Electronic Colloquium on Computational Complexity, Report No. 42 (2015)

Computations beyond Exponentiation Gates and Applications

Ilya Volkovich [*]

Abstract

In Arithmetic Circuit Complexity the standard operations are $\{+, \times\}$. Yet, in some scenarios exponentiation gates are considered as well (see e.g. [BB98, ASSS12, Kay12, KSS14]). In this paper we study the question of efficiently evaluating a polynomial given an oracle access to its power. That is, beyond an exponentiation gate. As applications, we show that:

- A reconstruction algorithm for a circuit class $\mathcal{C}$ can be extended to handle f^e for $f \in \mathcal{C}$.
- There exists an efficient algorithm for factoring sparse multiquadratic polynomials.
- There exists an efficient algorithm for testing whether two powers of sparse polynomials are equal. That is, $f^d \equiv g^e$ when f and g are sparse.

1 Introduction

Let $f(\bar{x}) \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a polynomial over the field $\mathbb{F}$. In this paper we study the following question: given $e \in \mathbb{N}$ and an oracle access to $f^e \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ can we efficiently construct an oracle access to f? That is, we wish to evaluate f on a set of points $\bar{a}, \bar{b}, \ldots$ (might be unknown upfront) given an oracle access to f^e. An efficient *randomized* algorithm for this problem was given in [KT90]. Where, in fact, a randomized polynomial factorization algorithm was given. In addition, in terms of circuit complexity, it was shown in [Val80, Kal85] that if f^e has a small circuit then so does f [1]. In this paper we focus on solving the algorithmic problem *deterministically.*

It is clear that as the first step, we should be able to extract e-th roots of field elements. For instance, if f is constant. We refer to such an algorithm as an e-th *root oracle* R_e. However, having root oracles is not enough for our task as demonstrated by the following example.

Let $h(x) = 3x - 4$ and $f = h^2$. Suppose that we wish to evaluate $h(x)$ at $x = 1, 2$ given an oracle access to $f(x)$ and using a square-root oracle R_2. As $f(1) = 1, f(2) = 4$ the oracle might return $h(1) = R_2(1) = 1$ and $h(2) = R_2(4) = 2$ (for example, returning the positive root). Note, however, that these evaluations are inconsistent with either $\pm h$! More generally, there could be e different $h_1, \ldots h_e$ polynomials that result in the same polynomial when raised the e-th power (i.e. $\forall i \in [n] : h_i^e = f$). Therefore, in order to prevent the aforementioned situation our algorithm should output an oracle access to exactly one of them. We summarize this in the following theorem which is our main technical contribution.

Theorem 1 (Main Technical Contribution)**.** *There exists a deterministic algorithm that given $e \in \mathbb{N}$, an e-th root oracle R_e and an oracle access to a polynomial $f^e \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ of degree at most d uses* $\mathrm{poly}(n, d, \log|\mathbb{F}|)$ *field operations and oracle calls to R_e, and outputs an oracle access to $\omega \cdot f$, where $\omega \in \mathbb{F}$ is such that $\omega^e = 1$.*

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

[1]When the characteristic of $\mathbb{F}$ is zero or coprime with e.

ISSN 1433-8092

We now discuss related problems and applications.

1.1 Multivariate Polynomial Factorization

One of the fundamental problems in algebraic complexity is the problem of polynomial factorization: given a polynomial $P \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ over a field $\mathbb{F}$, find its irreducible factors. Other than being natural, the problem has many applications such as list decoding [Sud97, GS99] and derandomization [KI04]. A large amount of research has been devoted to finding efficient algorithms for this problem (see e.g. [GG99]) and numerous *randomized* algorithms were designed [GK85, Kal89, KT90, GG99, Kal03, Gat06]. However, the question of whether there exist *deterministic* algorithms for this problem remains an interesting open question (see [GG99, Kay07]).

Perhaps the simplest factorization algorithm is a root oracle. We note that the best known *deterministic* root extraction algorithms over the finite fields have polynomial dependence on the field characteristic p (see e.g. [Sho91, GG99, GKL04, Kay07]). While in the *randomized* setting, this dependence is polynomial in $\log p$. In particular, there is no known efficient deterministic root extraction algorithm when p is large. Over fields with characteristic 0 (e.g. $\mathbb{Q}$) both the *deterministic* and the *randomized* complexities are polynomial in the bit-complexity of the coefficients (see [LLL82]). Therefore, we can say that root extraction is, perhaps, the simplest hard problem. For sake of uniformity we formulate all our results in terms of root oracles and $\log |\mathbb{F}|$ which stands for the bit-complexity of the coefficients in the underlying polynomials.

1.2 Polynomial Reconstruction

Let $\mathbb{F}$ be a field and $\mathcal{C}$ a class of circuits. The *reconstruction* problem for the class $\mathcal{C}$ is defined as follows. Given an oracle access to a polynomial $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$, computable by an a circuit from $\mathcal{C}$, output a circuit $C \in \mathcal{C}$ that computes f. A reconstruction algorithm is *efficient* if the number of queries it makes to f and its running time are polynomial in the size of the representation of f in the class $\mathcal{C}$. The reconstruction problem can be seen as the algebraic analog of the learning problem.

An immediate application of our main theorem is reconstruction beyond an exponentiation gate. More formally, we can efficiently extend a reconstruction algorithm for a circuit class $\mathcal{C}$ to handle polynomials of the form f^e when f is computable by a circuit $C \in \mathcal{C}$. Note that in general f^e might not be computable by a circuit in $\mathcal{C}$.

Theorem 2. *Let A be a deterministic (randomized) reconstruction algorithm for a circuit class $\mathcal{C}$, let $f \in \mathcal{C}$ and let $T(f)$ denote the number of operations A uses to reconstruct f. Then there exists a deterministic (randomized) algorithm that given $e \in \mathbb{N}$, an e-th root oracle R_e and an oracle access to the polynomial $f^e \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ of degree at most d, uses $\mathrm{poly}(n, d, \log |\mathbb{F}|, T(f))$ field operations and oracles calls to R_e and A, and outputs a circuit for $\omega \cdot f$, where $\omega \in \mathbb{F}$ is such that $\omega^e = 1$.*

As a corollary we get to extend reconstruction algorithms for specific classes of circuits. An *s-sparse polynomial* is polynomial with at most s (non-zero) monomials. Sparse polynomials were deeply studied (see e.g. [BOT88, KS01, LV03]) and, in fact, several efficient deterministic reconstruction algorithms were given. Our next result extends the reconstruction algorithm of [KS01] to powers of sparse polynomials.

Theorem 3. *Let $n, s, d, e \in \mathbb{N}$ and let $f(\bar{x}) \in \mathbb{F}[x_1, x_2, \dots, x_n]$ be an s-sparse polynomial of degree at most d. Then there exists a deterministic algorithm that given an oracle access to the polynomial $f^e \in \mathbb{F}[x_1, x_2, \dots, x_n]$ and an e-th root oracle R_e uses $\mathrm{poly}(n, d, e, s, \log|\mathbb{F}|)$ field operations and oracles calls, and outputs $\omega \cdot f$, where $\omega \in \mathbb{F}$ is such that $\omega^e = 1$.*

Read-once formulas are formulas in which each variable appears at most once. A *read-once polynomial* is a polynomial computable by a read-once formula. Those are the smallest possible polynomials that depend on all of their variables. Although they form a very restricted model of computation, read-once formulas received a lot of attention [HH91, KLN+93, AHK93, BHH95, BB98, BC98, SV14a, SV14b, Vol15]. Our next result extends the reconstruction algorithm of [SV14a] to powers of read-once polynomials. We note that the reconstruction algorithm of [BB98] actually deals with a richer model of read-once formulas with exponentiation gates. Yet, that algorithm is randomized.

Theorem 4. *Let $n, e \in \mathbb{N}$ and let $f(\bar{x}) \in \mathbb{F}[x_1, x_2, \dots, x_n]$ be a read-once polynomial. Then there exists a deterministic algorithm that given an oracle access to the polynomial $f^e \in \mathbb{F}[x_1, x_2, \dots, x_n]$ and an e-th root oracle R_e uses $n^{\mathcal{O}(\log n)} \cdot \mathrm{poly}(e, \log|\mathbb{F}|)$ field operations and oracles calls, and outputs a read-once formula Ψ that computes $\omega \cdot f$, where $\omega \in \mathbb{F}$ is such that $\omega^e = 1$.*

1.3 Sparse Polynomial Factorization

Coming up with an efficient deterministic factorization algorithm for sparse polynomials (given as a list of monomials) is a classical open question posed by von zur Gathen and Kaltofen in [GK85]. An inherent difficulty in tackling the problem lies within the fact that a factor of a sparse polynomial need not be sparse. Example 5.1 in [GK85] demonstrates that a blow-up in the sparsity of a factor can be super-polynomial over any field. Consequently, just writing down the irreducible factors as lists of monomials can take super-polynomial time. In fact, the randomized algorithm of [GK85] assumes that the upper bound on the sparsity of the factors is known. In light of this difficulty, a simpler problem was posed in that same paper: Given $m + 1$ sparse polynomials $f_1, f_2, \dots f_m, g$ test if $g = f_1 \cdot f_2 \cdot \dots \cdot f_m$. This problem is referred to as "testing sparse factorization".

Our main result gives a deterministic factorization algorithm for sparse multiquadratic polynomials.

Theorem 5. *Let $n, s \in \mathbb{N}$ and suppose $\mathrm{char}(\mathbb{F}) \neq 2$. There exists a deterministic algorithm that given an s-sparse multiquadratic polynomial $f(\bar{x}) \in \mathbb{F}[x_1, x_2, \dots, x_n]$ and a square root oracle R_2 uses $\mathrm{poly}(n, s, \log|\mathbb{F}|)$ field operations and oracle calls to R_2 and outputs the irreducible factors of $f(\bar{x})$. That is, a list $h_1, \dots, h_k$ of irreducible polynomials such that $f = h_1 \cdot \dots \cdot h_k$.*

Using techniques from Differential Field Theory we show that some identity testing algorithms could be extended to work beyond an exponentiation gate. In particular, we prove the following theorem which can be seen as testing symmetric sparse factorization. We note that setting $e = 1$ instantiates to testing sparse factorization in the case when $f_1 = f_2 = \dots = f_m$.

Theorem 6. *Let $n, s, d, e, \delta \in \mathbb{N}$ and let $f(\bar{x}), g(\bar{x}) \in \mathbb{F}[x_1, x_2, \dots, x_n]$ be two s-sparse polynomials of degree at most δ. Furthermore, suppose that $\mathrm{char}(\mathbb{F}) = 0$ or $\mathrm{char}(\mathbb{F}) > \delta \cdot \min(e, d)$. Then there exists a deterministic algorithm that given f and h uses $\mathrm{poly}(n, s, d, e, \delta, \log|\mathbb{F}|)$ field operations and tests whether $f^d = g^e$.*

We note that a similar result follows from the works of [ASSS12, BMS13]. We give a more direct and simple algorithm. In addition, our result can handle a slightly better regime of parameters.

1.4 Techniques

Our main technique is to convert an oracle access to a power of a polynomial f^e into an oracle access to the polynomial itself f. As was discussed in the first part of the Introduction, a necessarily condition is having an efficient root extraction algorithm for field elements, referred to as a "root oracle". Yet, as was demonstrated further, applying root oracles naivly can result in inconstensty. More specifically, as there could be e roots of a polynomial, differing only by a multiplicative factor of a root of unity of order e, a root oracle can mismatch the answers to different oracle queries. We solve this problem by introducing an anchor and matching all the queries to that anchor. More specifically, we fix a non-zero assignment $\bar{a}$ of f. For query point $\bar{b}$ we compute the root along the line $\ell_{\bar{a},\bar{b}}(t)$ that passes through $\bar{a}$ and $\bar{b}$. Thus, we reduce the problem from n variables to 1. Finally, we show how to use a root oracle to compute a root of a univariate polynomial. The latter is carried out via Squarefree decomposition. See Sections 2.2 and 3.1 for more details.

In order to deal with sparse multiquadratic polynomials, we first show that a factor of such a polynomial is also sparse. Next, we apply the quadratic formula to get explicit expressions for the factors. Yet, these expression involve square roots. Computing a square root of a polynomial h can be seen as computing $\pm f$ given $h = f^2$. To this end, we first apply our main technique to get an oracle access for f and then use a reconstruction algorithm for sparse polynomials to compute the polynomial. See Section 3.3 for more details.

1.5 Previous Results

Over the last three decades the question of derandomizing sparse polynomial factorization has seen only a very partial progress. In [SV10] Shpilka & Volkovich gave efficient deterministic factorization algorithms for sparse multilinear polynomials. This result was recently extended [Vol14] to the model of sparse polynomials that split into multilinear factors. For the testing version of the problem, Saha et al. [SSS13] presented an efficient deterministic algorithm for the special case when the sparse polynomials are sums of univariate polynomials.

1.6 Organization

We begin by some basic definitions and notation in Section 2 when in Section 2.2 we show how to compute a root of a univariate polynomial and in Section 2.3.1 we prove that a factor of a sparse multiquadratic polynomial is also sparse. In Section 3 we give all our results showing how to preform certain computations on polynomials given an oracle access to their powers. We begin (Section 3.1) by showing how convert an oracle access to f^e into an oracle access to f using an e-th root oracle, thus proving Theorem (Theorem 1) which is our main technical contribution. The first application is given in Section 3.2 where we show how to extend a reconstruction algorithm for a circuit class $\mathcal{C}$ to handle powers of polynomials from $\mathcal{C}$ (Theorem 2). As a corollary, we obtain an efficient reconstruction algorithm for powers of sparse (Theorem 3) and read-once (Theorem 4) polynomials. Our main application is given in Section 3.3 where we present the first efficient factorization algorithm for sparse multiquadratic polynomials, thus proving theorem Theorem 5. In Section 3.4, using different techniques but following the general line, we show how certain polynomial identity testing algorithms can be extended to handle powers of polynomials. We conclude the paper with discussion and open questions in Section 4.

2 Preliminaries

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

2.1 Polynomials

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

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

Definition 2.1 (Line)**.** *Given $\bar{a}, \bar{b} \in \mathbb{F}^n$ we define a* line *passing through $\bar{a}$ and $\bar{b}$ as $\ell_{\bar{a},\bar{b}} : \mathbb{F} \to \mathbb{F}^n$, $\ell_{\bar{a},\bar{b}}(t) \triangleq (1-t)\cdot \bar{a} + t \cdot \bar{b}$. In particular, $\ell_{\bar{a},\bar{b}}(0) = \bar{a}$ and $\ell_{\bar{a},\bar{b}}(1) = \bar{b}$.*

Definition 2.2 (Degrees, Leading Monomials, Leading Coefficients)**.** *The* leading monomial *of a polynomial f, $\mathrm{lm}(f)$ is defined as the largest non-zero monomial of f (with its coefficient) with respect to the lexicographical order of the monomials. The* degree *of f is defined as the degree of $\mathrm{lm}(f)$. Let $x_i \in \mathrm{var}(f)$. We can write: $f = \sum_{j=0}^{d} f_j \cdot x_i^j$ such that $\forall j, x_i \notin \mathrm{var}(f_j)$ and $f_d \not\equiv 0$. The* leading coefficient *of f w.r.t to x_i is defined as $\mathrm{lc}_{x_i}(f) \triangleq f_d$. The* individual degree *of x_i in f is defined as $\deg_{x_i}(f) \triangleq d$.*

It easy to see that for every $f, g \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ and $i \in [n]$ we have that: $\mathrm{lm}(f \cdot g) = \mathrm{lm}(f) \cdot \mathrm{lm}(g)$ and $\mathrm{lc}_{x_i}(f \cdot g) = \mathrm{lc}_{x_i}(f) \cdot \mathrm{lc}_{x_i}(g)$.

2.1.1 Partial Derivatives

The concept of a *partial derivative* of a multivariate function and its properties are well-known and well-studied for continuous domains (such as, $\mathbb{R}$, $\mathbb{C}$ etc.). This concept can be extended to polynomials and rational functions over arbitrary fields from a purely algebraic point of view. For more details we refer to reader to [Kap57].

Definition 2.3. *For a monomial $M = \alpha \cdot x_1^{e_1} \cdots x_i^{e_i} \cdots x_n^{e_n} \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ and a variable x_i we define the* partial derivative *of M with respect to x_i, as $\frac{\partial M}{\partial x_i} \triangleq \alpha e_i \cdot x_1^{e_1} \cdots x_i^{e_i - 1} \cdots x_n^{e_n}$. The definition can be extended to $\mathbb{F}[x_1, x_2, \ldots, x_n]$ by imposing linearity and to $\mathbb{F}(x_1, x_2, \ldots, x_n)$ via the quotient rule.*

Observe that the sum, product, quotient and chain rules carry over. In addition, when $\mathbb{F} = \mathbb{R}$ or $\mathbb{F} = \mathbb{C}$ the definition coincides with the analytical one. The following set of rational function plays an important role.

Definition 2.4 (Field of Constants)**.** *The* Field of Constants *of $\mathbb{F}(x_1, x_2, \ldots, x_n)$ is defined as $\mathrm{C}(\mathbb{F}(x_1, x_2, \ldots, x_n)) \triangleq \left\{ f \in \mathbb{F}(x_1, x_2, \ldots, x_n) \;\middle|\; \forall i \in [n], \frac{\partial f}{\partial x_i} \equiv 0 \right\}$.*

It is easy to see that the field of constants is, indeed, a field and in particular $\mathbb{F} \subseteq \mathrm{C}(\mathbb{F}(x_1, x_2, \ldots, x_n))$. Furthermore, this containment is proper for fields with positive characteristics and equality holds only for fields with characteristic 0. The following Lemma gives a precise characterization of $\mathrm{C}(\mathbb{F}(x_1, x_2, \ldots, x_n))$.

Lemma 2.5. *Let $\mathbb{F}$ be a field of characteristic p. Then for every $n \in \mathbb{N}$:*

1. $\mathrm{C}(\mathbb{F}(x_1, x_2, \ldots, x_n)) = \mathbb{F}$ *when $p = 0$.*
2. $\mathrm{C}(\mathbb{F}(x_1, x_2, \ldots, x_n)) = \mathbb{F}(x_1^p, x_2^p, \ldots, x_n^p)$ *when p is positive.*

2.1.2 Factors and Perfect Powers

Let $f, g \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be polynomials. We say that g *divides* f, or equivalently g is a factor of f, and denote it by $g \mid f$ if there exists a polynomial $h \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ such that $f = g \cdot h$. We say that f is *irreducible* if f is non-constant and cannot be written as a product of two non-constant polynomials. For $e \in \mathbb{N}$, we say that f is a *perfect e-th power* if there exists a polynomial $h \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ such that $f = h^e$. Equivalently, we say that h is f's e-th root.

Given the notion of divisibility we define the gcd of a set of polynomials in the natural way. Given the notion of irreducibility we can state the important property of the uniqueness of factorization,

Lemma 2.6 (Uniqueness of Factorization)**.** *Let $h_1^{e_1} \cdot \ldots \cdot h_k^{e_k} = g_1^{e'_1} \cdot \ldots \cdot g_{k'}^{e'_{k'}}$ be two factorizations of the same non-zero polynomial into irreducible, pairwise comprise factors. Then $k = k'$ and there exists a permutation $\sigma : [k] \to [k]$ such that $h_i \sim g_{\sigma(i)}$ and $e_i = e'_{\sigma(i)}$ for $i \in [k]$.*

By definition, the ratio α/β of two e-th of roots a field element (i.e. $\alpha^e = \beta^e \neq 0$) is a root of unity of order e. We show that the same holds for perfect roots of polynomials. More precisely, two e-th roots of the same polynomial differ only by a multiplicative factor ω satisfying $\omega^e = 1$.

Lemma 2.7. *Let $f(\bar{x}), h(\bar{x}), g(\bar{x}) \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be polynomials such that $f(\bar{x}) = h(\bar{x})^e = g(\bar{x})^e$ for some $e \in \mathbb{N}$. In addition, let $\alpha \in \mathbb{F}, \bar{a} \in \mathbb{F}^n$ such that $\alpha^e = f(\bar{a}) \neq 0$. Then*

1. *There exists $\omega \in \mathbb{F}$ such that $\omega^e = 1$ and $h(\bar{x}) = \omega \cdot g(\bar{x})$.*
2. *There exists a unique polynomial $u(\bar{x}) \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ s.t. $f(\bar{x}) = u(\bar{x})^e$ and $u(\bar{a}) = \alpha$.*

Proof.

1. If $h \equiv 0$ then clearly $g \equiv 0$ and the claim follows. Otherwise, let $h = h_1^{e_1} \cdot \ldots \cdot h_k^{e_k}$ and $g = g_1^{e'_1} \cdot \ldots \cdot g_{k'}^{e'_{k'}}$ be factorizations of h and g into irreducible, pairwise comprise factors, respectively. We have that $h_1^{e_1 \cdot e} \cdot \ldots \cdot h_k^{e_k \cdot e} = h^e = g^e = g_1^{e'_1 \cdot e} \cdot \ldots \cdot g_{k'}^{e'_{k'} \cdot e}$ are two factorizations of the same non-zero polynomial. By Lemma 2.6, $k = k'$ and, wlog $h_i \sim g_i$ and $e_i = e'_i$. Consequently, $h = \omega \cdot g$ for some $\omega \in \mathbb{F}$. Finally, $h^e = \omega^e \cdot g^e = \omega^e \cdot h^e$ and the claim follows.
2. First, note that $h(\bar{a})^e = f(\bar{a}) \neq 0$ and thus $h(\bar{a}) \neq 0$. Let us consider $u(\bar{x}) \overset{\Delta}{=} \frac{\alpha h(\bar{x})}{h(\bar{a})}$. By definition, $u(\bar{a}) = \frac{\alpha h(\bar{a})}{h(\bar{a})} = \alpha$ and $u(\bar{x})^e = \frac{\alpha^e h(\bar{x})^e}{h(\bar{a})^e} = \frac{f(\bar{a}) f(\bar{x})}{f(\bar{a})} = f(\bar{x})$. Now, suppose there exists a polynomial $v(\bar{x}) \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ satisfying the same properties. By the first part of the Lemma we have that $u = \omega \cdot v$ for some $\omega \in \mathbb{F}$. Therefore, $\alpha = u(\bar{a}) = \omega \cdot v(\bar{a}) = \omega \cdot \alpha$ implying that $\omega = 1$. Consequently, $u = v$.

□

2.2 Univariate Polynomials: Squarefree Decomposition and Root Computation

In this section we show how to compute the e-th roots of univariate polynomials using root oracles. We begin by discussing a Squarefree Decomposition of a polynomial. This is one of the steps in the majority of the polynomial factorization algorithms.

Definition 2.8 (Squarefree polynomials)**.** *We say that a polynomial $f(y) \in \mathbb{F}[y]$ is* squarefree *if $g(y)^2 \not| f(y)$ for every $g(y) \in \mathbb{F}[y]$.*

Definition 2.9 (Squarefree Decomposition)**.** *Let $f(y) \in \mathbb{F}[y]$ be polynomial of degree at most d. The* squarefree decomposition *of $f(y)$ is a sequence of pairwise coprime, squarefree polynomials $(g_1, \dots, g_d)$ such that $f = g_1 \cdot g_2^2 \cdot \dots \cdot g_d^d$.*

The next lemma shows that for monic polynomials the squarefree decomposition is unique. Moreover, this decomposition can be computed efficiently.

Lemma 2.10 (Theorem 14.23 of [GG99] and extensions)**.** *Let $f(y) \in \mathbb{F}[y]$ be a non-constant, monic polynomial of degree at most d. Then there exists a unique squarefree decomposition into a sequence of monic polynomials. Moreover, there exists a deterministic algorithm that given the polynomial $f(y)$ uses* $\mathrm{poly}(d, \log|\mathbb{F}|)$ *field operations and computes its squarefree decomposition.*

The squarefree decomposition gives rise to a simple e-th root computation algorithm for univariate polynomials. In addition, this algorithm can be used to test whether a univariate polynomial is indeed a perfect power.

Lemma 2.11. *Let $g(y) \in \mathbb{F}[y]$ be a non-constant, monic polynomial of degree at most d an let $(g_1, \dots, g_d)$ be its squarefree decomposition. Then $g(y) = h(y)^e$ for some $e \in \mathbb{N}$ and $h(y) \in \mathbb{F}[y]$ iff $g_i = 1$ when $e \not| i$.*

Proof. Let $(g_1, \dots, g_d)$ be as above. Consider the polynomial $h \triangleq \prod_{e \mid i} g_i^{i/e}$. We have that:

$$h^e = \prod_{e \mid i} g_i^i = \prod_i g_i^i = g$$

when the last equality follows from the property of g_i and we are done. For the other direction, let $g = h^e$ and let $(h_1, \dots, h_d)$ be the squarefree decomposition of $h(y)$. Consider the following sequence:

$$\hat{g}_i = \begin{cases} h_{i/e} & e \mid i \\ 1 & \text{otherwise} \end{cases}$$

We have that $\prod_i \hat{g}_i^i = \prod_{e \mid i} h_{i/e}^i = \prod_j h_j^{j \cdot e} = \left(\prod_j h_j^j \right)^e = h^e = g$. In addition, $(\hat{g}_1, \dots, \hat{g}_d)$ is a sequence of pairwise coprime, squarefree polynomials. By uniqueness, the sequence $(\hat{g}_1, \dots, \hat{g}_d)$ is squarefree decomposition of g and the claim follows. □

The following is immediate given the previous lemmas.

Corollary 2.12. *There exists a deterministic algorithm that given a non-constant, monic polynomial $f(y) \in \mathbb{F}[y]$ of degree at most d outputs a polynomial $h(y) \in \mathbb{F}[y]$ such that $f(y) = h(y)^e$ if one exists using* $\mathrm{poly}(d, \log|\mathbb{F}|)$ *field operations.*

We can extend the algorithm to handle arbitrary univariate polynomials by making a call to a root oracle.

Lemma 2.13. *There exists a deterministic algorithm that given $e \in \mathbb{N}$, an e-th root oracle R_e and a polynomial $f(y) \in \mathbb{F}[y]$ of degree at most d uses* $\mathrm{poly}(d, \log|\mathbb{F}|)$ *field operations and one oracle call to R_e and computes an e-th root of $f(y)$. That is, the algorithm outputs a polynomial $h(y) \in \mathbb{F}[y]$ such that $f(y) = h(y)^e$ if one exists. Otherwise, the algorithm rejects.*

Proof. If $f(y) = \alpha \in \mathbb{F}$ is a field element (i.e. a constant polynomial), output $R_e(\alpha)$. Otherwise, consider $\hat{f}(y) \triangleq f(y)/\mathrm{lc}(f)$. As $\hat{f}(y)$ is a non-constant, monic polynomial we can apply Corollary 2.12 to compute $\hat{h}(y) \in \mathbb{F}[y]$ such that $\hat{f}(y) = \hat{h}(y)^e$. In addition, let $\alpha = R_e(\mathrm{lc}(f))$. Output $\alpha \cdot \hat{h}(y)$. Observing that $(\alpha \cdot \hat{h}(y))^e = f(y)$ completes the proof. □

2.3 Sparse Polynomials

An *s-sparse polynomial* is polynomial with at most s (non-zero) monomials. We denote by $\|f\|$ the *sparsity* of f. In this section we list several results related to sparse polynomials. We begin with a corollary from [SV10] that shows that a sparse multilinear polynomial can be factored efficiently. Moreover, all its factors are sparse.

Lemma 2.14 ([SV10]). *Given a multilinear polynomial $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$, there is a* $\mathrm{poly}(n, \|f\|)$ *time deterministic algorithm that outputs the* irreducible *factors, $h_1, \ldots, h_k$ of f. Furthermore, $\|h_1\| \cdot \|h_2\| \cdot \ldots \cdot \|h_k\| = \|f\|$.*

The following result gives an efficient reconstruction algorithm for sparse polynomials.

Lemma 2.15 ([KS01]). *Let $n, s, d \in \mathbb{N}$. There exists a deterministic algorithm that given an oracle access to an s-sparse polynomial $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ of degree d uses* $\mathrm{poly}(n, s, d, \log|\mathbb{F}|)$ *field operations and outputs f.*

As a corollary we obtain an efficient algorithm for sparse polynomial division given an upper bound on the sparsity of the quotient polynomial.

Lemma 2.16 ([KS01, DdO14]). *Let $n, s, d, t \in \mathbb{N}$. Let $f, g \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be s-sparse polynomials of degree at most d. Then there exists an algorithm that given f, g uses* $\mathrm{poly}(n, d, s, t, \log|\mathbb{F}|)$ *field operations and computes the quotient polynomial of f and g if it a t-sparse polynomial. That is, if $f = gh$ for some $h \in \mathbb{F}[x_1, x_2, \ldots, x_n]$, $\|h\| \leq t$ then the algorithm outputs h. Otherwise, the algorithm rejects.*

2.3.1 Sparse Multiquadratic Polynomials

In this section we prepare the ground for our main application - efficient factorization algorithm for sparse multiquadratic polynomials. We begin by showing that a factor of a sparse multiquadratic polynomials is also sparse. Recall that in general a sparse polynomial can have a dense factor.

Lemma 2.17. *Let $0 \not\equiv f, g \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be polynomials such that g is multiquadratic. Then $f \mid g \implies \|f\| \leq \|g\|$.*

Proof. The proof is by induction on the number of variables. The base case is when $n = 0$. That is, $f, g \in \mathbb{F}$. Clearly, in this case $\|f\| = \|g\| = 1$ and the claim holds. Now suppose that $n \geq 1$. By definition, $f \cdot h = g$ for some $h \in \mathbb{F}[x_1, x_2, \ldots, x_n]$. We have two cases to consider: Suppose $\mathrm{var}(f) \cap \mathrm{var}(h) = \emptyset$. In this case $\|f\| \cdot \|h\| = \|g\|$ and hence $\|f\| \leq \|g\|$. Otherwise, pick $x_i \in \mathrm{var}(f) \cap \mathrm{var}(h)$. Since g is multiquadratic we can write $f = f_i x_i + f_0$ and $h = h_i x_i + h_0$ such that f_i, h_i, f_0 and h_0 do not depend on x_i. Therefore:

$$\|g\| = \|(f_i x_i + f_0) \cdot (h_i x_i + h_0)\| = \|f_i h_i x_i^2 + (f_0 h_i + f_i h_0) x_i + f_0 h_0\| \geq \|f_i h_i\| + \|f_0 h_0\|$$

By the induction hypothesis $\|f_i h_i\| \geq \|f_i\|$ and $\|f_0 h_0\| \geq \|f_0\|$. Consequently,

$$\|g\| \geq \|f_i h_i\| + \|f_0 h_0\| \geq \|f_i\| + \|f_0\| = \|f\|$$

implying the claim of the lemma. □

It is easy to see that this bound is tight. The following corollary is immediate by combining the bound with Lemma 2.16.

Corollary 2.18. *Let $n, s, d \in \mathbb{N}$. There exists an algorithm that given s-sparse multiquadratic polynomials $f, g \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ uses $\mathrm{poly}(n, s, d, \log|\mathbb{F}|)$ field operations and computes the quotient polynomial of f and g. That is, if $f = gh$ for some $h \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ then the algorithm outputs h. Otherwise, the algorithm rejects.*

We can extend the result to the case when a polynomial is a factor of a product of sparse multiquadratic polynomials. Note that such a product need not be either sparse or multiquadratic.

Corollary 2.19. *Let $0 \not\equiv f, g_1, \ldots, g_k \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be polynomials such that for all $i \in [k]$, g_i is multiquadratic. Then $f \mid g_1 \cdot \ldots \cdot g_k \implies \|f\| \leq \|g_1\| \cdot \ldots \cdot \|g_k\|$.*

Proof. Since $f \mid g_1 \cdot \ldots \cdot g_k$, we can write $f = f_1 \cdot \ldots \cdot f_k$ such that $f_i \mid g_i$. By the Lemma: $\|f_i\| \leq \|g_i\|$. Therefore: $\|f\| \leq \|f_1\| \cdot \ldots \cdot \|f_k\| \leq \|g_1\| \cdot \ldots \cdot \|g_k\|$. □

The following lemma shows that if a sparse multiquadratic polynomial over a field with an odd characteristic factors in a certain way, then the corresponding discriminant is a polynomial and, in fact, a sparse polynomial.

Lemma 2.20. *Suppose $\mathrm{char}(\mathbb{F}) \neq 2$. Let $f = a x_i^2 + b x_i + c \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a multiquadratic polynomial that can be factored as $f = g \cdot h$ when both g and h depend on x_i. Then there exists a multiquadratic polynomial $\Delta \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be such that $\Delta^2 = b^2 - 4ac$. Moreover, $\|\Delta\| \leq \|f\|^2$.*

Proof. Let $g = g_i x_i + g_0$ and $h = h_i x_i + h_0$. By comparing the coefficients of x_i on both sides of the equation we get that $a = g_i h_i$, $b = g_i h_0 + g_0 h_i$ and $c = g_0 h_0$. Therefore,

$$b^2 - 4ac = (g_i h_0 + g_0 h_i)^2 - 4 g_i h_i g_0 h_0 = (g_i h_0 - g_0 h_i)^2.$$

Consequently, selecting $\Delta \overset{\Delta}{=} g_i h_0 - g_0 h_i$ takes care of the first claim. The claim regarding the degree follows from the fact that the degree of every variable in $b^2 - 4ac$ is at most 4. Finally, as $(b + \Delta)(b - \Delta) = 4ac$, by Corollary 2.19: $\|b + \Delta\| \leq \|a\| \cdot \|c\|$, implying that

$$\|\Delta\| \leq \|a\| \cdot \|c\| + \|b\| \leq \|f\|^2.$$

□

3 Computations beyond an Exponentiation Gate and Application

In this section we give all our results showing how preform certain computations on polynomials given an oracle access to their powers.

3.1 Evaluation beyond an Exponentiation Gate

The most basic task for polynomial manipulation is evaluating a polynomial given via an oracle access. In this section we show how to transform an oracle access to the polynomial f^e into an oracle access to f itself. This can be thought of having an oracle equipped with a clever root extraction algorithm. Our main result is given in the following algorithm.

```
Input: Oracle access to a polynomial $f = g^e \in \mathbb{F}[x_1, x_2, \ldots, x_n]$; $\bar{a} \in \mathbb{F}^n$ s.t. $f(\bar{a}) \neq 0$;
$e \in \mathbb{N}$, $e$-th root oracle $R_e$.
Evaluation points $\bar{b}_1, \bar{b}_2, \ldots \in \mathbb{F}[x_1, x_2, \ldots, x_n]$
Output: $h(\bar{b}_1), h(\bar{b}_2), \ldots$ when $h(\bar{x}) \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ is a polynomial s.t. $h^e = f$.

1 $\alpha \leftarrow R_e(f(\bar{a}))$ /* Computed only once.                              */
2 Compute $h_{\bar{b}}(t)$ such that $h_{\bar{b}}(t)^e = f(\ell_{\bar{a},\bar{b}}(t))$ /* Invoking Lemma 2.13     */
3 $\beta \leftarrow h_{\bar{b}}(0)$ ;
4 return $h_{\bar{b}}(1) \cdot \alpha/\beta$
```

Algorithm 1: Polynomial Oracle Transformation

Lemma 3.1. *Let $h(\bar{x}) \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be such that $f(\bar{x}) = h(\bar{x})^e$ and $h(\bar{a}) = \alpha$. Then for every $\bar{b} \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ Algorithm 1 outputs $h(\bar{b})$.*

Proof. First, by Lemma 2.7 such a polynomial $h(\bar{x})$ exists and is unique. In addition, $\beta \neq 0$ since $\beta^e = h_{\bar{b}}(0)^e = f(\ell_{\bar{a},\bar{b}}(0)) = f(\bar{a}) \neq 0$. Therefore, the output of algorithm is well-defined. Next, we have that $h_{\bar{b}}(t)^e = f(\ell_{\bar{a},\bar{b}}(t)) = h(\ell_{\bar{a},\bar{b}}(t))^e$. By Lemma 2.7, $h_{\bar{b}}(t) = \omega \cdot h(\ell_{\bar{a},\bar{b}}(t))$ for some $\omega \in \mathbb{F}$. Therefore:

$$\frac{h_{\bar{b}}(1) \cdot \alpha}{\beta} = \frac{\omega \cdot h(\ell_{\bar{a},\bar{b}}(1)) \cdot \alpha}{h_{\bar{b}}(0)} = \frac{\omega \cdot h(\bar{b}) \cdot \alpha}{\omega \cdot h(\bar{a})} = h(\bar{b}).$$

□

Note that Algorithm 1 requires a non-zero point of $f(\bar{x})$ as an additional input. Generally speaking, finding such a point is the well-known problem of Polynomial Identity Testing (PIT) which is not known to have an efficient deterministic algorithm. We now argue that for our purposes we do not need a PIT algorithm.

Recall that we are in the setting where the root of $f(\bar{x})$ is evaluated on a sequence of points. Given each new query point $\bar{b} \in \mathbb{F}^n$ we can first evaluate $f(\bar{x})$ on $\bar{b}$. If $f(\bar{b}) \neq 0$, we can set $\bar{a} = \bar{b}$ and use this $\bar{a}$ as the non-zero input onwards. Observe that Algorithm 1 works for the case $\bar{a} = \bar{b}$ as well. However, one may ask what happens with the previous query points? Or, what if for all the query points $\bar{b}$ are zeros of f? Observe that if $f(\bar{b}) = 0$ then $h(\bar{b}) = 0$ for any $h(\bar{x}) \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ such that $h(\bar{x})^e = f(\bar{x})$. Therefore, there is no issue of inconsistency here and the oracle just needs to output 0. Consequently, we can patch Algorithm 1 by using the first non-zero query point as $\bar{a}$ (if one exists). Theorem 1 follows as a corollary of Lemma 3.1 and the above discussion.

3.2 Reconstruction beyond an Exponentiation Gate

An immediate application of the polynomial evaluation algorithm is reconstruction beyond an exponentiation gate. More formally, let A be a reconstruction algorithm for a circuit class $\mathcal{C}$. By definition, A requires an oracle access to $f \in \mathcal{C}$ to reconstruct it. We can extend the algorithm to reconstruct $f(\bar{x})$ given an oracle access to $f(\bar{x})^e$ and an e-th root oracle R_e, by simulating each query of A. However, in the spirit of Lemma 2.7 the reconstruction algorithm might end up outputting $\omega \cdot f(\bar{x})$ depending on the root oracle R_e at hand. This reasoning is summarized in Theorem 2. As a corollary we get the following:

Proof of Theorem 3. Apply Theorem 2 with Lemma 2.15. $\square$

Theorem 4 also follows as a corollary given the following result:

Lemma 3.2 ([SV14a]). *Let $n \in \mathbb{N}$. There exists a deterministic algorithm that given an oracle access to a read-once polynomial $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ uses $n^{\mathcal{O}(\log n)} \cdot \mathrm{poly}(\log |\mathbb{F}|)$ field operations and outputs a read-once formula Ψ that computes f.*

3.3 Deterministic Factorization of Sparse Multiquadratic Polynomials

For the case of sparse multiquadratic polynomials we can actually push those techniques further to obtain complete factorization thus proving Theorem 5. We now give the overview of the algorithm.

Suppose $\mathrm{char}(\mathbb{F}) \neq 2$. Let $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a multiquadratic polynomial and let x_i be a variable such that f factors as $f = g \cdot h$ when both g and h depend on x_i. We can view f as $f = ax_i^2 + bx_i + c$ when $a(\bar{x}), b(\bar{x})$ are $c(\bar{x})$ polynomials that do not depend on x_i. Given this view, we can express g and h in terms of a, b and c using the quadratic formula. That is, we can write

$$a \cdot f = (ax_i + b/2 + \Delta/2) \cdot (ax_i + b/2 - \Delta/2)$$

when Δ is a polynomial satisfying $\Delta^2 = b^2 - 4ac$. By Lemma 2.17, both factors are $\|f\|$-sparse so we could continue this process recursively. However, there are some issues with this approach. First, it is not clear that Δ is a polynomial since the expression $b^2 - 4ac$ might not be a perfect square. Next, suppose that Δ were a polynomial. Is it sparse? Answers to these question are given in Lemma 2.20. Finally, how do we compute Δ? For that purpose we apply Theorem 3 that allows us reconstruct a sparse polynomial f given an oracle access to its power f^e. Formally, an instantiation of Theorem 3 with $e = 2, d = 4n, s = \|f\|^2$ together with Lemma 2.20 give rise to the following corollary.

Corollary 3.3. *Suppose $\mathrm{char}(\mathbb{F}) \neq 2$. Let $f = ax_i^2 + bx_i + c \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a multiquadratic polynomial that can be factored as $f = g \cdot h$ when both g and h depend on x_i. Then there exists a deterministic algorithm that given $i \in [n]$, the polynomial $f(\bar{x})$ and a square root oracle R_2 uses $\mathrm{poly}(n, \|f\|, \log |\mathbb{F}|)$ field operations and oracles calls, and outputs a multiquadratic polynomial $\Delta \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ such that $\Delta^2 = b^2 - 4ac$ and $\|\Delta\| \leq \|f\|^2$.*

However, this still does not solve the problem entirely, as we obtain a factorization of $a \cdot f$ instead of f, while a need not be constant. Another issue is that f could factor differently: $f = (a'x_i^2 + b'x_i + c')h$ and in particular the polynomial $a = a' \cdot h$ could be reducible. We solve both problems by changing the way we apply recursion: we first recursively factorize $a(x)$ and then iteratively use Corollary 2.18 to write f as $f = \gcd(f, a) \cdot f'$. To finish the algorithm we need to observe that f' is either irreducible or factors as above. We now move the proof of Theorem 5.

```
Input: A multiquadratic polynomial $f(\bar{x}) \in \mathbb{F}[x_1, x_2, \ldots, x_n]$; A square root oracle $R_2$.
Output: A list $h_1, \ldots, h_k$ of the irreducible factors of $f$. That is, $f = h_1 \cdot \ldots \cdot h_k$.

1  $\hat{f} \leftarrow \mathrm{lc}_{x_n}(f)$ ;
2  if $\hat{f}$ is a constant then $S \leftarrow \emptyset$ else $S \leftarrow \mathsf{Factor}(\hat{f})$;
3  $u \leftarrow f$; $T \leftarrow \emptyset$;
4  foreach $h \in S$ do
5      $v \leftarrow u/h$; /* using the algorithm in Corollary 2.18.  */
6      if $v \neq \perp$ then $u \leftarrow v$ else $S \leftarrow S \setminus \{h\}$; $T \leftarrow T \cup \{h\}$;
7  end
8  if $\deg_{x_n}(u) = 1$ then
9      return $S \cup \{u\}$
10 else
11     Write $u = ax_n^2 + bx_n + c$ ;
12     Compute $\Delta \leftarrow \sqrt{b^2 - 4ac}$; /* using the algorithm in Corollary 3.3.  */
13     $\eta_+ \leftarrow ax_n + b/2 + \Delta/2$; $\eta_- \leftarrow ax_n + b/2 - \Delta/2$;
14     foreach $h \in T$ do
15         $v \leftarrow \eta_+/h$; /* using the algorithm in Corollary 2.18.  */
16         if $v \neq \perp$ then $\eta_+ \leftarrow v$; else $\eta_- \leftarrow \eta_-/h$;
17     end
18     $\gamma \leftarrow \mathrm{lm}(u)/\mathrm{lm}(\eta_+ \cdot \eta_-)$;
19     if $u = \gamma\eta_+ \cdot \eta_-$ then return $S \cup \{\gamma\eta_+, \eta_-\}$ else return $S \cup \{u\}$;
20 end
```

Algorithm 2: Factoring Sparse Multiquadratic Polynomials when $\mathrm{char}(\mathbb{F}) \neq 2$.

Proof of Theorem 5. The outline of the algorithm is given in Algorithm 2. First of all, as $f(\bar{x})$ is given to us as a list of monomials, we can assume wlog that $\mathrm{var}(f) = [n]$ by renaming the variables. The proof is by induction on $m(f) \triangleq |\mathrm{var}(\mathrm{lc}_{x_n}(f))|$.

Running time: Observe that throughout the execution of the algorithm $\|u\|, \|v\| \leq \|f\|$ and $\|\eta_+\|, \|\eta_-\| \leq \|f\|^2$. Initially, the bound holds by the definition of the polynomials. As each update results from a division, the claim regarding the sparsity follows from Lemma 2.17. Therefore, by Corollaries 2.18 and 3.3 we get that the total number of field operations and oracle calls to R_2 satisfies the following recurrent expression:

$$t(m, \|f\|) \leq t(m-1, \|f\|) + \mathrm{poly}(m, \|f\|, \log|\mathbb{F}|)$$

resulting in $t(m, \|f\|) = \mathrm{poly}(m, \|f\|, \log|\mathbb{F}|)$. As $m \leq n-1$, the claim regarding the running time follows.

Analysis: Suppose that $m(f) \geq 1$. We need to fix some notations. Let $f = h_1 \cdot \ldots \cdot h_k$ be a factorization of f into irreducible factors. Let g denote the product of those h_i-s that depend on x_n. Note that there can be at most two such factors. Therefore, we can write: $f = h_1 \cdot \ldots \cdot h_{k'} \cdot g$. Finally, let $\hat{g} = \mathrm{lc}_{x_n}(g)$ and let $\hat{g} = \hat{g}_1 \cdot \ldots \cdot \hat{g}_\ell$ be a factorization of $\hat{g}$ into irreducible factors. Note that $\gcd(g, \hat{g}) = 1$ since $x_n \notin \mathrm{var}(\hat{g})$ and g contains only the factors the depend on x_n. Moreover, given the above we get that:

$$\hat{f} = h_1 \cdot \ldots \cdot h_{k'} \cdot \hat{g} = h_1 \cdot \ldots \cdot h_{k'} \cdot \hat{g}_1 \cdot \ldots \cdot \hat{g}_\ell$$

is a factorization of $\hat{f}$ into irreducible factors. As $m(\hat{f}) < m(f)$, by the induction hypothesis the set S will contain the irreducible factors of $\hat{f}$. By the uniqueness of factorization, S will contain exactly the polynomials $\alpha_1 h_1, \ldots, \alpha_{k'} h_{k'}$ and $\beta_1 \hat{g}_1, \ldots, \beta_\ell \hat{g}_\ell$ for some $\{\alpha_i\}, \{\beta_j\} \subseteq \mathbb{F} \setminus \{0\}$. Consequently, the '**for each**' loop separates the h_i-s from $\hat{g}_j$-s by gradually dividing f by the containment of S. Observe, that at the end of the loop we get that: $S = \{\alpha_1 h_1, \ldots, \alpha_{k'} h_{k'}\}$, $T = \{\beta_1 \hat{g}_1, \ldots, \beta_\ell \hat{g}_\ell\}$. Moreover, as $u = f = h_1 \cdot \ldots \cdot h_{k'} \cdot g$ at the beginning of the loop and $\gcd(g, \hat{g}_j) = 1$ for every j, we get that

$$u = \frac{f}{\alpha_1 h_1 \cdot \ldots \cdot \alpha_{k'} h_{k'}} = \frac{g}{\gamma}$$

For some $\gamma \in \mathbb{F}$. Therefore, to complete the algorithm we need to compute the irreducible factors of u and concatenate them with S. Recall that by definition g (and hence u) is a product of at most two irreducible polynomials, both depending on x_n.

If $\deg_{x_n}(u) = \deg_{x_n}(g) = 1$ then u must be a single irreducible factor and thus $f = \alpha_1 h_1 \cdot \ldots \cdot \alpha_{k'} h_{k'} \cdot u$ is a factorization of f into irreducible factors. Otherwise, $\deg_{x_n}(u) = \deg_{x_n}(g) = 2$ and there can be two cases. If u is irreducible, then again $f = \alpha_1 h_1 \cdot \ldots \cdot \alpha_{k'} h_{k'} \cdot u$ is a factorization of f into irreducible factors and the algorithm will return this factorization since for every η_- and η_+ the identity test $u =^? \gamma\eta_+ \cdot \eta_-$ will fail. Otherwise, we can write u as a product of two irreducible polynomials, both depending on x_n. By Corollary 3.3 the discriminant polynomial Δ in Line 12 is computed successfully. As $\gamma u = g$ we have that $\hat{g} = \gamma a$. Consequently, we can write

$$u \cdot \hat{g}_1 \cdot \ldots \cdot \hat{g}_\ell = u \cdot \hat{g} = u \cdot \gamma a = \gamma\eta_+ \cdot \eta_-.$$

As each $\hat{g}_i$ is an irreducible polynomial, it must be the case that either $\hat{g}_i \mid \eta_+$ or $\hat{g}_i \mid \eta_-$. Thus, at Line 17 we have that $u = \gamma\eta_+ \cdot \eta_-$. We can easily compute γ by noting that

$$\text{lm}(u) = \text{lm}(\gamma\eta_+ \cdot \eta_-) = \gamma \text{lm}(\eta_+ \cdot \eta_-).$$

In conclusion, $f = \alpha_1 h_1 \cdot \ldots \cdot \alpha_{k'} h_{k'} \cdot \gamma\eta_+ \cdot \eta_-$ is a factorization of f into irreducible factors and the algorithm will return this factorization passing the identity test $u =^? \gamma\eta_+ \cdot \eta_-$.

The analysis of the base case $m(f) = 0$ is similar. First, note that if $u = f$ is irreducible then the algorithm will return $\{u\}$. Otherwise, we can write u as a product of two irreducible polynomials, both depending on x_n. By definition, $a \cdot u = \eta_+ \cdot \eta_-$. As $a \neq 0 \in \mathbb{F}$,

$$\gamma = \frac{\text{lm}(u)}{\text{lm}(\eta_+ \cdot \eta_-)} = \frac{\text{lm}(u)}{\text{lm}(a \cdot u)} = \frac{1}{a}$$

and hence

$$u = \frac{1}{a}\eta_+ \cdot \eta_- = \gamma\eta_+ \cdot \eta_-.$$

In conclusion we get that in the base case, $f = \gamma\eta_+ \cdot \eta_-$ is a factorization of f into irreducible factors and the algorithm will return this factorization passing the identity test $u =^? \gamma\eta_+ \cdot \eta_-$. This completes the proof. □

3.4 Polynomial Identity Testing beyond an Exponentiation Gate

Using techniques from Differential Field Theory we show how to transform an identity test of powers of polynomials into an identity test that involves partial derivatives of those same polynomials. This transformation can be applied for classes of polynomials that are closed under partial derivatives such as sparse polynomials.

Lemma 3.4. *Let $f(\bar{x}), h(\bar{x}) \not\equiv 0 \in \mathbb{F}(x_1, x_2, \ldots, x_n)$ and let $e, d \in \mathbb{N}$. There exists $c(\bar{x}) \in \mathbb{F}(x_1, x_2, \ldots, x_n)$ such that $f(\bar{x})^d = c(\bar{x}) \cdot h(\bar{x})^e$ and $\frac{\partial c}{\partial x_i} \equiv 0$ and iff $d \cdot h \cdot \frac{\partial f}{\partial x_i} = e \cdot f \cdot \frac{\partial h}{\partial x_i}$.*

Proof. ($\Rightarrow$) Suppose $f(\bar{x})^d = c(\bar{x}) \cdot h(\bar{x})^e$. Then

$$d \cdot h \cdot \frac{\partial f}{\partial x_i} = \frac{h}{f^{d-1}} \cdot \frac{\partial (f^d)}{\partial x_i} = \frac{h}{f^{d-1}} \cdot c(\bar{x}) \cdot e \cdot \frac{\partial h}{\partial x_i} \cdot h(\bar{x})^{e-1} = e \cdot c(\bar{x}) \cdot \frac{h(\bar{x})^e}{f^{d-1}} \cdot \frac{\partial h}{\partial x_i} = ef \cdot \frac{\partial h}{\partial x_i}.$$

($\Leftarrow$) Consider $c \triangleq \frac{f^d}{h^e}$. By definition:

$$\frac{\partial c}{\partial x_i} = \frac{1}{h^{2e}} \cdot \left(d \cdot \frac{\partial f}{\partial x_i} \cdot f^{d-1} \cdot h^e - e \cdot \frac{\partial h}{\partial x_i} \cdot h^{e-1} \cdot f^d \right) = \frac{f^{d-1}}{h^{e+1}} \cdot \left(d \cdot \frac{\partial f}{\partial x_i} \cdot h - e \cdot \frac{\partial h}{\partial x_i} \cdot f \right) \equiv 0$$

and the claim follows. □

The following theorem provides an algorithm for an identity testing of powers of polynomials over fields with zero or large enough characteristics.

Theorem 3.5. *Let $f(\bar{x}), h(\bar{x}) \not\equiv 0 \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be polynomials of degree at most δ and let $e, d \in \mathbb{N}$. Furthermore, suppose that $p \triangleq \text{char}(\mathbb{F}) = 0$ or $p > \delta \cdot \min(e, d)$. Then $f(\bar{x})^d = h(\bar{x})^e$ iff $\text{lm}(f)^d = \text{lm}(h)^e$ and for each $i \in [n]$ we have that $d \cdot h \cdot \frac{\partial f}{\partial x_i} = e \cdot f \cdot \frac{\partial h}{\partial x_i}$.*

Proof. ($\Rightarrow$) Follows from Lemma 3.4 and the definition of lm.
($\Leftarrow$) By iterative application of Lemma 3.4 we get that there exists $c(\bar{x}) \in \mathrm{C}(\mathbb{F}(x_1, x_2, \ldots, x_n))$ such that $f(\bar{x})^d = c(\bar{x}) \cdot h(\bar{x})^e$. We claim that $c(\bar{x}) \in \mathbb{F}$. Assume the contrary. Then, by Lemma 2.5 $p > 0$ and there exist $u(\bar{x}), v(\bar{x}) \in \mathbb{F}[x_1^p, x_2^p, \ldots, x_n^p]$ such that $\gcd(u, v) = 1$ and $c(\bar{x}) = \frac{u(\bar{x})}{v(\bar{x})}$. Therefore, we can write:

$$f(\bar{x})^d \cdot v(\bar{x}) = h(\bar{x})^e \cdot u(\bar{x}).$$

By definition

$$\mathrm{lm}(f)^d \cdot \mathrm{lm}(v) = \mathrm{lm}(h)^e \cdot \mathrm{lm}(u)$$

which implies that $\mathrm{lm}(v) = \mathrm{lm}(u)$. In particular, $v(\bar{x}), u(\bar{x}) \notin \mathbb{F}$ as $c(\bar{x}) \notin \mathbb{F}$ and thus $\deg(u), \deg(v) \geq p$. Assume wlog that $d \leq e$. Then $p > \delta d$. As $\gcd(u, v) = 1$ we get that $u^p \mid f^d$ which implies that $p \leq \delta d$ thus leading to a contradiction. Therefore, $c(\bar{x}) = \alpha \in \mathbb{F}$. By definition $\mathrm{lm}(f)^d = \alpha \cdot \mathrm{lm}(h)^e$, which implies that $\alpha = 1$ and we are done. $\square$

Theorem 6 follows an easy corollary by noting that the pre-conditions of Theorem 3.5 can be efficiently checked given two sparse polynomials.

4 Discussion & Open Questions

In this paper we study computations beyond a (single) exponentiation gate and present some applications, with the main one being the first efficient deterministic factorization algorithm for sparse multiquadratic polynomials over odd characteristics. Can we devise such algorithms for multicubic polynomials? Or more generally, when the individual degree of each variable is constant? One of the milestones on the route to this goal has to do with estimating the sparsity of the factors of such polynomials. To this end, we propose the following conjecture:

Conjecture 4.1. *There exists a function $\nu : \mathbb{N} \to \mathbb{N}$ such that if $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ is a polynomial with individual degrees at most d then $g \mid f \implies \|g\| \leq \|f\|^{\nu(d)}$.*

Our results show that $\nu(1) = \nu(2) = 2$. As we noted before, the value of $\nu(3)$ is unknown. We also note that the conjecture gives rise to an efficient deterministic algorithm for testing sparse factorization into polynomials with constant individual degrees.

Another milestone in sparse polynomial factorization is computing a root of a sparse polynomial. Theorem 6 allows us to test whether the polynomial f is an e-th root of the polynomial g. But can we actually compute f given g? Once again, an upper bound on the corresponding sparsity could be useful. We can get the desired result by combining this bound with Theorem 3. We propose the following conjecture:

Conjecture 4.2. *Suppose* $\mathrm{char}(\mathbb{F}) = 0$ *or "large enough". Let $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a polynomial. Then for every $e \in \mathbb{N}$: $\|f\| \leq \|f^e\|$.*

Example 6.1 in [Vol14] shows that when the field characteristic is close to the degree of the polynomial in question, even a square root of sparse polynomial could be very dense. Therefore, the bound could only hold for "large enough" (in terms of n, d etc..) characterstic. Finally, can we extend Theorem 6 to fields with "small" charactersitcs? Perhaps, by extending Lemma 3.4?

References

[AHK93] D. Angluin, L. Hellerstein, and M. Karpinski. Learning read-once formulas with queries. *J. ACM*, 40(1):185–210, 1993.

[ASSS12] M. Agrawal, C. Saha, R. Saptharishi, and N. Saxena. Jacobian hits circuits: Hitting-sets, lower bounds for depth-d occur-k formulas & depth-3 transcendence degree-k circuits. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC)*, pages 599–614, 2012.

[BB98] D. Bshouty and N. H. Bshouty. On interpolating arithmetic read-once formulas with exponentiation. *JCSS*, 56(1):112–124, 1998.

[BC98] N. H. Bshouty and R. Cleve. Interpolating arithmetic read-once formulas in parallel. *SIAM J. on Computing*, 27(2):401–413, 1998.

[BHH95] N. H. Bshouty, T. R. Hancock, and L. Hellerstein. Learning boolean read-once formulas with arbitrary symmetric and constant fan-in gates. *JCSS*, 50:521–542, 1995.

[BMS13] M. Beecken, J. Mittmann, and N. Saxena. Algebraic independence and blackbox identity testing. *Information & Computation*, 222:2–19, 2013.

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

[DdO14] Z. Dvir and R. Mendes de Oliveira. Factors of sparse polynomials are sparse. *CoRR*, abs/1404.4834, 2014.

[Gat06] J. von zur Gathen. Who was who in polynomial factorization:. In *ISSAC*, page 2, 2006.

[GG99] J. von zur Gathen and J. Gerhard. *Modern computer algebra*. Cambridge University Press, 1999.

[GK85] J. von zur Gathen and E. Kaltofen. Factoring sparse multivariate polynomials. *Journal of Computer and System Sciences*, 31(2):265–287, 1985.

[GKL04] S. Gao, E. Kaltofen, and A. G. B. Lauder. Deterministic distinct-degree factorization of polynomials over finite fields. *J. Symb. Comput.*, 38(6):1461–1470, 2004.

[GS99] V. Guruswami and M. Sudan. Improved decoding of reed-solomon codes and algebraic-geometry codes. *IEEE Transactions on Information Theory*, 45(6):1757–1767, 1999.

[HH91] T. R. Hancock and L. Hellerstein. Learning read-once formulas over fields and extended bases. In *Proceedings of the 4th Annual Workshop on Computational Learning Theory (COLT)*, pages 326–336, 1991.

[Kal85] E. Kaltofen. Polynomial-time reductions from multivariate to bi- and univariate integral polynomial factorization. *SIAM J. on computing*, 14(2):469–489, 1985.

[Kal89] E. Kaltofen. Factorization of polynomials given by straight-line programs. In S. Micali, editor, *Randomness in Computation*, volume 5 of *Advances in Computing Research*, pages 375–412. 1989.

[Kal03] E. Kaltofen. Polynomial factorization: a success story. In *ISSAC*, pages 3–4, 2003.

[Kap57] I. Kaplansky. *An Introduction to Differential Algebra*. Hermann, Paris, 1957.

[Kay07] N. Kayal. *Derandomizing some number-theoretic and algebraic algorithms*. PhD thesis, Indian Institute of Technology, Kanpur, India, 2007.

[Kay12] N. Kayal. An exponential lower bound for the sum of powers of bounded degree polynomials. *Electronic Colloquium on Computational Complexity (ECCC)*, 19:81, 2012.

[KI04] V. Kabanets and R. Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. *Computational Complexity*, 13(1-2):1–46, 2004.

[KLN$^+$93] M. Karchmer, N. Linial, I. Newman, M. E. Saks, and A. Wigderson. Combinatorial characterization of read-once formulae. *Discrete Mathematics*, 114(1-3):275–282, 1993.

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

[KSS14] S. Kopparty, S. Saraf, and A. Shpilka. Equivalence of polynomial identity testing and deterministic multivariate polynomial factorization. In *Proceedings of the 29th Annual IEEE Conference on Computational Complexity (CCC)*, pages 169–180, 2014.

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

[LLL82] A.K. Lenstra, H.W. Lenstr, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen,*, 261(4):515–534, 1982.

[LV03] R. J. Lipton and N. K. Vishnoi. Deterministic identity testing for multivariate polynomials. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 756–760, 2003.

[Sho91] V. Shoup. A fast deterministic algorithm for factoring polynomials over finite fields of small characteristic. In *ISSAC*, pages 14–21, 1991.

[SSS13] C. Saha, R. Saptharishi, and N. Saxena. A case of depth-3 identity testing, sparse factorization and duality. *Computational Complexity*, 22(1):39–69, 2013.

[Sud97] M. Sudan. Decoding of reed solomon codes beyond the error-correction bound. *Journal of Complexity*, 13(1):180–193, 1997.

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

[SV14a] A. Shpilka and I. Volkovich. On reconstruction and testing of read-once formulas. *Theory of Computing*, 10:465–514, 2014.

[SV14b] A. Shpilka and I. Volkovich. Read-once polynomial identity testing. *Computational Complexity*, 2014. (accepted).

[Val80] L. G. Valiant. Negation can be exponentially powerful. *Theoretical Computer Science*, 12(3):303–314, nov 1980.

[Vol14] I. Volkovich. Deterministically factoring sparse polynomials into multilinear factors. *Electronic Colloquium on Computational Complexity (ECCC)*, 21:168, 2014.

[Vol15] I. Volkovich. Characterizing arithmetic read-once formulae. *ACM Transactions on Computation Theory*, 2015. (accepted).