j) \succeq LM(p)$ (LM stands for the leading monomial and $\succeq$ refers to the monomial ordering). But now since the test is on $p + f$, at an intermediate stage of the algorithm we need to find a P_j such that $LM(P_j) \succeq LM(p + f)$, where f is semidiagonal. This is where we need to find the leading monomial of f, which we achieve by considering its dual representation (see Section 3 and Section 4).

1.3.2 Approach to solving Problem 2

The approach we take to check if $f = \prod_{i=1}^{t} g_i$ is quite intuitive - reduce the problem to checking divisibility and then use Chinese remaindering. But the two issues here are: how to check $g_i \mid f$, and that g_i's need not be coprime. So in general we need to check if g^d divides f for some $d \geq 1$. We show that such divisibility checks reduce to identity testing of a (slightly general) form of semidiagonal circuits. Finally, for Chinese remaindering to work, we need to say something about the coprimality of g_i and g_j. Towards this, we show an irreducibility result on polynomials of the form $f(x) + g(y) + h(z)$ that helps us conclude the proof (see Section 5).

2 Preliminaries: The dual representation

In this section, we recall how to express a power of a sum of univariate polynomials as a 'small' sum of product of univariates. The idea is already present in [Sax08]. We reproduce it here as we need it often.

Let $f = \left(\sum_{i=1}^{n} g_i(x_i)\right)^D$, where $g_i(x_i)$ (or simply g_i) is a univariate in x_i of degree at most d. Assume that the underlying field $\mathbb{F}$ has size greater than $(nD+1)$ and that char$(\mathbb{F}) = 0$, or greater than D. Let z be a new variable and $g = \sum_{i=1}^{n} g_i$. Then in the exponential power series expansion $e^{gz} = \sum_{j=0}^{\infty} \frac{g^j}{j!} z^j$, $f/D!$ is the coefficient of z^D. On the other hand, e^{gz} is also the product of the formal power series $e^{g_i z}$ for $1 \leq i \leq n$, i.e. $e^{gz} = \prod_{i=1}^{n} e^{g_i z}$. Define the polynomials $E_D(g_i, z) = \sum_{j=0}^{D} \frac{{g_i}^j}{j!} z^j$. Then the coefficient of z^D in the formal power series $\sum_{j=0}^{\infty} \frac{g^j}{j!} z^j$ is exactly equal to the coefficient of z^D in the polynomial $P(z) := \prod_{i=1}^{n} E_D(g_i, z)$. The idea is to use interpolation to express this coefficient of z^D as an $\mathbb{F}$-linear combination of few evaluations of $P(z)$ at different field points.

Suppose $P(z) = \sum_{j=0}^{nD} p_j z^j$, where p_j's are polynomials in $x_1, \ldots, x_n$. Choose $(nD + 1)$ distinct points $\alpha_0, \ldots, \alpha_{nD}$ from $\mathbb{F}$ and V be the $(nD + 1) \times (nD + 1)$ Vandermonde matrix $(\alpha_j^k)_{0 \leq j,k \leq nD}$. Then $\mathsf{V} \cdot (p_0, \ldots, p_{nD})^T = (P(\alpha_0), \ldots, P(\alpha_{nD}))^T$, which implies $(p_0, \ldots, p_{nD})^T = \mathsf{V}^{-1} \cdot (P(\alpha_0), \ldots, P(\alpha_{nD}))^T$. In other words, p_D can be expressed as an $\mathbb{F}$-linear combination of the $P(\alpha_j)$'s for $0 \leq j \leq nD$. Now, to complete the sketch, notice that $p_D = f/D!$ and each $P(\alpha_j)$ is a product of the univariates $E_D(g_i, \alpha_j)$ of degree (in x_i) at most dD. This proves the following theorem.

Theorem 2.1. *[Sax08] Given $f = \left(\sum_{i=1}^{n} g_i(x_i)\right)^D$, where g_i is a univariate in x_i of degree at most d, f can be expressed as a sum of $(nD+1)$ products of univariates of degree dD, in $\mathsf{poly}(n, d, D)$ time.* $\square$

3 Finding the leading monomial of semidiagonal circuits

In this section, we present an efficient algorithm to compute the leading monomial of a semidiagonal circuit f. This will be crucial in both the problems that we later solve.

Definition 2. (Monomial and coefficient) *Let m be a monomial and f a polynomial in the variables $x_1, \ldots, x_n$. Fix a monomial ordering $\succeq$ by $x_1 \succ \cdots \succ x_n$. $LM(f)$ denotes the* leading monomial *in f (conventionally, $LM(0) := \emptyset$ and we define $x_i \succ \emptyset$, for all i). We denote the* coefficient of m in f *by $[m]f$.*

We remark that given a semidiagonal circuit f finding $LM(f)$ is indeed a non-trivial problem since the leading monomials generated by the product terms $m \cdot \prod_{i=1}^{b} \ell_i^{e_i}$ in f might cancel off in such a way that $LM(f)$ is strictly smaller than any of the leading monomials of the terms $m \cdot \prod_{i=1}^{b} \ell_i^{e_i}$. Using Theorem 2.1 and by the definition of semidiagonal circuits, it is sufficient to present an algorithm for finding the leading monomial of a sum of product of univariates. Our argument is similar in spirit to that of Raz and Shpilka [RS04] non-commutative formula identity test, but with the added feature that it also finds the coefficient of the leading monomial, which is supposedly a harder problem in the general case.

Theorem 3.1. *Given $f = \sum_{i=1}^{k} \prod_{j=1}^{n} g_{ij}(x_j)$ over a field $\mathbb{F}$, where g_{ij} is a univariate in x_j of degree at most d, the leading monomial of f (and its coefficient) can be found in $\mathsf{poly}(nkd)$ time.*

Proof. The following discussion gives an efficient iterative algorithm to find both $LM(f)$ and $[LM(f)]f$.

In general, at the ℓ^{th} iteration we need to find the leading monomial of a polynomial of the form $f_\ell = \sum_{i=1}^{k} G_{i,\ell-1} \cdot \prod_{j=\ell}^{n} g_{ij}$, where $G_{i,\ell-1}$'s are polynomials in $x_1, \ldots, x_{\ell-1}$ and $LM(f) = LM(f_\ell)$. To begin, $\ell = 1$ and $G_{i0} = 1$ for all $1 \le i \le k$.

Fix an integer $\ell \in [1, n]$. Consider the monomials with nonzero coefficients occurring in the products $G_{i,\ell-1} \cdot g_{i\ell}$, for all $1 \le i \le k$. Let the set of these monomials be $M_\ell := \{m_1, \ldots, m_{\mu(\ell)}\}$ such that $m_1 \succ m_2 \succ \ldots \succ m_{\mu(\ell)}$. Similarly, consider the monomials with nonzero coefficients in the partial products $\prod_{j=\ell+1}^{n} g_{ij}$ for all $1 \le i \le k$ and call them $N_\ell := \{n_1, \ldots, n_{\nu(\ell)}\}$ (also assume that $n_1 \succ n_2 \succ \ldots \succ n_{\nu(\ell)}$). Then, for any i, $G_{i,\ell-1} \cdot g_{i\ell} = \sum_{r=1}^{\mu(\ell)} c_{ir} m_r$ and $\prod_{j=\ell+1}^{n} g_{ij} = \sum_{s=1}^{\nu(\ell)} d_{is} n_s$, where the coefficients c_{ir} and d_{is} belong to $\mathbb{F}$.

Notice that the monomials $m_r \cdot n_s$ are distinct for distinct tuples (r, s). The coefficient of $m_r \cdot n_s$ in f_ℓ is exactly $\sum_{i=1}^{k} c_{ir} d_{is}$. Put differently, if $\mathbf{c}_r$ is the k-dimensional row vector $(c_{1r}, \ldots, c_{kr})$ and $\mathbf{d}_s$ is the row vector $(d_{1s}, \ldots, d_{ks})$ then $[m_r \cdot n_s] f_\ell = \mathbf{c}_r \boldsymbol{.} \mathbf{d}_s^T$. Consider the $\mu(\ell) \times k$ matrix C with vectors $\mathbf{c}_r$ as rows, for all $1 \le r \le \mu(\ell)$, and D be the $k \times \nu(\ell)$ matrix with vectors $\mathbf{d}_s^T$ as columns, for all $1 \le s \le \nu(\ell)$. The coefficient of $m_r \cdot n_s$ is the $(r, s)^{th}$ entry of the product matrix CD.

We are interested in finding $LM(f_\ell)$. Now notice that the lexicographically smallest possible index (r, s) for which there is a nonzero entry in CD, gives the leading monomial $m_r \cdot n_s$ of f_ℓ. This is because of the monomial orderings $m_1 \succ m_2 \succ \ldots \succ m_{\mu(\ell)}$ and $n_1 \succ n_2 \succ \ldots \succ n_{\nu(\ell)}$. Therefore, if there is a row vector $\mathbf{c}_r$ in C that is $\mathbb{F}$-linearly dependent on the vectors $\mathbf{c}_1, \ldots, \mathbf{c}_{r-1}$ then $LM(f_\ell)$ can never be of the form $m_r \cdot n_s$ for any s, and so we can safely drop the row $\mathbf{c}_r$ from C. Since C is a $\mu(\ell) \times k$ matrix, the number of linearly independent rows of C is at most k. Hence, we can iteratively remove rows from C starting from the first row $\mathbf{c}_1$; at the r^{th} iteration dropping $\mathbf{c}_r$ if and only if $\mathbf{c}_r$ is $\mathbb{F}$-linearly dependent on $\mathbf{c}_1, \ldots, \mathbf{c}_{r-1}$. Finally, we are left with a new matrix $\tilde{C}$ with at most k rows.

The dropping of a row $\mathbf{c}_r$ from C corresponds to pruning the monomial m_r from the product $G_{i,\ell-1} \cdot g_{i\ell}$. By this we mean the following. Let $\tilde{M}_\ell$ be the subset of those monomials of M_ℓ such that $m_r \in \tilde{M}_\ell$ if and only if $\mathbf{c}_r$ is a row of $\tilde{C}$. Let $G_{i\ell}$ be the product $G_{i,\ell-1} \cdot g_{i\ell}$ projected to only the monomials in $\tilde{M}_\ell$ i.e. $G_{i\ell} := \sum_{m_r \in \tilde{M}_\ell} c_{ir} m_r$. Then the leading monomial of $f_{\ell+1} := \sum_{i=1}^{k} G_{i\ell} \cdot \prod_{j=\ell+1}^{n} g_{ij}$ is the same as $LM(f_\ell)$ for the reason explained in the last paragraph. The good part is that $\tilde{M}_\ell$ has at most k monomials and so do the polynomials $G_{i\ell}$ for all $1 \leq i \leq k$. Since the number of monomials in $G_{i\ell}$ does not grow with ℓ, it is easy to see that both $LM(f)$ and $[LM(f)]f$ can be found in $\mathsf{poly}(nkd)$ time by adapting the above discussion into an iterative algorithm. $\square$ $\square$

Corollary 3.2. *Let C be a semidiagonal circuit (Definition 1) of top fanin s and formal degree d. Then $LM(C)$ and $[LM(C)]C$ are computable in $\mathsf{poly}(s, (nd)^b)$ time.*

Proof. We do this by replacing each power of a linear polynomial in the product gates of C by its dual expression given by Theorem 2.1. Next, we partially expand out each product gate to still get a sum of products of univariates. Per product gate, this could blow up the final expression size to at most $(nd)^{O(b)}$. $\square$

4 Solving Problem 1

In Problem 1, suppose p and f be the polynomials computed by C_1 and C_2, respectively. We need to test if $p + f = 0$. Assume, without any loss of generality, that p and f are homogeneous polynomials having the same degree d. Let $p = \sum_{i=1}^{k} P_i$, where k is a constant and P_i is a product of d linear forms over $\mathbb{F}$, for all $1 \leq i \leq k$. Let $X = \{x_1, \ldots, x_n\}$ be the underlying set of variables. Also, suppose s is the number of product gates in C_2. Our algorithm builds upon the Kayal-Saxena test [KS07], which tests if $p = 0$. To put things in context, we briefly review their algorithm first.

4.1 Reviewing the Kayal-Saxena test

The algorithm in [KS07] uses recursion on the top fanin of C_1 to check if $p = 0$. At an intermediate level of the recursion, the algorithm finds out if a polynomial of the form, $\sum_{i=1}^{k'} \alpha_i T_i$, is zero over a local ring $\mathcal{R} \supseteq \mathbb{F}$, where $k' \leq k$ and α_i's are elements of $\mathcal{R}$. Each T_i is a product of linear polynomials over $\mathcal{R}$, where a linear polynomial over $\mathcal{R}$ is of the kind, $\sum_{i=1}^{n'} a_i x_{v_i} + \tau$, where $a_i \in \mathbb{F}$ and $\tau \in \mathcal{M}$, the unique maximal ideal of $\mathcal{R}$. In addition, there is at least one nonzero a_i, and $x_{v_i} \in X$ is a free variable over $\mathcal{R}$. At the start of the recursion, $\mathcal{R} = \mathbb{F}$ and $\mathcal{M} = (0)$, the null ideal.

The following Algorithm **ID** takes as input an $\mathbb{F}$-basis of a local ring $\mathcal{R}$, and checks if the polynomial $\sum_{i=1}^{k'} \alpha_i T_i$ is identically zero over $\mathcal{R}$ and returns YES or NO accordingly. The description of Algorithm **ID** given below is the same as in [KS07], except a slight modification in Step 3.1 which is somewhat necessary for our purpose.

Algorithm ID($\mathcal{R}, \langle \alpha_1, \ldots, \alpha_{k'} \rangle, \langle T_1, \ldots, T_{k'} \rangle$):
Step 1: (Rearranging terms) Order the terms $\alpha_i T_i$ so that $LM(T_1) \succeq LM(T_i)$, for all $2 \leq i \leq k'$. Let $p = \sum_{i=1}^{k'} \alpha_i T_i$.
Step 2: (Base Case) If $k' = 1$ check if $\alpha_1 = 0$. If so, return YES otherwise NO.
Step 3: (Verifying that $p = 0 \bmod T_1$) The product T_1 can be split as $T_1 = S_1 \ldots S_r$, with possible change in the constant α_1, such that each S_j is of the form, $S_j = (\ell_j + \tau_1) \cdot (\ell_j + \tau_2) \ldots (\ell_j + \tau_{t_j})$, where $\tau_i \in \mathcal{M}$ and ℓ_j is a linear form over $\mathbb{F}$. Further, for $i \neq j$, ℓ_i and ℓ_j are coprime linear forms over $\mathbb{F}$. Check if $p = 0 \bmod S_j$, for every $1 \leq j \leq r$. This is done in the following way.

> Step 3.1: (Applying a linear transformation) Find a free variable x_u with nonzero coefficient c_u in ℓ_j. Define an invertible linear transformation σ on the free variables (occuring in $T_1, \ldots T_{k'}$), sending ℓ_j to x_u, as follows: $\sigma(x_u) = c_u^{-1}(x_u - \ell_j + c_u x_u)$ and for any other free variable $x_v \neq x_u$, $\sigma(x_v) = x_v$.
>
> Step 3.2: (Recursively verify if $\sigma(p) = 0 \bmod \sigma(S_j)$) Define ring $\mathcal{R}'$ as, $\mathcal{R}' = \mathcal{R}[x_u]/(\sigma(S_j))$, where x_u is the same variable x_u in Step 3.1. Notice that, $\sigma(S_j)$ is of the form, $\sigma(S_j) = (x_u + \tau_1) \cdots (x_u + \tau_{t_j})$, and $\sigma(T_1) = 0 \bmod \sigma(S_j)$. For $2 \leq i \leq k'$, compute elements $\beta_i \in \mathcal{R}'$ and T_i' such that, $\sigma(T_i) = \beta_i T_i' \bmod \sigma(S_j)$, where T_i' is a product of linear polynomials over $\mathcal{R}'$.

Recursively call **ID**$(\mathcal{R}', \langle \beta_2, \ldots, \beta_{k'} \rangle, \langle T'_2, \ldots, T'_{k'} \rangle)$. If the recursive call returns NO then output NO and exit, otherwise declare $p = 0 \bmod S_j$.

Declare $p = 0 \bmod T_1$, if $p = 0 \bmod S_j$ for all $1 \leq j \leq r$.
Step 4: Compute $[LM(T_1)]p$ by considering i's such that $LM(T_i) = LM(T_1)$ and computing the sum (over such i's) of $\alpha_i \cdot \prod_j [LM(\ell_{ij})]\ell_{ij}$, where $T_i = \prod_j \ell_{ij}$. If $[LM(T_1)]p = 0$ then return YES else NO.
Remark - For the reader's reference, the correctness and complexity analysis of Algorithm **ID** are sketched in Appendix B.

4.2 Adapting Algorithm ID to solve Problem 1

Let us apply Algorithm **ID** to the polynomial $p + f$. We cannot apply it directly as $p + f$ is of unbounded top fanin. We intend to apply it partially and then 'jump' to the dual representation of the intermediate f. Since f is semidiagonal, the number of distinct linear forms (which are not variables) in each product term of C_2 is at most b (see Definition 1).

At an intermediate level of the recursion, Algorithm **ID** tests if a polynomial of the form, $q = \sum_{i=1}^{k'} \alpha_i T_i + \sum_{r=1}^{s'} \beta_r \omega_r$ is zero over a local ring $\mathcal{R}$, where $k' \leq k$, $s' \leq s$ and $\alpha_i, \beta_r \in \mathcal{R}$. As before, every T_i is a product of linear polynomials over $\mathcal{R}$. And each ω_r is a product of a monomial (in the free variables over $\mathcal{R}$) and $b + k - k'$ powers of distinct linear forms over $\mathcal{R}$. Further, $\deg(T_i)$ and $\deg(\omega_r)$ are bounded by d. The part $\sum_{r=1}^{s'} \beta_r \omega_r = \tilde{f}$ (say), in q, shows how the polynomial f in $p+f$ evolves with different levels of the recursion. At the beginning, $\tilde{f} = f$.

In Step 1 of Algorithm **ID**, we are required to find a term T_1 such that $LM(T_1) \succeq LM(T_i)$ for all $2 \leq i \leq k'$. The purpose of this step is to ensure that $LM(T_1) \succeq LM(q)$, something we need for arguing the correctness of the algorithm. But now, our polynomial q has an added term $\tilde{f}$. So, to ensure $LM(T_1) \succeq LM(q)$, we also need to find the leading monomial of $\tilde{f}$, which is a polynomial over $\mathcal{R}$. We will show in a short while how to find $LM(\tilde{f})$. Suppose, we know $LM(\tilde{f})$. If $LM(\tilde{f}) \succ LM(T_i)$ for all $1 \leq i \leq k'$ then surely $q \neq 0$ over $\mathcal{R}$ and the algorithm returns NO. Otherwise, there is a T_1 (say) such that $LM(T_1) \succeq LM(q)$ and Algorithm **ID** proceeds to Step 2 with that T_1.

In Step 2, the base case is no longer $k' = 1$ but instead $k' = 0$. In this case, we have to check if $\tilde{f} = 0$ in $\mathcal{R}$ and this can be done efficiently since we will show shortly how to find $LM(\tilde{f})$.

Step 3 remains the same as before, except that in Step 3.2 we also need to compute $\sigma(\tilde{f})$, where σ is the linear transformation. Notice that, σ maps only x_u to a linear polynomial and keeps other free variables intact. So, the product term $\sigma(\omega_r)$ has at most one power of a linear polynomial more than that of ω_r (as x_u gets replaced by $\sigma(x_u)$). Since the depth of the recursion of Algorithm **ID** is at most k, every ω_r in $\tilde{f}$ is the product of a monomial and at most $b + k$ powers of linear polynomials over $\mathcal{R}$: To begin with, every product term $m \cdot \prod_{i=1}^{b} \ell_i^{e_i}$ in f has at most b distinct (non-variable) linear forms, and at every level of the recursion, a linear transformation σ might replace a variable of m by a linear form. As $b + k$ is a constant, $\tilde{f}$ is a 'semidiagonal circuit over $\mathcal{R}$'.

Finally, in Step 4, we need to confirm that $[LM(T_1)]q = 0$. For this, we may have to find $[LM(\tilde{f})]\tilde{f}$, if $LM(\tilde{f})$ happens to be the same as $LM(T_1)$. This is taken care of by the way we find $LM(\tilde{f})$ which also gives us its coefficient. We now show how to find $LM(\tilde{f})$. (Note: at any recursive stage, with base ring $\mathcal{R}$, the 'monomials' are in the free variables and thus it is implicit that they take precedence over the other variables that moved inside $\mathcal{R}$.)

Dual representation of $\tilde{f}$ - Let $\ell^{d'}$ be a term occuring in the product ω_r, where $\ell = a_1 x_{v_1} + \ldots + a_{n'} x_{v_{n'}} + \tau$ is a linear polynomial over $\mathcal{R}$. (Assume that $x_{v_1}, \ldots x_{v_{n'}}$ are the free variables over $\mathcal{R}$). Replace τ by a formal variable z and use Theorem 2.1 to express $(a_1 x_{v_1} + \ldots + a_{n'} x_{v_{n'}} + z)^{d'}$ as, $(a_1 x_{v_1} + \ldots + a_{n'} x_{v_{n'}} + z)^{d'} = \sum_{i=1}^{(n'+1)d'+1} g_i(z) \cdot g_{i1}(x_{v_1}) \ldots g_{in'}(x_{v_{n'}})$, where g_i and g_{ij} are univariates over $\mathbb{F}$ of degree at most d'. Therefore, $\ell^{d'}$ can be expressed as, $\ell^{d'} = \sum_{i=1}^{(n'+1)d'+1} \gamma_i \cdot g_{i1}(x_{v_1}) \ldots g_{in'}(x_{v_{n'}})$, where $\gamma_i = g_i(\tau) \in \mathcal{R}$, in time $\mathsf{poly}(n, d, \dim_{\mathbb{F}} \mathcal{R})$. Since ω_r is a product of a monomial (in $x_{v_1}, \ldots x_{v_{n'}}$) and at most $b + k$ products of powers of linear forms over $\mathcal{R}$, using the above equation, each ω_r can be expressed as,

$\omega_r = \sum_{i=1}^{O((nd)^{b+k})} \gamma_{i,r} \cdot g_{i1,r}(x_{v_1}) \ldots g_{in',r}(x_{v_{n'}})$, where $\gamma_{i,r} \in \mathcal{R}$ and $g_{ij,r}$ is a polynomial over $\mathbb{F}$ of degree $O((b+k)d)$, in time $\mathsf{poly}((nd)^{b+k}, \dim_{\mathbb{F}} \mathcal{R})$.

Therefore, given the polynomial $\tilde{f}$ its representation $\tilde{f} = \sum_{i=1}^{O(s\cdot(nd)^{b+k})} \gamma_i \cdot g_{i1}(x_{v_1}) \ldots g_{in'}(x_{v_{n'}})$, (reusing symbols γ_i and g_{ij}) can be computed in time $\mathsf{poly}(s, (nd)^{b+k}, \dim_{\mathbb{F}} \mathcal{R})$.

Finding $LM(\tilde{f})$ - Let $\{e_1, \ldots, e_{\dim_{\mathbb{F}} \mathcal{R}}\}$ be an $\mathbb{F}$-basis of $\mathcal{R}$. In the representation of the polynomial $\tilde{f} = \sum_{i=1}^{O(s\cdot(nd)^{b+k})} \gamma_i \cdot g_{i1}(x_{v_1}) \ldots g_{in'}(x_{v_{n'}})$, let $\gamma_i = \sum_{j=1}^{\dim_{\mathbb{F}} \mathcal{R}} b_{ij} e_j$, where $b_{ij} \in \mathbb{F}$. Consider the polynomials, $q_j = \sum_{i=1}^{O(s\cdot(nd)^{b+k})} b_{ij} \cdot g_{i1}(x_{v_1}) \ldots g_{in'}(x_{v_{n'}})$, for all $1 \leq j \leq \dim_{\mathbb{F}} \mathcal{R}$. Then, $LM(\tilde{f})$ is the highest among the leading monomials of the polynomials q_j. From Theorem 3.1, the leading monomial (and its coefficient) of every q_j can be computed in time $\mathsf{poly}(s, (nd)^{b+k})$. Thus, $[LM(\tilde{f})]\tilde{f}$ can also be found in time $\mathsf{poly}(s, (nd)^{b+k}, \dim_{\mathbb{F}} \mathcal{R})$.

Using an analysis similar to the complexity analysis of Algorithm **ID** (see Appendix B.2) and observing that $\dim_{\mathbb{F}} \mathcal{R} \leq d^k$, we can show that Algorithm **ID**, adapted to work for $p + f$, takes time $\mathsf{poly}(s, (nd)^{b+k})$ (recall that s is the top fan-in of circuit C_2). This solves Problem 1 in deterministic polynomial time as promised.

5 Solving Problem 2

The naïve approach of multiplying all g_i's is infeasible because of intermediate swelling in the number of monomials. Consider the following example. Let $f = \prod_{i=1}^{n} (x_i^d - 1)$ be the input polynomial that has $s = 2^n$ monomials. Suppose that the factors (i.e. the g_i's) given to us are, $(x_1-1), (x_1-\zeta), \ldots, (x_1-\zeta^{d-1}), \ldots, (x_n-1), (x_n-\zeta), \ldots, (x_n-\zeta^{d-1})$, where ζ is a primitive d^{th} root of unity, and we want to check if f equals the product of these factors. If we do not follow any particular rule in multiplying these factors then we may as well end up with the intermediate product, $\prod_{i=1}^{n} (x_i^{d-1} + x_i^{d-2} + \ldots + 1)$, which has $d^n = s^{\log d}$ monomials. Thus, given f with s monomials we might have to spend time and space $O(s^{\log d})$ if we follow this naïve approach.

Notice that, when g_i's are linear polynomials, Problem 2 becomes a special case of Problem 1 and can therefore be solved in deterministic polynomial time. However, the approach given in Section 4 does not seem to generalize directly to the case when, instead of linear functions, g_i's are sums of univariates. This case is handled in this section.

5.1 Checking divisibility by (a power of) a sum of univariates

Given $g_1, \ldots, g_t$, group together the polynomials g_i's that are just $\mathbb{F}$-multiples of each other. After this is done, we need to check if f is equal to a product of the form $a \cdot g_1^{d_1} \ldots g_t^{d_t}$ (reusing symbol t), where $a \in \mathbb{F}$ and $g_i \neq b \cdot g_j$ for any $b \in \mathbb{F}$ if $i \neq j$. Suppose g_i and g_j are coprime for $i \neq j$. (This assumption is justified later in Section 5.2). Then, Problem 2 gets reduced to the problem of checking divisibility followed by a comparison of the leading monomials of f and $a \cdot g_1^{d_1} \ldots g_t^{d_t}$. The latter is easy as we have f and g_i's explicitly. Checking divisibility, however, is more technical and we do that in this section. We once again use the tool of dual representation, but on a slightly more general form of semidiagonal polynomials.

Theorem 5.1. *Checking divisibility of a sparse polynomial f by g^d, where g is a sum of univariates, can be done in deterministic polynomial time.*

Proof. Let $g = \sum_{i=1}^{n} u_i(x_i)$, where u_i is a univariate in x_i. Assume without loss of generality that $u_1 \neq 0$. Consider replacing the partial sum $\sum_{i=2}^{n} u_i(x_i)$ in g by a new variable y so that we get a bivariate $h = (u_1(x_1) + y)^d$, which is a sparse polynomial as well. Let $\deg_{x_1} u_1 = e$. Given any power of x_1, say x_1^k, we can employ long division with respect to the variable x_1 to find the remainder when x_1^k is divided by h. This division is possible since h is monic in x_1. It is not hard to see that the remainder, say r_k, thus obtained is a bivariate in x_1 and y with degree in x_1 less than ed and degree in y at most k.

To check if g^d divides f, do the following. For every monomial of f, replace the power of x_1 occuring in the monomial, say x_1^k, by the corresponding remainder r_k. After the replacement process is over, completely multiply out terms in the resulting polynomial, say $\tilde{f}(x_1, x_2, \ldots, x_n, y)$, and express it as sum of monomials in the variables $x_1, \ldots, x_n$ and y. Now notice that, $f \bmod g^d = \tilde{f}(x_1, x_2, \ldots, x_n, \sum_{i=2}^n u_i(x_i))$. Since degree of x_1 in $\tilde{f}$ is less than ed, polynomial g^d divides f if and only if $\tilde{f}(x_1, x_2, \ldots, x_n, \sum_{i=2}^n u_i(x_i))$ is identically zero. Polynomial $\tilde{f}$ with y evaluated at $\sum_{i=2}^n u_i(x_i)$ is of the form of a sum of products, where each product term looks like $m \cdot (\sum_{i=2}^n u_i(x_i))^j$ for some monomial m and integer j. This form is similar to that of a semidiagonal circuit (Definition 1), except that $\sum_{i=2}^n u_i(x_i)$ is a sum of univariates instead of a linear form.

To test if $\tilde{f}(x_1, x_2, \ldots, x_n, \sum_{i=2}^n u_i(x_i)) = 0$, use Theorem 2.1 to express it as a sum of product of univariates. Finally, invoke Theorem 3.1 to test if the corresponding sum of product of univariates is identically zero.

The time complexity of this reduction to identity testing includes computing each remainder r_k, which takes $\mathsf{poly}(k, ed)$ time. Assuming there are s monomials in f, to express $\tilde{f}$ as a sum of monomials in $x_1, x_2, \ldots, y$ it takes $\mathsf{poly}(s, D, ed)$ time, where D is the total degree of f. Accounting for the substitution of y by $\sum_{i=2}^n u_i(x_i)$, the total reduction time is $\mathsf{poly}(n, s, D, ed)$. Finally, testing if $\tilde{f}(x_1, x_2, \ldots, x_n, \sum_{i=2}^n u_i(x_i))$ is identically zero takes polynomial time (by Theorems 2.1 and 3.1). $\square$

5.2 Irreducibility of a sum of univariates

In this section, the polynomial g's are assumed to be sums of univariates. We show that if g_i and g_j depend on at least three variables then they are essentially irreducible. Hence, they are either coprime or same (upto a constant multiple). Of course, it is trivial to check the latter case and this makes our Chinese remaindering idea workable.

Theorem 5.2. *(Irreducibility) Let g be a polynomial over a field $\mathbb{F}$ that is a sum of univariates. Suppose g depends non-trivially on at least three variables. Then either g is irreducible, or g is a p-th power of some polynomial where $p = \mathsf{char}(\mathbb{F})$.*

Remark - Such a statement is false when g is a sum of just two univariates. For eg., the real polynomial $g := x_1^4 + x_2^4$ has factors $(x_1^2 + x_2^2 \pm x_1 x_2 \sqrt{2})$ (which is not even a sum of univariates!).

Denote $\frac{\partial h}{\partial x_i}$ by $\partial_i h$ for any polynomial h. We need the following observations (with g of Theorem 5.2).

Observation 5.3. *Let $g = u \cdot v$ be a non-trivial factorization. Then both u and v are monic in every variable that g depends on.*

Proof. If u is not monic in, say, x_1 then fix an ordering amongst the variables such that x_1 is the highest. Then the leading monomial of $g = u \cdot v$ is a mixed term which is not possible. $\square$

Observation 5.4. *If g is not a p-th power of any polynomial then it is square-free.*

Proof. Suppose not, then $g = u^2 v$ for some polynomials u and v. If g is not a p-th power, there must exist a variable x_i such that $0 \neq \partial_i g = 2uv\partial_i u + u^2 \partial_i v$. Since u divides the RHS, u must be a univariate (since $\partial_i g$ is a univariate in x_i) which is not possible (by Observation 5.3). $\square$

Proof of Theorem 5.2. Assume that g is not a p-th power of any polynomial. Then there exists a variable, say x_1, such that $\partial_1 g \neq 0$. Suppose $g = u \cdot v$; this means $\partial_i g = (\partial_i u)v + (\partial_i v)u$. Denote by $h_{x_i = \alpha}$, the polynomial h evaluated at $x_i = \alpha$, where $\alpha \in \bar{\mathbb{F}}$, the algebraic closure of $\mathbb{F}$.

Claim 5.5. *There exists an $\alpha \in \bar{\mathbb{F}}$ such that $u_{(x_1=\alpha)}, v_{(x_1=\alpha)} \neq 0$ and they share a common factor.*

Assume that the claim is true. There are two variables other than x_1, say x_2 and x_3, that appear in g. Then, $\partial_2 g = (\partial_2 u)v + (\partial_2 v)u$ implies that $(\partial_2 g)_{(x_1=\alpha)} = \partial_2 g = (\partial_2 u)_{(x_1=\alpha)} v_{(x_1=\alpha)} + (\partial_2 v)_{(x_1=\alpha)} u_{(x_1=\alpha)}$. Similarly, $\partial_3 g = (\partial_3 u)_{(x_1=\alpha)} v_{(x_1=\alpha)} + (\partial_3 v)_{(x_1=\alpha)} u_{(x_1=\alpha)}$. If both $\partial_2 g$ and $\partial_3 g$ are non-zero, then the last two equations forces gcd $\left(u_{(x_1=\alpha)}, v_{(x_1=\alpha)}\right)$ to divide $\partial_2 g$ and $\partial_3 g$. But this leads to a contradiction since the partial derivatives are univariates in x_2 and x_3, respectively.

Suppose $\partial_2 g$ is zero. Since g is square-free, u and v do not share any factor and hence all of $\partial_2 g, \partial_2 u$ and $\partial_2 v$ are zero, which implies that every occurrence of x_2 in g, u and v has exponent that is a multiple of $p = \text{char}(\mathbb{F})$. Hence, $g' = u' \cdot v'$, where $g(x_1, x_2, \cdots, x_n) =: g'(x_1, x_2^p, \cdots, x_n)$, $u(x_1, x_2, \cdots, x_n) =: u'(x_1, x_2^p, \cdots, x_n)$, and $v(x_1, x_2, \cdots, x_n) =: v'(x_1, x_2^p, \cdots, x_n)$. By inducting on this equation, we get the desired contradiction. $\square$

Proof of the above Claim. Consider the univariates (in x_1), $g_1 := \partial_1 g$, $u_1 := \partial_1 u$ and $v_1 := \partial_1 v$ and $w := \gcd(g_1, u_1, v_1)$. Then, $g_1/w = (u_1/w)\, v + (v_1/w)\, u$. Since x_1-degree of u_1 is less than that of g_1, the univariate g_1/w has degree in x_1 at least one. Substituting $x_1 = \alpha$, where α is a root of g_1/w, we get a non-trivial equation of the form $a \cdot v_{(x_1=\alpha)} + b \cdot u_{(x_1=\alpha)} = 0$, which implies that the $\gcd(u_{(x_1=\alpha)}, v_{(x_1=\alpha)})$ is non-trivial. And these two polynomials are certainly non-zero by Observation 5.3. $\square$

5.3 Finishing the argument

In Section 5.1, we have assumed that the polynomials $g_1, \ldots, g_t$ are irreducible. Although, Theorem 5.2 justifies our assumption when g_i depends on three or more variables, we need to slightly change our strategy for bivariates and univariates. In this case, we first take pairwise gcd of the bivariates (similarly, pairwise gcd of the univariates) and factorize accordingly till all the bivariates (similarly, the univariates) are mutually coprime. Taking gcd of two bivariate (monic) polynomials takes only polynomial time using Euclidean gcd algorithm. Once coprimeness is ensured, we can directly check if a bivariate $g_i^{d_i}$ divides f by expressing f as $f = \sum_j f_j m_j$, where f_j's are bivariate polynomials depending on the same two variables as g_i and m_j's are monomials in the remaining variables. Now, $g_i^{d_i} \mid f$ if and only if $g_i^{d_i} \mid f_j$ for all j. Once again, just like gcd, bivariate divisibility is a polynomial time computation (simply by long division). Finally, we can use Chinese remaindering to complete the argument in a similar fashion as in Section 5.1.

To summarize, we can check if $f = \prod_{i=1}^{t} g_i$, where g_i's are sums of univariates, in time $\mathsf{poly}(n, t, d, s)$, where d is the bound on the total degrees of f and the g_i's, and s is the number of monomials (with non-zero coefficients) in f.

6 Discussion and open problems

We conclude by posing two open questions related to the problems studied here. The first relates to depth-4 fanin 2 PIT.

Open Problem 1: Find a deterministic polynomial time algorithm to check if $f = \prod_{i=1}^{t} g_i^{d_i}$, where f is a sparse polynomial and the g_i's are mutually coprime, bounded degree polynomials.

One particular case of interest is when the g_i's are quadratic forms. Observe that a polynomial g^d divides f if and only if g divides f and g^{d-1} divides $\frac{\partial f}{\partial x_1}$ (assuming f depends on x_1 and $\deg(f) > \text{char}(\mathbb{F})$). Since $\frac{\partial f}{\partial x_1}$ is also sparse, using this observation, the problem eventually boils down to checking if g divides h, where both g and h are sparse polynomials. Now suppose g is a quadratic form. It is known that there exists an efficiently computable linear transformation σ on the variables such that $\sigma(g) = \sum_{i=1}^{r} x_i^2$, which is a sum of univariates. The polynomial g divides h if and only if $\sigma(g)$ divides $\sigma(h)$. We have shown how to divide a sparse polynomial by a sum of univariates. But, the issue here is that $\sigma(h)$ need not be sparse - it is an image of a sparse h under an invertible σ. Is it possible to resolve this issue?

The second relates to depth-4 higher fanin PIT.

Open Problem 2: Find a deterministic polynomial time algorithm to solve PIT on depth-4 circuits with bounded top fan-in k, where each of the k multiplication gates is a product of sums of univariate polynomials.

Note that, a solution for $k = 2$ easily follows from Theorem 5.2 and unique factorization. But, it is unclear how to solve this problem even for $k = 3$. The problem can also be seen as a certain generalization of bounded top fanin depth-3 PIT [KS07] to the case of depth-4 circuits.

References

[AB99] Manindra Agrawal and Somenath Biswas. Primality and Identity Testing via Chinese Remaindering. In *FOCS*, pages 202–209, 1999.

[Agr05] Manindra Agrawal. Proving Lower Bounds Via Pseudo-random Generators. In *FSTTCS*, pages 92–105, 2005.

[AKS04] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. Primes is in P. *Annals of Mathematics*, 160(2):781–793, 2004.

[ALM$^+$98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof Verification and the Hardness of Approximation Problems. *Journal of the ACM*, 45(3):501–555, 1998.

[AS09] Manindra Agrawal and Ramprasad Saptharishi. Classifying polynomials and identity testing. *Technical Report, IIT Kanpur*, 2009.

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

[CDGK91] Michael Clausen, Andreas W. M. Dress, Johannes Grabmeier, and Marek Karpinski. On Zero-Testing and Interpolation of k-Sparse Multivariate Polynomials Over Finite Fields. *Theor. Comput. Sci.*, 84(2):151–164, 1991.

[CK97] Zhi-Zhong Chen and Ming-Yang Kao. Reducing Randomness via Irrational Numbers. In *STOC*, pages 200–209, 1997.

[KI03] Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. In *STOC*, pages 355–364, 2003.

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

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

[KS07] Neeraj Kayal and Nitin Saxena. Polynomial Identity Testing for Depth 3 Circuits. *Computational Complexity*, 16(2):115–138, 2007.

[LFKN90] Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic Methods for Interactive Proof Systems. In *FOCS*, pages 2–10, 1990.

[Lov79] László Lovász. On determinants, matchings, and random algorithms. In *FCT*, pages 565–574, 1979.

[LV98] Daniel Lewin and Salil P. Vadhan. Checking Polynomial Identities over any Field: Towards a Derandomization? In *STOC*, pages 438–447, 1998.

[Raz10] Ran Raz. Tensor-rank and lower bounds for arithmetic formulas. In *STOC*, pages 659–666, 2010.

[RS04] Ran Raz and Amir Shpilka. Deterministic Polynomial Identity Testing in Non-Commutative Models. In *CCC*, pages 215–222, 2004.

[Sax08] Nitin Saxena. Diagonal Circuit Identity Testing and Lower Bounds. In *ICALP (1)*, pages 60–71, 2008.

[Sax09] Nitin Saxena. Progress on Polynomial Identity Testing. *Bulletin of the EATCS*, (90):49–79, 2009.

[Sch80] Jacob T. Schwartz. Fast Probabilistic Algorithms for Verification of Polynomial Identities. *J. ACM*, 27(4):701–717, 1980.

[Sha90] Adi Shamir. IP=PSPACE. In *FOCS*, pages 11–15, 1990.

[SS11] Nitin Saxena and C. Seshadhri. Blackbox identity testing for bounded top fanin depth-3 circuits: the field doesn't matter. *43rd ACM Symposium on Theory of Computing, (STOC)*, 2011. (available at ECCC, `http://www.eccc.uni-trier.de/report/2010/167/`).

[SV11] Shubhangi Saraf and Ilya Volkovich. Black-box identity testing of depth-4 multilinear circuits. *43rd ACM Symposium on Theory of Computing, (STOC)*, 2011.

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

[vzG83] Joachim von zur Gathen. Factoring Sparse Multivariate Polynomials. In *FOCS*, pages 172–179, 1983.

[Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. *EUROSAM*, pages 216–226, 1979.

A Extensions to small characteristic fields

Over fields of small characteristics, the dual representation for semidiagonal circuit has to be modified slightly. The following section gives an appropriate modification.

A.1 Preliminaries: The dual representation

This is a version of Theorem 2.1 that applies for fields of small characteristics by moving to appropriate *Galois rings* (like Galois fields but with prime-power characteristic!).

Theorem A.1. *Given $f = \left(\sum_{i=1}^{n} g_i(x_i)\right)^D$, where g_i is a univariate in x_i of degree at most d over a field $\mathbb{F}$ of characteristic p, we can find an integer $N \leq \mathsf{poly}(n, D)$ such that $p^N f$ can be expressed as a sum of $(nD+1)$ products of univariates of degree dD over the ring $\mathbb{Z}/p^{N+1}\mathbb{Z}$, in $\mathsf{poly}(n, d, D)$ time.*

Proof. Our main trick is to do computations on f over a ring of p-power characteristic. To exhibit this trick let us assume that $\mathbb{F}$ is a prime field $\mathbb{F}_p$. Consider $\sum_{i=1}^{n} g_i(x_i)$ as a polynomial over the rationals (i.e. view $\mathbb{F}_p$ as integers $\{0, \ldots, p-1\}$), then Theorem 2.1 gives an expression for f:

$$f \quad = \quad \sum_{i=1}^{nD+1} \prod_{j=1}^{n} g_{ij}(x_j)$$

Some of the coefficients on the RHS might be rationals with powers of p appearing in the denominator. By multiplying the above expression by a suitably large power of p (eg., the largest power in any denominator), the above can be rewritten as:

$$p^N \cdot f \quad = \quad \sum_i \prod_j g'_{ij}(x_j)$$

where the coefficients on the RHS are free of any p's in the denominator. Therefore, this is a nontrivial expression for $p^N \cdot f$ in the ring $\mathbb{Z}/p^{N+1}\mathbb{Z}$, as claimed. □

In the case of finite base field $\mathbb{F}_q = \mathbb{F}_p[x]/(\mu(x))$ with the leading coefficient of μ being one, we begin by considering f as a polynomial over the field $\mathbb{Q}[x]/(\mu(x))$ (note: $\mu(x)$ remains irreducible over $\mathbb{Q}$). Now the above proof proceeds in exactly the same way, thereby obtaining a nontrivial expression for $p^N \cdot f$ over the (Galois) ring $\mathbb{Z}[x]/(p^{N+1}, \mu(x))$.

In the case of characteristic p but infinite base field $\mathbb{F}_p(t)$, we begin by considering f as a polynomial over the field $\mathbb{Q}(t)$. The above proof gives a nontrivial expression for $p^N \cdot f$ over the ring: $\left(\mathbb{Z}/p^{N+1}\mathbb{Z}\right)[t]$ *localized* by its set of non-zerodivisors.

Thus, using a combination of the above tricks, we can get a dual representation over any extension of $\mathbb{F}_p$.

With this modified dual representation, Problems 1 and 2 can be tackled the same way as outlined earlier. The only additional requirement is to find the leading monomial and the associated coefficient of a semidiagonal circuit over these rings of prime-power characteristic.

A.2 Finding the leading monomial of semidiagonal circuits

Given a semidiagonal circuit f, Theorem A.1 can be used (repeatedly) to write $p^N f$ as a sum of product of univariates over a ring $\mathbb{Z}/p^{N+1}\mathbb{Z}$. If $f = \sum \alpha_m m$, where m is a monomial and α_m is the associated coefficient, then $p^N f = \sum (p^N \alpha_m) m$. Therefore, if we can find the leading monomial and coefficient of $p^N f$ over $\mathbb{Z}/p^{N+1}\mathbb{Z}$, we can derive the leading monomial and coefficient of f. Thus, we need a version of Theorem 3.1 over rings of the form $\mathbb{Z}/p^{N+1}\mathbb{Z}$.

One of the key elements in the proof of Theorem 3.1 is that we can prune down the list of vectors $\{\mathbf{c_1}, \cdots, \mathbf{c_r}\}$ from $\mathbb{F}^k$ to always ensure that we have the vector contributing to the leading monomial in our set. For this we required a "dimension argument" that states that if $r > k$ then there exists an i such that $\mathbf{c_i}$ can be written as a linear combination of the lower indexed vectors.

While working over more general rings, we need to work with 'vectors' (abusing terminology) over rings of the form $\mathbb{Z}/p^N\mathbb{Z}$. Unlike the case when we are working over a field, a linear combination of the form $\sum \alpha_i \mathbf{c_i} = 0$ does not necessarily translate to writing one of the $\mathbf{c_i}$'s as a combination of the preceeding vectors since some of the coefficients in the linear combination might not be invertible. Here is an example of Nk row vectors over $\mathbb{Z}/p^N\mathbb{Z}$ such that there is no $\mathbf{c_i}$ that can be written as a linear combination of the preceeding rows. Let $\mathbf{v}$ be the column vector $(p^{N-1}, p^{N-2}, \ldots, 1)^T$ and $\mathbf{0} = (0, \cdots, 0)^T$ be the column vector of N zeroes. Consider

$$\begin{bmatrix} \mathbf{v} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{v} & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{v} \end{bmatrix}_{Nk \times k}$$

However, the following lemma shows that such a $\mathbf{c_i}$ would indeed exist if we had more than Nk vectors.

Lemma A.2. *Let $\{\mathbf{c_1}, \cdots, \mathbf{c_r}\}$ be elements of $\left(\mathbb{Z}/p^N\mathbb{Z}\right)^k$. If $r > Nk$, then there exists an i such that $\mathbf{c_i}$ can be written as a linear combination of $\{\mathbf{c_1}, \cdots, \mathbf{c_{i-1}}\}$. That is,*

$$\mathbf{c_i} \quad = \quad \sum_{j=1}^{i-1} \alpha_j \mathbf{c_j} \qquad \text{where } \alpha_j\text{'s are from } \mathbb{Z}/p^N\mathbb{Z}.$$

Proof. Given an ordered set of vectors $\{\mathbf{c_1}, \cdots, \mathbf{c_r}\}$, we shall call a linear combination $\sum \alpha_j \mathbf{c_j}$ a "monic" linear combination if the highest indexed $\mathbf{c_i}$ appearing in the linear combination has coefficient 1.

Construct the $r \times k$ matrix whose rows are from $\{\mathbf{c_1}, \cdots, \mathbf{c_r}\}$. The goal is to repeatedly apply "monic row transformations", that is replacing a row $\mathbf{c_i}$ by $\mathbf{c_i} - \sum_{j<i} \alpha_j \mathbf{c_j}$. This would eventually induce a zero row, which will give us our desired linear combination. The proof proceeds by induction, handling one coordinate at a time.

Let i be the smallest index (if it exists) such that the first coordinate of $\mathbf{c_i}$ is non-zero modulo p. Replace every other (higher indexed) row $\mathbf{c_j}$ by $(\mathbf{c_j} - \alpha_j \mathbf{c_i})$ to ensure that its first coordinate is now zero modulo p. Observe that these transformations are monic. The first coordinate of every row, besides $\mathbf{c_i}$, will now be divisible by p. Dropping row i, we repeat the procedure to make all of them zero modulo p^2, and similarly for p^3 etc. Thus, by dropping at most N rows, we can ensure that the first coordinate of every row is 0 (modulo p^N).

We are now left with $\{\mathbf{d_1}, \cdots, \mathbf{d_{r'}}\}$, with $r' > r - N$, over $\left(\mathbb{Z}/p^N\mathbb{Z}\right)^{k-1}$ and by induction we would find a monic linear combination of the $\mathbf{d_i}$'s that is zero. It is not hard to see that this translates to a monic linear dependence amongst the $\mathbf{c_i}$'s since we only applied monic transformations. □

With this, we have a version of Theorem 3.1 over rings of prime-power characteristic.

Theorem A.3. *Given $f = \sum_{i=1}^{k} \prod_{j=1}^{n} g_{ij}(x_j)$ over a ring $\mathbb{Z}/p^N\mathbb{Z}$, where g_{ij} is a univariate in x_j of degree at most d, the leading monomial of f (and its coefficient) can be found in $\mathsf{poly}(nkdN)$ time.* □

It is now straightforward to show that Corollary 3.2 holds even over small characteristic fields: just apply Theorem A.1 repeatedly and finally invoke Theorem A.3.

A.3 Solving Problem 1

The proof proceeds in exactly the same lines as described in Section 4.2. The only difference is when we have to compute the leading monomial and its coefficient of a semidiagonal circuit $\tilde{f} = \sum \beta_r \omega_r$ over the local ring.

- In each linear function appearing in ω_r, replace the constant term τ by a fresh variable z. Applying Theorem A.1 to this expression, we get $p^N \tilde{f} = \sum \gamma_i \cdot g_{i1}(x_1) \cdots g_{in}(x_n)$ where $x_1, \cdots, x_n$ are the free variables, and $\gamma_i \in \mathcal{R}$ (note: $\mathcal{R}$ is now of characteristic p^{N+1}).
- Let $\{e_1, \cdots, e_{\dim \mathcal{R}}\}$ be a $(\mathbb{Z}/p^{N+1}\mathbb{Z})$-basis of $\mathcal{R}$, and let $\gamma_i = \sum b_{ij} e_j$. Then the leading monomial and coefficient of $p^N \tilde{f}$ can be computed from those of the polynomials $q_j = \sum_i b_{ij} \cdot g_{i1}(x_1) \cdots, g_{in}(x_n)$.
- Compute the leading monomial and coefficient of each q_j using Theorem A.3 and recover the leading monomial and coefficient of $\tilde{f}$.

The other parts of the proof are independent of the characteristic of the field and hence go through in exactly the same way.

A.4 Solving Problem 2

Again all our arguments were independent of the field except when we invoke semidiagonal PIT in Theorem 5.1. At that point we would now invoke Theorems A.1 and A.3.

B Correctness and complexity of Algorithm ID

B.1 Correctness of the Algorithm

Suppose Step 3 ensures that $p = 0 \bmod T_1$, where $LM(T_1) \succeq LM(T_i)$ for all $2 \leq i \leq k'$ which also means that $LM(T_1) \succeq LM(p)$. The way a linear form is defined over $\mathcal{R}$, it follows that $[LM(T_1)]T_1$ is a nonzero field element. Therefore, $p = \alpha \cdot T_1$ for some $\alpha \in \mathcal{R}$, implying that $p = 0$ if and only if $[LM(T_1)]p = 0$. This is verified in Step 4.

It remains to show the correctness of Step 3. In order to check if $p = 0 \bmod T_1$, the algorithm finds out if $p = 0 \bmod S_j$ for every $1 \leq j \leq r$. That this is a sufficient condition is implied by the following lemma (also known as 'Chinese Remaindering over local rings'). The way a local ring $\mathcal{R}$ is formed in Algorithm **ID** it has the property that every element $r \in \mathcal{R}$ can be uniquely expressed as $r = a + \tau$, where $a \in \mathbb{F}$ and $\tau \in \mathcal{M}$. Let ϕ be a projection map, taking r to a i.e. $\phi(r) = a$. This map naturally extends to polynomials over $\mathcal{R}$ by acting on the coefficients.

Lemma B.1. [KS07] *Let $\mathcal{R}$ be a local ring over $\mathbb{F}$ and $p, g, h \in \mathcal{R}[x_1, \ldots, x_n]$ be multivariate polynomials such that $\phi(g)$ and $\phi(h)$ are coprime over $\mathbb{F}$. If $p = 0 \bmod g$ and $p = 0 \bmod h$ then $p = 0 \bmod gh$.*

Since $\phi(S_j) = \ell_j^{t_j}$ and ℓ_i, ℓ_j are coprime for $i \neq j$, the correctness of Step 3 follows. Finally notice that, $p = 0 \bmod S_j$ if and only if $\sigma(p) = 0 \bmod \sigma(S_j)$ as σ is an invertible linear transformation. The check $\sigma(p) = 0 \bmod \sigma(S_j)$ is done recursively in Step 3.2. The recursion is on the top fanin, as $\sigma(p) = \sum_{i=2}^{k'} \beta_i T_i'$ over the local ring $\mathcal{R}'$, so that the top fanin of $\sigma(p)$ is lowered to $(k' - 1)$.

B.2 Complexity of the Algorithm

Note that, $\deg(T_i') \leq \deg(T_i)$ in Step 3.2. At the start, Algorithm **ID** is called on polynomial p. So, at every intermediate level $\deg(S_j) \leq \deg(T_1) \leq d$. Therefore, $\dim_{\mathbb{F}}(\mathcal{R}') \leq d \cdot \dim_{\mathbb{F}}(\mathcal{R})$. Time spent by Algorithm **ID** is at most $\mathsf{poly}(n, k', d, \dim_{\mathbb{F}}(\mathcal{R}))$ in Steps 1, 2, 3.1 and 4. Moreover, time spent in Step 3.2 is at most d times a smaller problem (with top fanin reduced by 1) while dimension of the underlying local ring gets raised by a factor at most d. Unfolding the recursion, we get the time complexity of Algorithm **ID** on input p to be $\mathsf{poly}(n, d^k)$.